DIPLOMARBEIT

Spectral Clustering with Sampling — A Graph Signal Processing Perspective

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplomingenieurs

unter der Leitung von
Dr. Peter Berger
Ao. Univ.-Prof. Dr. Gerald Matz
Institute of Telecommunications

eingereicht an der Technische Universität Wien
Fakultät für Elektrotechnik

von
Thomas Dittrich
1260174

Wien, 8. Juni 2018 _____

Abstract

Graph signal processing is an extension of classical signal processing to signals that are defined over the vertex set of graphs. This type of signal processing is necessary in applications where the data is best represented by a graph (like in social networks) and when the amount of data is big. Usually a graph is generated that represents the similarity of different data points and then the typically sparse weight matrix can be used for efficient analysis. One application of graph signal processing is clustering, where the aim is to identify groups of similar data items. This is used for data analysis in several fields ranging from biology over image processing to machine learning.

In this work we consider various clustering algorithms. We focus on signed spectral clustering with sampling, which is an algorithm that modifies the weight matrix of a given graph based on some sampled cluster labels and then performs signed spectral clustering. In machine learning nomenclature methods that are supported by sampled cluster labels are referred to as a semi-supervised clustering.

At first, algorithms for graph learning are presented. They are used as a pre-processing stage for graph-based clustering. Those algorithms are the ϵ-neighborhood graph and the k-nearest neighborhood graph. Next, k-means and spectral clustering, which are well known examples of unsupervised clustering algorithms, are discussed and then we introduce an original method that combines signed spectral clustering with sampling. In the last part signed spectral clustering with sampling is compared to clustering by harmonic functions and singular value projection in Monte-Carlo simulations. Our numerical results show that signed spectral clustering with sampling performs superior compared to existing algorithms.

Kurzfassung

Die Graphsignalverarbeitung ist eine Erweiterung der klassischen Signalverarbeitung, bei der die Signale auf der Knotenmenge eines Graphen definiert sind. Diese Art der Signalverarbeitung ist notwendig, wenn die Daten, wie in einem sozialen Netzwerk, als Graph gegeben sind, oder wenn die Menge der Daten sehr groß ist. Üblicherweise werden die Daten zuerst in einen Graphen übergeführt, der die Ähnlichkeit der Datenpunkte repräsentiert. Anschließend wird die üblicherweise spärlich besetzte Gewichtsmatrix verwendet, um eine Analyse der Daten effizient durchzuführen. Eine Anwendung der Graphsignalverarbeitung ist die Identifikation von Gruppen von ähnlichen Datenpunkten. Dieses sogenannte Clustering wird zum Beispiel in der Biologie, der Bildverarbeitung oder im maschinellen Lernen eingesetzt.

In dieser Arbeit beschäftigen wir uns mit Clustering-Algorithmen, wobei der Fokus auf vorzeichenbehaftetem Spectral Clustering mit Abtastung liegt. Dieser Algorithmus modifiziert die Gewichtsmatrix eines gegebenen Graphen basierend auf der abgetasteten Cluster-Zugehörigkeit. Anschließend wird vorzeichenbehaftetes Spectral Clustering auf die modifizierte Gewichtsmatrix angewendet. Verfahren, bei denen die Cluster-Zugehörigkeit abgetastet wird, wird in der Nomenklatur des maschinellen Lernens als semi-supervised Clustering Algorithmus.

Zu Beginn werden die beiden Algorithmen ϵ-neighborhood Graph und k-nearest neighborhood Graph zum Lernen von Graphen präsentiert. Danach werden k-means und Spectral Clustering, als Beispiele für Clustering Algorithmen ohne Abtastung diskutiert. Anschließend wird Spectral Clustering zu vorzeichenbehaftetem Spectral Clustering mit Abtastung erweitert. Zum Abschluss wird der erweiterte Algorithmus mittels Monte-Carlo Simulationen mit zwei anderen Clustering Algorithmen verglichen. Es zeigt sich, dass der neue Algorithmus bessere Ergebnisse liefert als die beiden Vergleichsalgorithmen.

Danksagung

An dieser stelle möchte ich mich ganz besonders bei meinen Betreuern Dr. Peter Berger und Ao. Univ.-Prof. Dr. Gerald Matz bedanken, die mir bei der Durchfühfung dieser Arbeit mit hilfreichen Kommentaren, konstruktiver Kritik und motivierenden Worten zur Seite gestanden sind. Mein Dank gilt auch meinen Freunden, mit denen ich immer einen Ausgleich zum Studium finden konnte, um anschließend wieder gestärkt an die Arbeit zu gehen. Zum Ende möchte ich mich noch bei meiner Familie bedanken, die mich immer unterstützt haben und mir ganz besonders in schweren Zeiten den Rücken gestärkt haben.

Contents

1

Introduction

In applications with big and multidimensional datasets, like social networks, recommendation systems or image processing, classical signal processing is nor easily applicable. This is because the fundamental concepts, like frequency analysis, downsampling and filtering, rely on a regular structure of the time domain.

Graph signal processing (GSP) offers a promising approach to relax the time domain to irregular domains [1]. Here, the data set is modeled by a graph, i.e., every data point is represented by a node and a value, and nodes with similar values are connected by edges. The assignment of values to each node is referred to as a graph signal, which is a function from the node set to the signal space [2]. The resulting graph can then be represented by its weight matrix, which is typically sparse as every node is only connected to a small number of other nodes relative to the total number of data points. This sparsity is a very important property, as it enables the efficient implementation of matrix multiplications [3]. Nevertheless, there is the problem that the aforementioned fundamental concepts require adaption to general.

A basic building block of GSP is the graph Laplacian. This is a matrix that represents a difference operation on the graph and can be seen as a discretization of differential operators [4]. A typical approach is then to define the spectrum of the graph based on the eigendecomposition of the Laplacian [5, 6]. This means that the graph Fourier transform (GFT) is a transformation to the basis of eigenvectors of the Laplacian and every eigenvector represents one frequency component. Similar to filtering in classical signal processing it is now possible to filter signals in the spectral domain by

multiplying the spectra and transforming the result to the graph domain by an inverse GFT. Via the spectrum it is also possible to define translation, modulation and dilation on graphs [1].

But before it is possible to apply those concepts to a graph, the graph has to be generated (learned) from the data. In some applications like social networks or image processing, this is inherently given by the data, but in others, like recommendation systems, there is a need for so called graph learning methods. Two of the best known graph learning methods are the ϵ-neighborhood graph and the k-nearest neighborhood graph (KNN), which rely solely on the similarity of data values [7]. There are also other, more complex methods that aim to generate a graph on which the data is smooth in some sense, i.e., the values of a graph signal may not exhibit large changes along edges. An example for those methods is total variation (TV) minimization [8].

In the case that the graph is known and it is only possible to get access to the values of a certain amount of nodes, classical signal processing would go for equally spaced sampling and recover the signal by sinusoid interpolation, which works perfectly for bandlimited signals. For graphs there is in general no possibility to get equally spaced samples due to the complex structure of the graph and so there exists the problem of sampling set selection and signal recovery. For sampling set selection there is already research on the theoretical limits for the reconstruction from noisy samples with random sampling and experimentally designed sampling [9, 10, 11]. Those performance bounds are all based on smooth graph signals, among which the bandlimited model is easiest to be compared to classical signal processing. For a bandlimited graph signal the spectrum obtained by the GFT is non-zero only in a certain number of spectral components. This even allows the selection of a sampling set that guarantees perfect reconstruction if the measured data is noise free [12]. In the presence of noise or for other smoothness models, the methods can be designed efficiently in different ways. Possible approaches are to greedily select nodes such that the mean squared error is minimized [13], or to find a bipartition of the graph by maximum spanning trees [14]. The problem of graph signal recovery is typically formulated as a mean squared error minimization problem with the constraint that the signal has to fulfill some smoothness criterion. Possible smoothness criteria can, for example, be based on kernel functions [15] or the TV [16].

A concept of GSP that is very important in machine learning is clustering. The aim of clustering is to find groups of similar data points that are highly connected to members of the same group but only loosely connected to nodes from different clusters. The applications for clustering include community detection in social networks, anomaly detection in image processing,

and phenetic clustering in biology. The clustering problem can be seen as a special case of signal recovery as the labels of the clusters can be interpreted as graph signal. Semi-supervised clustering, i.e., knowledge of some cluster labels, is then equivalent to sampling of the associated graph signal. Well known algorithms for unsupervised clustering are k-means [17] and spectral clustering (SC) [18, 7] and for semi-supervised clustering there are, for example, [19, 20, 21, 22, 23].

1.1 Contribution

This work aims to introduce some of the concepts of GSP in more detail, give an overview of the process of detecting clusters in a set of data points, and provide in-depth explanations of signed spectral clustering with sampling (sSCs) [24]. sSCs is an extension of the work from [19], which makes use of signed spectral clustering (sSC). The idea of [19] was to modify the weight matrix of the graph depending on the similarity of the cluster labels for every pair of samples; when both nodes have the same cluster label, the corresponding entry in the weight matrix is set to 1 and if the labels are different, it is set to 0. In sSCs this modification of the weight matrix is extended such that for equal cluster labels the entry is set to w_{sim} and for different labels it is set to $-w_{dis}$. This modification allows to not only reward labeling samples from the same cluster equally, but also to reward labeling samples from different clusters differently. The latter would not be possible in the case of $w_{dis} = 0$ from [19]. In an approach to find optimal values for w_{sim} and w_{dis} we will also analyze the non-relaxed version of sSC, which is a minimization of the signed ratio cut (sRC), and derive lower bounds for those two values for which the labeling of samples is enforced to be consistent to the sampled values.

In our experiments we will show that sSCs outperforms its predecessor and also some other popular clustering algorithms, especially for low numbers of samples. Then it will be used to analyze two different kinds of sampling strategies, namely uniform node sampling and uniform edge sampling, and finally its performance will be analyzed for varying w_{sim} and w_{dis}, where we will see that the derived bounds are more conservative for the non-relaxed case than it would be necessary for the relaxed case.

1.2 Notation

Throughout this work we will use the following notation:

- scalars will be denoted by italic letters ($N \in \mathbb{N}$, $a \in \mathbb{R}$);
- vectors will be denoted by lowercase, boldface letters ($\boldsymbol{x} \in \mathbb{R}^N$);
- matrices will be denoted by uppercase, boldface letters ($\boldsymbol{A} \in \mathbb{R}^{N \times M}$);
- the ith element of a vector $\boldsymbol{x}$ will be denoted by x_i;
- the ith row and the jth column of a matrix $\boldsymbol{A}$ will be written as $\boldsymbol{A}_{i:}$ and $\boldsymbol{A}_{:j}$, respectively;
- the ith element of the jth column of a matrix $\boldsymbol{A}$ will be written as A_{ij};
- sets and families of sets will be denoted by calligraphic letters or greek letters ($\mathcal{V}$, ε);
- the set difference of the sets $\mathcal{A}, \mathcal{B}$ is $\mathcal{A} \setminus \mathcal{B} = \{a : a \in \mathcal{A} \wedge a \notin \mathcal{B}\}$

Unless stated otherwise we will always use undirected, simple and self-loop free graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with the vertex set $\mathcal{V} = \{1, \ldots, N\}$, the edge set $\mathcal{E}$, and the weight matrix $\boldsymbol{W}$. An undirected graph is a graph for which every edge is an unordered pair of vertices, i.e., $\mathcal{E}$ is a family of sets with every edge $\varepsilon \in \mathcal{E}$ being $\varepsilon \subseteq \mathcal{V}$ and $|\varepsilon| = 2$. For the weight matrix this means that it is symmetric and if and only if $\{i, j\} \in \mathcal{E}$ then $W_{ij} = W_{ji}$ are non-zero. For simple graphs every pair of nodes is at most connected by one edge. The property of being self-loop free means that there exists no edge that connects a node to itself; thus, the diagonal of the weight matrix must be zero. For simplicity of notation we will use $N = |\mathcal{V}|$ and for subsets $\mathcal{V}_i \subseteq \mathcal{V}$ it will be $N_i = |\mathcal{V}_i|$.

2

Background

Before the introduction of the clustering algorithms we will cover some of the basics of GSP in more detail to build a mathematical background. We will start with the definition of balanced graphs to ease the understanding of what it means that clusters exist in a graph, then we will introduce the GSP concepts of graph signals, graph Laplacian, TV and GFT. Finally, as central parts of SC and sSCs, the graph cut and the ratio cut (RC) will be defined, both unsigned and signed.

2.1 Balanced Graphs

The notion of balanced graphs comes from social balance. [25] described this as "a friend of my friend is my friend", "an enemy of my friend is my enemy", and "an enemy of my enemy is my friend". This means that if a social graph is traversed on a cycle, there may be a transition from a friend of the first node in the cycle to a foe of that node and in the end there must be a transition, back to the friends of the first node. In between it is possible that there are arbitrary many transitions from foe to friend and vice versa. A cycle on a graph is a finite sequence of nodes, for which the first node and the last node is equal and every successive pair of nodes is connected by an edge. This leads to the definition of balanced graphs:

Definition 2.1.1 (balanced graph [26])**.** *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ is said to be balanced if for every cycle the number of edges with negative weights is even.*

With this definition, a balanced graph has exactly two clusters, which means that edges between nodes in the same cluster have positive weights and edges between nodes in different clusters have negative weights. This rule for positive weights and negative weights can be defined for an arbitrary $k \in \mathbb{N}$ as follows:

Definition 2.1.2 (k-clustering [27])**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ be a signed graph and $k \in \mathbb{N}$. A k-clustering of $\mathcal{G}$ is a partitioning $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ of $\mathcal{V}$ with*

- $i, j \in \mathcal{V}_m \Rightarrow W_{ij} \geq 0$,
- $i \in \mathcal{V}_m \wedge j \in \mathcal{V}_n \wedge m \neq n \Rightarrow W_{ij} \leq 0$.

An example of a balanced graph with a cycle that traverses four edges with negative weights is shown in Fig. 2.1

As it is very restrictive to allow only two clusters, [27] extended this definition by simply allowing zero or at least two negative weights along a cycle.

Definition 2.1.3 (weakly balanced graph [27])**.** *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ is said to be weakly balanced if there is no cycle that traverses exactly one edge with negative weight.*

A cycle in a weakly balanced graph is then allowed to stay either within a cluster, thus, having only positive weights along its edges, or visit arbitrary many clusters that are different to the one where it started, for which there must be at least two negative weights. Namely, the edge where the

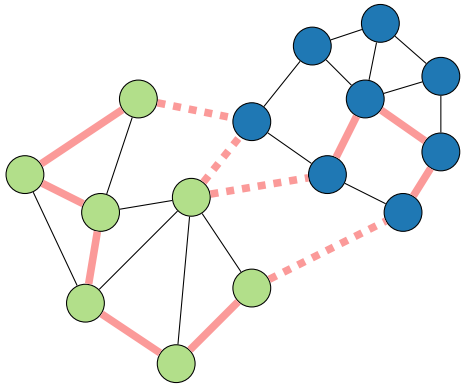


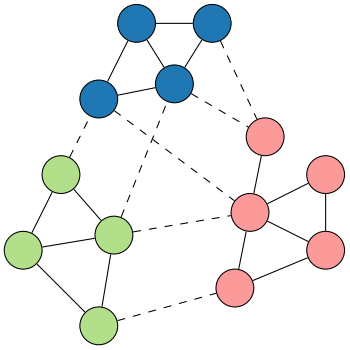
Figure 2.1: In this example of a balanced graph, the two clusters are represented by green and blue, solid lines have a positive weight whereas dashed lines have a negative weight. The thick lines show a circle that visits every cluster and traverses in total four edges with negative weight.

cycle leaves the cluster of the starting node and where it enters again. The following theorem shows the existence of k-clusterings in weakly balanced graphs.

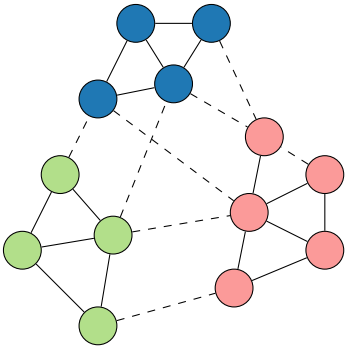
Theorem 2.1.1 (clustering of balanced graphs [27])**.** *For a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ *a* k*-clustering for some* $k \in \mathbb{N}$ *exists if and only if* $\mathcal{G}$ *is weakly balanced.*

Proof. See [27]. □

To illustrate this result, Fig. 2.2 shows an example of a weakly balanced graph with three clusters and how one additional edge can destroy the property of being weakly balanced.



(a) A weakly balanced graph with three clusters.



(b) The same graph with one additional negative edge. This edge destroys the weak balance.

Figure 2.2: An example of a weakly balanced graph compared to a graph that is not weakly balanced. The clusters are indicated by different colors, solid lines represent edges with positive weights and dashed lines represent edges with negative weights.

2.2 Graph Signals

In conventional signal processing, signals are defined over a time-domain which can be, for example, $\mathbb{R}$, $\mathbb{Z}$, intervals in $\mathbb{R}$ or $\mathbb{Z}$, or any cartesian product thereof. Typically those signals are referred to as continuous-time signal or discrete-time signal. The subject of graph signal processing are signals for which the time-domain is extended to general graphs.

Definition 2.2.1 (graph signal)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ be a graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. A graph signal over this graph is then a function*

$$\mathbf{x}(\cdot) : \mathcal{V} \to \mathbb{R}^L$$

that assigns a vector of dimension L to every graph vertex.

As the number N of vertices is finite, the number of vector values is also finite and so the graph signal can be represented by a vector $\boldsymbol{x}_i$ with

$$\boldsymbol{x}_i = \mathbf{x}(i) \tag{2.1}$$

for every $i \in \mathcal{V}$. Often it is useful to represent the complete graph signal in terms of the matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times L}. \tag{2.2}$$

For $L = 1$ we write $\boldsymbol{x}$ instead of $\boldsymbol{X}$.

2.3 Graph Laplacian

In order to measure the smoothness of a graph signal, [1] defined the edge derivative of a signal $\boldsymbol{x}$ with respect to the edge (i, j) at node i for an unsigned graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ as

$$\left.\frac{\partial \boldsymbol{x}}{\partial(i,j)}\right|_i = \sqrt{W_{ij}}(x_i - x_j), \tag{2.3}$$

which further defines the graph gradient[1] at node i as

$$\nabla_i \boldsymbol{x} = \left[\left.\frac{\partial \boldsymbol{x}}{\partial(i,1)}\right|_i, \ldots, \left.\frac{\partial \boldsymbol{x}}{\partial(i,N)}\right|_i\right] \tag{2.4}$$

and the local variation $\|\nabla_i \boldsymbol{x}\|_2$ via

$$\|\nabla_i \boldsymbol{x}\|_2^2 = \sum_{j\in\mathcal{V}}\left(\left.\frac{\partial \boldsymbol{x}}{\partial(i,1)}\right|_i\right)^2 = \sum_{j\in\mathcal{V}} W_{ij}(x_i - x_j)^2. \tag{2.5}$$

As a measure of global smoothness [1] also defined the discrete p-Dirichlet form of $\boldsymbol{x}$ as

$$S_p(\boldsymbol{x}) = \frac{1}{p}\sum_{i\in\mathcal{V}} \|\nabla_i \boldsymbol{x}\|_2^p. \tag{2.6}$$

In the case of $p = 2$, for a symmetric $\boldsymbol{W}$, this is

$$\begin{aligned}
S_2(\boldsymbol{x}) &= \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} W_{ij}(x_i - x_j)^2 \\
&= \sum_{i,j\in\mathcal{V}} \frac{1}{2}\big(W_{ij}x_i^2 + W_{ji}x_j^2\big) - W_{ij}x_i x_j \\
&= \sum_{i,j\in\mathcal{V}} W_{ij}\big(x_i^2 - x_i x_j\big) \\
&= \sum_{i\in\mathcal{V}}\left(x_i^2 \sum_{j\in\mathcal{V}} W_{ij} - \sum_{j\in\mathcal{V}} W_{ij}x_i x_j\right) \\
&= \sum_{i,j\in\mathcal{V}} x_i(D_{ij} - W_{ij})x_j \\
&= \boldsymbol{x}^T(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{x},
\end{aligned} \tag{2.7}$$

[1] Note that [1] defines this vector only for existing edges, but as the weight of a non-existent edge is zero, the corresponding entry in the gradient vector is also zero and thus it has no influence on later calculations.

with the diagonal degree matrix $\boldsymbol{D}$, $D_{ii} = \sum_{j \in \mathcal{V}} W_{ij}$ [6, 5]. The matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ is called the graph Laplacian. The name Laplacian was chosen, because for a long time it was believed that it converges to a continuous Laplacian operator if the number of nodes goes to infinity, until finally [28] provided a proof for that.

The most interesting property of the graph Laplacian is that it is positive semi-definite. This can easily be seen because $\boldsymbol{L}$ is hermitian, has only non-negative diagonal entries, and is diagonal dominant. Furthermore, a constant vector corresponds to the eigenvalue 0 because

$$(\boldsymbol{L}\mathbf{1})_i = (\boldsymbol{D}\mathbf{1} - \boldsymbol{W}\mathbf{1})_i = D_{ii} - \sum_{j \in \mathcal{V}} W_{ij} = 0, \tag{2.8}$$

for all i, $1 \leq i \leq N$.

In some applications the graph Laplacian is used with different normalizations, namely the symmetrically normalized graph Laplacian $\boldsymbol{L}^{(sym)} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$ [18] and the random walk Laplacian $\boldsymbol{L}^{(rw)} = \boldsymbol{D}^{-1}\boldsymbol{L}$ [29]. Both normalizations have only ones on their diagonal but there is still a difference in the remaining parts. The symmetrically normalized Laplacian has off-diagonal entries

$$L_{ij}^{(sym)} = -\frac{W_{ij}}{\sqrt{D_{ii}D_{jj}}}, \tag{2.9}$$

which shows that it is symmetric, and the random walk Laplacian has off-diagonal entries

$$L_{ij}^{(rw)} = -\frac{W_{ij}}{D_{ii}}, \tag{2.10}$$

which are the negative transition probabilities from node i to j in a random walk.

2.3.1 Signed Graph Laplacian

Previously, the graph Laplacian was only defined for unsigned graphs, but in Chapter 4 also algorithms that make use of negative edges are presented. Thus, the graph Laplacian needs to be extended to signed graphs. One possible way to do this is given in [30], where the authors introduced the signed graph Laplacian by means of graph drawing with antipodal proximity. They extend the idea of Laplacian eigenmaps [31] to negative weights by imposing that for a neighbor that is connected with a negative weight, the opposite coordinate should be used when computing the mean of the neighbors. By doing that they end up with the signed degree matrix

$$\bar{D}_{ii} = \sum_{j \in \mathcal{V}} |W_{ij}| \tag{2.11}$$

and the signed Laplacian

$$\bar{\boldsymbol{L}} = \bar{\boldsymbol{D}} - \boldsymbol{W}. \tag{2.12}$$

With the same argument as for the Laplacian, the signed Laplacian can be shown to be positive semi-definite and, according to [30], it is even positive-definite if the graph is unbalanced. Also the quadratic form has a similar form, when written as a sum

$$\begin{aligned}
\boldsymbol{x}^T \bar{\boldsymbol{L}} \boldsymbol{x} &= \sum_{i,j \in \mathcal{V}} x_i \bar{L}_{ij} x_j \\
&= \sum_{i,j \in \mathcal{V}} x_i^2 |W_{ij}| - x_i W_{ij} x_j \\
&= \sum_{i,j \in \mathcal{V}} |W_{ij}| (x_i^2 - \operatorname{sign}(W_{ij}) x_i x_j) \\
&= \frac{1}{2} \sum_{i,j \in \mathcal{V}} |W_{ij}| (x_i^2 - \operatorname{sign}(W_{ij}) x_i x_j + (\operatorname{sign}(W_{ij}) x_j)^2) \\
&= \frac{1}{2} \sum_{i,j \in \mathcal{V}} |W_{ij}| (x_i - \operatorname{sign}(W_{ij}) x_j)^2,
\end{aligned} \tag{2.13}$$

which will be useful in Section 2.7.1.

2.4 Total Variation

Another outcome for a global measure of smoothness from (2.6) with $p = 1$ is the TV [1]

$$\mathrm{TV}(\boldsymbol{x}) = S_1(\boldsymbol{x}) = \sum_{i \in \mathcal{V}} \|\nabla_i \boldsymbol{x}\|_2 = \sum_{i \in \mathcal{V}} \sqrt{\sum_{j \in \mathcal{V}} W_{ij} (x_i - x_j)^2}. \tag{2.14}$$

In image processing, this is known as the isotropic TV [32], which can be seen as being l_2 based. Another widely used modification thereof is the anisotropic TV [32, 8], which is l_1 based

$$\mathrm{TV}_{l_1}(\boldsymbol{x}) = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} W_{ij} |x_i - x_j|. \tag{2.15}$$

This measure has already been used in image processing in 1992 by [33] for denoising purposes, but there was no link to GSP. This link can be established by providing one graph node for every pixel in the image and then connecting every node to its four direct neighbors in the interior of the image, and to three, respectively, two neighbors at the boundary.

2.5 Graph Fourier Transform

The most popular tool of signal processing is the Fourier transform, and for real implementations especially the discrete Fourier transform (DFT). The aim of such a transformation is to represent a given signal in a different basis, which consists of orthogonal basis functions. In the Fourier transform, the basis functions are sinusoids which carry information about the frequency. For graphs, the concept of frequencies does not directly exist. The only way to get a similar concept is to measure how fast the signal changes. According to [1, 6] the eigenvectors of the graph Laplacian form a proper orthogonal basis which also shows this kind of "frequency". [1] also mentions that with increasing eigenvalue, the corresponding eigenvectors exhibit more rapid changes. This is illustrated in Fig. 2.3 for the second to the fifth eigenvector of the graph Laplacian. The first eigenvector is skipped, because it is constant and thus, it is not that informative anyway. It can be seen that in the second eigenvector there is one transition from negative values to positive values over the whole graph, whereas the fifth eigenvector already shows one transition in the upper left part of the graph.

Given the eigendecomposition

$$\boldsymbol{L} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T, \tag{2.16}$$

the GFT of an L-dimensional graph signal $\boldsymbol{X} \in \mathbb{R}^{N\times L}$ is given by

$$\hat{\boldsymbol{X}} = \boldsymbol{V}^T\boldsymbol{X} \tag{2.17}$$

and the inverse GFT is given by

$$\boldsymbol{X} = \boldsymbol{V}\hat{\boldsymbol{X}}. \tag{2.18}$$

A special case of the GFT is obtained for the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with

$$\mathcal{E} = \{\{N, 1\}, \{i, i+1\} : 1 \leq i < N\} \tag{2.19}$$

and $\boldsymbol{W} \in \{0, 1\}^{N\times N}$. This graph has a circulant weight matrix and Laplacian and thus the eigenvector matrix $\boldsymbol{V}$ corresponds to the DFT matrix. This means, that for a signal on this graph the GFT is equivalent to the DFT. Indeed, this reasoning can be extended to any cyclic graph.

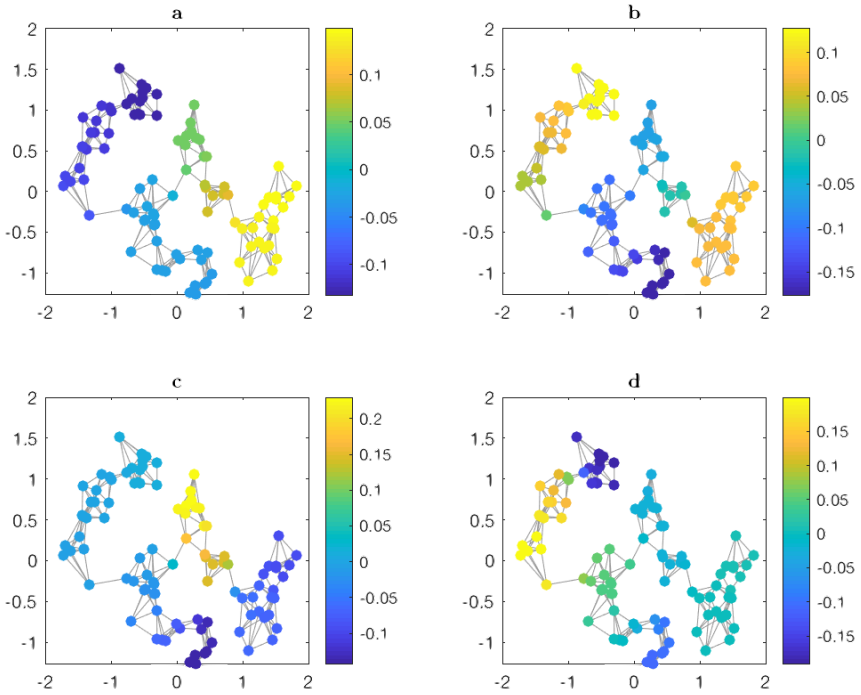


Figure 2.3: Second (a) to fifth (d) eigenvector of the graph Laplacian. The graph was generated from the two-dimensional data points as a KNN with five neighbors and Gaussian similarity with $\sigma_x = 1$ (cf. Chapter 3). The eigenvectors are used as graph signal and their amplitude is displayed by colors.

2.6 Graph Cut

In a graph with clusters it is typically the case that nodes in the same cluster are highly connected whereas there are only few edges between distinct clusters. Therefore it is desired to find a partitioning of the graph that results in a small number of cut edges. The graph cut is a measure for this number of cut edges for a weighted graph.

Definition 2.6.1 (graph cut)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ be a weighted graph with vertex set $\mathcal{V}$, edge set $\mathcal{E}$, and weight matrix $\boldsymbol{W}$. The graph cut for a partitioning $\{\mathcal{V}_1, \mathcal{V}_2\}$ of $\mathcal{V}$ is defined as*

$$\gamma(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1} \sum_{j \in \mathcal{V}_2} W_{ij}.$$

The minimum value of the graph cut is obtained when using $(\mathcal{V}_1, \mathcal{V}_2) = (\mathcal{V}, \emptyset)$ or $(\mathcal{V}_1, \mathcal{V}_2) = (\emptyset, \mathcal{V})$, in those cases one of the sums is over the empty set and so the resulting value is zero. This shows that it is not recommended to use the graph cut as objective for a minimization problem without any additional constraints.

Nevertheless, an interesting property of the graph cut can be obtained for the vector $\boldsymbol{x}$ with

$$x_i = \begin{cases} a, & i \in \mathcal{V}_1, \\ -b, & i \in \mathcal{V}_2. \end{cases} \tag{2.20}$$

This vector encodes the cluster label of node i into the sign of x_i and from (2.7) it can be seen that for this vector the quadratic form $\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x}$ is equal to the graph cut, up to a multiplicative constant:

$$\begin{aligned} \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} =& \frac{1}{2} \sum_{i,j \in \mathcal{V}} W_{ij} (x_i - x_j)^2 \\ =& \frac{1}{2} \sum_{i,j \in \mathcal{V}_1} W_{ij} (a - a)^2 \\ &+ \frac{1}{2} \sum_{i,j \in \mathcal{V}_2} W_{ij} (b - b)^2 \\ &+ \sum_{i \in \mathcal{V}_1} \sum_{j \in \mathcal{V}_2} W_{ij} (a + b)^2 \\ =& (a + b)^2 \sum_{i \in \mathcal{V}_1} \sum_{j \in \mathcal{V}_2} W_{ij} \\ =& (a + b)^2 \gamma(\mathcal{V}_1, \mathcal{V}_2). \end{aligned} \tag{2.21}$$

2.6.1 Signed Cut

For signed graphs it is necessary to split the definition of the graph cut because otherwise negative weights could compensate positive weights. The sum of all positive edge weights in a cut will be denoted by

$$\gamma^+(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1} \sum_{j \in \mathcal{V}_2} \max\{W_{ij}, 0\} \tag{2.22}$$

and the sum of all intra-set edges with negative weights by

$$\gamma^-(\mathcal{V}_1) = -\sum_{i \in \mathcal{V}_1} \sum_{j \in \mathcal{V}_1} \min\{W_{ij}, 0\}. \tag{2.23}$$

When minimizing those sums jointly it is favored to cut edges with negative weights (thus, they do not appear within a set of the partitioning) and leave

positive weights untouched (thus, they will not appear in the sum of positive cuts).

2.7 Ratio Cut

The ratio cut (RC) was first proposed in [34] where it was used to find clusters in VLSI designs which could be modeled in form of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with vertex set $\mathcal{V}$, edge set $\mathcal{E}$, and weight matrix $\boldsymbol{W}$. The idea of the RC was that balanced graph cuts should be favored in the minimization. This was achieved by normalizing the graph cut by the size of the partition sets. The RC was defined for a partitioning $\{\mathcal{V}_1, \mathcal{V}_2\}$ of $\mathcal{V}$ as

$$\rho_2(\mathcal{V}_1, \mathcal{V}_2) = \frac{\gamma(\mathcal{V}_1, \mathcal{V}_2)}{N_1 N_2}, \tag{2.24}$$

where it can be seen that the extreme cases of $(\mathcal{V}_1, \mathcal{V}_2) = (\mathcal{V}, \emptyset)$ or $(\mathcal{V}_1, \mathcal{V}_2) = (\emptyset, \mathcal{V})$ result in a division by zero and are thus no candidates for the minimum RC. A more recent approach is to define the RC for a k-partitioning (cf. [7]), so that it can be used for graphs with multiple clusters.

Definition 2.7.1 (ratio cut)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ denote a graph with vertex set $\mathcal{V}$, edge set $\mathcal{E}$ and weight matrix $\boldsymbol{W}$. The RC for a partitioning $\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k\}$ of $\mathcal{V}$ is defined as*

$$\rho(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) = \sum_{i=1}^{k} \frac{\gamma(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{N N_i}.$$

Note here that in the case of $k = 2$ the definitions of [7] and [34] are equivalent. To show that consider the partitioning $\mathcal{V}_1, \mathcal{V}_2$, then

$$\begin{aligned} N\rho(\mathcal{V}_1, \mathcal{V}_2) &= \frac{\gamma(\mathcal{V}_1, \mathcal{V} \setminus \mathcal{V}_1)}{N_1} + \frac{\gamma(\mathcal{V}_2, \mathcal{V} \setminus \mathcal{V}_2)}{N_2} \\ &= \frac{\gamma(\mathcal{V}_1, \mathcal{V}_2)}{N_1} + \frac{\gamma(\mathcal{V}_2, \mathcal{V}_1)}{N_2} \\ &= \frac{N_2\gamma(\mathcal{V}_1, \mathcal{V}_2) + N_1\gamma(\mathcal{V}_2, \mathcal{V}_1)}{N_1 N_2} \\ &= \frac{N_2\gamma(\mathcal{V}_1, \mathcal{V}_2) + N_1\gamma(\mathcal{V}_1, \mathcal{V}_2)}{N_1 N_2} \\ &= \frac{N\gamma(\mathcal{V}_1, \mathcal{V}_2)}{N_1 N_2} \\ &= N\rho_2(\mathcal{V}_1, \mathcal{V}_2). \end{aligned} \tag{2.25}$$

By normalizing (2.21) and properly choosing a and b depending on N_1 and N_2, the Rayleigh-quotient becomes $\frac{\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \propto \rho(\mathcal{V}_1, \mathcal{V}_2)$ for vectors that are defined according to (2.20). With (2.21) and $\boldsymbol{x}^T\boldsymbol{x} = \sum_{i\in\mathcal{V}} x_i^2 = N_1a^2 + N_2b^2$ we get

$$\frac{\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = \frac{(a+b)^2}{N_1a^2 + N_2b^2}\gamma(\mathcal{V}_1, \mathcal{V}_2). \tag{2.26}$$

One choice for a and b was proposed by [35]

$$a = \frac{N_1}{N}, \qquad b = \frac{N_2}{N} \tag{2.27}$$

and another one by [7]

$$a = \sqrt{\frac{N_2}{N_1}}, \qquad b = \sqrt{\frac{N_1}{N_2}}. \tag{2.28}$$

With (2.26) the Rayleigh-quotient is $\frac{\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = \rho(\mathcal{V}_1, \mathcal{V}_2)$ and with (2.27) it is $\frac{\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = N\rho(\mathcal{V}_1, \mathcal{V}_2)$. In the latter equation there is a multiplicative constant, but most of the time the exact value of the RC is not of interest as it is used as a objective function for minimization.

For the application of the RC in Section 4.2.1 it is required to derive an upper bound for the RC among all graphs.

Theorem 2.7.1 (range of RC)**.** *For any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with $\boldsymbol{W} \in [0, w_{\max}]^{N\times N}$, the RC of a k-partitioning is in the range $[0, (k-1)w_{\max}]$.*

Proof. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ be a graph with all weights being in the range $[0, w_{\max}]$ with $w_{\max} > 0$ and $\mathcal{V}_1, \ldots, \mathcal{V}_k$ be a partitioning of the vertex set. The RC cannot become negative as there are no negative elements in the sum. Indeed 0 is the smallest possible RC, which is obtained by the graph with no edges connecting distinct vertex sets of the partitioning. The maximum is obtained, when all inter-set edges exist. and all have weight $w_{\max}$. The RC is then given by

$$\begin{aligned}
\rho(\mathcal{V}_1, \ldots, \mathcal{V}_k) &= \sum_{i=1}^{k} \frac{\gamma(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{NN_i} \\
&= \frac{w_{\max}}{N} \sum_{i=1}^{k} \frac{N_i(N - N_i)}{N_i} \\
&= \frac{w_{\max}}{N} \sum_{i=1}^{k} (N - N_i) \\
&= w_{\max}(k-1).
\end{aligned} \tag{2.29}$$

□

2.7.1 Signed Ratio Cut

For signed graphs the convention is that positive edges connect similar nodes and negative edges connect dissimilar nodes. The magnitude indicates the level of similarity or dissimilarity, respectively. Directly minimizing the RC on signed graphs favors to cut negative edges and put nodes that are connected by positive edges into the same cluster. However, as there will be a tight relation between the sRC and the signed Laplacian, the sRC was defined by [30] as follows.

Definition 2.7.2 (Signed ratio cut)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ be a signed graph and $\mathcal{V}_1$, $\mathcal{V}_2$ be a partitioning of $\mathcal{V}$, then the sRC is defined as*

$$\bar{\rho}(\mathcal{V}_1, \mathcal{V}_2) = \frac{2\gamma^+(\mathcal{V}_1, \mathcal{V}_2) + \gamma^-(\mathcal{V}_1) + \gamma^-(\mathcal{V}_2)}{N_1 N_2}. \tag{2.30}$$

This definition penalizes cuts across edges with positive weights connecting $\mathcal{V}_1$ and $\mathcal{V}_2$ and across edges with negative weights within $\mathcal{V}_1$ and within $\mathcal{V}_2$. Thus, it becomes favorable to cut edges with negative weights. Similar to the relation of the RC and the graph Laplacian, there is a relation between the sRC and the signed graph Laplacian, i.e., for a vector

$$x_v = \begin{cases} a, & v \in \mathcal{V}_1, \\ -a, & v \in \mathcal{V}_2, \end{cases} \tag{2.31}$$

with $a \geq 0$, there is the relation

$$\begin{aligned} \boldsymbol{x}^T \bar{\boldsymbol{L}} \boldsymbol{x} =& \frac{1}{2} \sum_{i,j \in \mathcal{V}} |W_{ij}| (x_i - \operatorname{sign}(W_{ij}) x_j)^2 = \\ =& -\frac{1}{2} \sum_{i,j \in \mathcal{V}_1} \min\{0, W_{ij}\} 4a^2 \\ & -\frac{1}{2} \sum_{i,j \in \mathcal{V}_2} \min\{0, W_{ij}\} 4a^2 \\ & + \sum_{i \in \mathcal{V}_1} \sum_{j \in \mathcal{V}_2} \max\{0, W_{ij}\} 4a^2 \\ =& 2a^2 \big(2\gamma^+(\mathcal{V}_1, \mathcal{V}_2) + \gamma^-(\mathcal{V}_1) + \gamma^-(\mathcal{V}_2)\big) \\ =& 2a^2 N_1 N_2 \bar{\rho}(\mathcal{V}_1, \mathcal{V}_2). \end{aligned} \tag{2.32}$$

With the choice

$$a = \frac{1}{2} \left(\sqrt{\frac{N_1}{N_2}} + \sqrt{\frac{N_2}{N_1}} \right) \tag{2.33}$$

from [30], this becomes a relation between the quadratic form of the signed Laplacian and the sRC

$$\begin{aligned}\boldsymbol{x}^T\bar{\boldsymbol{L}}\boldsymbol{x} &= \frac{1}{2}\left(\frac{N_1}{N_2}+2+\frac{N_2}{N_1}\right)(2\gamma^+(\mathcal{V}_1,\mathcal{V}_2)+\gamma^-(\mathcal{V}_1)+\gamma^-(\mathcal{V}_2)) \\ &= \frac{N^2}{2N_1N_2}(2\gamma^+(\mathcal{V}_1,\mathcal{V}_2)+\gamma^-(\mathcal{V}_1)+\gamma^-(\mathcal{V}_2)) \\ &= \frac{1}{2}\bar{\rho}(\mathcal{V}_1,\mathcal{V}_2).\end{aligned} \tag{2.34}$$

Such a relation would not be possible when using the RC, because with negative weights, the RC can become negative but the signed graph Laplacian is positive (semi-)definite.

Next the minimum and maximum values of the sRC will be analyzed. Consider the partitioning $\{\mathcal{V}_1,\mathcal{V}_2\}$ of $\mathcal{V}$. The sRC is non-negative and so its minimum is attained for the graph without edges with $\bar{\rho}(\mathcal{V}_1,\mathcal{V}_2)=0$. For the upper bound first the case of an unsigned graph $\mathcal{G}=(\mathcal{V},\mathcal{E},\boldsymbol{W})$ with $\boldsymbol{W}\in[0,w^+_{\max}]^{N\times N}$ is considered. Again, the maximum is attained, when all inter-cluster edges exist. Then,

$$\begin{aligned}\bar{\rho}(\mathcal{V}_1,\mathcal{V}_2) &= \frac{2\gamma^+(\mathcal{V}_1,\mathcal{V}_2)+\gamma^-(\mathcal{V}_1)+\gamma^-(\mathcal{V}_2)}{N_1N_2} \\ &= \frac{2w_{\max}N_1N_2}{N_1N_2} \\ &= 2.\end{aligned} \tag{2.35}$$

For a signed graph with $\boldsymbol{W}\in[w^-_{\max},w^+_{\max}]^{N\times N}$ and $w^-_{\max}<0<w^+_{\max}$ the maximum is obtained for a partitioning $\mathcal{V}_1$, $\mathcal{V}_2$ when all inter-cluster edges exist and have weight $w^+_{\max}$ and all intra-cluster edges exist and have weight $w^-_{\max}$. For a graph without self-loops, this is

$$\begin{aligned}\bar{\rho}(\mathcal{V}_1,\mathcal{V}_2) &= \frac{2\gamma^+(\mathcal{V}_1,\mathcal{V}_2)+\gamma^-(\mathcal{V}_1)+\gamma^-(\mathcal{V}_2)}{N_1N_2} \\ &= \frac{2N_1N_2+N_1(N_1-1)+N_2(N_2-1)}{N_1N_2} \\ &= 2+\frac{N_1-1}{N_2}+\frac{N_2-1}{N_1},\end{aligned} \tag{2.36}$$

which is in the range from $4\big(1-\frac{1}{N}\big)$, obtained for $N_1=N_2=\frac{N}{2}$, to N, obtained for $N_1=N-1$.

3

Graph Learning

For the application of graph based clustering methods like SC it is essential to find a graph whose structure captures the similarity of the data. This means that the M-dimensional graph signal $\mathbf{x}(\cdot) : \mathcal{V} \to \mathbb{R}^M$ is required to fulfill some smoothness criterion on the graph like being globally smooth, bandlimited, or approximately bandlimited [10]. In practice, GSP techniques are applied on huge data sets for which it is likely that graph learning with such a constraint is computationally impractical. This leaves us with methods for which efficient algorithms exist. In this chapter, methods to measure similarity of data and two methods for graph learning will be explained in more detail, namely the ϵ-neighborhood graph and the KNN.

3.1 Similarity Measure

Additionally to the existence of edges, also the weight can be used to encode the similarity of nodes. In methods like [8], this is done automatically but for methods like the ϵ-neighborhood graph and KNN, similarity measures are required as a means to define the weights of the obtained graph. Those similarity measures are typically designed such that their value decreases when a certain distance increases. It is thus a prerequisite that the data is from a metric space.

A very intuitive way to fulfill the requirement of decreasing similarity with increasing distance is

$$s(\boldsymbol{x}, \boldsymbol{y}) = -d(\boldsymbol{x}, \boldsymbol{y})^2, \tag{3.1}$$

where $d(\cdot, \cdot)$ is the metric defined for the data. This convention was used in [36] with $d(\cdot, \cdot)$ being the Euclidean distance. The use of this similarity measure for graph based algorithms is problematic, as the highest possible similarity is 0, which usually is the weight of a non-existent edge, and basically there is no lower bound for the similarity. Thus, a non-existent edge would need a weight of $-\infty$ and so conventional linear algebra methods cannot be applied.

A more convenient approach to measure similarity is Gaussian similarity

$$s(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\frac{d(\boldsymbol{x}, \boldsymbol{y})^2}{2\sigma_s^2}}, \tag{3.2}$$

again with $d(\cdot, \cdot)$ being the metric defined for the data. This similarity measure has been widely used [18, 7, 37]. Compared to the first approach the image of this function is better suited for graph weights, as it is in the interval $[0, 1]$. Points that lie infinitely far apart are connected by an edge with weight 0, i.e., they are not connected in the graph, and the closer points lie together, the higher the weight will be.

In some applications it is not interesting how far points lie apart, but how big the difference in orientation is. A similarity measure that addresses this problem is cosine similarity

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \, \|\boldsymbol{y}\|}, \tag{3.3}$$

which has the prerequisite that the data must be from an inner product space. This similarity measure is often used when comparing text documents [38], where every document can be translated to vectors with each element denoting the frequency of a single word (or phrase) [39]. It is 1 if $\boldsymbol{x}$ and $\boldsymbol{y}$ are

perfectly aligned, 0 if orthogonal and -1 if they show in opposite directions. The latter property can be useful for clustering techniques, as it is a measure of dissimilarity.

There exist also more complicated approaches. E.g., [40] approximated areas with similar colors in images by several Gaussian distributions and represented the full image by a mixture of those Gaussians. The similarity of two images was then measured in terms of the Kullback-Leibler divergence. [41] measures similarity in music by calculating the spectrum of songs and comparing those spectra by means of the Earth Mover's Distance [42].

3.2 ϵ-Neighborhood Graph

Given a set of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{D}$, a similarity measure $s(\cdot, \cdot) : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$, and the minimum similarity $\epsilon \in \mathbb{R}$, the ϵ-neighborhood graph is defined as the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with vertices $\mathcal{V} = \{1, \ldots, N\}$, edge set $\mathcal{E}$ and weight matrix $\boldsymbol{W}$. The edge set is defined as

$$\mathcal{E} = \{\{i, j\} \subseteq \mathcal{V} : i \neq j \wedge s(\boldsymbol{x}_i, \boldsymbol{x}_j) > \epsilon\}, \tag{3.4}$$

which means that the edge $\{i, j\}$ exists if it is no self-loop and the similarity of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is larger than the prescribed minimum similarity ϵ. The weight matrix is then

$$W_{ij} = \begin{cases} s(\boldsymbol{x}_i, \boldsymbol{x}_j), & (i, j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

Here it should be noted that the graph need not be a weighted graph. For an unweighted graph the weight matrix is simply $\boldsymbol{W} \in \{0, 1\}^{N \times N}$ with $W_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and 0 otherwise. For this case it is not necessary to specify the similarity measure, instead one could replace the condition $s(\boldsymbol{x}_i, \boldsymbol{x}_j) > \epsilon$ in (3.4) by $d(\boldsymbol{x}_i, \boldsymbol{x}_j) < \epsilon$.

In both cases, the value for ϵ has to be chosen carefully, as it can be used to control the number of edges generated in the graph. For efficient GSP methods it is necessary that the weight matrix is sparse and so the number of edges must be small. In an attempt to keep the number of edges small, one could first calculate the distribution of similarity values and then set ϵ such that a certain number of edges exists. With this approach it is unlikely that outliers are connected to the graph.

3.3 k-nearest Neighborhood Graph

A simple way to come around the sparsity problem of the ϵ-neighborhood graph is the KNN. The KNN is a graph that is generated from data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{D}$ by connecting each point to its $k \in \mathbb{N}$ nearest neighbors. I.e., for the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with vertices $\mathcal{V} = \{1, \ldots, N\}$, the edge set is defined as

$$\mathcal{E} = \{\{i, j\} \subseteq \mathcal{V} : \exists l \in \mathbb{N} : 1 < l <= k + 1 \wedge j = p_l^{(i)}\}, \tag{3.6}$$

where $p_l^{(i)}$ is an index permutation such that

$$s\Big(\boldsymbol{x}_{p_l^{(i)}}, \boldsymbol{x}_i\Big) \geq s\Big(\boldsymbol{x}_{p_{l+1}^{(i)}}, \boldsymbol{x}_i\Big), \tag{3.7}$$

i.e., it orders the points with decreasing similarity to $\boldsymbol{x}_i$. $l = 1$ is excluded because $p_l^{(i)} = i$, which would generate a self-loop. The weight matrix for the KNN is defined the same way as for the ϵ-neighborhood graph and it is again possible to generate this graph in an unweighted manner by setting all weights to 1 and replacing (3.7) by

$$d\Big(\boldsymbol{x}_{p_l^{(i)}}, \boldsymbol{x}_i\Big) \leq d\Big(\boldsymbol{x}_{p_{l+1}^{(i)}}, \boldsymbol{x}_i\Big). \tag{3.8}$$

The number of edges generated for the KNN is always in the interval

$$|\mathcal{E}| = \left[\left\lfloor \frac{N}{k+1} \right\rfloor \frac{k(k+1)}{2}, kN\right]. \tag{3.9}$$

The upper bound of the interval corresponds to the case that for every node k edges are generated without reusing the edges generated for other nodes. The lower bound of the interval corresponds to reusing every edge.

In the literature there is another definition of the KNN, which is the mutual k-nearest neighborhood graph (mKNN) [7]. For this graph, the edge set is defined as

$$\mathcal{E} = \{\{i, j\} \subseteq \mathcal{V} : \exists l_1, l_2 \in \mathbb{N} : 1 < l_1, l_2 <= k+1 \wedge j = p_l^{(i)} \wedge i = p_l^{(j)}\}. \tag{3.10}$$

In this definition there is also the constraint $i = p_l^{(j)}$, which means that not only j has to be among the k nearest neighbors of i but also i has to be among the k nearest neighbors of j. This results in

$$|\mathcal{E}| \in \left[k, \left\lceil \frac{N}{k+1} \right\rceil \frac{k(k+1)}{2}\right]. \tag{3.11}$$

Compared to the ϵ-neighborhood graph, it is relatively easy to control the number of edges for the KNN without iterations by setting the value of k appropriately.

4

Clustering Algorithms

A central element of GSP and machine learning is the detection of clusters in data sets. The machine learning community differentiates between unsupervised clustering, where no labeled training data is available, and semi-supervised clustering, where labeled training data is partially available. In signal processing jargon this would correspond to clustering without sampling and clustering with sampling, respectively. In the GSP community, there has already been research on the recovery of signals from sampled data [9, 12, 43, 16] and also on the theoretical limits of signal recovery [10, 11]. In this chapter, the focus will be on the reconstruction of cluster labels by means of sampling.

Algorithm 1 k-means

input: data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, $t_{\max}$

1: initialize the cluster centers $\boldsymbol{c}_1^{(0)}, \ldots, \boldsymbol{c}_K^{(0)}$ randomly, $t = 0$
2: **repeat**
3: estimate clusters $\mathcal{X}_k^{(t+1)}$ from (4.1)
4: calculate new cluster centers $\boldsymbol{c}_k^{(t+1)}$ from (4.2)
5: $t \leftarrow t + 1$
6: **until** $\forall k \in \{1, \ldots, K\} : \boldsymbol{c}_k^{(t)} = \boldsymbol{c}_k^{(t-1)} \vee t > t_{\max}$

output: cluster labels $\boldsymbol{l}$ with $l_i = k$ if $\boldsymbol{x}_i \in \mathcal{X}_k^{(t+1)}$

4.1 Clustering without Sampling

4.1.1 k-Means

According to [44], k-means is a prototype method, which means that it attempts to find points $\boldsymbol{c}_k$ in the domain of the data that represent the centers of each cluster. These cluster centers are referred to as prototypes. The input data to k-means is a set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^L$. The algorithm starts with random prototypes $\boldsymbol{c}_k^{(0)}$ and then iteratively estimates the clusters by assigning every node to the cluster of the closest prototype

$$\mathcal{X}_k^{(t+1)} = \left\{ \boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} : d\left(\boldsymbol{x}, \boldsymbol{c}_k^{(t)}\right) < d\left(\boldsymbol{x}, \boldsymbol{c}_l^{(t)}\right) \text{ for all } l \neq k \right\} \quad (4.1)$$

for all k and recalculates the prototypes as the mean of all data points that are assigned to this prototype, i.e.,

$$\boldsymbol{c}_k^{(t+1)} = \frac{1}{\left|\mathcal{X}_k^{(t+1)}\right|} \sum_{\boldsymbol{x} \in \mathcal{X}_k^{(t+1)}} \boldsymbol{x}. \quad (4.2)$$

The overall scheme is summarized in Algorithm 1.

4.1.2 Multidimensional Scaling

Classical multidimensional scaling (MDS) is a method for multivariate data analysis that transforms a set of N data points $y_i \in \mathcal{M}$ from a metric space $(\mathcal{M}, d(\cdot, \cdot))$ to the K-dimensional Euclidean space. The algorithm of MDS depends on the metric $d(\cdot, \cdot)$ and is stated in [45] as listed in Algorithm 2. It first calculates the pairwise distances in the original metric space and puts them together in the matrix $\boldsymbol{P}^{(2)}$ of squared proximities, then it subtracts the row- and column-mean and adds the mean over the whole matrix

Algorithm 2 multidimensional scaling (MDS)

input: data $y_1, \dots, y_N$, metric $d(\cdot, \cdot)$ on the data

1: Set up the matrix of squared proximities $P_{ij}^{(2)} = d^2(y_i, y_j)$

2: Apply the double centering: $\boldsymbol{C} = -\frac{1}{2}\boldsymbol{J}\boldsymbol{P}^{(2)}\boldsymbol{J}$ (with $\boldsymbol{J} = \boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$)

3: Calculate the eigenvalue decomposition $\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$

4: Let $\boldsymbol{X}^T$ be the first K columns of $\boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}}$

output: embedded coordinates $\boldsymbol{X}$

to get the double centered matrix $\boldsymbol{C}$. Finally, the Euclidean coordinates $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in$ are obtained as the rows of the first K eigenvectors weighted with the square root of the corresponding eigenvalues. In the end the matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$ holds all Euclidean coordinates.

By calculating the pairwise distances in the Euclidean space it can be seen that they are equal to the distance in the original metric space if $K = N$. With $\boldsymbol{x}_i$ being the i-th row of $\boldsymbol{V}\boldsymbol{\Lambda}$ we get that $\boldsymbol{x}_i^T\boldsymbol{x}_j$ is the ij-th entry of $\boldsymbol{C}$ and so

$$\begin{aligned}
\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 =& \boldsymbol{x}_i^T\boldsymbol{x}_i + \boldsymbol{x}_j^T\boldsymbol{x}_j - 2\boldsymbol{x}_i^T\boldsymbol{x}_j \\
=& C_{ii} + C_{jj} - 2C_{ij} \\
=& -\frac{1}{2}\boldsymbol{J}_{i:}\boldsymbol{P}^{(2)}\boldsymbol{J}_{:i} - \frac{1}{2}\boldsymbol{J}_{j:}\boldsymbol{P}^{(2)}\boldsymbol{J}_{:j} + \boldsymbol{J}_{i:}\boldsymbol{P}^{(2)}\boldsymbol{J}_{:j} \\
=& -\frac{1}{2}\left(P_{ii}^{(2)} - \frac{2}{N}\boldsymbol{P}_{i:}^{(2)}\mathbf{1} + \frac{1}{N^2}\mathbf{1}^T\boldsymbol{P}^{(2)}\mathbf{1}\right) \\
& -\frac{1}{2}\left(P_{jj}^{(2)} - \frac{2}{N}\boldsymbol{P}_{j:}^{(2)}\mathbf{1} + \frac{1}{N^2}\mathbf{1}^T\boldsymbol{P}^{(2)}\mathbf{1}\right) \\
& +\left(P_{ij}^{(2)} - \frac{1}{N}\boldsymbol{P}_{i:}^{(2)}\mathbf{1} - \frac{1}{N}\boldsymbol{P}_{j:}^{(2)}\mathbf{1} + \frac{1}{N^2}\mathbf{1}^T\boldsymbol{P}^{(2)}\mathbf{1}\right) \\
=& P_{ij}^{(2)} = d\left(\boldsymbol{y}_i, \boldsymbol{y}_j\right)^2.
\end{aligned} \tag{4.3}$$

Another observation is that the matrix $\boldsymbol{J}$ has rank $N-1$ and so there exist at most $N-1$ linearly independent eigenvectors and so $K \leq N-1$ is sufficient. Applied to graphs, this means that if the pairwise distances on a graph define a metric, the respective graph can perfectly be embedded in the $(N-1)$-dimensional euclidean space such that the Euclidean distance is equal to the graph distance.

In step 4 of the algorithm the eigenvectors are weighted with the eigenvalues, which means that the respective entries in $\boldsymbol{X}$ are dominant. A heuristic approach would be to only use the eigenvectors corresponding to the largest

eigenvalues, as this keeps the error in the distance small. As the distances between points in different clusters are high, they will be separated in the embedded coordinates and thus, the clusters should be visible in $\boldsymbol{X}$ and detectable with k-means.

4.1.3 Spectral Clustering

In the case that clusters cannot be separated into convex subsets of the underlying space it is not possible to get a good performance with k-means [44]. A very popular method that can also work for non-convex clusters is SC. SC is a graph clustering algorithm, that in contrast to k-means cannot be used directly for the clustering of a set of data-points. It has to be preceded by a graph learning algorithm (cf. Chapter 3).

Derivation of the algorithm

The idea of SC is to minimize the RC of the graph, i.e., it aims to solve the NP-hard combinatorial optimization problem

$$\begin{aligned} \min_{\mathcal{V}_1,\dots\mathcal{V}_k} \quad & \rho(\mathcal{V}_1,\mathcal{V}_2,\dots,\mathcal{V}_k) \\ \text{s.t.} \quad & \bigcup_{i=1}^{k} \mathcal{V}_i = \mathcal{V}, \\ & \mathcal{V}_i \cap \mathcal{V}_j = \emptyset \text{ for } i \neq j. \end{aligned} \tag{4.4}$$

As will be seen later, a relaxed solution to this problem is given by the eigenvectors of the graph Laplacian. The space that is spanned by those eigenvectors is called the Laplacian spectrum of the graph [6]. That explains the name of this approach. The authors of [35] used the equivalence of the RC and the Rayleigh quotient and derived a lower bound on the RC which was given by the second smallest eigenvalue of the graph Laplacian. Based on that they used a so-called 'spectral heuristic' which is: If the eigenvalue is a lower bound on the RC, the corresponding eigenvector must be close to the optimum solution of $\boldsymbol{x}$. Indeed as it is mentioned in [7], the eigenvector corresponding to the second smallest eigenvalue is the solution to the relaxed problem

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^N} \quad & \boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} \\ \text{s.t.} \quad & \|x\|_2 = 1, \\ & \boldsymbol{x}^T\mathbf{1} = 0, \end{aligned} \tag{4.5}$$

Algorithm 3 spectral clustering (SC)

input: data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$

1: learn a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ from the data

2: $\boldsymbol{D} \leftarrow \text{diag}\Big(\sum_{j \in \mathcal{V}} W_{ij}\Big)$, $\boldsymbol{L} \leftarrow \boldsymbol{D} - \boldsymbol{W}$

3: $\boldsymbol{y} \leftarrow$ eigenvector corresponding to the second smallest eigenvalue of $\boldsymbol{L}$

4: $\boldsymbol{l} \leftarrow \text{sign}(\boldsymbol{y})$

output: $\boldsymbol{l}$

which follows from the Rayleigh-Ritz theorem. [7] also discusses the relaxation of (4.4) for $k \geq 2$

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{R}^{N \times k}} \quad & \text{Tr}\big(\boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X}\big), \\ \text{s.t.} \quad & \boldsymbol{X}^T \boldsymbol{X} = \boldsymbol{I}. \end{aligned} \tag{4.6}$$

The solution is again given by the Rayleigh-Ritz theorem, i.e., $\boldsymbol{X}$ is the matrix containing the k eigenvectors corresponding to the k smallest eigenvalues of $\boldsymbol{L}$.

The solutions of those relaxed problems do not yet give the solution to the cluster assignment, they first have to be mapped to discrete cluster labels. The way this problem is treated in SC in the two-dimensional space is that the signs of the resulting vector $\boldsymbol{x}$ are used as cluster labels, whereas in the case of k-clusters, the rows of $\boldsymbol{X}$ are interpreted as coordinates in $\mathbb{R}^k$ and then k-means is run on this set of points to find a clustering. The overall method is listed in Algorithm 3

In [18] the authors derived a performance bound for SC. Their result shows that under certain assumptions the coordinates defined by the eigenvectors corresponding to the smallest k eigenvalues (the zero eigenvalue is omitted) are centered around k orthogonal vectors $\boldsymbol{r}_1, ..., \boldsymbol{r}_k$ that can thus be interpreted as cluster centers. This result shows that the transformation of the graph to its spectrum yields a space in which the clusters are well separated.

Extensions to signed graphs

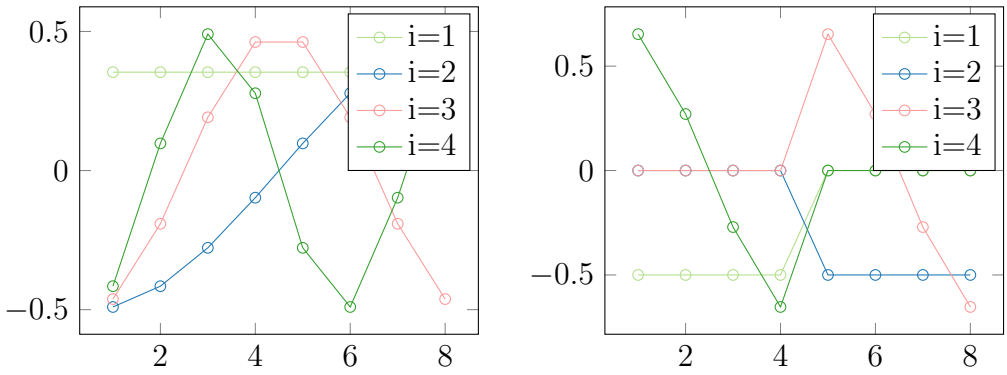
In [30, 46] SC was analyzed for graphs with negative edges. Such graphs exist for example in social networks, where friends are connected by edges with positive weights and foes by edges with negative weights. [46] explained the effects of negative weights on the eigenvectors of the signed Laplacian for the example of a spring-mass system, in which every mass is modeled by

a node and the springs are modeled by edges, i.e., it is modeled by a linear graph. Fig. 4.1a shows the first four eigenvectors of the signed Laplacian of such a linear graph with all weights being equal to 1. It can be seen that the eigenvectors represent different frequencies for the movement of the masses and that all masses try to move close to its adjacent masses. Fig. 4.1b shows the first four eigenvectors in the case that the system is disconnected at the fourth link. Indeed it can be seen from those eigenvectors that the two subsystems can move independently, as every eigenvector is zero on one subsystem and non-zero on the other, and that the first non-zero frequency corresponds to the second non-zero frequency of the system with 8 nodes. In Fig. 4.1c again the system consists of two subsystems of masses and springs with all edges having weight 1, but this time the subsystems are connected by an edge with weight -1. For every frequency in this system, there is a corresponding frequency in the system with all weights being 1 (Fig. 4.1a) but the first half of nodes is mirrored about 0.

In the examples with two subgraphs that are either disconnected or connected by an edge with negative weight, the effects of clusters on the eigenvectors can already be seen. For disconnected clusters it is already possible that the eigenvector corresponding to the eigenvalue zero is non-constant, but this is a special case that usually does not exist in reality. Opposite to that, with negative edge weights, the constant vector is never in the spectrum of the signed Laplacian and so it is always sufficient to compute the first eigenvector.

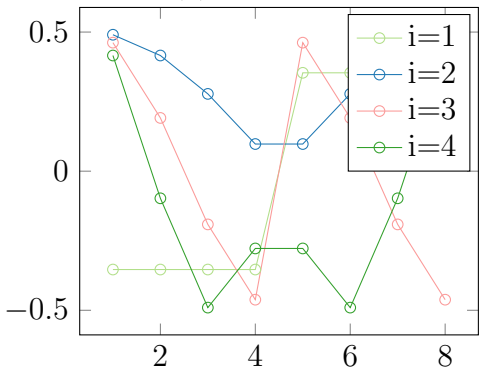
The application of SC to the signed Laplacian is denoted by sSC. With (2.34) it can be seen that sSC is a relaxed version of the minimization of the sRC. The sSC scheme is listed in Algorithm 4.

The extension of sSC to multiple clusters is not as straightforward as it was for SC. Indeed, [25] showed that there is no encoding of the cluster labels in a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times K}$ that minimizes the sRC. A way to solve this issue is weighted kernel k-means.



(a) $W_{45} = 1$.

(b) $W_{45} = 0$.



(c) $W_{45} = -1$.

Figure 4.1: First four eigenvectors of the signed Laplacian of a linear graph with 8 nodes and $W_{i,i+1} = 1$ if $i \neq 4$.

Algorithm 4 signed spectral clustering (sSC)

input: signed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$

1: $\bar{\boldsymbol{D}} \leftarrow \text{diag}\left(\sum_{j \in \mathcal{V}} |W_{ij}|\right)$, $\bar{\boldsymbol{L}} \leftarrow \bar{\boldsymbol{D}} - \boldsymbol{W}$
2: **if** at least one negative edge **then**
3: $\quad \boldsymbol{y} \leftarrow$ eigenvector corresponding to the smallest eigenvalue of $\bar{\boldsymbol{L}}$
4: **else**
5: $\quad \boldsymbol{y} \leftarrow$ eigenvector corresponding to the second smallest eigenvalue of $\bar{\boldsymbol{L}}$
6: **end if**
7: $\boldsymbol{l} \leftarrow \text{sign}(\boldsymbol{y})$

output: $\boldsymbol{l}$

4.2 Clustering with Sampling

In some scenarios it is possible to get access to the cluster labels of some nodes in the graph. E.g., in semi-supervised learning training data for which the cluster labels are known is provided to the algorithms. For real-life scenarios, like in sociological studies, it is also possible to provide sampled data by doing surveys. By providing this additional information (i.e., a subset of the cluster labels) it can be expected that the performance of clustering algorithms that make use of this information is better than the performance of algorithms that do not use the information. Existing algorithms that make use of label samples are [19, 23, 21, 20].:

- [19] is an extension of SC that modifies the weight matrix based on the similarity of the sampled data.
- [23] adds additional constraints to the minimization of the RC, which provides candidate solutions for clustering, from which the solution with minimum RC is selected.
- [20] models the cluster labels by a graph signal $\boldsymbol{x}$ which has to be a harmonic function. The unknown labels are then estimated by $\boldsymbol{Lx} = \mathbf{0}$ with the constraint that the known labels have to appear in $\boldsymbol{x}$.
- Contrary to the other approaches [21] samples the weight of the edges and it is based on the concept of balanced graphs. For those the weight matrix has low rank. The unknown edges are reconstructed by minimizing the rank of a matrix with the constraint that the known edges have to be the same.

In those algorithms, there are two types of sampling strategies: edge sampling and node sampling. Edge sampling means, that an edge $\{i, j\}$ is picked randomly and the information that is returned from sampling is either $+1$ or -1 depending on i and j being in the same cluster or in different clusters, respectively. For node sampling, sampling M nodes means that a set $\mathcal{S}$ of M distinct graph nodes is selected and every edge in $\mathcal{S} \times \mathcal{S}$ is sampled. As the elements (i, j), with $i, j \in \mathcal{S}$ and $i = j$, do not provide any information, there are $\frac{M(M-1)}{2}$ edges that could provide information. To get a comparison of edge sampling and node sampling, the numbers of different pairs in edge-sampling is always $\frac{M(M-1)}{2}$.

4.2.1 Spectral Clustering with Sampling

The RC in SC penalizes the cut of edges with large positive weights and favors to cut edges with small weights. The idea for the extension of SC in

[19] is that between all sampled nodes edges will be added and their weight will be set to a high value if they lie in the same cluster and to a low value if they lie in different clusters. At the beginning the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ is built from the data as described in Section 4.1.3. The cluster labels will be denoted by $\boldsymbol{l}$ and the sampling set is defined as $\mathcal{S} = \{s_1, s_2, .., s_M\} \subseteq \mathcal{V}$ and $\mathcal{S}_1$ and $\mathcal{S}_2$ are the set of nodes sampled from the first and the second cluster, respectively. We have $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$.

For every finite graph there exists a maximum weight $w_{max} > 0$ such that $W_{i,j} \in [0, w_{max}]$. In the following it will be assumed that $w_{max} = 1$, which can always be obtained by rescaling the weights. Such a rescaling does not change the result of SC as it only scales the eigenvalues but not the corresponding eigenvectors and also it does not change the order of eigenvalues. It is now possible to modify the weights according to the similarity of the samples or by their dissimilarity. In the work of [19] the weights between similar nodes were set to a high value and between dissimilar nodes it was set to zero. This can be done by duplicating $\boldsymbol{W}$ and modifying the edges between sampled nodes as

$$\widetilde{W}_{ij} = \begin{cases} W_{ij}, & i, j \notin \mathcal{S}, \\ w_{sim}, & i, j \in \mathcal{S}_1 \text{ or } i, j \in \mathcal{S}_2, \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

The resulting similarity graph $\widetilde{\mathcal{G}} = \left(\mathcal{V}, \widetilde{\mathcal{E}}, \widetilde{\boldsymbol{W}}\right)$ is a graph with non-negative edges and can be used as input to SC. It can already be expected that for large w_{sim} SC will yield a better clustering for $\mathcal{G}_{sim}$ than for $\mathcal{G}$. This has three reasons

- New edges are generated;
- edges between dissimilar nodes are deleted;
- high weights make cutting more expensive.

Consider the case of two clusters shown in Fig. 4.2. In this graph SCs would not modify any edges and clearly the minimum cut is obtained when the graph is cut by a vertical line in the middle. As the size of the resulting clusters is equal this is also the minimum RC. This shows that the two sampled nodes would get the same label. Consequently, correct labeling of the sampled nodes can not be enforced via this approach.

As a means to penalize this false labeling of sampled nodes we proposed

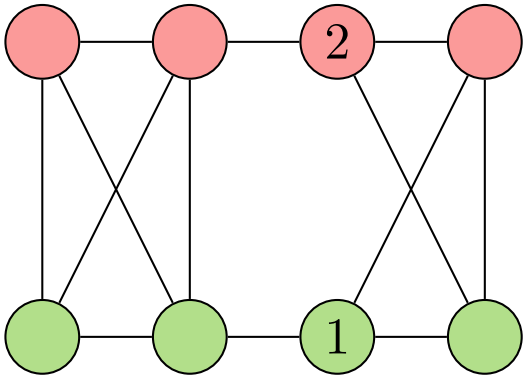


Figure 4.2: An example of a graph for which spectral clustering with sampling (SCs) would result in falsely labeled sampled nodes. The colors indicate the original clusters and the nodes with numbers indicate samples with their corresponding label. The weights of the edges are all one.

Algorithm 5 signed spectral clustering with sampling (sSCs)

input: data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, sampling set $\mathcal{S}$, w_{sim}, w_{dis}

1: learn the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ from the data
2: $\boldsymbol{W}_{sim}^{dis} \leftarrow \boldsymbol{W}$
3: modify the entries of $\boldsymbol{W}_{sim}^{dis}$ according to (4.8)
4: run sSC on $\widehat{\mathcal{G}} = \left(\mathcal{V}, \widehat{\mathcal{E}}, \widehat{\boldsymbol{W}}\right)$

output: cluster labels $\boldsymbol{l}$ resulting from sSC

to include negative weights for dissimilar nodes [24]. With the modification

$$\widehat{W}_{ij} = \begin{cases} W_{ij}, & i,j \notin \mathcal{S}, \\ w_{sim}, & i,j \in \mathcal{S}_1 \text{ or } i,j \in \mathcal{S}_2, \\ -w_{dis}, & \text{otherwise.} \end{cases} \tag{4.8}$$

with $w_{dis} > 0$ we can prevent SC from cutting those edges. Obviously, the resulting graph $\widehat{\mathcal{G}} = \left(\mathcal{V}, \widehat{\mathcal{E}}, \widehat{\boldsymbol{W}}\right)$ is a signed graph. Hence, we use sSC, the algorithm for sSCs herby obtained is listed in Algorithm 5. While it was still possible that all samples end up in the same cluster with $w_{dis} = 0$ without any penalty it is now favorable to separate samples from different clusters.

For this modification, a bound for w_{sim} and w_{dis} can be derived. This will be done in two steps, first Lemma 4.2.1 and Lemma 4.2.2 present an upper bound for the case that all samples have been labeled correctly and a lower bound for the case that at least one sampled node has been labeled falsely,

then, in Theorem 4.2.1, the two lemmas will be joined to give a criteria, when the choice of w_{sim} and w_{dis} yields consistent clustering results for the sampled nodes.

Lemma 4.2.1 (maximum sRC with correct labeling)**.** *For any graph the sRC with the modified weight matrix and all nodes being labeled correctly is*

$$\bar{\rho}(\mathcal{V}_1, \mathcal{V}_2) \leq 2.$$

Proof. Assuming that all nodes have been labeled correctly, there are no edges with negative weights within either $\mathcal{V}_1$ or $\mathcal{V}_2$. Thus, they do not contribute anything to the sRC and so it is equivalent to calculate the sRC for the unsigned graph, where all negative weights have been removed. According to (2.35) the bound $\bar{\rho}(\mathcal{V}_1, \mathcal{V}_2) \leq 2$ holds. □

Lemma 4.2.2 (minimum sRC with falsely labeled nodes)**.** *For any graph the sRC with the modified weight matrix and at least one falsely labeled sample is*

$$\bar{\rho}(\mathcal{V}_1, \mathcal{V}_2) \geq \min_{i \in \{1,2\}} \left\{ \frac{2w_{sim}(M_i - 1) + w_{dis}M_{3-i}}{\frac{N^2}{4}} \right\}$$

as long as

$$w_{dis} > \frac{2w_{sim}}{\max\{M_1, M_2\}}.$$

Proof. The minimum possible sRC among all graphs with at least one falsely labeled sample is obtained when there are no edges in the graph except those that were generated from sampling. The initial graph with no edges is sufficient for the minimum, as it has no edges that can increase the sRC and there is no graph that could result in a smaller sRC, although there are graphs that result in the same value, e.g., consider a graph with both clusters being disconnected.

A lower bound for the sRCcan be obtained by finding an upper bound for the denominator and a lower bound for the numerator. For the denominator we get with $\mathcal{V}_1 + \mathcal{V}_2 = \mathcal{V}$, $\frac{N^2}{4} \geq N_1 N_2$ and for the numerator we proceed as follows:

Let M_{ij} be the number of sampled nodes from cluster i that end up in cluster j after clustering. With $M_{ii} + M_{ij} = M_i$ the numerator of the sRC is $f(\cdot) : \mathbb{R}^2 \to \mathbb{R}$

$$\begin{aligned} f(M_{11}, M_{22}) =& 2\gamma^+(\mathcal{V}_1, \mathcal{V}_2) + \gamma^-(\mathcal{V}_1) + \gamma^-(\mathcal{V}_2) \\ =& 2w_{sim}(M_{11}M_{12} + M_{22}M_{21}) + w_{dis}M_{21}M_{11} + w_{dis}M_{12}M_{22} \\ =& 2w_{sim}(M_{11}(M_1 - M_{11}) + M_{22}(M_2 - M_{22})) \\ &+ w_{dis}M_{11}(M_2 - M_{22}) + w_{dis}M_{22}(M_1 - M_{11}). \end{aligned} \tag{4.9}$$

The minimum sRC with at least one falsely labeled node is obtained by minimizing $f(\cdot)$ on the set $\{0,\dots,M_1\}\times\{0,\dots,M_2\}\setminus\{(0,0)\cup(M_1,M_2)\}$. $(0,0)$ has to be excluded here because it corresponds to the case that all nodes are put in the wrong cluster, but this is again a correct clustering by exchanging the labels of the clusters. The cases of all nodes being placed in the same cluster can also be excluded, because in the case $M_1 \leq M_2$, $w_{dis} > \frac{2w_{sim}}{M_2}$ and so

$$\begin{aligned} f(0,M_2) = f(M_1,0) &= w_{dis}M_1M_2 \\ &= M_2w_{dis} + (M_1-1)M_2w_{dis} \\ &> M_2w_{dis} + 2(M_1-1)w_{sim} = f(M_1-1,M_2). \end{aligned} \tag{4.10}$$

For $M_2 \leq M_1$ the righthand side would be replaced by $f(M_1, M_2-1)$, which shows that the minimum can never be obtained if all sampled nodes are put into the same cluster.

With the constraint $0 < M_{11} < M_1$, the minimum for (4.9) is obtained as $2w_{sim}(M_1-1)+w_{dis}M_2$ at $M_{11} = M_1 - 1$ and $M_{22} = M_2$. Vice versa, for $0 < M_{22} < M_2$ the minimum is obtained as $2w_{sim}(M_2-1)+w_{dis}M_1$ at $M_{11} = M_1$ and $M_{22} = M_2 - 1$. To get the minimum on $\{0,\dots,M_1\}\times\{0,\dots,M_2\}\setminus\{(0,0)\cup(M_1,M_2)\cup(0,M_2)\cup(M_1,0)\}$ it is only left to take the minimum over those two cases. □

By imposing that a partitioning with all sampled nodes being labeled correctly has to be better in terms of sRC than one with at least one falsely labeled node, we get

Theorem 4.2.1 (consistent labeling with sRC)**.** *Minimizing the sRC yields consistent labels for the sampled nodes when*

$$\frac{N^2}{2} < \min_{i\in\{1,2\}}\{2w_{sim}(M_i-1)+w_{dis}M_{3-i}\}$$

and

$$w_{dis} > \min_{i\in\{1,2\}}\left\{\frac{2w_{sim}}{M_i}\right\}.$$

Proof. Let $\{\mathcal{V}_1^c,\mathcal{V}_2^c\}$ and $\{\mathcal{V}_1^f,\mathcal{V}_2^f\}$ denote the partitioning with all sampled nodes being correctly labeled and at least one node falsely labeled, respectively. From Lemma 4.2.2 we know $\bar{\rho}\left(\mathcal{V}_1^f,\mathcal{V}_2^f\right) \geq \min_{i\in\{1,2\}}\left\{\frac{2w_{sim}(M_i-1)+w_{dis}M_{3-i}}{\frac{N^2}{4}}\right\}$.

By choosing w_{sim} and w_{dis} such that $\frac{N^2}{2} \leq \min_{i\in\{1,2\}}\{2w_{sim}(M_i-1)+w_{dis}M_{3-i}\}$, Lemma 4.2.1 implies that

$$\bar{\rho}(\mathcal{V}_1^c,\mathcal{V}_2^c) < \bar{\rho}\left(\mathcal{V}_1^f,\mathcal{V}_2^f\right). \tag{4.11}$$

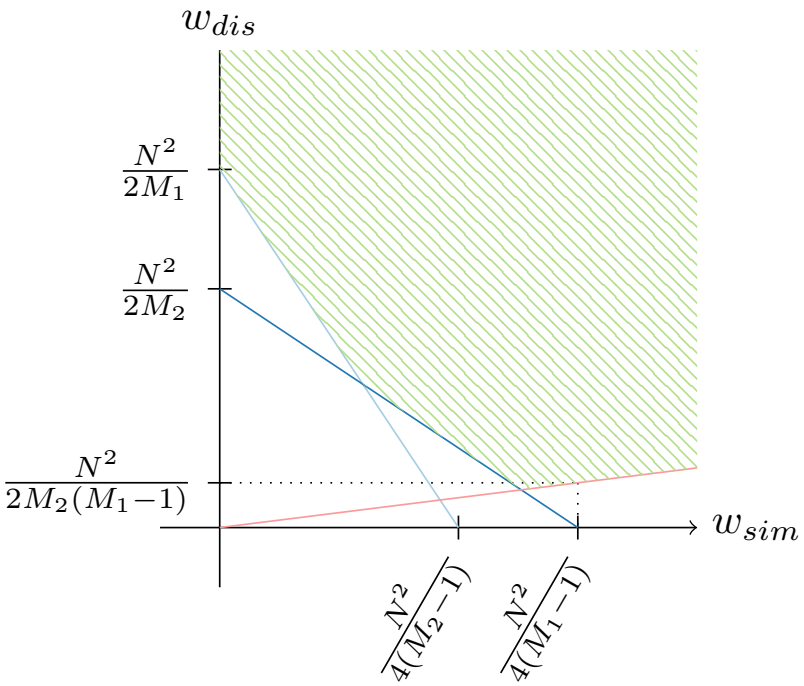


Figure 4.3: The admissible region from Theorem 4.2.1 is given by the hatched area (, assuming that $M_1 < M_2$). The blue lines correspond to equality in the first constraint of Theorem 4.2.1 and represent the cases of $i = 1$ (dark blue) and $i = 2$ (light blue) and the red line corresponds to equality in the second constraint.

is true for any graph, and hence correctly labeled sample nodes are always favorable compared to falsely labeled samples. □

There is no explicit solution for either w_{sim} or w_{dis} without further constraints. One idea is to choose w_{sim} and w_{dis} based on the admissible region which is illustrated in Fig. 4.3. It can be seen that the admissible region is bounded by three lines and so there are three points that could be interesting for the minimization of $w_{sim} + w_{dis}$, namely the points where $w_{sim} = 0$, the point where the two blue lines intersect, or the point where the dark blue line and the red line intersect.

4.2.2 Low Rank Reconstruction of Adjacency Matrix

For (weakly) balanced graphs, the clustering is directly visible in the adjacency matrix as all intra-cluster edges have positive weights and all inter-cluster edges have negative weights. In [21] the authors presented a method

that aims to reconstruct this adjacency matrix from the adjacency matrix of an observed subgraph. In this approach the underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}^*, \boldsymbol{W}^*)$ is assumed to be fully connected and its weight matrix is defined as

$$W_{ij}^* = \begin{cases} 1, & i \text{ and } j \text{ are in the same cluster,} \\ -1, & i \text{ and } j \text{ are in different clusters.} \end{cases} \tag{4.12}$$

Note here, that the main diagonal of the weight matrix is non-zero and thus, the graph has self-loops. This is a necessary prerequisite for the algorithm and can easily be obtained for every graph by setting the weights of the self-loops to one. The sampled weight matrix is then defined as

$$W_{ij} = \begin{cases} W_{ij}^*, & (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise,} \end{cases} \tag{4.13}$$

where $\mathcal{E}$ is the set of observed edges and $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ is the observed graph. This definition of the observed graph is equivalent to the modified graph in sSCs with $w_{sim} = w_{dis} = 1$, when the original graph has no edges. But contrary to SCs and sSCs, this approach samples the edge weights and not the cluster labels.

For the reconstruction of $\boldsymbol{W}^*$ from $\boldsymbol{W}$ [21] distinguishes between two types of methods, the local method and the global method. The local method predicts the unknown entries of $\boldsymbol{W}^*$ by means of a measure of imbalance $\mu(\cdot) : \mathbb{R}^{N \times N} \to \mathbb{R}_0^+$ which is positive if it is applied to the weight matrix of a balanced graph and zero otherwise. The predicted entries are then given by

$$\hat{W}_{ij} = \operatorname{sign}\Big(\mu\Big(\boldsymbol{W}^{-(i,j)}\Big) - \mu\Big(\boldsymbol{W}^{+(i,j)}\Big)\Big), \tag{4.14}$$

where $\boldsymbol{W}^{-(i,j)}$ and $\boldsymbol{W}^{+(i,j)}$ are duplicates of $\boldsymbol{W}$ with the ij-th entry set to -1 and 1, respectively. Intuitively, this means that it measures which graph is more unbalanced and then it decides for the other.

The global method is based on the rank of $\boldsymbol{W}^*$. For $K \leq 2$, $\operatorname{rank}(\boldsymbol{W}^*) = 1$ and otherwise $\operatorname{rank}(\boldsymbol{W}^*) = K$. This is because rows, whose index is in the same cluster, are equal (thus, there are at most K linearly independent rows) and for $K > 2$ it is possible to construct K linearly independent eigenvectors. For $K = 2$ the rows whose indices are in the second cluster are the inverse to the rows with indices in the first cluster. With this knowledge the authors formalized the problem as an NP-hard minimization problem

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{R}^{N \times N}} \quad & \operatorname{rank}(\boldsymbol{X}) \\ \text{s.t.} \quad & X_{ij} = W_{ij}^*, \forall (i,j) \in \mathcal{E}, \\ & X_{ij} \in \{-1, 1\}, \forall (i,j) \notin \mathcal{E} \end{aligned} \tag{4.15}$$

Algorithm 6 Sign prediction via singular value projection [21]

input: weight matrix $\boldsymbol{W}$ of the observed graph, rank k, tolerance ϵ, max iteration t_{max}, step size η

1: Initialize $\boldsymbol{X}^{(0)} \leftarrow \boldsymbol{0}$ and $t \leftarrow 0$
2: **repeat**
3: $\quad \hat{\boldsymbol{X}}^{(t)} \leftarrow \boldsymbol{X}^{(t)} - \eta\Big(P\Big(\boldsymbol{X}^{(t)}\Big) - \boldsymbol{W}\Big)$
4: $\quad [\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}] \leftarrow$ top k results of the singular value decomposition of $\hat{\boldsymbol{X}}^{(t)}$
5: $\quad \boldsymbol{X}^{(t+1)} \leftarrow \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
6: $\quad t \leftarrow t + 1$
7: **until** $\left\| P\Big(\boldsymbol{X}^{(t)}\Big) - \boldsymbol{W} \right\|_F^2 \leq \epsilon \vee t > t_{max}$
8: $\bar{\boldsymbol{X}} \leftarrow \operatorname{sign}\Big(\boldsymbol{X}^{(t)}\Big)$

output: $\bar{\boldsymbol{X}}$

for which it is possible to formulate conditions for perfect recovery when sampling the edges uniformly. Such a recovery condition is given by [21, Theorem 18], where a lower bound on the number of samples is prescribed such that $\boldsymbol{W}^*$ can be recovered perfectly with a certain probability. As an attempt to reduce the complexity the singular value projection

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{R}^{N \times N}} \quad & \|\mathcal{P}(\boldsymbol{X})\|_F^2 \\ \text{s.t.} \quad & \operatorname{rank}(\boldsymbol{X}) \leq k, \end{aligned} \tag{4.16}$$

with $\mathcal{P}(\cdot) : \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times N}$ being the projection operator

$$P(X)_{ij} = \begin{cases} X_{ij}, & (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise}, \end{cases} \tag{4.17}$$

was proposed. The procedure solving (4.16) is listed in Algorithm 6.

In their experiments [21] showed that the performance of the global method is better compared to the local method, both in terms of clustering performance and in computation time.

5

Numerical Experiments

In the numerical experiments, we compare the performance of sSCs to that of the global approach, singular value projection (SVP) of [21], and clustering by harmonic functions (HFC) [20]. Afterwards, we study different types of sampling and configurations for w_{sim} and w_{dis} in sSCs: first, the influence of different sample sizes M and degrading parameters in the data models will be studied for node- and edge sampling, second, the effect of node- and edge sampling on sSCs will be compared for fixed parameters in the data models and fixed number of samples; third, the bound from Theorem 4.2.1 will be analyzed for cases when it is met with equality and cases where it is not fulfilled. With the last analysis, a configuration for sSCs will be obtained beyond which no further improvement can be expected. The metric that is used for comparison is the number of incorrectly labeled nodes N_{err} and the data models are a two moon model (TMM), random cluster graphs (RCGs), a spiral model (SM), and a concentric circle model (CCM) (cf. Section 5.1).

Although it would be possible to use larger values for w_{sim} and w_{dis} we restrict to the choices 0 and 1 (as this was already the case when SCs was presented in [19]). Only in the experiments where the bounds will be analyzed, the sampling values will be allowed to become larger.

Throughout the experiments we will generate all graphs with $N = 1000$ nodes.

5.1 Data Models

In the experiments we used three model, namely TMM, SM and CCM, for which the data and the cluster labels were drawn from a random distribution and then a graph was formed by KNN with 5 neighbors. In the fourth model, RCG, the graph was generated by randomly creating edges. Fig. 5.1 shows one realization of each model in which the cluster labels were encoded by different colors. Since there is no position information in RCG, a graph drawing algorithm MDS from (Algorithm 2 with the graph metric being the minimum number of hops) had to be used.

The idea of TMM (and also SM and CCM) is that the support of each cluster should be a non-convex region. For TMM these regions are formed by two half circles where the end of of the first cluster is in the center of the second cluster and vice versa. The data points $\boldsymbol{x}_i$ for this model are generated by performing the following steps independently for every $i \in \{1, \ldots, N\}$:

- Draw a cluster label $l_i \in \{1, 2\}$ with both values being equally likely.
- Draw the angle ϕ_i from a uniform distribution $\phi_i \sim \mathcal{U}([0, \pi])$.
- Generate Gaussian noise $\boldsymbol{n}_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{2}\boldsymbol{I}\right)$.

The resulting data point is then

$$\boldsymbol{x}_i = \begin{bmatrix} \frac{3}{2} - l_i + \cos(\phi_i) \\ (2l_i - 3)\sin(\phi_i) \end{bmatrix} + \boldsymbol{n}_i. \tag{5.1}$$

For SM the first three steps are the same, except that the orientation ϕ_i is drawn uniformly from $\phi_i \sim \mathcal{U}([0, 2\pi])$. The data points for this model are then generated such that the clusters form interlacing spirals, which is accomplished by

$$\boldsymbol{x}_i = \left(\frac{\phi_i}{\pi} + 1\right) \begin{bmatrix} \cos(\phi_i + (l_i - 1)\pi) \\ \sin(\phi_i + (l_i - 1)\pi) \end{bmatrix} + \boldsymbol{n}_i. \tag{5.2}$$

In CCM the clusters are arranged as concentric circles by drawing l_i, ϕ_i and $\boldsymbol{n}_i$ the same way as for SM. The data $\boldsymbol{x}_i$ is constructed as

$$\boldsymbol{x}_i = l_i \begin{bmatrix} \cos(\phi_i) \\ \sin(\phi_i) \end{bmatrix} + \boldsymbol{n}_i. \tag{5.3}$$

For RCG the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ is generated such that the two clusters have exactly the same size and between any pair of two nodes v_i, $v_j \in \mathcal{V}$ an

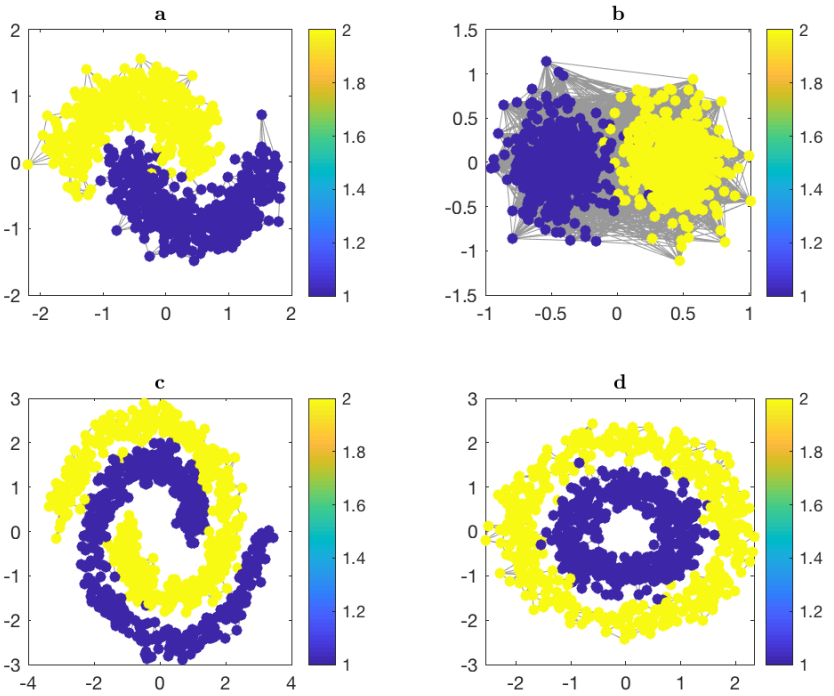


Figure 5.1: Examples for clustered data models: (a) TMM, (b) RCG, (c) SM and (d) CCM.

edge exists with probability

$$\mathrm{P}\{\{i,j\} \in \mathcal{E}\} = \begin{cases} 0, & i = j, \\ p_0, & i,j \in \mathcal{C}_1 \vee i,j \in \mathcal{C}_2, \\ p_1, & \text{otherwise}, \end{cases} \tag{5.4}$$

with $\mathcal{C}_1$ and $\mathcal{C}_2$ being the two clusters, p_0 being the intra-cluster probability and p_1 the inter-cluster probability.

5.2 Comparison of Clustering Algorithms

In the first experiment, the performance of SVP and sSCs with $w_{dis} = 1$ and $w_{dis} = 0$ is studied for edge sampling with varying M and. The results of this experiment are shown in Fig. 5.2 in terms of the empirical cumulative distribution function (ECDF) of N_{err}. It can be seen that for $M = 100$ all algorithms and configurations perform very well, but as the number of samples

decreases, also the performance decreases. For $M = 2$, SVP completely fails, as N_{err} is close to $\frac{N}{2}$ which is only as good as a random guess, but sSCs still yields good results in many cases. The bad performance for SVP is because it is required that for any node at least one sampled edge ends at this node, so that it has a chance to put the node into the correct cluster. If there is no such edge the result would be a pure guess. In the case of $M = 2$, only 2 edges are sampled and so this condition cannot be fulfilled. On the other hand, sSCs takes the structure of the graph into account and also inherits the performance from SC, which means that it would still work even without any samples. It can further be observed that $w_{dis} = 1$ yields better results than $w_{dis} = 0$ in the case of $M = 10$, which was already expected due to the construction of sSCs. Also, when looking at the examples in Fig. 5.1 it can be seen that the clusters in TMM can already be separated with relatively good quality by cutting with a horizontal line, for CCM a good cut would be a circle with radius 1.5 only for SM it is not possible to find such a simple shape. This simplicity of the cut shape is reflected in the clustering results, as the performance deteriorates when going from TMM to CCM and from CCM to SM.

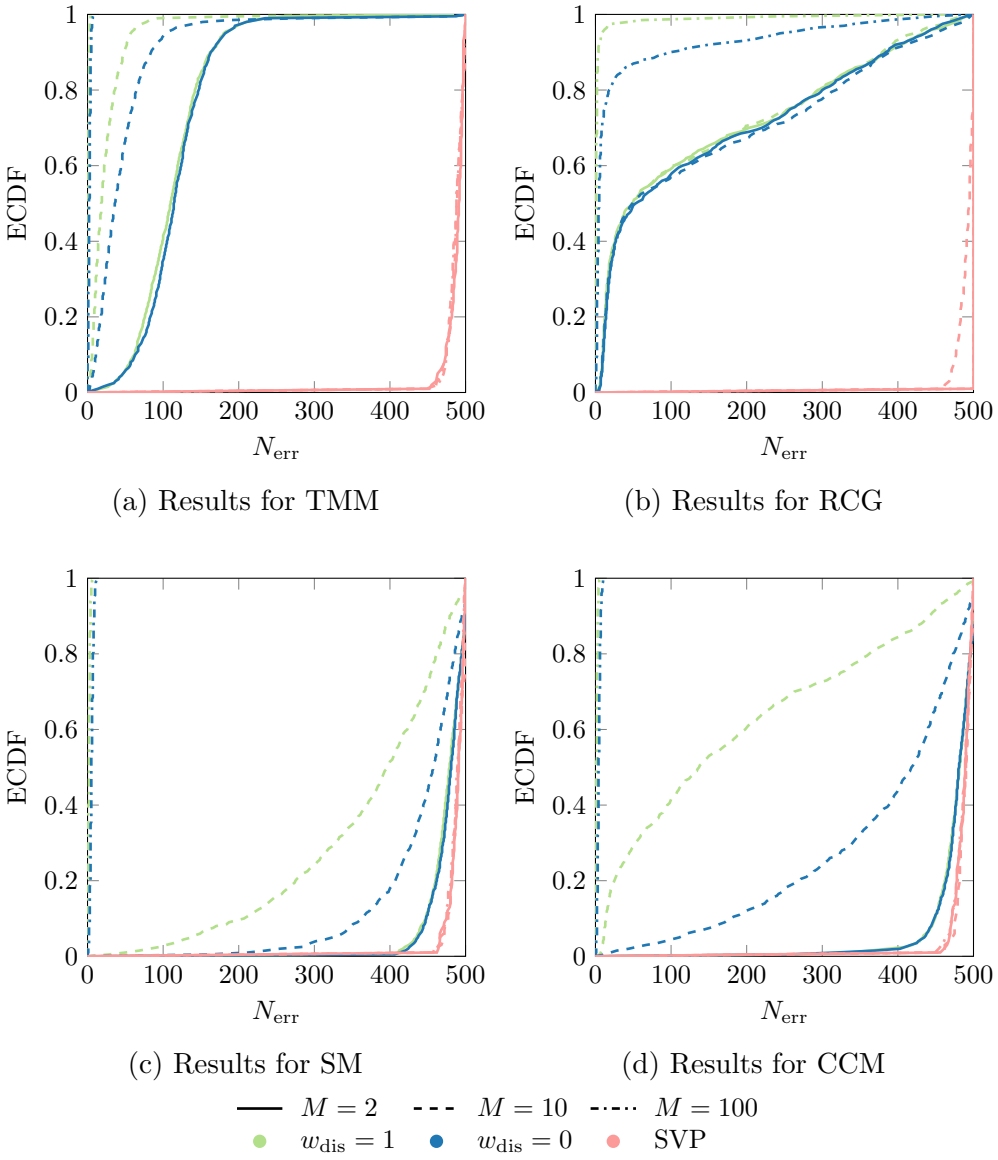
As it was seen above, SVP cannot compete with sSCs at least for the test problems for a low number of samples and also it is computationally very complex. Thus, it will not be considered anymore in the following experiments.

The next experiment analyzes the performance of sSCs for different noise variances in the data and for different inter-cluster edge probabilities. The results of this experiment are shown in Fig. 5.3 using box-plots. For TMM and RCG, both configurations of sSCs work almost equally well. The behavior that N_{err} increases with increasing σ and p_1 is expected because those two parameters control the separability of the clusters. In SM and CCM there is a major difference in the performance, i.e., there are cases for which $w_{dis} = 1$ still works but $w_{dis} = 0$ completely fails. This confirms that the extension of SCs to sSCs was a good choice. Also it can be observed that for all models there is a point at which the performance of clustering first starts to degrade. This point is $\sigma = 0.3$ for the geometric models and $p_1 = 0.02$ for RCG. Hence, in all other experiments we will use $\sigma = 0.3$ and $p_1 = 0.02$ since here clustering is challenging but still possible reasonably well.

We next compare sSCs to HFC for the case of node sampling. Whereas sSCs performs better than HFC for TMM and RCG, it performs worse for SM and CCM. This can be seen in Fig. 5.4 in the ECDF of N_{err} for different values of M and also in Fig. 5.5 in the box-plot of N_{err} for different values of σ and p_{inter}. But in all cases where HFC performs better, the median of N_{err} is larger than 250, which means that one fourth of all nodes has been labeled

falsely and so the clustering result is too bad to be considered as useful. In the cases, where the results are satisfactory sSCs performs better than HFC and for RCG it can even be seen that HFC completely fails. To illustrate this behavior, Fig. 5.6 shows the estimated cluster labels for RCG and CCM. In both cases the same graph with the same sampling points has been fed into HFC and sSCs with $w_{dis} = 1$. For RCG, HFC only assigns the nodes that were sampled from the second cluster to this cluster and all the others to the first cluster, whereas sSCs puts most of the nodes into the correct cluster. The failure of HFC can be explained as follows: If a sample is only loosely connected to the graph, it has only minor influence on the clustering result, whereas heavily connected samples have a big impact. If all samples from one cluster are loosely connected, they will be the only nodes that are assigned to this cluster. For CCM the argument is different: here it may happen that all samples from the inner cluster were taken from the upper half and all samples from the outer cluster were taken from the lower half. sSCs then favors to cut along a straight line in the middle of the graph to minimize the number of cut edges. On the other hand HFC assigns all nodes to the cluster of the sampled nodes for which their connection is strongest. In the given example, this is a bigger portion of the inner cluster and a smaller portion of the outer cluster than for sSCs. Thus, HFC has a better performance in this case.

In the previous experiments it was seen that sSCs with $w_{dis} = 1$ performs better than $w_{dis} = 0$. The reason for this can be found in the clustering of the sampled nodes. Fig. 5.7 shows the ECDF of the number of falsely labeled samples $N_{\text{err,sample}}$ relative to the total number of samples in percent, $N_{\text{err,rel}} = \frac{N_{\text{err,sample}}}{M} \cdot 100\,\%$. It can be seen that $w_{dis} = 0$ performs worse than $w_{dis} = 1$. With $w_{dis} = 0$, it happens more frequently that sampled nodes are all put together in the same cluster and thus also many of the other nodes are clustered incorrectly. This is because $w_{dis} = 0$ does not put any penalty on having all samples in the same cluster.



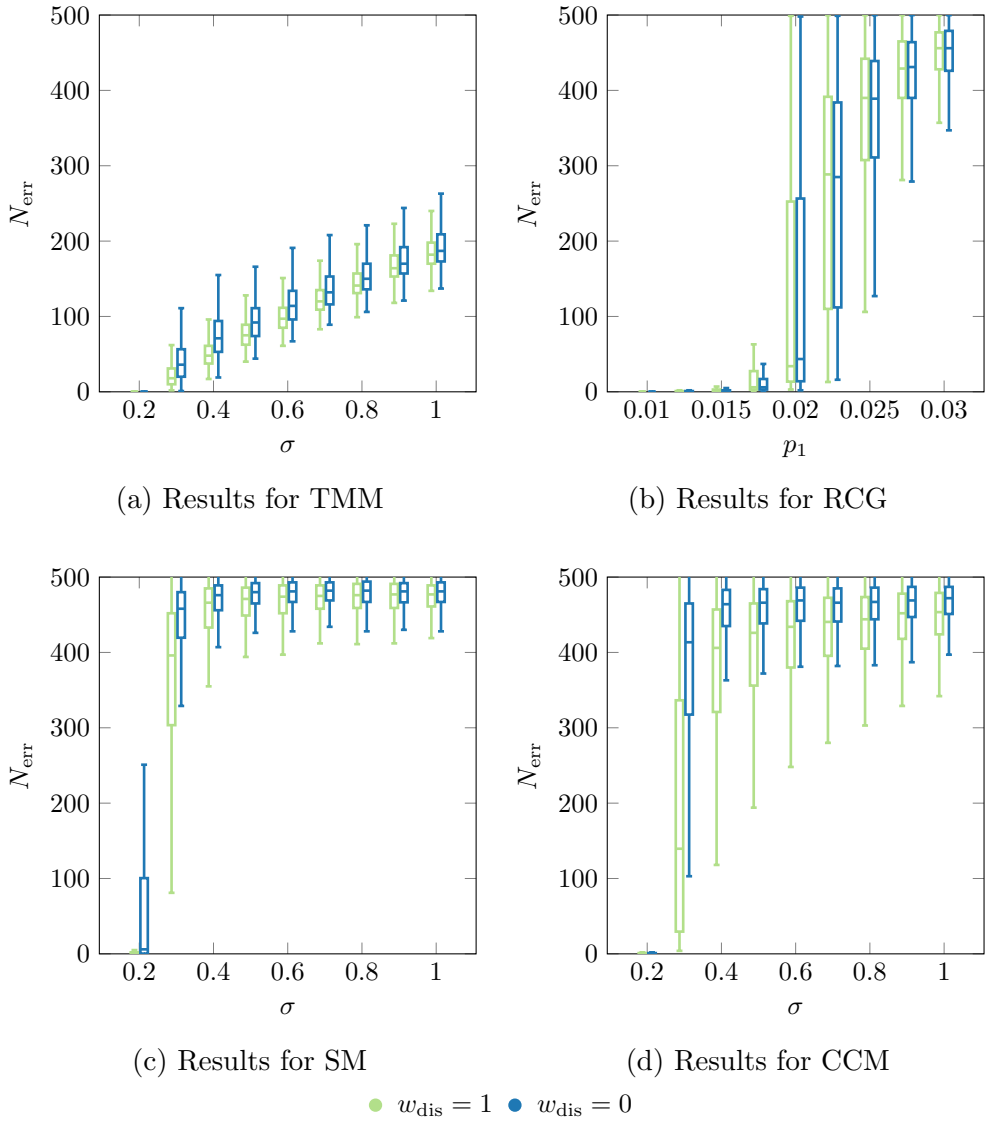
(a) Results for TMM

(b) Results for RCG

(c) Results for SM

(d) Results for CCM

Figure 5.2: Comparison of SVP and sSCs with $w_{dis} = 0$ and $w_{dis} = 1$ in terms of the ECDF of N_{err} for $M \in \{2, 10, 100\}$ and edge sampling. The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, $\sigma = 0.3$ for the noise in TMM, SM and CCM, and $p_{intra} = 0.05$ and $p_{inter} = 0.02$ in RCG.



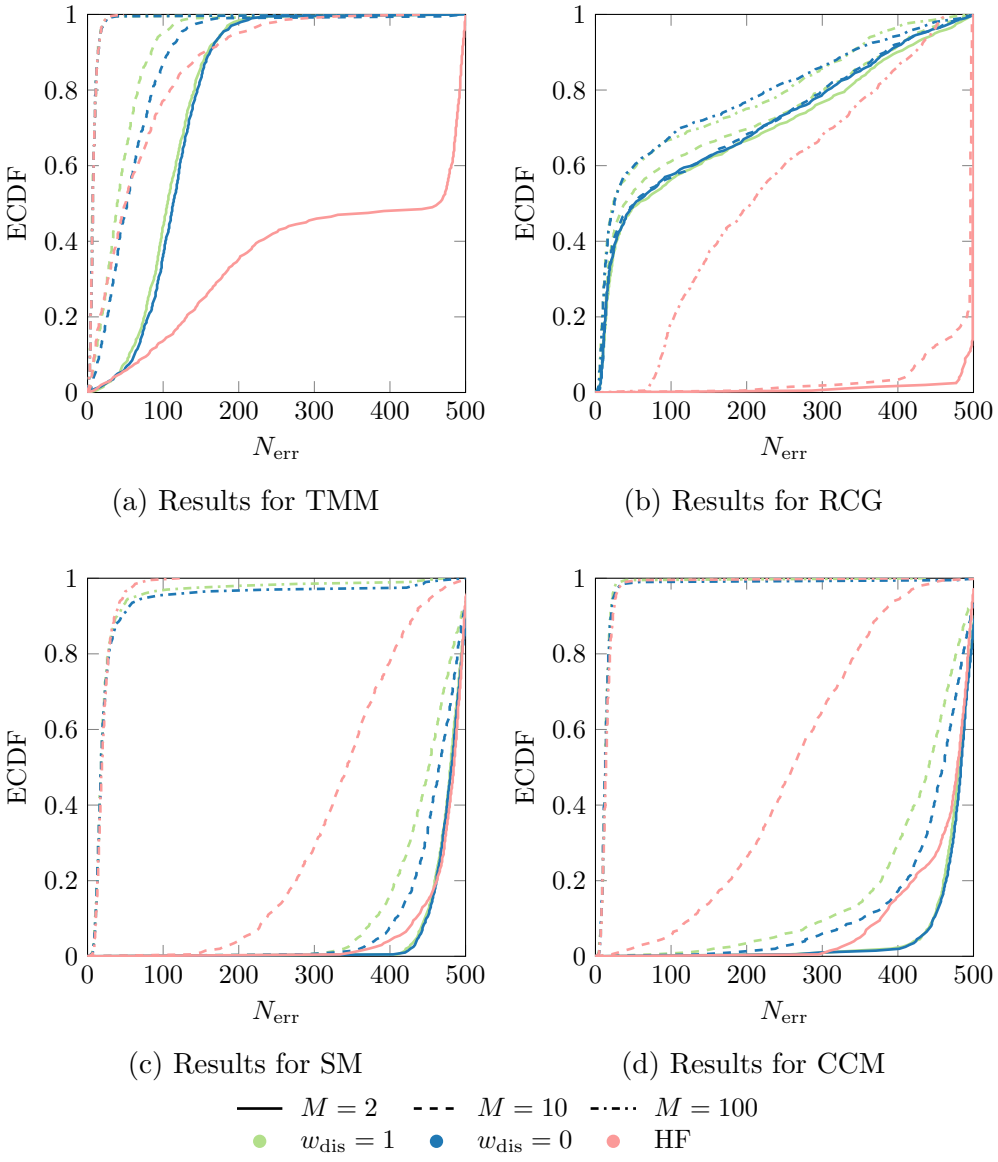
(a) Results for TMM

(b) Results for RCG

(c) Results for SM

(d) Results for CCM

Figure 5.3: Comparison of sSCs with $w_{dis} = 0$ and $w_{dis} = 1$ as a box-plot of N_{err} for $M = 10$ and edge sampling over varying σ for the noise in TMM, SM and CCM and varying p_{inter} in RCG. The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, and $p_{intra} = 0.05$ in RCG.



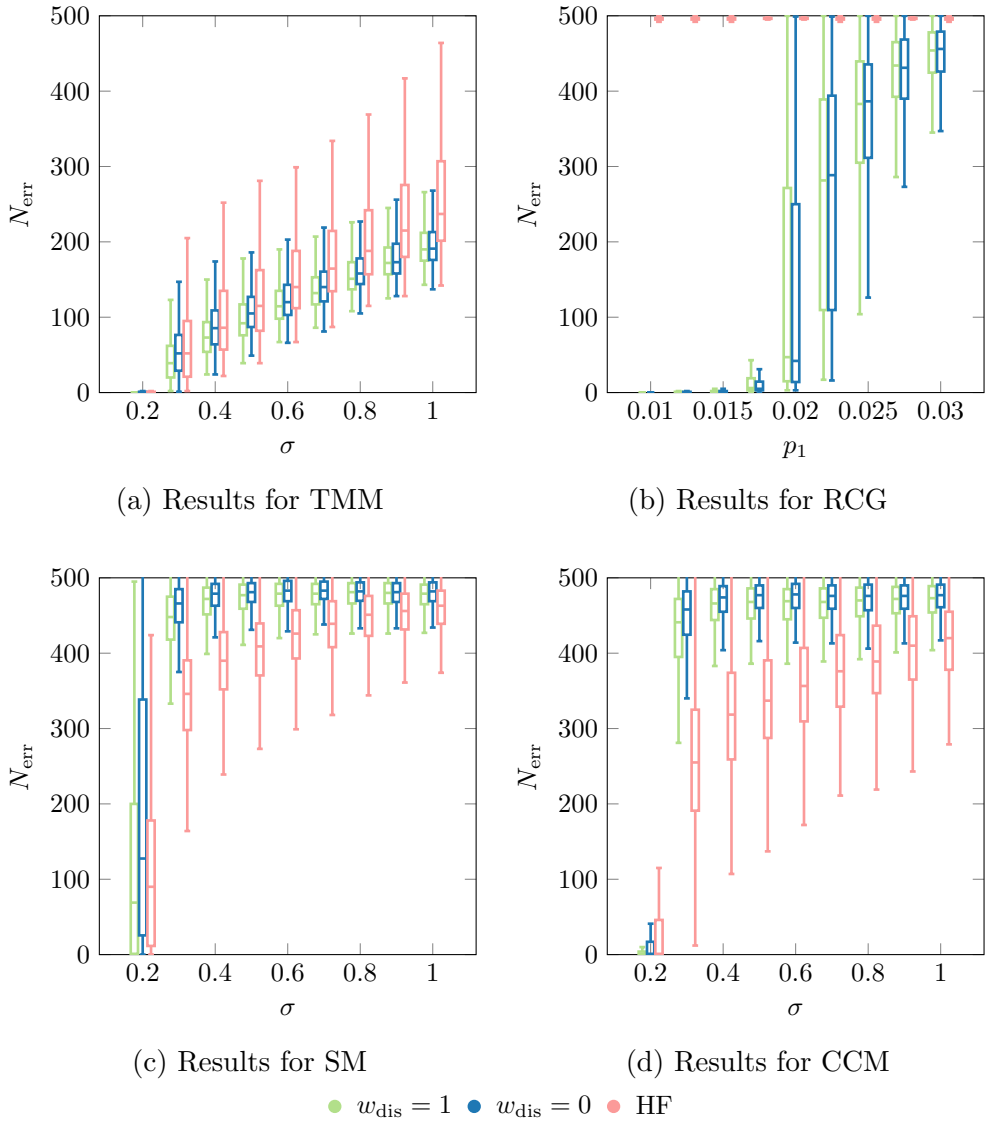
(a) Results for TMM

(b) Results for RCG

(c) Results for SM

(d) Results for CCM

Figure 5.4: Comparison of HFC and sSCs with $w_{dis} = 0$ and $w_{dis} = 1$ in terms of the ECDF of N_{err} for $M \in \{2, 10, 100\}$ and node sampling. The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, $\sigma = 0.3$ for the noise in TMM, SM and CCM, and $p_{intra} = 0.05$ and $p_{inter} = 0.02$ in RCG.



(a) Results for TMM

(b) Results for RCG

(c) Results for SM

(d) Results for CCM

Figure 5.5: Comparison of HFC and sSCs with $w_{dis} = 0$ and $w_{dis} = 1$ as a box-plot of N_{err} for $M = 10$ and node sampling over varying σ for the noise in TMM, SM and CCM and varying p_{inter} in RCG. The graphs were generated with $N = 1000$, 5 neighbors in KNN, and $p_{intra} = 0.05$ in RCG.

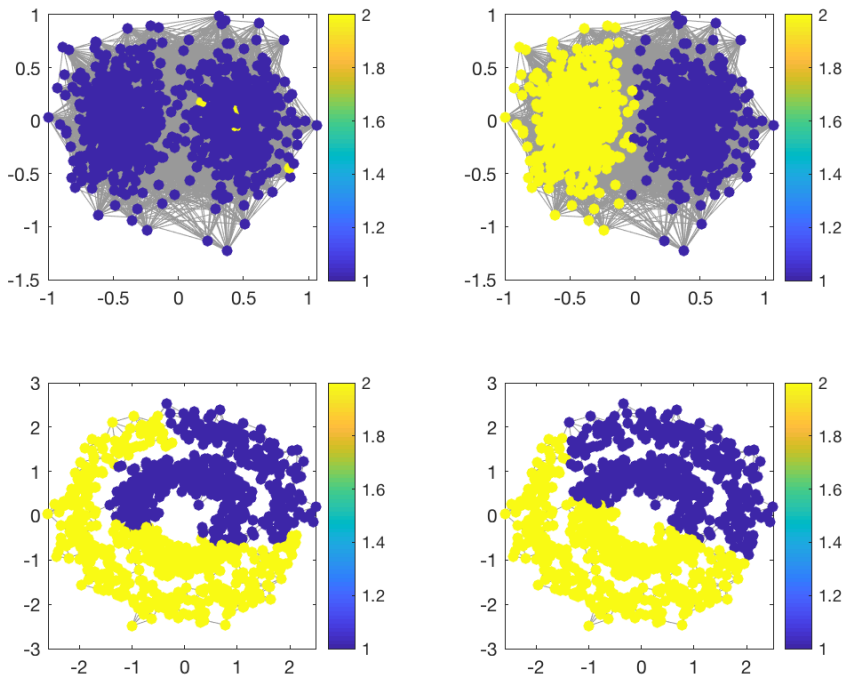
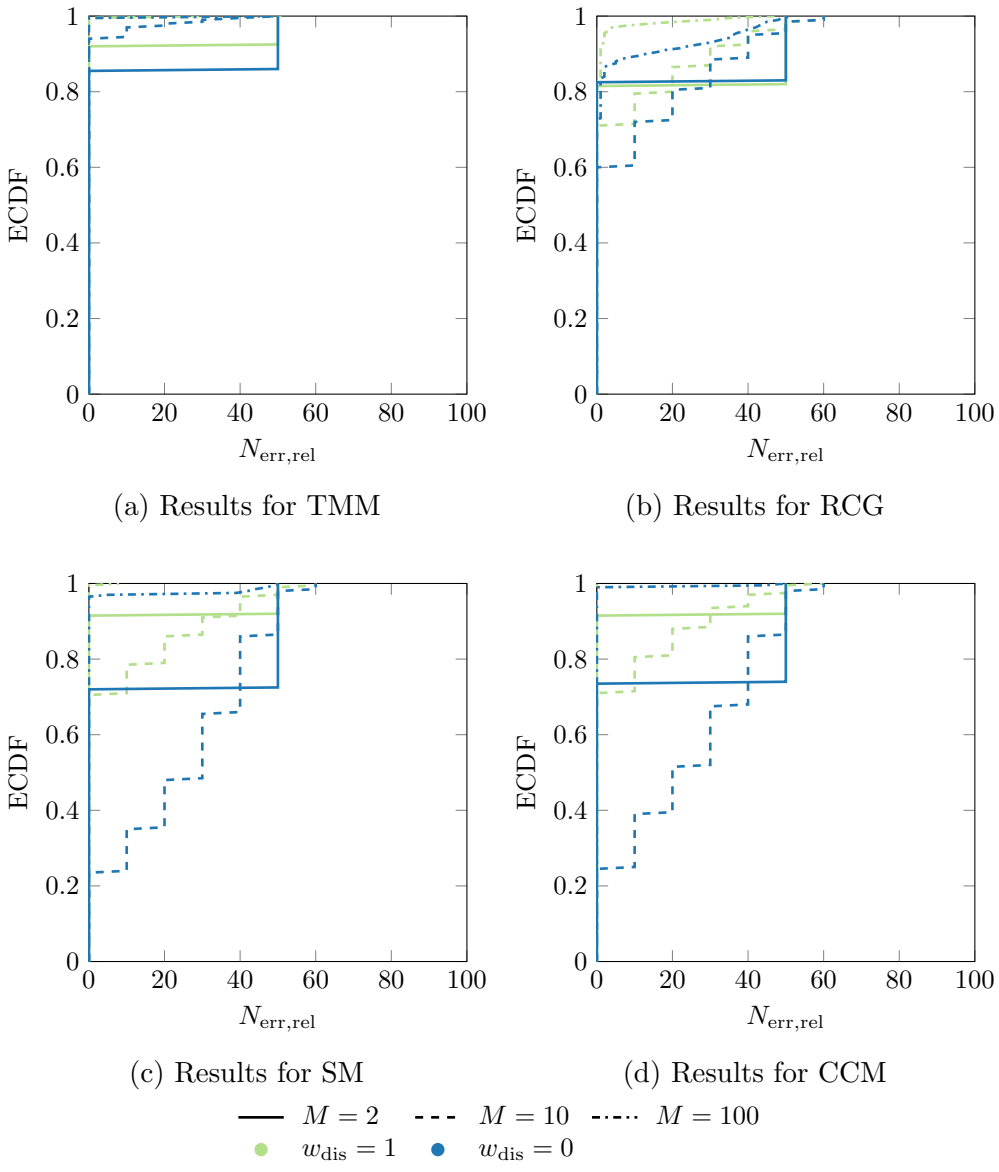


Figure 5.6: Clustering results from HFC (left) and sSCs with $w_{dis} = 1$ (right) for RCG (top) and CCM (bottom). The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, $\sigma = 0.3$ for the noise in CCM, and $p_{intra} = 0.05$ and $p_{inter} = 0.02$ in RCG.



(a) Results for TMM

(b) Results for RCG

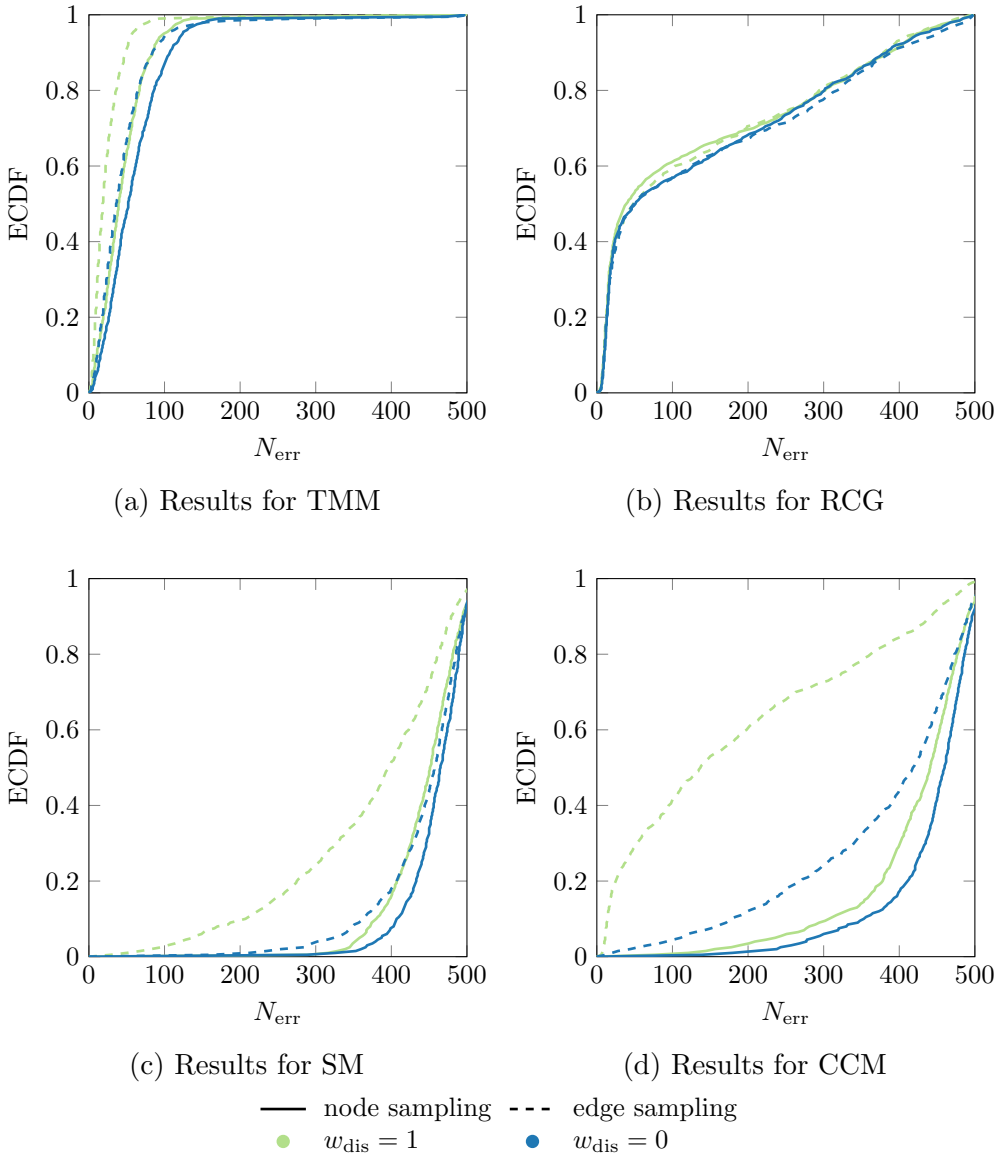
(c) Results for SM

(d) Results for CCM

Figure 5.7: Comparison of HFC and sSCs with $w_{dis} = 0$ and $w_{dis} = 1$ in terms of the ECDF of the number of falsely labeled samples relative to the total number of samples for $M \in \{2, 10, 100\}$ and node sampling. The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, $\sigma = 0.3$ for the noise in TMM, SM and CCM, and $p_{intra} = 0.05$ and $p_{inter} = 0.02$ in RCG.

5.3 Performance of Different Sampling Strategies

As it could have already been expected, the performance is different for node sampling and edge sampling. Fig. 5.8 shows the results of sSCs when used in combination with node sampling and edge sampling. It can be seen that the performance of edge sampling is better than that of node sampling for all models except RCG. The performance advantage of edge sampling is due to the facts that with node sampling not all generated edges give additional information. For example, when three nodes are sampled, three bits of information are generated, from which only two are not redundant: if the first node is in the same cluster as node two, and node two is in the same cluster as node three, the information about node one and three is implicitly given. For edge sampling it is very likely that three edges are connected to different nodes and so all three edges carry non-redundant information. For RCG this effect is only minor for $w_{dis} = 0$, because the number of edges in the graph is already very big and so positive edges do not carry a lot of information. For $w_{dis} = 1$ it is even the opposite because with node sampling there is a minimum amount of $M - 1$ negative edges if at least one node is sampled from a different cluster than the others. This minimum number of negative weights does not exist for edge sampling. Thus, the amount of edges that do carry important information is bigger for node sampling.



(a) Results for TMM

(b) Results for RCG

(c) Results for SM

(d) Results for CCM

Figure 5.8: Comparison of node sampling and edge sampling for sSCs with $w_{dis} = 0$ and $w_{dis} = 1$ in terms of the ECDF for $M = 10$. The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, $\sigma = 0.3$ for the noise in TMM, SM and CCM, and $p_{intra} = 0.05$ and $p_{inter} = 0.02$ in RCG.

5.4 Analysis of Weight Bounds

For the case of $w_{dis} = 0$ the bound in Theorem 4.2.1 is

$$w_{sim} \geq \frac{N^2}{4\min\{M_1, M_2\}},$$

but as M_i is unknown before runtime, the approximation $M_i \approx \frac{M}{2}$ will be used and so for $N = 1000$ and $M = 10$ the bound is $w_{sim} \geq 50000$. As this bound was derived for the worst case of an unconnected graph versus a fully connected graph, it is expected to be very conservative. Thus, the performance of sSCs with $w_{dis} = 0$ was analyzed for five values that are logarithmically spaced from 1 to 50000. The results of those simulations are shown in Fig. 5.9. It can be seen that in the case of TMM and RCG the performance of $w_{sim} = 1$ is already good and there is not much space left for improvement, but for SM and CCM this is not the case and it can be seen that the performance improves significantly. Only for $w_{sim} > 15$ there is no further improvement.

When $w_{dis} \geq 0$ is allowed, the whole admissible region from Fig. 4.3 can be analyzed. Again, the exact values of M_i are unknown and thus we use $M_i \approx \frac{M}{2}$. We will first study the special case $w_{sim} = w_{dis}$ and afterwards investigate distinct weight values along the straight line separating the admissible region from the inadmissible region.

With $w_{sim} = w_{dis} = w$ and $M_i \approx \frac{M}{2}$ for $i \in \{1, 2\}$, the bound from Theorem 4.2.1 reads

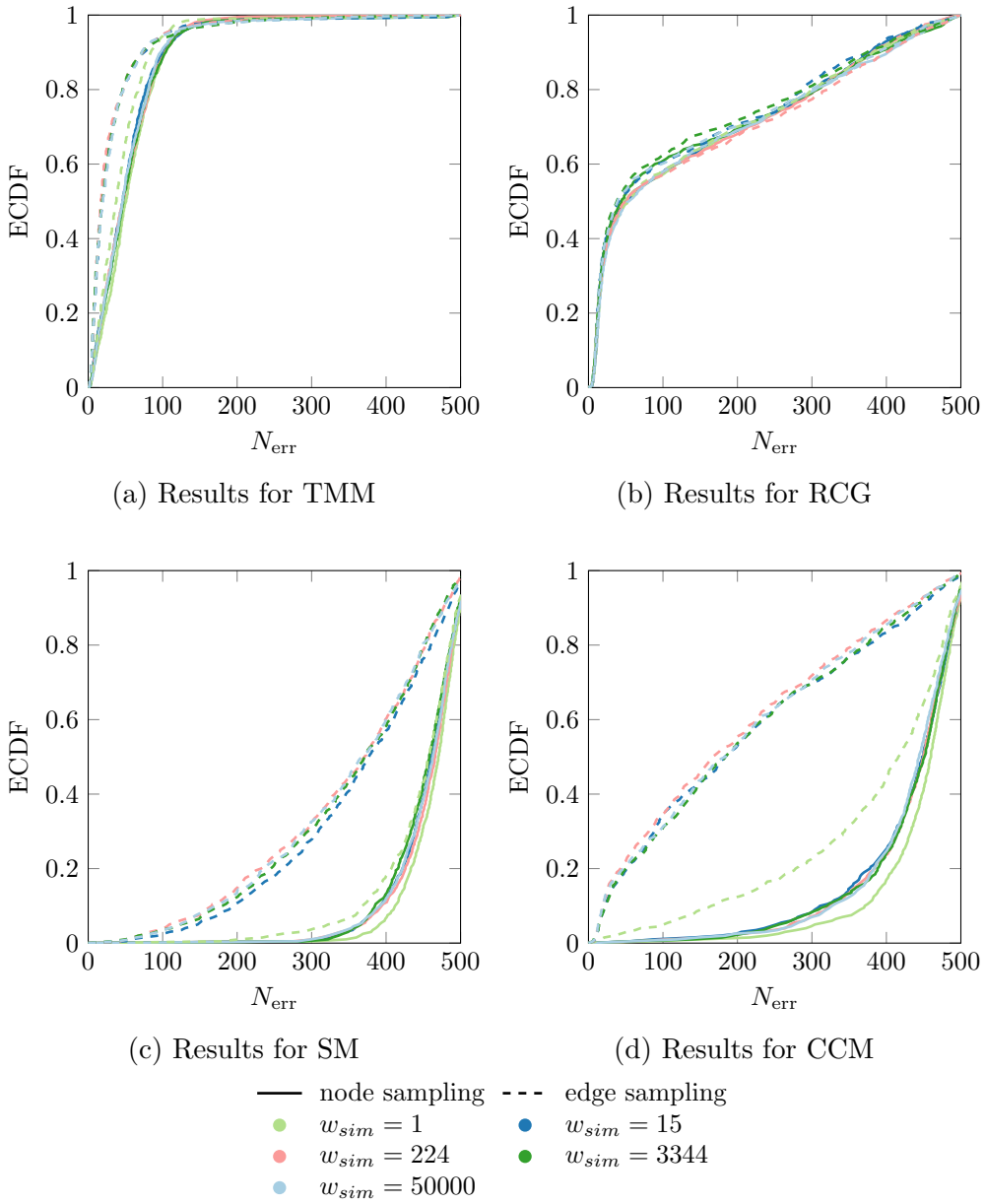
$$w > \frac{N^2}{2 \min\limits_{i \in \{1,2\}} \{2(M_i - 1) + M_{3-i}\}} \approx \frac{N^2}{3M - 2} \tag{5.5}$$

which is fulfilled by $w = 38463$ for $N = 1000$ and $M = 10$. Fig. 5.10 shows the results for logarithmically spaced values in the range from 1 to 38463. It can be observed that for TMM and RCG there are only minor changes in performance that are due to random fluctuations. Also for CCM there is only little space left for improvement when using edge sampling, but for node sampling there is an obvious increase in performance when $w > 1$ (although the overall performance here is already rather bad which renders it useless for clustering). The biggest performance increase occurs for SM with edge sampling, where $w = 1$ gives acceptable clustering results, but a value of $w \geq 14$ results in very good clustering performance.

Also, when choosing w_{sim} and w_{dis} along the boundary between the admissible region and the inadmissible region, a difference between the configurations can be identified. When either w_{sim} or w_{dis} is close to zero the

performance is worse than when their distance to zero is in the same order of magnitude, which is similar to the former experiments. But also if only one of them is allowed to be non-zero, it makes a difference which of these two values is chosen for that. In general, the case $w_{dis} > 0$ performs better than the case $w_{sim} > 0$. This is because the unmodified weight matrix already contains positive values and so negative weights can have a greater impact on the performance. Such a behavior can also be expected as $w_{dis} = 0$ lies in the inadmissible region of Fig. 4.3.

Overall, the results of this experiment suggest that for sSCs it is advantageous to choose both w_{sim} and w_{dis} greater than one and also both in the same order of magnitude. A good performance for the examples considered can already be obtained for $w_{sim} = w_{dis} = 14$ without any substantial performance gains when increasing the values.



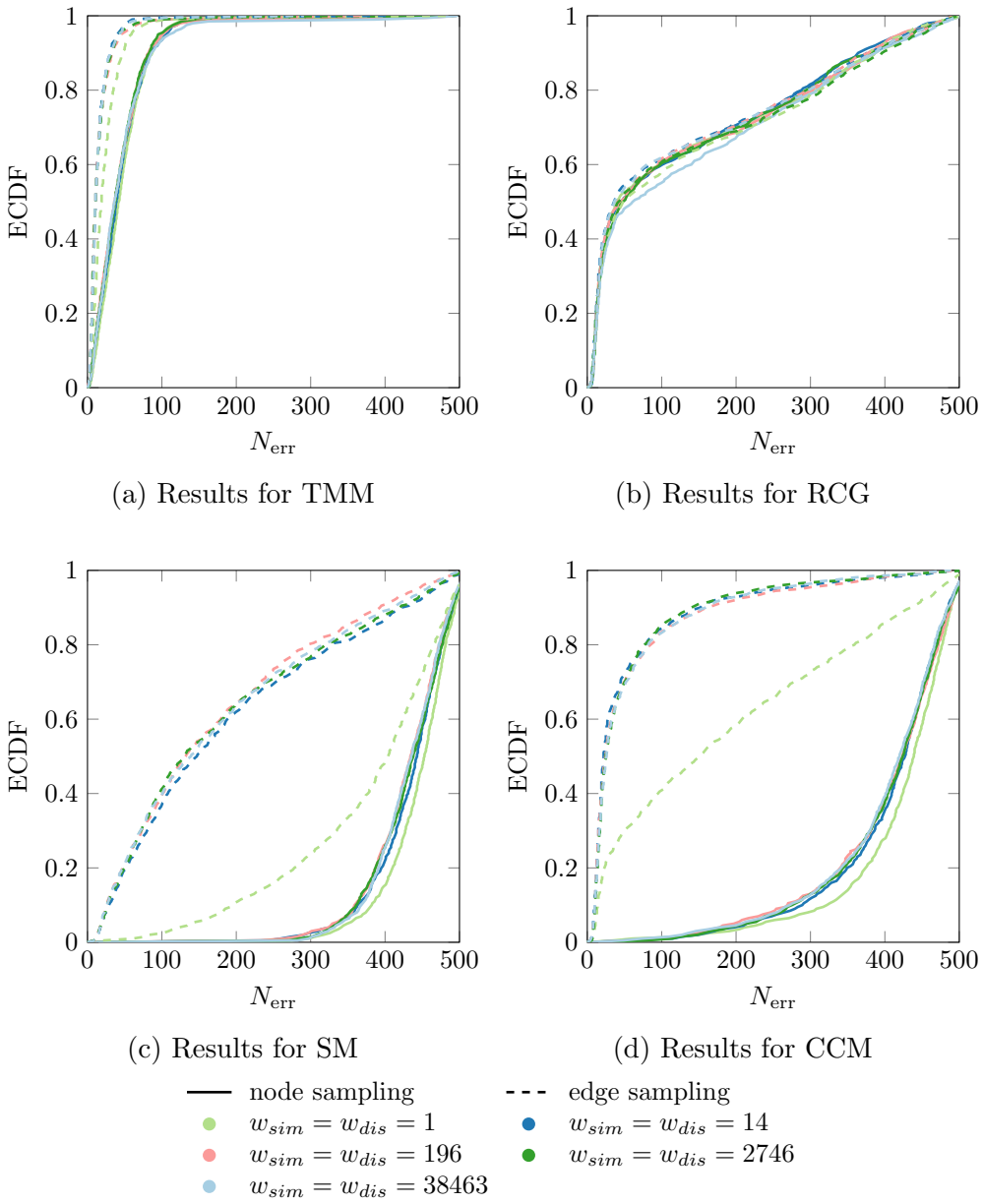
(a) Results for TMM

(b) Results for RCG

(c) Results for SM

(d) Results for CCM

Figure 5.9: Comparison of different values for w_{sim} with $w_{dis} = 0$ for sSCs in terms of ECDF for $M = 10$. The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, $\sigma = 0.3$ for the noise in TMM, SM and CCM, and $p_{intra} = 0.05$ and $p_{inter} = 0.02$ in RCG.



(a) Results for TMM

(b) Results for RCG

(c) Results for SM

(d) Results for CCM

Figure 5.10: Comparison of different values for w_{sim} and w_{dis} with $w_{sim} = w_{dis}$ for sSCs in terms of ECDF for $M = 10$. The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, $\sigma = 0.3$ for the noise in TMM, SM and CCM, and $p_{intra} = 0.05$ and $p_{inter} = 0.02$ in RCG. Comparison of (2) for different w_{sim}.

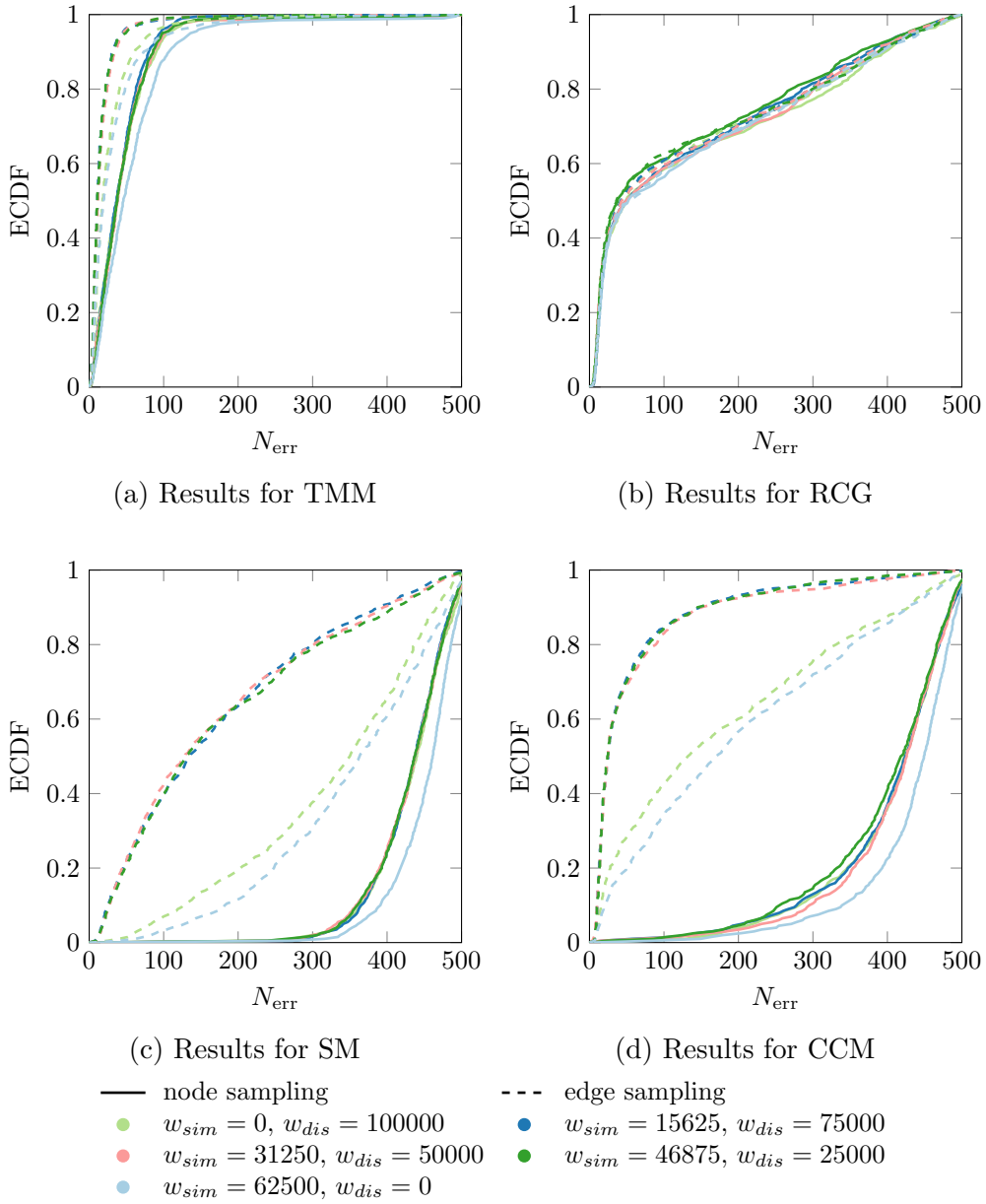


Figure 5.11: Comparison of different values for w_{sim} and w_{dis} along the line that separates the admissible region from the inadmissible region in the case that $M_1 = M_2$ for sSCs in terms of ECDF for $M = 10$. The graphs were generated with $N = 1000$, 5 neighbors in KNN, $\sigma_s = 1$ in the similarity measure, $\sigma = 0.3$ for the noise in TMM, SM and CCM, and $p_{intra} = 0.05$ and $p_{inter} = 0.02$ in RCG.

6

Conclusions and Outlook

6.1 Conclusions

In this work we gave a short introduction to some GSP concepts with a focus on those that are required for graph-based SC. We then introduced the work on sSC, which modifies the weight matrix of a graph according to the similarity of sampled cluster labels, to include sampling in SC. This work was then extended to sSCs to also incorporate dissimilarity information. For the last modification we derived analytic results to find optimal values for the modifications of the weight matrix. To also get a different view on clustering, we introduced MDS and SVP.

In the experiments sSCs was tested against sSC, SVP and HFC. We concluded that sSCs is superior to the other algorithms. Then we analyzed the performance of node sampling against edge sampling which showed the expected behavior that edge sampling performs better than node sampling. In the last experiments we analyzed our theoretical bounds for the optimum values in sSCs and could see that the derived bounds are rather conservative when applied to graphs with clusters.

6.2 Outlook

Although our approach seems very promising, there are still open questions and results that could be improved further:

- sSCs makes use of sSC, which can only be used for two clusters. In reality the data often has multiple clusters. Thus, sSCs needs to be extended to this case.
- For bounds for w_{sim} and w_{dis} we made use of graphs without any edges and fully connected graphs, which obviously do not have any cluster structure. It may be possible to derive tighter bounds for certain classes of graphs with cluster structure.
- In the comparison of edge sampling and node sampling, there was a lot of redundant information in node sampling. This leaves the open question of how many edges and nodes need to be sampled in the different strategies to get comparable results.
- The idea of using negative values to encode dissimilarity information seems to be promising and so one could also try to incorporate this information to signal recovery by TV minimization.

Acronyms

CCM	concentric circle model
CDF	cumulative distribution function
DFT	discrete Fourier transform
ECDF	empirical cumulative distribution function
GFT	graph Fourier transform
GSP	graph signal processing
HFC	clustering by harmonic functions
KNN	k-nearest neighborhood graph
MDS	multidimensional scaling
mKNN	mutual k-nearest neighborhood graph
RC	ratio cut
RCG	random cluster graph
SC	spectral clustering
SCs	spectral clustering with sampling
SM	spiral model
sRC	signed ratio cut
sSC	signed spectral clustering
sSCs	signed spectral clustering with sampling
SVP	singular value projection
TMM	two moon model
TV	total variation

Bibliography

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[2] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[3] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Mag.*, vol. 31, no. 5, pp. 80–90, 2014.

[4] G. Gilboa and S. Osher, "Nonlocal operators with applications to image processing," *Multiscale Modeling & Simulation*, vol. 7, no. 3, pp. 1005–1028, 2008.

[5] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[6] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann, "The Laplacian spectrum of graphs," *Graph theory, combinatorics, and applications*, vol. 2, no. 871-898, p. 12, 1991.

[7] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[8] P. Berger, G. Hannak, and G. Matz, "Graph learning based on total variation minimization," 2018, in preparation.

[9] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal recovery on graphs: Random versus experimentally designed sampling," in *Int. Conf. Sampling Theory and Applications*, May 2015, pp. 337–341.

[10] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal recovery on graphs: Fundamental limits of sampling strategies," *IEEE Trans. Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 539–554, Dec. 2016.

[11] L. F. O. Chamon and A. Ribeiro, "Universal bounds for the sampling of graph signals," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, March 2017, pp. 3899–3903.

[12] N. Tremblay, P.-O. Amblard, and S. Barthelmé, "Graph sampling with determinantal processes," in *European Signal Processing Conf.*, 2017, pp. 1674–1678.

[13] L. F. O. Chamon and A. Ribeiro, "Greedy sampling of graph signals," *IEEE Trans. Signal Processing*, vol. 66, no. 1, pp. 34–47, 2018.

[14] H. Q. Nguyen and M. N. Do, "Downsampling of signals on graphs via maximum spanning trees," *IEEE Trans. Signal Processing*, vol. 63, no. 1, pp. 182–191, Jan. 2015.

[15] M. Belkin, P. Niyogi, and V. Sindhwani, "On manifold regularization." in *AISTATS*, 2005, p. 1.

[16] P. Berger, G. Hannak, and G. Matz, "Graph signal recovery via primal-dual algorithms for total variation minimization," *IEEE J. Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 842–855, Sept. 2017.

[17] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *J. Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[18] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

[19] S. Kamvar, D. Klein, and C. Manning, "Spectral learning," in *Int. Joint Conf. Artificial Intelligence*. Stanford InfoLab, 2003.

[20] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. Int. Conf. Machine learning (ICML-03)*, 2003, pp. 912–919.

[21] K.-Y. Chiang, C.-J. Hsieh, N. Natarajan, I. S. Dhillon, and A. Tewari, "Prediction and clustering in signed networks: a local to global perspective," *The J. Machine Learning Research*, vol. 15, no. 1, pp. 1177–1213, 2014.

[22] X. Bresson, T. Laurent, D. Uminsky, and J. Von Brecht, "Multiclass total variation clustering," in *Advances in Neural Information Processing Systems*, 2013, pp. 1421–1429.

[23] X. Wang, B. Qian, and I. Davidson, "On constrained spectral clustering and its applications," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 1–30, 2014.

[24] T. Dittrich, P. Berger, and G. Matz, "Semi-supervised spectral clustering using the signed Laplacian," manuscript submitted for publication.

[25] K.-Y. Chiang, J. J. Whang, and I. S. Dhillon, "Scalable clustering of signed networks using balance normalized cut," in *Proc. 21st ACM Int. Conf. Information and Knowledge Management*. ACM, 2012, pp. 615–624.

[26] F. Harary *et al.*, "On the notion of balance of a signed graph." *The Michigan Mathematical Journal*, vol. 2, no. 2, pp. 143–146, 1953.

[27] J. A. Davis, "Clustering and structural balance in graphs," *Human relations*, vol. 20, no. 2, pp. 181–187, 1967.

[28] M. Hein, J.-Y. Audibert, and U. Von Luxburg, "From graphs to manifolds–weak and strong pointwise consistency of graph Laplacians," in *Int. Conf. Computational Learning Theory*. Springer, 2005, pp. 470–485.

[29] M. Meila and J. Shi, "A random walks view of spectral segmentation," in *AISTATS*. Citeseer, 2001.

[30] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak, "Spectral analysis of signed graphs for clustering, prediction and visualization," in *Proc. SIAM Int. Conf. Data Mining*, 2010, pp. 559–570.

[31] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[32] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Trans. Image Processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

[33] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[34] Y.-C. Wei and C.-K. Cheng, "Towards efficient hierarchical designs by ratio cut partitioning," in *IEEE Int. Conf. Computer-Aided Design*. IEEE, 1989, pp. 298–301.

[35] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Computer-aided Design of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[36] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[37] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Int. Conf. Computer Vision*. IEEE, 1998, pp. 839–846.

[38] A. Singhal *et al.*, "Modern information retrieval: A brief overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35–43, 2001.

[39] Y. H. Li and A. K. Jain, "Classification of text documents," *The Computer Journal*, vol. 41, no. 8, pp. 537–546, 1998.

[40] J. Goldberger, S. Gordon, and H. Greenspan, "An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures," in *Proc. IEEE Int. Conf. on Computer Vision*, 2003, p. 487.

[41] B. Logan and A. Salomon, "A music similarity function based on signal analysis." in *IEEE Int. Conf. Multimedia and Expo*, 2001, pp. 22–25.

[42] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[43] P. Berger, G. Hannak, and G. Matz, "Coordinate descent accelerations for signal recovery on scale-free graphs based on total variation minimization," in *Proc. European Signal Process. Conf.*, Aug. 2017, pp. 1689–1693.

[44] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York, NY: Springer New York, 2009.

[45] F. Wickelmaier, "An introduction to MDS," *Sound Quality Research Unit, Aalborg University, Denmark*, vol. 46, no. 5, 2003.

[46] A. V. Knyazev, "Signed Laplacian for spectral clustering revisited," *arXiv preprint arXiv:1701.01394*, 2017.

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct, insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.
Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Datum
Unterschrift
Name Thomas Dittrich