Machine learning for network data

Proefschrift

ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen,
op gezag van de rector magnificus prof. dr. Th.L.M. Engelen,
volgens besluit van het college van decanen
in het openbaar te verdedigen op maandag 9 februari 2015
om 14:30 uur precies

door

Twan Mandela van Laarhoven

geboren op 1 februari 1985
te Balloo

Promotor:

Prof. dr. Tom Heskes

Copromotor:

Dr. Elena Marchiori

Manuscriptcommissie:

Prof. dr. Lutgarde Buydens (voorzitter)
Prof. dr. Marcel Reinders (TU Delft)
Dr. Jean-Philippe Vert (MINES ParisTech, Frankrijk)

SIKS Dissertation Series No. 2015-03

The research reported in this thesis has been carried out under the auspices of the Dutch Research School for Information and Knowledge Systems (SIKS), and the institute for Computing and Information Science (iCIS) of the Radboud University Nijmegen.

This research was supported by NWO grant Graphs for Multi Task Learning (612.066.927).

Copyright © 2015 Twan van Laarhoven

ISBN 978-94-6295-076-4
Printed by: Proefschriftmaken.nl || Uitgeverij BOXPress
Published by: Uitgeverij BOXPress, 's-Hertogenbosch

Contents

1 Introduction **1**

Part 1 Network clustering with quality functions **5**

2 What is network clustering? **7**
2.1 Applications of network clustering 8
2.2 Issues with clustering 9
2.3 Outline of this first part of the thesis 10

3 Axioms for hard clustering **13**
3.1 Introduction 13
3.1.1 Related work 15
3.2 Definitions and notation 16
3.3 On the form of axioms 17
3.4 Axioms for graph clustering quality functions 18
3.4.1 Locality 19
3.4.2 Continuity 21
3.4.3 Summary of axioms 22
3.5 Modularity 23
3.6 Adaptive scale modularity 25
3.6.1 Relation to other quality functions 26
3.6.2 Parameter dependence analysis 27
3.7 Conclusion and open questions 28
3.A Proof of Theorem 3.10, modularity is rich 31
3.B Proof of Theorem 3.15, adaptive scale modularity is rich 32
3.C Proof of Theorem 3.16, adaptive scale modularity is monotonic . . 36

4 Comparing quality functions for network clustering **37**

4.1 Introduction . . . 38
4.2 Quality functions . . . 40
4.2.1 Notation . . . 40
4.2.2 The considered quality functions . . . 40
4.2.3 Resolution biases . . . 42
4.2.4 Controlling the size of clusters . . . 45
4.3 The Louvain method . . . 46
4.4 Experiments . . . 47
4.4.1 Community detection benchmarks . . . 47
4.4.2 Quality functions versus optimization . . . 51
4.4.3 Real world community graphs . . . 53
4.4.4 Artificial nearest neighbor data . . . 54
4.4.5 Real world nearest neighbor datasets . . . 55
4.5 Conclusions . . . 56

5 Finding Protein Complexes, an application of network clustering 59
5.1 Introduction . . . 60
5.1.1 Related work . . . 61
5.2 Methods . . . 62
5.3 Experiments . . . 63
5.3.1 PPI networks . . . 63
5.3.2 Complex validation . . . 64
5.3.3 Precision vs. recall . . . 65
5.3.4 Networks from Single Studies . . . 67
5.3.5 Randomly perturbed graphs . . . 67
5.4 Discussion . . . 70
5.5 Conclusions . . . 71

6 Axioms for soft clustering and Non-negative Matrix Factorization 73
6.1 Introduction . . . 73
6.1.1 Related work . . . 75
6.1.2 Definitions and Notation . . . 75
6.2 Non-negative Matrix Factorization . . . 77
6.3 Resolution-limit-free functions for hard clustering . . . 78
6.4 Resolution-limit-free functions for soft clustering . . . 80
6.4.1 Locality . . . 81
6.4.2 Characterizing local quality functions . . . 83
6.4.3 Fixed number of clusters . . . 84
6.4.4 Varying number of clusters . . . 85
6.5 NMF as a probabilistic model . . . 86

6.5.1 A local prior . . . 88
6.5.2 Analysis of the quality functions on two types of graphs . . 90
6.6 Conclusion . . . 92
6.A Proof of Theorem 6.2 . . . 93
6.B Proof of Theorem 6.4 . . . 94
6.C Proof of Theorem 6.5 . . . 94
6.D Proof of Theorem 6.7, additive quality functions are local . . . 95

Part 2 Link prediction in biological bipartite graphs 97

7 The link prediction problem 99
7.1 Drug–target interaction data . . . 100
7.2 Evaluation . . . 102
7.3 Outline of this part of the thesis . . . 102

8 Interaction prediction with Gaussian Interaction Profiles 105
8.1 Introduction . . . 106
8.2 Gaussian Interaction Profile Kernel . . . 107
8.3 Integrating Chemical and Genomic Information . . . 108
8.4 The RLS-avg classifier . . . 108
8.5 RLS-Kron classifier . . . 109
8.6 Comparison methods . . . 110
8.7 Evaluation . . . 111
8.7.1 Analysis . . . 114
8.7.2 Kernels' relevance . . . 115
8.7.3 New predicted interactions . . . 116
8.7.4 Surprising interactions . . . 118
8.8 Discussion . . . 119

9 Interaction prediction for new drugs 123
9.1 Introduction . . . 123
9.2 Methods . . . 124
9.2.1 The GIP method . . . 124
9.2.2 Weighted nearest neighbor for unseen drug compounds . . 126
9.2.3 A method to show the bias of a LOOCV procedure . . . 126
9.3 Experiments . . . 127
9.3.1 Comparison with other methods . . . 128
9.4 Discussion . . . 129

10 Biases in drug–target interaction data 135

10.1 Introduction . . . 135
10.1.1 Related work . . . 137
10.2 Materials . . . 137
10.3 Validation procedures . . . 138
10.4 Biases . . . 139
10.5 Avoiding the bias . . . 141
10.6 Conclusions . . . 144

11 Discussion **147**
11.1 So, what is clustering? . . . 147
11.1.1 Robustness of clustering . . . 148
11.1.2 The relevance of locality . . . 148
11.2 Data and validation . . . 149
11.3 Algorithms . . . 150

Table of notations **151**

Bibliography **155**

Summary **169**

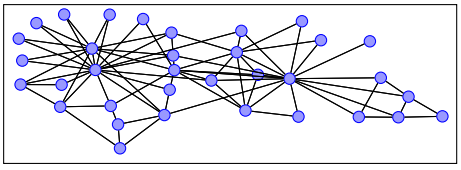
Samenvatting **171**

Acknowledgements **173**

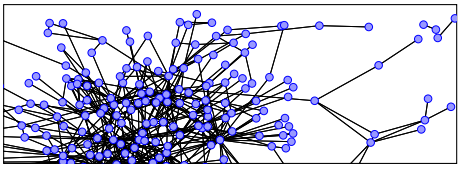
Chapter 1

Introduction

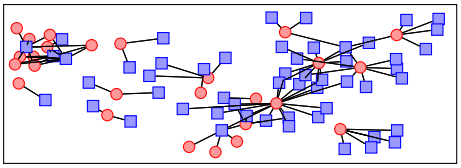
Networks are a convenient and flexible representation for many datasets. Network data often concerns interactions or relations. These can be relations between people, such as being friend; connections between neurons in the brain; or simply the notion of two things being similar to each other. But also the interactions between proteins, drugs, etc. can be considered as a network. Figure 1.1 gives an illustration of the different kinds of network datasets that are used in this thesis. Many more networks exist.



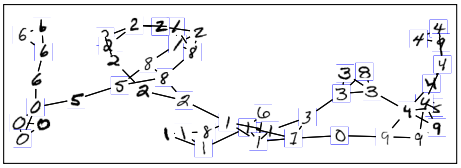
(a) Zachary's karate club. The nodes are people, and edges are associations.



(b) A subset of a protein–protein interaction network. Nodes are proteins, edges indicate physical interactions.



(c) A drug–target interaction network. Square nodes are drugs, circles are target proteins. Edges indicate interaction.



(d) A nearest neighbor network for a subset of the MNIST digit classification dataset. Nodes are digit images, and there is an edge between each image and its two nearest neighbors.

Figure 1.1: An illustration of different network datasets.

In general, a network (also called a graph), consists of a set of nodes and a set of connections between pairs of nodes. These connections, called edges, can be directed or undirected, and they may be associated with a weight. Sometimes there is additional structure, for example extra information about the individual nodes.

Given a network, there are several machine learning questions that can be asked. What sets these machine learning problems apart from traditional graph theory is that they involve some kind of inference. The goal is to look beyond the network at hand, and discover something about the underlying structure. Of course, this relies on the assumption that there is a certain underlying structure. If this assumption is wrong, then it may be impossible to say anything about the network. Additionally, we can never get certainty about the results, and we have to settle for very likely ones.

In this thesis we look at two different such machine learning problems on networks.

In Part 1 we look at finding clusters in networks. Clustering is a prototypical unsupervised machine learning problem. The goal is to split a dataset into clusters, where the elements of the clusters are similar in some sense. In our setting, the elements of the dataset are the nodes in a network, and similarity is defined by the edges of that network.

When performing clustering, the underlying assumption is that there actually is a cluster structure in the network. Many networks have a cluster structure at certain scales, but there are also networks without well-defined clusters.

We will use an approach based on the optimization of a quality function. We will not consider how to perform this optimization problem. Instead, this part of the thesis focuses on the more fundamental issue of choosing the right quality function to optimize in the first place.

In Part 2 we will look at predicting links in bipartite networks. A bipartite network is one where the nodes can be split into two sets, and the only edges are between nodes in different sets. Many biological networks take this form, in particular the interaction network of drugs and target proteins which we consider in this part of the thesis.

A peculiar aspect of this interaction network is that only positive interactions is known. That is, if there is an edge between two nodes, this means that they are know that to interact. But if there is no edge, then it does not mean that there is no interaction, instead, it can mean that this interaction has not been experimentally tested.

So, there might be many missing links in the network. The goal of link prediction is to infer the likely missing links, so that experimentalists know

which interactions they should test. This can be seen as a more traditional supervised machine learning problem, where task is to predict whether there is an edge between any two nodes, based on features of the nodes and of the potential edge. The training data are then the existing edges in the network in addition to side information about the nodes.

To be able to perform this link prediction we rely on the assumption that 'similar' nodes interact with the same or similar sets other nodes. In the case of drug–target interaction this is a sensible assumption. These interactions happen because a drug compound can attach to a protein, which depends on the three-dimensional shape of (part of) the drug molecule. And drugs with a similar chemical structure will likely have a similar shape.

Part 1

Network clustering with quality functions

Chapter 2

What is network clustering?

In this first part of the thesis we look at finding clusters in networks, and in particular at an approach based on quality functions.

Clustering is the task of finding clusters in a dataset. Informally, a cluster is a set of objects that are close to each other. In the network setting, we can interpret closeness by the connections: a cluster is a set of tightly connected nodes. [1] Clustering is an unsupervised machine learning problem. That means that no labels or other annotations are used. The only input is the network itself.

This problem goes by various names: graph partitioning, graph clustering, network clustering and community detection. The former names come from a computer science field, and focus on the abstract properties of graphs. On the other hand, the term community detection comes from the social sciences. In that field scientists have tried to find structure in the behavior of groups of people and social animals. People form communities, groups of people that all know each other, and chiefly interact among themselves.

Even when objects are described by a set of attributes, the clustering problem can be transformed into a graph clustering task. This is done by constructing a similarity graph, for instance the k-nearest neighbor graph (see e.g. Brito *et al.*, 1997; Franti, Virmajoki, and Hautamaki, 2006).

The observant reader may notice that we haven't yet defined what a 'cluster' actually is, so far a 'cluster' is an abstract thing. In general there are several different notions of cluster and therefore of clustering. The simplest is what we call *hard clustering*. In hard clustering the goal is graph partitioning. That is, a clustering of a graph is a partition of its nodes. The clusters in such a clustering

[1] This is not the only possible way to cluster networks. An alternative is to consider two nodes to be close if they connect to similar sets of other nodes. But we will not treat that interpretation in this work.

are non-overlapping sets nodes, and each node is in exactly one cluster.

The alternative is what we call *soft clustering*, where nodes can belong to more than one cluster, memberships can be fuzzy, and clusters can overlap. We will save the treatment of soft clustering until Chapter 6, and work in the hard clustering setting until then.

Even if we know what a clustering is, it doesn't yet tell us what a *good* clustering is. In contrast to classification, there is no clear definition of the quality of a clustering. One might define a *quality function*, which gives a score to any clustering of a graph. But this just shifts the problem to deciding what a good quality function is.

Several different such quality functions have been defined in the literature, and graph clustering is often performed by directly optimizing one of them. The 'optimal' clustering is then one that maximizes the quality.[2] Finding this optimal clustering is a computationally intractable discrete optimization problem (at least for the quality functions for which hardness is known, see for instance (Brandes *et al.*, 2008; Fortunato, 2010; Schaeffer, 2007)). Therefore all effective and scalable methods are based on heuristic and approximate techniques. We refer the reader to the surveys on this topic, e.g., Fortunato (2010), von Luxburg (2007), and Schaeffer (2007).

2.1 Applications of network clustering

As mentioned in the introduction, one natural application of network clustering is in finding communities in social networks. A classic example of such a clustering is Zachary's karate club (Zachary, 1977). In this network the nodes are the members of a karate club, and edges indicate their friendships. A conflict led to this club splitting in two, which we can take to be a clustering of the network.

Nowadays, online social networks exist with millions of nodes and orders of magnitude more edges. There has been a lot of interest from social scientists in analyzing these networks.

Similar to these social networks are the network of links on the world wide web, or the network of citations between scientific articles or authors. In these networks it is possible to find communities based on areas of common interest.

Another application area is in brain connectivity. Using Magnetic Resonance Imaging (MRI), more and more detailed brain connectivity networks are being developed. In these networks the nodes are small areas of the brain, typically the voxels observed by the MRI scanner. These voxels still contain many neurons,

[2]Throughout this thesis we will use maximization. Some objective functions are traditionally minimized, in those cases we negate them.

and so they have no real physical meaning. But at a slightly larger scale, tightly connected brain regions are thought to perform a common function. Network clustering can be used to find these regions.

At a smaller scale, there are biological interaction networks. In these networks the nodes are things like proteins or drugs, and edges indicate known interactions. In human cells, proteins can form complexes, which are clusters of different proteins that physically interact to perform a common function. It is hard to observe these complexes directly, so network clustering methods can be used to find them.

Besides these kinds of data that are naturally described as networks, there are many datasets that consist of tables of objects with attributes. Even in this case, the clustering problem can be transformed into a network clustering task. This is done by constructing a similarity graph, for instance the k-nearest neighbor graph (see e.g. Brito *et al.*, 1997; Franti, Virmajoki, and Hautamaki, 2006). As the name suggests, in this nearest neighbor network each node is connected to the k other nodes that are most similar to it, according to some similarity metric. An advantage of this network transformation compared to directly clustering the data, is that the similar metric only has to be accurate locally. This allows one to find clusters that have a very complicated shape in the original data space, as long as they are sufficiently separated.

2.2 Issues with clustering

A recurring problem for any kind of clustering is determining the number of clusters. For some clustering methods the number of clusters is an explicit parameter, usually called k[3]. When given a network, these methods will find k clusters, no more and no less.

Other methods can automatically determine the number of clusters, by selecting the clustering with the highest quality among all possible clusterings. It might seem like this neatly solves the problem of determining the number of clusters. But this is not necessarily true.

For instance, the 'modularity' quality function has a so called *resolution limit* (Fortunato and Barthélemy, 2007). Optimizing this quality function we will never be able to observe small communities, and as the total network becomes larger, so does the size of the smallest observable cluster. Other quality functions have a similar resolution bias, either towards large clusters or towards small ones.

[3]This is a different parameter from the k in the k-nearest neighbor graph.

Real world networks do not describe some grand unchanging truth about the world. Rather, they contain a sample taken at a specific time, based on noisy measurements. So there might be missing edges, as well as incorrect edges. For example, edges in a protein interactions are determined by a statistical significance threshold, interactions that fall just below this threshold are not included. New experiments can change the network. Or think of a social network, where who is friends with who can change all the time.

Additionally, in many cases the set of nodes in the network is just a subset of all possible nodes. For example, only the proteins measured so far are included in most interaction networks, and citation networks are often restricted to a certain research areas.

This uncertainty about networks has an important implication for network clustering. It means that methods should be robust, that is to say, not sensitive to these errors. Ideally, a small change to the network should result in only a small change to the clustering. Similarly, considering a larger or different subset of nodes, should not

2.3 Outline of this first part of the thesis

In **Chapter 3** we try to make it more precise what good graph clustering quality functions are. We do this by investigating properties that intuitively ought to be satisfied by any graph clustering quality function. Two axioms tailored for graph clustering quality functions are introduced, and the four axioms introduced in previous work on distance based clustering are reformulated and generalized for the graph setting. We show that modularity, a standard quality function for graph clustering, does not satisfy all of these six properties. This motivates the derivation of a new family of quality functions, adaptive scale modularity, which does satisfy the proposed axioms. Adaptive scale modularity has two parameters, which give greater flexibility in the kinds of clusterings that can be found. Standard graph clustering quality functions, such as normalized cut and unnormalized cut, are obtained as special cases of adaptive scale modularity.

Next we take a more experimental angle. There are not many real world datasets for which a ground truth clustering is know. But we can use a random model to generate graphs with a known clustering structure. Results of a recent comparative experimental assessment of methods applied to such benchmark graphs indicate that the two best methods use different quality functions, but a similar local search-based optimization procedure, called the Louvain method. This observation motivates the following research question: given the Louvain optimization procedure, how much does the choice of the quality function influence the results, and in what way?

In **Chapter 4** we address this question empirically in a broad graph clustering context, that is, when graphs are either given as such or are k-nearest neighbor graphs generated from a given (classification) dataset. We consider normalized cut, modularity, and infomap; as well as two new objective functions. We show that all these objective functions have a resolution bias, that is, they tend to prefer either small or large clusters. When removing this bias, by forcing the objective function to generate a given number of clusters, The Louvain method achieves similar performance across the considered objective functions on benchmark networks with built-in community structure. These results indicate that the resolution bias is the most important difference between objective functions in graph clustering when using the Louvain method.

In **Chapter 5** we apply these new methods to a concrete network clustering problem. In particular, we experimentally analyze the performance of the clustering algorithms when applied to protein-protein interaction (PPI) networks. We use publicly available yeast protein networks from past studies, as well as the present BioGRID database. Furthermore, to test robustness of the methods, various types of randomly perturbed networks obtained from the BioGRID data are also considered. Results of extensive experiments show improved or competitive performance over MCL, a state-of-the-art algorithm for complex detection in PPI networks, in particular on BioGRID data, where the w-log-v method introduced in Chapter 4 obtains excellent accuracy and robustness performance.

In **Chapter 6** we gradually shift our focus from hard clustering over to soft clustering. We will look at so called resolution-limit-free quality functions do not suffer from a resolution limit. This property was previously introduced for hard clustering, that is, graph partitioning. In this chapter we investigate the resolution-limit-free property in the context of Non-negative Matrix Factorization (NMF) for hard and soft graph clustering. To use NMF in the hard clustering setting, a common approach is to assign each node to its highest membership cluster. We show that in this case symmetric NMF is not resolution-limit-free, but that it becomes so when hardness constraints are used as part of the optimization. In soft clustering, nodes can belong to more than one cluster, with varying degrees of membership. In this setting resolution-limit-free turns out to be too strong a property. Therefore we propose *locality*, which states that changing one part of the graph does not affect the clustering of other parts of the graph. We argue that this is a desirable property, provide conditions under which NMF quality functions are local, and propose a novel class of local probabilistic NMF quality functions.

Chapter 3

Axioms for hard clustering

We investigate properties that intuitively ought to be satisfied by graph clustering quality functions, i.e. functions that assign a score to a clustering of a graph. Graph clustering, also known as network community detection, is often performed by optimizing such a function. Two axioms tailored for graph clustering quality functions are introduced, and the four axioms introduced in previous work on distance based clustering are reformulated and generalized for the graph setting. We show that modularity, a standard quality function for graph clustering, does not satisfy all of these six properties. This motivates the derivation of a new family of quality functions, adaptive scale modularity, which does satisfy the proposed axioms. Adaptive scale modularity has two parameters, which give greater flexibility in the kinds of clusterings that can be found. Standard graph clustering quality functions, such as normalized cut and unnormalized cut, are obtained as special cases of adaptive scale modularity.

In general, the results of our investigation indicate that the considered axiomatic framework covers existing 'good' quality functions for graph clustering, and can be used to derive an interesting new family of quality functions.

3.1 Introduction

Following the work by Kleinberg (2002) there have been various contributions to the theoretical foundation and analysis of clustering, such as axiomatic frameworks for quality functions (Ackerman and Ben-David, 2008), for criteria to compare clusterings (Meila, 2005), uniqueness theorems for specific types of clustering (Zadeh and Ben-David, 2009; Ackerman, Ben-David, and Loker, 2010a; Carlsson *et al.*, 2013), taxonomy of clustering paradigms (Ackerman, Ben-David,

This chapter is based on van Laarhoven and Marchiori (2014a), "Axioms for Graph Clustering Quality Functions", published in the Journal of Machine Learning Research.

and Loker, 2010b), and characterization of diversification systems (Gollapudi and Sharma, 2009).

Kleinberg focused on clustering functions, which are functions from a distance function to a clustering. He showed that there are no clustering functions that simultaneously satisfy three intuitive properties: scale invariance, consistency and richness. Ackerman and Ben-David (2008) continued on this work, and showed that the impossibility result does not apply when formulating these properties in terms of quality functions instead of clustering functions, where consistency is replaced with a weaker property called monotonicity.

Both of these previous works are formulated in terms of distance functions over a fixed domain. In this chapter we focus on weighted graphs, where the weight of an edge indicates the strength of a connection. The clustering problem on graphs is also known as network community detection.

Graphs provide additional freedoms over distance functions. In particular, it is possible for two points to be unrelated, indicated by a weight of 0. These zero-weight edges in turn make it natural to consider graphs over different sets of nodes as part of a larger graph. Secondly, we can allow for self loops. Self loops can indicate internal edges in a node. This notation is used for instance by Blondel *et al.* (2008), where a graph is contracted based on a fine-grained clustering.

In this setting, where edges with weight 0 are possible, Kleinberg's impossibility result does not apply. This can be seen by considering the connected components of a graph. This is a graph clustering function that satisfies all three of Kleinberg's axioms: scale invariance, consistency and richness (see Section 3.4.2).

Our focus is on the investigation of graph clustering quality functions, which are functions from a graph and a clustering to a real number 'quality'. A notable example is modularity (Newman and Girvan, 2004). In particular we ask which properties of quality functions intuitively ought to hold, and which are often assumed to hold when reasoning informally about graph clustering. Such properties might be called axioms for graph clustering.

The rest of this chapter is organized as follows: Section 3.2 gives basic definitions. Next, Section 3.3 discusses different ways in which properties could be formulated.

In Section 3.4 we propose an axiomatic framework that consists of six properties of graph clustering quality functions: the (adaption of) the four axioms from Kleinberg (2002) and Ackerman and Ben-David (2008) (permutation invariance, scale invariance, richness and monotonicity); and two additional properties specific for the graph setting (continuity and the weak locality).

Then, in Section 3.5, we show that modularity does not satisfy the monotonicity and weak locality properties.

This result motivates the analysis of variants of modularity, leading to the derivation of a new parametric quality function in Section 3.6, that satisfies all properties. This quality function, which we call adaptive scale modularity, has two parameters, M and γ which can be tuned to control the resolution of the clustering. We show that quality functions similar to normalized cut and unnormalized cut are obtained in the limit when M goes to zero and to infinity, respectively. Furthermore, setting γ to 0 yields a parametric quality function similar to that proposed by Reichardt and Bornholdt (2004).

3.1.1 Related work

Previous axiomatic studies of clustering quality functions have focused mainly on hierarchical clustering and on weakest and strongest link style quality functions (Kleinberg, 2002; Ackerman and Ben-David, 2008; Zadeh and Ben-David, 2009; Carlsson *et al.*, 2013). Papers in this line of work that focused also on the partitional setting include Puzicha, Hofmann, and Buhmann, 1999; Ackerman, Ben-David, Brânzei, *et al.*, 2012; Ackerman, Ben-David, Loker, and Sabato, 2013. Puzicha, Hofmann, and Buhmann (1999) investigated a particular class of clustering quality functions obtained by requiring the function to decompose into a certain additive form. Ackerman, Ben-David, Brânzei, *et al.* (2012) considered clustering in the weighted setting, in which every data point is assigned a real valued weight. They performed a theoretical analysis on the influence of weighted data on standard clustering algorithms. Ackerman, Ben-David, Loker, and Sabato (2013) analyzed robustness of clustering algorithms to the addition of a small set of points, and investigated the robustness of popular clustering methods.

All these studies are framed in terms of distance (or similarity and dissimilarity) functions.

Bubeck and von Luxburg (2009) studied statistical consistency of clustering methods. They introduced the so-called nearest neighbor clustering and showed its consistency also for standard graph based quality functions, such as normalized cut, ratio cut, and modularity. Here we do not focus on properties of methods to optimize clustering quality, but on natural properties that quality functions for graph clustering should satisfy.

Related works on graph clustering quality functions mainly focus on the so-called resolution limit, that is, the tendency of a quality function to prefer either small or large clusters. In particular, Fortunato and Barthélemy (2007) proved that modularity may not detect clusters smaller than a scale which depends on

the total size of the network and on the degree of interconnectedness of the clusters.

To mitigate the resolution limit phenomenon, the quality function may be extended with a so-called resolution parameter. For example, Reichardt and Bornholdt (2006) proposed a formulation of graph clustering (therein called network community detection) based on principles from statistical mechanics. This interpretation leads to the introduction of a family of quality functions with a parameter that allows to control the clustering resolution. In Section 3.6.1 we will show that this extension is a special case of adaptive scale modularity.

Traag, Van Dooren, and Nesterov (2011) formalized the notion of resolution-free quality functions, that is, not suffering from the resolution limit, and provided a characterization of this class of quality functions. Their notion is essentially an axiom, and we will discuss the relation to our axioms in Section 3.4.1.

3.2 Definitions and notation

Throughout this part of the thesis we will be talking about graphs. We will restrict ourselves to undirected graph, but we do allow weighted edges. Formally a (weighted) *graph* is a pair (V, A) of a finite set V of nodes and a function $A : V \times V \to \mathbb{R}_{\geqslant 0}$ of edge weights. For compactness we view A as an adjacency matrix, and write $a_{ij} = A(i, j)$. Edges with larger weights represent stronger connections, so $a_{ij} = 0$ means that there is no edge between nodes i and j. Note that this is the opposite of the convention used in distance based clustering. We explicitly allow for self loops, that is, nodes for which $a_{ii} > 0$.

A graph $G' = (V', A')$ is a *subgraph* of $G = (V, A)$ if $V' \subseteq V$ and $a'_{ij} = a_{ij}$ for all $i, j \in V'$. Conversely, given a set $V' \subseteq V$, the subgraph of $G = (V, A)$ *induced* by V is the subgraph of G who's set of nodes is V'. That is, (V', A'), where A' is the same as A restricted to the nodes in V'.

If $G' = (V', A')$ is a subgraph of $G = (V, A)$, then the *neighborhood* of G' is the set of nodes in V that have an edge to some node in V', together with the nodes in V. That is, $\text{neighborhood}(G', G) = V' \cup \{i \in V \mid \exists_{j \in V'}(a_{ij} > 0)\}$. We call the subgraph of G induced by the neighborhood of G' the *neighborhood graph* of G' in G.

If G' is a common subgraph of both G_1 and G_2, and the neighborhood graph of G' in G_1 is equal to the neighborhood graph of G' in G_2, then we say that G' is a common subgraph with *consistent neighborhood*.

A (hard) *clustering* C of a graph $G = (V, A)$ is a partition of its nodes. That is, $\bigcup C = V$ and for all $c_1, c_2 \in C$, $c_1 \cap c_2 \neq \emptyset$ if and only if $c_1 = c_2$. When two nodes i and j are in the same cluster in clustering C, i.e. when $i, j \in c$ for some $c \in C$, then we write $i \sim_C j$. Otherwise we write $i \not\sim_C j$.

A clustering C is a *refinement* of a clustering D, written $C \sqsubseteq D$, if for every cluster $c \in C$ there is a cluster $d \in D$ such that $c \subseteq d$.

A *graph clustering quality function* (or objective function) Q is a function from graphs G and clusterings of G to real numbers. We adopt the convention that a higher quality indicates a 'better' clustering. As a generalization, we will sometimes work with parameterized *families of quality functions*. A single quality function can be seen as a family with no parameters.

An edge between two nodes in the same cluster is called a *within cluster* edge, an edge between nodes in different clusters is a *between cluster* edge.

The *strength* of a node is the sum of weights of all edges incident to it, $s_i(G) = \sum_{j \in V} a_{ij}$. For unweighted graphs, the strength of a node is equal to its degree.

The *volume* of a cluster c is the sum of degrees of nodes in c, $v_c(G) = \sum_{i \in c} \sum_{j \in V} a_i j$. The total volume of a graph is $v_V(G)$. For an unweighted undirected graph this is equal to twice the number of edges.

The *within cluster weight* is the total weight of within cluster edges, $w_c(G) = \sum_{i,j \in c} a_i j$. For readability we leave the argument G in v and w implicit when it is clear from context.

3.3 On the form of axioms

There are three different ways to state potential axioms for clustering:

1. As a property of clustering functions, as in Kleinberg, 2002. For example, scale invariance of a clustering function $\hat{C}$ would be written as "$\hat{C}(G) = \hat{C}(\alpha G)$, for all graphs G, $\alpha > 0$". I.e. the optimal clustering is invariant under scaling of edge weights.

2. As a property of the values of a quality function Q, as in Ackerman and Ben-David (2008). For example "$Q(G, C) = Q(\alpha G, C)$, for all graphs G, all clustering C of G, and $\alpha > 0$". I.e. the quality is invariant under scaling of edge weights.

3. As a property of the relation between qualities of different clustering, or equivalently, as a property of an ordering of clusterings for a particular graph. For example "$Q(G, C) \geqslant Q(G, D) \Rightarrow Q(\alpha G, C) \geqslant Q(\alpha G, D)$".I.e. the 'better than' relation for clusterings is invariant under scaling of edge weights.

The third form is slightly more flexible than the other two. Any quality function that satisfies a property in the second style will also satisfy the corresponding property in the third style, but the converse is not true. Note also

that if D is not restricted in a property in the third style, then one can take $\hat{C}(G) = \operatorname{argmax}_C Q(G, C)$ to obtain a clustering function and an axiom in the first style.

Most properties are more easily stated and proved in the second, absolute, style. Therefore, we adopt the second style unless doing so requires us to make specific choices.

3.4 Axioms for graph clustering quality functions

Kleinberg defined three axioms for distance based clustering functions. In Ackerman and Ben-David (2008) the authors reformulated these into four axioms for clustering quality functions. These axioms can easily be adapted to the graph setting.

The first property that one expects for graph clustering is that the quality of a clustering depends only on the graph, that is, only on the weight of edges between nodes, not on the identity of nodes. We formalize this in the permutation invariance axiom,

Definition 3.1 (Permutation invariance)**.** *A graph clustering quality function* Q *is* permutation invariant *if for all graphs* $G = (V, A)$ *and all isomorphisms* $f : V \to V'$*, it is the case that* $Q(G, C) = Q(f(G), f(C))$*; where* f *is extended to graphs and clusterings by* $f(C) = \{\{f(i) \mid i \in c\} \mid c \in C\}$ *and* $f((V, A)) = (V', (i, j) \mapsto A(f^{-1}(i), f^{-1}(j)))$*.*

The second property, scale invariance, requires that the quality doesn't change when edge weights are scaled uniformly. This is an intuitive axiom when one thinks in terms of units: a graph with edges in "m/s" can be scaled to a graph with edges in "km/h". The quality should not be affected by such a transformation, perhaps up to a change in units.

Ackerman and Ben-David (2008) defined scale invariance by insisting that the quality stays equal when distances are scaled. In contrast, in Puzicha, Hofmann, and Buhmann (1999) the quality should scale proportional with the scaling of distances. We generalize both of these previous definitions by only considering the relations between the quality of two clusterings.

Definition 3.2 (Scale invariance)**.** *A graph clustering quality function* Q *is* scale invariant *if for all graphs* $G = (V, A)$*, all clusterings* C_1, C_2 *of* G *and all constants* $\alpha > 0$*,* $Q(G, C_1) \leqslant Q(G, C_2)$ *if and only if* $Q(\alpha G, C_1) \leqslant Q(\alpha G, C_2)$*. Where* $\alpha G = (V, (i, j) \mapsto \alpha A(i, j))$ *is a graph with edge weights scaled by a factor* α*.*

This formulation is flexible enough for single quality functions. However, families of quality functions could have parameters that are also scale depen-

dent. For such families we therefore propose to use as an axiom a more flexible property that also allows the parameters to be scaled,

Definition 3.3 (Scale invariant family)**.** *A family of quality function* Q_P *parameterized by* $P \in \mathcal{P}$ *is* scale invariant *if for all constants* $P \in \mathcal{P}$ *and* $\alpha > 0$ *there is a* $P' \in \mathcal{P}$ *such that for all graphs* $G = (V, A)$*, and all clusterings* C_1, C_2 *of* G*,* $Q_P(G, C_1) \leqslant Q_P(G, C_2)$ *if and only if* $Q_{P'}(\alpha G, C_1) \leqslant Q_{P'}(\alpha G, C_2)$*.*

Thirdly, we want to rule out trivial quality functions. This is done by requiring richness, i.e. that by changing the edge weights any clustering can be made optimal for that quality function.

Definition 3.4 (Richness)**.** *A graph clustering quality function* Q *is* rich *if for all sets* V *and all non-trivial partitions* C^* *of* V*, there is a graph* $G = (V, A)$ *such that* C^* *is the* Q*-optimal clustering of* V*, i.e.* $\operatorname{argmax}_C Q(G, C) = C^*$*.*

The last axiom that Ackerman and Ben-David consider is by far the most interesting. Intuitively, we expect that when the edges within a cluster are strengthened, or when edges between clusters are weakened, that this does not decrease the quality. Formally we call such a change of a graph a consistent improvement,

Definition 3.5 (Consistent improvement)**.** *Let* $G = (V, A)$ *be a graph and* C *a clustering of* G*. A graph* $G' = (V, A')$ *is a* C-consistent improvement of G *if for all nodes* i *and* j*,* $a'_{[}ij \geqslant a_i j$ *whenever* $i \sim_C j$ *and* $a'_{[}ij \leqslant a_i j$ *whenever* $i \not\sim_C j$*.*

We say that a quality function that does not decrease under consistent improvement is monotonic. In previous work this axiom is often called consistency.

Definition 3.6 (Monotonicity)**.** *A graph clustering quality function* Q *is* monotonic *if for all graphs* G*, all clusterings* C *of* G *and all* C*-consistent improvements* G' *of* G *it is the case that* $Q(G', C) \geqslant Q(G, C)$*.*

3.4.1 Locality

In the graph setting it also becomes natural to look at combining different graphs. With distance functions this is impossible, since it is not clear what the distance between nodes from the two different sets should be. But for graphs we can take the edge weight between nodes not in both graphs to be zero, in which case the two graphs are subgraphs of a larger graph with a consistent neighborhood.

Consider adding nodes to one side of a large network, then we would not want the clustering on the other side of the network to change if there is no

direct connection. For example, if a new protein is discovered in yeast, then the clustering of unrelated proteins in humans should remain the same. Similarly, we can consider any two graphs with disjoint node sets as one larger graph. Then the quality of clusterings of the two original graphs should relate directly to quality on the combined graph.

In general, local changes to a graph should have only local consequences to a clustering. Or in other words, the contribution of a single cluster to the total quality should only depend on nodes in the neighborhood of that cluster.

Definition 3.7 (Weak locality)**.** *A graph clustering quality function* Q *is* weakly local *if for all graphs* $G_1 = (V_1, A_1)$, $G_2 = (V_2, A_2)$, *and common subgraphs* $G_s = (V_s, A_s)$ *of* G_1 *and* G_2 *with a consistent neighborhood, and for all clusterings* C_c, D_c *of* V_s, C_1 *of* $V_1 \setminus V_s$ *and* C_2 *of* $V_2 \setminus V_s$, *if* $Q(G_1, C_c \cup C_1) \geqslant Q(G_1, D_c \cup C_1)$ *then* $Q(G_2, C_c \cup C_2) \geqslant Q(G_2, D_c \cup C_2)$.

Any quality function that has a preference for a fixed number of clusters will not be weakly local. On the other hand, a quality function that is written as a sum over clusters, where each summand depends only on properties of nodes and edges in one cluster and not on global properties, is weakly local.

Ackerman, Ben-David, and Loker (2010a) defined a similar locality property for clustering functions. Their definition differs from ours in three ways. First of all, they looked at k-clustering, where the number of clusters is given and fixed. Secondly, their locality property only implies a consistent clustering when the rest of the graph is removed, corresponding to $V_2 = V_1 \cap V_c$. They do not consider the other direction, where more nodes and edges are added. Finally, their locality property requires only a common subgraph G_s of both graphs, not that this subgraph has a consistent neighborhood. That means that clustering functions should also give the same results if edges with one endpoint in G_s are removed.

Relation to resolution-limit-free quality functions

Traag, Van Dooren, and Nesterov (2011) introduced the notion of *resolution-limit-free* quality functions, which is similar to weak locality. They then showed that resolution-limit-free quality functions do not suffer from the resolution limit as described by Fortunato and Barthélemy (2007). Their definition is as follows.

Definition 3.8 (Resolution-limit-free)**.** *Call a clustering* C *of a graph* G Q*-optimal if for all clustering* C' *of* G *we have that* $Q(G, C) \geqslant Q(G, C')$. *Let* C *be a* Q*-optimal clustering of a graph* G_1. *Then the quality function* Q *is called resolution-limit-free if for each subgraph* G_2 *induced by* $D \subset C$, *the partition* D *is also* Q*-optimal.*

There are three differences compared to our weak locality property. First of all, Definition 3.8 refers only to the optimal clustering, not to the quality, i.e. it is a property in the style of Kleinberg. Secondly, weak locality does not require that G_2 be a subgraph of G_1. Weak locality is stronger in that sense. Thirdly, and perhaps most importantly, in the subgraph G_2 induced by $D \subset C$, edges from a node in D to nodes not in D will be removed. That means that while the induced subgraph is a subgraph of both G_1 and G_2, it does not necessarily have a consistent neighborhood. So in this sense weak locality is weaker than resolution-limit-freedom.

The notion of resolution-limit-free quality functions was born out of the need to avoid the resolution limit of graph clustering. And indeed weak locality is not enough to guarantee that a quality function is free from this resolution limit.

We could look at a stronger version of locality, which does not require that the subgraph G_s has a consistent neighborhood in G_1 and G_2. Such a *strong locality* property would imply resolution-limit-freeness. However, it is a very strong property in that it rules out many sensible quality functions. In particular, a strongly local quality function can not depend on the weight of edges entering or leaving a cluster, because that weight can be different in another graph containing the subgraph induced by that cluster.

The solution used by Traag, Van Dooren, and Nesterov is to use the number of nodes instead of the volume of a cluster. In this way they obtain a resolution-limit-free variant of the Potts model by Reichardt and Bornholdt (2004), which they call the constant Potts model. But this comes at the cost of scale invariance.

3.4.2 Continuity

In the context of graphs, perhaps the most intuitive clustering function is finding the connected components of a graph. As a quality function, we could write

$$Q_{\text{coco}}(G, C) = \mathbf{1}[C = \hat{C}_{\text{coco}}(G)],$$

where the function $\hat{C}_{\text{coco}}$ yields the connected components of a graph, and $\mathbf{1}[x]$ is the indicator function that is 1 when x holds, and 0 otherwise.

This quality function is clearly permutation invariant, scale invariant, rich, and weakly local. Since a consistent change can only remove edges between clusters and add edges within clusters, the coco quality function is also monotonic.

In fact, all of Kleinberg's axioms (reformulated in terms of graphs) also hold for $\hat{C}_{\text{coco}}$, which seems to refute their impossibility result. However, the impossibility proof can not be directly transfered to graphs, because it involves a multiplication and division by a maximum distance. In the graph setting this

would be multiplication and division by a minimum edge weight, which can be zero.

Still, despite connected components satisfying all previously defined properties (except for strong locality), it is not a very useful quality function. In many real-world graphs, most nodes are part of one giant connected component (Bollobás, 2001). We would also like the clustering to be influenced by the weight of edges, not just by their existence. A natural way to rule out such degenerate quality functions is to require continuity.

Definition 3.9 (Continuity)**.** *A quality function* Q *is* continuous *if a small change in the graph leads to a small change in the quality. Formally,* Q *is continuous if for every* $\epsilon > 0$ *and every graph* $G = (V, A)$ *there exists a* $\delta > 0$ *such that for all graphs* $G' = (V, A')$*, if* $a_{ij} - \delta < a'_{ij} < a_{ij} + \delta$ *for all nodes* i *and* j*, then* $Q(G', C) - \epsilon < Q(G, C) < Q(G', C) + \epsilon$ *for all clusterings* C *of* G*.*

Connected components clustering is not continuous, because adding an edge with a small weight δ between clusters changes the connected components, and hence dramatically changes the quality.

Continuous quality functions have an important property in practice, in that they provide a degree of robustness to noise. A clustering that is optimal with regard to a continuous quality function will still be close to optimal after a small change to the graph.

3.4.3 Summary of axioms

We propose to consider the following six properties as axioms for graph clustering quality functions,

1. Permutation invariance (Definition 3.1),
2. Scale invariance (Definition 3.2),
3. Richness (Definition 3.4),
4. Monotonicity (Definition 3.6),
5. Weak locality (Definition 3.7), and
6. Continuity (Definition 3.9).

As mentioned previously, for families of quality functions we replace scale invariance by scale invariance for families (Definition 3.3).

In the next section we will show that this set of axioms is consistent by defining a quality function and a family of quality functions that satisfies all

of them. Additionally, the fact that there are quality functions that satisfy only some of the axioms shows that they are (at least partially) independent.

3.5 Modularity

For graph clustering one of the most popular quality functions is modularity (Newman and Girvan, 2004), despite its limitations (Good, de Montjoye, and Clauset, 2010; Traag, Van Dooren, and Nesterov, 2011),

$$Q_{\text{modularity}}(G, C) = \sum_{c \in C} \left(\frac{w_c}{w_V} - \left(\frac{v_c}{v_V} \right)^2 \right). \tag{3.1}$$

It is easy to see that modularity is permutation invariant, scale invariant and continuous.

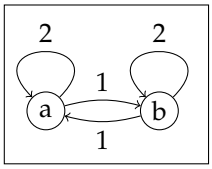
Theorem 3.10. *Modularity is rich.*

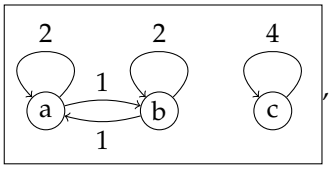
The proof of Theorem 3.10 is in Appendix 3.A.

An important aspect of modularity is that volume and within weight are normalized with respect to the total volume of the graph. This ensures that the quality function is scale invariant, but it also means that the quality can change in unexpected ways when the total volume of the graph changes. This leads us to Theorem 3.11.

Theorem 3.11. *Modularity is not weakly local.*

Proof. Consider the graphs

$G_1 =$ 

$G_2 =$ 

,

which have a common subgraph G_s with nodes $V_s = \{a, b\}$ that has a consistent neighborhood in G_1 and G_2. Note that we draw the graphs as directed graphs, to make it clear that each undirected edge is counted twice for the purposes of volume and within cluster weight. Now take the clusterings $C_s = \{\{a\},\{b\}\}$ and $D_s = \{\{a, b\}\}$ of V_s; $C_1 = \{\}$ of $V_1 \setminus V_s$; and $C_2 = \{\{c\}\}$ of $V_2 \setminus V_s$. Then

$$Q_{\text{modularity}}(G_1, C_s \cup C_1) = 1/6 > 0 = Q_{\text{modularity}}(G_1, D_a \cup C_1),$$

while

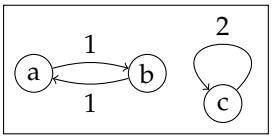
$$Q_{\text{modularity}}(G_2, C_s \cup C_2) = 23/50 < 24/50 = Q_{\text{modularity}}(G_2, D_a \cup C_2).$$

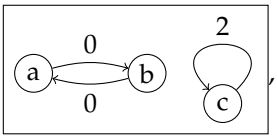
This counterexample shows that modularity is not weakly local. □

Even without changing the node set, changes in the total volume can be problematic, as shown by the following theorem.

Theorem 3.12. *Modularity is not monotonic.*

Proof. Consider the graphs

$G =$ 

$G' =$ 

,

and the clustering $C = \{\{a\},\{b\},\{c\}\}$. G' is a C-consistent improvement of G, because the weight of a between-cluster edge is decreased. The modularity of C in G is $Q_{\text{modularity}}(G, C) = 1/8$, while the modularity of C in G' is $Q_{\text{modularity}}(G', C) = 0$. So modularity can decrease with a consistent change of a graph, and hence it is not a monotonic quality function. □

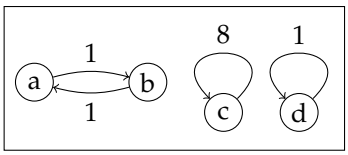
Monotonicity might be too strong a condition. When the goal is to find a clustering of a single graph, we are not actually interested in the absolute value of a quality function. Rather, what is of interest is the optimal clustering, and which changes to the graph preserve this optimum. At a smaller scaler, we can look at the relation between two clusterings. If C is better then D on a graph G, then on what other graphs is C better then D?

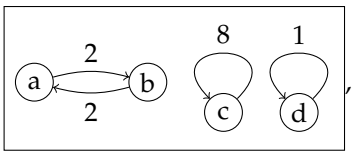
We therefore define a relative version of monotonicity, in the hopes that modularity does satisfy this weaker version.

Definition 3.13 (Relative monotonicity)**.** *A quality function* Q *is* relatively monotonic *if for all graphs* G *and* G' *and clusterings* C *and* D*, if* G' *is a* C*-consistent improvement of* G *and* G *is a* D*-consistent improvement of* G' *and* $Q(G, C) \geqslant Q(G, D)$ *then* $Q(G', C) \geqslant Q(G', D)$*.*

Theorem 3.14. *Modularity is not relatively monotonic.*

Proof. Take the graphs

$G =$ 

$G' =$ 

,

and the clusterings $C = \{\{a, b, c\},\{d\}\}$ and $D = \{\{a\},\{b\},\{c, d\}\}$. G' is a C-consistent improvement of G, because the weight of a within cluster edge is increased. G is a D-consistent improvement of G', because the weight of a between cluster edge

is decreased. However $Q_{\text{modularity}}(G, C) = 20/121 > 16/121 = Q_{\text{modularity}}(G, D)$ while $Q_{\text{modularity}}(G', C) = 24/169 < 28/121 = Q_{\text{modularity}}(G', D)$. This counterexample shows that modularity is not relatively monotonic. □

3.6 Adaptive scale modularity

The problems with modularity stem from the fact that the total volume can change when changes are made to the graph. It is therefore natural to look at a variant of modularity where the total volume is replaced by a constant M,

$$Q_{\text{M-fixed}}(G, C) = \sum_{c \in C} \left(\frac{w_c}{M} - \left(\frac{v_c}{M} \right)^2 \right).$$

This quality function is obviously weakly local. It is also a scale invariant family parameterized by M. However, this fixed scale modularity quality function is *not* scale invariant for any fixed scale $M > 0$.

We might hope that fixed scale modularity would be monotonic, because it doesn't suffer from the problem where changes in the edge weights affect the total volume. Unfortunately, fixed scale modularity has problems when the volume of a cluster starts to exceed $M/2$. In that case, increasing the weight of within cluster edges starts to decrease the fixed scale modularity. Looking at a cluster c with volume $v_c = w_c + b_c$,

$$\frac{\partial Q_{\text{M-fixed}}(G, C)}{\partial w_c} = \frac{1}{M} - \frac{2v_c}{M^2}. \tag{3.2}$$

This derivative is negative when $2v_c > M$, so in that case increasing the weight of a within-cluster edge will decrease the quality. Hence fixed scale modularity is not monotonic.

The above argument also suggests a possible solution: add $2v_c$ to the normalization factor M. Or more generally, add γv_c with $\gamma \geqslant 2$, which leads to the quality function

$$Q_{M,\gamma}(G, C) = \sum_{c \in C} \left(\frac{w_c}{M + \gamma v_c} - \left(\frac{v_c}{M + \gamma v_c} \right)^2 \right). \tag{3.3}$$

This *adaptive scale modularity* quality function is clearly still permutation invariant, continuous and weakly local. For $M = 0$ it is also scale invariant. Since the value of M should scale along with the edge weights, adaptive scale modularity is a scale invariant family parameterized by M. Additionally, we have the following two theorems:

Theorem 3.15. *Adaptive scale modularity is rich for all* $M \geqslant 0$ *and* $\gamma \geqslant 1$.

Theorem 3.16. *Adaptive scale modularity is monotonic for all* $M \geqslant 0$ *and* $\gamma \geqslant 2$.

The proofs of these theorems can be found in Appendices 3.B and 3.C.

This shows that adaptive scale modularity satisfies all six axioms we have defined for families of graph clustering quality functions, and the six axioms for single quality functions when $M = 0$. This shows that our extended set of axioms is consistent.

3.6.1 Relation to other quality functions

Interestingly, in the limit as M goes to 0, the adaptive-scale quality function becomes similar to normalized cut (Shi and Malik, 2000) with an added constant,

$$Q_{0,\gamma}(G, C) = \frac{1}{\gamma} \sum_{c \in C} \Big(\frac{w_c}{v_c} - \frac{1}{\gamma} \Big).$$

This 0-adaptive modularity is also scale invariant as a single quality function.

Conversely, when M goes to infinity the quality goes to 0. However, the quality function approaches unnormalized cut in behavior:

$$\lim_{M \to \infty} M \cdot Q_{M,\gamma}(G, C) = \sum_{c \in C} w_c.$$

This expression is similar to the Constant Potts model (CPM) by Traag, Van Dooren, and Nesterov (2011),

$$Q_{\text{cpm}}(G, C) = \sum_{c \in C} \big(w_c - \gamma n_c^2 \big). \tag{3.4}$$

In contrast to the quality functions discussed thus far, CPM uses the number of nodes instead of volume to control the size of clusters. Like adaptive scale modularity, the constant Potts model satisfies all six axioms (as a family).

As stated before, the fixed scale and adaptive scale modularity quality functions are a scale invariant family; they are not scale invariant for a fixed value of M (except for $M = 0$). This is not a large problem in practice, since scale invariance is often sacrificed to overcome the resolution limit of modularity (Fortunato and Barthélemy, 2007). In fact, fixed scale modularity is proportional to the quality function introduced by Reichardt and Bornholdt (2004),

$$Q_{\text{RB}}(G, C) = \sum_{c \in C} \Big(w_c - \gamma_{\text{RB}} \frac{v_c^2}{v_V} \Big) = M \cdot Q_{\text{M-fixed}}(G, C),$$

with $M = v_V / \gamma_{\text{RB}}$.

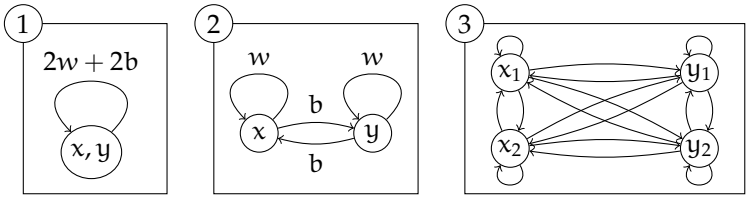


Figure 3.1: An illustration of the possible outcomes when clustering a two-clique network. Clusters are indicated by circles. In outcome (3), the vertical edges each have weight $w/4$, while the horizontal and diagonal ones have weight $b/4$.

3.6.2 Parameter dependence analysis

There has been a lot of interest in the so called resolution limit of modularity.

This problem can be illustrated with a simple graph that consists of a ring of cliques, where each clique is connected to the next one with a single edge. We would like the clusters in the optimal clustering to correspond to the cliques in the ring. It was observed by Fortunato and Barthélemy (2007) that, as the number of cliques in the ring increases, at some point the clustering with the highest modularity will have multiple cliques per cluster.

This resolution problem stems from the fact that the behavior of modularity depends on the total volume of the graph. Both the fixed scale and adaptive scale modularity quality functions instead have a parameter M, and hence do not suffer from this problem. In fact, any weakly local quality function will not have a resolution limit in the sense of Fortunato and Barthélemy. A similar observation was made by Traag, Van Dooren, and Nesterov (2011) in the context of modularity like quality functions.

In real situations graphs are not uniform as in the ring-of-cliques model. But we can still take simple uniform problems as a building block for larger and more complex graphs, since for weakly local quality functions the rest of the network doesn't matter beyond the immediate neighborhood. Therefore we will look at a simple problem with two subgraphs of varying sizes connected by a varying number of edges. More precisely, we take two cliques each with within weight w, connected by edges with weight b. The total volume of this (sub)graph is then $2w + 2b$.

There are three possible outcomes when clustering such a two-clique network: (1) the optimal solution has a single cluster; (2) the optimal solution has two clusters, corresponding to the two cliques; (3) the optimal solution has more than two clusters, splitting the cliques apart. See Figure 3.1 for an illustration. Which of these outcomes is desirable depends on the circumstances.

Another heterogeneous resolution limit model was proposed by Lancichinetti and Fortunato (2011). In this situation there are two cliques of equal size connected by a single edge, and a random subgraph. Now the ideal solution would be to find three clusters, one for each clique and one for the random subgraph. The optimal split of the random subgraph will roughly cut it in half, with a fixed fraction of the volume being between the two clusters (Reichardt and Bornholdt, 2007). So this model can be considered as a combination of two instances of our simpler problem, one for the two cliques and one for the random subgraph[1]. Hence, we want outcome (2) for the cliques, and outcome (1) for the random subgraph.

In Figure 3.2 we show which graphs give which outcomes for adaptive scale modularity with various parameter settings. The first column, $\gamma = 0$, is of particular interest, since it corresponds to fixed scale modularity and hence also to Q_{RB} and to modularity in certain graphs. In the third row we can see that when $2v = 2w + 2b > M = 100$ the cliques are split apart. This is precisely the region in which monotonicity no longer holds. Overall, the parameter M has the effect of determining the scale; each row in this figure is merely the previous row magnified by a factor 10. Increasing M has the effect of merging small clusters. On the other hand, the γ parameter controls the slope of the boundary between outcomes (1) and (2), i.e. the fraction of edges that should be within a cluster. This is most clearly seen when $M = 0$, while otherwise the effect of M dominates for small clusters.

3.7 Conclusion and open questions

In this chapter we presented an axiomatic framework for graph clustering quality functions consisting of six properties. We showed that modularity does not satisfy the monotonicity property. This motivated the derivation of a new family of quality functions, adaptive scale modularity, that satisfies all properties and has standard graph clustering quality functions as special cases. Results of an experimental parameter dependence analysis showed the high flexibility of adaptive scale modularity. However, adaptive scale modularity should not be considered the solution to all the problems of modularity, but rather an example of how axioms can be used in practice.

An overview of the discussed axioms and quality functions can be found in Table 3.1. Many more quality functions have been proposed in the literature, so this list is by no means exhaustive. An interesting topic for future research is to

[1] Lancichinetti and Fortunato include edges between the cliques and the random subgraph to ensure that the entire network is connected, these edges are not relevant to the problem

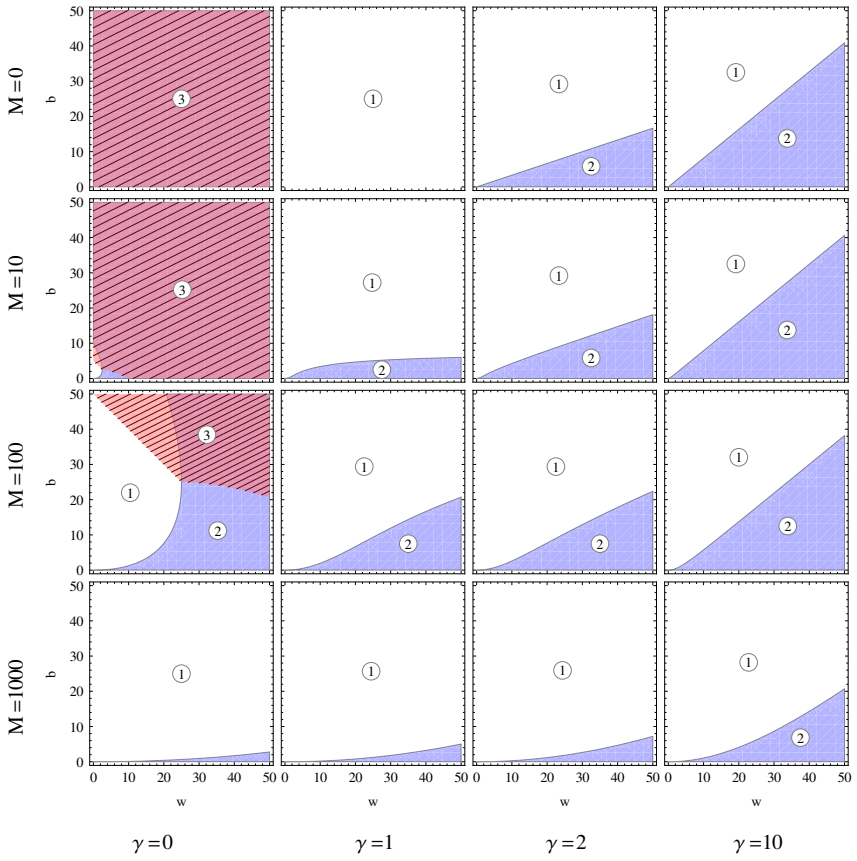


Figure 3.2: The behavior of $Q_{M,\gamma}$ for varying parameter values. The graph consists of two subgraphs with w internal weight each, connected by an edge with weigh b. Hence the volume of the total graph is $2w + 2b$. In region (1) the optimal clustering has a single cluster, In region (2) (light blue) the optimal clustering separates the subgraphs. In region (3) (red, hatched) the subgraphs themselves will be split apart.

make a survey of which existing quality functions satisfy which of the proposed properties.

We also investigated resolution-limit-free quality functions as defined in (Traag, Van Dooren, and Nesterov, 2011). As illustrated in Section 3.6.2, adaptive scale modularity allows to perform clustering at various resolutions, by varying the values of its two parameters. However it is not resolution-limit-free.

This chapter did not address questions such as finding a best quality function

	Permutation invariance	Scale invariance	Scale invariance (family)	Richness	Monotonicity	Weak locality	Continuity
Connected components	✓	✓	n.a.	✓	✓	✓	–
Modularity	✓	✓	n.a.	✓	–	–	✓
Reichardt and Bornholdt (2004)	✓	✓	✓	✓	–	–	✓
Fixed scale modularity	✓	$M = 0$	✓	✓	–	✓	✓
Adaptive scale modularity	✓	$M = 0$	✓	$\gamma \geqslant 1$	$\gamma \geqslant 2$	✓	✓
Constant Potts Model	✓	–	✓	$\gamma > 0$	✓	✓	✓
Normalized cut	✓	✓	n.a.	–	✓	✓	✓

Table 3.1: Overview of quality functions discussed in this chapter and the properties they satisfy.

(Almeida *et al.*, 2011), or selecting a significant resolution scale (Traag, Krings, and Dooren, 2013). The aim was to provide necessary conditions about what a good quality function is, in order to rule out and/or to improve quality functions. The proposed axioms and the introduction of adaptive scale modularity are an effort in this direction.

We also did not address the question of finding a clustering with the highest quality. Finding the optimal value of quality functions such as modularity is NP-hard (Brandes *et al.*, 2008), but several heuristic and approximation algorithms have been developed. One class of algorithms uses a divisive approach, see for instance Newman (2006) and Ruan and Zhang (2008). For such a tactic to be valid, an optimal or close to optimal clustering of a subgraph should also be a near optimal clustering of the entire graph. This is ensured by weak locality. Recently Dinh and Thai (2013) proposed polynomial-time approximation algorithms for the modularity maximization in the context of scale free networks. It would be interesting to investigate the suitability of these algorithms for adaptive scale modularity maximization.

In this work we have only looked at non-negative weights, undirected graphs, and only at hard partitioning. An extension to graphs with negative weights, to directed graphs and to overlapping clusters remains to be investigated. Another open problem is how to use these axioms for reasoning about quality functions

and clustering algorithms.

3.A Proof of Theorem 3.10, modularity is rich

The proofs of richness rely on clique graphs,

Definition 3.17 (Clique graph)**.** *Let* V *be a set of nodes,* C *be a partition of* V*, and* k *be a positive constant. The* clique graph of C with edge weight k *is defined as* $G = (V, A)$ *where* $a_{ij} = k$ *if* $i \sim_C j$ *and* $a_{ij} = 0$ *otherwise.*

Proof. Let V be a set of nodes and $C \neq \{V\}$ be a clustering of V. Let $G = (V, A)$ be a clique graph of C with edge weight 1. Note that $a_{ii} = 1$, so any possible cluster will have a positive volume. Let D be a clustering of G with maximal modularity.

Suppose that there is a cluster $d \in D$ that contains $i, j \in d$ with $i \not\sim_C j$. Then we can split the cluster into $d_1 = \{k \in d \mid k \sim_C i\}$ and $d_2 = \{k \in d \mid k \not\sim_C i\}$. Because there are no edges between nodes in d_1 and nodes in d_2, it is the case that $w_d = w_{d_1} + w_{d_2}$. Both d_1 and d_2 are non-empty and have a positive volume, so $v_d^2 = (v_{d_1} + v_{d_2})^2 < v_{d_1}^2 + v_{d_2}^2$. Therefore $Q_{\text{modularity}}(G, D) < Q_{\text{modularity}}(G, D \setminus \{d\} \cup \{d_1, d_2\})$. So D does not have maximal modularity, which is a contradiction.

Suppose, on the other hand that all clusters $d \in D$ are a subset of some cluster in C, i.e. D is a refinement of C. Then either $D = C$, or there are two clusters $d_1, d_2 \in D$ that are both a subset of the same cluster $c \in C$. In the latter case we can combine the two clusters into $d = d_1 \cup d_2$. The within weight of this combined cluster is $w_d = |d|^2 = w_{d_1} + w_{d_2} + 2|d_1||d_2|$. The squared volume of the combined cluster is $v_d^2 = |d|^2|c|^2 = v_{d_1}^2 + v_{d_2}^2 + 2|d_1||d_2||c|^2$. So this changes increases the modularity by

$$\begin{aligned} &Q_{\text{modularity}}(G, D \setminus \{d_1, d_2\} \cup \{d\}) - Q_{\text{modularity}}(G, D) \\ &\quad = 2|d_1||d_2|/v_V - 2|d_1||d_2||c|^2/v_V^2 \\ &\quad = 2|d_1||d_2|(v_V - |c|^2)/v_V^2 > 0, \end{aligned}$$

which contradicts the assumption that D has maximal modularity. Therefore the only optimal clustering of G is C. Note that the above inequality only holds when $|c|^2 = v_c < v_V$, which is the case because $C \neq \{V\}$.

When $C = \{V\}$, a clique graph will not work; because both $\{V\}$ and the clustering that assigns half the nodes to one cluster, and half to another have modularity equal to 0. In this case, instead define $G = (V, A)$ by $a_{ij} = 1$ if $i \neq j$ and 0 if $i = j$. Then the modularity for C is $q(G, \{V\}) = 0$. Any cluster d in a clustering D will have $v_d = |d|(|V| - 1)$ and $w_d = |d|(|d| - 1)$. Therefore the contribution of this cluster to the total quality is $-|d|(|V| - |d|)/(|V|^2(|V| - 1))$,

which is negative when $|d| < |V|$. So the modularity of any clustering other than $\{V\}$ will be negative, hence $\{V\}$ is the only optimal clustering.

Since for every C we can construct a graph where C is the only optimal clustering, modularity is rich.

□

3.B Proof of Theorem 3.15, adaptive scale modularity is rich

Denote by $f_C(d)$ the largest fraction of any cluster from C that is contained in a cluster d.

$$f_C(d) = \max_{c \in C} \frac{|c \cap d|}{|c|}.$$

For any clustering D we have that

$$\sum_{d \in D} f_C(d) = \sum_{d \in D} \max_{c \in C} \frac{|c \cap d|}{|c|} \leqslant \sum_{d \in D} \sum_{c \in C} \frac{|c \cap d|}{|c|} = |C|. \tag{3.5}$$

And since $f_C(d) \leqslant 1$ for all clusters d, we also have that

$$\sum_{d \in D} f_C(d) \leqslant |D|. \tag{3.6}$$

Lemma 3.18. *For a clique graph of C it is the case that $w_d/v_d \leqslant f_C(d)$.*

Proof. Given a cluster d and a clique graph G of C with weight $k > 0$, the volume of d is

$$v_d = \sum_{c \in C} k|c \cap d||c|,$$

and the within cluster weight is

$$w_d = \sum_{c \in C} k|c \cap d|^2.$$

Therefore

$$w_d \leqslant \sum_{c \in C} k|c \cap d||c| f_C(d) = v_d f_C(d).$$

And hence $w_d/v_d \leqslant f_C(d)$. □

Lemma 3.19. *Let G be the clique graph of a clustering C with weight k, and let $0 < \beta < 1$ be a constant. Then $\sum_{d \in D}(w_d/v_d - \beta) = (1-\beta)|C|$ if $D = C$, while $\sum_{d \in D}(w_d/v_d - \beta) < (1-\beta)|C| - \epsilon$ if $D \neq C$, where $\epsilon = \min(\beta, 1-\beta, 1/|V|)/2$.*

Proof. Suppose that $D = C$, then for every cluster $c \in C$, $w_c = \nu_c = k|c|^2$, and so

$$\sum_{c \in C} \left(\frac{w_d}{\nu_d} - \beta \right) = (1 - \beta)|C|.$$

Otherwise, $D \neq C$. Assume by contradiction that

$$\sum_{d \in D} \left(\frac{w_d}{\nu_d} - \beta \right) \geqslant (1 - \beta)|C| - \min(\beta, 1/|V|)/2.$$

By Lemma 3.18,

$$\begin{aligned} &|C| - \beta(|C| + 1) \\ <&|C| - \beta|C| - \epsilon \\ \leqslant &\sum_{d \in D} (\frac{w_d}{\nu_d} - \beta) \\ \leqslant &\sum_{d \in D} (f_C(d) - \beta) \\ \leqslant &|C| - \beta|D|. \end{aligned}$$

Since $\beta > 0$, this implies that $|D| < |C| + 1$.

Additionally, since $f_C(d) \leqslant 1$ for all clusters $d \in D$,

$$\begin{aligned} &(1 - \beta)(|C| - 1) \\ <&(1 - \beta)|C| - \epsilon \\ \leqslant &\sum_{d \in D} (f_C(d) - \beta) \\ \leqslant &(1 - \beta)|D| \end{aligned}$$

Since $\beta < 1$, this implies that $|D| > |C| - 1$. Hence $|D| = |C|$.

Suppose that $f_C(d) < 1$ for some $d \in D$, which implies that $|c \cap d| < |c|$. Because edges are discrete, this can only happen when $|c \cap d| \leqslant |c| - 1$ for all clusters c. And the size of clusters is bounded by $|c| \leqslant |V|$. Hence $f_C(d) \leqslant (|V| - 1)/|V| = 1 - 1/|V|$. And since for all other clusters d', $f_C(d') \leqslant 1$, we then

have

$$\begin{aligned}
&\sum_{d\in D}(f_C(d)-\beta)\\
\leqslant&(1-\beta)|D|-1/|V|\\
<&(1-\beta)|C|-\epsilon\\
\leqslant&\sum_{d\in D}(w_d/v_d-\beta)\\
\leqslant&\sum_{d\in D}(f_C(d)-\beta),
\end{aligned}$$

which is a contradiction. Hence, it must be the case that $f_C(d) = 1$ for all clusters $d \in D$. By the definition of f_C this means that for every d there is a cluster $c \in C$ such that $|c \cap d| = |c|$, and therefore $c \subseteq d$. Since the clusters are disjoint and $|D| = |C|$, this implies that $D = C$. Which is a contradiction, so $\sum_{d\in D}(w_d/v_d - \beta) < (1-\beta)|C| - \epsilon$. □

When $M = 0$, the adaptive scale modularity reduces to $w_d/(\gamma v_d) - |D|/\gamma^2$, and the above lemma is enough to prove richness. For non-zero values of M, we can get 'close enough' by choosing large enough edge weights. This is formalized in the following lemma.

Lemma 3.20. *Let d be a cluster in a clustering of a clique graph of C with weight k. Then*

$$\frac{w_d}{v_d} - \beta - \beta M/k \leqslant q(d)/\beta \leqslant \frac{w_d}{v_d} - \beta + 2\beta^2 M/k,$$

where

$$q(d) = \frac{w_d}{M + v_d/\beta} - \Big(\frac{v_d}{M + v_d/\beta}\Big)^2$$

denotes the contribution of d to the M-adaptive modularity.

Proof. Since clusters are non-empty, and in a clique graph $a_i i = k$, it follows that

$v_d \geqslant w_d \geqslant k$. So

$$\begin{aligned}
&q(d)/\beta \\
=&\frac{\beta M w_d + v_d w_d - \beta v_d^2}{(\beta M + v_d)^2} \\
=&\frac{w_d}{v_d} - \beta + \frac{\beta^2 M(\beta M + 2v_d) - \beta^2 M^2 w_d/v_d - \beta M w_d}{(\beta M + v_d)^2} \\
\leqslant&\frac{w_d}{v_d} - \beta + \frac{\beta^2 M(\beta M + 2v_d)}{(\beta M + v_d)^2} \\
\leqslant&\frac{w_d}{v_d} - \beta + \frac{2\beta^2 M(\beta M + 2v_d)}{(\beta M + v_d)(\beta M + 2v_d)} \\
=&\frac{w_d}{v_d} - \beta + \frac{2\beta^2 M}{\beta M + v_d} \\
\leqslant&\frac{w_d}{v_d} - \beta + \frac{2\beta^2 M}{k}.
\end{aligned}$$

And since $w_d \leqslant v_d$,

$$\begin{aligned}
&q(d)/\beta \\
=&\frac{w_d}{v_d} - \beta + \frac{\beta^2 M(\beta M + 2v_d) - \beta^2 M^2 w_d/v_d - \beta M w_d}{(\beta M + v_d)^2} \\
\geqslant&\frac{w_d}{v_d} - \beta - \frac{\beta^2 M^2 + \beta M v_d}{(\beta M + v_d)^2} \\
=&\frac{w_d}{v_d} - \beta - \frac{\beta M}{\beta M + v_d} \\
\geqslant&\frac{w_d}{v_d} - \beta - \frac{\beta M}{k}.
\end{aligned}$$

□

Combining these lemmas yields the proof of the general theorem:

Proof. Given a clustering C. Define $\beta = 1/\gamma$. If $\gamma > 1$ then $0 < \beta < 1$. Pick $k > 3|V|\beta^2 M/\epsilon$ where ϵ is defined as in Lemma 3.19.

Let G be the clique graph of C with weight k. Let $D \neq C$ be a clustering of

G. Then by Lemmas 3.19 and 3.20,

$$\begin{aligned}
&Q_{M,\gamma}(G,D)/\beta \\
=&\sum_{d\in D} q(d) \\
\leqslant&\sum_{d\in D} (w_d/v_d - \beta + 2\beta^3 M/k) \\
\leqslant&(1-\beta)|C| + 2|D|\beta^3 M/k - \epsilon \\
\leqslant&(1-\beta)|C| + 2|V|\beta^2 M/k - \epsilon \\
<&(1-\beta)|C| - |V|\beta^2 M/k \\
\leqslant&(1-\beta)|C| - |C|\beta^2 M/k \\
=&\sum_{c\in C} (w_c/v_c - \beta + \beta^2 M/k) \\
\leqslant&Q_{M,\gamma}(C)/\beta.
\end{aligned}$$

Hence the quality is maximal for C. Since there is a clique graph and k for every clustering, adaptive scale modularity is rich. □

3.C Proof of Theorem 3.16, adaptive scale modularity is monotonic

Proof. Given a constants $M > 0$ and $\gamma \geqslant 2$, a graph G and a clustering C of G. Let $c \in C$ be any cluster. Writing the volume of c as $v_c = w_c + b_c$, the contribution of this cluster to the quality of G is $q(w_c, b_c)$ where

$$q(w,b) = \frac{w}{M+\gamma w+\gamma b} - \Big(\frac{w+b}{M+\gamma w+\gamma b}\Big)^2.$$

The partial derivatives of q are

$$\begin{aligned}
\frac{\partial q(w,b)}{\partial w} &= \frac{M^2+(\gamma-2)M(w+b)+\gamma b(M+\gamma w+\gamma b)}{(M+\gamma w+\gamma b)^3} \geqslant 0 \\
\frac{\partial q(w,b)}{\partial b} &= -\frac{\gamma w M+(w+b)(M+\gamma^2 w)}{(M+\gamma w+\gamma b)^3} \leqslant 0.
\end{aligned}$$

This means that q is a monotonically non-decreasing function in w and a non-increasing function in b.

For any graph G′ that is a C-consistent change of G, it holds that $w'_c \geqslant w_c$ and $b'_c \leqslant b_c$. So $q(w'_c, b'_c) \geqslant q(w_c, b_c)$. And therefore $Q_{M,\gamma}(G', C) \geqslant Q_{M,\gamma}(G, C)$. So adaptive scale modularity is monotonic. □

Chapter 4

Comparing quality functions for network clustering: The resolution bias matters most

Results of a recent comparative experimental assessment of methods for network community detection applied to benchmark graphs indicate that the two best methods use different objective functions but a similar local search-based optimization procedure, called the Louvain method.

This observation motivates the following research question: given the Louvain optimization procedure, how much does the choice of the objective function influence the results, and in what way? In this chapter we address this question empirically in a broad graph clustering context, that is, when graphs are either given as such or are k*-nearest neighbor graphs generated from a given dataset. We consider normalized cut, modularity, and infomap; as well as two new objective functions. We show that all these objective functions have a resolution bias, that is, they tend to prefer either small or large clusters. When removing this bias, by forcing the objective function to generate a given number of clusters, the Louvain method achieves similar performance across the considered objective functions on benchmark networks with built-in community structure. These results indicate that the resolution bias is the most important difference between objective functions in graph clustering with the Louvain method.*

Spectral clustering is an alternative to local search optimization, and has been used to optimize the popular normalized cut and modularity objectives. We show experimen-

This chapter is based on van Laarhoven and Marchiori (2013a) "Graph clustering with local search optimization: The resolution bias of the objective function matters most", published in Physical Review E. The software used in this chapter is available on `http://cs.ru.nl/~tvanlaarhoven/clustering2012/`.

tally that the Louvain method often achieves superior performance compare to spectral clustering on various benchmark, real-life and k*-nearest neighbor graphs. These results, the flexibility of the Louvain method and its efficiency, provide arguments in favor of this optimization method.*

4.1 Introduction

Due to the intrinsic difficulty of the problem, graph clustering has been tackled by many researchers, yielding a vast amount of heuristic and approximate methods as well as interesting experimental and theoretical results. We refer the reader to the surveys on this topic, e.g., (Fortunato, 2010; von Luxburg, 2007; Schaeffer, 2007). Many methods for graph clustering are based on optimizing a global objective or quality function. The 'optimal' clustering is then the one that maximizes the quality (throughout this chapter we will use maximizations; some objective functions are traditionally minimized, in those cases we negate them). This discrete optimization problem is computationally intractable (at least for the quality functions for which hardness is known, see for instance (Brandes *et al.*, 2008; Fortunato, 2010; Schaeffer, 2007)). Therefore all effective and scalable methods are based on heuristic and approximate techniques.

One can distinguish two main classes of heuristic for optimizing clustering quality functions. The first one is based on relaxing the discrete cluster labels to continuous variables, and solving the resulting problem with spectral methods. To convert the continuous clustering to a discrete one, a separate step is used, usually k-means clustering (see e.g. the review by von Luxburg, 2007). This principled spectral approach is only possible for some quality functions, such as normalized cut (Shi and Malik, 2000) or modularity (Newman, 2006).

The other class of optimization methods is directly based on (local heuristic) discrete optimization. The goal is to find, among all partitions of the data set, the best one according to a given quality function (see e.g. the review by Fortunato, 2010). Although heuristic in nature, this latter approach has broader applicability since any quality function can be used.

A central issue in network community detection is the resolution limit of quality functions, which has been investigated from multiple perspectives, in particular for modularity (Arenas, Fernández, and Gómez, 2008; Lambiotte, 2010; Lancichinetti and Fortunato, 2011; Reichardt and Bornholdt, 2004; Traag, Van Dooren, and Nesterov, 2011). In particular, Fortunato and Barthélemy (2007) showed that modularity optimization is unable to detect small clusters; Good, de Montjoye, and Clauset (2010) showed that the modularity function exhibits extreme degeneracies such that the globally maximal modularity partition is

typically hidden among an exponential number of structurally dissimilar, high modularity solutions.

An experimental study by Lancichinetti and Fortunato (2009) showed that the common spectral methods are far from optimal for the purpose of graph community detection on benchmark graphs. In their review, the two best methods are those of Blondel *et al.* (2008) and Rosvall and Bergstrom (2008). Both of these methods use a similar local search optimization procedure, here called the Louvain method, after the university of Louvain. This method is based on moving nodes between clusters, and constructing a clustering bottom-up. In principle this procedure can be used with any graph clustering quality function. Since good results are obtained with at least two different quality functions, this raises the following questions: In how far does the clustering result of this optimization method depend on the quality function that is being optimized? And in what way does the choice of the quality function influence the results?

In order to address these questions we consider five quality functions, namely normalized cut, modularity, infomap, and two novel simple quality functions. These novel quality functions are designed in such a way that (1) clusterings are better if they contain more within cluster edges, and (2) clusters should not be too small or too large. First, we analyze the resolution bias of these functions, by showing that their optimum is achieved for clusterings consisting of either relatively small or relatively large communities. Next, we apply the Louvain method to these quality functions on benchmark graphs. Results indicate that diverse performance is achieved across the different types of quality functions. We introduce a procedure to automatically control the resolution bias of a quality function. In this way we force the method to output a fixed number of clusters for each quality function. Results of experiments show that the resolution bias plays a central role for the difference in performance of the quality functions. When the resolution bias is 'removed' by fixing the number of clusters, the performance of the Louvain method across these quality functions becomes much more similar.

Spectral clustering is a principled alternative to local search based optimization, and has been used to optimize the popular normalized cut and modularity quality functions (Shi and Malik, 2000; Newman, 2006). We show experimentally that the Louvain method often achieves superior performance compared to spectral clustering on various benchmark, real-life and k-nearest neighbor graphs. These results confirm the findings reported by Lancichinetti and Fortunato (2009) also for k-nearest neighbor graphs. In general these results, the flexibility of the Louvain method and its efficiency, provide arguments in favor of this optimization method.

The chapter is structured as follows. In Section 4.2 we present the five quality

functions, analyze their resolution bias, and introduce a procedure for controlling the size of clusters. In Section 4.3 we describe the Louvain optimization method. In Section 4.4 we apply this method to the quality functions. We show that the resolution bias is the most important difference between the quality functions, and that the Louvain method has difficulties in optimizing specific quality functions. Furthermore, we compare the Louvain method to spectral clustering on benchmark and real-life networks and k-nearest neighbor graphs. Conclusions are reported in Section 4.5.

4.2 Quality functions

4.2.1 Notation

In this chapter we use the same notation defined in Section 3.2.

To briefly recap, a clustering C is a partition of nodes V of a graph. That is, a set of disjoint sets of nodes which we call clusters, that together cover all nodes. So, every node is in exactly one cluster.

A quality function Q assigns a quality to such a clustering of a graph.

To simplify the notation, we will keep the graph argument $G = (V, A)$ implicit everywhere, that is, we assume that there is some specific graph under consideration. We denote the total volume of this graph by $M = v_V$.

As a shorthand, we then define the *normalized volume* of a cluster as $\widehat{v}_c = v_c/M$, and the *normalized within weight* as $\widehat{w}_c = w_c/M$.

4.2.2 The considered quality functions

Different quality functions have been proposed for graph clustering. Perhaps the most well known is normalized cut (Shi and Malik, 2000), which is defined as

$$Q_{\text{NCut}}(C) = -\sum_{c \in C} \frac{v_c - w_c}{v_c}.$$

Maximizing this quality function minimizes the number of between cluster edges, called the cut size.

Another common quality function is modularity, introduced by Girvan and Newman (2002). We already mentioned this quality function in the previous chapter. With the shorthands of normalized volume and normalized within weight it can be written as

$$Q_{\text{modularity}}(C) = \sum_{c \in C} \left(\widehat{w}_c - \widehat{v}_c^2\right).$$

Table 4.1: Quality functions considered in this chapter.

Quality function	Expression
normalized cut	$\sum_{c\in C} -(v_c - w_c)/v_c$
modularity	$\sum_{c\in C} (\widehat{w}_c - \widehat{v}_c^2)$
w-log-v	$\sum_{c\in C} -\widehat{w}_c \log(\widehat{v}_c)$
parabola	$\sum_{c\in C} \widehat{w}_c(1-\widehat{v}_c)$
infomap	$\sum_{c\in C} h(\widehat{v}_c + \widehat{v}_c - \widehat{w}_c)$ $-2\sum_{c\in C} h(\widehat{v}_c - \widehat{w}_c) + h\big(\sum_{c\in C}(\widehat{v}_c - \widehat{w}_c)\big)$

Both of these quality functions can be written as a sum over all clusters,

$$Q(C) = \sum_{c\in C} q(c), \tag{4.1}$$

for some function q. This means that it makes sense to look at the quality of just a subset of the clusters, or of the clustering of just a subset of the nodes.

A notable quality function that does not follow this pattern is infomap (Rosvall and Bergstrom, 2008). This quality function is based on the length of a code for paths through the graph. In addition to a sum of per-cluster scores, infomap also include a global term based on the probability of an edge exiting a cluster[1],

$$\begin{aligned} Q_{\text{infomap}}(C) = &\sum_{c\in C} h\big(\widehat{v}_c + \widehat{v}_c - \widehat{w}_c\big) \\ &- 2\sum_{c\in C} h\big(\widehat{v}_c - \widehat{w}_c\big) + h\Big(\sum_{c\in C}(\widehat{v}_c - \widehat{w}_c)\Big), \end{aligned}$$

where $h(x) = -x\log(x)$. The original infomap quality function contains an additional term,

$$Q_{\text{infomap(Rosvall and Bergstrom, 2008)}}(C) = Q_{\text{infomap}}(C) + \sum_{i\in V} h(\widehat{v}_{\{i\}}),$$

which is needed to make the quality function correspond to a code length. However, since this last term is the same for all clusterings, we don't include it. In addition, without this extra term, the quality of the trivial clustering with one cluster is exactly 0.

There are many more possible quality functions that could be used for graph clusterings. Some considerations for designing such functions are that:

[1]This expression for the infomap quality function is based on the source code of the tools provided by Rosvall et.al., `http://www.tp.umu.se/~rosvall/code.html`

1. All else being equal, clusters are better if they contain more within cluster edges.
2. Clusters should not be too small or too large.

For the first consideration, we can look at the ratio between the volume of a cluster and the number of within edges. For the second consideration, we use a weight function $f(\widehat{v}_c)$ where $f(0) = f(1) = 0$, while $f(x) > 0$ for $0 < x < 1$. Combining these ingredients leads to an quality function of the form

$$Q(C) = \sum_{c \in C} \frac{\widehat{w}_c}{\widehat{v}_c} f(\widehat{v}_c). \tag{4.2}$$

Two simple functions that fit the criteria for f are the parabola $f(x) = x(1-x)$ and the function $h(x) = -x\log(x)$. They give the quality function

$$Q_{\text{parabola}}(C) = \sum_{c \in C} \left(\widehat{w}_c - \widehat{w}_c \widehat{v}_c\right), \tag{4.3}$$

and

$$Q_{\text{w-log-v}}(C) = -\sum_{c \in C} \widehat{w}_c \log(\widehat{v}_c). \tag{4.4}$$

Of course, there are infinite other possibilities. We focus on these two because the former is similar to modularity, while the latter resembles infomap while being much simpler.

Table 4.1 lists all the different graph clustering quality functions that we will consider in this chapter. Many other quality functions have been discussed in the literature, see Fortunato (2010) for an overview. Many of them do not apply in our setting, because they assign a score to a single cluster, not to a clustering. Therefore it is not clear how the cluster scores should be combined into a score for a clustering. When the number of clusters is fixed one could use the sum of scores, but when the number of clusters is allowed to vary this will often not give a good quality function.

4.2.3 Resolution biases

It was shown by Fortunato and Barthélemy (2007) that modularity has a resolution limit, in the sense that it tends to combine small communities into larger ones. Specifically, in a network which has L edges, there is a characteristic number of edges, such that communities with less than $\sqrt{L/2}$ edges are not visible. Kumpula *et al.* (2007) have generalized this result by showing that the graph clustering framework introduced by Reichardt and Bornholdt (2004) also has a

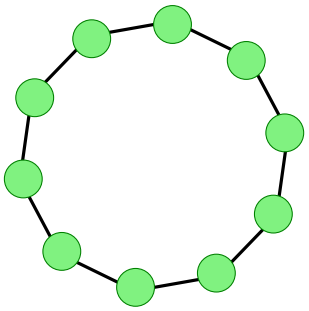


Figure 4.1: Model graph for showing the resolution limits. The circles represent strongly connected 'modules' with $q - r$ internal edges, while the lines represent r edges each.

resolution threshold. The model contains a parameter by which this threshold can be tuned, but no a priori principle is known to select the proper value. They conclude that single global optimization criteria do not seem capable for detecting all communities if their size distribution is broad (Kumpula *et al.*, 2007).

In the sequel we show that the other clustering quality functions here considered have resolution limits. In fact, these are not just limits, but a general bias towards certain cluster sizes. For example, the w-log-v quality function has a resolution limit at smaller cluster sizes, and it always leads to smaller clusters than modularity.

Consider a graph that has n densely connected *modules*, which are loosely connected in a ring (Fortunato and Barthélemy, 2007). Figure 4.1 illustrates such a graph. The modules themselves could be single nodes, cliques or other subgraphs, we are only interested in their volume. In particular, imagine each module having $q - r$ internal edges, and connected to both of its neighbors with r edges each. The volume of a single module X is then $v_X = 2q$, while the volume of the entire graph is $M = 2nq$.

A cluster X_m consisting of m adjacent modules will have normalized volume $\widehat{v}_{X_m} = m/n$. And since all but $2r$ of the edges will be within the cluster, the normalized within weight will be $\widehat{w}_{X_m} = (m - r/q)/n$. By symmetry, we would expect all clusters in the optimal clustering C^* to have the same size (assuming m divides n). There will then be n/m such clusters. So, the total modularity of this clustering is

$$Q_{\text{modularity}}(C^*) = \frac{n}{m}\left(\frac{m - r/q}{n} - \left(\frac{m}{n}\right)^2\right).$$

This expression has a maximum at $m = \sqrt{nr/q}$. So, the larger the graph, or the

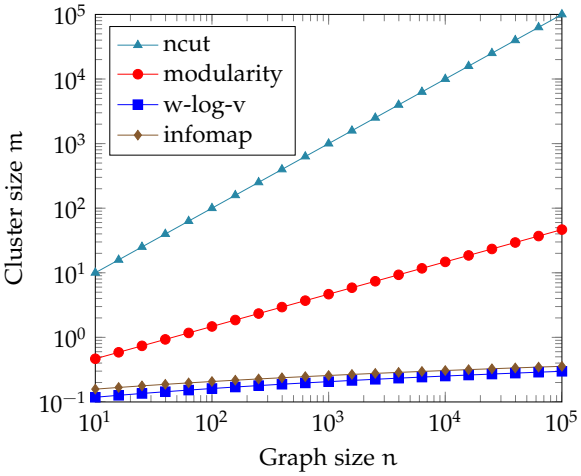


Figure 4.2: The resolution limits of graph clustering quality functions as a function of the graph size. We used $r = 1$ and $q = 46$, which corresponds to modules that are cliques with 10 nodes. The resolution limit of the parabola quality function is the same as that of modularity.

less dense the connections in each module, the larger the clusters that are found. The parabola quality function reaches a maximum at the same point.

For the w-log-v quality function

$$Q_{\text{w-log-v}}(C^*) = -\frac{n}{m}\left(\frac{m}{n} - \frac{r}{qn}\right)\log\left(\frac{m}{n}\right). \tag{4.5}$$

The optimum is at $m = W(enr/q)r/q$ where W is the Lambert W function.

The normalized and ratio cut quality functions behave differently,

$$Q_{\text{NCut}}(C^*) = -\frac{n}{m}\frac{(r/q)(1/n)}{m/n} = -\frac{nr}{m^2 q}. \tag{4.6}$$

This expression has no maximum value, it increases as m gets larger. Since the size of a cluster can not be larger than n, the actual optimum is at $m = n$, i.e. when all modules are in a single cluster.

Finally, for infomap there is no analytical expression for the optimum cluster size, but it can be easily calculated numerically. Figure 4.2 shows the cluster size of the optimal clustering as a function of the number of modules. This optimal size was found by numerical optimization of the quality in terms of the cluster size m. For this figure we have used modules with $r = 1$ and $q = 46$, which corresponds to 45 internal edges, i.e. cliques with 10 nodes. Other module sizes

give qualitatively similar results. For each of the quality functions, the optimal cluster size depends only on the ratio r/q and the number of modules n.

Note that in this figure, for the w-log-v and infomap quality functions the theoretically optimal clustering always has less than 1 clique per cluster, which in practice means that the cliques are perfectly clusterable. To actually see the resolution limit in action for these quality functions, the number of cliques must be *very* large. For example for the value of the w-log-v quality function for $m = 2$ overtakes the value for $m = 1$ when $n > 2^{91} \approx 10^{27}$.

The resolution bias discussed in this section reflects preferences towards certain sizes of clusters, in a situation where all vertices are similar. There are other biases that come into play when the graph is less uniform and when the sizes of clusters will differ (Lancichinetti and Fortunato, 2011).

4.2.4 Controlling the size of clusters

Several generalizations of modularity have been proposed that allow for control over the size of the clusters (Reichardt and Bornholdt, 2004; Angelini *et al.*, 2007; Lambiotte, 2010). Each of these quality functions has a parameter that controls the trade-off between the size of the clusters and the strength of edges within clusters. For example, the quality function introduced by Reichardt and Bornholdt (2004) is, in our notation,

$$Q_{RB}(C, \gamma) = -\sum_{c \in C} \left(\widehat{w}_c - \gamma \widehat{v}_c^2\right). \tag{4.7}$$

In this chapter we also consider other quality functions besides modularity, and hence we would like to add similar size-control parameters to them. In the previous section we have shown that the size of the clusters depends on the size of the graph. Often this dependency is implicit, through the use of the normalized volume and normalized within weight. This dependence can be used to control the cluster sizes.

The idea is to embed the graph in a larger graph, with total volume αM, and thereby change the optimal clustering. Since quality functions such as modularity are a sum over clusters in the form of (4.1), we can look at the contribution to the modularity of a clustering C of the original graph. Denote this contribution by

$$\begin{aligned} Q^{\text{embed}}_{\text{modularity}}(C, \alpha) &= -\sum_{c \in C} \left(\widehat{w}_c/\alpha - (\widehat{v}_c/\alpha)^2\right) \\ &= \frac{1}{\alpha^2}\Big(Q_{\text{modularity}}(C) - (\alpha - 1)\sum_{c \in C} \widehat{w}_c\Big). \end{aligned} \tag{4.8}$$

The optimal clustering does not change when the quality function is multiplied by a constant. Therefore, embedding within a larger graph is equivalent to adding a term to the quality function,

$$Q^+(C, \beta) = Q(C) + \beta \sum_{c \in C} \widehat{w}_c. \tag{4.9}$$

This holds also for the parabola and the w-log-v quality functions.

On the other hand, the normalized cut quality function does not depend on the size of the graph at all. Despite this, we can still use (4.9) to adjust that quality function.

In this way we get a *family* of quality functions parameterized by β for each original quality function. Note also that $Q^+_{\text{modularity}}$ is equivalent to Q_{RB} with $\gamma = 1 + \beta$; and it is equivalent to Q_{NL} introduced in (Lambiotte, 2010) with $t = 1/(1 + \beta)$.

By adjusting the parameter β, the size of the clusters can be controlled. A negative value of β corresponds to embedding the graph in a larger one, so it will lead to fewer larger clusters. A positive β will lead to more and smaller clusters. However, we have to be careful with large positive values of β, since that punishes within cluster edges, instead of rewarding them.

Since a large part of the difference between quality functions lies in the different preferred cluster sizes, this added flexibility might be enough to get rid of much of these differences. Suppose, for example that the number of clusters is known. Then we can use binary search to look for a value of β that leads to the desired number of clusters. The resolution bias of the quality function is then no longer important, since by fixing the number clusters we also fix their average size.

4.3 The Louvain method

The optimization procedure that we use is the local search method developed by Blondel *et al.* (2008), which is usually called the Louvain method, after the university of Louvain. It was initially proposed for optimizing modularity, but the same method can also be used for any other graph clustering quality function. The method is very fast, and can deal with millions of nodes in seconds. We will briefly describe the algorithm here.

Initially, each node is assigned to a singleton cluster. Then, iteratively, nodes are moved between clusters as long as the quality improves. For each node, only moves to neighboring cluster are considered; where neighboring clusters are those clusters that contain neighboring nodes. The nodes are visited in a random order.

The most expensive part of the algorithm is recomputing the value of the quality function. For quality functions that are written as a sum over the clusters, as in (4.1), this computation can be done efficiently, because only two terms of the sum change when a node is moved between clusters.

Because the quality increases with each move, eventually a local optimum will be reached. However, the clusters in this local optimum will often be too small. It is just that they can not be improved by moving *single nodes*. We will call the clusters found at this point *small clusters*. The next step is to repeat the optimization procedure, but this time moving entire small clusters instead of single nodes. Effectively, we are then clustering a condensed graph, where each node in the condensed graph is a small cluster.

The step of moving small clusters is again repeated until convergence. The clusters at that point become the new small clusters. At some point no small clusters will be moved, and then the algorithm stops.

Several variations to this basic recipe are possible. For instance, if the clusters become too large, one could apply the clustering algorithm from scratch to only the nodes in a single cluster. This might lead to a better optimum. However, we do not find this step to improve the results in our experiments. Another improvement is to simply run the algorithm several times with different random seeds, and to pick the best solution.

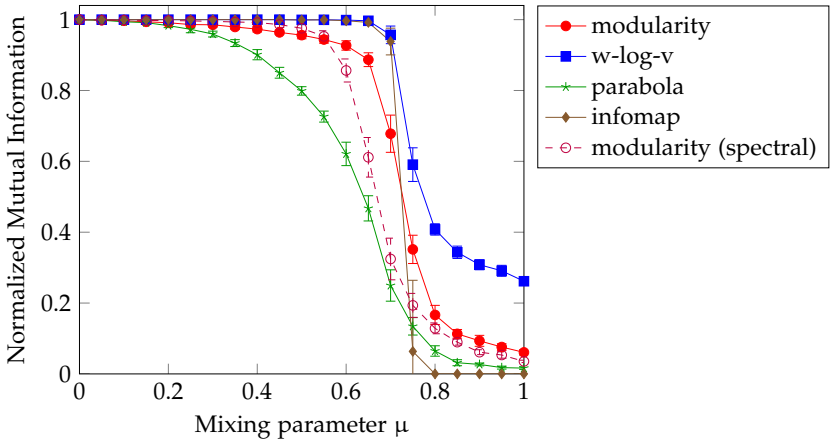
4.4 Experiments

4.4.1 Community detection benchmarks

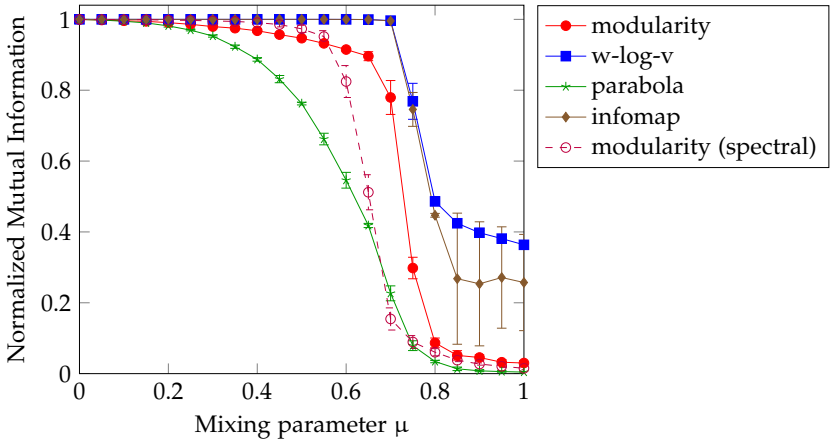
We consider the LFR graph generator by Lancichinetti, Fortunato, and Radicchi (2008) which constructs networks with built-in community structure. In this benchmark, the size of each cluster is drawn from a power-law distribution; as is the degree of each node. These benchmarks are specifically constructed to closely resemble real world graphs. Indeed, it has previously been observed that real world graphs also have such a power-law degree distribution (Clauset, Shalizi, and Newman, 2009).

The LFR model has several parameters. The most important is the mixing parameter μ, which controls the fraction of edges that are between clusters. Essentially this is the amount of noise in the graph. If $\mu = 0$ all edges are within cluster edges, if $\mu = 1$ all edges are between nodes in different clusters.

Other parameters control the number of nodes, the distributions of cluster sizes, the distribution of degrees, etc. If something is known about the target graph, then these parameters should be chosen to match that graph. However, in this chapter we do not try to match any particular graph. We therefore follow



(a) Small 1000



(b) Big 5000

Figure 4.3: Normalized mutual information as a function of the mixing parameter, for various quality functions; on the Small 1000 and Big 5000 datasets. The error bars indicate standard deviation.

the settings used by Lancichinetti and Fortunato (2009). They describe four benchmarks. Two with 'small clusters' of between 10 and 50 nodes, and two with 'large clusters' of between 20 and 100 nodes. Each graph has either 1000 or 5000 nodes in total.

To measure the quality of a clustering, we compare it to the ground truth

with the normalized mutual information (NMI) metric (Danon *et al.*, 2005),

$$\hat{I}(C_1, C_2) = \frac{2I(C_1, C_2)}{H(C_1) + H(C_2)}, \tag{4.10}$$

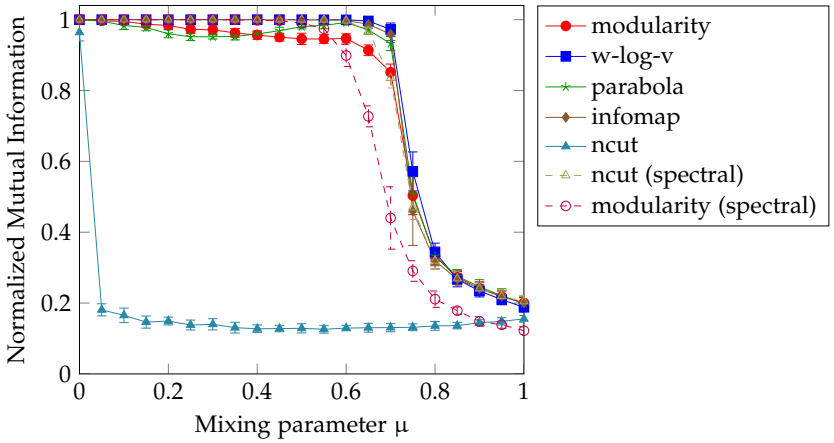
where I is mutual information and H is entropy. Figure 4.3 shows the normalized mutual information as a function of the mixing parameter for the different clustering quality functions. We used the Louvain method to optimize all quality functions. We did not include normalized cut, since without adjustment this quality function always leads to a single cluster. We only show the results for the benchmark with 1000 nodes and small clusters and the benchmark with 5000 nodes and large clusters. The results for the other two benchmarks are similar.

For comparison, besides the Louvain method, we also include two spectral clustering methods in our experiments. First of all a simple method that approximately maximizes the normalized cut quality by solving a generalized eigenvalue problem for the graph Laplacian, and then finds discrete clusters using k-means (Shi and Malik, 2000). Secondly the method of Newman (2006), which uses eigenvectors of the modularity matrix. Based on these eigenvectors a few (two or three) clusters are found, which are then recursively subdivided until the optimal modularity is reached. The clusterings are further optimized by a Kernighan-Lin style algorithm (Kernighan and Lin, 1970), which moves single nodes around in a similar fashion to the first iteration of the Louvain algorithm. These two methods represent the complete opposite approach to clustering. Whereas the Louvain method uses a greedy search to grow clusters from the bottom up, these spectral methods use a smooth approximation to repeatedly subdivide the graph.

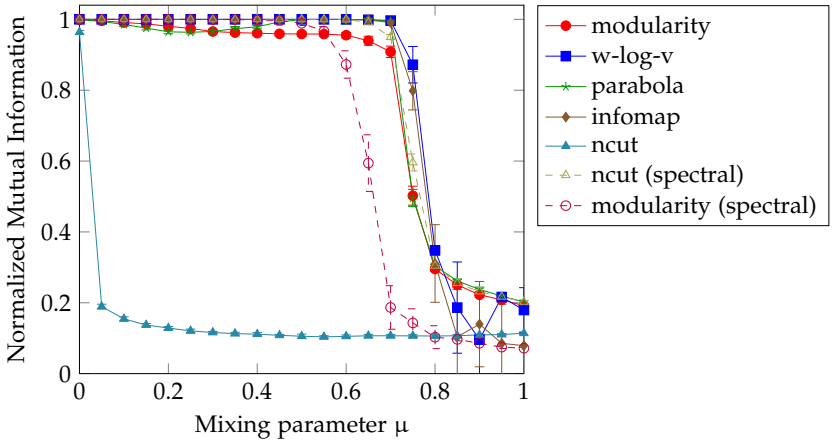
Optimizing the w-log-v and infomap quality functions always leads to a perfect recovery of the clustering up to $\mu = 0.65$ on the small dataset and $\mu = 0.7$ on the big dataset. For higher μ, infomap suddenly gives a clustering with normalized mutual information 0. This is the clustering where all nodes are put into a single cluster. Notice the large standard deviation on the Big 5000 dataset. In some cases the optimizer finds the trivial clustering, and in other cases it finds a clustering comparable to that found with the w-log-v quality function.

The w-log-v quality function does not have the instability of infomap, and the performance normalized mutual information decreases more gradually. The other two quality functions, modularity and parabola, perform worse. As we show next, this mainly due to the failure to recover the right number of clusters.

When the true number of clusters is known, we can adjust the quality function to get the desired number of clusters, as described in Section 4.2.4. In this case it is also possible to use spectral clustering to optimize the normalized cut quality function, which requires the number of clusters as an input parameter.



(a) Small 1000



(b) Big 5000

Figure 4.4: Normalized mutual information as a function of the mixing parameter, when the number of clusters has been fixed to the actual number of clusters. We show results for the Small 1000 and Big 5000 datasets. The error bars indicate standard deviation.

Figure 4.4 shows the results on the same benchmark graphs when forcing the number of clusters to be equal to the number of clusters in the ground truth.

The behavior of the different quality functions is now very similar. However, with the normalized cut quality function we are still unable to find the right clustering. This is due only to the optimizer, because when normalized cut is

optimized with spectral clustering, the correct clustering is found.

4.4.2 Quality functions versus optimization

One might wonder in how far the results of these experiments depend on the quality function, and how much they depend on how that function is being optimized. To distinguish between the two, we compare the quality of the clustering found by the Louvain algorithm to the quality of the ground truth clustering. If the quality is higher at the ground truth clustering, then this indicates that optimizer has failed to find a good enough clustering. On the other hand, if the quality of the ground truth clustering is lower, then the optimizer has found a clustering that is 'better than the ground truth' according to the quality function. That means that this quality function is unsuitable in this situation. As a baseline, we also compare with a clustering obtained on a randomly rewired graph with the same degree distribution.

Figure 4.4.2 shows the value of the quality functions for different mixing parameters. We can see that the quality of the ground truth crosses that of the randomly rewired graph at different points for different quality functions. Beyond this point, there is little hope of recovering the true clustering, since the graph has then no more cluster structure than a random one. We can also see cases where the optimizer fails, such as with the w-log-v quality function at $\mu = 0.75$. Here the ground truth has a higher quality than the clustering found by the optimizer. Repeating the optimization 20 times leads to a slightly better optimum, but not yet to the ground truth. Even more repetitions can further improve performance, but only slightly.

The parabola quality function shows a different picture. The clustering found by the optimizer has a lower quality than the ground truth in many cases. This means that the Louvain method often fails to find the optimal clustering or one close to it. The clustering that is found instead has too few clusters. In Section 4.2.3 we showed that the parabola function has the same resolution bias as modularity, while optimizing modularity with the Louvain method does not give a clustering with a lower quality than the ground truth. This means that the resolution bias does not tell the whole story. Another important aspect of the results seem to be how easy the quality function is to optimize with the Louvain method.

With the infomap quality function, the clustering found for the randomly rewired graph always has quality 0. This corresponds to a clustering where all nodes belong to the same cluster. This clustering is always a possible one, but it is not always found by the optimizer. For example, at $\mu = 0.7$, the optimizer sometimes finds an infomap quality that is greater than 0. In these cases the

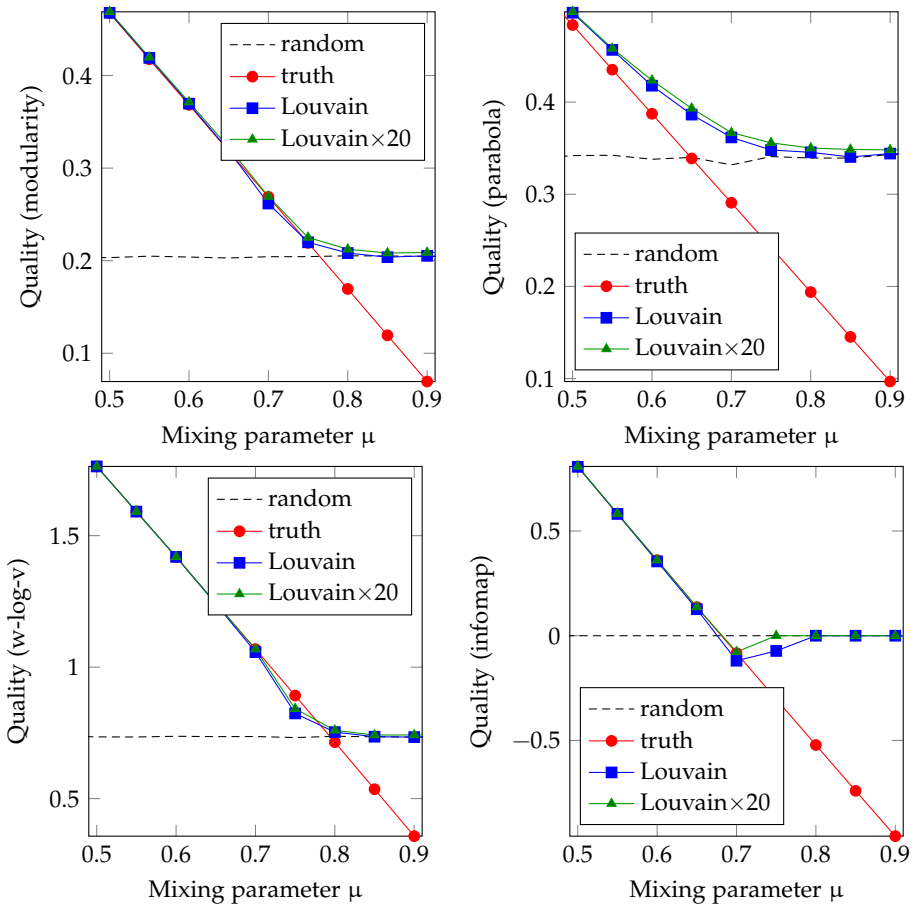


Figure 4.5: The quality of clustering, as a function of the mixing parameter on the Small 1000 dataset. 'random' is the quality reached by the optimizer on a randomly rewired graph, 'truth' is the quality of the ground truth clustering. Louvain is the value reached by the optimizer, and Louvain×20 is the best quality out of 20 restarts. The quality functions shown are: modularity (top left), parabola (top right), w-log-v (bottom left), and infomap (bottom-right). Note that the quality values are on an essentially arbitrary scale, and it makes no sense to compare values for different quality functions.

Table 4.2: The normalized mutual information for optimizing the various quality functions on real world networks. The best results are indicated in boldface. The Louvain method is used for optimization except for the results marked with (sp.), where spectral clustering is used. In the first part of the table the number of clusters is forced equal to the true number, which is shown in the 'clusters' column. In the second part the number of clusters is left free, in parenthesis is the number of clusters that are found for each method.

	Zachary	Football	Pol. Books	Pol. Blogs
vertices	34	115	105	1490
clusters	2	12	3	2
modularity	67.7%	**92.4%**	55.4%	11.5%
parabola	67.7%	**92.4%**	55.4%	11.5%
w-log-v	67.7%	**92.4%**	**57.4%**	11.5%
infomap	30.1%	**92.4%**	**57.4%**	11.5%
ncut (spectral)	**73.2%**	**92.4%**	54.2%	0.9%
modularity (spectral)	67.7%	89.5%	54.2%	**52.2%**
modularity	**58.8%** (4)	89.0% (10)	**56.0%** (5)	37.2% (278)
parabola	**58.8%** (4)	82.0% (8)	46.9% (6)	33.9% (280)
w-log-v	42.8% (6)	**92.4%** (12)	40.7% (11)	25.1% (314)
infomap	56.8% (3)	**92.4%** (12)	53.7% (5)	33.9% (303)
modularity (spectral)	**58.8%** (4)	85.2% (9)	52.1% (4)	**52.2%** (2)

optimizer is stuck in a local maximum that is not globally optimal.

4.4.3 Real world community graphs

We next applied the optimizer to several small real-world networks. We only looked at networks for which some kind of ground truth clustering is know. For other networks, often only a modularity score is reported in the literature. But since we use several different quality functions, this makes no sense in our context. The networks we considered are:

- Zachary's karate club (Zachary, 1977).
- Football: A network of American college football games (Girvan and Newman, 2002).
- Political books: A network of books about US politics (Krebs, 2004). The clusters are left-wing, right-wing and neutral books.

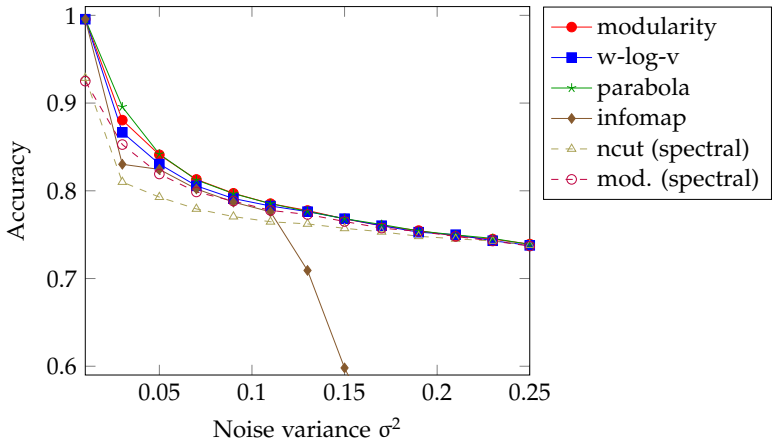


Figure 4.6: Accuracy of the clustering methods on the two moons dataset, as a function of the variance of the noise.

- Political blogs: Hyperlinks between weblogs on US politics (Adamic and Glance, 2005).

In each case, we force the number of clusters found by the methods to be the same as the number of clusters in the dataset. The first part of Table 4.2 gives the results of these experiments. In most experiments the spectral methods give the best results. We believe that this is due to the small number of clusters. The difference is especially large for the Political Blogs dataset, which is the largest dataset in this experiment. Since the spectral methods start with a single large cluster, the final solution with just two or three clusters is a relatively close to this starting point. In contrast, the Louvain method starts from singleton clusters, which are gradually merged.

The second part of the table gives results when the number of clusters is not fixed. In these experiments the results are more varied. Observe that for the football dataset the modularity and parabola quality functions no longer find the same clustering as the other quality functions, instead giving fewer clusters. This is due to the biases of these quality functions.

4.4.4 Artificial nearest neighbor data

We now consider the applicability of the Louvain method to clustering nearest neighbor graphs. We follow the setup from Bühler and Hein (2009).

First we ran experiments on the two moons dataset. This dataset consists of points on two half-circles, that are offset from each other, embedded in a d

dimensional space, and have added Gaussian noise.

For each point x_i in the dataset we add edges to its k-nearest neighbors with the weights

$$a'_{ij} = \exp(-4\|x_i - x_j\|^2/\|x_i - x_i^k\|^2),$$

where x_i^k is the k-nearest neighbor of x_i. To make the graph symmetric, we take the maximum weight over the two edge directions, $a_{ij} = \max(a'_{ij}, a'_{ji})$.

In our experiments we used $n = 2000$ points of dimension $d = 100$, and $k = 10$ neighbors of each point. The optimizer is restricted to finding 2 clusters. We evaluate the performance by looking at the leave-one-out accuracy. That is, the fraction of points that have the same label as the majority of the other nodes in the same cluster. Figure 4.6 shows the accuracy as a function of the variance of the noise.

The results are in some ways opposite to those on the LFR benchmark. For these K nearest neighbor graphs, the modularity and parabola quality functions outperform w-log-v and infomap. We conjecture that this has to do with the resolution bias of the methods. In the LFR benchmarks we search for more clusters, around 40, compared to only 2 in this experiment. Thus, the quality functions that are biased towards larger clusters will perform better here. However, at the moment we have no proof or additional evidence to support this conjecture.

4.4.5 Real world nearest neighbor datasets

We used the same construction of a nearest neighbor graph outlined in the previous paragraph also on real-world and UCI datasets. In each case, we force the number of clusters to be the same as the number of classes in the dataset. Table 4.3 contains the results of these experiments. Since this is a classification task, we have also measured the performance with leave one out accuracy instead of normalized mutual information. The LOO accuracy is the fraction of points that would be correctly classified if the most common label among all other points in the same cluster is used as that cluster's label. Table 4.4 contains the LOO accuracy results.

On the iris dataset all methods except spectral modularity optimization achieve the same high accuracy. This can be explained by the small size of the dataset and the relatively easy classification task. The iris dataset was previously used in a comparison of different multi-resolution methods (Granell, Gómez, and Arenas, 2012), the accuracy reported in that paper is the same 96% that we found. On the MNIST and USPS datasets, the Louvain method significantly outperforms spectral clustering. These datasets have many classes, many features and are not completely balanced. On the other hand, on the ringnorm dataset spectral normalized cut optimization perform much better than other

Table 4.3: The normalized mutual information of various methods on real world datasets. The best results are indicated in boldface. In the first part of the table the number of clusters is forced equal to the true number, which is shown in the 'clusters' column. In the second part the number of clusters is left free, in parenthesis is the number of clusters that are found for each method.

	MNIST	USPS	IRIS	COIL20	WAVEFORM	RINGNORM	FACES
vertices	70000	9298	150	1440	5000	7400	624
clusters	10	10	3	20	2	2	20
modularity	91.1%	87.9%	**86.4%**	92.5%	41.7%	7.8%	86.2%
parabola	**91.9%**	87.9%	**86.4%**	92.4%	**42.1%**	9.0%	86.2%
w-log-v	87.7%	**89.8%**	**86.4%**	**92.8%**	41.6%	0.0%	85.7%
infomap	87.6%	89.6%	**86.4%**	92.4%	41.3%	0.0%	85.7%
ncut (sp.)	76.3%	79.9%	**86.4%**	91.9%	36.5%	**14.4%**	**86.4%**
modularity (sp.)	22.7%	35.7%	74.0%	49.4%	12.6%	2.0%	62.8%
modularity	**82.8%** (18)	84.0% (17)	60.4% (9)	88.7% (27)	28.4% (6)	2.4% (19)	**88.4%** (32)
parabola	82.6% (18)	**84.4%** (16)	61.1% (9)	**88.8%** (27)	28.1% (6)	3.8% (5)	**88.4%** (32)
w-log-v	45.0% (2058)	52.4% (473)	51.6% (18)	74.1% (143)	12.8% (298)	**4.8%** (559)	80.4% (92)
infomap	46.7% (1523)	55.0% (332)	54.5% (15)	76.7% (107)	13.8% (193)	4.6% (469)	81.9% (76)
modularity (sp.)	51.8% (382)	59.1% (122)	**64.5%** (8)	75.7% (110)	**29.3%** (5)	3.4% (8)	80.3% (61)

methods.Overall, as on the two-moons dataset, the parabola quality function gives the best results.

The second part of the Table 4.4 shows that, when the number of clusters is not fixed, the w-log-v quality function has the highest or close to the highest accuracy in all cases. But this is merely because the w-log-v quality function has an optimum with the most clusters, and the accuracy is nearly always higher with such a more fine grained clustering. On the other hand, the normalized mutual information is higher when the number of clusters is closer to the true number of clusters. In this regard, the modularity and parabola quality functions give the best results.

4.5 Conclusions

The results of our investigation show that the choice of quality function matters for graph clustering with the Louvain method. The quality function has two

Table 4.4: The classification accuracy of various methods on real world datasets. The best results are indicated in boldface. In the first part of the table the number of clusters is forced equal to the true number, which is shown in the 'clusters' column. In the second part the number of clusters is left free, in parenthesis is the number of clusters that are found for each method.

	MNIST	USPS	iris	coil20	waveform	ringnorm	faces
vertices	70000	9298	150	1440	5000	7400	624
clusters	10	10	3	20	2	2	20
modularity	95.0%	89.7%	**96.0%**	85.6%	80.0%	66.1%	**77.2%**
parabola	**96.8%**	89.7%	**96.0%**	85.2%	**80.3%**	67.4%	**77.2%**
w-log-v	84.2%	91.2%	**96.0%**	82.9%	80.0%	50.5%	76.0%
infomap	84.2%	**92.6%**	**96.0%**	82.8%	79.3%	50.5%	76.0%
ncut (sp.)	75.8%	80.5%	**96.0%**	**86.9%**	79.5%	**72.0%**	76.4%
modularity (sp.)	31.0%	43.1%	83.3%	43.0%	69.5%	58.1%	48.6%
modularity	**96.8%**	96.7%	96.0%	84.9%	81.3%	60.8%	84.8%
parabola	96.7%	96.6%	**96.7%**	85.2%	81.4%	61.9%	84.8%
w-log-v	96.8%	**96.8%**	96.0%	**95.4%**	**86.3%**	**68.3%**	**99.4%**
infomap	96.8%	96.8%	94.7%	94.7%	85.7%	67.9%	97.2%
modularity (sp.)	79.1%	84.9%	**96.7%**	87.8%	83.4%	58.6%	89.5%

important main effects.

First of all we have shown that different graph clustering quality functions have different resolution biases. These form the largest difference between the different quality functions. Our experiments show that on benchmark network graphs with built-in community structure, when controlling the number of clusters, the clusterings found with different quality functions are very similar. However, when the number of clusters is not fixed, the resolution bias has a large influence on the performance of the method.

Secondly, some quality functions are easier to optimize with the Louvain method than others. For example, in the experiments on the LFR benchmarks, the clustering found with the parabola quality function is often not optimal for that quality function. In addition, optimizing other quality functions such as normalized cut turns out to be very hard. For that quality function spectral methods are more suitable.

For nearest neighbor graphs, the Louvain method often significantly outperforms spectral clustering, while never performing significantly worse. When the number of clusters is fixed to a small number, the modularity and parabola quality functions give the best results. When the number of clusters is not fixed, the w-log-v quality function finds the most clusters, and has the best accuracy.

But the modularity and parabola quality functions stay close to the true number of clusters, and they give the best NMI.

The way we adjust the number of clusters, by embedding the graph in a larger one, works best when we want to decrease the number of clusters. For some quality functions, in particular normalized cut, we instead wish to increase the number of clusters. Other adjustments to the quality function might work better in that case, for example adding a term to directly reward the number of clusters. Such an adjustment will lead to another family of quality functions, perhaps with different resolution bias characteristics.

The parabola and modularity quality functions have the same resolution bias, $\sqrt{nr/q}$. However, the behavior of the two quality functions on the LFR benchmarks differ significantly. This is in part due to the inability of the Louvain method to find the optimal clustering for the parabola quality function, but it seems that the quality functions also differ in other ways. An avenue for future work is to improve the resolution bias model to show how these quality functions differ.

In this chapter we have only considered undirected graphs. Each of the quality functions can be adapted to directed graphs by using a variation of the volume that is based on the indegree or outdegree of nodes, or on a combination of the two. In (Rosvall and Bergstrom, 2008), the infomap quality function is defined based on the outdegree and on edges leaving a cluster. It is not clear what the advantages are of directly using undirected graphs for clustering, or how the clustering of a graph should differ from the clustering of its transpose.

Chapter 5

Finding Protein Complexes, an application of network clustering

Unraveling the community structure of real-world networks is an important and challenging problem. Recently, it has been shown that methods based on optimizing a clustering measure, in particular modularity, have a resolution bias, e.g. communities with sizes below some threshold remain unresolved. This problem has been tackled by incorporating a parameter in the method which influences the size of the communities. Methods incorporating this type of parameter are also called multi-resolution methods. In this chapter we consider fast greedy local search optimization of a clustering quality function with two different quality functions incorporating a resolution parameter: modularity and a function we introduced in a recent work, called w-log-v. We analyze experimentally the performance of the resulting algorithms when applied to protein-protein interaction (PPI) networks. Specifically, publicly available yeast protein networks from past studies, as well as the present BioGRID database, are considered. Furthermore, to test robustness of the methods, various types of randomly perturbed networks obtained from the BioGRID data are also considered. Results of extensive experiments show improved or competitive performance over MCL, a state-of-the-art algorithm for complex detection in PPI networks, in particular on BioGRID data, where w-log-v obtains excellent accuracy and robustness performance.

This chapter is based on van Laarhoven and Marchiori (2012) "Robust community detection methods with resolution parameter for complex detection in protein protein interaction networks", which received the best paper award at the 7th IAPR international conference on Pattern Recognition in Bioinformatics.

5.1 Introduction

The development of advanced high-throughput technologies and mass spectrometry has boosted the generation of experimental data on protein-protein interaction and shifted the study of protein interaction to a global, network level. In particular, it has been shown that groups of proteins interacting more with each other than with other proteins, often participate in similar biological processes and often form protein complexes performing specific tasks in the cell. Detecting protein complexes, consisting of proteins sharing a common function, is important, for instance for predicting a biological function of uncharacterized proteins. To this aim protein-protein interaction (PPI) networks have been used as a convenient graph-based representation for the comparative analysis and detection of (putative) protein complexes (X. Li *et al.*, 2010). A PPI network is a graph where nodes are proteins and edges represent interactions between proteins.

Detecting protein complexes in a PPI network can be formalized as a graph-clustering problem. Clustering amounts to divide data objects into groups (clusters) in such a way that objects in the same cluster are more similar to each other than to objects in the other clusters. Since clustering is an ill-posed and computationally intractable problem, many methods have been introduced, in particular for graph-clustering (see e.g. the recent review by Fortunato, 2010). Effective methods for graph-clustering contain a parameter whose tuning affects the community structure at multiple resolution scales. These methods are also called multi-resolution methods (see e.g. Lambiotte, 2010). The resolution parameter(s) can be used in two main ways: as a parameter to be tuned; or as a way to generate clusterings at multiple resolution scales, which can then be used to analyze the clustering behavior of objects across multiple resolutions (Lewis *et al.*, 2010), or to ensemble the results to produce a consensus clustering (Ronhovde and Nussinov, 2009).

In Chapter 4, the resolution bias of state-of-the-art community detection methods has been analyzed, and a simple yet effective quality function was introduced. Results indicated that the Louvain method, based on greedy local search optimization, is robust to the choice of the clustering quality function, when a multi-resolution parameter is added to the quality function.

The goal of this chapter is to investigate experimentally the performance of such multi-resolution methods when applied to PPI networks, with respect to data generated from different laboratory technologies as well as with respect to random removal or shuffling of edges in the network. This latter investigation is motivated by the fact that PPI data are still not fully reliable, with the potential inclusion of both false positive and false negative interactions (see e.g. the

discussion in X. Li *et al.*, 2010). Specifically, we consider the Louvain method for optimizing two different quality functions incorporating a resolution parameter: modularity (Girvan and Newman, 2002) and a the w-log-v introduced in Chapter 4.

To analyze their performance we consider the yeast Saccharomyces cerevisiae, which is a well studied model organism for higher eukaryotes with several protein interaction data generated from diverse laboratory technologies. Specifically, we consider six PPI networks from past studies and the present BioGRID curated database of protein interactions (Stark *et al.*, 2006). In order to assess robustness with respect to random perturbations of the graph, we generate a large collection of networks using the BioGRID data, by either removing or by adding a percentage of randomly selected edges, or by randomly shuffling edges while keeping the original degree of each node.

Results of the experiments indicate improved performance of modularity and w-log-v over MCL (the Markov Cluster Algorithm) (Van Dongen, 2008), a state-of-the art method for community detection in PPI networks based on stochastic flow in graphs. MCL was found to achieve best overall performance in yeast PPI networks (Brohee and van Helden, 2006) and competitive performance with methods for overlapping community detection in PPI networks (Nepusz, Yu, and Paccanaro, 2012).

In particular best performance is achieved by w-log-v on the BioGRID data, and excellent robustness on randomly perturbed versions of this network. Since PPI networks are known to be noisy with respect to the presence of both false positive and false negative interactions, the high robustness shown by the proposed algorithm substantiates its effectiveness on this type of data.

5.1.1 Related work

A vast literature on protein complex detection with PPI networks exists (see e.g. the review X. Li *et al.*, 2010). Previous related works on multi resolution algorithms for clustering PPI networks either apply an algorithm multiple times with different values of the resolution parameter in order to investigate how proteins cluster at different resolution scales, e.g. Lewis *et al.* (2010), or tune the resolution parameter in order to choose a best setting for the considered type of networks, e.g. Brohee and van Helden (2006) and Nepusz, Yu, and Paccanaro (2012). Here we aim at investigating thoroughly effectiveness and robustness of two such algorithms by means of an extensive experimental analysis.

In Brohee and van Helden (2006) a comparative assessment of clustering algorithms for PPI networks was conducted. In particular, robustness was analyzed, with respect to alterations (addition and/or removal of randomly selected

edges) of a test graph which was constructed using a number of yeast complexes annotated in the MIPS database, by linking each pair of proteins belonging to the same complex. The considered algorithms with parameters tuned on the test graph, were then applied to various yeast datasets. Results showed that MCL with inflation (resolution) parameter value equal to 1.8 was performing best on the considered datasets. Robustness of MCL when applied to yeast PPI networks has previously also been analyzed in Pu *et al.* (2007). According to their results, MCL is rather robust across different networks and with respect to missing or noisy information on protein-protein associations. Here we show that greedy local search optimization of a clustering quality function (e.g. w-log-v) incorporating a resolution parameter achieves improved robustness (and accuracy) on the BioGRID data.

5.2 Methods

In Chapter 4 we showed that the Louvain method works well for finding clusters in networks. This optimization method is independent of the quality function that is optimized. Hence we are essentially free to choose the quality function to best fit the application. In this chapter we will limit ourselves to two of the quality functions that were discussed in the previous chapter. The first is the popular modularity (Girvan and Newman, 2002),

$$\text{modularity}(C) = \sum_{c \in C} \left(\widehat{w}_c - \widehat{v}_c^2 \right).$$

Here C denotes a clustering, i.e. a set of clusters. For a particular cluster $c \in C$, its volume is $v_c = \sum_{i \in c, j \in V} a_{ij}$, i.e. the sum of the weight of edges incident to nodes in c, which is equivalent to the sum of degrees. Based on the volume we define the normalized volume as $\widehat{v}_c = v_c / v_V$, where V is the set of all nodes. Finally $w_c = \sum_{i,j \in c} a_{ij}$ is the within cluster volume, and $\widehat{w}_c = w_c / v_V$ is its normalized variant.

The second quality function we consider is w-log-v, which was introduced in Chapter 4. An advantage of this quality function over modularity is that it allows more diverse cluster sizes. Because the sizes of protein complexes can differ widely, we believe that this is a useful property. The w-log-v quality function is defined as

$$\text{w-log-v}(C) = -\sum_{c \in C} \widehat{w}_c \log(\widehat{v}_c).$$

Using either of the above quality functions directly for the task of clustering a PPI network is not advisable. Both quality functions were designed for community detection; and communities are usually relatively large, much larger than

protein complexes. Therefore, optimizing these quality functions will lead to a clustering with a small number of large clusters. This inability to find small clusters is termed the resolution limit of clustering (Fortunato and Barthélemy, 2007).

To overcome the resolution limit, we add a parameter to the quality functions as follows,

$$\text{modularity}_\alpha(C) = \text{modularity}(C) - \alpha \sum_{c \in C} \widehat{w}_(c),$$
$$\text{w-log-v}_\alpha(C) = \text{w-log-v}(C) - \alpha \sum_{c \in C} \widehat{w}_(c).$$

By increasing the parameter α, the clustering is punished for within cluster edges, and hence the optimal clustering will have smaller clusters. Alternatively, by decreasing the parameter α, the clustering is rewarded for within cluster edges, so the optimal clustering will then have larger clusters.

In Section 4.2.4 we showed that, because the overall scale of the quality function is irrelevant for optimization, the modification is equivalent to assuming that the overall volume of the graph is different. For modularity, the adjustment corresponds to assuming that the graph has volume $(1-\alpha)v_(V)$, while for w-log-v it corresponds to assuming the volume is $e^{-\alpha}v_(V)$. This equivalent interpretation provides some intuition for the resolution parameter: when we tune α to find smaller clusters, the quality function is equivalent to that for finding the clusters in a smaller graph.

5.3 Experiments

5.3.1 PPI networks

We downloaded a set of protein interactions from version 3.1.88[1] of the BioGRID database (Stark *et al.*, 2006). This database contains a collection of protein interactions from different sources, and discovered with different methods. In this work we only consider interactions found by physical experiments, not those based on genetics.

The BioGRID also contains in full several datasets from high throughput experimental studies, including Uetz *et al.* (2000), Ho *et al.* (2002), Gavin, Bosche, *et al.* (2002), Gavin, Aloy, *et al.* (2006), Krogan *et al.* (2006), and Collins *et al.* (2007). We consider these subnetworks as separate datasets in our experiments. These datasets are generated with different experimental techniques: the Collins

[1]This version was released on April 25th, 2012

Table 5.1: Sizes of the different datasets. The last column lists the number of MIPS complexes that are (partially) contained in each dataset.

Dataset	Nodes	Edges	Complexes
Uetz *et al.* (2000)	927	823	20
Ho *et al.* (2002)	1563	3596	43
Gavin, Bosche, *et al.* (2002)	1352	3210	50
Gavin, Aloy, *et al.* (2006)	1430	6531	53
Krogan *et al.* (2006)	2674	7075	75
Collins *et al.* (2007)	1620	9064	63
BioGRID, all physical	5967	68486	97

(Collins *et al.*, 2007), Krogan (Krogan *et al.*, 2006) and Gavin (Gavin, Aloy, *et al.*, 2006) datasets include the results of TAP tagging experiments only, while the BioGRID dataset contains a mixture of TAP tagging, Y2H and low-throughput experimental results (Nepusz, Yu, and Paccanaro, 2012). Table 5.1 lists the sizes of these datasets.

5.3.2 Complex validation

For validation, we compare clusters with the complexes from the MIPS database (Mewes *et al.*, 2002)[2], which we take as the gold standard. MIPS specified a hierarchy of complexes and subcomplexes. Since we deal only with non-overlapping clustering, we only include (sub)complexes at the bottom of this hierarchy. And to avoid degenerate cases, we include only complexes with at least 3 proteins. We also exclude the complexes in category 550, since these are unconfirmed. They were found with computational clustering methods, using as input the same high throughput datasets that we consider.

In addition to the complexes from MIPS, we also use a set of complexes derived from the Gene Ontology annotations of the Saccharomyces Genome Database (Cherry *et al.*, 2011). This dataset was created and also used by (Nepusz, Yu, and Paccanaro, 2012).

To compare clusters found by a method to either of these sets of gold standard complexes, we use the overlap score (Brohee and van Helden, 2006),

$$\omega(A, B) = \frac{|A \cap B|^2}{|A||B|}.$$

We consider a cluster to *match* a complex if their overlap score is at least 0.25. This threshold is also used in other works, e.g. (Nepusz, Yu, and Paccanaro,

[2]We used the latest version at the time of writing, which was released on May 18th, 2006

2012; Chua *et al.*, 2008). When the cluster and complex have the same size, a match then corresponds to the intersection containing at least half of the nodes in the complex and cluster.

Based on this matching we define *precision* as the fraction of clusters that are matched to any complex. Conversely, we define *recall* as the fraction of complexes that are matched to any cluster. Note that we use the terminology from other works such as Chua *et al.* (2008). These notions differ from the more standard definitions of precision and recall, because a cluster can match more than one complex and vice versa.

It is clearly possible to achieve a high precision or a high recall with a degenerate clustering. For example, by returning just a single easy to find cluster that matches a complex, the precision will be 1 at the cost of a low recall. And by returning all possible (overlapping) clusters, the recall will be 1 at the cost of a low precision. We therefore use the F_1 score, which is the harmonic mean of precision and recall, as a trade-off between the two scores.

For each of the methods, we include only clusters that contain at least 3 proteins. As a result, not all proteins will be in a cluster. We call the fraction of proteins that are in a cluster the *coverage* of a clustering.

The precision and recall as defined above depend heavily on the chosen threshold; and when few complexes are matched, the scores are very sensitive to noise. Therefore, we also look at the positive predictive value (PPV) and cluster-wise sensitivity scores (Brohee and van Helden, 2006), which are based directly on the size of the intersection between complexes and clusters,

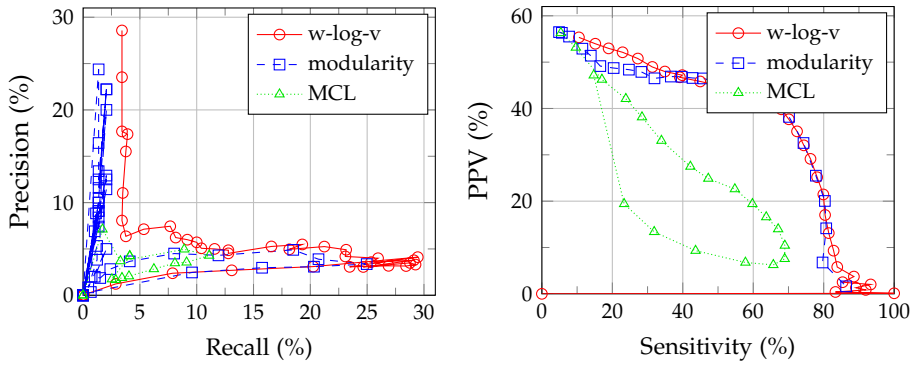
$$\text{PPV} = \frac{\sum_{A \in C} \max_{B \in C*} |A \cap B|}{\sum_{A \in C} \sum_{B \in C*} |A \cap B|} \qquad \text{Sensitivity} = \frac{\sum_{B \in C*} \max_{A \in C} |A \cap B|}{\sum_{B \in C*} |B|},$$

where C is the set of predicted clusters and $C*$ is the set of gold standard complexes. Note that the asymmetry between the denominators is to account for the case of overlapping clusters.

5.3.3 Precision vs. recall

We took the BioGRID all physical dataset, and computed the precision and recall for a wide range of settings of the resolution control parameter α. These results are shown in Figure 5.1 (left). For comparison we also include results with the MCL algorithm for different settings of the inflation parameter. For readability we have applied smoothing in the form of merging points that are very close together.

The first thing that we observe is that despite smoothing, the figure is very noisy in some places. This is not very surprising considering how precision and



(a) Precision vs. Recall

(b) Sensitivity vs. PPV

Figure 5.1: Precision vs. Recall (left) and Sensitivity vs. PPV (right) on the BioGRID dataset.

recall are calculated. Consider a small change in the clustering, such as removing a protein from a cluster. This change might cause the cluster to no longer match a particular complex. If there are no other clusters that matched that complex, then the recall goes down, otherwise it stays the same. Similarly, if this are no other complexes matching the cluster, then the precision goes down. While obviously the change in the two scores is related, the relation is not monotonic, one can change while the other does not.

As the resolution control parameter α goes up, the methods find more clusters; and as a result the recall goes up while the precision goes down. However, after a certain point many of the clusters will become too small, and they will be removed before matching. This decreased coverage causes the recall to go down again.

To get a less noisy picture, we have also plotted the PPV and sensitivity scores, in Figure 5.1 (right). The overall trend in this plot is the same as for the precision and recall: the w-log-v method slightly dominates modularity optimization, which in turn has significantly better results than MCL.

The best parameter settings according to the F_1 score are $\alpha = 2.8$ for w-log-v, $\alpha = 0.97$ for modularity, and inflation 2.7 for MCL. We will use these settings for the remainder of the experiments. As discussed in Section 5.2, the parameter α corresponds to assuming a different volume of the graph. The optimal setting for w-log-v corresponds to considering a graph with 16 times fewer edges, while the optimal setting for modularity corresponds to 33 times fewer edges. The difference between the two quality functions comes from their inherent res-

olution bias, by default w-log-v has a bias towards smaller clusters compared to modularity, and therefore the quality function needs less adjustment.

5.3.4 Networks from Single Studies

We next compare the scores on the subnetworks from single studies. The results of this experiment are shown in Table 5.2. The "MIPS method" is based on the gold standard complexes, but including only proteins that occur in the dataset under investigation. It represents the best possible scores.

On most datasets w-log-v has the best recall and F_1 score, except on the datasets from Gavin et al. Gavin, Bosche, *et al.* (2002) and Gavin, Aloy, *et al.* (2006), where MCL performs significantly better. The precision of modularity optimization is often slightly better than that for w-log-v optimization. This is due to the fact that with the settings chosen in the previous paragraph, we find more clusters with w-log-v optimization. Hence, in general recall will be higher at the cost of lower precision.

5.3.5 Randomly perturbed graphs

To further test the robustness of the methods, we applied them to randomly perturbed networks. We performed three different experiments, all starting from the BioGRID network.

1. Removing a randomly chosen subset of the interactions.
2. Randomly adding new spurious interactions between pairs of proteins.
3. Randomly rewire a subset of the edges, while maintaining the degree of each node. Note that such a move both removes an observed interaction and adds a new spurious one.

We varied the amount of edges affected by each type of perturbation. Each experiment was repeated 10 times with different seeds for the random number generator, we calculated the mean and standard deviation of the F_1 score across these repetitions. The results are shown in Figure 5.2. When edges are removed, the performance of all methods degrades similarly. On the other hand, the Louvain method results are much more robust to the addition of extra edges than MCL. Also note that the standard deviation is much larger with the MCL method. That means that for some rewired graphs the method gives reasonably good results, while for others the result is very bad. Unsurprisingly, the experiment with rewired edges sits somewhere in between the two other experiments.

Table 5.2: Results of applying the different methods to subnetworks for single studies. The best result for each dataset is highlighted in bold.

Method	Clusters	Coverage	Precision	Recall	Sens.	PPV	F_1
			Uetz *et al.* (2000)				
MIPS	20	2.5%	85.0%	11.6%	9.3%	69.1%	20.5%
w-log-v	173	21.1%	4.6%	**6.2%**	6.7%	73.6%	**5.3%**
modularity	160	**21.8%**	**5.0%**	5.5%	**7.0%**	70.1%	5.2%
MCL	143	17.2%	4.2%	4.1%	5.3%	**75.0%**	4.2%
			Ho *et al.* (2002)				
MIPS	43	5.2%	88.4%	26.0%	17.5%	67.8%	40.2%
w-log-v	278	46.0%	2.5%	**6.8%**	11.8%	**69.3%**	3.7%
modularity	257	**46.2%**	**2.7%**	6.2%	**12.0%**	67.4%	**3.8%**
MCL	227	35.5%	1.8%	2.7%	10.7%	64.4%	2.1%
			Gavin *et al.* (2002)				
MIPS	50	7.9%	92.0%	32.9%	27.9%	62.5%	48.4%
w-log-v	202	**39.7%**	12.4%	**21.2%**	**21.6%**	**72.0%**	15.6%
modularity	199	39.0%	11.6%	19.9%	19.9%	69.6%	14.6%
MCL	177	34.5%	**14.7%**	**21.2%**	21.4%	71.1%	**17.4%**
			Gavin *et al.* (2006)				
MIPS	53	7.8%	92.5%	34.9%	28.3%	63.6%	50.7%
w-log-v	193	**40.9%**	13.5%	21.2%	23.3%	**72.3%**	16.5%
modularity	188	37.9%	13.3%	19.9%	22.6%	70.6%	15.9%
MCL	164	33.4%	**16.5%**	**21.9%**	**24.0%**	71.6%	**18.8%**
			Krogan *et al.* (2006)				
MIPS	75	8.6%	97.3%	50.7%	38.3%	66.1%	66.7%
w-log-v	401	58.9%	8.0%	**25.3%**	26.6%	72.6%	12.1%
modularity	314	**60.4%**	**9.6%**	23.3%	**26.8%**	70.3%	**13.5%**
MCL	380	43.9%	7.4%	21.2%	22.4%	**73.5%**	10.9%
			Collins *et al.* (2007)				
MIPS	63	8.9%	98.4%	43.8%	32.9%	65.6%	60.7%
w-log-v	194	**38.8%**	20.1%	**32.2%**	27.3%	**75.1%**	**24.8%**
modularity	177	34.3%	20.3%	29.5%	26.7%	72.8%	24.1%
MCL	172	36.8%	**20.9%**	30.1%	**29.1%**	69.9%	24.7%
			BioGRID, all physical				
MIPS	97	7.9%	96.9%	64.4%	55.3%	70.2%	77.4%
w-log-v	505	**84.5%**	**5.9%**	**22.6%**	**38.6%**	69.4%	**9.4%**
modularity	599	83.6%	4.2%	19.2%	35.6%	**72.1%**	6.9%
MCL	283	69.0%	5.3%	11.6%	28.7%	40.6%	7.3%

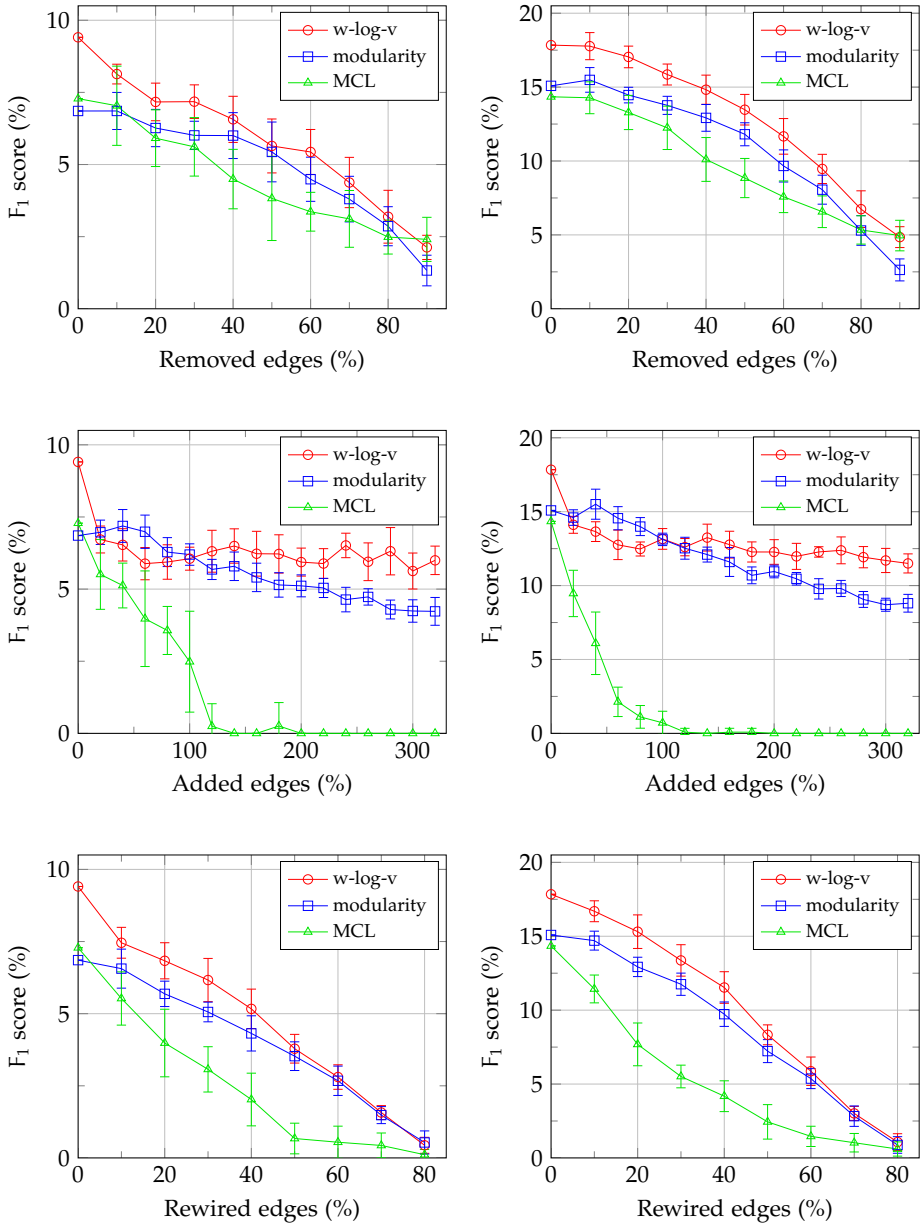


Figure 5.2: F_1 score when a fraction of the edges is added (top), removed (middle) or rewired (bottom) at random in the BioGRID dataset. Error bars indicate standard deviation, measured over 10 runs. The left plots use MIPS complexes as the gold standard, the right plots use SGD complexes.

5.4 Discussion

Because of the incompleteness of both the PPI data and of knowledge on true complexes, care must be taken in the interpretation of the results. The "MIPS method", that is, the best possible method based on the MIPS complexes, covers only a small part of the proteins present in each of the datasets. Conversely, not all MIPS complexes are covered by the datasets, so the recall is always smaller than the precision. In general, results show that for each dataset, the majority of clusters induced by the intersection of that dataset with the complexes in MIPS, match a complex; with a percentage varying between 85% and 98%. These values provide upper bounds on the maximum precision and recall achievable on the considered dataset.

On all datasets the algorithms obtain precision smaller than recall: the difference of these values provides information on the fraction of clusters matching more than one complex. For instance on the Uetz dataset, there is almost a one-to-one correspondence between clusters and matched complexes (e.g. 4.6% precision and 6.2% recall for w-log-v), while on the BioGRID this relation is clearly one-to-many (e.g. 5.9% precision and 22.6% recall for w-log-v).

There are complexes that are matched by only one method: specifically, 6 complexes are matched only by w-log-v, 5 only by modularity, and 4 only by MCL. Comparing w-log-v and MCL, there are 18 complexes found only by w-log-v and 4 found only by MCL. This is not too surprising, since MCL has a rather low recall. An example of a complex detected by w-log-v and not by MCL is the Signal recognition particle (SRP) complex, consisting of six proteins, one of the complexes involved in Transcription and/or in the Nucleus.

The improved performance of w-log-v on the BioGRID data appears mainly due to its capability to generate a large number of clusters matching multiple complexes (high recall). Nevertheless, Figure 5.1 shows that the precision vs. recall curve of w-log-v dominates the curve of the other two methods.

Robustness of a community detection method is and important issue also in the context of PPI networks, since they are known to contain a high amount of false negative and false positive interactions. Indeed, limitations of experimental techniques as well as the dynamic nature of protein interaction are responsible for the high rate of false-positives and false-negatives generated by high-throughput methods. For instance, Y2H screens have false negative rates in the range from 43% to 71% and TAP has false negative rates of 15%-50%, and false positive rates for Y2H could be as high as 64% and for TAP experiments they could be as high as 77% (Edwards *et al.*, 2002). Results show that w-log-v achieves best robustness the under random addition, removal and rewiring of a percentage of edges in the BioGRID network. Such high robustness substantiates

the effectiveness of w-log-v on this type of data.

5.5 Conclusions

This chapter analyzed the performance of two fast algorithms on PPI networks that optimize in a greedy way a clustering quality function with resolution parameter. An extensive experimental analysis was conducted on PPI data from previous studies as well as on the present BioGRID database. Results indicated improved performance of the considered algorithms over a state-of-the-art method for complex detection in PPI networks, in particular with respect to robustness. These results indicate that the considered algorithms provide an efficient, robust and effective approach for protein complex discovery with PPI networks. Interesting issues for future work include the assessment of the algorithms' robustness with respect to tailored models of false positive and false negative interactions which are present in data generated by specific technologies, as well as the extension of the considered methods to detect overlapping clusters of high quality.

Chapter 6

Axioms for soft clustering and Non-negative Matrix Factorization

Many graph clustering quality functions suffer from a resolution limit, the inability to find small clusters in large graphs. So called resolution-limit-free quality functions do not have this limit. This property was previously introduced for hard clustering, that is, graph partitioning.

We investigate the resolution-limit-free property in the context of Non-negative Matrix Factorization (NMF) for hard and soft graph clustering. To use NMF in the hard clustering setting, a common approach is to assign each node to its highest membership cluster. We show that in this case symmetric NMF is not resolution-limit-free, but that it becomes so when hardness constraints are used as part of the optimization. In soft clustering, nodes can belong to more than one cluster, with varying degrees of membership. In this setting resolution-limit-free turns out to be too strong a property. Therefore we introduce locality*, which roughly states that changing one part of the graph does not affect the clustering of other parts of the graph. We argue that this is a desirable property, provide conditions under which NMF quality functions are local, and propose a novel class of local probabilistic NMF quality functions for soft graph clustering.*

6.1 Introduction

Graph clustering, also known as network community detection, is an important problem with real-life applications in diverse disciplines such as life and social sciences (Schaeffer, 2007; Fortunato, 2010). Graph clustering is often performed by optimizing a quality function, which is a function that assigns a score to a

This chapter is based on"Local quality functions for graph clustering with Non-negative Matrix Factorization", accepted for publication in Physical Review E.

clustering. During the last decades, many such functions (and algorithms to optimize them) have been proposed. However, relatively little effort has been devoted to the theoretical foundation of graph clustering quality functions, e.g. Ackerman and Ben-David, 2008. In this chapter we try to provide a contribution in this direction by studying desirable locality properties of NMF quality functions for hard and soft graph clustering.

We focus on the resolution-limit-free property, a property of hard graph clustering, recently introduced by Traag, Van Dooren, and Nesterov (2011). Informally this property states that a subset of an optimal clustering in the original graph should also be an optimal clustering in the induced subgraph containing only the nodes in the subset of clusters. As the name suggests, resolution-limit-free quality functions do not suffer from the so-called resolution limit, that is, the inability to find small clusters in large graphs. In the seminal work by Fortunato and Barthélemy (2007), it was shown that modularity (Newman and Girvan, 2004), a popular quality function used for network community detection, has a resolution limit, in the sense that it may not detect clusters smaller than a scale which depends on the total size of the network and on the degree of interconnectedness of the clusters.

Our goal is to investigate resolution-limit-freeness and other locality properties of Non-negative Matrix Factorization (NMF) graph clustering quality functions. NMF (Paatero and Tapper, 1994; Lee and Seung, 1999) is a popular machine learning method initially used to learn the parts of objects, like human faces and text documents. It finds two non-negative matrices whose product provides a good approximation to the input matrix. The non-negative constraints lead to a parts-based representation because they allow only additive, not subtractive, combinations. Recently NMF formulations have been proposed as quality functions for graph clustering, see for instance the surveys Wang *et al.* (2011) and T. Li and C. H. Q. Ding (2013).

We consider symmetric and asymmetric NMF formulations based on Euclidean loss and a Bayesian NMF quality function recently proposed by Psorakis *et al.* (2011), which can automatically determine the number of clusters.

The resolution-limit-free property is stated in the setting of hard clustering, where a clustering is a partition of the nodes. In contrast, NMF produces a soft clustering. Nodes have varying degrees of memberships of each clusters, and the clusters can overlap. To use NMF in the hard clustering setting, a common approach is to assign each node to its highest membership cluster.

In Section 6.3 we show that hard clustering based on NMF in this way is, in general, not resolution-limit-free. For symmetric NMF we show that resolution-limit-freeness can be obtained by using orthogonality constraints as part of the optimization, and that the resulting function is strongly linked to the Constant

Potts Model (CPM). CPM was introduced by Traag, Van Dooren, and Nesterov as the simplest formulation of a (non-trivial) resolution-limit-free method. It is a variant of the Potts model by Reichardt and Bornholdt (2004).

We argue in Section 6.4 that in the soft clustering setting, resolution-limit-freeness is a too strong property and propose an alternative desirable locality property for soft graph clustering. We characterize an interesting class of local quality functions and show that symmetric and asymmetric NMF belong to this class. We show that Bayesian NMF is not local in general and that it suffers from a resolution limit. In Section 6.5 we introduce a novel class of probabilistic NMF quality functions that are local, and hence do not suffer from a resolution limit.

6.1.1 Related work

The notion of resolution limit was introduced in Fortunato and Barthélemy (2007), which detected a limitation of modularity, considered a state-of-the-art method for community detection. In Chapter 4 we showed empirically that the resolution limit is the most important difference between quality functions in graph clustering optimized using a fast local search algorithm, the Louvain method (Blondel *et al.*, 2008). Traag, Van Dooren, and Nesterov (2011) introduced the notion of resolution-limit-free objective functions, which provides the motivation of this study.

Other local properties of quality functions for clustering have been considered in theoretical studies but mainly in the hard setting, for distance based clustering in Ackerman, Ben-David, and Loker (2010a) and for graph clustering in Chapter 3 of this thesis. Locality as defined in Ackerman, Ben-David, and Loker (2010a) is a property of clustering functions, therein defined as functions mapping a data set and a positive integer k to a partition of the data into k clusters. This notion of locality was used together with other properties to characterize linkage based clustering. The weak locality property considered in Chapter 3 is part of an axiomatic study of quality functions for hard graph clustering. It states that local changes to a graph should have only local consequences to a clustering. It is slightly weaker than the locality property considered in this study, which corresponds more closely to the property there called strong locality.

6.1.2 Definitions and Notation

In this chapter we reuse the definition of graphs defined in Section 3.2. But the notion of clustering will be different.

As before, a (weighted) *graph* is a pair (V, A) of a finite set V of nodes and a function $A : V \times V \to \mathbb{R}_{\geqslant 0}$ of edge weights; and we use the abbreviation $a_{ij} = A(i, j)$. Edges with larger weights represent stronger connections, so $a_{ij} = 0$ means that there is no edge between nodes i and j.

A graph $G' = (V', A')$ is a *subgraph* of $G = (V, A)$ if $V' \subseteq V$ and $a'_{ij} = a_{ij}$ for all $i, j \in V'$.

Different clustering methods use different notions of a 'cluster' and of a 'clustering'. For instance, in symmetric NMF a clustering is a matrix of membership coefficients; while in non-symmetric NMF there are two such matrices. Some methods also have additional parameters for each cluster. In this chapter we allow different types of 'cluster' for different methods, but we use a common definition of 'clustering'.

Formally, each of these types of clusters can be specified by an injective function $\mathcal{C}$ from sets of nodes to sets of things which we call clusters. For a set of nodes s, for every cluster $c \in \mathcal{C}(s)$ we call s the *support* of c, written as $\operatorname{supp}(c) = s$. The set of all clusters with support on a subset of V is $\mathcal{C}^*(V) = \bigcup_{s \subseteq V} \mathcal{C}(s)$. In this paper we consider four types of clusters, which will be introduced in the next section.

A *clustering* of V is a multiset of clusters with support on a subset of V. Note that we use multisets instead of sets, to allow a clustering to contain two identical copies of the same cluster. For brevity, we also say that C is a clustering of a graph G if C is a clustering of the nodes of G. If, in a slight abuse of notation, we define the support of a clustering as the union of the support of all clusters in that clustering, then the clusterings of V are those multisets of clusters for which the support is a subset of V.

Note that this general definition implies that for certain clusterings the clusters can overlap, and some nodes can be in no cluster at all. We believe that this is a reasonable definition, because if we allow nodes to be in more than one cluster, there is little reason to not also allow them to be in less than one cluster.

Additionally, if C and D are clusterings of G, then their multiset sum $C \uplus D$ is also a clustering of G[1], as is any subclustering (submultiset) of C. And if G is a subgraph of G', then C and D are also clusterings of G'.

The symmetric difference of two clusterings is denoted $C \triangle D$, and is defined as the symmetric difference of multisets, that is $C \triangle D = (C \setminus D) \cup (D \setminus C)$.

As in the rest of this thesis, graph clustering is cast as an optimization problem. The objective that is being optimized is the *clustering quality function*, which is a function from graphs G and clusterings of G to real numbers. In this thesis we take the convention that the quality is maximized.

[1] $C \uplus D$ denotes multiset sum, the multiplicity of c in $C \uplus D$ is the sum of multiplicities of c in C and of c in D.

Given a clustering quality function Q, and a clustering C of some graph G. We say that C is Q-*optimal* if $Q(G, C) \geqslant Q(G, C')$ for all clusterings C' of G.

6.2 Non-negative Matrix Factorization

At its core, Non-negative Matrix Factorization (Paatero and Tapper, 1994; Lee and Seung, 1999) decomposes a matrix A as a product $A \approx WH^T$, where all entries in W and H are non-negative. For graph clustering the matrix A is the adjacency matrix of a graph. For undirected graphs the adjacency matrix is symmetric, in which case it makes sense to decompose it as $A \approx HH^T$. Note that such a symmetric factorization has to be enforced explicitly, since the optimal non-symmetric factorization of a symmetric matrix does not necessarily have $W = H$ (Catral *et al.*, 2004).

The columns of W and H can be interpreted as clusters. To fit with the definitions from the previous paragraph we need to take a slightly different view. In the case of symmetric NMF, a cluster with support s is a function that assigns a positive real number to each node in s, so $\mathcal{C}_{\text{SymNMF}}(s) = \mathbb{R}^s_{>0}$. Equivalently, for a fixed set of nodes, we can represent a cluster as a vector of non-negative numbers with an entry for each node in V, such that the entries for the nodes not in s are zero, that is, $\mathcal{C}^*_{\text{SymNMF}}(V) \approx \mathbb{R}^V_{\geqslant 0}$. For a cluster c we denote this vector as h_c, and a multiset of such vectors can be seen as a matrix H. The support of c then coincides with the standard notion of support of the vector h_c, that is, the set s of nodes for which the entry is non-zero. This representation of clusters in terms of a non-negative vector h_c is more standard and more convenient than the one in terms of a function from s to positive real numbers, and we use it in the rest of the paper.

For non-symmetric NMF, a cluster is a tuple $c = (w_c, h_c)$ of two such vectors. That is, $\mathcal{C}^*_{\text{AsymNMF}}(V) = \mathbb{R}^V_{\geqslant 0} \times \mathbb{R}^V_{\geqslant 0}$, with $\text{supp}((w_c, h_c)) = \text{supp}(w_c) \cup \text{supp}(h_c)$. For Bayesian NMF (Psorakis *et al.*, 2011) each cluster also contains a β_c parameter. That is, $\mathcal{C}^*_{\text{BayNMF}}(V) = \mathbb{R}^V_{\geqslant 0} \times \mathbb{R}^V_{\geqslant 0} \times \mathbb{R}_{>0}$.

A common notion to all NMF methods is that they predict a value for each edge. For symmetric NMF with per cluster membership vector h_c this prediction can be written as $\hat{a}_{ij} = \sum_{c \in C} h_{ci} h_{cj}$. For asymmetric NMF with cluster memberships w_c and h_c we can write $\hat{a}_{ij} = \sum_{c \in C} w_{ci} h_{cj}$.

The optimization problem then tries to ensure that $\hat{a}_{ij} \approx a_{ij}$. Different methods can have different interpretations of the '$\approx$' symbol, and they impose different regularizations and possibly additional constraints. Perhaps the simplest NMF quality function for undirected graphs uses Euclidean distance and no

additional regularization,

$$Q_{\text{SymNMF}}(G, C) = -\frac{1}{2} \sum_{i,j \in V} (a_{ij} - \hat{a}_{ij})^2.$$

6.3 Resolution-limit-free functions for hard clustering

Before we investigate the resolution limits of NMF, we will first look at traditional 'hard' clustering, where each node belongs to exactly one cluster. In this setting a cluster is simply a subset of the nodes, and its support is the cluster itself. That is, $\mathcal{C}_{\text{hard}}(s) = s$. There is the additional non-overlapping or orthogonality constraint on clusters: In a valid hard clustering C of V, each node $i \in V$ is in exactly one cluster $c_i \in C$. For symmetric NMF we may formulate these constraints as

$$\sum_{i \in V} h_{ci} h_{di} = 0 \qquad \text{for all } c, d \in C, c \neq d, \text{and}$$

$$\sum_{c \in C} h_{ci} = 1 \qquad \text{for all } i \in V.$$

Traag, Van Dooren, and Nesterov (2011) introduced a locality property of clustering quality functions, and called the functions that satisfy this property *resolution-limit-free*. Their definition is as follows.

Definition 6.1 (Resolution-limit-free)**.** *Let* C *be a* Q*-optimal clustering of a graph* G_1*. Then the quality function* Q *is called* resolution-limit-free *if for each subgraph* G_2 *induced by* $D \subset C$*, the partition* D *is a* Q*-optimal clustering of* G_2*.*

Thus in the setting of hard clustering, a quality function is resolution-limit-free if any subset of clusters from an optimal clustering is also an optimal clustering on the graph that contains only the nodes and edges in those clusters.

NMF has been extended with a post-processing step to yield a hard clustering. This is done by assigning each node to the cluster with the largest membership coefficient.

We can now ask if NMF with this post-processing is resolution-limit-free. In Figure 6.1 we give a counterexample that answers this question negatively for the NMF based methods of Psorakis *et al.* (2011) and C. Ding, He, and Simon (2005).

This counterexample consists of two cliques and one almost-clique. Additionally there is a node with unclear membership. When the entire graph is considered, its membership of one cluster is slightly higher, when one clique and its incident edges are removed, its membership of another cluster is slightly

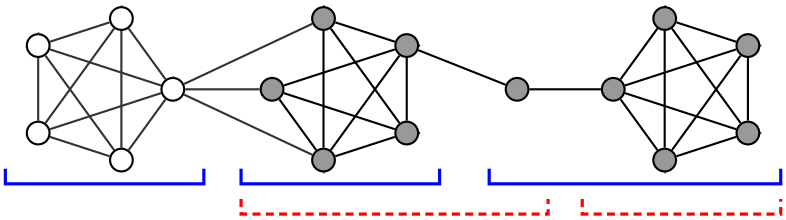


Figure 6.1: A counterexample that shows that NMF quality functions are not resolution limit free. When considering the entire graph, the first (solid blue) clustering is optimal. When considering only the gray nodes, the second (dashed red) clustering is optimal. The membership of the middle node is very unclear, it belongs to two clusters to almost the same degree. When another part of a cluster changes this can tip the balance one way or the other.

higher. This difference is very small. For example, with C. Ding, He, and Simon's method in the optimal clustering of the large graph, the disputed node belongs to the second and third cluster with membership coefficients 0.2306 and 0.2311 respectively; while in the smaller subgraph the membership coefficients are 0.2284 and 0.2607.

Traag, Van Dooren, and Nesterov (2011) showed that the Constant Potts model (CPM) is the simplest formulation of any (non-trivial) resolution-limit-free method. The CPM quality function $Q_{\text{cpm}}(G, C)$ can be formulated as

$$Q_{\text{cpm}}(G, C) = \sum_{i,j \in V} (a_{ij} - \gamma) \mathbf{1}[c_i = c_j],$$

where $\mathbf{1}[c_i = c_j]$ is 1 if nodes i and j belong to the same cluster, and 0 otherwise.

Symmetric NMF and CPM are closely related. This can be shown with a technique similar to that used by C. Ding, He, and Simon (2005) to link symmetric NMF and spectral clustering.

Theorem 6.2. *Symmetric NMF is an instance of CPM with $\gamma = 1/2$ and orthogonality constraints relaxed.*

The proof of this theorem can be found in Appendix 6.A.

The CPM *is* resolution-limit-free. Therefore in order to perform hard clustering using symmetric NMF it is preferable to act on the quality function, for instance by enforcing orthogonality as done in (C. Ding, He, and Simon, 2005; C. Ding, T. Li, *et al.*, 2006), instead of assigning each node to the cluster with the highest membership coefficient.

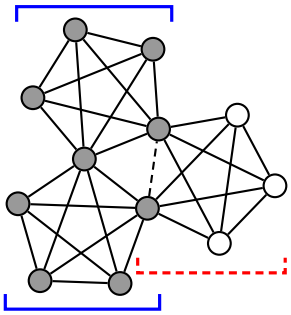


Figure 6.2: Three cliques sharing two nodes each. The obvious clustering consists of three overlapping clusters, with the three central nodes in two clusters each. The white nodes are not in the support of the solid blue clusters.

6.4 Resolution-limit-free functions for soft clustering

We could still try to directly adapt Definition 6.1 to the soft clustering setting, by defining what a graph induced by a subclustering is. The obvious idea is to include all nodes in the support of the subclustering. So for a clustering C of G, the graph G' induced by $D \subseteq C$ would contain only the nodes which are in at least one cluster in D, that is, $V' = \text{supp}(D)$, and all edges between these nodes from the original graph.

However, in contrast to the hard clustering case, an optimal soft clustering might have clusters in $C \setminus D$ that overlap with clusters in D. This makes the notion of resolution-limit-free too restrictive, since it effectively disallows any interesting uses of overlapping clusters.

Consider the graph with three overlapping 5-cliques shown in Figure 6.2. In an NMF style method such as (C. Ding, He, and Simon, 2005), the optimal clustering of this graph will have three overlapping clusters, corresponding to the three cliques. The subgraph introduced by the support of the solid blue clusters includes just the dark nodes, but neither cluster covers both nodes incident to the dashed edge. Therefore, with these two clusters the prediction $\hat{a}$ for this edge will be 0. But the optimal clustering of this subgraph would have a non-zero prediction for this edge. In other words, the optimal clustering for the induced subgraph is not the same as the solid blue clustering, even the support of the clusters is not the same. Hence no NMF method is resolution-limit-free in this sense.

An alternative approach is to only consider subclusterings with disjoint support in the definition of resolution-limit-free. That is, with $\text{supp}(D) \cap \text{supp}(C \setminus D) = \emptyset$. Unfortunately this variant has the opposite problem: the condition

almost never holds. So, many quality functions would trivially satisfy this variant of resolution-limit-freeness. For example, the optimal clusterings in NMF methods based on a Poisson likelihood will always have overlapping clusters covering every edge, so the disjointness condition only holds when the graph has multiple connected components.

Clearly we need a compromise.

6.4.1 Locality

The resolution-limit-free property looks at the behavior of a clustering quality function on graphs of different sizes. Intuitively a quality function suffers from a resolution limit if optimal clusterings at a small scale depend on the size of the entire graph.

As shown in the previous paragraph we can not just zoom in to the scale of any subclustering D by discarding the rest of the graph.

But if we let go of only considering the optimal clustering, it does become possible to zoom in only partially, leaving the part of the graph covered by clusters that overlap clusters in D intact. If D is an optimal clustering of the original graph, then it should be a 'locally optimal' clustering of the smaller graph in some sense.

We take this to mean that if a clustering D is better than some other clustering D' on the original graph, then the same holds on the smaller graph, as long as D and D' induce the same zoomed in graph.

It then makes sense to not only consider zooming in by discarding the rest of the graph, but also consider arbitrary changes to the rest of the graph, as well as arbitrary changes to clusters not overlapping with D or D'.

More precisely, if one subclustering D is better than another subclustering D' on a subgraph G_S of some graph G_1, and one changes the graph to G_2 in such a way that the changes to the graph and to the clustering are disjoint from this subgraph G_S, then D will stay a better clustering than D'. This idea is illustrated in Figure 6.3.

To formalize this idea we introduce the notion of agreement. We say that two clusterings C_1 of G_1 and C_2 of G_2 *agree* on a common subgraph $G_S = (V_S, A_S)$ of G_1 and G_2 if $\operatorname{supp}(C_1 \triangle C_2) \cap V_S = \emptyset$. Note that this subgraph can be the smallest subgraph containing $\operatorname{supp}(D)$ and $\operatorname{supp}(D')$. This leads to the following definition.

Definition 6.3 (Locality)**.** *A clustering quality function* Q *is* local *if for all graphs* G_1, G_2, *and common subgraphs* G_S *of* G_1 *and* G_2, *for all clusterings* C_1 *of* G_1 *and* C_2 *of* G_2 *that agree on* G_S, *and clusterings* D, D' *of* G_S, *it is the case that* $Q(G_1, C_1 \uplus D) \geqslant Q(G_1, C_1 \uplus D')$ *if and only if* $Q(G_2, C_2 \uplus D) \geqslant Q(G_2, C_2 \uplus D')$.

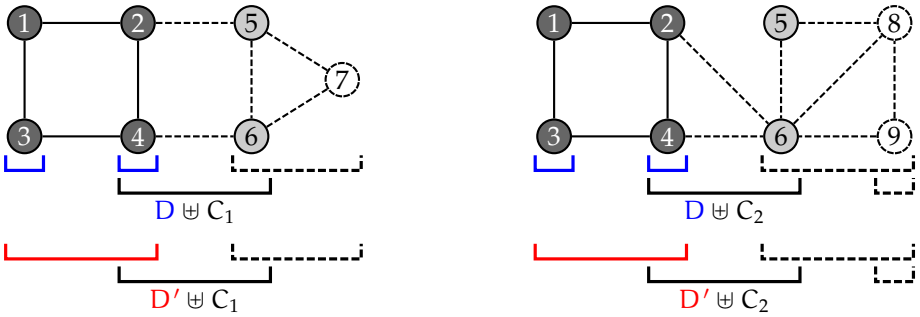


Figure 6.3: An example illustrating locality. Between the left and right side, the dashed part of the clustering and the dashed part of the graph changes. The top and bottom clusterings differ only on the constant part (red or blue), and these differences do not overlap with changing clusters (dashed). Therefore if the top clustering has a higher quality than the bottom clustering on the left graph, then the same must hold on the right graph. Formally, the dark gray nodes are in the common subgraph G_S, the light gray nodes are in $\text{supp}(C_1 \cap C_2)$. The blue clustering is D, the red clustering D', the solid black clusters are in both C_1 and C_2 and the dashed clusters are in only one of C_1 and C_2. Since the dashed clusters don't cover the dark gray nodes, the black clusterings agree on the dark gray subgraph.

Locality as defined in Ackerman, Ben-David, and Loker (2010a) differs from our definition because it is a property of clustering functions. The weak locality property considered in Chapter 3 differs from the definition in this chapter because it also enforces that the graphs agree 'on the neighborhood' of the common subgraph. Here we instead require agreement between overlapping *clusters*. There we also briefly discussed and dismissed a 'strong locality' property, which is the hard clustering equivalent of Definition 6.3.

Even in the case of hard clustering locality and resolution-limit-free are not equivalent. For hard clustering, locality implies resolution-limit-freeness, but the converse is not true.

Theorem 6.4. *If a hard clustering quality function is local, then it is resolution-limit-free.*

Theorem 6.5. *If a hard clustering quality function is resolution-limit-free then it is not necessarily local.*

The proofs of these theorems are in Appendix 6.B and 6.C

6.4.2 Characterizing local quality functions

Many quality functions can be written as a sum with a term for each edge, characterizing a goodness of fit, a term for each node, controlling the amount of overlap, and a term for each cluster, indicating some kind of complexity penalty. There might also be a constant term not actually depending on the clustering, and so not affecting the optimum. We call such quality functions additive.

Definition 6.6. *A qualify function is* additive *if it can be written as*

$$\begin{aligned} Q(G, C) = q_{\mathit{graph}}(G) &+ \sum_{c \in C} q_{\mathit{clus}}(c) \\ &+ \sum_{i \in V} q_{\mathit{node}}\big(\{c \in C \mid i \in \operatorname{supp}(c)\}\big) \\ &+ \sum_{i \in V} \sum_{j \in V} q_{\mathit{edge}}\big(a_{ij}, \{c \in C \mid i, j \in \operatorname{supp}(c)\}\big) \end{aligned}$$

for some functions q_{graph}, q_{clus}, q_{node}, q_{edge}.

Note that q_{node} can depend on all clusters that contain node i, and q_{edge} can depend on all clusters that contain the edge ij.

Theorem 6.7. *If a quality function is additive, then it is local.*

The proof of this theorem can be found in Appendix 6.D. The converse of Theorem 6.7 does not hold; not all local quality functions are additive. For example, any monotonic function of a local quality function is also local.

Another example are quality functions that use higher order interactions, that is, it includes terms not only for nodes and edges, but also for triangles and larger structures. For instance, the clique percolation method (Palla *et al.*, 2005) finds clusters which are cliques. That method is local, but it is not additive. We could imagine including higher-order terms in the definition of additivity,

$$Q(G, C) = \cdots + \sum_{i,j,k \in V} q_{\text{triangle}}(a_{ij}, a_{ik}, a_{jk}, \{c \in C \mid i, j, k \in \operatorname{supp}(c)\}),$$

and so on. But for most purposes the edge term is sufficient; and the local quality functions that we consider in this chapter are all additive in the sense of Definition 6.6.

Additivity provides additional insight into how quality functions behave: the quality is composed of the goodness-of-fit of a the clustering to nodes and edges (and perhaps larger structures), together with a cost term for each cluster. By Theorem 6.7, it also gives us a convenient way to *prove* that a certain quality

function is local, while locality can more convenient if we want to *reason* about the behavior of a quality function.

For symmetric NMF, $\hat{a}_{ij}$ can be written as a sum over clusters that contain nodes i and j,

$$\hat{a}_{ij} = \sum_{c \in C \text{ s.t. } i,j \in \text{supp}(c)} h_{ci} h_{cj}.$$

As a consequence, NMF quality functions without regularization, such as Q_{SymNMF}, are additive. Therefore these quality functions are local.

Many regularization terms can also be encoded in an additive quality function. For example the L2 term $\sum_{c \in C} \sum_{i \in V} h_{ci}^2$ is a sum over clusters and independent of the graph, and so it fits in q_{clus}.

6.4.3 Fixed number of clusters

The question of automatically finding the right number of clusters is still not fully solved. Therefore in most NMF based clustering methods the number of clusters k is specified by the user.

For most quality functions, if they are optimized directly without taking this restriction into account, then the number of clusters will tend to infinity. So we somehow need to fix the number of clusters.

The most direct way to incorporate this restriction of a fixed number of clusters is by adding it as a constraint to the quality function. That is, use $Q(G, C, k) = Q(G, C) + \mathbf{1}[|C| = k]\infty$. Strictly speaking this is not a function to the real numbers. But we never need the fact that Q is such a function, all we need is that the quality of different clusterings can be compared. Unfortunately, encoding a fixed k restriction in the quality function violates locality.

Take two clusterings C and D of a graph G, with a different number of clusters. Let C', D' and G' be copies of C, D and G on a disjoint set of nodes, and let k be $|C| + |D|$. Then the quality $Q(G \cup G', D \uplus C', k)$ is finite, while $Q(G \cup G', D \uplus D', k)$ is infinite. On the other hand, $Q(G \cup G', C \uplus C', k)$ is infinite, while $Q(G \cup G', C \uplus D', k)$ is finite. This contradicts locality.

Instead, we need to consider the restriction on the number of clusters as separate from the quality function. In that case the definition of locality can be used unchanged.

Equivalently, if we call a clustering consisting of k clusters a k-clustering, then we can extend the definitions of locality to take the restricted number of clusters into account. This approach is also used by Ackerman, Ben-David, and Loker (2010a).

If we call a function $Q(G, C, k)$ for graphs G, clusterings C and number of clusters k a *fixed-size quality function*, then this leads to the following fixed-size variant of locality.

Definition 6.8 (Fixed size locality)**.** *A fixed-size quality function* Q *is* fixed-size local *if for all graphs* G_1, G_2 *and a common subgraph* G_S*, for all* k_1*-clusterings* C_1 *of* G_1 *and* k_2*-clusterings* C_2 *of* G_2 *that agree on* G_S*, and* m*-clustering* D *of* G_S *and* m'*-clusterings* D' *of* G_S*, it is the case that* $Q(G_1, C_1 \uplus D, k_1+m) \geqslant Q(G_1, C_1 \uplus D', k_1+m')$ *if and only if* $Q(G_2, C_2 \uplus D, k_2 + m) \geqslant Q(G_2, C_2 \uplus D', k_2 + m')$.

Every local quality function that does not depend on k is fixed-size local when combined with a constraint that the number of clusters must be k. And so NMF with a fixed number of clusters is fixed-size local.

6.4.4 Varying number of clusters

Psorakis *et al.* (2011) formulated a Bayesian formulation of NMF for overlapping community detection that uses automatic relevance determination (ARD) (V. Y. F. Tan and C. Févotte, 2009) to determine the number of clusters. Their quality functions can be written as

$$\begin{aligned} Q_{\text{BayNMF}}(G, C) = &- \sum_{i \in V} \sum_{j \in V} \left(a_{ij} \log \frac{a_{ij}}{\hat{a}_{ij}} + \hat{a}_{ij} \right) \\ &- \frac{1}{2} \sum_{c \in C} \left(\sum_{i \in V} \beta_c w_{ci}^2 + \sum_{i \in V} \beta_c h_{ci}^2 - 2|V| \log \beta_c \right) \\ &- \sum_{c \in C} \left(\beta_c b - (a-1) \log \beta_c \right) - \kappa, \end{aligned}$$

where each cluster is a triple $c = (w_c, h_c, \beta_c)$ of two vectors and a scalar, and κ is a constant. ARD works by fixing the number of clusters to some upper bound. In the optimal clustering many of these clusters c will be empty, that is, have $\operatorname{supp}(c) = \emptyset$.

This quality function is *not* additive, for two reasons. First of all, there is the term $2|V| \log \beta_c$ for each cluster, which stems from the half-normal priors on W and H. This term depends on the number of nodes. Secondly, the κ term actually depends on the number of clusters and the number of nodes, since it contains the normalizing constants for the hyperprior on β, as well as constant factors for the half-normal priors. For a fixed graph and fixed number of clusters the κ term can be ignored, however.

As a result, Psorakis *et al.*'s method is also not local, as the following counterexample shows:

Theorem 6.9. *Q_{BayNMF} is not local.*

Proof. Consider a graph G_1, consisting of a ring of $n = 10$ cliques, where each clique has $m = 5$ nodes, and two edges connecting it to the adjacent cliques.

We follow Psorakis *et al.*, and use hyperparameters $a = 5$ and $b = 2$. This choice is not essential, similar counterexamples exist for other hyperparameter values. As might be hoped, the Q_{BayNMF}-optimal clustering C_1 of this graph then puts each clique in a separate cluster, with a small membership for the directly connected nodes in adjacent cliques.

This clustering is certainly better than the clustering C_2 with 5 clusters each consisting of two cliques, and 5 empty clusters.

However, on a larger graph with two disjoint copies of G_1, the clustering with two copies of C_2 is better than the clustering with two copies of C_1.

But by locality we would have $Q_{\text{BayNMF}}(G_1 \cup G_1', C_1 \uplus C_1') \geqslant Q_{\text{BayNMF}}(G_1 \cup G_1', C_2 \uplus C_1')$ as well as $Q_{\text{BayNMF}}(G_1 \cup G_1', C_2 \uplus C_1') \geqslant Q_{\text{BayNMF}}(G_1 \cup G_1', C_2 \uplus C_2')$, where the primed variables indicate copies with disjoint nodes. So Q_{BayNMF} is not local. □

In the above counterexample things don't change if one uses a ring of 20 cliques instead of two disjoint rings of 10 cliques. This is closer to the original characterization of the resolution limit by Fortunato and Barthélemy (2007). In a ring of 20 cliques, the solution with 10 clusters is better than the solution with 20 clusters. But it is harder to show that this violates locality.

6.5 NMF as a probabilistic model

NMF can be seen as a maximum likelihood fit of a generative probabilistic model. The quality function that is optimized is then the log likelihood of the model conditioned on the observed graph,

$$Q(C, G) = \log P(C|G).$$

One assumes that there is some underlying hidden cluster structure, and the edges in the graph depend on this structure. The clustering structure in turn depends on the nodes under consideration. So, by Bayes rule we may decompose $P(C|G)$ as

$$P(C|V, A) = P(A|C, V)P(C|V)P(V)/P(V, A).$$

The terms $P(V)$ and $P(V, A)$ are constant given the graph, so the quality function becomes

$$Q(C, G) = \log P(A|C, V) + \log P(C|V) + \kappa,$$

where $\kappa = \log P(V) - \log P(V, A)$ is a constant. The first term is the likelihood of the edges given the clustering, the second factor is the prior probability of a clustering for a certain set of nodes.

To make the above general formulation into an NMF model, one assumes that the edge weights are distributed independently, depending on the product of the membership matrices. Then a prior is imposed on the membership coefficients. Usually a conjugate prior is used, which for Gaussian likelihood has a half-normal distribution, and for Poisson likelihood has a gamma distribution. So the simplest symmetric Gaussian NMF method would be

$$\begin{aligned} a_{ij} &\sim \mathcal{N}(\hat{a}_{ij}, 1) \\ \hat{a}_{ij} &= \sum_c h_{ci} h_{cj} \\ h_{ci} &\sim \mathcal{HN}(\sigma) \end{aligned}$$

Which leads to the quality function

$$\begin{aligned} Q(C, G) = -\frac{1}{2} \sum_{i,j \in V} (a_{ij} - \hat{a}_{ij})^2 - \frac{1}{2\sigma^2} \sum_{c \in C} \sum_{i \in V} h_{ci}^2 \\ + |V|^2 \log \sqrt{2\pi} + |C||V| \log \sqrt{\pi\sigma^2/2}. \end{aligned}$$

This is a regularized variant of symmetric NMF discussed previously.

Such a model implicitly assumes a fixed number of clusters; and the corresponding quality function will not be local if the number of clusters is not fixed. Intuitively, this happens because the model has to 'pay' the normalizing constant of the prior distribution for each h_{ci}, the number of which is proportional to the number of clusters.

The method of Psorakis *et al.* also stems from a probabilistic model. They use a Poisson likelihood and a half-normal prior. Note that these are not conjugate. For finding the maximum likelihood solution conjugacy is not important. Using a conjugate prior becomes important only when doing variational Bayesian inference or Gibbs sampling (Cemgil, 2009).

To determine the number of clusters, Psorakis *et al.* put a gamma hyperprior on the inverse variance β. This allows a sharply peaked distribution on w_c and

h_c when the support of a cluster is empty. The model is

$$\begin{aligned} a_{ij} &\sim \text{Poisson}(\hat{a}_{ij}) \\ \hat{a}_{ij} &= \sum_c h_{ci} w_{cj} \\ h_{ci} &\sim \mathcal{HN}(1/\sqrt{\beta}_c) \\ w_{ci} &\sim \mathcal{HN}(1/\sqrt{\beta}_c) \\ \beta_c &\sim \text{Gamma}(a, b) \end{aligned}$$

As shown in Section 6.4.4, the corresponding quality function is not local. The problems stem from the priors on W, H and β, which depend on the number of nodes and clusters. We will next try to find a different prior that is local.

6.5.1 A local prior

To get a local quality function from a probabilistic model, that does not assume a fixed number of clusters, we clearly need a different prior. The approach we take will be to construct an additive quality function, which is local by Theorem 6.7.

First assume as above that the likelihoods of the edges are independent and depending on the product of membership degrees, that is $P(A|C,V) = \prod_{ij} P(a_{ij}|\hat{a}_{ij})$. This fits nicely into the fourth term, q_{edge}, of an additive quality function.

Without loss of generality we can split the prior into two parts. First the support of each cluster is determined, and based on this support the membership coefficients are chosen. If we define $S = \{\text{supp}(c) | c \in C\}$, then this means that

$$P(C|V) = P(C|V,S)P(S|V).$$

Just like C, S should be seen as a multiset, since multiple clusters can have the same support. A reasonable choice for the first term $P(C|V,S)$ is to assume that the clusters are independent, and that the membership coefficients inside each cluster are also independent, so

$$\begin{aligned} C &= \{C_s \mid s \in S\} \\ P(C_s|V,s) &= \prod_{c \in C} \Big(\prod_{i \in s} P(h_{ci}) \prod_{i \in V \setminus s} \delta(h_{ci}, 0) \Big), \end{aligned}$$

where δ is the Kronecker delta, which forces h_{ci} to be zero for nodes not in s. The logarithm of $P(C|V,S)$ is a sum of terms that depend only on a single cluster, so it can be encoded in the q_{clus} term of an additive quality function.

Now consider $P(S|V)$. If we know nothing about the nodes, then the two simplest aspects of S we can look at are: (1) how many clusters cover each node, and (2) how many nodes are in each cluster. The only local choice for (1) is to take the number of clusters that cover node i, $n_i = \#\{s \in S \mid i \in s\}$, be independent and identically distributed according to some $f(n_i)$. While for (2), the probability of a cluster $s \in S$ must be independent of the other clusters. And since we have no information about the nodes, the only property of s we can use is its size. This suggests a prior of the form

$$P(S|V) = \frac{1}{Z} \prod_{i \in V} f(n_i) \prod_{s \in S} g(|s|),$$

where $n_i = |\{s \in S \mid i \in s\}|$ is the number of clusters covering node i. The term $f(n_i)$ is local to each node, and can be encoded in q_{node}. The term $g(|s|)$ is local to each cluster, and can therefore be encoded in q_{clus}. The normalizing constant Z depends only on V, and so it can be encoded in q_{graph}.

If we take $f(n_i) = \mathbf{1}[n_i = 1]$ and $g(|s|) = (|s| - 1)!$, then the prior on S is exactly a Chinese Restaurant Process (Pitman, 2006). If we relax f, then we get a generalization where nodes can belong to multiple clusters. Another choice is $f(n_i) = \mathbf{1}[n_i = 1]$ and $g(|s|) = 1$. Then the prior on S is the flat prior over partitions, which is commonly used for hard clustering.

Yet another choice is to put a Poisson prior on either the number of clusters per node or the number of nodes per cluster. That is, take $f(n_i) = \lambda^{n_i}/(n_i!)e^{-\lambda}$ for some constant λ, or do the same for g. This parameter allows the user to tune the number or size of clusters that are expected a-priori.

To summarize, we obtain a local quality function of the form

$$\begin{aligned} Q(G, C) = &\sum_{i \in V} \log f(|\{c \in C \mid i \in \operatorname{supp}(c)\}|) \\ &+ \sum_{c \in C} \log g(|\operatorname{supp}(c)|) \\ &+ \sum_{c \in C} \sum_{i \in \operatorname{supp}(c)} \log P(h_{ci}) \\ &+ \sum_{i,j \in V} \log P(a_{ij} \mid \hat{a}_{ij}) \\ &+ \kappa, \end{aligned}$$

which has four independent parts: a score for a node being in a certain number of clusters, a score for the size of each cluster, a prior for each non-zero membership coefficient, and the likelihood of an edge a_{ij} given the $\hat{a}_{ij}$.

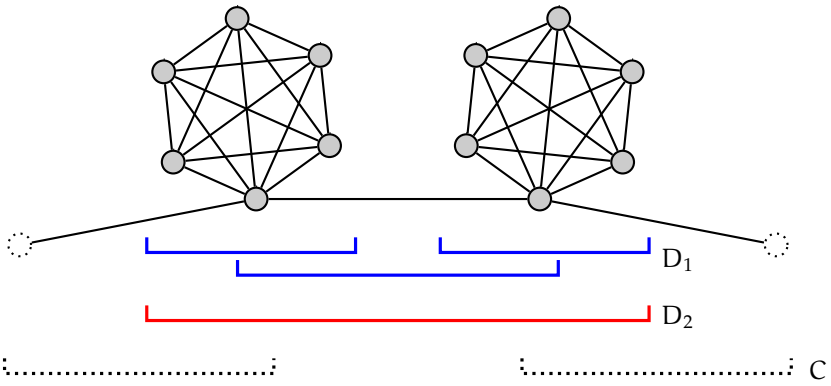


Figure 6.4: Two possible clusterings in a subgraph of a ring of cliques. In the first clustering (D_1, blue), the two cliques are in separate clusters, and there is a third cluster for the edge between them. In the second clustering (D_2, red) two cliques are put into a single cluster. A third possibility is to include the middle edge in a cluster together with one of the two cliques. A clustering of this entire subgraph will also include two clusters covering the connecting edges (C, dotted).

The discrete nature of this quality function makes it harder to optimize. It is not clear if the multiplicative gradient algorithm that is commonly employed for NMF (Lee and Seung, 2000) can be adapted to deal with a prior on the support of clusters. On the other hand, it might become possible to use discrete optimization methods, such as the successful Louvain method used for modularity maximization.

6.5.2 Analysis of the quality functions on two types of graphs

We will now investigate the local quality function proposed in the previous section.

First consider the original resolution limit model (Fortunato and Barthélemy, 2007), which consists of a ring of cliques. Two possible clusterings of a part of such a ring are illustrated in Figure 6.4.

If a quality function is local, then we know that if $D_1 \uplus C$ is a better clustering than $D_2 \uplus C$ in this subgraph, then D_1 will also be better than D_2 as part of a larger graph. In other words, if the cliques are clustered correctly in a small ring, then this is true regardless of the number of cliques in the ring (unless a clustering with very large clusters is suddenly better).

We have performed experiments with the prior from the previous section, to

see what the optimal clustering will be in practice. We use a Poisson likelihood, a half normal prior on the supported membership coefficients (with precision $\beta = 1$), a Poisson prior on the number of clusters-per-node (with $\lambda = 1$) and a flat prior on the number of nodes per cluster. To find the optimal clustering we use a general purpose optimization method, combined with a search over the possible supports of the clusters.

The left part of Figure 6.5 shows that, as expected, the optimal solution is always to have one cluster per clique when using the local quality function. For comparison we also looked at the simpler non-local NMF method without a prior on the support. In that case the optimal solution depends strongly on the prior on membership coefficients β. If β is small, then there is a penalty for every zero in the membership matrix, and hence a penalty on the number of clusters that increases with the number of nodes. If β is large enough, then the probability density $p(0) > 1$, and this penalty becomes a 'bonus'. In that case adding even an empty cluster would improve the quality, and the optimal clustering has an infinite number of clusters.

The method of Psorakis *et al.* has the same resolution limit problem, but to an even larger extent. To automatically determine the number of clusters, this method keeps the actual number of clusters fixed to a large upper bound, for which the authors take the number of nodes. This means that there are very many clusters which will be empty in the optimal solution. For these empty clusters, the parameter β_c becomes very large. And as said in the previous paragraph, this results in a bonus for empty clusters. Hence the method will tend to maximize the number of empty clusters, which results in a few large clusters actually containing the nodes. For this experiment we used the prior β_c Gamma$(5, 2)$, as is also done in the code provided by Psorakis *et al.* Note that the jaggedness in the plot is due to the fact a ring of n cliques can not always be divided evenly into m clusters of equal size. Between 24 and 50 cliques, the optimal number of clusters is always 8 or 9.

The right part of Figure 6.5 shows the influence of the parameter λ of the Poisson prior that we put on the number of clusters per node. When λ becomes smaller, it becomes *a priori* more likely for a node to be in only a single cluster, or in fact, to be in no cluster at all. It actually requires a quite strong prior to get two cliques to merge into one cluster, when using 5-cliques, we need λ to be smaller than approximately 10^{-5}.

A ring of cliques is not a realistic model of real world graphs, since on most graphs the clustering is not as clear-cut as it is there. The clustering problem can be made harder by removing edges inside the cliques, which are then no longer cliques, and better called modules; or by adding more edges between the modules.

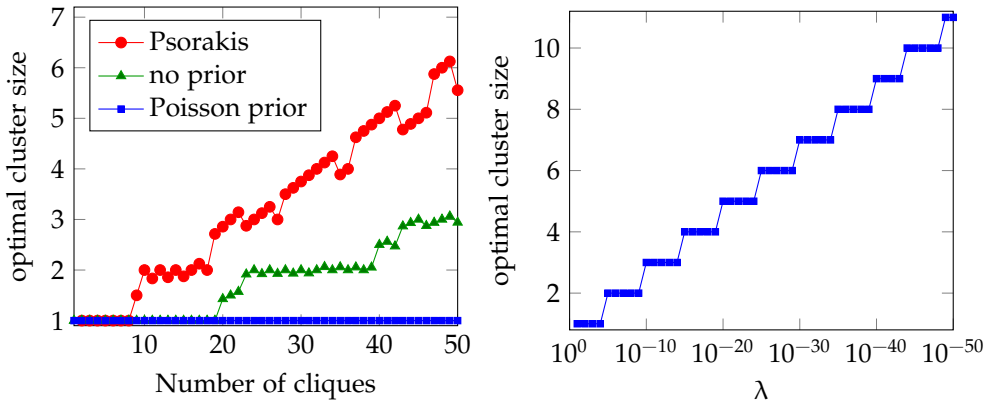


Figure 6.5: Optimal cluster size (average number of cliques per cluster) in a ring of n 5-cliques. Left: varying the number of cliques. Right: varying the λ parameter of the Poisson prior on the number of clusters per node. The number of cliques in the ring doesn't matter because of locality.

We consider such a generalization, where there are two modules connected by zero or more edges. We then generated random modules and random between module edges. The two modules are either clustered together in one big cluster, or separated. In Figure 6.6 we show simulation results of such a more realistic situation. As we can see, as the number of between module edges increases, or the number of within module edges decreases, it becomes more likely to combine the two modules into one cluster. At the threshold between the two situations, the number of between module edges is roughly equal to the number of within module edges. This matches the notion of a *strong community*, which is defined by Radicchi *et al.* (2004) as a set of nodes having more edges inside the cluster than edges leaving the cluster. A theoretical justification of these empirical results is beyond the scope of this work.

6.6 Conclusion

To our knowledge, this work is the first to investigate resolution-limit-free and local NMF quality functions for graph clustering. We gave a characterization of a class of good (i.e. local) additive quality functions for graph clustering that provides a modular interpretation of NMF for graph clustering. The definitions of locality and of additive quality functions are general, and can also be applied to other soft clustering methods. We proposed the class of local probabilistic NMF quality functions. The design and assessment of efficient algorithms for

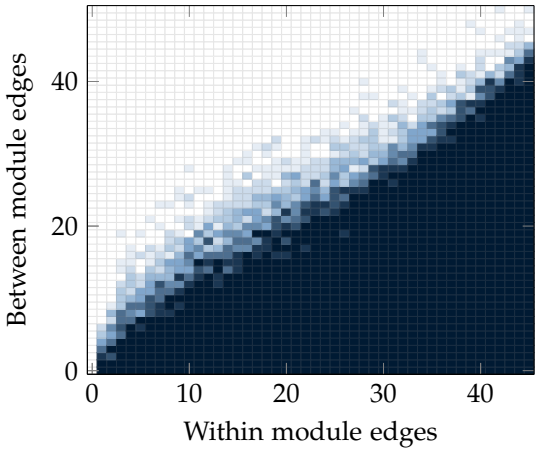


Figure 6.6: Varying the number of within and between module edges. The modules each have 10 nodes. A Poisson prior on the number of clusters per node ($\lambda = 1$) was used. We consider two possible clusterings: (a) A solution with three clusters, two clusters for the two modules and one cluster for the between module edges. And (b) the solution with a single cluster containing all nodes. The color in the plot indicates which clustering has a higher quality. In the dark region, the clustering (a) with three clusters is better. In the light region, the solution (b) with a single cluster is better. Results are the average over 10 random graphs with the given number of edges.

optimizing these quality functions remains to be investigated.

Results of this chapter provide novel insights on NMF for hard clustering, on the resolution limit of Bayesian NMF for soft clustering, and on the beneficial role of a local prior in probabilistic formulations of NMF.

6.A Proof of Theorem 6.2

Proof. Recall that in symmetric NMF, $\hat{a}$ is defined as $\hat{a}_{ij} = \sum_{c \in C} h_{ci} h_{cj}$. With orthogonality constraints, any two nodes i and j are either in the same cluster, in which case $\hat{a}_{ij} = 1$, or they are in different clusters, in which case $\hat{a}_{ij} = 0$. So $\hat{a}_{ij} = \hat{a}_{ij}^2 = \mathbf{1}[c_i = c_j]$.

Symmetric NMF is given by the optimization problem

$$\operatorname{argmax}_C \, Q_{\text{SymNMF}}(G, C) = -\frac{1}{2} \sum_{i,j \in V} (a_{ij} - \hat{a}_{ij})^2.$$

Expanding the square shows that this is equivalent to

$$\operatorname{argmax}_C \sum_{i,j \in V} \left(a_{ij} \hat{a}_{ij} - \frac{1}{2} \hat{a}_{ij}^2 \right).$$

With orthogonality constraints this is equivalent to

$$\operatorname{argmax}_C \sum_{i,j \in V} \left(a_{ij} - \frac{1}{2} \right) \mathbf{1}[c_i = c_j],$$

which is the CPM objective with $\gamma = 1/2$. □

6.B Proof of Theorem 6.4

Proof. Let Q be a local hard cluster quality function, and C be a Q-optimal clustering of a graph $G_1 = (V_1, A_1)$. Consider the subgraph G_2 induced by $D \subset C$.

Let $C_1 = C \setminus D$ and $C_2 = \emptyset$, and let $G_S = G_2$. Because C is a partition of V_1, we have that $\operatorname{supp}(C_1)$ is disjoint from G_S, and so C_1 and C_2 agree on G_S.

Then for each clustering D' of G_2 we have $Q(G_1, C_1 \uplus D) \geqslant Q(G_1, C_1 \uplus D')$ because $C_1 \cup D = C$ is an optimal clustering of G_1. By locality it follows that $Q(G_2, C_2 \uplus D) \geqslant Q(G_2, C_2 \uplus D')$.

So D is a Q-optimal clustering of G_2. □

6.C Proof of Theorem 6.5

Proof. Consider the following hard clustering quality function

$$Q(G, C) = \max_{c \in C} |c| + \min_{c \in C} |c|.$$

For each graph $G = (V, A)$, the clustering $C = \{V\}$ is the single Q-optimal clustering, with quality $2|V|$. Since there are no strict subsets of C the quality function Q is trivially resolution-limit-free.

Now consider the graphs G_1 with nodes $\{1, 2, \ldots, 7\}$ and G_2 with nodes $\{1, 2, \ldots, 6\}$, both with no edges. These graphs have a common subgraph G_S with nodes $\{1, 2, \ldots, 6\}$. Take the clusterings $D = \{\{1, 2, 3, 4\}, \{5\}, \{6\}\}$, $D' = \{\{1, 2, 3\}, \{4, 5, 6\}\}$, $C_1 = \{\{7\}\}$ and $C_2 = \{\}$. Then $Q(G_1, C_1 \uplus D) = 5 > 4 = Q(G_1, C_1 \uplus D')$, while $Q(G_2, C_2 \uplus D) = 5 < 6 = Q(G_2, C_2 \uplus D')$.

So Q is not local.

This counterexample is illustrated in Figure 6.7. □

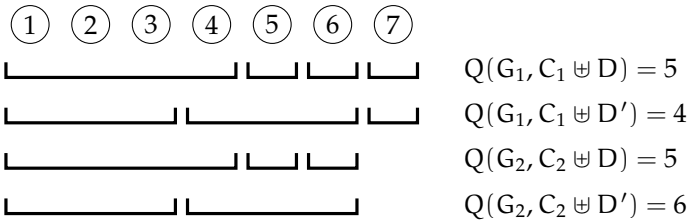


Figure 6.7: The counterexample from the proof of Theorem 6.5.

6.D Proof of Theorem 6.7, additive quality functions are local

Proof. Let Q be an additive quality function. Let G_1 G_2 and $G_S = (V, A)$ be graphs such that G_S is a subgraph of both G_1 and G_2.

Let C_1 be a clustering of G_1, C_2 a clustering of G_2 and, D, D' clusterings of G_S such that C_1 and C_2 agree on G_S.

Let $E = C_1 \cap C_2$. Then for every node $i \in \operatorname{supp}(C_1 \setminus C_2)$, we have $i \notin V$, which implies that $i \notin \operatorname{supp}(D)$ and $i \notin \operatorname{supp}(D')$. So

$$\{c \in C_1 \uplus D \mid i \in \operatorname{supp}(c)\} = \{c \in C_1 \uplus D' \mid i \in \operatorname{supp}(c)\} = \{c \in C_1 \mid i \in \operatorname{supp}(c)\}.$$

Conversely, for every node $i \notin \operatorname{supp}(C_1 \setminus C_2)$, we have

$$\{c \in C_1 \uplus D \mid i \in \operatorname{supp}(c)\} = \{c \in E \uplus D \mid i \in \operatorname{supp}(c)\}.$$

Therefore

$$\begin{aligned}
&Q(G_1, C_1 \uplus D) - Q(G_1, C_1 \uplus D') \\
&\quad = \sum_{c \in D} q_{\text{clus}}(c) - \sum_{c \in D'} q_{\text{clus}}(c) \\
&\quad + \sum_{i \in V} q_{\text{node}}\big(\{c \in E \uplus D \mid i \in \operatorname{supp}(c)\}\big) \\
&\quad - \sum_{i \in V} q_{\text{node}}\big(\{c \in E \uplus D' \mid i \in \operatorname{supp}(c)\}\big) \\
&\quad + \sum_{i,j \in V} q_{\text{edge}}\big(a_{ij}, \{c \in E \uplus D \mid i, j \in \operatorname{supp}(c)\}\big) \\
&\quad - \sum_{i,j \in V} q_{\text{edge}}\big(a_{ij}, \{c \in E \uplus D' \mid i, j \in \operatorname{supp}(c)\}\big),
\end{aligned}$$

and similarly for G_2 and C_2 in place of the G_1 and C_1.

Which implies that

$$Q(G_1, C_1 \uplus D) - Q(G_1, C_1 \uplus D') = Q(G_2, C_2 \uplus D) - Q(G_2, C_2 \uplus D').$$

And so

$$Q(G_1, C_1 \uplus D) \geqslant Q(G_1, C_1 \uplus D')$$

if and only if

$$Q(G_2, C_2 \uplus D) \geqslant Q(G_2, C_2 \uplus D').$$

In other words, Q is local. □

Part 2

Link prediction in biological bipartite graphs

Chapter 7

The link prediction problem

In the second part of this thesis we turn our attention to another machine learning problem on networks. In particular, we will look at link prediction in bipartite networks.

Bipartite networks are a class of networks where the nodes fall into two groups, and edges only appear between nodes of different groups. While methods for general networks can be used in this setting, more specialized methods will have an advantage.

In addition, other data is often available in practice. For instance one type of nodes might be proteins, for which the protein sequence is known. By using this side information, better predictions are possible. A general method can often not take advantage of this side information, but a specialized method can.

Previously, in Chapter 5, we looked at a protein–protein interaction network. Several other networks exist that describe interactions between two types of entities. For instance between proteins and drug compounds (Yamanishi, Araki, *et al.*, 2008), between proteins and ligands (Jacob and J.-P. Vert, 2008), between enzymes and metabolites (Faulon *et al.*, 2008), and so on. Of these, we will focus on the interaction between drug compounds and target proteins. We call this the *drug–target interaction network*.

The *in silico* prediction of interaction between drugs and target proteins is a core step in the drug discovery process for identifying new drugs or novel targets for existing drugs, in order to guide and speed up the laborious and costly experimental determination of drug–target interaction (Haggarty *et al.*, 2003).

Drug–target interaction data are available for many classes of pharmaceutically useful target proteins including enzymes, ion channels, GPCRs and nuclear receptors (Hopkins and Groom, 2002). Several publicly available databases have

been built and maintained, such as KEGG BRITE (Kanehisa *et al.*, 2006), DrugBank (Wishart *et al.*, 2008), GLIDA (Okuno *et al.*, 2008), SuperTarget and Matador (Günther *et al.*, 2008), and BRENDA (Schomburg *et al.*, 2004), and ChEMBL (Overington, 2009), containing drug–target interaction and other related sources of information, like chemical and genomic data.

A property of the current drug–target interaction databases is that they contain a rather small number of drug–target pairs which are experimentally validated interactions. This motivates the need for developing methods that predict true interacting pairs with high accuracy.

Recently, machine learning methods have been introduced to tackle this problem. They can be viewed as instances of the more general link prediction problem, see Lü and Zhou (2011) for a recent survey of this topic. These methods are motivated by the observation that similar drugs tend to target similar proteins (Schuffenhauer *et al.*, 2003; Klabunde, 2007). This property was shown for instance for chemical (Martin, Kofron, and Traphagen, 2002) and side effect similarity (Campillos *et al.*, 2008), and motivated the development of an integrated approach for drug–target interaction prediction (Jaroch and Weinmann, 2006). A desirable property of this approach is that it does not require the 3D structure information of the target proteins, which is needed in traditional methods based on docking simulations (Cheng *et al.*, 2007).

The current state-of-the-art for the *in silico* prediction of drug–target interaction is formed by methods that employ similarity measures for drugs and for targets in the form by kernel functions, like Bleakley and Yamanishi (2009), Jacob, Hoffmann, *et al.* (2008), Wassermann, Geppert, and Bajorath (2009), Yamanishi, Araki, *et al.* (2008), and Yamanishi, Kotera, *et al.* (2010). Kernels provide a way to abstract from the details of the side information. A method for link prediction does not have to know what information the kernel represents, just that it gives similarities between nodes of the same class. By using kernels, multiple sources of information can also be easily combined for performing prediction (Schölkopf, Tsuda, and J. P. Vert, 2004).

7.1 Drug–target interaction data

We used four drug–target interaction networks in humans involving enzymes, ion channels, G-protein-coupled receptors (GPCRs) and nuclear receptors; first analyzed by Yamanishi, Araki, *et al.* (2008). We worked with the datasets provided by these authors, in order to facilitate benchmark comparisons with the current state-of-the-art algorithms that do the same. We use these datasets as

Table 7.1: The number of drugs and target proteins, their ratio, and the number of interactions in the drug–target datasets from Yamanishi, Araki, *et al.* (2008).

Dataset	Drugs	Targets	n_D/n_T	Interactions
Enzyme	445	664	0.67	2926
Ion Channel	210	204	1.03	1476
GPCR	223	95	2.35	635
Nuclear Receptor	54	26	2.08	90

they are without adding new interactions from source databases.[1] Table 7.1 lists some properties of the datasets.

Aside from an interaction network, the datasets each also include two kernels. One for the drug compounds and one for the target proteins.

Amino acid sequences of the target (human) proteins were obtained from the KEGG GENES database (Kanehisa *et al.*, 2006). Sequence similarity between proteins was computed using a normalized version of Smith–Waterman score (Smith and Waterman, 1981). This score is computed by sequence alignment; sequences that are more similar when aligned get a higher score. We denote the resulting kernel by $K_{\text{genomic,d}}$, for *genomic* similarity on *d*rugs.

Drug–target interaction information was retrieved from the KEGG BRITE (Kanehisa *et al.*, 2006), BRENDA (Schomburg *et al.*, 2004), SuperTarget (Günther *et al.*, 2008) and DrugBank (Wishart *et al.*, 2008) databases. Chemical structures of the compounds was derived from the DRUG and COMPOUND sections in the KEGG LIGAND database (Kanehisa *et al.*, 2006). The chemical structure similarity between compounds was computed using SIMCOMP (Hattori *et al.*, 2003). SIMCOMP tries to compute the similarity of two small molecules by graph matching. Since graph matching is an NP hard problem, SIMCOMP uses an approximation based on a randomized search, which leads to the problem that the resulting matrix is neither symmetric nor positive definite. So it is not a kernel[2]. If we call the matrix produced by SIMCOMP S, then we can make it symmetric with $S_{sym} = (S + S^T)/2$, and add a small multiple of the identity matrix to enforce the positive definite property. This resulted in a kernel $K_{\text{chemical,t}}$, for the *chemical* similarity of *t*argets.

[1] These datasets are publicly available at `http://web.kuicr.kyoto-u.ac.jp/supp/yoshi/drugtarget/`.

[2] This similarity matrix is the one available online. The preprocessing to turn it into a kernel is our own.

7.2 Evaluation

One can distinguish between prediction for 'known' drug compounds or targets, for which at least one interaction is present in the training set; and prediction for 'unseen' or 'new' drug compounds or targets, for which no interaction is available in the training set. This results in four possible settings for predicting drug-target interaction, depending on whether the drug compounds and/or targets are known or unseen (Yamanishi, Araki, *et al.*, 2008). That is,

1. To predict whether an unseen drug compound will interact with an unseen target protein. We call this the 'unseen' setting.
2. To train a model to predict with which known drugs a previously unseen target protein will interact. We call this the 'unseen target' setting.
3. To train a model to predict with which known targets a previously unseen drug will interact. We call this the 'unseen drug' setting.
4. To find new putative interactions between drugs and targets already in the dataset. We call this the 'pairs' setting.

Formally we are given a set $D = \{d_1, d_2, \ldots, d_{n_d}\}$ of drug compounds and a set $T = \{t_1, t_2, \ldots, t_{n_t}\}$ of target proteins. There is also a set of known interactions between drugs and targets. If we consider these interactions as edges, then they form a bipartite network. We can characterize this network by the $n_d \times n_t$ adjacency matrix Y. That is, $y_{ij} = 1$ if drug d_i interacts with target t_j, and $y_{ij} = 0$ otherwise. A method can then use the matrix Y of known interactions, and the kernels defined previously.

Instead of looking at binary classification, it is more useful to look at a ranking of all possible interactions. So the task is to rank all drug–target pairs (d_i, t_j) such that highest ranked pairs are the most likely to interact.

7.3 Outline of this part of the thesis

In **Chapter 8** we show that a simple machine learning method that uses the drug–target network as the only source of information is capable of predicting true interaction pairs with high accuracy. Specifically, we introduce interaction profiles of drugs (and of targets) in a network, which are binary vectors specifying the presence or absence of interaction with every target (drug) in that network. We define a kernel on these profiles, called the Gaussian Interaction Profile (GIP) kernel, and use a simple classifier, (kernel) Regularized Least Squares (RLS), for prediction drug–target interactions. We test the effectiveness of RLS

with the GIP kernel on the four drug–target interaction networks discussed in this chapter. The proposed algorithm achieves area under the precision-recall curve (AUPR) up to 92.7, significantly improving over results of state-of-the-art methods. Moreover, we show that also using the kernels based on chemical and genomic information further increases accuracy, with a neat improvement on small datasets. These results substantiate the relevance of the network topology (in the form of interaction profiles) as source of information for predicting drug–target interactions.

Chapter 9 extends this method to predict interactions with 'unseen' drug compounds or target proteins. That is, drug compounds (or target proteins) for which there are no experimentally validated interactions in the dataset. We show that a simple weighted nearest neighbor procedure is highly effective for this task. We integrate this procedure into a recent machine learning method for drug–target interaction we developed in previous work. Results of experiments indicate that the resulting method predicts true interactions with high accuracy also for new drug compounds and achieves results comparable or better than those of recent state-of-the-art algorithms.

Chapter 10 focuses on the crucial issue of data bias. We show that the four datasets used in this part contain a bias because of the way they have been constructed: all drug compounds and target proteins have at least one interaction and some of them have only a single interaction. We show that this bias can be exploited by prediction methods to achieve an optimistic generalization performance as estimated by cross-validation procedures, in particular leave-one-out cross validation. We discuss possible ways to mitigate the effect of this bias, by changing the validation procedure or the datasets. In general, results indicate that the data bias should be taken into account when assessing the generalization performance of machine learning methods for the *in silico* prediction of drug–target interactions.

Chapter 8

Drug–target interaction prediction with Gaussian Interaction Profiles

The in silico *prediction of potential interactions between drugs and target proteins is of core importance for the identification of new drugs or novel targets for existing drugs. However, only a tiny portion of all drug–target pairs in current datasets are experimentally validated interactions. This motivates the need for developing computational methods that predict true interaction pairs with high accuracy.*

We show that a simple machine learning method that uses the drug–target network as the only source of information is capable of predicting true interaction pairs with high accuracy. Specifically, we introduce interaction profiles of drugs (and of targets) in a network, which are binary vectors specifying the presence or absence of interaction with every target (drug) in that network. We define a kernel on these profiles, called the Gaussian Interaction Profile (GIP) kernel, and use a simple classifier, (kernel) Regularized Least Squares (RLS), for prediction drug–target interactions. We test comparatively the effectiveness of RLS with the GIP kernel on four drug–target interaction networks used in previous studies. The proposed algorithm achieves area under the precision-recall curve (AUPR) up to 92.7*, significantly improving over results of state-of-the-art methods. Moreover, we show that using also kernels based on chemical and genomic information further increases accuracy, with a neat improvement on small datasets. These results substantiate the relevance of the network topology (in the form of interaction profiles) as source of information for predicting drug–target interactions.*

This chapter is based on van Laarhoven, Nabuurs, and Marchiori (2011), "Gaussian interaction profile kernels for predicting drug–target interaction", published in Bioinformatics. The software, datasets and supplementary results for this chapter are available on `http://cs.ru.nl/~tvanlaarhoven/drugtarget2011/`.

8.1 Introduction

In the previous chapter we have introduced the drug–target interaction prediction problem. Several settings were discussed. In this chapter we focus on the 'pairs' setting, where both the drugs and targets are known. That is, we use known interactions for predicting novel ones.

We want to analyze the relevance of the topology of drug–target interaction networks as source of information for predicting interactions. We do this by introducing a kernel that captures the topological information. Using a simple machine learning method we then compare this kernel to kernels based on other sources of information.

Specifically, we start from the assumption that two drugs that interact in a similar way with the targets in a known drug–target interaction network, will also interact in a similar way with new targets. We formalize this property by describing each drug with an interaction profile, a binary vector describing the presence or absence of interaction with every target in that network. The interaction profile of a target is defined in a similar way. From these profiles we construct the Gaussian Interaction Profile kernel.

We show that interaction profiling can be effectively used for accurate prediction of drug–target interaction. Specifically, we propose a simple regularized least square algorithm incorporating a product of kernels constructed from drug and target interaction profiles. We test the predictive performance of this method on four drug–target interaction networks in humans involving enzymes, ion channels, GPCRs and nuclear receptors. These experiments show that using *only* information on the topology of the drug–target interaction, in the form of interaction profiles, excellent results are achieved as measured by the area under the precision-recall curve (AUPR) (Davis and Goadrich, 2006). In particular, on three of the four considered datasets the performance is superior to the best results of current state-of-the-art methods which use multiple sources of information.

We further show that the proposed method can be easily extended to also use other sources of information in the form of suitable kernels. Results of experiments where also chemical and genomic information on drugs and targets is included show excellent performance, with AUPR score of 91.5, 94.3, 79.0 and 68.4 on the four datasets, achieving an improvement of 7.4, 13.0, 12.3 and 7.2 over the best results reported in Bleakley and Yamanishi (2009). A thorough analysis of the results enable us to detect several new putative drug–target interactions, see http://cs.ru.nl/~tvanlaarhoven/drugtarget2011/new-interactions/.

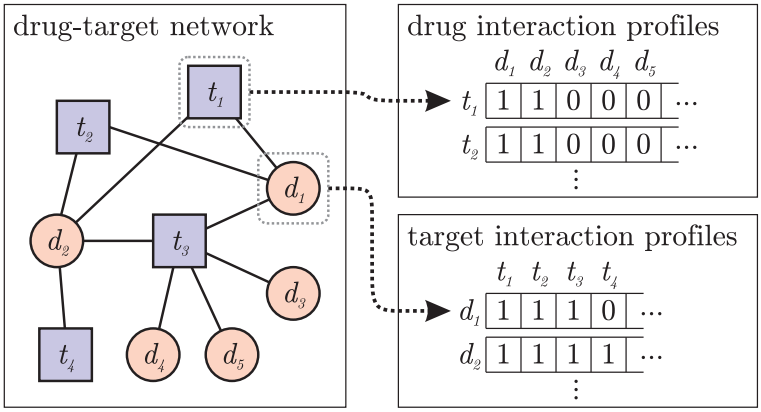


Figure 8.1: An illustration of the construction of interaction profiles from a drug–target interaction network. Circles are drugs, squares are targets. In this example the interaction profile of target t_1 indicates that it interacts with drugs d_1 and d_2, but not with d_3, d_4 or d_5.

8.2 Gaussian Interaction Profile Kernel

Our method is based on the assumption that drugs exhibiting a similar pattern of interaction and non-interaction with the targets of a drug–target interaction network are likely to show similar interaction behavior with respect to new targets. We use a similar assumption on targets. We therefore introduce the (target) *interaction profile* y_{d_i} of a drug d_i to be the binary vector encoding the presence or absence of interaction with every target in the considered drug–target network. This is nothing more than row i of the adjacency matrix Y. Similarly, the (drug) interaction profile $y_{t_j}^T$ of a target protein t_j is a vector specifying the presence or absence of interaction with every drug in the considered drug–target network. The interaction profiles generated from a drug–target interaction network can be used as feature vectors for a classifier. Figure 8.1 illustrates the construction of interaction profiles.

Following the current state-of-the-art for the drug–target interaction prediction problem, we will use kernel methods, and hence construct a kernel from the interaction profiles. This kernel does not include any information beyond the topology of the drug–target network.

One of the most popular choices for constructing a kernel from a feature vector is the Gaussian kernel, also known as the Radial Basis Function (RBF) kernel. This kernel is, for drugs d_i and d_j,

$$K_{GIP,d}(d_i, d_j) = \exp(-\beta_d \|y_{d_i} - y_{d_j}\|^2).$$

A kernel for the similarities between target proteins, $K_{GIP,t}$, can be defined analogously. We call these kernels Gaussian Interaction Profile (GIP) kernels.

The parameter β_d controls the kernel bandwidth. We set

$$\beta_d = \tilde{\beta}_d \Big/ \Big(\frac{1}{n_d} \sum_{i=1}^{n_d} |y_{d_i}|^2 \Big).$$

That is, we normalize the parameter by dividing it by the average number of interactions per drug. With this choice the kernel values become independent of the size of the dataset. In principle the new bandwidth parameters $\tilde{\beta}_d$ and $\tilde{\beta}_t$ could be set with cross-validation, but in this chapter we simply use $\tilde{\beta}_d = \tilde{\beta}_t = 1$.

There are other ways to construct a kernel from interaction profiles. For example, Basilico and Hofmann (2004) propose using the correlation of interaction profiles. We have performed brief experiments with these other kernels, which show that Gaussian Interaction Profile kernels consistently outperform kernels based on correlation or inner products. The detailed results of these experiments are included in Supplementary Table S1.

8.3 Integrating Chemical and Genomic Information

To combine the interaction profile kernel with the chemical and genomic kernels, we use a simple weighted average,

$$K_d = \alpha_d K_{chemical,d} + (1 - \alpha_d) K_{GIP,d}$$
$$K_t = \alpha_t K_{genomic,t} + (1 - \alpha_t) K_{GIP,t}.$$

For the reported results of our evaluation we use simply the unweighted average, for both drugs and targets, i.e. $\alpha_d = \alpha_t = 0.5$. In Section 8.7.2 we further analyze the effect of these parameters on the predictive performance of the method.

8.4 The RLS-avg classifier

In principle we could use the Gaussian Interaction Profile kernels with any kernel based classification or ranking algorithm. We choose to use a very basic classifier, the (kernel) Regularized Least Squares (RLS) classifier. While Least Squares is primarily used for regression, when a good kernel is used it has classification accuracy similar to that of Support Vector Machines (Rifkin and Klautau, 2004). Our own experiments confirm this finding. In the RLS classifier, the predicted values $\hat{y}$ with a given kernel K have a simple closed form solution,

$$\hat{y} = K(K + \sigma I)^{-1} y,$$

where σ is a regularization parameter. Higher values of σ give a smoother result, while for $\sigma = 0$ we get $\hat{y} = y$, and hence no generalization at all. The value $\hat{y}$ is a real valued score, which we can interpret as a confidence.

The RLS classifier is sensitive to the encoding used for y. Here we use 1 for encoding interacting pairs and 0 for non-interacting ones. Brief experiments have shown that the classifier is not sensitive to this choice, as long as the value used for non-interactions is close to 0. Using a value very different from 0, like -1, would place too much weight on non-interactions. The classifier would then try to avoid predicting pairs that look like non-interactions, rather than predicting pairs that look like interactions.

In the previous sections we defined kernels on drugs and kernels on target proteins. There are several ways in which we can use kernels in both of these dimensions. Following other works, like Bleakley and Yamanishi (2009) and Xia *et al.* (2010), a simple and effective approach is to apply the classifier for each drug independently using, only the target kernel; and also for each target independently using only the drug kernel. Then the final score for a drug–target pair is a combination of the two outputs.

Here we use the average of the output values, and denote the resulting method by RLS-avg. Observe that in the formulation of the RLS classifier we use, performing independent prediction amounts to replacing the vector y with the matrix Y, hence the prediction of RLS-avg is

$$\hat{Y} = \frac{1}{2}\big(K_d(K_d + \sigma I)^{-1}Y\big) + \frac{1}{2}\big(K_t(K_t + \sigma I)^{-1}Y^T\big)^T.$$

Note this model is slightly different from using the Kronecker sum kernel (Kashima, Oyama, *et al.*, 2009). Because regularization is performed for drugs and targets separately in the RLS-avg method, rather than jointly.

8.5 RLS-Kron classifier

A better alternative is to combine the kernels into a larger kernel that directly relates drug–target pairs. This is done with the Kronecker product kernel (Basilico and Hofmann, 2004; Ben-Hur and Noble, 2005; Oyama and Manning, 2004; Hue and J.-P. Vert, 2010). The Kronecker product $K_d \otimes K_t$ of the drug and target kernels is

$$K((d_i, t_j), (d_k, t_l)) = K_d(d_i, d_k)K_t(t_j, t_l).$$

With this kernel we can make predictions for all pairs at once,

$$\mathrm{vec}(\hat{Y}^T) = K(K + \sigma I)^{-1}\,\mathrm{vec}(Y^T),$$

where $\mathrm{vec}(Y^T)$ is the a vector of all interaction pairs, created by stacking the columns of Y^T. We call this method RLS-Kron.

Using the Kronecker product kernel directly would involve calculating the inverse of an $n_d n_t \times n_d n_t$ matrix, which would take $\mathcal{O}((n_d n_t)^3)$ operations, and would also require too much memory. We use a more efficient implementation based on eigendecompositions, previously presented in Raymond and Kashima (2010).

Let $K_d = V_d \Lambda_d V_d^T$ and $K_t = V_t \Lambda_t V_t^T$ be the eigendecompositions of the two kernel matrices. Since the eigenvalues(vectors) of a Kronecker product are the Kronecker product of eigenvalues(vectors), for our Kronecker product kernel we have simply $K = K_d \otimes K_t = V \Lambda V^T$, where $\Lambda = \Lambda_d \otimes \Lambda_t$ and $V = V_d \otimes V_t$. The matrix that we want to invert, $K + \sigma I$ has these same eigenvectors V, and eigenvalues $\Lambda + \sigma I$. Hence

$$K(K + \sigma I)^{-1} = V \Lambda (\Lambda + \sigma I)^{-1} V^T.$$

To efficiently multiply this matrix with $\mathrm{vec}(Y^T)$ we can use a further property of the Kronecker product, namely that $(A \otimes B)\,\mathrm{vec}(X) = \mathrm{vec}(BXA^T)$. Combining these facts we get that the RLS prediction is

$$\hat{Y} = V_d Z^T V_t^T,$$

where

$$\mathrm{vec}(Z) = (\Lambda_d \otimes \Lambda_t)(\Lambda_d \otimes \Lambda_t + \sigma I)^{-1}\,\mathrm{vec}(V_t^T Y^T V_d).$$

So, to make a RLS prediction using the Kronecker product kernel we only need to perform the two eigendecompositions and some matrix multiplications, bringing the runtime down to $\mathcal{O}(n_d^3 + n_t^3)$. The efficiency of this computation could be further improved yielding a quadratic computational complexity by applying recent techniques for large scale kernel methods for computing the two kernel decompositions (Kashima, Idé, *et al.*, 2009; Wu *et al.*, 2006).

8.6 Comparison methods

In order to assess globally the performance of our method, we compare it against current state-of-the-art algorithms. To the best of our knowledge, the best results on these datasets obtained so far are those reported by Bleakley and Yamanishi (2009), where the Bipartite Local Models (BLM) approach was introduced. These results were achieved by combining the output scores of the Kernel Regression Method (KRM) (Yamanishi, Araki, *et al.*, 2008) and BLM by taking their maximum value. We briefly recall these methods here.

In the KRM method, drugs and targets are embedded into a unified space called the 'pharmacological space'. A regression model is learned between the chemical structure (respectively, genomic sequence) similarity space and this pharmacological space. Then new potential drugs and targets are mapped into the pharmacological space using this regression model. Finally, new drug–target interactions are predicted by connecting drugs and target proteins that are closer than a threshold in the pharmacological space.

The BLM method is similar to our RLS-avg method. In the BLM method, the presence or absence of a drug–target interaction is predicted as follows. First, the target is excluded, and a training set is constructed consisting of two classes: all other known targets of the drug in question, and the targets not known to interact with that drug. Second, a Support Vector Machine that discriminates between the two classes is constructed, using the available genomic kernel for the targets. This model is then used to predict the label of the target, and hence the interaction or non-interaction of the considered drug–target pair. A similar procedure is applied with the roles of drugs and targets reversed, using the chemical structure kernel instead. These two results are combined by taking the maximum value.

8.7 Evaluation

In order to compare the performance of the methods, we performed systematic experiments simulating the process of bipartite network inference from biological data on four drug–target interaction networks. These experiments are done by full leave-one-out cross-validation (LOOCV) as follows. In each run of the method, one drug–target pair (interacting or non-interacting) is left out by setting its entry in the Y matrix to 0. Then we try to recover its true label using the remaining data.

Note that when leaving out a drug–target pair the Y matrix changes, and therefore the GIP kernel has to be recomputed.

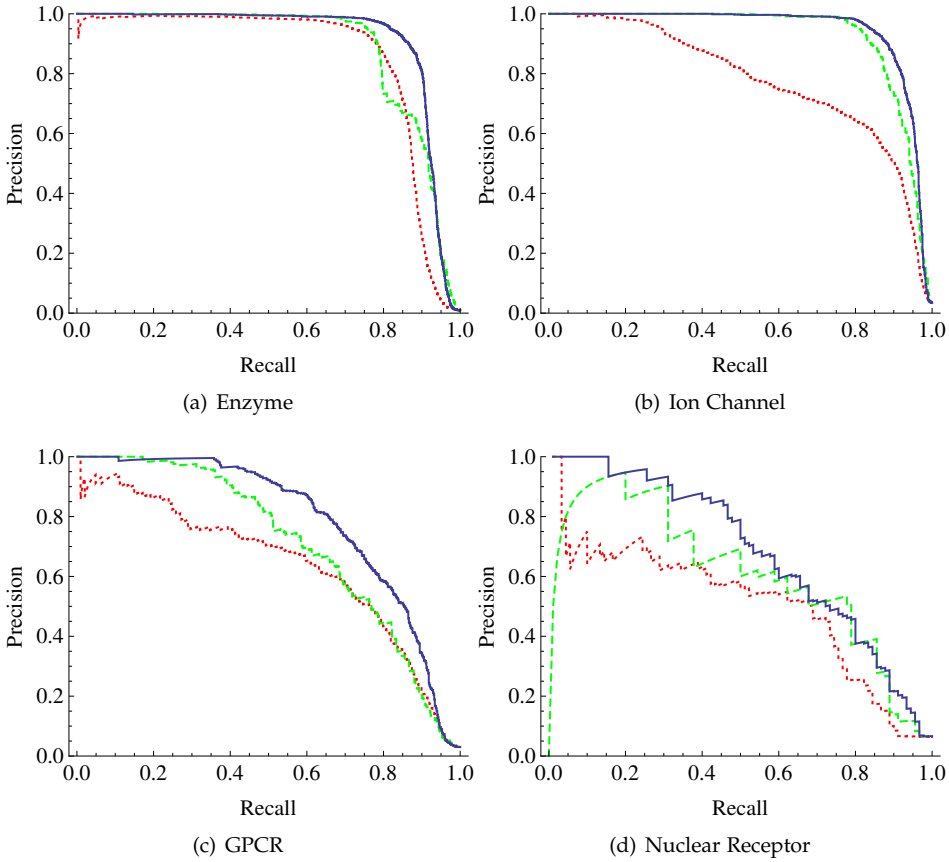
We also performed a variation of these experiments using 5 trials of 10-fold cross-validation. We recomputed the GIP kernels for each fold, also for 10-fold cross-validation. So no information about the removed interactions was leaked in this way.

The results can be found in Supplementary Table S2; we observed no large differences compared to the results obtained using LOOCV.

In all experiments we have chosen the values for the parameters in an uninformative way. In particular, we set the regularization parameter $\sigma = 1$ for both RLS methods; and as stated before, we set the kernel bandwidths $\tilde{\beta}_d = \tilde{\beta}_t = 1$ for both the drug and target interaction profile kernels.

Table 8.1: Results on the drug target interaction datasets. The AUC and AUPR scores are normalized to 100. For each dataset, the the highest AUC/AUPR score is marked in boldface.

Dataset	Method	Kernel	AUC	AUPR
Enzyme	BY09 (auc)	chem/gen	97.6	83.3
	BY09 (aupr)	chem/gen	97.3	84.1
	RLS-avg	GIP	98.2	88.1
	RLS-avg	chem/gen	96.6	84.5
	RLS-avg	avg.	97.9	90.5
	RLS-Kron	GIP	**98.3**	88.5
	RLS-Kron	chem/gen	96.6	85.6
	RLS-Kron	avg.	97.8	**91.5**
Ion Channel	BY09 (auc)	chem/gen	97.3	78.1
	BY09 (aupr)	chem/gen	93.5	81.3
	RLS-avg	GIP	98.5	91.8
	RLS-avg	chem/gen	97.1	80.7
	RLS-avg	avg.	98.1	93.2
	RLS-Kron	GIP	**98.6**	92.7
	RLS-Kron	chem/gen	97.1	77.5
	RLS-Kron	avg.	98.4	**94.3**
GPCR	BY09	chem/gen	**95.5**	66.7
	RLS-avg	GIP	94.5	70.0
	RLS-avg	chem/gen	94.7	66.0
	RLS-avg	avg.	95.0	77.1
	RLS-Kron	GIP	94.7	71.3
	RLS-Kron	chem/gen	94.8	63.8
	RLS-Kron	avg.	95.4	**79.0**
Nuclear Receptor	BY09	chem/gen	88.1	61.2
	RLS-avg	GIP	88.7	60.4
	RLS-avg	chem/gen	86.4	54.7
	RLS-avg	avg.	**92.5**	67.0
	RLS-Kron	GIP	90.6	61.0
	RLS-Kron	chem/gen	85.9	51.1
	RLS-Kron	avg.	92.2	**68.4**



(a) Enzyme (b) Ion Channel

(c) GPCR (d) Nuclear Receptor

Figure 8.2: Precision–recall curves for the RLS-Kron method. The red dotted line corresponds to using only the chemical and genomic kernels. The green dashed line corresponds to using only the GIP kernels. The blue solid line corresponds to the average of the two types of kernels. On all datasets the average kernel shows a small improvement over either other kernel type alone.

We assessed the performance of the methods with the following two quality measures generally used in this type of studies: AUC and AUPR. Specifically, we computed the ROC curve of true positives as a function of false positives, and considered the area under the ROC curve (AUC) as quality measure (see for instance Fawcett, 2006). Furthermore, we considered the the precision–recall curve (Raghavan, Jung, and Bollmann, 1989), that is, the plot of the ratio of true positives among all positive predictions for each given recall rate. The area under this curve (AUPR) provides a quantitative assessment of how well,

on average, predicted scores of true interactions are separated from predicted scores of true non-interactions. For this task, because there are few true drug–target interactions, the AUPR is a more significant quality measure than the AUC, as it punishes much more the existence of false positive examples found among the best ranked prediction scores (Davis and Goadrich, 2006).

Table 8.1 contains the results for the two RLS-based classifiers, RLS-avg and RLS-Kron, each with three different kernel combinations:

- GIP: Using only the Gaussian Interaction Profile kernels, i.e. $K_d = K_{GIP,d}$ and $K_t = K_{GIP,t}$, corresponding to $\alpha_d = \alpha_t = 1$.
- chem/gen: Using only the chemical structure and genomic sequence similarity, so $K_d = K_{chemical,d}$ and $K_t = K_{genomic,t}$, corresponding to $\alpha_d = \alpha_t = 0$.
- avg: Using the average of the two types of kernels, corresponding to $\alpha_d = \alpha_t = 0.5$.

For comparison, we have also included in the table as BY09 (AUC) and BY09 (AUPR) the best results from the combined BML and KRM methods from Bleakley and Yamanishi (2009). For the GPCR and Nuclear Receptor datasets, the method with the highest AUC is the same as the one with the highest AUPR, therefore it is included only once, as BY09.

8.7.1 Analysis

Using only the GIP kernel, our Kronecker product RLS method has AUPR scores of 88.5, 92.7, 71.3 and 61.0 on the Enzyme, Ion Channel, GPCR and Nuclear Receptor datasets respectively. These results are superior to the results from using only the chemical and genomic kernels.

Overall the RLS-Kron and RLS-avg methods have comparable AUC scores. However, the RLS-Kron has a better AUPR when using the GIP kernel, and a worse AUPR when using the chemical and genomic kernels. We believe that this problem is due to the poor quality of the chemical similarity kernel, to which the RLS-Kron method is more sensitive.

Note also that the RLS-avg method is comparable to Bleakley and Yamanishi's bipartite local model (BLM) approach. The differences are that whereas we use a RLS classifier, they use Support Vector Machines; and whereas we use the average to combine results, they use the maximum value. It is therefore not surprising that when using the chemical and genomic kernels the results of the RLS-avg method are very similar to their results.

In all cases the best results are obtained when the GIP kernels are combined with the chemical and genomic kernels. With the RLS-Kron method we then

obtain AUPR scores of 91.5, 94.3, 79.0 and 68.4 on the four datasets, which is an improvement of 7.4, 13.0, 12.3 and 7.2 over the best results reported by Bleakley and Yamanishi (2009). Figure 8.2 shows the precision–recall curves for the RLS-Kron method. Compared to other methods, the RLS-Kron method with the average kernels achieves a good precision also at higher recall values, especially on the larger datasets (Enzyme and Ion Channel).

8.7.2 Kernels' relevance

In the previous section we have shown that using a mix of the GIP kernels and the chemical and genomic kernels gives results superior to either type of kernel alone. In order to determine the relative importance of the network topology compared to chemical and sequence similarity, we have investigated the change in prediction performance when varying the parameters α_d and α_t between 0 (chemical/genomic kernels only) and 1 (interaction profiles kernels only). For computational reasons we have used 10-fold cross-validation instead of leave-one-out.

In Figure 8.3 we have plotted the AUPR and AUC scores on the GPCR dataset for the different parameter values. Lighter colors correspond to higher values. Because of space limitations, plots for the other datasets are included in Supplementary Figures S1 and S2. For all datasets the optimal AUPR is obtained using a mix of the drug and target kernels. Using the parameters $\alpha_d = \alpha_t = 0.5$, as we did in the previous section, seems to be a good choice across the datasets. Also note that the choice of α_d is more important than the choice of α_t. This seems to indicate that the sequence similarity for targets is more informative than the

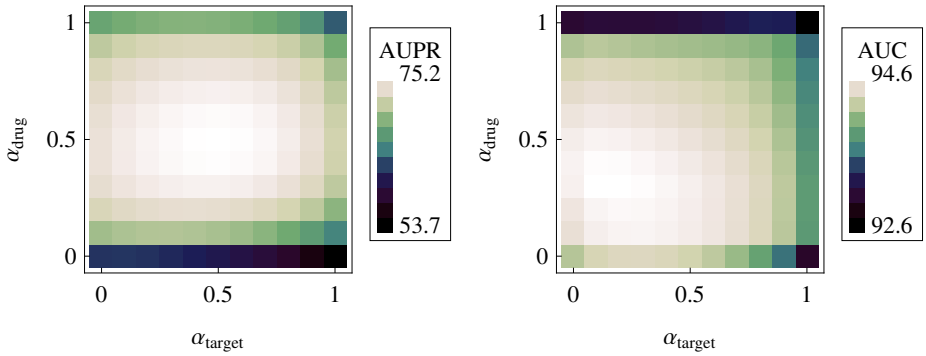


Figure 8.3: AUPR and AUC scores for the GPCR dataset with different weightings of the kernels. Lighter colors are better. For all datasets $\alpha_d = \alpha_t = 0.5$ gives near optimal results.

chemical similarity for drugs. A similar observation was also made in Bleakley and Yamanishi (2009). The poor performance of the RLS-Kron method when using only chemical and genomic kernels that we observed in the previous section appears to be due entirely to this uninformative chemical similarity.

On the larger datasets (Enzyme and Ion Channel) the optimal AUC is obtained with $\alpha_d = 1$, while that choice gives the worst results on the smaller datasets. This can be explained by noting that when there are few drugs, there is less information available for each entry of GIP target kernel, and hence this kernel will be of a lower quality. We have confirmed this hypothesis by testing different sized subsets of the Ion Channel dataset, where we observe the same effect on small subsets. The full results of that experiment are available in Supplementary Figure S3.

8.7.3 New predicted interactions

In order to analyze the practical relevance of the method for predicting novel drug–target interactions, we conducted an experiment similar to that described by Bleakley and Yamanishi (2009). We ranked the non-interacting pairs according to the scores computed for leave-one-out cross-validation experiments. We estimate the most highly ranked drug–target pairs as most likely to be putative interactions. A list of the top 20 new interactions predicted for each of the four data sets can be found in Supplementary Tables S3–S6.

Table 8.2 lists the top 10 new interactions predicted for the GPCR dataset. We have looked up these predicted interactions in ChEMBL (Overington, 2009) (version 9), DrugBank (Wishart *et al.*, 2008) and the latest online version of KEGG DRUG (Kanehisa *et al.*, 2006). A significant fraction of the predictions (4 out of 10) is found in one or more of these databases. One should bear in mind that a large fraction of the interactions in these databases are already included in the training data, and hence are not counted as new interactions. Moreover these databases are incomplete, so if a predicted interaction is not present in one of the used databases, this does not necessarily mean it does not exist. For this dataset, we started with only 635 known drug–target interactions and 20550 drug–target pairs not known to interact. Of these 20550 we selected 10 as putative drug–target interaction, and found that at least 4 of them are experimentally verified. These findings support the practical relevance of the proposed method.

We compared the newly predicted interactions generated by RLS-Kron-avg and those generated by Bleakley and Yamanishi (2009), here referred to as BY09. Specifically, given a dataset, for each method we extracted from its top x new predictions those that have been experimentally validated (that is, that could be found in ChEMBL, DrugBank or KEGG DRUG). Table 8.3 contains a summary

Table 8.2: The top 10 new interactions predicted in the GPCR dataset, 4 have been confirmed. Interactions that appear in the ChEMBL database are marked with "[C]", interactions in Drugbank are marked with "[D]", and interactions in Kegg are marked with "[K]". The NN column gives the similarity to the nearest drug interacting with the same target, and to the nearest target interacting with the same drug.

Rank	Pair	Description	NN
1	**D00283**	**Clozapine**	0.769
[C,D]	**hsa1814**	**DRD3: dopamine receptor D3**	0.455
2	**D02358**	**Metoprolol**	0.750
[C,D]	**hsa154**	**ADRB2: beta-2 adrenergic receptor**	0.434
3	D00604	Clonidine hydrochloride	0.933
	hsa147	ADRA1B: alpha-1B adrenergic receptor	0.435
4	D03966	Eglumegad	0.036
	hsa2914	GRM4: glutamate receptor, metabotropic 4	0.768
5	D00255	Carvedilol	0.380
	hsa152	ADRA2C: alpha-2C adrenergic receptor	0.489
6	**D04625**	**Isoetharine**	0.737
[K]	**hsa154**	**ADRB2: beta-2 adrenergic receptor**	0.434
7	D03966	Eglumegad	0.036
	hsa2917	GRM7: glutamate receptor, metabotropic 7	0.758
8	**D02340**	**Loxapine**	0.769
[D]	**hsa1812**	**DRD1: dopamine receptor D1**	0.205
9	D00503	Perphenazine	0.857
	hsa1816	DRD5: dopamine receptor D5	0.529
10	D00682	Carboprost tromethamine	0.914
	hsa5739	PTGIR: prostaglandin I2 receptor (IP)	0.150

of the results for $x = 20, 50, 80$. Looking at the top 20 predictions it seems that the two methods perform best on different datasets. For the top 50 and top 80 predictions, the results indicate the capability of RLS-Kron-avg to predict successfully more new interactions than BY09.

We then compared the resulting two sets of confirmed new predictions among the top 50, by looking at common predictions and at interactions uniquely predicted by only one of the two methods. The results for the four datasets can be found in Supplementary Tables S7–S10.

On the Enzyme dataset BY09 and RLS-Kron-avg both successfully predicted 15 new interactions, with 10 common predictions. On the Ion Channel dataset, BY09 and RLS-Kron-avg successfully predicted 14 and 12 new interactions, respectively, of which only 1 interaction was predicted by both methods. Although

Table 8.3: The number of highly ranked new interactions that are found in at least one of the three considered databases (ChEMBL, DrugBank or KEGG DRUG).

Dataset	Method	Top 20	Top 50	Top 80
Enzyme	BY09	6 (30%)	15 (30%)	17 (21%)
	RLS-Kron-avg	11 (55%)	15 (30%)	22 (28%)
Ion Channel	BY09	11 (55%)	14 (28%)	18 (22%)
	RLS-Kron-avg	8 (40%)	12 (24%)	22 (28%)
GPCR	BY09	13 (65%)	22 (44%)	30 (38%)
	RLS-Kron-avg	9 (45%)	28 (56%)	40 (50%)
Nuclear Receptor	BY09	5 (25%)	15 (30%)	22 (28%)
	RLS-Kron-avg	9 (45%)	20 (40%)	22 (28%)

BY09 found slightly more confirmed interactions they were less diverse, since 11 of them involve interactions between (different types of) the voltage-gated sodium channel alpha subunit target and only 2 drugs: Prilocaine and Tocainide. On the other hand, RLS-Kron-avg found interactions 4 different classes of targets and 10 different drugs. On the GPCR dataset, BY09 and RLS-Kron-avg successfully predicted 22 and 28 new interactions, respectively, with 14 common predictions. Finally, on the Nuclear Receptor dataset, BY09 and RLS-Kron-avg successfully predicted 15 and 20 new interactions, respectively. Among them, 13 were in common.

In general, the two methods seem to differ in the type of new predictions made. While there is always an overlap of new interactions between the two methods, there is also always a subset of new interactions which RLS-Kron-avg can successfully predict but BY09 fails to predict and vice-versa. Moreover, there seems to be a slight tendency of BY09 to generate new successful predictions that are less diverse than those generated by RLS-Kron-avg. However, we were not able to identify any differential biological bias of the methods towards the detection of specific types of interactions.

8.7.4 Surprising interactions

A closer inspection shows that many of the predicted interactions are not very surprising. For example, the GPCR dataset contains the interaction between Clozapineand Dopamine receptor D1. The drug Loxapineis very similar to Clozapine, and it is therefore to be expected that our method also predicts Loxapineto interact with Dopamine receptor D1. An analogous thing happens

with very similar target proteins. In order to provide a quantitative measure of how surprising these predictions are, we computed the similarity of a the drug and target in an interaction pair to their Nearest Neighbor (NN), that is, the most similar drug (with respect to chemical structure similarity) and target (with respect to sequence similarity) in the training set, respectively. These similarities, which we call surprise scores, are listed in the NN column of Table 8.2. An inspection of the surprise scores shows that the majority of the drug–target pairs predicted by our method consist of a drug and a target very similar to a drug and a target already known to interact, and therefore they are not very surprising. This phenomenon is common to any computational approach that uses similarity between objects for inferring interaction.

To assess the ability of our method to also predict more surprising interactions, we have looked specifically at the predicted interactions where there is no similar drug interacting with the same target or similar target interacting with the same drug in the dataset. We pick a threshold value and consider drugs (targets) to be dissimilar if their chemical (genomic) similarity is less than this threshold. We have used the threshold 0.5 for the chemical similarity and 0.25 for the genomic similarity.

When only these 'surprising' pairs are considered, we find, as expected, that fewer of them are present in the ChEMBL, DrugBank and KEGG databases. But we still find more interactions among the highly ranked 'surprising' pairs compared to to those that are ranked lower. For example, on the GPCR dataset, 89 of the 500 highest ranked pairs were surprising, and 10 of them (11%) were found in one of the databases. See the online Suplementary Material for details.

8.8 Discussion

We have presented a new kernel that leads to good predictive performance as measured by AUPR on the task of predicting interactions between drugs and target proteins. An interesting aspect of our Gaussian Interaction Profile kernel is that it uses no properties beyond the interactions themselves. This means that knowing the sequence of proteins and chemical structure of drugs is perhaps not as important for this task as previously thought. For example, on the Ion Channel dataset our method with only the GIP kernel has an AUC score of 98.6 and an AUPR score of 92.7, which improves upon the state-of-the-art, while using less prior information.

Besides the GIP kernel we have also introduced the RLS-Kron algorithm that combines a kernel on drugs and a kernel on targets using the Kronecker product. Compared to previous methods that do prediction with the two kernels inde-

pendently and then combine the results, this new method represents a small but consistent improvement.

By combining the GIP kernel with chemical and genomic information we get a method with excellent performance. This method has AUPR scores of 91.5, 94.3, 79.0 and 68.4 on four datasets of drug–target interaction networks in humans, representing an average improvement of 10 points over previous results. The AUPR is a particularly relevant metric for this problem, because it is very sensitive to the correctness of the highest ranked predictions. The large improvement in AUPR suggests that the top ranked putative drug–target interactions found by our method are more likely to be correct than those found with previous methods.

A limitation of all machine learning methods for finding new drug–target interactions is that they are sensitive to inherent biases contained in the training data. It would be interesting to try and analyze the bias of existing datasets of drug–target interaction, but this is out of the scope of the present chapter.

Note also that the datasets by Yamanishi, Araki, *et al.* (2008) used in this chapter do not include any singletons: each drug interacts with at least one target, and each target interacts with at least one drug. This property could affect the cross-validation results, by allowing a limited form of cheating. However, the experiments in Section 8.7.3 show that our method also works when tested in other ways. In Chapter 10 we further investigate this bias in the dataset.

A further limitation of the approach used in this chapter is that it can only be applied to detect new interactions for a target or a drug for which at least one interaction has already been established. Therefore, biologists can use the method as guidance for extending their knowledge about the interaction of a drug or of a target, not for discovering interactions of a new drug or target (that is, one for which no interaction is known). In particular, our method is useful for experimentalist to aid in experimental design and interpretation, especially in solving problems related to drug-target selectivity and polypharmacology (Metz and Hajduk, 2010; Merino *et al.*, 2010). In the next chapter (Chapter 9) we address this issue, and extend the method to work for drugs and target proteins with no known interactions.

There are several ways in which the result might further be improved. So far we have used uninformative choices of the parameters: $\tilde{\beta} = 1$, $\sigma = 1$ and $\alpha = 0.5$. Of these choices we have only investigated the last one. Perhaps with tuning of the other parameters better predictions are possible, although one has to be careful not to over-fit them to the data.

Another avenue for improvement is in using more information about drugs and targets. Since combining the GIP kernel with chemical and genomic kernels leads to a better predictive performance, perhaps adding different information in

the form of additional kernels would yield further improvements. These kernels could be interaction profile kernels based on other types data, such as protein–protein interaction networks. Similarly, for each pair of interacting drug and target more information is known beyond the fact they interact. For example, the type of interaction, the binding strength, the mechanism of discovery and its uncertainty might all be known. In this chapter we have made no use of this additional information, nor did we attempt to predict the type or strength of interactions.

Chapter 9

Interaction prediction for new drugs with weighted nearest neighbor

In silico discovery of interactions between drug compounds and target proteins is of core importance for improving the efficiency of the laborious and costly experimental determination of drug–target interaction. Drug–target interaction data are available for many classes of pharmaceutically useful target proteins including enzymes, ion channels, GPCRs and nuclear receptors. However, current drug–target interaction databases contain a small number of drug–target pairs which are experimentally validated interactions. In particular, for some drug compounds (or targets) there is no available interaction. This motivates the need for developing methods that predict interacting pairs with high accuracy also for these 'unseen' drug compounds (or targets). We show that a simple weighted nearest neighbor procedure is highly effective for this task. We integrate this procedure into a recent machine learning method for drug–target interaction we developed in previous work. Results of experiments indicate that the resulting method predicts true interactions with high accuracy also for unseen drug compounds and achieves results comparable or better than those of recent state-of-the-art algorithms.

9.1 Introduction

In this chapter we generalize the applicability of the method introduced in the previous chapter to so-called *unseen drug compounds*, that is, drug compounds for which no interactions are known. The method, hereafter referred to as just

This chapter is based on van Laarhoven and Marchiori (2013b), "Predicting Drug-Target Interactions for New Drug Compounds Using a Weighted Nearest Neighbor Profile", published in PLoS ONE. The software and datasets used in this chapter is available at `http://cs.ru.nl/~tvanlaarhoven/drugtarget2013/`.

GIP, uses known interactions of a drug for predicting novel ones by means of a regularized least square algorithm incorporating a product of kernels constructed from drug compound and target interaction profiles. We propose a simple weighted nearest neighbor algorithm, called WNN, for constructing an interaction score profile for a new drug compound using chemical and interaction information about known compounds in the dataset. The WNN method can be used as a stand-alone algorithm for predicting interactions for unseen drug compounds. It can also be directly incorporated into the GIP method for handling unseen drug compounds. We call the resulting combination WNN-GIP. The methods can be directly adapted to handle also unknown targets or both unknown drug compounds and targets.

We test the predictive performance of WNN and WNN-GIP on four drug–target interaction networks in humans involving enzymes, ion channels, GPCRs and nuclear receptors. Results as measured by the area under the curve (AUC) and area under the precision-recall curve (AUPR) (Davis and Goadrich, 2006) show that the weighted nearest neighbor profile algorithm and its incorporation into the GIP method are capable to predict true interactions for unseen drug compounds with satisfactory accuracy. The algorithms achieve competitive or better results than the recent state-of-the-art algorithms KBMF2K (Gönen, 2012) and BLM-NII (Mei *et al.*, 2013). KBMF2K is based on a fully probabilistic approach to model drug-target interaction, which can be applied to discover target (respectively drug compound) interactions for unseen drug compounds (respectively target proteins). Results in Gönen (2012) indicate improved accuracy over the method introduced in Yamanishi, Kotera, *et al.* (2010). BLM-NII is an extension of the BLM method (Bleakley and Yamanishi, 2009) to deal with unseen drug compounds (or targets). In BLM-NII a drug-target interaction for a unseen drug compound is inferred by constructing an estimated interaction profile from the drug compounds in the training data. The resulting profile is then used as label information to learn an interaction model for that drug compound with the BLM method.

9.2 Methods

9.2.1 The GIP method

Machine learning methods for tackling the drug–target interaction problem are mainly based on the assumption that drug compounds exhibiting a similar pattern of interaction and non-interaction with the target proteins in a drug–target interaction network are also likely to show similar interaction behavior with respect to unseen targets. And a similar assumption on the target proteins is also

made. Here we use the method introduced in Chapter 8. It is based on the so-called (target) *interaction profile* y_{d_i} of a drug compound d_i, which is defined to be row i of the adjacency matrix Y; and on the (drug compound) interaction profile $y_{t_j}^T$ of a target protein t_j, defined to be column j of Y. The interaction profiles generated from a drug–target interaction network are used as feature vectors for a classifier. A kernel from the interaction profiles is constructed using topology of the drug–target network, defined for drug compounds d_i and d_j as follows:

$$K_{GIP,d}(d_i, d_j) = \exp(-\beta_d \|y_{d_i} - y_{d_j}\|^2).$$

where

$$\beta_d = \tilde{\beta}_d \Big/ \Big(\frac{1}{n_d} \sum_{i=1}^{n_d} |y_{d_i}|^2\Big).$$

A kernel $K_{GIP,t}$ for the similarities between target proteins is defined analogously. Moreover, the kernels $K_{chemical,d}$ and $K_{genomic,t}$ are considered, containing information about the chemical and genomic space. They are constructed from the chemical and genomic similarity matrices S_d and S_g between drug compounds and between targets, by applying a simple transformation to make them symmetric and positive definite. The interaction profile kernel can be easily combined with these kernels using a weighted average.

The kernel for drug compounds and the kernel for target proteins can be combined using the Kronecker product $K_d \otimes K_t$, such that for drug-target pairs (d_i, t_i) and (d_j, t_j)

$$K((d_i, t_i), (d_j, t_j)) = K_d(d_i, d_j) K_t(t_i, t_j).$$

For each drug compound with at least one known interaction in the training data, a score interaction profile $\hat{y}$ is computed from its interaction profile y and the kernel matrix K, using the Regularized Least Squared (RLS) classifier. This is achieved by means of the simple closed form solution

$$\hat{y} = K(K + \sigma I)^{-1} y,$$

where σ is a regularization parameter.

We refer the reader to Chapter 8 for a more detailed description and analysis of this method.

For simplicity, in the rest of this chapter we call the method that uses the the RLS algorithm and the Kronecker product kernel 'GIP'.

9.2.2 Weighted nearest neighbor for unseen drug compounds

We want to extend GIP to unseen drug compounds, that is, compounds for which no interaction is known. To this aim, we propose a simple weighted nearest neighbor procedure. For an unseen drug compound, its chemical similarity with other known drug compounds and their corresponding profiles are used in order to infer a score interaction profile for that drug compound.

Specifically, the score interaction profile y^d_{WNN} of an unseen drug compound d is the weighted sum of the profiles of the drug compounds in the training data, where a higher weight is assigned to profiles of those drug compounds more similar to d. Let $y_1, \ldots, y_{n_d}$ be the interaction profiles of the other compounds in the dataset (that is, the rows of Y), listed in decreasing order with respect to their chemical similarity to d. Then

$$y^d_{WNN} = \sum_{i=1}^{n_d} w_i y_i,$$

where the weights w_i's are computed using a given decay value $T \leqslant 1$ as $w_i = T^{i-1}$. For computational reasons we only sum over drug compounds with weight at least 10^{-4}. In our experiments we choose the decay rate T with 5 fold cross-validation to maximize AUC. We call the resulting procedure WNN.

An extension of GIP to handle unseen drug compounds using WNN, hereafter called WNN-GIP, can be directly formulated: for each new drug compound d, add y^d_{WNN} as new row to the matrix Y and apply GIP to predict the score interaction profile $\hat{y}$ of d.

9.2.3 A method to show the bias of a LOOCV procedure

In a recent paper (Mei *et al.*, 2013) the BLM-NII algorithm is introduced and assessed using the following leave-one-out cross validation (LOOCV) procedure. Each compound with only one interaction in Y is treated as a 'new candidate' in the cross validation and the BLM-NII procedure is applied to it. We observe that in this way a strong prior is implicitly used in the cross validation, namely the fact that the considered compound had at least one interaction.

To illustrate how this prior introduces a bias on the results, we consider the following simple procedure, called Const. Const constructs an all '1's profile for the drug compounds or target proteins with only one interaction.

We can incorporate Const into GIP in the same way as WNN, giving the Const-GIP method. With this method all possible interactions for drug/targets with only one interaction will be ranked before interactions with drugs/targets that also have other interactions. Essentially, for such interactions the method

only has to do half the work, since the fact that the drug/target is correct can be known with certainty. In real world situations there are also drug compounds that interact with none of the target under consideration, and vice versa, which would invalidate the Const-GIP method.

9.3 Experiments

We perform a comparative experimental analysis of the proposed algorithms and two recently published methods, Gönen (2012) and Mei *et al.* (2013) .

We follow the experimental procedure adopted in Yamanishi, Kotera, *et al.* (2010) and Gönen (2012). Specifically, for each dataset, drug compounds are split into five subsets of roughly equal size. Each subset is then used in turn as the test set and training is performed on the data consisting of the remaining four subsets. This procedure is repeated five times.

Results are assessed using the AUC and AUPR quality measures, generally used in this type of studies. Specifically, the ROC curve of true positives as a function of false positives is computed, and the area under the ROC curve (AUC) is considered as quality measure (see for instance Fawcett, 2006). Furthermore, the precision–recall curve is computed, that is, the plot of the ratio of true positives among all positive predictions for each given recall rate. The area under this curve (AUPR) provides a quantitative assessment of how well, on average, predicted scores of true interactions are separated from predicted scores of true non-interactions. Since there are few true drug–target interactions, the AUPR is a more informative quality measure than the AUC, as it punishes much more the existence of false positive examples found among the top ranked prediction scores (Davis and Goadrich, 2006).

Average AUC and AUPR results and standard deviations are reported in Table 9.1. They indicate that a WNN-GIP has slightly better (average) AUC on all datasets except Enzyme. However, WNN has slightly better AUPR than WNN-GIP. By itself the GIP method does not work well in this setting, which is to be expected, since it was not designed to handle unseen drugs.

To estimate the statistical significance of the AUC results we used the method described in E. R. DeLong, D. M. DeLong, and Clarke-Pearson (1988). To determine significance of the AUPR results we used bootstrapping.

The last column of Table 9.1 lists the average value of the decay rate T over the folds and repetitions. In general, the larger dataset have a higher (slower) decay rate, which means that more neighbors are taken into account.

Table 9.1: Results of 5 fold cross validation: average AUC and AUPR over 5 runs. Standard deviation is reported between parentheses. The best AUC and AUPR results are indicated in bold, results that are not significantly different from the best (at $\alpha = 0.05$) are indicated in italic.

Dataset	Method	AUC (std)	AUPR (std)	T (std)
Enzyme	GIP	0.685 (0.006)	0.150 (0.008)	
	WNN	0.819 (0.004)	**0.299** (0.023)	0.809 (0.068)
	WNN-GIP	**0.861** (0.004)	*0.280* (0.014)	0.908 (0.019)
	KBMF2K	0.812 (0.004)	*0.287* (0.021)	
Ion Channel	GIP	*0.637* (0.008)	*0.179* (0.013)	
	WNN	0.757 (0.006)	**0.249** (0.046)	0.535 (0.200)
	WNN-GIP	0.775 (0.006)	*0.233* (0.024)	0.730 (0.171)
	KBMF2K	**0.802** (0.006)	*0.245* (0.023)	
GPCR	GIP	0.679 (0.014)	0.260 (0.023)	
	WNN	0.848 (0.008)	0.308 (0.032)	0.713 (0.084)
	WNN-GIP	**0.872** (0.008)	*0.311* (0.021)	0.702 (0.081)
	KBMF2K	0.840 (0.009)	**0.347** (0.028)	
Nuclear Receptor	GIP	0.758 (0.026)	0.357 (0.060)	
	WNN	0.788 (0.027)	*0.434* (0.068)	0.305 (0.205)
	WNN-GIP	**0.839** (0.023)	**0.456** (0.065)	0.527 (0.103)
	KBMF2K	*0.810* (0.025)	0.354 (0.063)	

9.3.1 Comparison with other methods

We consider the two following recent methods: KBMF2K (Gönen, 2012) and BLM-NII (Mei *et al.*, 2013).

KBMF2K is based on a Bayesian formulation that combines dimensionality reduction, matrix factorization and binary classification for predicting drug–target interaction networks using only chemical similarity between drug compounds and genomic similarity between target proteins.

In BLM-NII a drug-target interaction for an unseen drug compound d is inferred by constructing an estimated interaction profile for d as follows. For each target, an entry of the profile for d is defined as the sum of the similarity values of d and each of the drug compounds interacting with that target. The resulting profile is then used as label information to learn an interaction model for d by means of the BLM method.

Comparison with KBMF2K

To compare results of WNN and WNN-GIP with those reported in Gönen (2012), we follow the experimental procedure therein used (described in the previous section). Table 9.1 also includes the AUC and AUPR for the KBMF2K method. They indicate similar performance of KBMF2K and the simpler WNN algorithm, and slightly better overall results achieved by WNN-GIP, except on the Ion Channel dataset.

We could test the prediction capability of the proposed methods on unknown drug–target interactions of the given network using the procedure adopted in Gönen (2012). Therein, the complete interaction network for each dataset is used as training data, and the predictions on non-interacting pairs in the training set are ranked with respect to their interaction scores. However, since each drug compound or target in the training set has at least one interaction, we do not need to use WNN and the results are those of GIP. We report the top five predicted interactions for each dataset in Table 9.2 and Table 9.3. The full lists of all predicted interactions ranked by interaction score can be found in `http://cs.ru.nl/~tvanlaarhoven/drugtarget2013/`.

Comparison with BLM-NII

Table 9.4 shows the results of the LOOCV experiments. As expected, both Const-GIP and BLM-NII achieve very good results, with comparable AUC, and slightly better AUPR performance achieved by Const-GIP. To asses the statistical significance of these differences we used an upper bound on the variance of the AUC and AUPR for BLM-NII, because the actual variance is unknown. With this bound the differences in AUC scores are not statistically significant.

In general, these results indicate that cross validation should be applied and interpreted with care. Note that the cross validation procedure used in the comparison with KBMF2K is also positively biased, since we know that each 'unseen' drug compound has at least one interaction, but there the bias is much smaller.

9.4 Discussion

In this work, we proposed a simple yet effective procedure to predict interaction profiles for unknown drug compounds and show how it can be directly integrated into a recent machine learning algorithm for the in-silico prediction of drug–target interactions. The novelty of our approach comes in the use of a weighted nearest neighbor procedure for inferring a profile for a drug compound by using interaction profiles of the compounds in the training data, where

Table 9.2: Highest ranked predicted new interactions for each of the datasets. Interactions found in ChEMBL, Matador, DrugBank and KEGG are indicated in italic and marked as C, M, D and K respectively.

Rank / Found	Drug compound / Target protein	
Enzyme		
1	*D00574*	*Aminoglutethimide*
[M]	*hsa1589*	*cytochrome P450, family 21, subfamily A, polypeptide 2*
2	*D00542*	*Halothane*
[C,M,D]	*hsa1571*	*cytochrome P450, family 2, subfamily E, polypeptide 1*
3	*D00139*	*Methoxsalen*
[M,D]	*hsa1543*	*cytochrome P450, family 1, subfamily A, polypeptide 1*
4	*D00437*	*Nifedipine*
[M]	*hsa1585*	*cytochrome P450, family 11, subfamily B, polypeptide 2*
5	*D00437*	*Nifedipine*
[C,M,D]	*hsa1559*	*cytochrome P450, family 2, subfamily C, polypeptide 9*
Ion Channel		
1	*D00438*	*Nimodipine*
[D,K]	*hsa779*	*calcium channel, voltage-dependent, L type, alpha 1S subunit*
2	D00726	Metoclopramide
	hsa1138	cholinergic receptor, nicotinic, alpha 5 (neuronal)
3	*D03365*	*Nicotine*
[C,D]	*hsa1137*	*cholinergic receptor, nicotinic, alpha 4 (neuronal)*
4	D02098	Proparacaine hydrochloride
	hsa8645	KCNK5: potassium channel, subfamily K, member 5
5	*D00552*	*Benzocaine*
[K]	*hsa6331*	*sodium channel, voltage-gated, type V, alpha subunit*

Table 9.3: Continuation of Table 9.2.

Rank / Found	Drug compound / Target protein	
GPCR		
1	*D00283*	*Clozapine*
[C,M,D]	*hsa1814*	*dopamine receptor D3*
2	*D02358*	*Metoprolol*
[C,D]	*hsa154*	*adrenoceptor beta 2, surface*
3	D00604	Clonidine hydrochloride
	hsa147	adrenoceptor alpha 1B
4	*D03966*	*Eglumetad*
[C]	*hsa2914*	*glutamate receptor, metabotropic 4*
5	*D00255*	*Carvedilol*
[C]	*hsa152*	*adrenoceptor alpha 2C*
Nuclear Receptor		
1	D00316	Etretinate
	hsa6096	RAR-related orphan receptor B
2	*D00182*	*Norethindrone*
[C]	*hsa2099*	*estrogen receptor 1*
3	*D00348*	*Isotretinoin*
[K]	*hsa5915*	*retinoic acid receptor, beta*
4	D01132	Tazarotene
	hsa6097	RAR-related orphan receptor C
5	*D00348*	*Isotretinoin*
[K]	*hsa5916*	*retinoic acid receptor, gamma*

each profile is weighted using information about chemical similarity between drug compounds integrated with a simple decay scheme. The method can be directly modified to predict interaction scores of unknown targets (or of both unknown targets and drug compounds).

We performed a comparative assessment of the proposed methods on four different drug–target interaction networks from humans involving enzymes, ion channels, GPCRs and nuclear receptors. Results indicated that WNN is competitive in predicting interaction for unknown drug compounds with more involved machine learning methods recently proposed, notably a fully probabilistic method based on a Bayesian formulation that combines kernel-based nonlinear dimensionality reduction, matrix factorization and binary classification.

Table 9.4: Results of LOOCV on pairs. Results of BLM-NNII are from Mei *et al.* (2013). The best AUC and AUPR results are indicated in bold, results that are not significantly different from the best (at $\alpha = 0.05$) are indicated in italic, see the main text for details.

Dataset	Method	AUC	AUPR
Enzyme	GIP	0.978	0.915
	WNN	0.558	0.141
	WNN-GIP	0.983	0.944
	Const	0.577	0.179
	Const-GIP	**0.991**	**0.969**
	BLM-NII	*0.988*	0.929
Ion Channel	GIP	0.984	0.943
	WNN	0.528	0.125
	WNN-GIP	0.986	0.953
	Const	0.535	0.138
	Const-GIP	**0.991**	**0.966**
	BLM-NII	*0.990*	0.950
GPCR	GIP	0.954	0.790
	WNN	0.580	0.219
	WNN-GIP	0.972	0.863
	Const	0.604	0.266
	Const-GIP	**0.988**	**0.910**
	BLM-NII	*0.984*	0.865
Nuclear Receptor	GIP	0.922	0.684
	WNN	0.694	0.478
	WNN-GIP	0.958	0.857
	Const	0.744	0.568
	Const-GIP	**0.989**	**0.926**
	BLM-NII	*0.981*	0.866

Furthermore we showed that the direct integration of WNN in a recent kernel based machine learning method provides a general and powerful tool for finding drug-target interactions.

The computational complexity of WNN is $O(n_d^2 + n_t^2)$, while the computational complexity of WNN-GIP is dominated by the RLS prediction using the Kronecker product kernel, which is $O(n_d^3 + n_t^3)$ as implemented in Chapter 8, but can be further improved yielding a quadratic computational complexity by applying recent techniques for large-scale kernel methods for computing the two kernel decompositions, e.g. Kashima, Idé, *et al.* (2009). Therefore WNN-GIP is more efficient than KBMF2K, since the total time complexity of *each iteration* in the variational approximation method used in KBMF2K is $O(Rn_d^3 + Rn_t^3 + R^3)$, where R is the subspace dimensionality used in the method.

A limitation of our approach is that it does not make a difference between an inactive target and a target that has not been measured for a compound.

Compounds with a higher mutual chemical similarity also have a higher chance of having the same bioactivity. This information could be considered by WNN by determining directly the weights from the similarity, instead of using the proposed ranking-based decay mechanism. In this way all the compounds with high similarity would be considered with a high weight and all the compounds with low similarity would only have a minor contribution to the final predicted profile. On the same reasoning there is also a similarity threshold from where the chance is so low that two compounds have the same profile that it would be better not to predict something in the first place. In particular for new screening data from very large screening libraries chances are high that none of the references are really similar to the screening hits, which would most likely have a detrimental effect in the overall prediction performance, if predictions would be made for all such compounds. Many published target prediction algorithms apply such "applicability domain" or confidence estimations for their predictions. WNN could be modified to address this issue for instance by including a binary annotation based on a similarity threshold, or a more advanced procedure based on the similarities of all compounds considered for the generation of the profile.

Chapter 10

Biases in drug–target interaction data

Network based prediction of interaction between drug compounds and target proteins is a core step in the drug discovery process. The availability of drug–target interaction data has boosted the development of machine learning methods for the in silico *prediction of drug–target interactions. In this chapter we focus on the crucial issue of data bias.*

We show that four popular datasets contain a bias because of the way they have been constructed: all drug compounds and target proteins have at least one interaction and some of them have only a single interaction. We show that this bias can be exploited by prediction methods to achieve an optimistic generalization performance as estimated by cross-validation procedures, in particular leave-one-out cross validation. We discuss possible ways to mitigate the effect of this bias, in particular by adapting the validation procedure. In general, results indicate that the data bias should be taken into account when assessing the generalization performance of machine learning methods for the in silico *prediction of drug–target interactions.*

The datasets and source code for this article are available at `http://cs.ru.nl/~tvanlaarhoven/bias2014/`

10.1 Introduction

In our recent work on predicting drug–target interactions described in Chapter 9, we discovered that a positive bias was implicitly introduced in a published method. This motivated the two main research questions we will address in this chapter:

This chapter is based on van Laarhoven and Marchiori (2014b), "Biases of drug–target interaction network data", published in the proceedings of the 9th international conference on Pattern Recognition in Bioinformatics. The datasets and source code for this chapter are available at `http://cs.ru.nl/~tvanlaarhoven/bias2014/`

1. How does data bias affect the results of procedures used to estimate the generalization performance of a method?

2. Can we quantify and avoid such bias?

Cross-validation (CV) (Kohavi, 1995) is typically used to assess the generalization performance of methods in the above mentioned settings. The dataset is repeatedly partitioned into two disjoint parts, a training set and a hold-out set. For each partition, the training set is used to construct the predictor and the hold-out set is used for testing. Popular variants are 10-fold CV, where the data is partitioned into ten folds, and each fold is used once as the hold-out set, and leave-one-out cross-validation (LOOCV), where each example constitutes one hold-out set. In the context of drug–target interaction various cross-validation settings can be defined, depending on what is considered an example (e.g. a single drug–target pair or all interactions with a single drug compound) and on the selected CV procedure.

As before, we consider the four popular drug–target interaction datasets in humans involving enzymes, ion channels, G-protein-coupled receptors (GPCRs) and nuclear receptors from Yamanishi, Araki, *et al.* (2008). These data have been used as benchmark datasets in many recent works, e.g. Bleakley and Yamanishi (2009), Chen, Liu, and Yan (2012), Gönen (2012), van Laarhoven, Nabuurs, and Marchiori (2011), van Laarhoven and Marchiori (2013b), Mei *et al.* (2013), and Wassermann, Geppert, and Bajorath (2009).

In this chapter we show experimentally that these datasets contain a bias which may lead to optimistic CV generalization results. Furthermore, the extent to which this bias affects the results can differ for different methods. As a result, it is unclear whether a method with better CV results on these datasets will also have better performance in real applications.

Specifically, these datasets have been constructed in such a way that each drug compound and target protein has at least one interaction. Furthermore, some drug compound and/or targets have only a single interaction.

We show how this bias can be incorporated into a baseline prediction method in such a way that it significantly increases the LOOCV generalization performance. We investigate how this bias can be reduced and quantified. We show experimentally that 5- or 10-fold CV reduces (but does not eliminate) the bias. Furthermore, the presence of this bias can be quantified by separating the performance metrics for drug compounds and targets with just one interaction from that for other drug–target interaction pairs. This provides an alternative procedure to assess the generalization performance of a method by highlighting the effect of the data bias.

In general, our results provide a contribution towards the understanding of CV procedures in the presence of data bias in the context of drug-target interaction networks.

10.1.1 Related work

Dataset bias has been investigated in different domains, e.g. in ligand based virtual screening (Baumann and Rohrer, 2008), where local clustering and global spread of the considered benchmark datasets were identified influencing validation results, and in object recognition (Torralba and Efros, 2011), where current state of recognition datasets have been comparatively analyzed and evaluated based on criteria including relative data bias and cross-dataset generalization. To the best of our knowledge, this is the first time that drug–target interaction network data bias is analyzed.

The dangers of CV have been studied by the machine learning community in various contexts. For instance, in Isaksson *et al.* (2008) CV and bootstrapping in small sample classification are investigated. A fundamental problem is that the uncertainty in a point estimate obtained with these procedures is unknown and may be quite large. The authors therefore suggest that the final classification performance should be reported in the form of a Bayesian confidence interval or using some other method that yields conservative measures of the uncertainty. Furthermore, in Rao and Fung (2008) it was empirically shown that when the number of algorithms is large, LOOCV is not an effective estimate of generalization performance for the algorithm that has the best cross-validation performance. The authors showed that this behavior worsens as the sample size decreases, and as the dimensionality and number of algorithms increase. The phenomenon of under-estimating cross validation error was also demonstrated on some benchmark data sets, and was shown to be worse for datasets with higher dimensionality.

10.2 Materials

In Yamanishi, Araki, *et al.* (2008) datasets were introduced for the drug–target prediction problem. These datasets are based on four different domains: enzymes, ion channels, GPCRs and nuclear receptors. The datasets are constructed in such a way that only the proteins that have an interacting drug are included, and for each domain only the drugs that interact with at least one protein are included. It turns out that this property introduces problems for validation.

Table 10.1: The number of drug compounds, the number of target proteins, the number of interactions and the number of unique interaction pairs (interactions which are the only one for a drug or target) in the drug–target datasets from Yamanishi, Araki, *et al.* (2008).

Dataset	Drugs	Targets	Interactions	Unique
Enzyme	445	664	2926	451 (15%)
Ion Channel	210	204	1476	103 (7%)
GPCR	223	95	635	132 (21%)
Nuclear Receptor	54	26	90	44 (49%)

In Table 10.1 we give an overview of the four datasets as they are used in recent publications[1]. As can be seen in the last column, a large fraction of the drug compounds and target have just one interaction in the dataset. Or equivalently, there are many interactions which are the only interaction for a drug–target. We call such interacting pairs *unique*.

As in previous chapters, the interactions in a dataset are encoded in a matrix y_{dt}, such that $y_{dt} = 1$ if drug compound d interacts with target protein t, and $y_{dt} = 0$ otherwise. Besides this interaction information, there is also other information available on the drugs and targets themselves, encoded in kernel matrices. We will not use the kernels in this chapter.

10.3 Validation procedures

As discussed in Chapter 7, there are four ways in which these datasets of interactions can be used by machine learning methods. Of these the two most common ones are:

1. The 'unseen drug' setting, that is, to train a model to predict with which targets a previously unseen drug will interact.

2. The 'pairs' setting, where the goal is to find new putative interactions between drugs and targets already in the dataset.

An overview of the prediction setting and type of CV used in state-of-the-art methods applied to these datasets are shown in Table 10.2. In this work we focus primarily on the 'pairs' setting, which is used by most of the methods listed in the table. We will briefly discuss the 'unseen drug' setting in our experiments as well.

[1]This table is the same as Table 7.1, but here we include information on the unique interactions.

Table 10.2: A list of papers that used the interaction data in Table 10.1, showing the type of prediction setting ('unseen drug' or 'pairs') and type of CV procedure used.

	Unseen drug	Pairs	CV procedure
Yamanishi *et al.*, 2008	✓	✓	10-fold CV
Bleakley *et al.*, 2009	✓	✓	LOOCV, 10-fold CV
van Laarhoven *et al.*, 2011[2]	-	✓	LOOCV, 10-fold CV
Chen *et al.*, 2012	-	✓	LOOCV
Gönen, 2012	✓	-	5-fold CV
Mei *et al.*, 2013	-	✓	LOOCV, 10-fold CV
van Laarhoven *et al.*, 2013b[3]	✓	✓	LOOCV, 5-fold CV

Usually methods are compared by looking at the ranking of interactions they produce in a cross-validation setting. That is, each drug–target pair is assigned a score by each method, where only other interacting pairs are shown to the method. Then the pairs are ranked based on these scores and the quality of the ranking is compared using AUC, AUPR or other summary statistics. Specifically, the ROC curve of true positives as a function of false positives is computed, and the area under the ROC curve (AUC) is considered as quality measure, see for instance Fawcett (2006). Furthermore, the precision–recall curve is computed, that is, the plot of the ratio of true positives among all positive predictions for each given recall rate. The area under this curve (AUPR) is a more informative quality measure than the AUC, as it punishes much more the existence of false positive examples found among the top ranked prediction scores (Davis and Goadrich, 2006).

10.4 Biases

Suppose that a method is tested using LOOCV. Then if a unique interaction $(\mathtt{d}, \mathtt{t})$ is left out, the method will see a row (or column) of zeros in the matrix. But we know that the dataset does not have such rows or columns, since each drug and target has at least one interaction. We can therefore know with certainty that this pair interacts. This process is illustrated in Figure 10.1.

Consider a simple baseline method, that ranks drug–target pairs by the number of adjacent pairs that are known to interact, where two drug–target pairs are

[2]Chapter 8 of this thesis.
[3]Chapter 9 of this thesis.

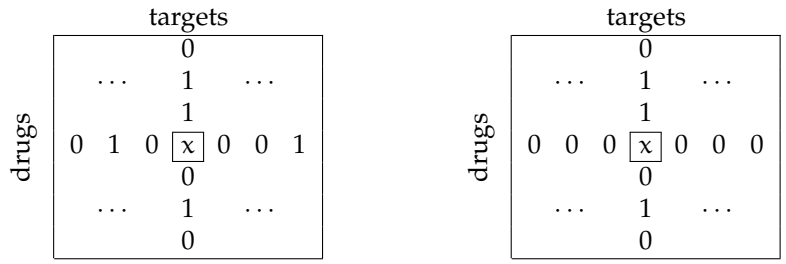


Figure 10.1: In the LOOCV procedure, the task is to predict a single unknown drug–target interaction, assuming all other interactions are known. This is indicated by x in the matrix of drug–target interactions. Because of the construction of the dataset, we can know with certainty that in the second matrix $x = 1$, otherwise this drug compound would not be included in the dataset.

adjacent if they share a drug or a target. This number of adjacent interacting pairs for the pair (d, t) is

$$a_{dt} = a_{dt}^{\text{drug}} + a_{dt}^{\text{targ}}, \quad \text{where} \quad a_{dt}^{\text{drug}} = \sum_{d' \neq d} y_{d't}, \quad a_{dt}^{\text{targ}} = \sum_{t' \neq t} y_{dt'}.$$

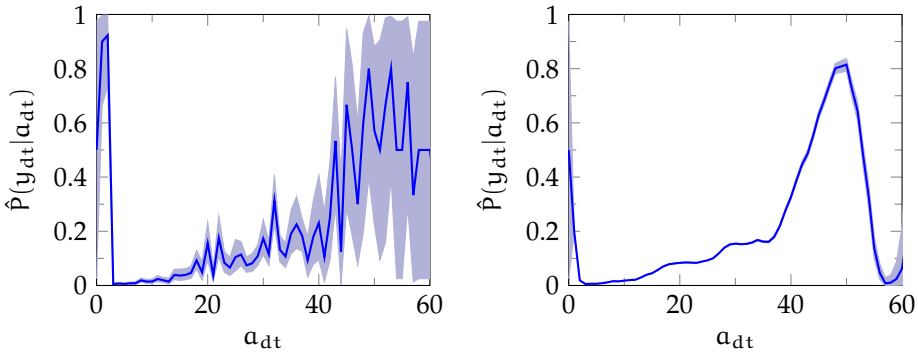
At first glance we would expect drugs or targets that already have many known interactions to be more promiscuous, and therefore also more likely to interact with other drugs and targets. But as explained in the previous paragraph that is not the case when LOOCV is used.

To test this effect, in Figure 10.2 we have plotted the fraction of pairs that interact against a_{dt}. This is an empirical estimate $\hat{P}(y_{dt}|a_{dt})$ of the probability that d and t interact given the number of adjacent pairs for (d, t). Overall there is indeed a trend for larger a_{dt} to correspond to a higher probability of interacting. But for very low a_{dt} we see the bias in action: the probability of such pairs interacting is very high, since many of them are unique interactions.

A method can exploit this knowledge as follows. Consider the biased variant of the baseline method, which is the same as the baseline, except that it ranks the pairs with no observed adjacent pairs sharing a drug or with no pairs sharing a target before all other drug–target pairs. More precisely, instead of ranking pairs by a_{dt}, they are ranked by

$$a_{dt}^{\text{unique}\mapsto\infty} = \begin{cases} \infty & \text{if } a_{dt}^{\text{drug}} = 0 \text{ or } a_{dt}^{\text{targ}} = 0 \\ a_{dt} & \text{otherwise.} \end{cases}$$

In Table 10.3 we compare the LOOCV performance of this biased method to the unbiased baseline.



(a) LOOCV (b) 10 fold CV

Figure 10.2: Probability of a drug–target pair interacting given the number of adjacent interactions. The first plot shows this probability for LOOCV in the GPCR dataset, the second plot for 10-fold cross validation. The shaded area indicates a 95% confidence interval based on a uniform prior.

To estimate the statistical significance of the AUC results we used the method described in E. R. DeLong, D. M. DeLong, and Clarke-Pearson (1988). To determine significance of the AUPR results we used bootstrapping.

The difference between the unbiased and the biased methods is purely due to the unique interactions. In Table 10.3 we also show the AUC and AUPR split up for just the unique and non-unique interactions. With the unbiased baseline method, the AUC for unique interactions is barely above random chance level, while the biased baseline method achieves a perfect AUC. The overall AUC is a weighted average of the AUCs for unique and non-unique interactions, where the weight corresponds to the fraction of unique interactions. For example, for the GPCR dataset, $79\% \cdot 0.863 + 21\% \cdot 1.000 = 0.891$. Such a relation does not hold for AUPR scores, but the overall picture is similar.

10.5 Avoiding the bias

It seems that the biased results stem from the use of LOOCV. And so one would hope to avoid this problem by using 10 fold CV instead. As the right part of Figure 10.2 shows, this indeed reduces the bias, but it does not completely eliminate it.

We have repeated the experiment from the previous section with 10-fold CV instead of LOOCV. This is the setting used by, for instance Yamanishi, Araki,

Table 10.3: Performance of the unbiased baseline method and the biased variant when tested with LOOCV. The best results for each dataset are indicated in bold.

Dataset	AUC			AUPR		
Method	overall	unique	other	overall	unique	other
Enzyme						
Baseline	0.880	0.668	0.919	0.101	0.006	0.102
Biased	**0.931**	1.000	0.919	**0.301**	1.000	0.102
Ion Channel						
Baseline	0.850	0.528	0.874	0.244	0.003	0.254
Biased	**0.883**	1.000	0.874	**0.355**	1.000	0.254
GPCR						
Baseline	0.796	0.542	0.863	0.157	0.009	0.168
Biased	**0.891**	1.000	0.863	**0.420**	1.000	0.168
Nuclear Receptor						
Baseline	0.703	0.511	0.887	0.152	0.044	0.143
Biased	**0.942**	1.000	0.887	**0.682**	1.000	0.143

et al. (2008). As seen in the Table 10.4, exploiting the data bias still improves the AUC and AUPR scores for unique interactions, but this comes at the cost of the performance for non-unique interactions. In general, with k-fold cross-validation on a dataset with n drugs/targets, for each unique interacting pair, there are on the order of n/k non-interacting pairs that will be excluded in the same fold. These pairs will appear similar to the unique interaction ones. As the dataset becomes larger, there will be more such pairs.

However, it is still possible to beat the baseline method by making a trade-off between the increased performance on unique interactions and decreased performance on other interactions. For example, one can introduce the 'slight bias' method that ranks pairs which appear to be unique as if they have k adjacent interactions. So it ranks pairs by $a_{dt}^{\text{unique}\mapsto k}$ for some $k < \infty$. By tuning this parameter k we can tune the trade-off. In our experiments we chose k with cross validation. As shown in Table 10.4, this method achieves best AUC and AUPR on all but the smallest dataset; and in all cases shows a significant improvement over the baseline method.

So far we have considered the bias in the pairs setting. Results suggest that perhaps this validation setting should not be used. An alternative is the unseen drug setting, where one or more rows are left out in their entirety from the drug–target interaction matrix. This means that it becomes impossible to see if a

Table 10.4: Performance of the unbiased baseline method and the biased variants when tested with 10 fold CV. The best results for each dataset are indicated in bold, results in italic do not differ significantly from the best (at $\alpha = 0.05$).

Dataset	AUC			AUPR		
Method	overall	unique	other	overall	unique	other
Enzyme						
Baseline	0.879	0.669	0.917	0.098	0.006	0.099
Slight bias	**0.900**	0.818	0.915	**0.101**	0.012	0.097
Biased	0.862	0.982	0.840	0.056	0.135	0.027
Ion Channel						
Baseline	0.849	0.530	0.873	0.246	0.003	0.254
Slight bias	**0.859**	0.695	0.871	**0.248**	0.005	0.252
Biased	0.836	0.987	0.824	0.123	0.128	0.098
GPCR						
Baseline	0.795	0.543	0.859	0.154	0.009	0.163
Slight bias	**0.841**	0.801	0.853	**0.168**	0.025	0.155
Biased	0.827	0.975	0.788	0.116	0.180	0.057
Nuclear Receptor						
Baseline	0.697	0.533	0.885	0.154	0.047	0.155
Slight bias	0.857	0.884	0.846	0.247	0.177	0.124
Biased	**0.878**	0.967	0.781	**0.351**	0.473	0.070

pair is unique for a certain drug. But there are still interactions that are unique for a target. As shown in Table 10.5, this bias can still be exploited for improving CV performance, even when using 5- or 10-fold cross-validation.

Another option is to separate the unique interactions from the non-unique interactions when doing validation. As shown in our experiments, the non-unique interactions are not sensitive to the same bias. A good solution would be to only consider the AUC and AUPR scores for the non-unique interactions when comparing different methods. This still introduces a bias of a different kind, however, since some drug compounds and targets will be unnecessarily excluded.

A different way to validate a method is to seek confirmation of the predictions in other datasets. This is done by for instance Yamanishi, Kotera, *et al.* (2010), van Laarhoven, Nabuurs, and Marchiori (2011), and Gönen (2012), where the 10 highest rank predictions are looked up in the literature, and in newer versions of the KEGG BRITE, DrugBank Chembl, SuperTarget and Mata-

Table 10.5: Performance of the baseline method and biased variants in the unseen drug setting, when validated with 5-fold CV. The best results for each dataset are indicated in bold, results in italic do not differ significantly from the best (at $\alpha = 0.05$).

Dataset	AUC			AUPR		
Method	overall	unique	other	overall	unique	other
Enzyme						
Baseline	0.723	0.320	0.802	0.040	0.003	0.039
Slight bias	**0.772**	0.637	0.814	**0.041**	0.003	0.040
Biased	0.747	0.868	0.743	0.023	0.018	0.016
Ion Channel						
Baseline	0.699	0.602	0.710	*0.079*	0.010	0.075
Slight bias	**0.701**	0.677	0.707	**0.080**	0.010	0.075
Biased	*0.698*	0.797	0.694	0.064	0.017	0.059
GPCR						
Baseline	0.766	0.562	0.819	0.094	0.012	0.088
Slight bias	**0.782**	0.664	0.813	**0.095**	0.013	0.087
Biased	0.750	0.747	0.747	0.062	0.025	0.047
Nuclear Receptor						
Baseline	0.616	0.585	0.650	*0.140*	0.067	0.109
Slight bias	*0.647*	0.633	0.653	**0.144**	0.070	0.109
Biased	**0.670**	0.699	0.626	*0.126*	0.084	0.059

dor databases. A problem with such validation is that it is hard to quantify the performance, because only a few interactions are verified, and because these databases are extended between the publication of different papers.

Perhaps the most principled way of avoiding biases in validation is to act on the data and construct more realistic datasets. For this problem, that means that the dataset should also include compounds that interact with none of the targets, or targets for which there is no known interacting compound. The question then becomes which other drug compounds and proteins to include in the dataset. This possibility remains to be explored.

10.6 Conclusions

We have shown that popular benchmark data for the drug–target interaction problem are biased because they include only drug compounds and target pro-

teins with at least one interaction. This bias can be quantified by looking at the CV performance on these unique interactions separately from non-unique interactions. The bias is the largest with leave-one-out cross-validation in the pairs setting. But even with 5- or 10-fold cross-validation and in the unseen drugs setting there is still a significant bias. Our analysis indicates that results of CV procedures to assess the predictive performance of methods for drug–target interaction networks should be interpreted with care because they could be possibly positively affected by bias contained in the considered datasets.

The baseline method discussed in this chapter does not use the similarity information of drug compounds or target proteins at all. Hence, the performance is far below the state of the art. However, the effects of the bias carry over to other methods. For any ranking method r_{dt} we can define a variant $r_{dt}^{\text{unique}\mapsto k}$ that exploits the dataset bias and thereby boosts the performance on unique interacting pairs.

We have not performed an empirical study of the prevalence of biases in published methods. Of course none of the methods in Table 10.2 exploit the bias in quite such a blatant way as our 'biased baseline' method. Still, there could be methods that inadvertently take more advantage of the bias than others, for example in the choice of parameter values or in the way they handle specific types of drug–target pairs.

In this work we have focused on a single group of datasets, with a specific type of interaction, drug–target interaction. It remains to be investigated whether other datasets for the drug–target interaction prediction problem and datasets for other similar problems have the same bias. It would also be interesting to consider other interaction datasets, such as the drug–target, enzyme–motabolite and protein–ligand datasets from (Keiser, Roth, *et al.*, 2007; Campillos *et al.*, 2008; Faulon *et al.*, 2008; Jacob and J.-P. Vert, 2008; Keiser, Setola, *et al.*, 2009).

Chapter 11

Discussion

In this chapter we will look back at questions that were asked at the beginning of this thesis, and at possible new questions that arise.

11.1 So, what is clustering?

In the first part of this thesis I asked the question: "what is network clustering?". In chapters 3 and 6 I have made an attempt at answering that question, by giving properties that clustering methods should satisfy. These properties form a useful set of axioms for reasoning about clustering.

We know that the set of axioms is sound, that is, there is a quality function that satisfies all of them. We may wonder if the set of axioms is also complete, i.e. if it completely fixes the quality function or clustering method. If that were the case we would have completely defined what clustering is. Though I have no proof, I believe this not to be the case.

The only work I am aware of that gives a complete characterization of a form of clustering is (Zadeh and Ben-David, 2009), where the authors give axioms for single linkage hierarchical clustering. But a large risk of trying to completely characterize a certain clustering method is that the axiomatization becomes trivial. That is, the axioms are essentially saying "clusters should be the clusters found by method X".

As said above, the axioms allow us to reason about clustering. But we do not yet have a useful calculus for clustering, nor do we know what exactly it would look like. One area where these axioms are useful is in the design of algorithms. For example, locality means that a change to one part of a clustering doesn't affect the optimal clustering on unrelated parts of the network. This allows the optimization to be parallelized, with different machines solving clustering

optimization problems on parts of the network. Another property useful for optimization, which was not mentioned in Chapter 3, is the ability to zoom out. That is, to combine multiple nodes into a larger 'group' node. This property is used by the Louvain optimization algorithm.

In Chapter 6 we saw how locality can also be defined for soft clustering. But besides locality, Chapter 3 contains another important axiom for hard clustering, monotonicity. It is not yet clear what monotonicity would mean for soft and overlapping clusters. Edges are then no longer completely inside or outside a cluster, rather they are in some clusters to some degree. In fact, with Poisson likelihood NMF clustering, every edge will be inside some cluster, otherwise the quality is $-\infty$.

11.1.1 Robustness of clustering

Real data will be noisy. We would like this noise to not affect results too much. This is an important motivation of several of the axioms for network clustering.

None of the axioms directly guarantees that the clustering is robust to noise, and this is impossible to guarantee in general. After all, at some point a tiny change in the graph must lead to a change in the discrete clustering. But the axioms do allow us to make some limited statements about robustness. With a method that satisfies locality, noise in one part of the graph will not affect the clustering in another part of the graph. And with monotonicity additive noise inside clusters will not affect the quality of the clustering.

In our experiments on protein–protein interaction data we have shown that some clustering methods are much more robust to noise than others. There are ways to improve the robustness of a clustering method, which we did not investigate there. A promising idea is to use ensemble methods, where several different near optimal clusterings are combined. How to combine these clusterings is still an open problem, though.

11.1.2 The relevance of locality

A motivation for the locality axiom is the fact the observed network may just be a subsample of all objects and relations in the real world. But there are actually two ways in which a graph with a clustering structure may be subsampled.

First of all, we may select nodes and their edges independently. Suppose, for example, that we randomly select 0.1% of all nodes. Then with high confidence, we can assume that the sample actually has nodes from all clusters, assuming that they are not too small. More concretely, when selecting 0.1% of the people on earth, you can be confident that you get someone from every country (except

perhaps Vatican city). In this setting, locality is not a relevant property of the clustering method, and we may even specify the number of clusters in advance.

On the other hand, what if sampling depends on the cluster or relations? You may sample the network of co-authorships, starting with me, my coauthors and so on. Then it becomes much more likely that the sample completely misses parts of the network, say those concerning English literature. In this setting locality becomes important if we want to draw conclusions of the entire network based on just the sample.

In practice the situation is of course somewhere in between. We don't really know how the protein interaction network was sampled. There might be a focus on disease related proteins, or on proteins that are easily isolated in the lab. In this case requiring locality is the safe choice.

11.2 Data and validation

Data is always an important issue when doing machine learning. No matter how good your methods are, you can't make good inferences with bad data.

There are large network datasets with millions of nodes, to which network clustering can be applied. But for these datasets only the nodes and edges are known. We don't know the cluster structure, or even if the data has a cluster structure at all. So the question becomes how to know if the clusters found by a method are actually sensible, and how to compare different methods. For this reason we have used smaller networks where a ground truth clustering is known, together with synthetic data.

With the drug–target network used in the second part of this thesis we can at least use cross-validation to validate our predictions. But even this cross-validation is still problematic in several ways. First of all, only the positives are known, so all validation is based on the assumption that if no interaction is observed then no interaction exists. Secondly, in Chapter 10 we saw that the drug–target interaction network only includes drugs and targets with at least one known interaction.

Note that we know of these two problems because of meta information. That is, we know how the dataset was constructed. Without this meta information we might still be able to detect the second problem based on the degree distribution of the network, but the first problem is not detectable.

A third problem with the drug–target dataset is that it is unweighted and unlabeled: edges only indicate that a drug and target interact, not the strength of the interaction, or what kind of interaction it is. Having this extra information might make better predictions possible.

Still, despite these problems, it is possible to predict new interactions. An important ingredient in these successful predictions are the kernels for drug compounds and target proteins. Besides augmenting them with information from the network, we have not looked at the kernels at all. This is an area where improvements might be possible, but it would require more domain specific knowledge, and likely methods very different from the research in this thesis.

What if your data does not take the form of a network? One option is to use a k nearest neighbor graph of a dataset. These kNN graphs have a special structure. They are directed graphs, where each node has out-degree k. It might be possible to take advantage of this structure in a clustering algorithm. For instance, normalization of edge weights by node degree might not be needed if all nodes have a similar degree.

It is also not clear if the kNN graph is the only way of converting a general dataset to a graph. Nor do we know what parameters to use, what value of k, how to weight the edges, and how to make the graph symmetric. If the graph is later used as input for a certain clustering algorithm, it might be possible to determine what parameters should then be used to generate the graph.

The interaction profile kernel works in the other direction. It can be used to apply kernel methods to network data. Again, this kernel has a special structure, and it might be possible to harness it to get better results.

11.3 Algorithms

A large part of this thesis is focused on theory instead of algorithms, and so several algorithmic questions remain unanswered. The most important missing piece is a general method for soft clustering. There are methods for non-negative matrix factorization, but these only work for a small set of quality functions. In addition, their running time is linear in the number of clusters, which can be a problem in large graphs. The experiments in Chapter 6 were performed with a general purpose optimization tool. This works fine for toy problems, but doesn't scale up to larger networks. Compare this to the situation for hard clustering. In Chapter 4 we used the Louvain method, which can efficiently find a good (hard) clustering of very large networks. It would be very useful if this method can be extended to soft clustering.

Table of notations

Notations used in Part 1.

Symbol	Page	Meaning
A	16	Edge weight function or adjacency matrix of a graph.
a_{ij}	16	$A(i,j)$; The weight of the edge between i and j, or 0 if there is no edge.
$\hat{a}_{ij}$	77	Predicted cluster membership in NMF: an element of the product of membership matrices.
C	16	A clustering: a set of clusters.
$\mathcal{C}(s)$	76	The set of all clusters with support equal to s.
$\mathcal{C}^*(s)$	76	The set of all clusters with support a subset of s.
G	16	A graph: a tuple (V, A).
$h(x)$	42	The 'entropy' function: $h(x) = -x \log x$.
h_{ci}	77	The membership coefficient for node i in cluster c.
$\mathcal{HN}(\sigma)$	87	The half-normal distribution with parameter σ.
$\mathcal{N}(\mu, \sigma)$	87	The normal distribution with mean μ and standard deviation σ.
Q	17	A quality function: a function from graphs and clusterings to real numbers.
$\mathbb{R}_{\geqslant 0}$	16	The set of non-negative real numbers.
$\mathbb{R}_{>0}$	77	The set of positive real numbers.
$s_i(G)$	17	The strength of node i in graph G.
$\operatorname{supp}(c)$	76	Support of a cluster: the set of nodes with non-zero membership, or equivalently the set s for which $c \in \mathcal{C}(s)$.
V	16	The set of vertices in a graph.
$v_c(G)$	17	The volume of cluster or set of nodes c in graph G.
$\widehat{v}_c(G)$	40	Normalized volume of cluster c: $v_c(G)/v_V(G)$.

Symbol	Page	Meaning
$w_c(G)$	17	The within cluster weight of cluster c in graph G.
$\widehat{w}_c(G)$	40	Normalized within cluster weight of cluster c: $w_c(G)/\nu_V(G)$.
w_{ci}	77	The second membership coefficients next to h_{ci} in asymmetric NMF.
β_c	77	A per cluster hyper parameter in the method of Psorakis *et al.*
λ	89	Parameter of the Poisson prior on the number of clusters per node.
μ	47	Mixing parameter for the LFR graph model: fraction of edges that are between clusters.
$\mathbf{1}[p]$	21	The indicator function: 1 if p is holds, 0 otherwise.
$i \sim_C j$	16	Nodes i and j are in the same cluster in clustering C.
$C \triangle D$	76	Symmetric difference of two clusterings: the set of clusters in exactly one of C and D.
$C \uplus D$	76	The sum of multisets, the multiplicity of c in $C \uplus D$ is the sum of the multiplicities of c in C and of c in D.
$C \sqsubseteq D$	17	Refinement of clusterings.

Notations used in Part 2.

Symbol	Page	Meaning
a_{dt}	140	The number of adjacent interacting pairs for the pair (d, t).
D	102	Set of all drugs compounds under consideration.
$K_{\text{chemical,d}}$	101	Kernel for drugs based on chemical similarity.
$K_{\text{genomic,t}}$	101	Kernel for proteins based on sequence similarity.
$K_{\text{GIP,d}}$, $K_{\text{GIP,t}}$	107	Kernels for drugs and targets based on Gaussian Interaction Profiles.
K_d	108	Combined kernel for drugs compounds.
K_t	108	Combined kernel for target proteins.
n_d	102	The number of drug compounds: $n_d = \lvert D \rvert$.
n_t	102	The number of target proteins: $n_t = \lvert T \rvert$.
T	102	Set of all target proteins under consideration.
$\text{vec}(M)$	110	View a matrix as a vector by stacking its columns.
y_{ij}	102	Known interaction between drug d_i and target t_i, 1 if they interact, 0 if no interaction is known.

Symbol	Page	Meaning
$\hat{Y}$	109	Predicted interaction scores.
y_{d_i}	107	Interaction Profile of drug i: the ith row of Y.
$y_{t_j}^T$	107	Interaction Profile of target j: the jth column of Y.
α	108	Weight for combining different drug (or target) kernels
β	108	Bandwidth of the GIP kernel.
$\tilde{\beta}$	108	Size independent bandwidth parameter of the GIP kernel.
$A \otimes B$	109	Kronecker product of two matrices.

Bibliography

M. Ackerman, S. Ben-David, D. Loker, and S. Sabato (2013). "Clustering Oligarchies". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 31. JMLR Workshop and Conference Proceedings, pp. 66–74 (cited on page 15).

M. Ackerman and S. Ben-David (2008). "Measures of Clustering Quality: A Working Set of Axioms for Clustering". In: *NIPS*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Curran Associates, Inc., pp. 121–128 (cited on pages 13–15, 17–19, 74).

M. Ackerman, S. Ben-David, S. Brânzei, and D. Loker (2012). "Weighted Clustering". In: *AAAI*. Ed. by J. Hoffmann and B. Selman. AAAI Press (cited on page 15).

M. Ackerman, S. Ben-David, and D. Loker (2010a). "Characterization of Linkage-based Clustering". In: *COLT 2010: The 23rd Conference on Learning Theory*, pp. 270–281 (cited on pages 13, 20, 75, 82, 84).

M. Ackerman, S. Ben-David, and D. Loker (2010b). "Towards Property-Based Classification of Clustering Paradigms". In: *NIPS*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Curran Associates, Inc., pp. 10–18 (cited on page 13).

L. Adamic and N. Glance (2005). "The Political Blogosphere and the 2004 U.S. Election: Divided They Blog". In: *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43 (cited on page 54).

H. Almeida, D. Guedes, W. M. Jr., and M. J. Zaki (2011). "Is There a Best Quality Metric for Graph Clusters?" In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis. Vol. 6911. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 44–59. ISBN: 978-3-642-23779-9 (cited on page 30).

L. Angelini, D. Marinazzo, M. Pellicoro, and S. Stramaglia (2007). "Natural clustering: the modularity approach". In: *Journal of Statistical Mechanics: Theory and Experiment* 2007.08, p. L08001 (cited on page 45).

A. Arenas, A. Fernández, and S. Gómez (2008). "Analysis of the structure of complex networks at different resolution levels". In: *New Journal of Physics* 10.5, p. 053039 (cited on page 38).

J. Basilico and T. Hofmann (2004). "Unifying collaborative and content-based filtering". In: *ICML '04: Proceedings of the 21st International Conference on Machine learning*. ACM, pp. 65–72 (cited on pages 108, 109).

K. Baumann and S. Rohrer (2008). "Exploring benchmark dataset bias in ligand based virtual screening". In: *Chemistry Central Journal* 2.Suppl 1, P1. ISSN: 1752-153X (cited on page 137).

A. Ben-Hur and W. S. Noble (2005). "Kernel methods for predicting protein–protein interactions". In: *Bioinformatics* 21.suppl. 1, pp. i38–i46 (cited on page 109).

K. Bleakley and Y. Yamanishi (2009). "Supervised prediction of drug-target interactions using bipartite local models." In: *Bioinformatics* 25.18, pp. 2397–2403 (cited on pages 100, 106, 109, 110, 114–116, 124, 136, 139).

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre (2008). "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10, P10008. ISSN: 1742-5468 (cited on pages 14, 39, 46, 75).

B. Bollobás (2001). "The Evolution of Random Graphs – the Giant Component". In: Cambridge University Press, pp. 130–159. ISBN: 9780521797221 (cited on page 22).

U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner (2008). "On Modularity Clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 20.2, pp. 172–188. ISSN: 1041-4347 (cited on pages 8, 30, 38).

M. Brito, E. Chávez, A. Quiroz, and J. Yukich (1997). "Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection". In: *Statistics & Probability Letters* 35.1, pp. 33–42. ISSN: 0167-7152 (cited on pages 7, 9).

S. Brohee and J. van Helden (2006). "Evaluation of clustering algorithms for protein-protein interaction networks". In: *BMC Bioinformatics* 7.1, p. 488 (cited on pages 61, 64, 65).

S. Bubeck and U. von Luxburg (2009). "Nearest Neighbor Clustering: A Baseline Method for Consistent Clustering with Arbitrary Objective Functions". In: *Journal of Machine Learning Research* 10, pp. 657–698. ISSN: 1532-4435 (cited on page 15).

T. Bühler and M. Hein (2009). "Spectral clustering based on the graph p-Laplacian". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: ACM, pp. 81–88. ISBN: 978-1-60558-516-1 (cited on page 54).

M. Campillos, M. Kuhn, A.-C. Gavin, L. J. Jensen, and P. Bork (2008). "Drug target identification using side-effect similarity". In: *Science* 321.5886, pp. 263–266 (cited on pages 100, 145).

G. Carlsson, F. Mémoli, A. Ribeiro, and S. Segarra (2013). "Axiomatic Construction of Hierarchical Clustering in Asymmetric Networks". In: *ICASSP 2013: IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5219–5223 (cited on pages 13, 15).

M. Catral, L. Han, M. Neumann, and R. J. Plemmons (2004). "On reduced rank nonnegative matrix factorizations for symmetric matrices". In: *Linear Algebra and its Applications* 393, pp. 107–126 (cited on page 77).

A. T. Cemgil (2009). "Bayesian Inference for Nonnegative Matrix Factorisation Models". In: *Computational Intelligence and Neuroscience* 2009. ISSN: 1687-5265 (cited on page 87).

X. Chen, M.-X. Liu, and G.-Y. Yan (2012). "Drug-target interaction prediction by random walk on the heterogeneous network". In: *Mol Biosyst.* 8.7, pp. 1970–1978 (cited on pages 136, 139).

A. C. Cheng, R. G. Coleman, K. T. Smyth, Q. Cao, P. Soulard, D. R. Caffrey, A. C. Salzberg, and E. S. Huang (2007). "Structure-based maximal affinity model predicts small-molecule druggability". In: *Nature Biotechnology* 25.1, pp. 71–75 (cited on page 100).

J. M. Cherry *et al.* (2011). "Saccharomyces Genome Database: the genomics resource of budding yeast". In: *Nucleic Acids Research* (cited on page 64).

H. N. Chua, K. Ning, W. K. Sung, H. W. Leong, and L. Wong (2008). "Using indirect protein-protein interactions for protein complex prediction". In: *Journal of Bioinformatics and Computational Biology* 6.3, pp. 435–466. ISSN: 0219-7200 (cited on page 65).

A. Clauset, C. R. Shalizi, and M. E. J. Newman (2009). "Power-Law Distributions in Empirical Data". In: *SIAM Rev.* 51 (4), pp. 661–703. ISSN: 0036-1445 (cited on page 47).

S. Collins, P. Kemmeren, X.-C. Zhao, J. Greenblatt, F. Spencer, F. Holstege, J. Weissman, and N. Krogan (2007). "Towards a comprehensive atlas of the physical interactome of Saccharomyces cerevisiae". In: *Molecular Cellular Proteomics*, pp. 600381–600200 (cited on pages 63, 64).

L. Danon, J. Duch, A. Arenas, and A. Díaz-Guilera (2005). "Comparing community structure identification". In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.09, P09008 (cited on page 49).

J. Davis and M. Goadrich (2006). "The relationship between Precision-Recall and ROC curves". In: *ICML '06: Proceedings of the 23rd International Conference on Machine learning*. ACM, pp. 233–240 (cited on pages 106, 114, 124, 127, 139).

E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson (1988). "Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach". In: *Biometrics* 44.3, pp. 837–845. ISSN: 0006-341X (cited on pages 127, 141).

C. Ding, X. He, and H. D. Simon (2005). "On the equivalence of nonnegative matrix factorization and spectral clustering". In: *SIAM International Conference on Data Mining* (cited on pages 78–80).

C. Ding, T. Li, W. Peng, and H. Park (2006). "Orthogonal Nonnegative Matrix T-factorizations for Clustering". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. Philadelphia, PA, USA: ACM, pp. 126–135. ISBN: 1-59593-339-5 (cited on page 79).

T. N. Dinh and M. T. Thai (2013). "Community Detection in Scale-Free Networks: Approximation Algorithms for Maximizing Modularity". In: *IEEE Journal on Selected Areas in Communications* 31.6, pp. 997–1006 (cited on page 30).

A. M. Edwards, B. Kus, R. Jansen, D. Greenbaum, J. Greenblatt, and M. Gerstein (2002). "Bridging structural biology and genomics: assessing protein interaction data with known complexes". In: *Trends in Genetics* 18.10, pp. 529–536. ISSN: 01689525 (cited on page 70).

J.-L. Faulon, M. Misra, S. Martin, K. Sale, and R. Sapra (2008). "Genome scale enzyme–metabolite and drug–target interaction predictions using the sig-

nature molecular descriptor". In: *Bioinformatics* 24.2, pp. 225–233 (cited on pages 99, 145).

T. Fawcett (2006). "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8, pp. 861–874 (cited on pages 113, 127, 139).

S. Fortunato (2010). "Community detection in graphs". In: *Physics Reports* 486, pp. 75–174 (cited on pages 8, 38, 42, 60, 73).

S. Fortunato and M. Barthélemy (2007). "Resolution limit in community detection". In: *Proceedings of the National Academy of Sciences of the United States of America* 104.1, pp. 36–41 (cited on pages 9, 15, 20, 26, 27, 38, 42, 43, 63, 74, 75, 86, 90).

P. Franti, O. Virmajoki, and V. Hautamaki (2006). "Fast Agglomerative Clustering Using a k-Nearest Neighbor Graph". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 28.11, pp. 1875–1881. ISSN: 0162-8828 (cited on pages 7, 9).

A.-C. Gavin, P. Aloy, *et al.* (2006). "Proteome survey reveals modularity of the yeast cell machinery". In: *Nature* 440.7084, pp. 631–636 (cited on pages 63, 64, 67).

A.-C. Gavin, M. Bosche, *et al.* (2002). "Functional organization of the yeast proteome by systematic analysis of protein complexes." In: *Nature* 415.6868, pp. 141–7 (cited on pages 63, 64, 67).

M. Girvan and M. E. J. Newman (2002). "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 99.12, pp. 7821–7826 (cited on pages 40, 53, 61, 62).

S. Gollapudi and A. Sharma (2009). "An axiomatic approach for result diversification". In: *Proceedings of the 18th international conference on World wide web*, pp. 381–390 (cited on page 14).

M. Gönen (2012). "Predicting drug-target interactions from chemical and genomic kernels using Bayesian matrix factorization". In: *Bioinformatics* 28.18, pp. 2304–2310 (cited on pages 124, 127–129, 136, 139, 143).

B. H. Good, Y. A. de Montjoye, and A. Clauset (2010). "Performance of modularity maximization in practical contexts". In: *Physical Review E* 81.4, p. 046106 (cited on pages 23, 38).

C. Granell, S. Gómez, and A. Arenas (2012). "Unsupervised Clustering Analysis: a Multiscale Complex Networks Approach". In: *Internat. J. Bifur. Chaos Appl. Sci. Engrg.* 22.7 (cited on page 55).

S. Günther *et al.* (2008). "SuperTarget and Matador: resources for exploring drug-target relationships." In: *Nucleic Acids Research* 36.Database issue, pp. D919–D922 (cited on pages 100, 101).

S. J. Haggarty, K. M. Koeller, J. C. Wong, R. A. Butcher, and S. L. Schreiber (2003). "Multidimensional Chemical Genetic Analysis of Diversity-Oriented Synthesis-Derived Deacetylase Inhibitors Using Cell-Based Assays". In: *Chemistry & Biology* 10.5, pp. 383–396 (cited on page 99).

M. Hattori, Y. Okuno, S. Goto, and M. Kanehisa (2003). "Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways". In: *Journal of the American Chemical Society* 125.39, pp. 11853–65 (cited on page 101).

Y. Ho *et al.* (2002). "Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry." In: *Nature* 415.6868, pp. 180–183 (cited on pages 63, 64).

A. L. Hopkins and C. R. Groom (2002). "The druggable genome." In: *Nature reviews. Drug discovery* 1.9, pp. 727–730 (cited on page 99).

M. Hue and J.-P. Vert (2010). "On learning with kernels for unordered pairs". In: *ICML '10: Proceedings of the 27th International Conference on Machine Learning*. Ed. by J. Fürnkranz and T. Joachims. Haifa, Israel: Omnipress, pp. 463–470 (cited on page 109).

A. Isaksson, M. Wallman, H. Göransson, and M. G. Gustafsson (2008). "Cross-validation and bootstrapping are unreliable in small sample classification". In: *Pattern Recognition Letters* 29.14, pp. 1960–1965 (cited on page 137).

L. Jacob, B. Hoffmann, B. Stoven, and J.-P. Vert (2008). "Virtual screening of GPCRs: an *in silico* chemogenomics approach". In: *BMC Bioinformatics* 9, p. 363 (cited on page 100).

L. Jacob and J.-P. Vert (2008). "Protein-ligand interaction prediction: an improved chemogenomics approach". In: *Bioinformatics* 24.19, pp. 2149–2156 (cited on pages 99, 145).

S. E. Jaroch and H. Weinmann, *eds.* (2006). *Chemical Genomics: Small Molecule Probes to Study Cellular Function*. Ed. by S. E. Jaroch and H. Weinmann. Ernst Schering Research Foundation Workshop. Berlin: Springer (cited on page 100).

M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa (2006). "From genomics to chemical genomics: new developments in KEGG." In: *Nucleic Acids Research* 34.Database issue, pp. D354–357 (cited on pages 100, 101, 116).

H. Kashima, T. Idé, T. Kato, and M. Sugiyama (2009). "Recent Advances and Trends in Large-Scale Kernel Methods". In: *IEICE Transactions* 92-D.7, pp. 1338–1353 (cited on pages 110, 133).

H. Kashima, S. Oyama, Y. Yamanishi, and K. Tsuda (2009). "On Pairwise Kernels: An Efficient Alternative and Generalization Analysis". In: *PAKDD '09: Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pp. 1030–1037 (cited on page 109).

M. J. Keiser, B. L. Roth, B. N. Armbruster, P. Ernsberger, J. J. Irwin, and B. K. Shoichet (2007). "Relating protein pharmacology by ligand chemistry". In: *Nat. Biotechnol.* 25.2, pp. 197–206 (cited on page 145).

M. J. Keiser, V. Setola, *et al.* (2009). "Predicting new molecular targets for known drugs". In: *Nature* 462.7270, pp. 175–181 (cited on page 145).

B. W. Kernighan and S. Lin (1970). "An Efficient Heuristic Procedure for Partitioning Graphs". In: *Bell System Technical Journal* 49.1, pp. 291–307 (cited on page 49).

T. Klabunde (2007). "Chemogenomic approaches to drug discovery: similar receptors bind similar ligands." In: *British Journal of Pharmacology* 152, pp. 5–7 (cited on page 100).

J. M. Kleinberg (2002). "An Impossibility Theorem for Clustering". In: *NIPS*. Ed. by S. Becker, S. Thrun, and K. Obermayer. MIT Press, pp. 446–453. ISBN: 0-262-02550-7 (cited on pages 13–15, 17, 18).

R. Kohavi (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., pp. 1137–1143. ISBN: 1-55860-363-8 (cited on page 136).

V. Krebs (2004). *New Political Patterns*. Editorial (cited on page 53).

N. J. Krogan *et al.* (2006). "Global landscape of protein complexes in the yeast Saccharomyces cerevisiae". In: *Nature* 440.7084, pp. 637–643 (cited on pages 63, 64).

J. M. Kumpula, J. Saramaki, K. Kaski, and J. Kertesz (2007). "Limited resolution in complex network community detection with Potts model approach". In: *The European Physical Journal B* 56, p. 41 (cited on pages 42, 43).

T. van Laarhoven and E. Marchiori (2012). "Robust community detection methods with resolution parameter for complex detection in protein protein in-

teraction networks". In: *Proceedings of the 7th IAPR international conference on Pattern Recognition in Bioinformatics*. PRIB'12. Berlin, Heidelberg: Springer-Verlag, pp. 1–13. ISBN: 978-3-642-34122-9 (cited on page 59).

T. van Laarhoven and E. Marchiori (2013a). "Graph clustering with local search optimization: The resolution bias of the objective function matters most". In: *Physical Review E* 87 (1), p. 012812 (cited on page 37).

T. van Laarhoven and E. Marchiori (2013b). "Predicting Drug-Target Interactions for New Drug Compounds Using a Weighted Nearest Neighbor Profile". In: *PLoS ONE* 8.6, e66952 (cited on pages 123, 136, 139).

T. van Laarhoven and E. Marchiori (2014a). "Axioms for Graph Clustering Quality Functions". In: *Journal of Machine Learning Research* 15, pp. 193–215 (cited on page 13).

T. van Laarhoven and E. Marchiori (2014b). "Biases of drug–target interaction network data". In: *Proceedings of the 9th IAPR international conference on Pattern Recognition in Bioinformatics*. PRIB'14. Springer-Verlag. ISBN: 978-3-319-09191-4 (cited on page 135).

T. van Laarhoven, S. B. Nabuurs, and E. Marchiori (2011). "Gaussian interaction profile kernels for predicting drug–target interaction". In: *Bioinformatics* 27.21, pp. 3036–3043 (cited on pages 105, 136, 139, 143).

R. Lambiotte (2010). "Multi-scale modularity in complex networks". In: *Proceedings of the 8th Int. Symp. on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pp. 546–553 (cited on pages 38, 45, 46, 60).

A. Lancichinetti and S. Fortunato (2009). "Community detection algorithms: A comparative analysis". In: *Physical Review E* 80 (5), p. 056117 (cited on pages 39, 48).

A. Lancichinetti, S. Fortunato, and F. Radicchi (2008). "Benchmark graphs for testing community detection algorithms". In: *Physical Review E* 78 (4), p. 046110 (cited on page 47).

A. Lancichinetti and S. Fortunato (2011). "Limits of modularity maximization in community detection". In: *Physical Review E* 84, p. 066122 (cited on pages 28, 38, 45).

D. D. Lee and H. S. Seung (1999). "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755, pp. 788–791. ISSN: 0028-0836 (cited on pages 74, 77).

D. D. Lee and H. S. Seung (2000). "Algorithms for Non-negative Matrix Factorization". In: *14th Annual Conference on Neural Information Processing Systems*. NIPS 2000. MIT Press, pp. 556–562 (cited on page 90).

A. Lewis, N. Jones, M. Porter, and C. Deane (2010). "The function of communities in protein interaction networks at multiple scales". In: *BMC Syst Biol.* 4:100 (cited on pages 60, 61).

T. Li and C. H. Q. Ding (2013). "Nonnegative Matrix Factorizations for Clustering: A Survey". In: *Data Clustering: Algorithms and Applications*. Chapman and Hall/CRC, pp. 149–176. ISBN: 9781466558212 (cited on page 74).

X. Li, M. Wu, C. K. Kwoh, and S. K. Ng (2010). "Computational approaches for detecting protein complexes from protein interaction networks: a survey". In: *BMC Genomics* 11.Suppl 1, S3 (cited on pages 60, 61).

L. Lü and T. Zhou (2011). "Link prediction in complex networks: A survey". In: *Physica A: Statistical Mechanics and its Applications* 390.6, pp. 1150–1170. ISSN: 0378-4371 (cited on page 100).

U. von Luxburg (2007). "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4, pp. 395–416. ISSN: 0960-3174 (cited on pages 8, 38).

Y. C. Martin, J. L. Kofron, and L. M. Traphagen (2002). "Do structurally similar molecules have similar biological activity?" In: *Journal of Medicinal Chemistry* 45.19, pp. 4350–4358 (cited on page 100).

J.-P. Mei, C.-K. Kwoh, P. Yang, X. Li, and J. Zheng (2013). "Drug-target interaction prediction by learning from local information and neighbors". In: *Bioinformatics* 29.2, pp. 238–245 (cited on pages 124, 126–128, 132, 136, 139).

M. Meila (2005). "Comparing clusterings: an axiomatic view". In: *Proceedings of the 22nd international conference on Machine learning*. ACM, pp. 577–584 (cited on page 13).

A. Merino, A. K. Bronowska, D. B. Jackson, and D. J. Cahill (2010). "Drug profiling: knowing where it hits". In: *Drug Discovery Today* 15.17-18, pp. 749–756. ISSN: 1359-6446 (cited on page 120).

J. T. Metz and P. J. Hajduk (2010). "Rational approaches to targeted polypharmacology: creating and navigating protein-ligand interaction networks". In: *Current Opinion in Chemical Biology* 14.4. Next Generation Therapeutics, pp. 498–504. ISSN: 1367-5931 (cited on page 120).

H. W. Mewes, D. Frishman, U. Gildener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Minsterötter, S. Rudd, and B. Weil (2002). "MIPS: a

database for genomes and protein sequences". In: *Nucleic Acids Res* 30, pp. 31–34 (cited on page 64).

T. Nepusz, H. Yu, and A. Paccanaro (2012). "Detecting overlapping protein complexes in protein-protein interaction networks". In: *Nature Methods* 9 (5), pp. 471–472. ISSN: 1548-7091 (cited on pages 61, 64).

M. E. J. Newman (2006). "Finding community structure in networks using the eigenvectors of matrices". In: *Physical Review E* 74 (3), p. 036104 (cited on pages 30, 38, 39, 49).

M. E. J. Newman and M. Girvan (2004). "Finding and evaluating community structure in networks". In: *Physical Review E* 69 (2), p. 026113 (cited on pages 14, 23, 74).

Y. Okuno, A. Tamon, H. Yabuuchi, S. Niijima, Y. Minowa, K. Tonomura, R. Kunimoto, and C. Feng (2008). "GLIDA: GPCR—ligand database for chemical genomics drug discovery—database and tools update". In: *Nucleic Acids Research* 36.suppl 1, pp. D907–D912 (cited on page 100).

J. Overington (2009). "ChEMBL. An interview with John Overington, team leader, chemogenomics at the European Bioinformatics Institute Outstation of the European Molecular Biology Laboratory (EMBL-EBI). Interview by Wendy A. Warr." In: *Journal of Computer-Aided Molecular Design* 23.4, pp. 195–198. ISSN: 0920-654X (cited on pages 100, 116).

S. Oyama and C. D. Manning (2004). "Using Feature Conjunctions Across Examples for Learning Pairwise Classifiers". In: *Proceedings of the 15th European Conference on Machine Learning*. Vol. 3201. ECML'04. Springer, pp. 322–333 (cited on page 109).

P. Paatero and U. Tapper (1994). "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values". In: *Environmetrics* 5.2, pp. 111–126 (cited on pages 74, 77).

G. Palla, I. Derenyi, I. Farkas, and T. Vicsek (2005). "Uncovering the overlapping community structure of complex networks in nature and society". In: *Nature* 435 (7043), pp. 814–818 (cited on page 83).

J. Pitman (2006). *Combinatorial Stochastic Processes*. Berlin: Springer-Verlag (cited on page 89).

I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon (2011). "Overlapping community detection using Bayesian non-negative matrix factorization". In: *Physical Review E* 83.6, pp. 066114+ (cited on pages 74, 77, 78, 85–87, 91, 152).

S. Pu, J. Vlasblom, A. Emili, J. Greenblatt, and S. J. Wodak (2007). "Identifying functional modules in the physical interactome of Saccharomyces cerevisiae." In: *Proteomics* 7.6, pp. 944–960 (cited on page 62).

J. Puzicha, T. Hofmann, and J. M. Buhmann (1999). "A Theory of Proximity Based Clustering: Structure Detection by Optimization". In: *Pattern Recognition* 33, pp. 617–634 (cited on pages 15, 18).

F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi (2004). "Defining and identifying communities in networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 101.9, pp. 2658–2663 (cited on page 92).

V. V. Raghavan, G. S. Jung, and P. Bollmann (1989). "A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance". In: *ACM Transactions on Information Systems* 7.3, pp. 205–229 (cited on page 113).

R. B. Rao and G. Fung (2008). "On the Dangers of Cross-Validation. An Experimental Evaluation". In: *SDM*. SIAM, pp. 588–596 (cited on page 137).

R. Raymond and H. Kashima (2010). "Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs". In: *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*. ECML PKDD'10. Barcelona, Spain: Springer-Verlag, pp. 131–147. ISBN: 3-642-15938-9, 978-3-642-15938-1 (cited on page 110).

J. Reichardt and S. Bornholdt (2004). "Detecting Fuzzy Community Structures in Complex Networks with a Potts Model". In: *Physical Review Letters* 93 (21), p. 218701 (cited on pages 15, 21, 26, 30, 38, 42, 45, 75).

J. Reichardt and S. Bornholdt (2006). "Statistical mechanics of community detection". In: *Physical Review E* 74 (1), p. 016110 (cited on page 16).

J. Reichardt and S. Bornholdt (2007). "Partitioning and modularity of graphs with arbitrary degree distribution". In: *Physical Review E* 76 (1), p. 015102 (cited on page 28).

R. Rifkin and A. Klautau (2004). "In defense of one-vs-all classification". In: *Journal of Machine Learning Research* 5, pp. 101–141 (cited on page 108).

P. Ronhovde and Z. Nussinov (2009). "Multiresolution community detection for megascale networks by information-based replica correlations". In: *Physical Review E* 80.1, pp. 016109+ (cited on page 60).

M. Rosvall and C. T. Bergstrom (2008). "Maps of random walks on complex networks reveal community structure." In: *Proceedings of the National Academy of*

Sciences of the United States of America 105.4, pp. 1118–1123 (cited on pages 39, 41, 58).

J. Ruan and W. Zhang (2008). "Identifying network communities with a high resolution". In: *Physical Review E* 77 (1), p. 016104 (cited on page 30).

S. E. Schaeffer (2007). "Graph clustering". In: *Computer Science Review* 1.1, pp. 27–64. ISSN: 15740137 (cited on pages 8, 38, 73).

B. Schölkopf, K. Tsuda, and J. P. Vert, *eds.* (2004). *Kernel Methods in Computational Biology*. Ed. by B. Schölkopf, K. Tsuda, and J. P. Vert. MIT Press. ISBN: 9780262195096s (cited on page 100).

I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg (2004). "BRENDA, the enzyme database: updates and major new developments". In: *Nucleic Acids Research* 32.suppl. 1, pp. D431–433 (cited on pages 100, 101).

A. Schuffenhauer, P. Floersheim, P. Acklin, and E. Jacoby (2003). "Similarity metrics for ligands reflecting the similarity of the target proteins". In: *Journal of Chemical Information and Computer Sciences* 43.2, pp. 391–405 (cited on page 100).

J. Shi and J. Malik (2000). "Normalized Cuts and Image Segmentation". In: vol. 22. 8. Washington, DC, USA: IEEE Computer Society, pp. 888–905 (cited on pages 26, 38–40, 49).

T. F. Smith and M. S. Waterman (1981). "Identification of common molecular subsequences." In: *Journal of Molecular Biology* 147.1, pp. 195–197 (cited on page 101).

C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers (2006). "BioGRID: a general repository for interaction datasets." In: *Nucleic Acids Research* 34.Database-Issue, pp. 535–539 (cited on pages 61, 63).

A. Torralba and A. A. Efros (2011). "Unbiased look at dataset bias". In: *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '11. Washington, DC, USA: IEEE Computer Society, pp. 1521–1528. ISBN: 978-1-4577-0394-2 (cited on page 137).

V. A. Traag, G. Krings, and P. V. Dooren (2013). "Significant Scales in Community Structure". In: *Scientific Reports* 3 (cited on page 30).

V. A. Traag, P. Van Dooren, and Y. E. Nesterov (2011). "Narrow scope for resolution-limit-free community detection". In: *Physical Review E* 84 (1), p. 016114 (cited on pages 16, 20, 21, 23, 26, 27, 29, 38, 74, 75, 78, 79).

P. Uetz *et al.* (2000). "A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae". In: *Nature* 403.6770, pp. 623–627 (cited on pages 63, 64).

V. Y. F. Tan and C. Févotte (2009). "Automatic relevance determination in non-negative matrix factorization". In: *Proc. Workshop on Signal Processing with Adaptative Sparse Structured Representations (SPARS)*. St-Malo, France (cited on page 85).

S. Van Dongen (2008). "Graph Clustering Via a Discrete Uncoupling Process". In: *SIAM Journal on Matrix Analysis and Applications* 30.1, pp. 121–141 (cited on page 61).

F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding (2011). "Community Discovery Using Nonnegative Matrix Factorization". In: *Data Min. Knowl. Discov.* 22.3, pp. 493–521. ISSN: 1384-5810 (cited on page 74).

A. M. Wassermann, H. Geppert, and J. Bajorath (2009). "Ligand prediction for orphan targets using support vector machines and various target-ligand kernels is dominated by nearest neighbor effects." eng. In: *Journal of Chemical Information and Modeling* 49, pp. 2155–2167 (cited on pages 100, 136).

D. S. Wishart, C. Knox, A. C. C. Guo, D. Cheng, S. Shrivastava, D. Tzur, B. Gautam, and M. Hassanali (2008). "DrugBank: a knowledgebase for drugs, drug actions and drug targets." In: *Nucleic Acids Research* 36.Database issue, pp. D901–906 (cited on pages 100, 101, 116).

G. Wu, E. Chang, Y. K. Chen, and C. Hughes (2006). "Incremental approximate matrix factorization for speeding up support vector machines". In: *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 760–766 (cited on page 110).

Z. Xia, L.-Y. Wu, X. Zhou, and S. T. Wong (2010). "Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces". In: *BMC Systems Biology* 4.Suppl 2, S6 (cited on page 109).

Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kanehisa (2008). "Prediction of drug-target interaction networks from the integration of chemical and genomic spaces". In: *Bioinformatics* 24 (13), pp. i232–i240 (cited on pages 99–102, 110, 120, 136–139, 141).

Y. Yamanishi, M. Kotera, M. Kanehisa, and S. Goto (2010). "Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework." In: *Bioinformatics* 26.12, pp. i246–i254 (cited on pages 100, 124, 127, 143).

W. W. Zachary (1977). "An information flow model for conflict and fission in small groups". In: *Journal of Anthropological Research* 33, pp. 452–473 (cited on pages 8, 53).

R. B. Zadeh and S. Ben-David (2009). "A uniqueness theorem for clustering". In: *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. UAI '09. Montreal, Quebec, Canada: AUAI Press, pp. 639–646. ISBN: 978-0-9749039-5-8 (cited on pages 13, 15, 147).

Summary

This thesis focuses on two different machine learning problems in the context of networks.

In part one of this thesis we look at clustering, the problem of finding groups of nodes in networks. There is no precise definition of clustering, but a common approach is to say that a 'good' clustering is one with a high quality, according to some quality function. But still, there are many possible choices of quality functions, and is not clear which are the right ones for a certain problem.

Therefore, in **Chapter 3** we give an axiomatic formalization of network clustering quality functions. We discussed six desirable properties that we call axioms, and show that some popular quality functions do not satisfy all of them. This leads us to introduce a novel and flexible quality function that does satisfy all the discussed properties.

Chapter 4 considers the same problem of comparing quality functions for network clustering from a more empirical perspective. We compare the performance of the different quality functions on progressively harder artificial benchmarks, as well as real world datasets. The main difference between the quality functions is in the number of clusters that are found, and if that number is kept fixed, most quality functions behave similarly. We also introduce several new quality functions.

In **Chapter 5** we look at a concrete application of clustering, to protein–protein interaction networks. We show how the method introduced in the previous chapter can be used to find protein complexes, and experimentally validate the efficacy of the method for this purpose. An important advantage of out method compared to others is also that it is very robust to errors in the network.

In the real world, clusters are often not as crisp as we made them out to be so far. In **Chapter 6** we therefore look at soft clustering, where the nodes can belong to more than one cluster, and the memberships need not be binary. We again take an axiomatic approach, and look at locality properties in particular. We argue that locality is a desirable property because it allows local reasoning about

parts of a clustering. This leads us to introduce a new prior for probabilistic Non-negative Matrix Factorization, a popular way of finding overlapping or soft clusterings that is otherwise not local.

In part two of this thesis we lookat another machine learning problem on networks, that of link prediction. In particular, the prediction of new interactions between drug compounds and target proteins.

In **Chapter 8** we introduce a kernel method for this interaction prediction problem. Stated simply, the idea is that two drugs are similar if they interact with the same or a similar set of target proteins. We construct a kernel based on this profile of known interactions, and show that together with a simple ranking method it can lead to state-of-the-art performance.

This kernel is only useful for predicting new interactions between drugs and targets for which some interactions are already known. In **Chapter 9** we address this issue, and use a simple weighted nearest neighbor scheme to extend the kernel also to drugs and targets with no known interactions.

In **Chapter 10** we investigated the biases of the drug–target interaction datasets used in our experiments, and also in several similar works. The problem is that the dataset only contains drug compounds and target proteins with at least one known interaction. This bias can be easily exploited to make a method look very good in cross-validation experiments. We then propose ways in which the evaluation procedure can be made robust to this problem and how the dataset could be corrected.

Samenvatting

Dit proefschrift behandeld twee verschillende Machine Learning onderwerpen die beide te maken hebben met netwerken.

In het eerste deel kijken we naar clustering, waar het doel is om groepen van knopen te vinden in een netwerk. Er is geen preciese definitie van clustering, maar een veel gebruikte aanpak is om te zeggen dat een clustering goed is als deze een hoge 'score' heeft volgens een bepaalde scorefunctie. Dit verschuift slechts het probleem, want er zijn veel verschilende scorefuncties voor netwerk clustering te vinden in de literatuur, en het is niet duidelijk welke de beste is voor een bepaald probleem.

Daarom geven we in **hoofdstuk 3** een axiomatische formalisatie van netwerk clustering scorefuncties. We beschrijven zes eigenschappen die elke goede scorefunctie naar onze mening zou moeten hebben. Vervolgens laten we van een aantal populaire scorefuncties zien dat ze niet aan al deze eigenschappen voldoen. Dit motiveert de introductie van een nieuwe en flexibele scorefunctie, die wel aan alle beschreven eigenschappen voldoet.

Hoofdstuk 4 bekijkt probleem van het kiezen van een goede scorefunctie van een meer empirische kant. We vergelijken verschillende scorefuncties door ze uit te testen op steeds moeilijkere artificiële netwerken, en ook op een aantal echte datasets. Het belangrijkste verschil tussen de scorefuncties blijkt het aantal clusters dat wordt gevonden. Als dat aantal wordt vastgezet dan gedragen de scorefuncties zich vergelijkbaar. Ook introduceren we een aantal nieuwe scorefuncties.

In **hoofdstuk 5** richten we ons op een concrete toepassing van clustering. We gebruiken clustering voor het vinden van eiwitcomplexen in een eiwit–eiwit interactie netwerk. Hiervoor gebruiken we de methoden uit het vorige hoofdstuk. Middels een aantal experimenten laten we zien dat de methode zeer effectief is voor dit doel, met als belangrijk voordeel boven andere methoden het feit dat het veel robuuster is voor kleine veranderingen of fouten in het netwerk.

In het echte leven zijn clusters vaak niet zo vastomlijnd zijn als waar we tot nu toe van uit zijn gegaan. Daarom kijken we in **hoofdstuk 6** naar 'zachte' en overlappende clustering. Hier kunnen knopen deel uit maken van meer dan een cluster, en hun lidmaatschap kan partieel zijn in plaats van een harde 'ja' of 'nee'. We gebruiken opnieuw een axiomatische aanpak, en kijken in het bijzonder naar lokaliteit. Lokaliteit is een belangrijke eigenschap, die het mogelijk maakt om lokaal te redeneren over delen van een clustering. Vervolgens introduceren we een nieuwe prior voor probabilistische Niet-negatieve Matrix Factorisatie, een veel gebruikte manier om dergelijke zachte clusterings te vinden, die zonder deze prior niet lokaal is.

Het tweede deel van dit proefschrift gaat over een ander Machine Learning onderwerp: het voorspellen van verbindingen in een netwerk. In het bijzonder het voorspellen van nieuwe interacties tussen geneesmiddelen en eiwitten.

Hoofdstuk 8 beschrijft een kernel methode voor het voorspellen van dergelijke interacties. Kort samengevat is het achterliggende idee dat twee geneesmiddelen op elkaar lijken als ze interacties hebben met dezelfde eiwitten. We bouwen een kernel op basis van de bekende interacties, en laten zien dat samen met een simpele regressie methode deze kernel state-of-the-art resultaten behaald.

Deze kernel is echter alleen nuttig voor het voorspellen van nieuwe interacties tussen geneesmiddelen en eiwitten waarvoor al een aantal interacties bekend is. In **hoofdstuk 9** lossen we dat gebrek op met een simpele gewogen Nearest Neighbor model. Hiermee kunnen we de kernel uit te breiden naar geneesmiddelen en eiwitten zonder bekende interacties.

In **hoofdstuk 10** onderzoeken we hoe de eigenschappen van de door ons en anderen gebruikte datasets een vertekend beeld kunnen geven bij experimentele validatie. Het probleem is dat in de dataset alleen geneesmidellen en eiwitten zitten met tenminste één bekende interactie. De selectie is dus niet volledig willekeurig, en die eigenschap kan eenvoudig gebruikt worden om een methode goed te laten lijken bij cross-validatie experimenten. We laten zien hoe de validatieprocedure kan worden aangepast om dit probleem te voorkomen.

Acknowledgements

Getting a PhD always seemed like this obvious thing to do after finishing my masters degree. And indeed research is a lot of fun. But some of the aspects that come with it, such as writing down your findings, are a bit less so. Because of that, completing my PhD thesis was harder than I initially thought. And I certainly couldn't have done it on my own.

I would therefore first and foremost like to thank my supervisor, Elena Marchiori, for motivating me to finish what I started and for providing directions during four years of research. And want to thank Tom Heskes for his role as my promotor, which included always being there to answer questions, even though I hardly had any.

I'd also like to thank my other colleagues at the Intelligent Systems section for providing interesting conversations about various research topics, as well as other topic during lunch breaks and Friday afternoon drinks; and for providing a nice work atmosphere in general. This includes not just the Machine Learning people, but also everyone from the foundations group, with whom I have spend quite a bit of time, even though they work on completely different (but still interesting) topics. So Adriana, Ali, Bram, Carst, Daniel, Dimitris, Elena S., Fabio, Freek, Helle, Herman, Jelle, Jonce, Joris, Josef, Maya, Max, Nicole, Pavol, Ridho, Robbert, Saiden, Saskia, Simone, Suzan, Thijs, Tjeerd, Tom C., Wout, thank you for the interesting times.

Besides these people at the computer science institute, this work would not have been possible without the support of my other friends and my family.

Finally, I should thank the Netherlands Organization for Scientific Research (NWO) for funding project 612.066.927, which made my PhD project possible in the first place.

SIKS Dissertatiereeks

2009

2009-01 Rasa Jurgelenaite (RUN)
Symmetric Causal Independence Models

2009-02 Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques

2009-03 Hans Stol (UvT)
A Framework for Evidence-based Policy Making Using IT

2009-04 Josephine Nabukenya (RUN)
Improving the Quality of Organisational Policy Making using Collaboration Engineering

2009-05 Sietse Overbeek (RUN)
Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality

2009-06 Muhammad Subianto (UU)
Understanding Classification

2009-07 Ronald Poppe (UT)
Discriminative Vision-Based Recovery and Recognition of Human Motion

2009-08 Volker Nannen (VU)
Evolutionary Agent-Based Policy Analysis in Dynamic Environments

2009-09 Benjamin Kanagwa (RUN)
Design, Discovery and Construction of Service-oriented Systems

2009-10 Jan Wielemaker (UVA)
Logic programming for knowledge-intensive interactive applications

2009-11 Alexander Boer (UVA)
Legal Theory, Sources of Law & the Semantic Web

2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin)
Operating Guidelines for Services

2009-13 Steven de Jong (UM)
Fairness in Multi-Agent Systems

2009-14 Maksym Korotkiy (VU)
From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)

2009-15 Rinke Hoekstra (UVA)
Ontology Representation - Design Patterns and Ontologies that Make Sense

2009-16 Fritz Reul (UvT)
New Architectures in Computer Chess

2009-17 Laurens van der Maaten (UvT)
Feature Extraction from Visual Data

2009-18 Fabian Groffen (CWI)
Armada, An Evolving Database System

2009-19 Valentin Robu (CWI)
Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets

2009-20 Bob van der Vecht (UU)
Adjustable Autonomy: Controling Influences on Decision Making

2009-21 Stijn Vanderlooy (UM)
Ranking and Reliable Classification

2009-22 Pavel Serdyukov (UT)
Search For Expertise: Going beyond direct evidence

2009-23 Peter Hofgesang (VU)
Modelling Web Usage in a Changing Environment

2009-24 Annerieke Heuvelink (VUA)
Cognitive Models for Training Simulations

2009-25 Alex van Ballegooij (CWI)
"RAM: Array Database Management through Relational Mapping"

2009-26 Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services
2009-27 Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web
2009-28 Sander Evers (UT)
Sensor Data Management with Probabilistic Models
2009-29 Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications
2009-30 Marcin Zukowski (CWI)
Balancing vectorized query execution with bandwidth-optimized storage
2009-31 Sofiya Katrenko (UVA)
A Closer Look at Learning Relations from Text
2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)
Architectural Knowledge Management: Supporting Architects and Auditors
2009-33 Khiet Truong (UT)
How Does Real Affect Affect Affect Recognition In Speech?
2009-34 Inge van de Weerd (UU)
Advancing in Software Product Management: An Incremental Method Engineering Approach
2009-35 Wouter Koelewijn (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling
2009-36 Marco Kalz (OUN)
Placement Support for Learners in Learning Networks
2009-37 Hendrik Drachsler (OUN)
Navigation Support for Learners in Informal Learning Networks
2009-38 Riina Vuorikari (OU)
Tags and self-organisation: a metadata ecology for learning resources in a multilingual context
2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)
Service Substitution – A Behavioral Approach Based on Petri Nets
2009-40 Stephan Raaijmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language
2009-41 Igor Berezhnyy (UvT)
Digital Analysis of Paintings
2009-42 Toine Bogers
Recommender Systems for Social Bookmarking
2009-43 Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients
2009-44 Roberto Santana Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations
2009-45 Jilles Vreeken (UU)
Making Pattern Mining Useful
2009-46 Loredana Afanasiev (UvA)
Querying XML: Benchmarks and Recursion

2010

2010-01 Matthijs van Leeuwen (UU)
Patterns that Matter
2010-02 Ingo Wassink (UT)
Work flows in Life Science
2010-03 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents
2010-04 Olga Kulyk (UT)
Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments
2010-05 Claudia Hauff (UT)
Predicting the Effectiveness of Queries and Retrieval Systems
2010-06 Sander Bakkes (UvT)
Rapid Adaptation of Video Game AI
2010-07 Wim Fikkert (UT)

Gesture interaction at a Distance

2010-08 Krzysztof Siewicz (UL)
Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments

2010-09 Hugo Kielman (UL)
A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging

2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children

2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning

2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis

2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques

2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration

2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-Organizational Models

2010-16 Sicco Verwer (TUD)
Efficient Identification of Timed Automata, theory and practice

2010-17 Spyros Kotoulas (VU)
Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications

2010-18 Charlotte Gerritsen (VU)
Caught in the Act: Investigating Crime by Agent-Based Simulation

2010-19 Henriette Cramer (UvA)
People's Responses to Autonomous and Adaptive Systems

2010-20 Ivo Swartjes (UT)
Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative

2010-21 Harold van Heerde (UT)
Privacy-aware data management by means of data degradation

2010-22 Michiel Hildebrand (CWI)
End-user Support for Access to Heterogeneous Linked Data

2010-23 Bas Steunebrink (UU)
The Logical Structure of Emotions

2010-24 Dmytro Tykhonov
Designing Generic and Efficient Negotiation Strategies

2010-25 Zulfiqar Ali Memon (VU)
Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective

2010-26 Ying Zhang (CWI)
XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines

2010-27 Marten Voulon (UL)
Automatisch contracteren

2010-28 Arne Koopman (UU)
Characteristic Relational Patterns

2010-29 Stratos Idreos(CWI)
Database Cracking: Towards Auto-tuning Database Kernels

2010-30 Marieke van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval

2010-31 Victor de Boer (UVA)
Ontology Enrichment from Heterogeneous Sources on the Web

2010-32 Marcel Hiel (UvT)
An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems

2010-33 Robin Aly (UT)
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval

2010-34 Teduh Dirgahayu (UT)
Interaction Design in Service Compositions

2010-35 Dolf Trieschnigg (UT)

Proof of Concept: Concept-based Biomedical Information Retrieval

2010-36 Jose Janssen (OU)
Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification

2010-37 Niels Lohmann (TUE)
Correctness of services and their composition

2010-38 Dirk Fahland (TUE)
From Scenarios to components

2010-39 Ghazanfar Farooq Siddiqui (VU)
Integrative modeling of emotions in virtual agents

2010-40 Mark van Assem (VU)
Converting and Integrating Vocabularies for the Semantic Web

2010-41 Guillaume Chaslot (UM)
Monte-Carlo Tree Search

2010-42 Sybren de Kinderen (VU)
Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach

2010-43 Peter van Kranenburg (UU)
A Computational Approach to Content-Based Retrieval of Folk Song Melodies

2010-44 Pieter Bellekens (TUE)
An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain

2010-45 Vasilios Andrikopoulos (UvT)
A theory and model for the evolution of software services

2010-46 Vincent Pijpers (VU)
e3alignment: Exploring Inter-Organizational Business-ICT Alignment

2010-47 Chen Li (UT)
Mining Process Model Variants: Challenges, Techniques, Examples

2010-48 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2010-49 Jahn-Takeshi Saito (UM)
Solving difficult game positions

2010-50 Bouke Huurnink (UVA)
Search in Audiovisual Broadcast Archives

2010-51 Alia Khairia Amin (CWI)
Understanding and supporting information seeking tasks in multiple sources

2010-52 Peter-Paul van Maanen (VU)
Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention

2010-53 Edgar Meij (UVA)
Combining Concepts and Language Models for Information Access

---

2011

2011-01 Botond Cseke (RUN)
Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-02 Nick Tinnemeier(UU)
Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-03 Jan Martijn van der Werf (TUE)
Compositional Design and Verification of Component-Based Information Systems

2011-04 Hado van Hasselt (UU)
Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms

2011-05 Base van der Raadt (VU)
Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

2011-06 Yiwen Wang (TUE)
Semantically-Enhanced Recommendations in Cultural Heritage

2011-07 Yujia Cao (UT)

Multimodal Information Presentation for High Load Human Computer Interaction

2011-08 Nieske Vergunst (UU)
BDI-based Generation of Robust Task-Oriented Dialogues

2011-09 Tim de Jong (OU)
Contextualised Mobile Media for Learning

2011-10 Bart Bogaert (UvT)
Cloud Content Contention

2011-11 Dhaval Vyas (UT)
Designing for Awareness: An Experience-focused HCI Perspective

2011-12 Carmen Bratosin (TUE)
Grid Architecture for Distributed Process Mining

2011-13 Xiaoyu Mao (UvT)
Airport under Control. Multiagent Scheduling for Airport Ground Handling

2011-14 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2011-15 Marijn Koolen (UvA)
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

2011-16 Maarten Schadd (UM)
Selective Search in Games of Different Complexity

2011-17 Jiyin He (UVA)
Exploring Topic Structure: Coherence, Diversity and Relatedness

2011-18 Mark Ponsen (UM)
Strategic Decision-Making in complex games

2011-19 Ellen Rusman (OU)
The Mind's Eye on Personal Profiles

2011-20 Qing Gu (VU)
Guiding service-oriented software engineering - A view-based approach

2011-21 Linda Terlouw (TUD)
Modularization and Specification of Service-Oriented Systems

2011-22 Junte Zhang (UVA)
System Evaluation of Archival Description and Access

2011-23 Wouter Weerkamp (UVA)
Finding People and their Utterances in Social Media

2011-24 Herwin van Welbergen (UT)
Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior

2011-25 Syed Waqar ul Qounain Jaffry (VU)
Analysis and Validation of Models for Trust Dynamics

2011-26 Matthijs Aart Pontier (VU)
Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots

2011-27 Aniel Bhulai (VU)
Dynamic website optimization through autonomous management of design patterns

2011-28 Rianne Kaptein(UVA)
Effective Focused Retrieval by Exploiting Query Context and Document Structure

2011-29 Faisal Kamiran (TUE)
Discrimination-aware Classification

2011-30 Egon van den Broek (UT)
Affective Signal Processing (ASP): Unraveling the mystery of emotions

2011-31 Ludo Waltman (EUR)
Computational and Game-Theoretic Approaches for Modeling Bounded Rationality

2011-32 Nees-Jan van Eck (EUR)
Methodological Advances in Bibliometric Mapping of Science

2011-33 Tom van der Weide (UU)
Arguing to Motivate Decisions

2011-34 Paolo Turrini (UU)
Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

2011-35 Maaike Harbers (UU)
Explaining Agent Behavior in Virtual Training
2011-36 Erik van der Spek (UU)
Experiments in serious game design: a cognitive approach
2011-37 Adriana Burlutiu (RUN)
Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference
2011-38 Nyree Lemmens (UM)
Bee-inspired Distributed Optimization
2011-39 Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games
2011-40 Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development
2011-41 Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control
2011-42 Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution
2011-43 Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge
2011-44 Boris Reuderink (UT)
Robust Brain-Computer Interfaces
2011-45 Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection
2011-46 Beibei Hu (TUD)
Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
2011-47 Azizi Bin Ab Aziz (VU)
Exploring Computational Models for Intelligent Support of Persons with Depression
2011-48 Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
2011-49 Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

---

2011

2012-01 Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda
2012-02 Muhammad Umair(VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
2012-03 Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories
2012-04 Jurriaan Souer (UU)
Development of Content Management System-based Web Applications
2012-05 Marijn Plomp (UU)
Maturing Interorganisational Information Systems
2012-06 Wolfgang Reinhardt (OU)
Awareness Support for Knowledge Workers in Research Networks
2012-07 Rianne van Lambalgen (VU)
When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
2012-08 Gerben de Vries (UVA)
Kernel Methods for Vessel Trajectories
2012-09 Ricardo Neisse (UT)
Trust and Privacy Management Support for Context-Aware Service Platforms
2012-10 David Smits (TUE)
Towards a Generic Distributed Adaptive Hypermedia Environment
2012-11 J.C.B. Rantham Prabhakara (TUE)

Process Mining in the Large: Preprocessing, Discovery, and Diagnostics

2012-12 Kees van der Sluijs (TUE)
Model Driven Design and Data Integration in Semantic Web Information Systems

2012-13 Suleman Shahid (UvT)
Fun and Face: Exploring non-verbal expressions of emotion during playful interactions

2012-14 Evgeny Knutov (TUE)
Generic Adaptation Framework for Unifying Adaptive Web-based Systems

2012-15 Natalie van der Wal (VU)
Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes

2012-16 Fiemke Both (VU)
Helping people by understanding them - Ambient Agents supporting task execution and depression treatment

2012-17 Amal Elgammal (UvT)
Towards a Comprehensive Framework for Business Process Compliance

2012-18 Eltjo Poort (VU)
Improving Solution Architecting Practices

2012-19 Helen Schonenberg (TUE)
What's Next? Operational Support for Business Process Execution

2012-20 Ali Bahramisharif (RUN)
Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing

2012-21 Roberto Cornacchia (TUD)
Querying Sparse Matrices for Information Retrieval

2012-22 Thijs Vis (UvT)
Intelligence, politie en veiligheidsdienst: verenigbare grootheden?

2012-23 Christian Muehl (UT)
Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction

2012-24 Laurens van der Werff (UT)
Evaluation of Noisy Transcripts for Spoken Document Retrieval

2012-25 Silja Eckartz (UT)
Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application

2012-26 Emile de Maat (UVA)
Making Sense of Legal Text

2012-27 Hayrettin Gurkok (UT)
Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games

2012-28 Nancy Pascall (UvT)
Engendering Technology Empowering Women

2012-29 Almer Tigelaar (UT)
Peer-to-Peer Information Retrieval

2012-30 Alina Pommeranz (TUD)
Designing Human-Centered Systems for Reflective Decision Making

2012-31 Emily Bagarukayo (RUN)
A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure

2012-32 Wietske Visser (TUD)
Qualitative multi-criteria preference representation and reasoning

2012-33 Rory Sie (OUN)
Coalitions in Cooperation Networks (COCOON)

2012-34 Pavol Jancura (RUN)
Evolutionary analysis in PPI networks and applications

2012-35 Evert Haasdijk (VU)
Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics

2012-36 Denis Ssebugwawo (RUN)
Analysis and Evaluation of Collaborative Modeling Processes

2012-37 Agnes Nakakawa (RUN)

A Collaboration Process for Enterprise Architecture Creation

2012-38 Selmar Smit (VU)
Parameter Tuning and Scientific Testing in Evolutionary Algorithms

2012-39 Hassan Fatemi (UT)
Risk-aware design of value and coordination networks

2012-40 Agus Gunawan (UvT)
Information Access for SMEs in Indonesia

2012-41 Sebastian Kelle (OU)
Game Design Patterns for Learning

2012-42 Dominique Verpoorten (OU)
Reflection Amplifiers in self-regulated Learning

2012-43 *Withdrawn*

2012-44 Anna Tordai (VU)
On Combining Alignment Techniques

2012-45 Benedikt Kratz (UvT)
A Model and Language for Business-aware Transactions

2012-46 Simon Carter (UVA)
Exploration and Exploitation of Multilingual Data for Statistical Machine Translation

2012-47 Manos Tsagkias (UVA)
Mining Social Media: Tracking Content and Predicting Behavior

2012-48 Jorn Bakker (TUE)
Handling Abrupt Changes in Evolving Time-series Data

2012-49 Michael Kaisers (UM)
Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions

2012-50 Steven van Kervel (TUD)
Ontologogy driven Enterprise Information Systems Engineering

2012-51 Jeroen de Jong (TUD)
Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching

---

2013

2013-01 Viorel Milea (EUR)
News Analytics for Financial Decision Support

2013-02 Erietta Liarou (CWI)
MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing

2013-03 Szymon Klarman (VU)
Reasoning with Contexts in Description Logics

2013-04 Chetan Yadati (TUD)
Coordinating autonomous planning and scheduling

2013-05 Dulce Pumareja (UT)
Groupware Requirements Evolutions Patterns

2013-06 Romulo Goncalves(CWI)
The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience

2013-07 Giel van Lankveld (UvT)
Quantifying Individual Player Differences

2013-08 Robbert-Jan Merk (VU)
Making enemies: cognitive modeling for opponent agents in fighter pilot simulators

2013-09 Fabio Gori (RUN)
Metagenomic Data Analysis: Computational Methods and Applications

2013-10 Jeewanie Jayasinghe Arachchige (UvT)
A Unified Modeling Framework for Service Design

2013-11 Evangelos Pournaras (TUD)
Multi-level Reconfigurable Self-organization in Overlay Services

2013-12 Marian Razavian (VU)
Knowledge-driven Migration to Services

2013-13 Mohammad Safiri(UT)

Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly

2013-14 Jafar Tanha (UVA)
Ensemble Approaches to Semi-Supervised Learning Learning

2013-15 Daniel Hennes (UM)
Multiagent Learning - Dynamic Games and Applications

2013-16 Eric Kok (UU)
Exploring the practical benefits of argumentation in multi-agent deliberation

2013-17 Koen Kok (VU)
The PowerMatcher: Smart Coordination for the Smart Electricity Grid

2013-18 Jeroen Janssens (UvT)
Outlier Selection and One-Class Classification

2013-19 Renze Steenhuizen (TUD)
Coordinated Multi-Agent Planning and Scheduling

2013-20 Katja Hofmann (UvA)
Fast and Reliable Online Learning to Rank for Information Retrieval

2013-21 Sander Wubben (UvT)
Text-to-text generation by monolingual machine translation

2013-22 Tom Claassen (RUN)
Causal Discovery and Logic

2013-23 Patricio de Alencar Silva(UvT)
Value Activity Monitoring

2013-24 Haitham Bou Ammar (UM)
Automated Transfer in Reinforcement Learning

2013-25 Agnieszka Anna Latoszek-Berendsen (UM)
Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System

2013-26 Alireza Zarghami (UT)
Architectural Support for Dynamic Homecare Service Provisioning

2013-27 Mohammad Huq (UT)
Inference-based Framework Managing Data Provenance

2013-28 Frans van der Sluis (UT)
When Complexity becomes Interesting: An Inquiry into the Information eXperience

2013-29 Iwan de Kok (UT)
Listening Heads

2013-30 Joyce Nakatumba (TUE)
Resource-Aware Business Process Management: Analysis and Support

2013-31 Dinh Khoa Nguyen (UvT)
Blueprint Model and Language for Engineering Cloud Applications

2013-32 Kamakshi Rajagopal (OUN)
Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development

2013-33 Qi Gao (TUD)
User Modeling and Personalization in the Microblogging Sphere

2013-34 Kien Tjin-Kam-Jet (UT)
Distributed Deep Web Search

2013-35 Abdallah El Ali (UvA)
Minimal Mobile Human Computer Interaction

2013-36 Than Lam Hoang (TUe)
Pattern Mining in Data Streams

2013-37 Dirk Börner (OUN)
Ambient Learning Displays

2013-38 Eelco den Heijer (VU)
Autonomous Evolutionary Art

2013-39 Joop de Jong (TUD)
A Method for Enterprise Ontology based Design of Enterprise Information Systems

2013-40 Pim Nijssen (UM)
Monte-Carlo Tree Search for Multi-Player Games

2013-41 Jochem Liem (UVA)
Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning

2013-42 Léon Planken (TUD)
Algorithms for Simple Temporal Reasoning

2013-43 Marc Bron (UVA)
Exploration and Contextualization through Interaction and Concepts

---

2014

2014-01 Nicola Barile (UU)
Studies in Learning Monotone Models from Data

2014-02 Fiona Tuliyano (RUN)
Combining System Dynamics with a Domain Modeling Method

2014-03 Sergio Raul Duarte Torres (UT)
Information Retrieval for Children: Search Behavior and Solutions

2014-04 Hanna Jochmann-Mannak (UT)
Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation

2014-05 Jurriaan van Reijsen (UU)
Knowledge Perspectives on Advancing Dynamic Capability

2014-06 Damian Tamburri (VU)
Supporting Networked Software Development

2014-07 Arya Adriansyah (TUE)
Aligning Observed and Modeled Behavior

2014-08 Samur Araujo (TUD)
Data Integration over Distributed and Heterogeneous Data Endpoints

2014-09 Philip Jackson (UvT)
Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language

2014-10 Ivan Salvador Razo Zapata (VU)
Service Value Networks

2014-11 Janneke van der Zwaan (TUD)
An Empathic Virtual Buddy for Social Support

2014-12 Willem van Willigen (VU)
Look Ma, No Hands: Aspects of Autonomous Vehicle Control

2014-13 Arlette van Wissen (VU)
Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains

2014-14 Yangyang Shi (TUD)
Language Models With Meta-information

2014-15 Natalya Mogles (VU)
Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare

2014-16 Krystyna Milian (VU)
Supporting trial recruitment and design by automatically interpreting eligibility criteria

2014-17 Kathrin Dentler (VU)
Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability

2014-18 Mattijs Ghijsen (VU)
Methods and Models for the Design and Study of Dynamic Agent Organizations

2014-19 Vincius Ramos (TUE)
Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support

2014-20 Mena Habib (UT)
Named Entity Extraction and Disambiguation for Informal Text: The Missing Link

2014-21 Kassidy Clark (TUD)
Negotiation and Monitoring in Open Environments

2014-22 Marieke Peeters (UU)
Personalized Educational Games - Developing agent-supported scenario-based training

2014-23 Eleftherios Sidirourgos (UvA/CWI)
Space Efficient Indexes for the Big Data Era
2014-24 Davide Ceolin (VU)
Trusting Semi-structured Web Data
2014-25 Martijn Lappenschaar (RUN)
New network models for the analysis of disease interaction
2014-26 Tim Baarslag (TUD)
What to Bid and When to Stop
2014-27 Rui Jorge Almeida (EUR)
Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty
2014-28 Anna Chmielowiec (VU)
Decentralized k-Clique Matching
2014-29 Jaap Kabbedijk (UU)
Variability in Multi-Tenant Enterprise Software
2014-30 Peter de Kock Berenschot (UvT)
Anticipating Criminal Behaviour
2014-31 Leo van Moergestel (UU)
Agent Technology in Agile Multiparallel Manufacturing and Product Support
2014-32 Naser Ayat (UVA)
On Entity Resolution in Probabilistic Data
2014-33 Tesfa Tegegne Asfaw (RUN)
Service Discovery in eHealth
2014-34 Christina Manteli (VU)
The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems
2014-35 Joost van Oijen (UU)
Cognitive Agents in Virtual Worlds: A Middleware Design Approach
2014-36 Joos Buijs (TUE)
Flexible Evolutionary Algorithms for Mining Structured Process Models
2014-37 Maral Dadvar (UT)
Experts and Machines United Against Cyberbullying
2014-38 Danny Plass-Oude Bos (UT)
Making brain-computer interfaces better: improving usability through post-processing
2014-39 Jasmina Maric (UvT)
Web Communities, Immigration and Social Capital
2014-40 Walter Oboma (RUN)
A Framework for Knowledge Management Using ICT in Higher Education
2014-41 Frederik Hogenboom (EUR)
Automated Detection of Financial Events in News Text
2014-42 Carsten Eickhoff (CWI/TUD)
Contextual Multidimensional Relevance Models
2014-43 Kevin Vlaanderen (UU)
Supporting Process Improvement using Method Increments
2014-44 Paulien Meesters (UvT)
Intelligent Blauw. Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden
2014-45 Birgit Schmitz (OU)
Mobile Games for Learning: A Pattern-Based Approach
2014-46 Ke Tao (TUD)
Social Web Data Analytics: Relevance, Redundancy, Diversity
2014-47 Shangsong Liang (UVA)
Fusion and Diversification in Information Retrieval

2015

2015-01 Niels Netten (UVA)
Machine Learning for Relevance of Information in Crisis Response
2015-02 Faiza Bukhsh (UvT)
Smart auditing: Innovative Compliance Checking in Customs Controls