THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Learning with Geometric Embeddings of Graphs

FREDRIK D. JOHANSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2016

Learning with Geometric Embeddings of Graphs
FREDRIK D. JOHANSSON

ISBN 978-91-7597-491-0

Doktorsavhandlingar vid Chalmers Tekniska Högskola
Ny serie nr. 4172
ISSN 0346-718X
Technical Report No. 137D
Department of Computer Science and Engineering
Machine Learning Research Group, Division for Computing Science

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

Cover:
Geometric embedding of a five node graph.

Chalmers Reproservice
Göteborg, Sweden 2016

Till Anna

Abstract

Graphs are natural representations of problems and data in many fields. For example, in computational biology, interaction networks model the functional relationships between genes in living organisms; in the social sciences, graphs are used to represent friendships and business relations among people; in chemoinformatics, graphs represent atoms and molecular bonds. Fields like these are often rich in data, to the extent that manual analysis is not feasible and machine learning algorithms are necessary to exploit the wealth of available information. Unfortunately, in machine learning research, there is a huge bias in favor of algorithms operating only on continuous vector valued data, algorithms that are not suitable for the combinatorial structure of graphs. In this thesis, we show how to leverage both the expressive power of graphs and the strength of established machine learning tools by introducing methods that combine geometric embeddings of graphs with standard learning algorithms. We demonstrate the generality of this idea by developing embedding algorithms for both simple and weighted graphs and applying them in both supervised and unsupervised learning problems such as classification and clustering. Our results provide both theoretical support for the usefulness of graph embeddings in machine learning and empirical evidence showing that this framework is often more flexible and better performing than competing machine learning algorithms for graphs.

Acknowledgements

When I finished my undergraduate education, I had no intention to pursue an academic career, but a little encouragement from my advisor Devdatt Dubhashi made me apply for this position. I am very happy that I did. My girlfriend Anna has been a huge support from the moment I applied, and even though I had little hope of getting the job, she was convinced otherwise. I would like to thank all of my friends and family for helping me through these years.

Along the way I have met some incredible people and I am sorry that I can not list the names of everyone with whom I have shared this experience. It's been a pleasure working and playing squash with my friends and officemates Olof Mogren and Mikael Kågebäck. I had so much fun working with Vinay Jethava, who helped guide me during my first experiences with research. Devdatt has been a huge support and inspiration, always aiming a little bit higher than I would dare myself. I've learned a lot from our collaboration, and I hope it will continue.

My thanks to Staffan Truvé and Svetoslav Marinov who gave me my first insights into research in industry, and to Chiranjib Bhattacharyya who was my first international collaborator. I'm also very happy to have had Peter Dybjer, David Sands, Gerardo Schneider and Henk Wymeersch around for support. Peter Damaschke, it was your course that got me into machine learning in the first place. Thank you Christos and Aristide, Prasanth, Richard and Luis, Alexander, Joel, Nina, Azam, Chien-Chung, Jacob, Ankani and Leonid for being part of my work life. It was great having you all around. Ana, Jonna and Eva, you helped me so much with the little things that mean a lot.

Finally, I would like to acknowledge the Swedish Foundation for Strategic Research (SSF) for funding my employment as part of the research project Data-driven Secure Business Intelligence. During my graduate education I also spent four months at New York University working with Uri Shalit and David Sontag, and at Columbia University with Tony Jebara. These visits were funded in part by the Sweden-America Foundation.

List of publications

This thesis is based on the following manuscripts.

Paper I F. D. Johansson, A. Chattoraj, C. Bhattacharyya, and D. Dubhashi (2015). "Weighted Theta Functions and Embeddings with Applications to Max-Cut, Clustering and Summarization". *Advances in Neural Information Processing Systems (NIPS)*, pp. 1018–1026

Paper II F. D. Johansson and D. Dubhashi (2015). "Learning with Similarity Functions on Graphs using Matchings of Geometric Embeddings". *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 467–476

Paper III F. D. Johansson, V. Jethava, D. Dubhashi, and C. Bhattacharyya (2014). "Global graph kernels using geometric embeddings". *Proceedings of the 31st International Conference on Machine Learning (ICML)*. ed. by T. Jebara and E. P. Xing. JMLR Workshop and Conference Proceedings, pp. 694–702

Paper IV L. Hermansson, T. Kerola, F. Johansson, V. Jethava, and D. Dubhashi (2013). "Entity disambiguation in anonymized graphs using graph kernels". *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. ACM, pp. 1037–1046

The following manuscripts have been published, but are not included in this thesis.

Paper V F. D. Johansson, U. Shalit, and D. Sontag (2016). "Learning Representations for Counterfactual Inference". *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. JMLR Workshop and Conference Proceedings

Paper VI L. Hermansson, F. D. Johansson, and O. Watanabe (2015). "Generalized Shortest Path Kernel on Graphs". *Discovery Science*. Springer International Publishing, pp. 78–85

Paper VII N. Tahmasebi et al. (2015). Visions and open challenges for a knowledge-based culturomics. *International Journal on Digital Libraries* **15**.2-4, 169–187

Paper VIII M. Kågebäck, F. Johansson, R. Johansson, and D. Dubhashi (2015). Neural context embeddings for automatic discovery of word senses. *Proceedings of NAACL-HLT*, 25–32

Paper IX F. D. Johansson, O. Frost, C. Retzner, and D. Dubhashi (2015). "Classifying Large Graphs with Differential Privacy". *Modeling Decisions for Artificial Intelligence*. Springer, pp. 3–17

Paper X F. Axelsson, B. Rydback, F. Johansson, J. Bengtsson, and S. Marinov (2014). "Data-driven Coreference Resolution for Swedish". *Proceedings of the Swedish Language Technology Conference (SLTC). 2014.*

Paper XI F. Johansson, V. Jethava, and D. Dubhashi (2013). "DLOREAN: Dynamic LOcation-aware REconstruction of multiwAy Networks". *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*. IEEE, pp. 1012–1019

Paper XII F. Johansson, T. Färdig, V. Jethava, and S. Marinov (2012). "Intent-aware temporal query modeling for keyword suggestion." *PIKM*. ed. by A. S. Varde and F. M. Suchanek. ACM, pp. 83–86

Contribution summary

Paper I	Co-designed the study, performed the empirical work and analysis and wrote the manuscript.
Paper II	Co-designed the study, contributed to the analysis, performed the empirical work, and wrote most of the manuscript.
Paper III	Co-designed the study, performed the empirical work and most of the analysis and wrote most of the manuscript.
Paper IV	Co-designed the study, contributed to the empirical work and the analysis and wrote most of the manuscript.

Contents

Part I
Extended summary

Chapter 1

Introduction

Recent years have seen a dramatic increase in the collection and representation of data in the form of graphs. Examples include social networks (Wasserman and Faust, 1994), made up of people and their relationships, and gene interaction networks representing the functional interactions between genes in living organisms (Xenarios et al., 2002). At the same time, the need for tools to automatically sift through and analyze graph data has grown rapidly. Machine learning offers tools of exactly this kind, and many labor-intensive tasks such as labelling or categorizing graph data can now be alleviated, if not solved, using machine learning methods (S. Vishwanathan, Schraudolph, Kondor, and K. M. Borgwardt, 2010). Problems of this nature arise in diverse fields, ranging from chemoinformatics and bioinformatics to social sciences, where graphs are well suited to represent information such as molecular compounds or friend networks. For example, in drug development, some candidate compounds will be harmful to humans while some will not. Predicting which compounds are beneficial based on molecular structure, rather than through in-vivo studies, can lead to large savings (Debnath et al., 1991). Another area rich in graph data is social network analysis (Wasserman and Faust, 1994). In social networks, every node represents a person and every edge a relationship or an interaction between two persons. Discovering communities or determining influencers in such networks are important problems for marketing firms that can target campaigns towards groups of people, or individuals who are likely to influence others (Fortunato, 2010).

A key issue in applications of almost any machine learning algorithm is representation of data. In order to truly harness the power of such methods when dealing with graphs, the graphs must be represented in a way that is efficient and informative to the algorithm in question. Most algorithms for important learning problems such as classification, clustering and dimensionality reduction, are designed for data represented by vectors of real values, $\mathbf{x} \in \mathbb{R}^d$. Think for example of linear regression where the predictive model is defined by the inner product $\mathbf{w}^\top \mathbf{x}$ of a parameter vector $\mathbf{w}$ and the data vector $\mathbf{x}$. Graphs, defined as sets of nodes and edges, are combinatorial in nature and do not fit immediately into this paradigm, and while for example the adjacency matrix offers a numeric representation, it is very different from the kind of tabular data often associated with classical machine learning applications. Overcoming this hurdle is an

important challenge for several reasons: 1) Making established machine learning methods applicable to graphs means that any further improvements to these methods benefit graph applications as well. 2) Algorithms operating directly on the combinatorial representation of graphs are often computationally expensive and not feasible to use with large datasets.

This thesis addresses the issue of representing graphs in ways that make them accessible to general purpose machine learning algorithms. So-called geometric embeddings are introduced and combined with both existing and new learning algorithms to form competitive methods for tasks like classification and clustering of graphs. Specifically, the Lovász number is described and used to introduce the novel Lovász graph kernel for simple graphs. This is later generalized for weighted graphs, and an approximation algorithm is developed to enable fast computation. Each method is motivated through both theoretical and empirical results, several of which represent state-of-the-art results.

Thesis outline. This thesis is an extended summary of four published papers, all of which are included in the appendix. In Paper I we generalize Lovász's famous geometric graph embedding to accommodate graphs with weights on nodes and edges. In Paper III we use the original (unweighted) embedding to define graph kernels, designed to capture global properties of graphs. In Paper II we employ these ideas to develop a general framework for classifying graphs using matchings of geometric embeddings of graphs. Paper IV applies graph kernels to the problem of relational entity disambiguation.

In the remainder of chapter 1 we give definitions for concepts and problems that form the basis of this thesis. Chapter 2 introduces geometric embeddings of graphs, and in particular, the generalized (weighted) Lovász embedding developed in Paper I. In chapter 3, we give several methods for learning with graphs using geometric embeddings, based on the results of papers I–IV. Chapter 4 presents the application of graph classification to entity disambiguation developed in Paper IV.

1.1 Graphs

The contents of this thesis revolve completely around *graphs*, structures used to represent relations among a set of entities. A graph $G = (V, E)$ comprises a set $V = \{v_1, \ldots, v_n\}$ of *nodes* or *vertices* and a set $E \subseteq V \times V$ of pairs of nodes called *edges*. Nodes typically represent objects like people or atoms, and edges represent connections like friendship or molecular bonds. Nodes are often denoted simply by their index, i, and the edge from v_i to v_j by the tuple (i, j). The characteristics of the edge set E is often referred to as the graph *structure* or *connectivity*. It is common to represent the edge set by the *adjacency matrix* A, with elements a_{ij} such that $a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ otherwise. Unless otherwise stated, we use the notation $n = |V|$ and $m = |E|$ to denote the number of nodes and edges respectively.

If it holds that $(i, j) \in E \Rightarrow (j, i) \in E$, for any edge $e = (i, j)$, we call the graph *undirected*. If this does not hold, the graph is *directed*. If it is clear from context that a graph is undirected, the edge set is more efficiently represented as a set of *unordered* pairs. We use a small social network as an example of an undirected graph. Three people, *Anna*, *Bob* and *Michael* go to the same school. Anna knows Bob and Bob knows Michael, but

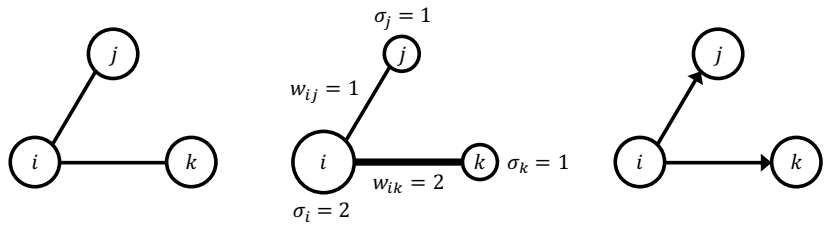


Figure 1.1.1: *Examples of graphs. A simple graph (left), a weighted graph with edge weights w_{ij} and node weights σ_i (middle) and a directed graph (right).*

Anna and Michael do not know each other. We may represent this as the undirected graph $G = (V, E)$ with $V = \{\text{Anna}, \text{Bob}, \text{Michael}\}$ and $E = \{(\text{Anna}, \text{Bob}), (\text{Bob}, \text{Michael})\}$. See figure 1.1.1 for examples of various types of graphs, and an abstract representation of the friends graph (left).

Graphs may be associated with a node labelling function $\sigma : V \longrightarrow \mathcal{L}$, assigning a label to each node from a set of labels $\mathcal{L}$. The set $\mathcal{L}$ may be ordered, e.g. to represent the importance of a node, or unordered when representing object categories. In the former case, σ is often referred to as the *node weight*. Furthermore, the graph may have an edge labelling function $w : E \longrightarrow \mathcal{L}_e$. Here, we limit ourselves to the case where $\mathcal{L}_e = \mathbb{R}$ and refer to w as the *edge weight*. Node labels and edge weights for a fixed graph $G = (V, E)$ are often represented respectively by the vector $\Sigma = [\sigma(v_1), \ldots, \sigma(v_n)]^\top$ and matrix W with elements $w_{ij} = \mathcal{L}_e((i, j))$. In general, we denote a labeled graph by $G = (V, E, \Sigma, W)$. If all nodes share the same label, i.e. σ is a constant function, we call the graph *unlabeled*. If all weights are equal, we call the graph *unweighted*. An undirected, unlabeled and unweighted graph is called *simple*. The complement $\bar{G}$ of a graph G is the graph where two nodes are adjacent if and only if they are *not* adjacent in G. For a more comprehensive exposition of graphs, see for example West et al. (2001).

1.2 Learning problems on graphs

This section introduces two common learning problems on graphs that are addressed in the later chapters of this thesis. We will see why special consideration is needed to adapt machine learning methods to graph data.

Graph classification

Classification is one of the problems most frequently addressed within machine learning. Here, we assume that the reader is already somewhat familiar with the classification problem and commonly used algorithms. In supervised classification problems, objects $x_1, \ldots, x_N \in \mathcal{X}$ are observed, together with labels $y(x) \in \mathcal{Y}$, and the goal is to predict the label $y(x')$ of a previously unseen object x'. Said differently, the goal is to learn the mapping from $\mathcal{X}$ to $\mathcal{Y}$ that gave rise to the data. In many applications, y is binary,

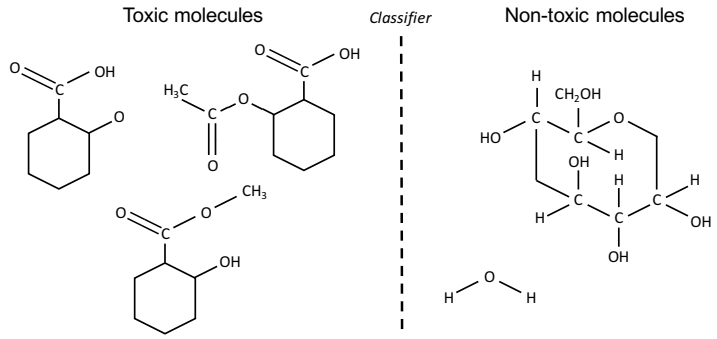


Figure 1.2.1: *An example graph classification problem - predicting the toxicity of molecules.*

$\mathcal{Y} = \{-1, +1\}$, typically representing the logical truth value of some property, e.g. whether an image depicts a cat or not.

When each object to classify is a graph, the problem is called *graph classification*. Applications of this nature arise in for example chemoinformatics, where the goal is to classify a molecular compound as toxic or harmless (Helma, King, Kramer, and Srinivasan, 2001). Here, nodes represent atoms and edges the molecular bonds, see figure 1.2.1. Graph classification of this kind stands in contrast to the setting where the objective is to classify each node of a single graph, a problem which is treated in 1.2. Formally, given a *training set* comprising pairs $\{(G^{(i)}, y_i)\}_{i=1}^{N}$ of graphs $G^{(i)}$ and class labels $y_i \in \mathcal{Y}$, we define graph classification to be the task of assigning labels to a new, previously unseen *test set* of graphs $\{G^{(N+j)}\}_{j=1}^{N_{test}}$.

Classification is a general problem associated with both rich theory and history in machine learning. One of the most widely used algorithms for classification is *support vector machines* (SVM) (Vapnik, 1995), a famous example of *kernel methods* (Schölkopf and Smola, 2001). SVMs belong to the class of *large-margin* methods which attempt not only to fit the data well but to do so with a large margin of error, in order to generalize well to new data. This means that training data should be correctly classified with a separator that is as far away from both classes of points as possible, see figure 1.2.2. The idea of large-margin methods is not only intuitive, but famously leads to strong upper bounds on the generalization error (Vapnik, 1995). With $(x_1, y_1), ..., (x_N, y_N)$ a set of labeled points, the hard-margin linear SVM solves the following optimization problem.

$$\underset{\mathbf{w}}{\text{minimize}} \quad \|\mathbf{w}\|_2 \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i) \geq 1 \quad \text{for } i = 1, \ldots, N \tag{1.2.1}$$

A limitation of many machine learning algorithms, including the standard SVM, is that they are applicable only to data in the form of real-valued vectors. The standard SVM is also limited by its separator being linear in the input space. The *kernel trick* (Schölkopf and Smola, 2001) removes both of these limitations by observing that several learning problems, such as the dual problem of SVMs, interact with data only through inner products of pairs of data points. The Lagrange dual of the (soft-margin) SVM, with

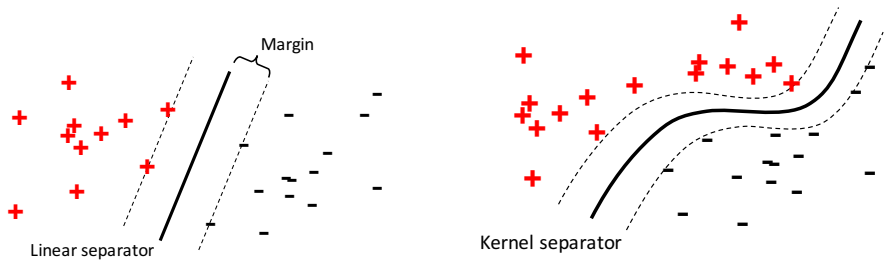


Figure 1.2.2: *Linear (left) and kernel (right) large-margin classifiers.*

multipliers α_i, is

$$\begin{array}{ll} \underset{\boldsymbol{\alpha}}{\text{maximize}} & \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^\top \mathbf{x}_j) \\ \text{subject to} & \sum_{i=1}^N \alpha_i y_i \text{ and } 0 \leq \alpha_i \leq C \qquad \text{for } i = 1, \ldots, N \ . \end{array} \tag{1.2.2}$$

The key realization of kernel methods is that inner products in the input space $\mathcal{X}$, such as in the objective above, may be replaced by an inner product $\phi(x_i)^\top \phi(x_j)$ of a mapping $\phi : \mathcal{X} \longrightarrow \mathcal{H}$ into a vector space $\mathcal{H}$. In effect, by choosing (or learning) a non-linear mapping ϕ, a non-linear classifier may be learned, without changing objective functions, see figure 1.2.2. Crucially however, the mapping ϕ need not be explicitly defined, but is only accessed implicitly through the so-called kernel function defined by $k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$. As a result, we may define ϕ through the kernel function k, instead of the other way around. A function $k : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ is a valid kernel only if it induces a mapping ϕ such that $k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$ (Schölkopf and Smola, 2001). There are several sufficient conditions on k for this to be true. A commonly used condition is that the Gram matrix $K = (k(\mathbf{x}_i, \mathbf{x}_j)))_{ij \in [n]}$ is positive semi-definite for any choice of sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

The procedure outlined above, called the kernel trick, lets us learn models by considering only similarities (kernels) between data points instead of the data points themselves. This idea opens up machine learning methods such as SVMs to data that are not usually represented by real-valued vectors, but have some natural notion of similarity. Graph kernels (S. Vishwanathan, Schraudolph, Kondor, and K. M. Borgwardt, 2010) attempt to define similarity functions between graphs that are informative to a learning algorithm. For example, if the goal is to classify molecular compounds as toxic or harmless to the human body, the kernel has to capture similarities in the atoms (node labels) and bonds (edges) making up the molecule. In the sequel, we develop graph kernels and apply them in classification tasks.

Clustering & community detection

Clustering, finding clusters of similar objects in a set, is perhaps the most common unsupervised machine learning problem. While easy to put into words, the objective of clustering problems is often hard to express mathematically. A common assumption is that there are k centroids in the input space, and that each object belongs to the closest

of the centroids. This leads naturally to k-means clustering which attempts to find both centroids and assignments that minimizes the sum of distances from objects to their respective centroids. Given a set of data points $x_1, \ldots, x_m$, seeks k centroids $\mu_1, \ldots, \mu_k$ and a partition $S = \{S_1, \ldots, S_k\}$ of the data points, the k-means objective is

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x_i - \mu_i\|^2 \ . \tag{1.2.3}$$

Finding the global optimum of (1.2.3) is NP-hard in the general case. As a result, practitioners often settle for a local optimum found by a greedy algorithm such as the widely used Lloyd's algorithm. A remaining issue is the choice of k, the number of clusters. In practice, the choice is often based on domain knowledge or heuristics.

In the world of graphs, clustering translates to *community detection* (Fortunato, 2010; M. E. J. Newman and Girvan, 2004), and the objective is to partition the set of nodes of a single graph into groups[1]. The name comes from applications to social networks, in which the goal is to find densely connected groups of people within a population, i.e. communities. Figure 1.2.3 visualizes an example problem and solution. Similar to clustering of vector valued data is that there is no universally agreed upon measure for what constitutes a good cluster or community structure (M. E. J. Newman and Girvan, 2004). In other words, not only do we not know the right algorithm - we don't know the right objective.

A common strategy in community detection is to look for communities such that nodes in the same communities are strongly connected (many within-cluster edges), and nodes from different communities are weakly connected (few between-cluster edges). The Girvan-Newman algorithm (Girvan and M. E. Newman, 2002), for example, repeatedly identifies and removes edges between communities, leaving only the well-connected communities behind. Modularity maximization (M. E. Newman, 2004; Blondel, Guillaume, Lambiotte, and Lefebvre, 2008) is one way that intuition about what constitutes a good community partition has been formalized. Modularity measures the within-cluster connectivity of a given partition, as compared to the same measure on a random graph with the same node set. Even though globally maximizing modularity is NP-hard (Brandes et al., 2008), it remains a popular method.

Instead of designing completely new clustering methods, such as modularity maximization, when working with graph data, we may represent the graph in a form admissible to existing clustering methods. In this thesis, we show that general purpose clustering algorithms, such as Lloyd's k-means clustering, can be used for community detection when applied to a so-called geometric embedding of a graph. This idea reduces the need for specialized methods and makes a larger body of algorithms applicable to graphs.

[1] Clustering of graphs could of course also mean partitioning multiple graphs into groups, rather than partitioning nodes. This is not addressed in the thesis however.

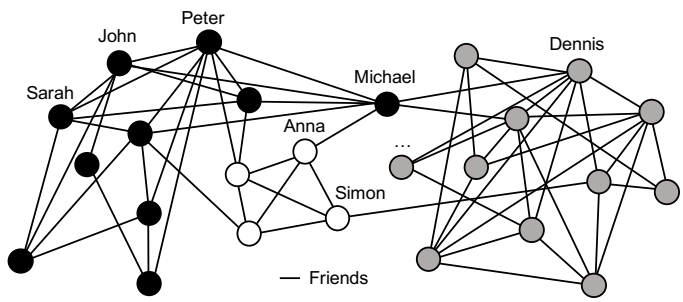


Figure 1.2.3: *Community detection in a social network. Detected communities are shaded in black, gray and white respectively. Note that the layout of the nodes is not available as input, and may convince the reader that the problem is simpler than it actually is.*

Chapter 2

Geometric embeddings of graphs

This chapter introduces geometric embeddings of graphs, including the embedding associated with the celebrated Lovász number (Lovász, 1979) which has had great impact on graph algorithms (Goemans, 1997). We cover classical results motivating the use of these embeddings in machine learning methods, as well as new results which generalize the Lovász embeddings to weighted graphs. In chapter 3, we see how these tools are useful in solving several machine learning problems.

In section 1.2, we saw that representing graphs in geometric spaces, or through kernel functions, can be very useful when applying machine learning methods, reducing the need for specializing learning algorithms to graphs. A natural first step in making the relational information of a graph accessible to learning algorithms is to use a *geometric embedding*. A geometric embedding U_G of a graph $G = (V, E)$ is a set of vectors $\{\mathbf{u}_i\}_{i \in V}$, each of which represents a node $i \in V$ in a space $\mathbb{R}^p$,

$$U_G := \{\mathbf{u}_i \in \mathbb{R}^p\}_{i \in V} \ . \tag{2.0.1}$$

Clearly, there is an infinite number of possible embeddings U_G of any graph G, and we need some criteria by which we can select one to use. An implicit goal for most embedding algorithms is to (approximately) preserve the connectivity of the graph G in the form of closeness or distance in the embedding space. For example, we could attempt to find an embedding such that the distance between $\mathbf{u}_i$ and $\mathbf{u}_j$ is small if nodes i and j are connected, and large if they are disconnected.

When embedding graphs for machine learning, our goal is to create a representation that is as informative to the learning algorithm as possible. By transferring the connectivity information to a geometric space, several goals are accomplished. First, as noted previously, a large family of machine learning methods, including classification and clustering algorithms, only applicable to vector valued data are now accessible through geometric embeddings. For example, performing k-means clustering (e.g. Lloyd (1982)) on the node embeddings is a straight-forward way of doing community detection, see section 1.2. Second, the embedding may provide a softer distance measure between nodes

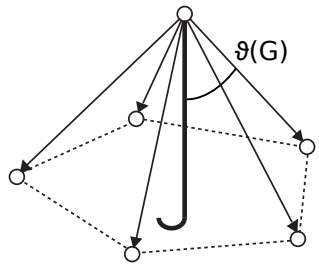


Figure 2.1.1: *Lovász number $\vartheta(G)$ and the orthogonal representation for the pentagon.*

than does the edge set. For example, two disconnected nodes that have a short path connecting them (e.g. a common neighbor) may be represented by vectors separated by a smaller distance in the embedding space than nodes that are connected only by long paths. Using a softer distance measure can also help to de-noise the data, i.e. reduce the impact of edges caused by noise.

A special case of geometric embeddings of graphs, that we will study further, is so-called *orthogonal* representations. We say that an embedding U_G is orthogonal if

$$(i,j) \notin E \Rightarrow \mathbf{u}_i^\top \mathbf{u}_j = 0 \ , \tag{2.0.2}$$

and *orthonormal* if also $\|\mathbf{u}_i\| = 1$ for all $i \in V$. The intuition is clear: if two objects are not connected, not similar, they should have no overlap in the vector space.

We note that this definition on its own does not provide fruitful ways of representing graphs. For example, letting each $\mathbf{u}_i$ be a different basis vector in $\mathbb{R}^n$, results in an orthonormal representation, but does not preserve the graph structure at all. Instead, we are interested in embeddings that capture the connectivity structure in a way that facilitates learning. In the sequel, we focus on a particular orthonormal representation, the one associated with the celebrated Lovász number.

2.1 Lovász number and embedding

Lovász number (Lovász, 1979), usually denoted $\vartheta(G)$, was introduced as a polynomial-time computable upper bound on the Shannon capacity of G, an important concept in information theory for which a polynomial time algorithm is not known. It was also shown to have the following attractive relationship with the clique number $\omega(G)$ and the chromatic number $\chi(G)$, both of which are NP-hard to compute.

$$\omega(G) \leq \vartheta(\bar{G}) \leq \chi(G) \tag{2.1.1}$$

Here $\bar{G}$ is the graph complement to G. The result above is sometimes referred to as Lovász *sandwhich theorem* (Knuth, 1993). Because of the polynomial complexity of computing $\vartheta(G)$ and its relation to several important quantities, known to be NP-hard to compute, Lovász number has received considerable attention since its introduction (Goemans, 1997).

Formally, $\vartheta(G)$ is defined as the smallest angle[1] of any cone, enclosing any orthonormal representation U_G of G,

$$\vartheta(G) = \min_{\mathbf{c}, U_G} \max_{i \in V} \frac{1}{(\mathbf{c}^\top \mathbf{u}_i)^2} \ , \tag{2.1.2}$$

where the minimization is taken over all orthonormal representations U_G and all unit vectors $\mathbf{c}$. An illustration of $\vartheta(G)$ for the pentagon graph can be seen in figure 2.1.1.

Since its introduction, it has had large impact on combinatorial optimization, graph theory and approximation algorithms (Goemans, 1997). $\vartheta(G)$ and the associated minimizing orthogonal representation, has been used to derive state-of-the-art approximation algorithms for max k-cut (Frieze and Jerrum, 1997), graph coloring (Karger, Motwani, and Sudan, 1998; Dukanovic and Rendl, 2008) and planted clique problems (Feige and Krauthgamer, 2000). These results provide ample motivation for using the embedding in machine learning methods for graphs.

It is well-known that $\vartheta(G)$ can be computed to arbitrary precision in time polynomial in the number of nodes, by means of solving a semi-definite program (Lovász, 1979). While polynomial, state-of-the-art algorithms for computing Lovász number have time complexities $O(n^5 \log n \cdot \epsilon^{-2})$ (Chan, Chang, and Raman, 2009) and $O(n^2 m \log n \cdot \epsilon^{-1} \log^3(\epsilon^{-1}))$ (Iyengar, Phillips, and Stein, 2011), where n and m are the number of nodes and edges respectively and ϵ the error. These complexities are typically prohibitively large for machine learning applications, which thrive on using large datasets. In the next section, we introduce a recent approximation to $\vartheta(G)$ (Jethava, Martinsson, Bhattacharyya, and Dubhashi, 2014) that greatly reduces the computational cost.

2.1.1 Kernel characterization and the SVM-ϑ approximation

Luz and Schrijver (2005) showed that $\vartheta(G)$ can be characterized by a convex quadratic programming problem. Jethava, Martinsson, Bhattacharyya, and Dubhashi (2014) built on this result by defining a simpler and equivalent formulation that made a crucial link to a very well known machine learning problem, namely the SVM. They observed that a one-class kernel SVM (Schölkopf, Platt, et al., 2001), like $\vartheta(G)$, searches for the minimum cone enclosing a set of vectors, and that for an optimally chosen kernel, the SVM and ϑ cones become equivalent. Crucially, they also observed that for a large family of graphs, a specific kernel with a closed-form characterization produces a constant factor approximation to ϑ.

Formally, for any graph $G = (V, E)$, such that $n = |V|$, it holds that

$$\vartheta(G) = \min_{\kappa \in \mathcal{K}_G} \omega(\kappa) \tag{2.1.3}$$

where $\omega(\kappa)$ is the solution to a kernel one-class SVM,

$$\omega(\kappa) = \max_{\substack{\alpha_i > 0 \\ i=1,\ldots,n}} 2 \sum_{i=1}^{n} \alpha_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j \kappa_{ij} \ . \tag{2.1.4}$$

[1] We note that $\vartheta(G)$ is really the inverse squared cosine of the half-angle of the cone, but as they grow and decrease together, $\vartheta(G)$ is often referred to as the "angle".

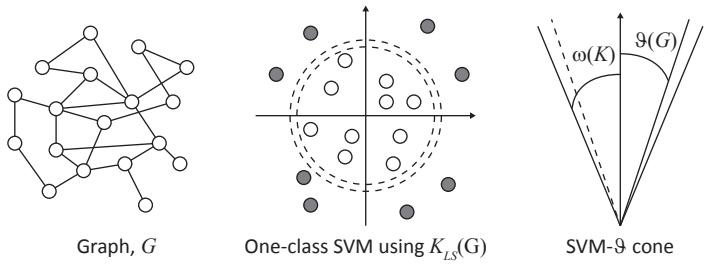


Figure 2.1.2: *The components of* SVM-ϑ *and an illustration of its relationship to* $\vartheta(G)$.

$\mathcal{K}_G$ is the set of kernel matrices that respect the same orthogonality constraints as the orthonormal representations defined in section 2.1,

$$\mathcal{K}_G := \{\kappa \in S_n^+ | \kappa_{ii} = 1, \forall i, \kappa_{ij} = 0, (i,j) \notin E\} \ , \tag{2.1.5}$$

and S_n^+ is the set of $n \times n$ positive semi-definite matrices. With slight abuse of notation, from now on, we let $\{\alpha_i\}_{i=1}^n$ denote the *maximizers* of (2.1.4).

By simply rewriting ϑ we have not gained anything in terms of computational complexity. The optimization over $\mathcal{K}_G$ involves solving a semi-definite program and the SVM simultaneously, which in general is no faster than computing $\vartheta(G)$. Instead, our hope is that a particular choice of kernel κ gives a good approximation to the minimum of (2.1.3), without doing the optimization over κ. As it happens, there is a choice of κ, that while not always optimal, gives good theoretical guarantees of this nature. We define this choice of κ below.

Definition 2.1.1 (Luz and Schrijver (2005))**.** *Let A be the adjacency matrix of G, $\rho \geq -\lambda_n(A)$, with $\lambda_n(A)$ the minimum eigenvalue of A. Then,*

$$\kappa_{LS}(G) = \frac{A}{\rho} + I \succeq 0 \ . \tag{2.1.6}$$

We refer to (2.1.6) as the LS-labelling (for Luz-Schrijver) of G. $\kappa_{LS}(G)$ can be thought of as a mapping from the adjacency matrix A to the set of positive semi-definite matrices. Jethava, Martinsson, Bhattacharyya, and Dubhashi (2014) showed that,

$$\omega(\kappa_{LS}(G)) = \sum_{i=1}^{n} \alpha_i \tag{2.1.7}$$

where α_i are the maximizers of (2.1.4). Henceforth, when referring to SVM-ϑ, we refer to the solution to of (2.1.4) with $\kappa = \kappa_{LS}$. SVM-ϑ is illustrated in figure 2.1.2, for this particular choice of κ. Jethava, Martinsson, Bhattacharyya, and Dubhashi (2014) proved that on families of graphs, referred to by them as SVM-ϑ graphs, $\omega(\kappa_{LS})$ is w.h.p. a constant factor approximation to $\vartheta(G)$,

$$\vartheta(G) \leq \omega(\kappa_{LS}) \leq \gamma\vartheta(G) \ . \tag{2.1.8}$$

Important graph families such as Erdős-Rényi random graphs and planted clique graphs have this property. SVM-ϑ is for a given kernel computable in $O(n^2)$ due to the one-class SVM (Hush, Kelly, Scovel, and Steinwart, 2006), but κ_{LS} requires $O(n^3)$ time due to the computation of the minimum eigenvalue of A.

2.2 Lovász embedding of weighted graphs

A natural generalization in any treatment of graphs is to accommodate graphs with weights on nodes and edges. This is also the case in machine learning. Node and edge weights are often associated with strong intuition or associations to real-world objects and we may for example interpret edge weights as the strength of a connection between two objects, and node weights as the importance or relevance of the nodes. While there is a version of the Lovász number for node-weighted graphs Knuth (1993), somewhat surprisingly, there is no classical definition of the Lovász number for graphs with weights on the edges. In Paper I, we introduce a unifying extension of the Lovász number and embedding for graphs with weights on both nodes and edges. We outline the results of this paper below.

We begin by observing that the orthogonality constraints, $\mathbf{u}_i^\top \mathbf{u}_j = 0, \forall (i,j) \notin E$, underpinning the classical definition of $\vartheta(G)$, make little sense for weighted graphs. In general weighted graphs all edges are present (albeit with different weights), and there can be no orthogonality constraints on the embedding if none of the weights equal 0. Furthermore, orthogonality constraints alone cannot capture the difference between $w_{ij} = 0.1$ and $w_{ij} = 1.0$. A naïve generalization, forcing the inner products of node embeddings to exactly equal the corresponding edge weight is too restrictive, and not always feasible (the weight matrix must be positive semidefinite).

An alternative version of Lovász number, and its bound on the Shannon capacity, was given by Delsarte (Schrijver, 1979), in which the equality constraints

$$\mathbf{u}_i^\top \mathbf{u}_j = 0, \forall (i,j) \notin E$$

are replaced by inequalities

$$\mathbf{u}_i^\top \mathbf{u}_j \leq 0, \forall (i,j) \notin E \ ,$$

resulting in the kernel characterization

$$\vartheta^1(G) = \min_{\kappa \in \mathcal{K}(G)} \omega(\kappa), \quad \mathcal{K}(G) := \{\kappa \succeq 0 \mid \kappa_{ij} \leq 0, \forall (i,j) \notin E\} \ .$$

Schrijver, 1979 also showed that, with $\alpha(G)$ the independence number of G,

$$\alpha(G) \leq \vartheta^1(G) \leq \vartheta(G) \ .$$

In our generalization of ϑ to weighted graphs, we build on the Delsarte version by incorporating edge weights in the inequality constraints. Let $G = (V, E, W)$ be an edge-weighted graph with $w_{ij} \in [0,1]$. We may interpret w_{ij} as the similarity of nodes i and j, e.g. the similarity of two sentences in a document or the (inverse) distance between two cities connected by roads. Our constraint for weighted graphs is

$$\mathbf{u}_i^\top \mathbf{u}_j \leq w_{ij}, \forall i,j \in V \ .$$

For a simple graph, with weights in $\{0, 1\}$, our inequality constraints reduce directly to those of Delsarte. Also recall that we are considering normalized embeddings, so that $\mathbf{u}_i^\top \mathbf{u}_i = 1$. This means that when $w_{ij} = 1$, there is no additional constraint from the inequalities, just like in the standard definition of ϑ. When w_{ij} is small, the constraint is strong, and when w_{ij} is large, it is weak. This means that weakly connected nodes will be forced apart in the embedding space, more so than nodes that have stronger connections.

Unlike for edge weighted graphs, there is a classical definition of Lovász number for graphs $G = (V, E, \Sigma)$ with only node weights $\sigma_i \in [0, 1]$, see for example Knuth (1993). It is defined by,

$$\min_{\{\mathbf{u}_i\}} \min_{\mathbf{c}} \max_i \frac{\sigma_i}{(\mathbf{c}^\top \mathbf{u}_i)^2}$$

with $\mathbf{u}_i$ subject to the usual orthogonality constraints.

We can incorporate this in our kernel generalization by adding a constraint on the diagonal of κ,

$$\kappa_{ii} = \frac{1}{\sigma_i} \ .$$

This constraints ensures that the norm of the embedding of each node is the inverse of the square root of the node weight. Trivially, this reduces to the standard definition if all weights equal 1. We may unify our definition for graphs with weights on both nodes and edges by the following set of kernels.

$$\vartheta^1(G, \sigma, W) = \min_{\kappa \in \mathcal{K}(G,\sigma,W)} \omega(\kappa), \quad \mathcal{K}(G, \sigma, W) := \left\{ \kappa \succeq 0 \mid \kappa_{ii} = \frac{1}{\sigma_i}, \kappa_{ij} \leq \frac{w_{ij}}{\sqrt{\sigma_i \sigma_j}} \right\}$$

Our definition generalizes both the unweighted (Delsarte) version of Lovász number, and the node-weighted version, to accommodate graphs with edge weights. If node and edge weights are uniform, $\sigma_i = 1 \ \forall i \in V$, $w_{ij} = 1 \ \forall (i, j) \in E$, the definition reduces to the original version. An interesting remaining problem is to relate this definition back to information theory and the Shannon capacity. For an overview of the weighted extensions presented in Paper I, see table 2.2.1.

Table 2.2.1: Characterizations of weighted theta functions. In the first row are characterizations following the original definition. In the second are kernel characterizations. The bottom row are versions of the LS-labelling (Jethava, Martinsson, Bhattacharyya, and Dubhashi, 2014). In all cases, $\|\mathbf{u}_i\| = \|\mathbf{c}\| = 1$. A refers to the adjacency matrix of G.

Unweighted	**Node-weighted**	**Edge-weighted**
$\min_{\{\mathbf{u}_i\}} \min_{\mathbf{c}} \max_i \frac{1}{(\mathbf{c}^\top \mathbf{u}_i)^2}$ $\mathbf{u}_i^\top \mathbf{u}_j \leq 0,\ \forall (i,j) \notin E$	$\min_{\{\mathbf{u}_i\}} \min_{\mathbf{c}} \max_i \frac{\sigma_i}{(\mathbf{c}^\top \mathbf{u}_i)^2}$ $\mathbf{u}_i^\top \mathbf{u}_j = 0,\ \forall (i,j) \notin E$	$\min_{\{\mathbf{u}_i\}} \min_{\mathbf{c}} \max_i \frac{1}{(\mathbf{c}^\top \mathbf{u}_i)^2}$ $\mathbf{u}_i^\top \mathbf{u}_j \leq w_{ij},\ i \neq j$
$\mathcal{K}_G = \{\kappa \succeq 0 \mid \kappa_{ii} = 1,$ $\kappa_{ij} = 0, \forall (i,j) \notin E\}$	$\mathcal{K}_{G,\sigma} = \{\kappa \succeq 0 \mid \kappa_{ii} = 1/\sigma_i,$ $\kappa_{ij} = 0, \forall (i,j) \notin E\}$	$\mathcal{K}_{G,W} = \{\kappa \succeq 0 \mid \kappa_{ii} = 1,$ $\kappa_{ij} \leq w_{ij}, i \neq j\}$
$\kappa_{LS} = \frac{A}{\lvert\lambda_n(A)\rvert} + I$	$\kappa^\sigma_{LS} = \frac{A}{\sigma_{max}\lvert\lambda_n(A)\rvert} + \mathrm{diag}(\sigma)^{-1}$	$\kappa^S_{LS} = \frac{W}{\lvert\lambda_n(W)\rvert} + I$

Chapter 3

Learning with graph embeddings

In this chapter, we describe several ways of using geometric embeddings of graphs (see section 2) to improve existing, or define new, machine learning algorithms. We introduce *graph kernels* and the kernel based on the Lovász embedding developed in Paper III, as well as a matching-based approach to graph classification, introduced in Paper II. Finally, we show how the Lovász embedding can be used to perform community detection, as described in Paper I.

3.1 Graph classification and graph kernels

One of the hurdles when applying machine learning methods to graphs is that many of the most widely used learning algorithms are designed for vector-valued representations of data only. Kernel methods on the other hand, see section 1.2, remove this constraint by accepting input in the form of similarity measures between observations (Schölkopf and Smola, 2001). Graph kernels (Gärtner, Flach, and Wrobel, 2003; S. Vishwanathan, Schraudolph, Kondor, and K. M. Borgwardt, 2010) are the specialization of this idea to graphs. They are similarity measures on graphs that adhere to the positive definiteness constraint of a kernel function, and enjoy all the theoretical benefits associated with kernel methods in general. Graph kernels have also gained popularity in practical applications, and have been used in diverse fields including computational biology (Schölkopf, Tsuda, and Vert, 2004), chemoinformatics (Mahé and Vert, 2009) and information retrieval (see Paper IV).

A natural way of measuring similarities between graphs doing so is to count all subgraph patterns, e.g. triangles or cycles, that appear in both graphs (Gärtner, Flach, and Wrobel, 2003). Counting *all* subgraph patterns however, implicitly solves the subgraph isomorphism problem, which is widely known to be NP-hard. As we have little hope of finding an efficient algorithm to solve this problem in the general case, many graph kernels have restricted the comparison to specific types of subgraphs, such as paths or

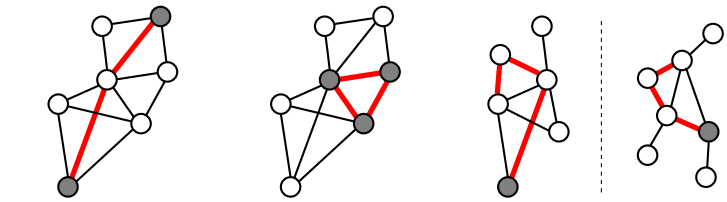


Figure 3.1.1: *Illustration of the subgraphs considered by different graph kernels. The shortest-path kernel (left) compares histograms of shortest-path lengths. The graphlet kernel (middle) compares counts of subgraph patterns of a certain size. The random walk kernel compares simultaneous random walks in two graphs.*

sub-trees. While not as precise as comparing all subgraphs, graph kernels represent an attractive trade-off between expressivity and computational efficiency (Ramon and Gärtner, 2003). For example, random walk kernels (Gärtner, Flach, and Wrobel, 2003; Kashima, Tsuda, and Inokuchi, 2003) compare weighted counts of random walks of every length. The shortest-path kernel (K. M. Borgwardt and Kriegel, 2005) compare features of the shortest paths between every pair of nodes in each graph, and subtree kernels (Ramon and Gärtner, 2003; Mahé and Vert, 2009) compare tree patterns. For an illustration of three widely-used kernels, see figure 3.1.1. Recently, there has been a lot of research into handling node and edge attributes efficiently. An important family of kernels in that line of work are the Weisfeiler-Lehman kernels (Shervashidze, Schweitzer, et al., 2011), based on the Weisfeiler-Lehman test of graph isomorphism.

One of the limitations of existing graph kernels is that they may fail to capture global properties of graphs. It has been shown that there are graph properties which cannot be captured by studying only small local structures. Perhaps the most celebrated result on this topic is Erdős' seminal proof of the existence of graphs with high girth and high chromatic number (Alon and Spencer, 1992, p. 41-42), graphs for which all small-sized subgraphs will be trees. It will seem to a kernel with local focus that the entire graphs are trees, when they in fact have large cycles. Because of this issue, we seek graph kernels and representations that capture precisely such global properties as well. The geometric embedding associated with Lovász number, introduced in chapter 2, will lead us to such a kernel in the next section.

We let $K : \mathcal{G} \times \mathcal{G} \longrightarrow \mathbb{R}$ denote a graph kernel for a space of graphs $\mathcal{G}$. Recall from section 1.2 that a kernel K is only valid if it induces an inner product, i.e. $K(G, G') = \phi(G)^\top \phi(G')$ in some Hilbert space. A useful notion when defining graph kernels is the concept of *R-convolution* kernels (Shervashidze, Schweitzer, et al., 2011; S. Vishwanathan, Schraudolph, Kondor, and K. M. Borgwardt, 2010). We define the R-convolution kernel below.

Definition 3.1.1 (Haussler (1999))**.** *Let χ and χ' be spaces and $k : \chi' \times \chi' \longrightarrow \mathbb{R}$ a positive semi-definite kernel. The R-convolution kernel for points $x, y \in \chi$, associated*

with finite subsets $\chi'_x \subseteq \chi'$ *and* $\chi'_y \subseteq \chi'$ *is defined by*

$$K(x,y) = \sum_{(x',y') \in \chi'_x \times \chi'_y} k(x',y') \ . \tag{3.1.1}$$

R-convolution kernels K are positive semidefinite as long as the base kernel k is positive semidefinite (Haussler, 1999). This property is useful when proving the validity of a new kernel, as it is sufficient to prove that the new kernel is an R-convolution kernel. The R-convolution kernel was later generalized to *mapping kernels* (Shin and Kuboyama, 2008). Classical graph kernels based on the R-convolution kernel compare features of small subgraphs or walks extracted from the original graphs. The graphlet kernel (Shervashidze, S. Vishwanathan, et al., 2009) counts instances of subgraph patterns of at most 5 nodes. The random walk kernel (Gärtner, Flach, and Wrobel, 2003) counts walks of any length, but the counts are often weighted with a factor decreasing exponentially with the length of the walk (S. V. N. Vishwanathan, K. M. Borgwardt, and Schraudolph, 2006). Subtree kernels (Ramon and Gärtner, 2003; Shervashidze and K. Borgwardt, 2009), consider tree patterns of a limited size. Furthermore, as Shervashidze, S. Vishwanathan, et al. (2009) identified, *"There is no theoretical justification on why certain types of subgraphs are better than others"*.

In the next section, we define a graph kernel based on Lovász ϑ which adopts a broader focus, in order to capture global features of graphs. Furthermore, in contrast to existing kernels, we provide theoretical justification for the usefulness of the kernel.

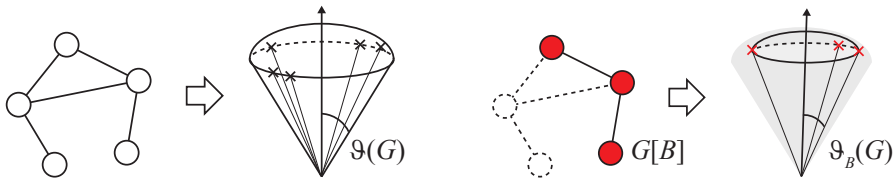
3.2 The Lovász ϑ kernel

This section introduces the graph kernels developed in Paper III, designed to capture important global properties of graphs, such as the girth or the clique number. In contrast to earlier graph kernels, focusing on local structure to increase efficiency, our kernels are designed to capture global properties of graphs. To remain efficient, the kernels still decompose into features of substructures, but in a manner that retains desired properties.

Motivated by the strong connection between $\vartheta(G)$ and global graph properties such as the size of the largest cut and the chromatic number, as described in section 2.1, we begin by defining the *Lovász ϑ kernel* using $\vartheta(G)$. We then define a kernel based on the SVM-ϑ approximation, enabling faster computation while retaining good accuracy. In Paper III, we show that for certain classification tasks, we can bound the separation margin induced by our kernels, providing theoretical justification for the choice of graph kernels in some applications. We also show empirically that our kernel is competitive with state-of-the-art graph kernels on established benchmark datasets.

In this chapter, when referring to an orthonormal representation $U_G = \{\mathbf{u}_1, \dots \mathbf{u}_n\}$, we always refer to the maximizer of (2.1.2). For notational convenience, we begin by defining the *Lovász value* of a subset of nodes $B \subseteq V$, which represents the angle of the smallest cone enclosing a subset of vectors $U_{G|B} \subseteq U_G$, as defined below.

Definition 3.2.1. *Let $G[B]$ be the subgraph of $G = (V, E)$ induced by $B \subseteq V$. Then, the*



(a) (b)

Figure 3.2.1: *An illustration of the difference between $\vartheta(G)$ (a) and $\vartheta_B(G)$ (b).*

Lovász value *of $G[B]$ is defined by,*

$$\vartheta_B(G) = \min_{\mathbf{c}} \max_{\mathbf{u}_i \in U_{G|B}} \frac{1}{(\mathbf{c}^\top \mathbf{u}_i)^2} \ , \tag{3.2.1}$$

where $U_{G|B} := \{\mathbf{u}_i \in U_G \,|\, i \in B\}$ and U_G is the maximizer of (2.1.2)*.*

The Lovász value of subgraphs will constitute the components that we compare across graphs, akin to e.g. subgraph patterns in the graphlet kernel. The crucial difference however, is that $\vartheta_B(G)$ depends on the global connectivity properties of the graph G, not just the structure of $G[B]$. Note for example that in general $\vartheta_B(G) \neq \vartheta(G[B])$. More specifically, $\vartheta_B(G)$ adheres to the global set of orthogonality constraints, defined by all of G. In contrast $\vartheta(G[B])$ uses only the information present in $G[B]$ and is therefore a completely local feature. The difference between these quantities is what we'll exploit in building our kernel, and is illustrated in figure 3.2.1.

We now present the formal definition of Lovász ϑ kernel in terms of the Lovász values of subgraphs.

Definition 3.2.2 (Lovász ϑ kernel)**.** *The Lovász ϑ kernel on two graphs, G, G', with a positive semi-definite kernel $k : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$, is defined by*

$$K(G, G') = \sum_{B \subseteq V} \sum_{\substack{C \subseteq V' \\ |C|=|B|}} \frac{1}{Z_{B,C}} \cdot k(\vartheta_B, \vartheta'_C) \ , \tag{3.2.2}$$

with $\vartheta_B = \vartheta_B(G)$, $\vartheta'_C = \vartheta_C(G')$, and $Z_{B,C} = \binom{n}{|B|}\binom{n'}{|B|}$.

The Lovász ϑ kernel compares Lovász values for all pairs of node subsets in two graphs. In effect, this corresponds to comparing the independence structure of subgraphs, as ϑ_B depends on which vectors in $U_{G|B}$ are orthogonal, which in turn depends on which nodes in B are independent. We can also prove the following result, important for any graph kernel.

Lemma 3.2.1 (Paper III, §3.1)**.** *The Lovász ϑ kernel, as defined in* (3.2.2)*, is a positive semi-definite kernel.*

Proof sketch. The proof involves showing that the kernel is an R-convolution kernel (Haussler, 1999). For a complete proof, see Paper III.

As Lovász number is prohibitively expensive to compute for most graphs (see section 2.1), we proceed to define a faster, approximate version of the Lovász ϑ kernel in the next section.

3.2.1 The SVM-ϑ kernel

To speed up computation of the Lovász ϑ kernel, while retaining similar properties, we introduced the SVM-ϑ kernel in Paper III. The kernel rests on an SVM-ϑ analogue of the Lovász value ϑ_B, which is used as a feature of subgraphs. We note that α_i, the optimizers of SVM-ϑ, adhere to the global optimality conditions of (2.1.4) defined by the edge set, and thus capture global properties of graphs. That is, similar the case for to ϑ_B, $\sum_{i\in B}\alpha_i(G) \neq \sum_{i\in B}\alpha_i(G[B])$ in general. Based on this observation, and the connection between $\omega(\kappa)$ and $\vartheta(G)$ as described in section 2.1.1, we let $\sum_{i\in B}\alpha_i$ serve as an analogue for ϑ_B in (3.2.2), when defining our new kernel.

Definition 3.2.3. *The* SVM-ϑ *kernel is defined, on two graphs G, G', with corresponding $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]$, $\boldsymbol{\alpha}' = [\alpha_1, \ldots, \alpha_{n'}]$ maximizers of* (2.1.4) *for $\kappa = \kappa_{LS}(G)$, with a positive semi-definite kernel $k : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$, as*

$$K(G, G') = \sum_{B\subseteq V} \sum_{\substack{C\subseteq V' \\ |C|=|B|}} \frac{1}{Z_{B,C}} k(\mathbf{1}^\top \boldsymbol{\alpha}_B, \mathbf{1}^\top \boldsymbol{\alpha}'_C) \tag{3.2.3}$$

where $\boldsymbol{\alpha}_B = [\alpha_{B(1)}, \ldots, \alpha_{B(d)}]^\top$ with $d = |B|$, $Z_{B,C} = \binom{n}{|B|}\binom{n'}{|C|}$ and $\mathbf{1}$ the all one vector of appropriate size.

We make a note that while $\sum_{i=1}^{n}\alpha_i$ is an upper bound on $\vartheta(G)$, and a constant-factor approximation for classes of graphs, the same cannot be said for the entire SVM-ϑ kernel in relation to the Lovász ϑ kernel. This is due to the fact that $\sum_{i\in B}\alpha_i$ is not a tight bound on $\vartheta_B(G)$ for all $B \subset V$, even for SVM-ϑ graphs. Nevertheless, the SVM-ϑ kernel is a valid kernel capable of capturing global properties of graphs, such as the clique number, as we will see in the next section. We can also show the following result.

Lemma 3.2.2 (Paper III, §4)**.** *The* SVM-ϑ *kernel, as defined in* (3.2.3)*, is a positive semi-definite kernel.*

Proof sketch. The proof involves showing that the kernel is an R-convolution kernel (Haussler, 1999). For a complete proof, see Paper III.

3.2.2 Efficient computation

Direct evaluation of the Lovász ϑ and SVM-ϑ kernels is computationally very expensive. This is easily realized by noting the sums over subsets in the definition of either kernel, for which the number of terms grows exponentially with the number of nodes in the graph. To apply these kernels to large graphs, we need to rely on approximate computation. In Paper III, we derive such a scheme based on sampling.

To approximate the sums of the Lovász ϑ and SVM-ϑ kernels, we note that both kernels can be written on the following form.

$$K(G, G') = \sum_{B \subseteq V} \sum_{\substack{C \subseteq V' \\ |C|=|B|}} \frac{1}{Z_{B,C}} k(f_B(G), f_C(G')) \tag{3.2.4}$$

Here, $f_B(G) = \vartheta_B(G)$ for the Lovász ϑ kernel and $f_B(G) = \sum_{j \in B} \alpha_j(G)$ for the SVM-ϑ kernel, and $Z_{B,C} = \binom{n}{|B|}\binom{n'}{|C|}$ for both. As this is the case, we derive an approximate computation scheme for the general form in (3.2.4) applicable to both kernels.

The key observation is that (3.2.4) is easily decomposed into pairs of subsets of nodes. Now, instead of considering all pairs, we sample a small (polynomial) number of subsets for each graph, resulting in overall polynomial complexity. Formally, let S_d and S'_d be multisets of t uniformly sampled subsets of V and V' respectively, such that $|S_d| = |S'_d| = d$. If $d > n$, let $S_d = \emptyset$ and analoguously for S'_d and n'. Then, define,

$$\hat{K}(G, G') = \sum_{d=1}^{d_{max}} \sum_{B \in S_d} \sum_{C \in S'_d} \frac{1}{|S_d||S'_d|} k(f_B(G), f_C(G')) \ . \tag{3.2.5}$$

Here the maximum size of subsets included is limited to d_max. It is plain to see that (3.2.5) converges to (3.2.4), when $d_{max} = n$ and the number of samples, k goes to infinity. In practice, it is of course not feasible to use an infinite number of samples. Instead, in Paper III, we provide sample complexity results, bounding the number of samples needed to achieve a given quality of approximation.

3.2.3 Empirical evaluation

For a comprehensive empirical evaluation of both the Lovász ϑ kernel and the SVM-ϑ kernel, on synthetic graphs with known global properties as well as real-world graphs used as benchmarks for graph kernels, see Paper III, section 5.

3.3 Graph classification using matchings of geometric embeddings

In the previous section, and in Paper III, we introduced a novel graph kernel based on the Lovász geometric embedding of graphs. The Lovász ϑ kernel compares graphs by measuring the minimum enclosing angle of subsets of node embeddings, but while Lovász ϑ is intimately associated with the angular distance between node embeddings, this similarity measure does not necessarily generalize well to other embedding algorithms. Furthermore, the exact kernel involves summing over all possible pairs of node subsets, which is impractical for most real-world graphs. Motivated by these issues, we developed a similarity measure on graphs in Paper II, based on *matchings* of geometric embeddings. We cover the important results of the paper below.

In many applications, observations represent composite objects, made up of several distinct parts. For example, cars have got engines, doors, wheels etc., and each of these components can be compared in isolation to their equivalent in another car. Graphs, decomposable by e.g. nodes, edges or subgraphs, also fall into this category, and as we have seen previously, graph kernels often successfully compare graphs by their respective components (S. Vishwanathan, Schraudolph, Kondor, and K. M. Borgwardt, 2010). Objects that decompose into collections of components have a very natural similarity measure associated with them - the total similarity of optimally matched components.

Consider two decomposable objects $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ such that $x_i, y_j \in \mathcal{Z}$ for all i, j, and a similarity measure between components $k : \mathcal{Z} \times \mathcal{Z} \longrightarrow \mathbb{R}$. Now, assuming $m > n$ let the similarity of X and Y, $K(X, Y)$, be the sum of similarities between components corresponding to an optimal matching p,

$$K(X, Y) = \max_{p \in S_m} \sum_{i=1}^{n} k(x_i, y_{p(i)}) \tag{3.3.1}$$

with S_m the set of permutations of m objects. Fröhlich, Wegner, Sieker, and Zell (2005) proposed using precisely this function, dubbed the *optimal assignment* (OA) kernel, as a similarity measure between composite objects. To apply this similarity function to pairs of graphs, we need to specify an appropriate decomposition, and a similarity measure on components k. Following the theme of this thesis, we let graphs G and G' be associated with geometric embeddings $U_G = \{\mathbf{u}_i\}_{i=1}^{n}$ and $U_{G'} = \{\mathbf{u}'_i\}_{i=1}^{m}$[1], and compare node embeddings by e.g. the cosine similarity.

$$K_U(G, G') = \max_{p \in S_m} \sum_{i=1}^{n} \mathbf{u}_i{}^\top \mathbf{u}'_{p(i)}$$

See figure 3.3.1 for a visualization of this idea. In Paper II, we use Cholesky decompositions of the adjacency matrix and graph Laplacian, the eigenvectors of the adjacency and Laplacian matrices, the incidence matrix and the Luz-Schrijver and Lovász embeddings as basis for matching, to construct similarity functions used in graph classification.

Unfortunately, Vert (2008) showed that optimal assignment does not always yield a valid (positive semidefinite) kernel, a result that makes the theory of kernel methods inapplicable in this case, including in particular kernel SVMs. While it is possible to use the matching function with SVMs anyway, as Vert remarks, it would be more assuring and satisfying if this was justified by a corresponding theory.

3.3.1 Learning with similarity functions

In spite of the negative result due to Vert (2008), we are interested in using optimal assignment for learning algorithms on graphs. Instead of applying kernel methods however, we build on the theory of learning using similarity functions, developed by Balcan, Blum, and Srebro (2008). They observe that kernels do not necessarily correspond to natural notions of similarity, and on the other hand, natural notions of similarity do not correspond

[1] We may view these embeddings as labels of the nodes, although that has little practical impact.

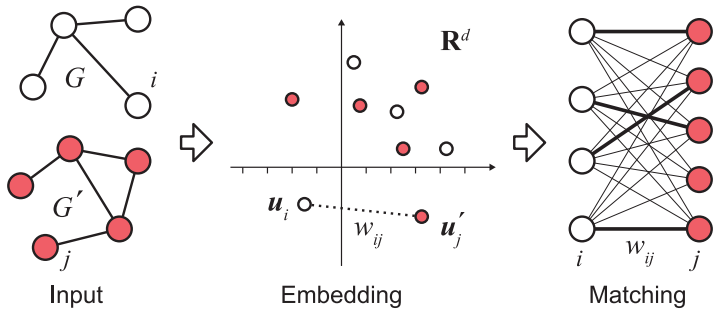


Figure 3.3.1: *Measuring similarity of graphs using matchings of geometric embeddings.*

to positive semidefinite kernels, and it might require significant additional work to coerce a similarity into a kernel, possibly also degrading the quality.

Consider a (binary) classification problem in which we observe samples (x, y), from a distribution P over $\mathcal{X} \times \{-1, +1\}$, and aim to learn the labelling function $y(x)$. In this setting, Balcan, Blum, and Srebro (2008) introduced the notion of *good* similarity functions. Goodness of a similarity function measures the margin between classes in a classification problem induced by the function. Specifically, for a similarity function $K : \mathcal{X} \times \mathcal{X} \longrightarrow [-1, 1]$, they define the following.

Definition 3.3.1. *A similarity function K is an (ϵ, γ)-good similarity function for a learning problem P if there exists a bounded weighting function w over $\mathcal{X}$, ($w(x) \in [0, 1]$ for all $x' \in \mathcal{X}$), such that at least a $1 - \epsilon$ mass of examples x satisfy:*

$$\mathbb{E}_{x' \sim P}[y(x)y(x')w(x')K(x, x')] \geq \gamma$$

Leaving aside the notion of a weighting function, this definition has the following intuitive interpretation: in expectation, points that are of the same class should be judged as similar ($K > \gamma$) by a good similarity function, and points of different classes should be judged as dissimilar ($K < -\gamma$). In Paper II, we expand on this idea by making a connection between optimal transport theory, the idea of similarity functions based on matchings, and the theory of good similarity functions due to Balcan, Blum, and Srebro (2008). Recall the definition of the transportation cost distance between distributions P and Q, with respect to a distance function d (Villani, 2003).

$$d(P, Q) := \min_{\pi(x,x')} \mathbb{E}_{\pi}[d(x, x')] \ ,$$

where the minimum is over all couplings of P and Q, i.e. over joint distributions π whose marginals are P and Q respectively. Observing that for any similarity function $K \in [-1, 1]$, we may form a dissimilarity function $\hat{K}(x, y) = (1 - K(x, y))/2 \in [0, 1]$, and corresponding transportation cost $\hat{K}(P, Q)$, we make the following definition.

Definition 3.3.2. *A similarity function K has margin γ with respect to a distribution P if the transportation cost $\hat{K}(P^i, P^j) \geq \gamma$ for $i \neq j$ where P^i is the distribution conditioned on the class with label i.*

This definition differs from that of Balcan, Blum, and Srebro (2008), in that it does not require a separate weighting function w(x). We go on to prove the following result about the learning capabilities of a good similarity function.

Theorem 3.3.1. *Let K be a similarity function with margin γ for a learning problem P with the measure concentration property. For any $\delta > 0$, let $S = \{x_1, \ldots, x_d\}$ be a sample of size $d = 8 \log 1/\gamma^2$ drawn independently at random from P. Consider the mapping $\phi^S : x \mapsto (K(x, x_i)/d)_{i \in [d]}$. With probability at least $1 - \delta$, the induced separator has margin at least $\gamma/2$.*

In Paper II, we show that for two well known graph classification problems, the OA similarity operating on the graph Laplacian embedding constitutes a *good* similarity function according to our definition. We evaluate empirically the performance of the OA similarity applied to seven different embeddings, including the Laplacian as well as the Lovász and spectral embeddings, on four different datasets. The OA kernel achieved state-of-the-art results for each dataset, both with and without inclusion of node labels.

3.4 Community detection with the Lovász embedding

In the previous parts of this thesis, the focus has rested firmly on supervised classification problems. In many real-world applications however, supervised learning is not possible, as there are no labeled data available. An important example of such problems is community detection, in which the goal is to discover meaningful groups of nodes in a graph, see section 1.2. In Paper I, we introduced a method for detecting communities in graphs based on the Lovász ϑ embedding and k-means clustering. By making use of geometric embeddings of graphs, we may convert the community detection problem to the more standard problem of clustering a set of points $\{\mathbf{u}_i\}_{i=1}^n \in \mathbb{R}^{d \times n}$, permitting the use of an arsenal of established techniques, such as k-means clustering. The Lovász and SVM-ϑ embeddings developed in previous chapter allow us to address two common problems with many existing clustering algorithms.

Problem 1: Number of clusters Many clustering algorithms rely on the user making a good choice of k, the number of clusters. As this choice can have dramatic effect on both the accuracy and speed of the algorithm, heuristics for choosing k, such as Pham, Dimov, and Nguyen (2005), have been proposed.

Problem 2: Initialization Popular clustering algorithms such as Lloyd's k-means, or expectation-maximization for Gaussian mixture models require an initial guess of the parameters. As a result, these algorithms are often run repeatedly with different random initializations.

In Paper I, we propose solutions to both problems based on our weighted generalization of Lovász number, $\vartheta^1(G)$. We begin by noting that $\vartheta^1(G)$ is a natural measure of group diversity in graphs. For complete graphs K_n, it is well known that $\vartheta(K_n) = 1$, and for empty graphs $\bar{K}_n$, $\vartheta(\bar{K}_n) = n$. We may interpret these graphs as having 1 and n clusters

respectively. Graphs with several disjoint clusters make a natural middle-ground. For a graph G that is a union of k disjoint cliques, $\vartheta(G) = k$. To solve Problem 1 therefore, we choose $k = \lceil \vartheta^1(G) \rceil$. We note that real graphs are typically not made up of complete subgraphs, but have both additional across-cluster edges and missing within-cluster edges. An interesting open problem is therefore to study the value of $\vartheta^1(G)$ as a function of this kind of noise.

The solution to problem 2 relies on the kernel characterization of $\vartheta^1(G)$. Recall from section 2.2 that,

$$\vartheta^1(G) = \min_{\kappa} \max_{\alpha_i} 2 \sum_{i=1}^{n} \alpha_i - \sum_{i,j=1}^{n} \kappa_{ij} \alpha_i \alpha_j \ .$$

When the support value α_i is non-zero, the induced geometric representation of node i is a support vector. Based on this observation, we propose a heuristic for initializing centroids in k-means: Initialize centroids by the set $\{\mathbf{u}_i\}_{i \in I}$ of node embeddings corresponding to the k nodes I that have the largest α_i.

We show in Paper I that a) community detection based on geometric embeddings of graphs is competitive with, and sometimes better than, special-purpose methods, b) our heuristics for initializing centroids and choosing k in k-means outperform other heuristics. The theoretical properties of these heuristics remain an open problem.

Chapter 4

Application – Entity disambiguation

Data mining applications increasingly deal with vast amounts of text data, often with references to entities such as people and companies. To enable efficient use of such data, it needs to be structured in a way that makes it accessible to both humans and algorithms. Annotating documents with the identities of people mentioned in the text is an example of information extraction of that kind. For instance, a user might be interested to know which cities TED talks curator Chris Anderson is visiting this year. An automated reply to such a query requires extraction of names and places from news texts, blogs or social media. This task is made difficult by the existence of Chris's namesakes: former Wired Magazine editor-in-chief Chris Anderson and basketball player Chris Anderson, among others. A naïve system considering only the names in isolation, essentially equating identity with identifier, would answer that all Chris's are the same person. In Paper IV, we show that this problem can be tackled using graph classification.

Resolving ambiguities such as the one above is called *entity disambiguation* or *entity resolution*, and is a problem which appears in many contexts. In its most general form, this problem is one of finding a mapping between a set of *identifiers* and a set of *entities*. In the example above, names are the identifiers and people are the entities. Related problems include record linkage (Fellegi and Sunter, 1969), deduplication (Culotta and McCallum, 2005), object distinction (Yin, Han, and Yu, 2007) and co-reference resolution (Haghighi and Klein, 2007). An subclass of entity disambiguation problems is *relational entity disambiguation* which makes use of graph structure between entities (Bhattacharya and Getoor, 2006; Bhattacharya and Getoor, 2004; Bekkerman and McCallum, 2005; Malin, 2005). Such information is available in many different contexts. In the example of documents such as news articles or blogs, entities are related through documents in which they are mentioned together. The resulting graph has nodes representing entities and edges representing documents in which they co-occur.

In this chapter and in Paper IV, we explore an important subproblem of relational entity disambiguation, namely that of determining which identifiers that are ambiguous, i.e. identifiers that are used to refer to several different entities. A successful solution

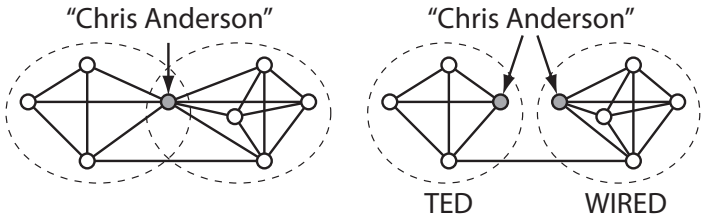


Figure 4.1.1: *A toy example of an identifier graph (left) and the corresponding entity graph (right). In this example, "Chris Anderson" is an ambiguous identifier.*

to this subproblem may be a valuable preprocessing step to otherwise computationally expensive disambiguation algorithms. We proceed to define this problem formally and give our approach to solving it using graph kernels. In Paper IV, we also propose extensions to existing graph kernels, tailored for the entity disambiguation task, and present an empirical evaluation showing that the proposed kernel extensions result in improved classification accuracy.

4.1 Entity disambiguation and graph classification

Henceforth, we let the term *entity* refer to a person or a company etc. while an *identifier* is a name or a label. If several entities have the same identifier, we say that the identifier is *ambiguous*. While a single entity may also have several identifiers, we assume that this is not the case here; we focus only on ambiguities. In the relational entity disambiguation setting, entities are assumed to be related according to some unknown graph structure, or *entity graph*. We assume that this graph can be partially observed through a graph of identifier relations, or *identifier graph*. Make sure to note the difference between these two graphs: the identifier graph contains ambiguities we wish to resolve while the entity graph does not. The difference is illustrated in figure 4.1.1.

Our running example is the setting in which identifiers are used in a corpus of documents. We let the identifier graph be the graph with one node for each identifier and an edge between every pair of identifiers co-occurring in at least one document. Edges are weighted by the significance of the relationships, such as number of co-occurrences. To provide mild anonymization of the data, we assume that the identifiers have been assigned in a pseudo-randomized fashion. In effect, the only information available to our method about the entities and their identifiers is the identifier graph.

We approach the following problem.

Definition 4.1.1 (Anonymized relational ambiguity detection)**.** *Given an undirected identifier graph $G = (V, E)$ with edge weights $w_{ij} \in \mathbb{R}^+$ and training data $S = \{(v_i, y_i) : 1 \leq i \leq m, v_i \in V, y_i \in \{\pm 1\}\}$ that labels certain nodes as ambiguous $(+)$ or unambiguous $(-)$,* anonymized relational ambiguity detection *is the task of classifying new nodes as $+1$ or -1. Each node of G may refer to a single entity or several underlying entities. The weight of an edge signifies the importance of the connection between two nodes.*

Algorithm 1 DETECTAMBIGUOUSNODES$(G = (V, E), \kappa, Y, S, T)$

Input: $G = (V, E)$
Input: $Y = \{y_i : i \in S \subset V, y_i \in \{\pm 1\}\}$ - Training labels
Input: $T \subset V$ - Test set
Input: κ - Neighborhood size.
for $v_i \in V$ **do**
  Set $G^{(i)} = \mathcal{N}_\kappa^{(i)}$ according to (4.1.1)
end for
Compute graph kernel matrix $K_{ij} = k(\mathcal{N}_\kappa^{(i)}, \mathcal{N}_\kappa^{(j)}),\ \forall i, j \in S$
Train an SVM with K and labels Y
Output: SVM classification of test nodes T.

Note that this definition does not include the actual separation of the entities that share identifiers. Nevertheless, this problem is of great importance, as pointing out which identifiers are ambiguous can represent very large computational savings for the more expensive task of resolving the ambiguities.

Consider figure 4.1.1 again, as it aims to illustrate some of the intuition behind our assumptions. To the left is an identifier graph and to the right the corresponding entity graph (assuming "Chris Anderson" is the only ambiguous identifier). In the figure, *two* individuals called Chris Anderson have been assigned only *one*, common identifier and thus *one* common node in the graph. This example shows how two otherwise only loosely connected communities (TED and Wired) can become strongly connected in the identifier graph through a single ambiguous node. In other words, it highlights our intuition that the graph structure surrounding "Chris Anderson" is indicative of whether the identifier is ambiguous or not. Although this example involves only people, we stress that nodes can represent any type of entity; an equally troublesome example would be that of the two *cities* Paris, France and Paris, Texas.

Motivated by our intuition that graph structure is indicative of ambiguity, we use graph classification to solve the ambiguity detection problem as stated in Definition 4.1.1. We let each node $v_i \in V$ be represented by its κ-neighborhood, $\mathcal{N}_\kappa^{(i)}$ as defined below.

Definition 4.1.2 (κ-neighborhood)**.** *Let $G = (V, E)$ be a graph. Then for any $v_i \in V$, the κ-neighborhood, $\mathcal{N}_\kappa^{(i)}$ is defined by*

$$\begin{aligned} V_\kappa^{(i)} &= \{v_i\} \cup \{v_j \in V : s(v_i, v_j) \leq \kappa\} \\ E_\kappa^{(i)} &= \{(v_p, v_q) : (v_p, v_q) \in E \wedge v_p, v_q \in V_\kappa^{(i)}\} \\ \mathcal{N}_\kappa^{(i)} &= (V_\kappa^{(i)}, E_\kappa^{(i)})\ . \end{aligned} \tag{4.1.1}$$

In the example of figure 4.1.1, the 1-neighborhood of the "Chris Anderson" node in the identifier graph is the entire graph. Each identifier is now represented by its neighborhood graph and a binary label indicating ambiguity: +1 for ambiguous and -1 for unambiguous. Given a training set consisting of an identifier graph G and a subset S nodes labeled by Y, our procedure is summarized in Algorithm 1.

Choice of graph kernel The choice of which kernel to use in Algorithm 1 can have a large impact on the outcome. In chapter 3 we introduced several classical graph kernels and two new ones developed as part of this thesis, all of which are applicable to the disambiguation problem. In Paper IV (see section 4) we make several improvements to existing graph kernels. The first extension specializes the random walk kernel (Gärtner, Flach, and Wrobel, 2003) to fit our problem. It is based on the observation that the graphs we are classifying are *pointed*; they have a distinguished node representing the identifier of interest, see figure 4.1.1. To this end, we consider a kernel variant counting only random walks originating from the distinguished node, not all possible walks. This observation could be applied to other kernels as well, such as subtree kernels (Shervashidze and K. Borgwardt, 2009) or the Lovász ϑ kernel defined in chapter 3.

For an empirical evaluation of our proposed method in the task of disambiguating identifiers in real-world datasets, see Paper IV, section 5. We show that our method is better than or competitive to state-of-the-art methods and that our kernel extensions increase classification accuracy. Unfortunately, this evaluation was made prior to the work on the Lovász ϑ kernel and its weighted extension, as well as the work on learning with matchings of geometric embeddings.

Chapter 5

Concluding remarks

This thesis has explored the idea of embedding graphs in geometric spaces in order to leverage powerful machine learning methods. In several instances, this has led to algorithms that are more general than special-purpose methods, but still achieve state-of-the-art results on important tasks. It is instructive to note that these results come from embedding algorithms that are agnostic to the learning problem at hand. While this fact speaks for the general applicability of geometric embeddings, it also suggests a natural next step: adapt the embedding to the learning problem. Learning appropriate embedding functions from data not only further reduces the reliance on hand crafted methods, it is also very much in line with the representation learning movement in machine learning of recent years. A strength of the methods developed in this thesis however, is that they can be used to theoretically bound performance on specific learning problems, a property typically not associated with representation learning. The marriage between data-driven methods and this type of analysis is an important open problem.

References

Alon, N. and J. Spencer (1992). *The Probabilistic Method*. Chichester: Wiley.

Axelsson, F., B. Rydback, F. Johansson, J. Bengtsson, and S. Marinov (2014). "Data-driven Coreference Resolution for Swedish". *Proceedings of the Swedish Language Technology Conference (SLTC). 2014*.

Balcan, M.-F., A. Blum, and N. Srebro (2008). A theory of learning with similarity functions. *Machine Learning* **72**.1-2, 89–112.

Bekkerman, R. and A. McCallum (2005). "Disambiguating Web appearances of people in a social network". *Proc. of WWW*.

Bhattacharya, I. and L. Getoor (2004). "Iterative record linkage for cleaning and integration". *Proc. of DMKD*.

— (2006). "Entity Resolutions in Graphs". *Mining Graph Data*. Wiley.

Blondel, V. D., J.-L. Guillaume, R. Lambiotte, and E. Lefebvre (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* **2008**.10, P10008.

Borgwardt, K. M. and H.-P. Kriegel (2005). "Shortest-path kernels on graphs". *Proceedings of ICDM*, pp. 74–81.

Brandes, U. et al. (2008). On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering* **20**.2, 172–188.

Chan, T.-H. H., K. L. Chang, and R. Raman (2009). "An SDP Primal-dual Algorithm for Approximating the Lovász-theta Function". *Proceedings of ISIT*. Coex, Seoul, Korea: IEEE Press, pp. 2808–2812.

Culotta, A. and A. McCallum (2005). "Joint deduplication of multiple record types in relational data". *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, pp. 257–258.

Debnath, A. K. et al. (1991). Structureactivity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry* **34** (2), 786–797.

Dukanovic, I. and F. Rendl (2008). A semidefinite programming-based heuristic for graph coloring. *Discrete Applied Mathematics* **156**.2, 180–189.

Feige, U. and R. Krauthgamer (2000). Finding and certifying a large hidden clique in a semirandom graph. *Random Structures & Algorithms* **16**.2, 195–208.

Fellegi, I. P. and A. B. Sunter (1969). A Theory for Record Linkage. *Journal of the American Statistical Association* **64**, 1183–1210.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports* **486**.3, 75–174.

Frieze, A. M. and M. Jerrum (1997). Improved Approximation Algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica* **18**.1, 67–81.

Fröhlich, H., J. K. Wegner, F. Sieker, and A. Zell (Aug. 2005). "Optimal assignment kernels for attributed molecular graphs". *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*. Ed. by L. de Raedt and S. Wrobel. Bonn, Germany: ACM Press, pp. 225–232.

Gärtner, T., P. Flach, and S. Wrobel (2003). On graph kernels: Hardness results and efficient alternatives. *Learning Theory and Kernel Machines*, 129–143.

Girvan, M. and M. E. Newman (2002). Community structure in social and biological networks. *Proceedings of the national academy of sciences* **99**.12, 7821–7826.

Goemans, M. X. (1997). Semidefinite programming in combinatorial optimization. *Math. Program.* **79**, 143–161.

Haghighi, A. and D. Klein (June 2007). "Unsupervised Coreference Resolution in a Nonparametric Bayesian Model". *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Haussler, D. (1999). *Convolution kernels on discrete structures*. Tech. rep. University of California at Santa Cruz.

Helma, C., R. D. King, S. Kramer, and A. Srinivasan (2001). The Predictive Toxicology Challenge 2000–2001. *Bioinformatics* **17**.1, 107–108. eprint: `http://bioinformatics.oxfordjournals.org/content/17/1/107.full.pdf+html`.

Hermansson, L., F. D. Johansson, and O. Watanabe (2015). "Generalized Shortest Path Kernel on Graphs". *Discovery Science*. Springer International Publishing, pp. 78–85.

Hermansson, L., T. Kerola, F. Johansson, V. Jethava, and D. Dubhashi (2013). "Entity disambiguation in anonymized graphs using graph kernels". *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. ACM, pp. 1037–1046.

Hush, D. R., P. Kelly, C. Scovel, and I. Steinwart (2006). QP Algorithms with Guaranteed Accuracy and Run Time for Support Vector Machines. *Journal of Machine Learning Research* **7**, 733–769.

Iyengar, G., D. J. Phillips, and C. Stein (2011). Approximating Semidefinite Packing Programs. *SIAM Journal on Optimization* **21**.1, 231–268.

Jethava, V., A. Martinsson, C. Bhattacharyya, and D. Dubhashi (2014). Lovasz theta function, SVMs and Finding Dense Subgraphs. *Journal of Machine Learning Research* **14**, 3495–3536.

Johansson, F. D., A. Chattoraj, C. Bhattacharyya, and D. Dubhashi (2015). "Weighted Theta Functions and Embeddings with Applications to Max-Cut, Clustering and Summarization". *Advances in Neural Information Processing Systems (NIPS)*, pp. 1018–1026.

Johansson, F. D. and D. Dubhashi (2015). "Learning with Similarity Functions on Graphs using Matchings of Geometric Embeddings". *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 467–476.

Johansson, F. D., O. Frost, C. Retzner, and D. Dubhashi (2015). "Classifying Large Graphs with Differential Privacy". *Modeling Decisions for Artificial Intelligence*. Springer, pp. 3–17.

Johansson, F. D., V. Jethava, D. Dubhashi, and C. Bhattacharyya (2014). "Global graph kernels using geometric embeddings". *Proceedings of the 31st International Conference on Machine Learning (ICML)*. Ed. by T. Jebara and E. P. Xing. JMLR Workshop and Conference Proceedings, pp. 694–702.

Johansson, F. D., U. Shalit, and D. Sontag (2016). "Learning Representations for Counterfactual Inference". *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. JMLR Workshop and Conference Proceedings.

Johansson, F., T. Färdig, V. Jethava, and S. Marinov (2012). "Intent-aware temporal query modeling for keyword suggestion." *PIKM*. Ed. by A. S. Varde and F. M. Suchanek. ACM, pp. 83–86.

Johansson, F., V. Jethava, and D. Dubhashi (2013). "DLOREAN: Dynamic LOcation-aware REconstruction of multiwAy Networks". *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*. IEEE, pp. 1012–1019.

Kågebäck, M., F. Johansson, R. Johansson, and D. Dubhashi (2015). Neural context embeddings for automatic discovery of word senses. *Proceedings of NAACL-HLT*, 25–32.

Karger, D. R., R. Motwani, and M. Sudan (1998). Approximate Graph Coloring by Semidefinite Programming. *J. ACM* **45**.2. Earlier version in FOCS'94, 246–265.

Kashima, H., K. Tsuda, and A. Inokuchi (2003). "Marginalized kernels between labeled graphs". *Proceedings of the 20th International Conference on Machine Learning*.

Knuth, D. E. (1993). *The sandwich theorem*. Stanford University, Department of Computer Science.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory* **28**.2, 129–137.

Lovász, L. (1979). On the Shannon capacity of a graph. *IEEE Transactions on Information Theory* **25**.1, 1–7.

Luz, C. J. and A. Schrijver (2005). A Convex Quadratic Characterization of the Lovász Theta Number. *SIAM J. Discrete Math.* **19**.2, 382–387.

Mahé, P. and J.-P. Vert (2009). Graph kernels based on tree patterns for molecules. *Machine Learning* **75**.1, 3–35.

Malin, B. (2005). "Unsupervised name disambiguation via social network similarity". *Workshop on link analysis, counterterrorism, and security*. Vol. 1401, pp. 93–102.

Newman, M. E. J. and M. Girvan (Feb. 2004). Finding and evaluating community structure in networks. *Phys. Rev. E* **69**.2, 026113. DOI: 10.1103/PhysRevE.69.026113. URL: http://link.aps.org/doi/10.1103/PhysRevE.69.026113.

Newman, M. E. (2004). Fast algorithm for detecting community structure in networks. *Physical review E* **69**.6, 066133.

Pham, D. T., S. S. Dimov, and C. Nguyen (2005). Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* **219**.1, 103–119.

Ramon, J. and T. Gärtner (2003). "Expressivity versus Efficiency of Graph Kernels". *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences at ECML/PKDD*. Ed. by L. D. Raedt and T. Washio, pp. 65–74.

Schölkopf, B., J. C. Platt, et al. (July 2001). Estimating the Support of a High-Dimensional Distribution. *Neural Computation* **13**.7. Ed. by and.

Schölkopf, B. and A. J. Smola (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press.

Schölkopf, B., K. Tsuda, and J.-P. Vert (2004). *Kernel methods in computational biology*. The MIT press.

Schrijver, A. (1979). A comparison of the Delsarte and Lovász bounds. *Information Theory, IEEE Transactions on* **25**.4, 425–429.

Shervashidze, N. and K. Borgwardt (2009). "Fast Subtree Kernels on Graphs". *Proceedings of NIPS*, pp. 1660–1668.

Shervashidze, N., P. Schweitzer, et al. (2011). Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research* **12**, 2539–2561.

Shervashidze, N., S. Vishwanathan, et al. (2009). "Efficient graphlet kernels for large graph comparison". *Proc. of AISTATS*.

Shin, K. and T. Kuboyama (2008). "A generalization of Haussler's convolution kernel: mapping kernel". *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 944–951.

Tahmasebi, N. et al. (2015). Visions and open challenges for a knowledge-based culturomics. *International Journal on Digital Libraries* **15**.2-4, 169–187.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

Vert, J.-P. (2008). The optimal assignment kernel is not positive definite. *arXiv preprint arXiv:0801.4061*.

Villani, C. (2003). *Topics in optimal transportation*. 58. American Mathematical Soc.

Vishwanathan, S. V. N., K. M. Borgwardt, and N. N. Schraudolph (2006). "Fast Computation of Graph Kernels." *NIPS*. Ed. by B. Schölkopf, J. Platt, and T. Hoffman. MIT Press, pp. 1449–1456. ISBN: 0-262-19568-2. URL: `http://dblp.uni-trier.de/db/conf/nips/nips2006.html#VishwanathanBS06`.

Vishwanathan, S., N. N. Schraudolph, R. Kondor, and K. M. Borgwardt (2010). Graph kernels. *JMLR*.

Wasserman, S. and K. Faust (1994). *Social network analysis: Methods and applications*. Cambridge Univ Pr.

West, D. B. et al. (2001). *Introduction to graph theory*. Vol. 2. Prentice hall Upper Saddle River.

Xenarios, I. et al. (2002). DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic acids research* **30**.1, 303–305.

Yin, X., J. Han, and P. Yu (2007). "Object distinction: Distinguishing objects with identical names". *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*.