DIPLOMA THESIS

Multi-class Clustering with Cluster Label Redundancy

performed at the

Institute of Telecommunications
TECHNISCHE UNIVERSITÄT WIEN

supervised by

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Gerald Matz
and
Univ.Ass. Dipl.-Ing. Thomas Dittrich

by

Iman REZAEI, BSc.
Matr. Nr. 01528639

Vienna, November 19, 2020

Abstract

Big data analysis has attracted enormous attention during the last decade. In this field, graph signal processing is widely used in order to analyze data on graphs using a signal processing perspective. One of the key problems in this area is clustering, i.e., partitioning the data in homogeneous subgroups (clusters, communities) corresponding to subsets of nodes in the graph. The nodes within a cluster usually tend to be connected by edges that represent similarity relations. The ultimate goal in the clustering problem is to keep the number of nodes that are attributed to a wrong cluster as low as possible. Semi-supervise clustering is a special case of this problem in which some of the cluster labels are assumed to be known in advance and this prior knowledge is exploited to label (cluster) the rest of the data.

In this work, we explore the use of error correcting codes in the context of semi-supervised multi-class clustering problems. The key idea here is to employ error correcting codes for the cluster (class) labels in connection with various clustering schemes and to use the decoder of the error correcting code in order to detect and correct miss-classified data. The proposed algorithms for clustering with error-correcting code labels are studied experimentally in several different scenarios (graph size and topology, number and size of clusters). The performance of these methods is compared to conventional semi-supervised spectral clustering with one-hot encoded labels. The results show that with some modifications of the codeword structure of one-hot encoding, our error correcting approach outperforms existing clustering methods.

Acknowledgements

First and foremost, I would like to thank Ao.Univ.Prof. Dipl.-Ing. Dr.tech. Gerald Matz, for his continuous support and understanding. His guidance and motivation has steered me in the right direction throughout this work.
Furthermore, I am very grateful to Univ.Ass. Dipl.-Ing. Thomas Dittrich, for his patience and enthusiasm. I could not have imagined having a better advisor. Without his comprehensive knowledge and great support, I would not have been able to complete this work.
And my biggest thanks, from the bottom of my heart, goes to my beloved wife, Morvarid. Her encouragement and unconditional support kept me going on. Also, I would like to appreciate her effort for proofreading this thesis.

Contents

List of Figures

Chapter 1

Introduction

The field of big data analysis belongs to the wide range of machine learning applications, from pattern recognition and image segmentation to biological applications like protein networks. In recent years, big data is even more involved in our lives. One example is social networks like Facebook (figure 1.1) and LinkedIn (figure 1.2), and the other one is web-based purchasing networks like Amazon, which makes the importance of analysing the data in this field more obvious.

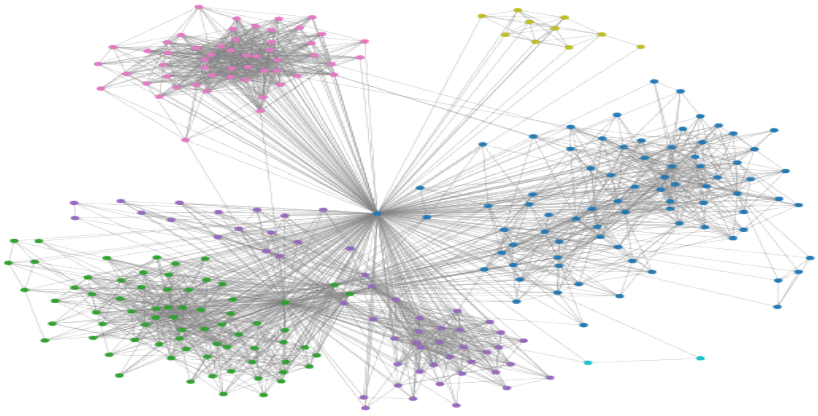


Figure 1.1: Example of friend connections in Facebook network graph [1] *"Colored dots (nodes) are friends and the lines (edges) are friendship connections. Since this is a Facebook friend network, everyone is connected to the central node (Facebook account owner)."* There are 6 different communities (clusters) in this example.

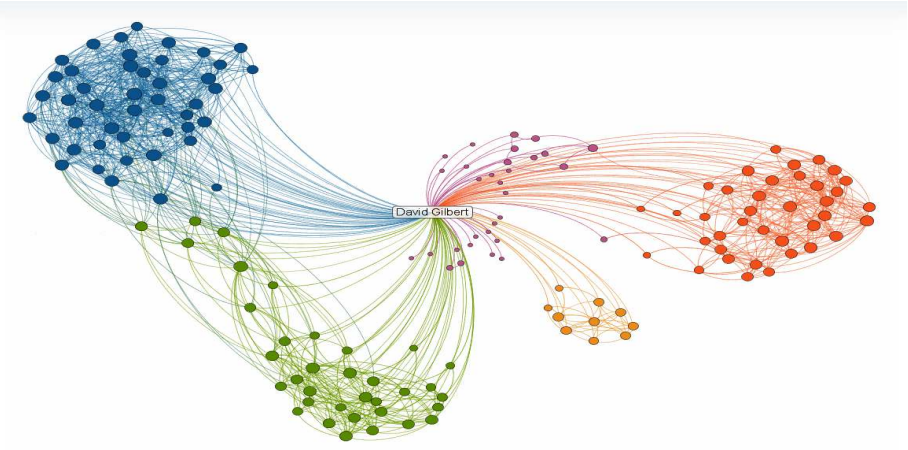


Figure 1.2: Example of different groups in LinkedIn network graph [2]
This is a subgraph of the Linkedin, showing a Linkedin user (the author of [2]) as the central node and his network of 163 connections (nodes). Each color (cluster) represents a group of his connections in the same geographical location.

Graph signal processing (GSP) is a method of representing and processing large data sets. The idea is to apply digital signal processing concepts and theories in graph domain [3]. In this form of data representation, the data points are the nodes (vertices) of the graph and the relation between the nodes is shown with the edges of the graph. For example in social networks, the users can be defined as graph vertices and the friendship between users as graph edges. Many similar scenarios can be analysed based on this strategy, such as speech recognition [4], communication networks [5] and road networks [6].

In general, the data can be analysed from different perspectives in the context of graph-based representation, such as graph learning, inpainting and clustering. In graph learning, we are dealing with a huge database, where the goal is figuring out the connections of the graph, which is referred to as graph adjacency matrix. Once the graph is learned, inpainting can be performed. In this kind of analyse, it is assumed that some of the node values are observed, and the goal is finding value of the rest of nodes. A novel approach for this kind of problem, which is also known as semi-supervised learning can be found in [7].

Another aspect that can be studied in this field is clustering data, such that each cluster is a group of the nodes with certain relations. In other words, we are trying to split the graph into sub-graphs. In this work, semi-supervised clustering is considered. Key assumption in semi-supervised clustering is that the class label for some of the nodes is known in advance and we exploit this information to classify the remaining nodes. Numerous approaches can be found for solving semi-supervised clustering problems from different perspectives. For example, in [8] spectral clustering perspective is applied. Harmonic functions point of view is considered in [9]. In [10], manifold regularization is proposed for the problem of pattern recognition. More ideas can be studied in [11], [12] and [13].

1.1 Objectives

In this work, the idea of incorporating error correcting codes to represent cluster labels in semi-supervised clustering algorithms is proposed. The goal is to increase clustering accuracy by means of error detection and correction capability of error correcting codes. We analyse the performance of this approach by applying error correction on two different semi-supervised methods.

First we apply semi-supervised spectral clustering and then integrate it with error correcting codes.

Next, we apply the same steps on the semi-supervised clustering based on harmonic functions. However, this idea can be applied on any kind of semi-supervised clustering method in general. Also this idea is applicable on multi-class (two or more) clustering schemes.

The idea of error correcting codes as cluster labels is previously applied in paper [14] for supervised learning methods. In our work, we extend the idea to semi-supervised multi-class clustering methods. Basically, it also can be considered as an extension of one-hot encoding for categorical data [15]. In one-hot encoding, as it will be explained in section 3.1, the goal is to represent the cluster labels by the sequence of bits, such that "1" is set in the bit position number of the cluster and "0" elsewhere. By applying it on all clusters, a codebook will be created, which is an identity matrix. We refer to this method as ordinary one-hot encoding. Then, by applying an error correcting approach, the rows of this codebook will be transformed to a higher domain. For example, in case of 4 clusters, where a 4×4 codebook is created by ordinary one-hot encoding, it can be transformed to a 4×7 codebook by applying a Hamming code (7,4) encoding [16] on top of that.

In our clustering approach, we exploit Stochastic Block Model (SBM) to generate a graph from a random model. This model is based on probabilities of having edges between different clusters and within the same cluster. After generating a random graph based on this model, we can apply our approach for different probability values for SBM. Also the performance will be analysed over a wide range of graph sizes with different number of clusters. Hamming codes are considered as an encoding method for our experiments.

As the final step, real data will be used as an input for our algorithm and the performance will be analysed.

It is expected that error correcting code on top of semi-supervised clustering algorithms outperforms the existing algorithms or at least represents the same level of accuracy.

Chapter 2

Background

An essential task in data analysis is clustering, which helps to see similarities between different data points. Basically, the goal is to group the nodes of the graph under some assumptions.

This chapter covers graph basics and clustering methods. More specifically, the concept of semi-supervised spectral clustering for multiple classes, which is the base of our approach, will be explained. Also some of the related algorithms in literature will be presented. In the last part, SBM is covered, which is used as our data model for simulating algorithms.

2.1 Graph Notation

A graph $\mathcal{G}$ is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V} = \{\mathcal{V}_1, ..., \mathcal{V}_n\}$ is the set of nodes or vertices, $\mathcal{E}$ is the edge set and $\mathbf{W}$ is the weight matrix of the graph, which contains the edge weights. If there is a similarity between nodes $\mathcal{V}_i$ and $\mathcal{V}_j$, there is an edge connecting those nodes. A given graph can be weighted or unweighted. Weighted graph means that there is a weight $\mathbf{W}_{ij} \geq 0$ for the edge connecting node $\mathcal{V}_i$ to $\mathcal{V}_j$. In this case, there is a similarity between the nodes i and j. If there is no edge, that means $\mathbf{W}_{ij} = 0$. In case there is no weight for the edges, a graph is called unweighted. Within this work, a graph is assumed to be undirected; that means $\mathbf{W}_{ij} = \mathbf{W}_{ji}$.

Basically, any graph is determined by its weight matrix, which is an $n \times n$ matrix (for n nodes). In case of unweighted graph (which is the case for our experiments throughout this thesis), we have $\mathbf{W}_{ij} = 1$ where there is a similarity between nodes and $\mathbf{W}_{ij} = 0$ elsewhere. Degree matrix and graph laplacian matrix are also matrices which are associated with any graphs.

2.2 Graph Laplacian Matrix

Before going into details of clustering methods, it is necessary to understand the concept of graph Laplacian matrix. In graph theory, graph Laplacian is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{2.1}$$

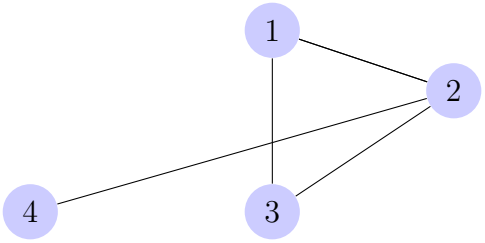


Figure 2.1: Unweighted graph with 4 nodes

where $\mathbf{D}$ is the degree matrix (diagonal) and $\mathbf{W}$ is the weight matrix. The degree matrix of a graph $\mathcal{G}$ is a diagonal matrix containing the degree of each node on its diagonal. A degree of a node is defined as the sum of weights of the edges that have incidence on that node. For example, consider an unweighted graph like figure 2.1 with 4 nodes. For this graph, the weight matrix will be as

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

For diagonal matrix $\mathbf{D}$, as the graph is unweighted, we just need to count the number of edges connected to each node. This leads to matrix $\mathbf{D}$ as

$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Based on that, Laplacian matrix $\mathbf{L}$ will be

$$\mathbf{L} = \mathbf{D} - \mathbf{W} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}.$$

Basically, $\mathbf{L}$ is a symmetric positive semi definite matrix. One important property of the graph Laplacian matrix is that we have equation

$$\mathbf{f}^T \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{n} \mathbf{W}_{ij} (\mathbf{f}_i - \mathbf{f}_j)^2,$$

for every vector $\mathbf{f} \in \mathbb{R}^n$ (proof in [17]). As we will see in the next section, this property plays an important role in relaxation of the graph cut optimization problem.

2.3 Graph Clustering

Most of the concepts of this section are derived from [17] which is an extensive tutorial on spectral clustering.

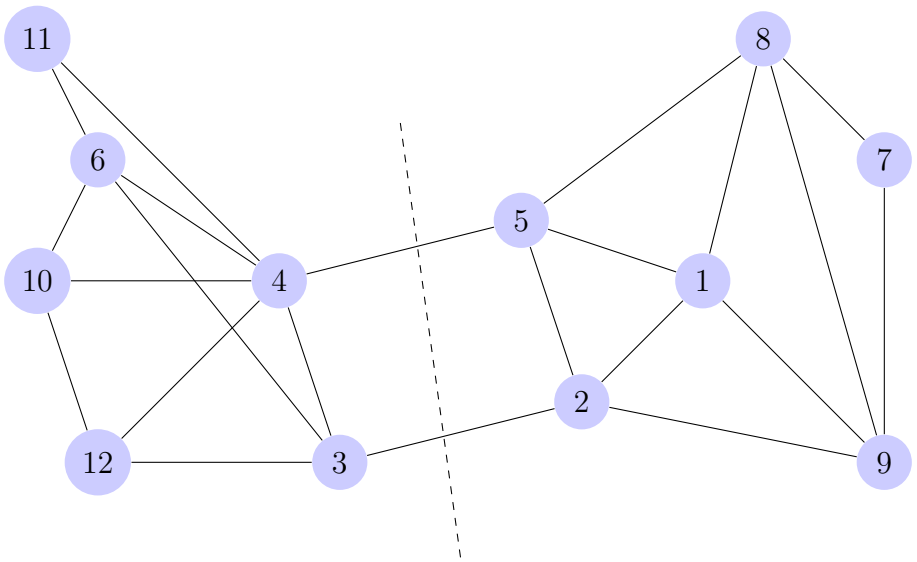


Figure 2.2: The idea of graph cut minimization for 2 clusters

One of the important tasks of graph-based big data analysis is to cluster the graph in order to group similar vertices (nodes). Spectral clustering is a powerful technique in this field, which has attracted a huge attention in recent years [18, 19, 20]. In what follows, we start from an intuitive idea to cluster the data points, which is the graph cut. Then concept of spectral clustering is presented.

2.3.1 Graph Cut

As mentioned in section 2.1, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ can be weighted with the weight matrix $\mathbf{W}$. The idea of graph cut is having a minimum number of edges between clusters, which is illustrated in figure 2.2 for two clusters. It can be seen in this figure that there is a minimum of two edges between two clusters. In order to cluster the graph based on graph cut concept, this graph is partitioned by a dashed line.

For a vertex set $\mathcal{V} = \{1, ..., N\}$, edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and edge weight (adjacency) matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$, graph cut for two clusters is defined (according to [21]) as

$$C(A_1, A_2) = \sum_{i \in A_1} \sum_{j \in A_2} W_{ij}.$$

$\mathcal{A}_1$ and $\mathcal{A}_2$ are two partitions of the vertex set, which means they are disjoint and their union is a full vertex set. For a set of K clusters, graph cut, which shall be minimized, is defined (according to [17]) as

$$C(A_1, ..., A_K) := \frac{1}{2} \sum_{i=1}^{K} \mathbf{W}(A_i, \overline{A}_i), \tag{2.2}$$

where $\overline{A}$ is a complement of A and the factor $\frac{1}{2}$ is introduced to avoid considering the edges twice. By defining the degree of a vertex $\mathcal{V}_i \in \mathcal{V}$ as

$$d_i = \sum_{j \in N} W_{ij},$$

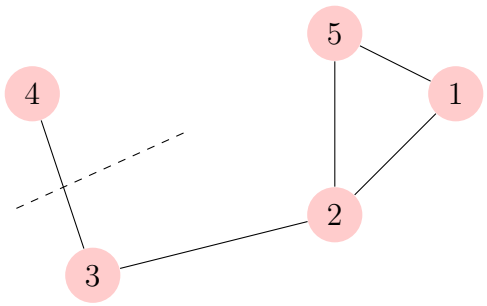


Figure 2.3: Disadvantage of graph cut

$W(A, \overline{A})$ for two sets A and $\overline{A}$ can be defined as

$$W(A, \overline{A}) = \sum_{i \in A} \sum_{j \in \overline{A}} W_{ij}.$$

In order to cluster based on the concept of graph cut, this equation shall be minimized. Therefore, for the case of K clusters, graph cut minimization is defined (according to [21]) as

$$\min_{A_1,...,A_K} \sum_{k=1}^{K} (\sum_{i \in A_k} \sum_{j \in A \setminus A_k} W_{ij}). \tag{2.3}$$

In general, a labeling function f can be defined for each node i as

$$f(i) = \begin{cases} 1 & i \in A_1 \\ -1 & i \in A_2 \end{cases},$$

in order to label the nodes with 1 and -1 as two cluster labels. As we will see in the chapter 3, graph cut point of view is a basis for clustering method based on optimization problem with linear constraints (algorithm 1).

2.3.2 Spectral Clustering

This section summarizes spectral clustering, which is based on [17] unless otherwise noted.
The disadvantage of minimization in graph cut perspective is that it separates one individual vertex from the rest of the graph. For example, in figure 2.3, node 4 is separated from the rest of the graph. The reason is that graph cut does not consider the cluster size.

To overcome this issue, the concept of Ratio Cut (RC) is defined as

$$RC(A_1, ..., A_K) := \frac{1}{2} \sum_{i=1}^{K} \frac{\mathbf{W}(A_i, \overline{A}_i)}{|A_i|}, \tag{2.4}$$

in order to balance the number of nodes in each cluster. With this optimization problem, in the ideal case RC is minimized when all the clusters have the same number of nodes. It can be shown that this minimization makes the problem NP-hard (Non-deterministic Polynomial problem) [22]. Therefore, in [23] some

relaxation on this problem is proposed, in order to get approximation of the solution. First the approximation on two clusters is applied, then it is generalized to multiple clusters.

As mentioned in previous section, we can define a labeling function for the nodes. For the case of two clusters (sets A and $\overline{A}$), N nodes and node set $\mathcal{V}$, it can be defined as a vector function $\mathbf{f} = (\mathbf{f}_1, ..., \mathbf{f}_N)^T \in \mathbb{R}^N$ with the entries

$$\mathbf{f}_i = \begin{cases} \sqrt{\frac{|\overline{A}|}{|A|}} & \mathcal{V}_i \in A \\ -\sqrt{\frac{|\overline{A}|}{|A|}} & \mathcal{V}_i \in \overline{A} \end{cases}. \tag{2.5}$$

If we sum up all the values of $\mathbf{f}_i$, that will be equal to zero, which indicates that the vector $\mathbf{f}$ is orthogonal to the vector of all ones (proved in section 5.1 of [17]). Also, we have the equation

$$\|\mathbf{f}\|^2 = \sum_{i=1}^{N} \mathbf{f}_i^2 = |A|\frac{|\overline{A}|}{|A|} + |\overline{A}|\frac{|A|}{|\overline{A}|} = |\overline{A}| + |A| = n \tag{2.6}$$

for the norm of f. Recall from section 2.2, $\mathbf{f}^T\mathbf{L}\mathbf{f}$ is related to Ratio Cut as

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{N} \mathbf{W}_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2 \propto RC(A, \overline{A}).$$

We can conclude that minimization of the RatioCut equation (2.4) is equivalent to minimization of $\mathbf{f}^T\mathbf{L}\mathbf{f}$ with linear constraints as

$$\begin{aligned} &\min_{\mathcal{A}\subset\mathcal{V}} \mathbf{f}^T\mathbf{L}\mathbf{f} \\ &\text{s.t.} \mathbf{f} \perp \mathbf{1}, \|\mathbf{f}\| = \sqrt{N} \end{aligned} \tag{2.7}$$

for the graph with N nodes, 2 clusters and Laplacian matrix $\mathbf{L}$. The challenge of NP hard problem still exists as it is a discrete optimization problem, where the vector $\mathbf{f}$ can only take two values in this case. We can relax this problem by allowing $\mathbf{f}$ to take any real values and solve the optimization problem

$$\begin{aligned} &\min_{\mathcal{A}\in\mathbb{R}^N} \mathbf{f}^T\mathbf{L}\mathbf{f} \\ &\text{s.t.} \mathbf{f} \perp \mathbf{1}, \|\mathbf{f}\| = \sqrt{N} \end{aligned} \tag{2.8}$$

instead. According to [24] (section 5.5.2) this optimization problem can be solved by Rayleigh-Ritz theorem, which leads to the eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}$. Finally, the last step is to assign a discrete value for each element of the vector $\mathbf{f}$, because of the relaxation. In case of two clusters, one simple way is to distinguish two clusters as

$$\begin{cases} \mathcal{V}_i \in A & \mathbf{f}_i \geq 0 \\ \mathcal{V}_i \in \overline{A} & \mathbf{f}_i < 0 \end{cases} \tag{2.9}$$

based on the sign of the values.

To generalize the idea to K clusters, the solution consists of K indicator vectors like $\mathbf{X}_{:j} = (\mathbf{X}_{1,j}, ..., \mathbf{X}_{n,j})^T$ with entries

$$\mathbf{X}_{i,j} = \begin{cases} \frac{1}{\sqrt{|A_j|}} & \mathcal{V}_i \in A_j \\ 0 & otherwise \end{cases}, \tag{2.10}$$

where $i = \{1, ..., N\}$ and $j = \{1, ..., K\}$. This way, we partition the vertex set $\mathcal{V}$ into K sets $\mathcal{A}_1, ..., \mathcal{A}_k$. Solution of the optimization problem will be the matrix $\mathbf{X} \in \mathbb{R}^{N \times K}$ with K indicator vectors. It can be shown that one property of the matrix $\mathbf{X}$ is that its columns are orthogonal to each other. In other words, we have $\mathbf{X}^T\mathbf{X} = \mathbf{I}$. Similar to the case of two clusters, minimization of the RC is equivalent to minimization of $\mathbf{X}_{:j}^T\mathbf{L}\mathbf{X}_{:j}$ for $j = \{1, ...K\}$. It leads us to the RC equation

$$RC(A_1, ..., A_k) = \sum_{j=1}^{K} \mathbf{X}_{:j}^T\mathbf{L}\mathbf{X}_{:j} = \sum_{j=1}^{K} (\mathbf{X}^T\mathbf{L}\mathbf{X})_{jj} = \mathrm{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) \tag{2.11}$$

for K clusters. That means minimizing the RC is equivalent to minimizing the trace of the matrix $\mathbf{X}^T\mathbf{L}\mathbf{X}$. Then the optimization problem can be written as

$$\begin{aligned} &\min_{A_1,...,A_k} \mathrm{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) \\ &\text{s.t.} \mathbf{X}^T\mathbf{X} = \mathbf{I} \end{aligned} \tag{2.12}$$

which is NP-hard and needs a relaxation. Therefore, any real values $\mathbb{R}^{N \times K}$ for the entries of the matrix $\mathbf{X}$ is acceptable. It leads us to the trace minimization

$$\begin{aligned} &\min_{\mathbf{X} \in \mathbb{R}^{N \times K}} \mathrm{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) \\ &\text{s.t.} \mathbf{X}^T\mathbf{X} = \mathbf{I} \end{aligned} \tag{2.13}$$

as our optimization problem. As mentioned for the case of two clusters, the solution consists of the K first eigenvectors of the matrix $\mathbf{L}$, which builds the matrix $\mathbf{X}$. Final step is to apply a clustering method (traditionally k-means algorithm) on the rows of the matrix $\mathbf{X}$, in order to find the cluster label of the nodes set.

As it will be explained in the chapter 3, in our proposed algorithm, one of the steps is to replace the k-means step by a decoder of error correcting codes.

2.3.3 Semi-Supervised Spectral Clustering

The key assumption in semi-supervised method is that some cluster labels are known. With this assumption, the goal is to '*split a dataset which is characterized by a graph into disjoint clusters*' [25]. In the framework of big data, one has to deal with a large amount of nodes, where only for a small number of nodes the label is known. In this work, two kinds of semi-supervised clustering methods, based on spectral clustering and harmonic functions are considered.

In the context of semi-supervised spectral clustering, by incorporating prior information, we are actually adding a linear constraint to the optimization problem in equation 2.13. So we need to solve the optimization problem

$$\begin{aligned} &\min_{\mathbf{X} \in \mathcal{R}^{N \times K}} \mathrm{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) \\ &\text{s.t.} \mathbf{X}^T\mathbf{X} = \mathbf{I}, \mathbf{X}_{\mathcal{L}:} = \mathbf{C} \end{aligned} \tag{2.14}$$

for the case, where we have a set $\mathcal{L}$ containing the nodes with the known clusters. Matrix $\mathbf{C}$ is defined to represent the respective clusters label in each row. As it will be presented in chapter 3, in our approach, each row is a codeword related to the individual cluster label and the number of columns is equal to the number of bits used.

2.4 Existing Methods in Error Correcting Codes for Multi-class Clustering Problem

As mentioned in chapter 1, the objective of this work is to incorporate error correcting codes to represent the cluster labels for semi-supervised clustering methods. The idea of employing error correcting codes for class labels is first used in [14], where it is applied on supervised learning methods. In supervised learning, we have a labeled training data and the goal is to estimate a function f(x) as a general solution for a new data x [14]. This could be very challenging for many cases, for example in speech recognition, where we would need really large amount of true labeled data in order to have an accurate result. Implementing error correcting code on top of this kind of supervised learning would make it even more challenging. For example in case of 15 bits codeword as an error correcting code for each cluster, we need to estimate the function f for each bit separately. In multiclass learning method introduced in [14], a codeword of the length n is assigned to each cluster as a distributed output representation. Then, a function f is learned for each of those n bits as a binary classifier. For a new example output data x (with n bits), calculating the function f for each bit of x would result in a n bit string as an output. Now the class, which has the minimum distance (minimum number of different bits) to this output, will be assigned as a true cluster label. This step can also be interpreted as a decoder of the error correcting code, where it can correct the bits which are in the wrong position. As a general rule, depending on the minimum Hamming distance d between cluster codewords, this approach can correct up to $\lfloor \frac{d-1}{2} \rfloor$ bits [14]. In general, the method of "*dividing one multi-class classification problem into a certain number of sub-problems of binary classifiers*" is called Error Correcting Output Code (ECOC) [26]. It is shown in [14] that the idea of ECOC in supervised learning outperforms the algorithms used to learn decision trees (based on the dataset from [27]) and neural networks.

The major difference between the ECOC algorithm in [14] and our work is that we are applying the algorithm on semi-supervised learning techniques. The advantage is that there is no need to have a whole set of labeled training data to estimate a function for a new data. Another difference of the approach introduced in [14] with our work, is that in [14] the idea is applied on binary classifier (function f) estimation, whereas we implement the idea of error correcting code on top of multi-class learning approaches. The assumption is that there is a small number of labeled data available as a prior knowledge for the problem. The details of the algorithm will be presented in the chapter 3.

The idea of ECOC strategy for multi-class learning is combined with Support Vector Machine (SVM) algorithm in [26] for speech recognition (ECOC-SVM method). In general, SVM is a method of learning binary classifiers by defining a seperating hyperplane [28]. The reason that SVM algorithm is chosen for speech

recognition is that in can work with small amount of training data [28]. Experimental results in [26] shows that ECOC-SVM method outperforms the traditional method for speech recognition based on Hidden Marcov Model (HMM).

One perspective of analysing ECOC is the design of the codewords. In [14] they formulated a property which should be satisfied for codeword design, according to which there should be no dependency between different rows and columns simultaneously. The dependency between different codewords is also analysed in [29] where it is concluded that "*The dependence among codeword bits reduces the error recovering power of ECOC*".

2.5 Stochastic Block Model

Before applying any kind of clustering algorithm, it is necessary to generate a graph and determine the weight matrix $\mathbf{W}$. In our work, the concept of random graphs is used, which is based on the probability of existing an edge in the graph. This idea initially introduced by Erdos-Renyi [30] called ER model for a graph with only one cluster. That means, only one probability (p) is defined for having edge between the nodes.

SBM is generalization of ER model for graphs with more than one cluster. According to [31], SBM model is defined by two probabilities: Probability of having edge between two nodes within the same cluster (p) and probability of having edge between two nodes that are from different clusters (q). It has been considered as a simplified model to present real networks [31]. In paper [32] a model is proposed for dynamic networks, which is defined as a network whose edges can be evolved over time, like social media networks.

In our work, SBM is used as our data model. The MATLAB implementation from [33] is applied in this work. For this model, first we define the number of clusters and size of each cluster. Assume we have defined a class membership vector $\mathbf{c}$, which classifies the graph into k clusters $\{c_1, ..., c_K\}$. According to [32], in order to generate a weight matrix based on SBM, two conditions must be fulfilled:

- *For any nodes $i \neq j$, the random variables $\mathbf{W}_{ij}$ are statistically independent.*
- *For any nodes $i \neq j$ and $i' \neq j'$, if i and i' are in the same class, i.e. $\mathbf{c}_i = \mathbf{c}_{i'}$, and j and j' are in the same class, i.e. $\mathbf{c}_j = \mathbf{c}_{j'}$, then the random variables $\mathbf{W}_{ij}$ and $\mathbf{W}_{i'j'}$ are identically distributed.*

For k clusters, the $k \times k$ matrix $\mathbf{P}$ is defined as a probability matrix, which contains the probabilities of forming edges between different clusters. From definition of the SBM, it is clear that the diagonal elements of the matrix $\mathbf{P}$ are equal to the probability value p and the off-diagonal elements are equal to the probability value q. According to the definition in [32], the elements of the weight matrix $\mathbf{W}$ can be either $\mathbf{W}_{ij} = 1$ if there is an edge between nodes i and j and $\mathbf{W}_{ij} = 0$ otherwise. In other words, its elements has a Bernoulli distribution, which takes the value 1 with probability p and 0 with probability q. As an example, consider a graph with 9 nodes and 3 clusters with class membership vector

$$\mathbf{c_0} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \end{bmatrix}^T.$$

Assuming a SBM setting as $p = 0.80$ and $q = 0.10$, the 3×3 probability matrix will be

$$\mathbf{P} = \begin{bmatrix} 0.80 & 0.10 & 0.10 \\ 0.10 & 0.80 & 0.10 \\ 0.10 & 0.10 & 0.80 \end{bmatrix}.$$

Based on this probability matrix, a random weight matrix $\mathbf{W}$ for the graph will be generated. As it will be presented in chapter 4, due to the randomness of our graph generating method in this work, the results are based on average behavior over a large number of simulations, so the results are reliable enough.

Chapter 3

Methodology

In this chapter, the proposed error correcting approach for semi-supervised clustering is presented. The idea is modifying semi-supervised clustering algorithms, such that there is less number of miss-clustered nodes in the algorithm output. To do so, cluster labels are presented by error correcting codes. Then, semi-supervised clustering method is applied by imposing the code-words, generated by applying error correcting codes to the known cluster labels. For the clustering output, the labels will be assigned to the data, based on their minimum distance to the known code-words.

Two semi-supervised clustering methods, which are used in our work, will be explained in the first section. The proposed error correcting approach is defined and explained in the second section. In the third section, the implementation of our approach is presented. Finally, the implementation steps are explained within an example.

3.1 Semi-Supervised Multi-Class Clustering Methods

In most of the cases, clustering methods lead to solving a quadratic optimization problem, like the one in equation (2.14) to find optimum values for class labels. For solving such optimization problem for clustering, we applied one spectral method based on linear constraints and one non-spectral method based on Harmonic function solution from [8] and [9] respectively.

3.1.1 Linear Constrained Optimization

First clustering method, used in our approach is based on spectral clustering. As explained in section 2.3, semi-supervised spectral clustering leads to minimization of the form

$$\begin{aligned} &\min_{\mathbf{X}\in\mathcal{R}^{n\times k}} \operatorname{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) \\ &\text{s.t.}\mathbf{X}^T\mathbf{X} = \mathbf{I}, \mathbf{X}_{\mathcal{L}:} = \mathbf{C}. \end{aligned} \tag{3.1}$$

One way of solving this optimization problem is to exploit the idea of projected power method which is introduced in [8]. In this paper, a method is proposed to solve the minimization of the form

$$\begin{aligned} &\min_{\mathbf{x}\in\mathcal{R}^{n\times k}} \mathbf{x}^T\mathbf{L}\mathbf{x} \\ &\text{s.t.}\|\mathbf{x}\| = 1, \mathbf{B}\mathbf{x} = \mathbf{c}, \end{aligned} \tag{3.2}$$

for any positive semi-definite matrix $\mathbf{L}$. The steps of finding the solution for minimization in equation (3.2) will be presented in implementation part in section 3.3. In this section, we will explain how it would be possible to extend the algorithm from vector to matrix and compute the minimization of the form of equation (3.1). With some modifications, this method is applicable on the matrix $\mathbf{X}$. From optimization problem in equation (3.1), it can be seen that for semi-supervised spectral clustering we have the condition that the norm of individual columns of $\mathbf{X}$ has to be equal to 1. In general, that's the condition for spectral clustering. With semi-supervised spectral clustering, additional constraint is imposed on some of the rows of $\mathbf{X}$ as shown in equation (3.1).
Consider an error correcting approach for the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with $\mathcal{V}$ and $\mathcal{E}$ as a node set and edge set respectively. We assume that the set $\mathcal{L} \subset \mathcal{V}$ contains the known labels. Therefore, for the corresponding rows of the matrix $\mathbf{X}$, we have the equation

$$\mathbf{X}_{l:} = \mathbf{C}_{i:},$$

where $l \in \mathcal{L}$ and $\mathbf{C}_{i:}$ is the corresponding codeword for the l^{th} node with $i \in \{1, ..., k\}$ for k clusters. With this optimization method, the goal is to calculate the remaining rows of the matrix $\mathbf{X}$ which is equal to $N - L$ rows.

As it can be seen in equation (3.2), the linear constraint $\mathbf{B}\mathbf{x} = \mathbf{c}$ is very similar to what we have in $\mathbf{X}_{l:} = \mathbf{C}_{i:}$ for the matrix $\mathbf{X}$. Therefore, if we want to write the equation (3.1) in the context of error correcting codes and similar to equation (3.2), then $\mathbf{c}$ is equivalent to one of the column vectors of the codebook $\mathbf{C}$ and $\mathbf{B}\mathbf{x}$ needs to be a sampling matrix, which selects the l^{th} row of the matrix $\mathbf{X}$ to be equal to the i^{th} codeword. We refer to the matrix $\mathbf{B}$ as selector. Essentially, matrix $\mathbf{B}$ has to be constructed such that we have

$$\mathbf{B}\mathbf{x} = \mathbf{C}_{:l}$$

for each column vector of $\mathbf{C}$. The construction of the codebook for error correcting approach will be explained in the next section. In this section, we just assume that the cluster labels are represented by a codeword, which are the rows of the codebook $\mathbf{C}$.

As mentioned in section 2.3.2, for spectral clustering, we have to take k smallest eigenvectors of Laplacian matrix of the graph for the case of k clusters. But if we perform the optimization with just one vector, which is the case in equation (3.2), we will get the smallest eigenvector. Therefore, the solution of this optimization problem will be one vector at the output. In order to generalize idea of the algorithm for solving equation 3.2 from vector $\mathbf{x}$ to matrix, the vectors of the matrix $\mathbf{X}$ should be computed in an iterative way, such that in each iteration one column vector of the matrix $\mathbf{X}$ (smallest eigenvector of the Laplacian matrix $\mathbf{L}$)

is calculated for the respected column vectors of the codebook $\mathbf{C}$ with the linear constraint $\mathbf{Bx} = \mathbf{C}_{:l}$. The iterations shall be continued until all column vectors of $\mathbf{C}$ are considered.

In order to make sure that the columns of the matrix $\mathbf{X}$ are not equal, one way is to modify the selector matrix $\mathbf{B}$, such that the solutions are orthogonal to each other. To do so, the solution of the previous iteration will be appended to the last row of the matrix $\mathbf{B}$ (we should take the transpose of the solution to convert it to the row vector). In order for matrix dimensions to be agreed, after appending the solution of the previous iteration, 0 is appended at the end of the corresponding column vector of $\mathbf{C}$ so that $\mathbf{BX}_{:j} = \mathbf{C}_{:j}$ holds true for all iterations $j \in \{1, ..., N\}$ with N being the number of columns of $\mathbf{C}$ (equal to the number of bits used for the codewords). This way, matrix $\mathbf{B}$ will be updated at the end of each iteration with the solution from previous iteration.

Another point is that in order to guarantee the orthogonality, we should use 1 and -1 for the matrices $\mathbf{C}$ and $\mathbf{X}$ instead of 1 and 0. The reason is that if we consider the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ as two successive solutions of the matrix $\mathbf{X}$, in order to get the orthogonality, the equation $\mathbf{v}_1^T\mathbf{v}_2 = 0$ must be fulfilled. If we use 1 and 0, then we would get strictly positive eigenvectors at each iteration, which would not be beneficial for having orthogonality.

Nevertheless, it turned out that enforcing orthogonality to $\mathbf{X}$ is not beneficial in general, because in practice the structure of matrix $\mathbf{C}$ can be different such that two consecutive columns are the same (for example in repetition code). In this kind of situation, at the second iteration the first and second vectors have the same labels, but in order to get orthogonality, they would need to have different values on all the other elements. That means, the second vector must be the inverted version of the first one so that they totally add up to zero. This would result in systematic errors, specially in case of repetition codes. However, this kind of situation might happen only in case of 0/1 encoding.

There is still some points that need to be considered when applying the algorithm from paper [8] to solve the minimization problem in our context. As it can be seen in equation (3.2), the solution is normalized to 1. It could make a problem to find the solution. For example, consider the extreme case where all the nodes are labelled in prior (although it is not the case that we are interested in). In that case, matrix B would be an identity matrix, which means all the nodes are labeled. As a consequence, column vector $\mathbf{x}$ will be equal to the respective column vector of the matrix $\mathbf{C}$. In such situation, column vector of $\mathbf{C}$ will have the norm equal to $\sqrt{n}$ with n being the total number of nodes. In this case, in order to have a norm 1 for our solution, we need to rescale the labels in $\mathbf{C}$. To do so, we divide the element of the matrix $\mathbf{C}$ to $\sqrt{n}$ so that it guarantees the norm 1 condition for the solution.

Another modification that is made on the algorithm, is enforcing an intended mean for each vector $\mathbf{x}$. The goal is to have an optimization solution close to the ground truth labels. As we have different mean of the ground truth labels (mean of the column vectors of matrix $\mathbf{C}$) for each iteration, it makes sense to enforce each vector of matrix $\mathbf{X}$ to have a mean of ground truth labels. As a consequence of that, for each iteration of the optimization, we have a different mean for the respective vector of $\mathbf{X}$ which is close to the ground truth labels. This mean is

calculated as

$$\overline{\mathbf{X}_{:j}} = \frac{\sum_{i=1}^{n} \mathbf{C}_{ij}}{n},$$

for n nodes and $j = \{1, ..., N\}$ where N is the number of bits.

The steps are explained in algorithm 1 for the graph with weight matrix $\mathbf{W}$ which is generated by SBM.

Algorithm 1: Solve $\min_{\mathbf{X} \in \mathcal{R}^{n \times k}} \mathrm{Tr}(\mathbf{X}^T \mathbf{L} \mathbf{X})$ s.t. $\mathbf{X}^T \mathbf{X} = \mathbf{I}, \mathbf{X}_{\mathcal{L}:} = \mathbf{C}$

Input: matrices $\mathbf{W}$,$\mathbf{B}$,$\mathbf{C}$

1. calculate laplacian matrix of the graph from $\mathbf{W} \rightarrow \mathbf{L}$
2. rescale matrix $\mathbf{C} \rightarrow \frac{\mathbf{C}}{\sqrt{n}}$
3. add intended mean for the solutions as a last row to $\mathbf{B}$
4. $j = 1$
5. ***repeat***
   solve equation 3.2 for $\mathbf{C}_{:j}$
   $j \leftarrow j + 1$
6. ***until*** j is equal to the number of columns of $\mathbf{C}$

Output: matrix X

Similar Approaches

In what follows, some related approaches in literature is presented. An alternative method to extend the algorithm in [8], from two classes to multiclass is studied in paper [34]. The algorithm is called *'Multi-Class Constrained Normalized Cut'* which is applied for object segmentation problem, where the goal is to partition an image into multiple segments based on the number of objects. Two different kinds of priors have been considered, namely Unary priors (for single elements) and pairwise priors (connection between the nodes). Also for each kind of prior, hard and soft constraints have been considered. In the context of unary priors, hard constraint indicates that whether a segment belongs to specific part of the object, and a soft constraint indicates a probability of belonging to some parts of the object. In the second type of the priors, pairwise priors, there could be '*must-link*' and '*cannot-link*' prior, which indicates that two specific segments belong to the same clusters or different clusters respectively. Traditionally, it was not possible to incorporate soft and cannot-link type of constraints into multiclass clustering optimization framework. The algorithm introduced in [34] is capable of handling multiclass problems with such priors. This can be considered as an alternative approach to algorithm 1 which we developed for multiclass clustering problems. One of the differences with our approach is that in [34], k smallest eigenvectors of the Laplacian matrix is computed, however in algorithm 1 the smallest eigenvector

is considered in each iteration.

Optimization problem with pairwise linear constraint is also studied in [35]. Again the goal is to incorporated the '*cannot-link*' type of constraints (in addition to '*must-link*' constraints) in order to develop an algorithm for multiclass spectral clustering. The idea which proposed in this paper, is to transform the data into spectral domain, then impose the constraints to the data in this transformed domain. Then, the ideal representation of the data is achieved based on the pairwise constraints. Based on the new representation in spectral domain, a cost function is defined. The optimization problem would be minimizing this cost function. The steps of this algorithm which is referred to as '*Constrained Clustering with Spectral Regularization*' is summarized in algorithm 1 of paper [35].

One of the limitations of the mentioned algorithms is that all of them are dealing with binary or pairwise constraints ('*must-link*' and '*cannot-link*'). Paper [36] introduces an approach for general type of constraints. Another assumption in [36] is that there is no need to satisfy all the constraints to solve the problem. Instead of that, they define a lower bound for constraints satisfaction. In other words, there is '*a flexibility in both representing and satisfying the constraints*' in [36]. In the next step, in order to solve the optimization problem, a group of possible solutions is derived. Out of this group, the solution which minimizes the optimization problem will be selected as a final solution. The steps of this approach can be studied in Algorithm 1 of paper [36].

Another approach for multiclass clustering based on graph cut optimization problem with linear constraints is studied in [37] for image segmentation problem. Lagrangian dual formulation is incorporated in this paper to solve the problem of minimizing the graph cut. For multiclass problem, constrained normalized cuts in the form of

$$N_{cut}^{k} = \sum_{l=1}^{k} \frac{C(A_l, V)}{assoc(A_l, V)}$$

is solved in an iterative way for k partitions. In each iteration, they decompose the optimization problem, so it ends up with a simple bipartition problem at the end of each iteration. This approach is studied and improved in paper [38], where '*Biased Normalized Cuts*' approach is proposed.

3.1.2 Harmonic Function Solution

There is an alternative approach to calculate columns of the matrix $\mathbf{X}$ based on [9]. In this paper, the problem of semi-supervised learning for the graphs with small amount of known labels is considered. The idea is to propagate nodes with the known clusters labels through unlabeled nodes in order to find the cluster label for unlabeled nodes based on the assumption that similar nodes are possessing similar labels. That's the motivation to perform energy minimization to solve the problem, because similar nodes need less energy to be connected to each other. The energy which should be minimized is (equation 2 in [9])

$$E(\mathbf{f}) = \frac{1}{2} \sum_{i,j} \mathbf{W}_{ij} (\mathbf{f}_i - \mathbf{f}_j)^2. \tag{3.3}$$

This is the energy which needed to connect an edge between the nodes i and j for the graph with known weight matrix $\mathbf{W}$. In this equation, $\mathbf{f}$ is a labeling

function for the nodes such that $f_i \in \{0, 1\}$ in case of 2 clusters for i in the node set $\mathcal{V}$. This minimization problem needs a relaxation as it is NP-hard to be computed. Therefore, the constraint to solve the problem is modified to $f_i \in \mathbb{R}$. According to paper [9], minimization of the equation (3.3) can be solved by exploiting '*Gaussian Random Fields*'. The point is that, minimization problem is equivalent to minimizing probability distribution of the labeling function $\mathbf{f}$, which leads to minimizing an exponential function as (based on section 2 of [9])

$$p(\mathbf{f}) \propto exp(-E(\mathbf{f}))|_{\mathbf{f}_l=L} = \exp\left(-\frac{1}{2}\sum_{i,j}\mathbf{W}_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2\right) \ |_{\mathbf{f}_l=L}, \tag{3.4}$$

where the condition $\mathbf{f}_l = L$ means that clusters labels are known for the node set $\mathcal{L} \subset \mathcal{V}$. It is shown in [9] that the function $\mathbf{f}$ which minimizes the distribution, satisfies the harmonic condition $\triangle\mathbf{f} = 0$ for the unlabeled nodes set $\mathcal{U}$. $\triangle$ is the Laplacian which is computed in the context of graphs as

$$\triangle_{UU} = \mathbf{D} - \mathbf{W}$$

That means, it is required to decompose the degree matrix $\mathbf{D}$ and weight matrix $\mathbf{W}$ into blocks as

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_{ll} & \mathbf{D}_{lu} \\ \mathbf{D}_{ul} & \mathbf{D}_{uu} \end{bmatrix}$$

and

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{ll} & \mathbf{W}_{lu} \\ \mathbf{W}_{ul} & \mathbf{W}_{uu} \end{bmatrix}$$

such that $\mathbf{W}_{uu}$, $\mathbf{W}_{ll}$ and $\mathbf{W}_{lu}$ means the weight of the edge connecting two unlabeled nodes, two labeled nodes and one labeled and one unlabeled nodes respectively. Also the vector function $\mathbf{f}$ needs to be decomposed into the labeled and unlabeled nodes as

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}_L \\ \mathbf{f}_U \end{bmatrix}.$$

So the goal is to find the minimum of the distribution of the unlabeled nodes with the condition on labeled nodes. This minimum happens at the mean of distribution where it has the minimum energy state. It can be calculated as (equation 5 of [9])

$$\mathbf{f}_u = (\mathbf{D}_{uu} - \mathbf{W}_{uu})^{-1}\mathbf{W}_{ul}\mathbf{f}_l. \tag{3.5}$$

Equation (3.5) is applied on all unlabeled nodes which leads to the vector $\mathbf{f_U}$. After calculating the vector $\mathbf{f_U}$, vector $\mathbf{f_L}$ (for the nodes with the known cluster labels) will be stacked on top of that, which creates one column vector of the matrix $\mathbf{X}$ for the optimization problem of equation (3.1).

As shown in equation (3.5), there is no iteration in this approach to find the solution. Main computational effort is a matrix inversion. So this algorithm is simpler and expected to be faster than the algorithm 1. The steps of finding each column are summarized in algorithm 2.

Algorithm 2: Compute real-valued function $\mathbf{f} : \mathcal{V} \rightarrow \mathbb{R}$ on graph $\mathcal{G}$

Input: $\mathbf{W}$, $\mathbf{f}_L$

1. Calculate degree matrix $\mathbf{D} \leftarrow \text{diag}(sum(\mathbf{W}))$
2. Decompose $\mathbf{W}$ and $\mathbf{D}$ in order to separate unlabeled nodes
3. Solve harmonic function equation 3.5 $\rightarrow$ Linear system of equations
4. Create the vector $\mathbf{f}$ by stacking $\mathbf{f}_L$ and $\mathbf{f}_U$ on top of each other

Output: Vector $\mathbf{f} = \begin{bmatrix} \mathbf{f}_L \\ \mathbf{f}_U \end{bmatrix}$

3.2 Error Correcting Code for Clustering

The key point of our proposed algorithm is using error correcting codes as cluster labels and creating a code-book. Basically, this algorithm can be considered as an extension of one-hot encoding for cluster labels. The concept of one-hot encoding and error correcting code on top of that will be explained in the following sub-sections.

3.2.1 Encoding with One Bit Per Cluster

In machine learning, in order to quantify categorical data, the method of encoding with one bit per cluster is applied, which is referred to as One-hot encoding [15]. Another example, where one-hot encoding is applied, is the unrelaxed version of the spectral clustering, which leads to a $N \times K$ matrix $\mathbf{X}$ with the entries as equation (2.10). For better understanding, a simple example from [15] will be explained. Consider a categorical data with three categories {'red','green','blue'}. In order to be able to work with most of the algorithms, these categories need to be converted to integer values, for example {1,2,3}. For some of machine learning algorithms, this ordered relationship between clusters is not acceptable. Therefore, transformation from integer encoding to one-hot encoding is necessary. In order to do so, integer values are replaced with the binary values, such that '1' is placed for the position of the cluster and '0' elsewhere. Table 3.1 illustrates this idea [15].

Table 3.1: One-hot encoding for cluster labels in categorical data

	CodeWord		
red	1	0	0
green	0	1	0
blue	0	0	1

As a consequence, instead of labeling clusters with {1,2,3}, we have a matrix with the size 3×3, where each row indicates a cluster label. It is in general a codebook as

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

We refer to this kind of encoding as ordinary one-hot encoding. In this kind of encoding, the Hamming distance between all clusters (rows) is 2 and it is not the maximum distance.

3.2.2 Higher Distance Compact Encoding

In case of 4 clusters, we could create the code-book $\mathbf{C}$, such that Hamming distance is higher and not the same for all clusters. For example, we could have Hamming distances of 3 as

$$\mathbf{C}_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

In general, the idea of maximizing the minimum Hamming distance can be applied to $k \geq 4$ clusters.

In our work, We refer to this modified encoding as Higher Distance Compact Encoding (HDCE). This codebook will then be transformed to higher dimension by applying error correcting codes. As we will see in the results, in order to have an improvement in clustering accuracy with error correcting codes, it is necessary to generate code-words with the maximum minimum Hamming distance.

3.2.3 Error Correcting Code Strategy

The idea of error correcting approach is to exploit code-book matrix $\mathbf{C}$ in ordinary one-hot encoding (or matrix $\mathbf{C_1}$ in HDCE) as our base data and transform it to higher dimension base. For example, if we use repetition code as our error correcting code and have the vector [1 0 0 0] associated to first cluster (first row of Code-book matrix), by applying repetition code, we will get the vector [1 0 0 0 1 0 0 0]. It will give us a matrix $\mathbf{C}$ with

- number of rows equal to the number of clusters
- and number of columns equal to the number of bits used for encoding.

That means at the output of the semi-supervised clustering algorithms, the clustered data will be transformed to higher dimensions. Nevertheless, repetition

codes are not recommenced to be applied, because there is no column separation in such encoding. As it will be explained in section 4.4, one of the codebook design requirements is having column separation as much as possible, which is not fulfilled in the codebooks based on the repetition code.

Assume that the output of the clustering algorithm is a $n \times k$ matrix $\mathbf{X}$ with n nodes and k clusters. By applying error correcting codes, matrix $\mathbf{X}$ will have higher dimension, such that the number of rows remains equal to the number of nodes, but the number of columns will be equal to the number of bits which is used for encoding. After applying clustering method, the rows of the matrix $\mathbf{X}$ shall be decoded based on the error correcting code applied.

In general, our approach consists of the following steps:

1. Specify the class membership vector $\rightarrow \mathbf{c}_0$
2. Create a code-book matrix based on the number of clusters $\rightarrow \mathbf{C}$ (or $\mathbf{C}_1$)
3. Generate a random graph (SBM in our case) based on the probability values p and q, as explained in section 2.5
4. Apply a semi-supervised clustering method based on the encoded known labels $\rightarrow$ matrix $\mathbf{X}$ as an output
5. Map the values of unlabeled nodes to real code-words by applying minimum distance decoding.

It is very important to choose a suitable clustering method, because we would have different performance for error correcting approach on top of different clustering methods. In this work, we analyse the performance of two semi-supervised clustering algorithms, which are explained in previous section.

3.2.4 Increasing the Number of Clusters with Fixed Bits

One aspect which should be considered when applying error correcting code is computational complexity of the clustering methods, which is increased by adding more bits. Therefore, in order to keep the complexity low, some experiments has to be performed to figure out the maximum amount of information that could be represented with the fixed number of bits. For this purpose, instead of increasing bits for the fixed number of clusters, we keep the number of bits to the fixed values, then we increase the number of clusters and evaluate the performance.

We experimented this aspect for 4 bits. Also Hamming code (7,4) has been applied as an error correcting method. The experiment setup and numerical result of this experiment will be presented in section 4.4. In this part, codebook design of the experiment will be explained. With 4 bits, there are $2^4 = 16$ possibilities for the datawords as

$$words = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

which will be transformed to a 16×7 matrix by applying Hamming code (7,4). That means there is a capacity for a maximum of 16 clusters. The starting point is choosing 4 cluster labels and creation of the codebook. After analysing the clustering performance with respect to this codebook, we add a new codeword from the remaining ones to the last row of the codebook to create a new one for 5 clusters. Then we continue this procedure until we achieve a codebook for which the error rate at the end of the clustering is more than 0.50. Based on that, we can estimate the number of clusters that can be packed in 4 bits with this specific settings.

In order to create the codebook, the design criteria introduced in [14] and [29] has been followed, based on which there should be minimum dependency between rows and columns of the codebook. With Hamming code (7,4) it is possible to achieve a maximum minimum Hamming distance of 4 between cluster labels for the case of 4 clusters. Based on that we constructed the codebook

$$\mathbf{C_4} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

At the next step, a 5^{th} cluster should be added to the last row of the matrix $\mathbf{C_4}$, such that the design criteria is still fulfilled. With 5 clusters the maximum minimum Hamming distance between cluster labels will be decreased to 3. As it will be explained in section 4.3, in order to satisfy the codebook design requirements, it is not feasible to choose the vectors of all zeros or all ones as cluster labels. Based on that, it remains 10 possibilities for the next codeword. Based on our experiments, adding the codeword [1 0 1 1 0 1 0] leads to lower number of errors among other possibilities, because it leads to the less dependency between columns. Based on

that, the codebook

$$\mathbf{C_5} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

was constructed for 5 clusters. The same strategy is applied for adding the next codewords. For the case of 6 clusters, the codebook

$$\mathbf{C_6} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

was constructed. As it can be seen, the maximun minimum Hamming distance is still 3. For 7 and 8 clusters, we constructed the codebooks $\mathbf{C}_7$ and $\mathbf{C}_8$ as

$$\mathbf{C_7} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and

$$\mathbf{C_8} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

respectively. Considering columns of the codebooks, we can see that correlation between the columns are being increased by adding a new codeword. For 9 clusters, adding the codeword [1 0 1 0 0 0 1] has been resulted in less errors among other possibilities. With this codeword, the codebook

$$\mathbf{C_9} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

was constructed for 9 clusters. Although the minimum Hamming distance of the codewords for all clusters from 5 to 9 was equal to 3, the correlation between column vectors of the codbooks increases by adding new codewords. As it will be illustrated in experimental results in section 4.5, for 9 clusters the number of errors is more than half of the total number of the nodes. Based on this results, we could say that for 4 bits, maximum number of 8 clusters can be achieved.

3.3 Implementation Based on Semi-Supervised Spectral Clustering

In this section, the minimization procedure for the equation (3.2) will be explained. In this work, two methods have been applied to compute this optimization problem. The first one was based on the algorithm introduced in [8] and the second one was based on the CVX toolbox of MATLAB. In what follows, both implementations will be presented and their performance will be compared.

3.3.1 Projected Power Method

The algorithm that is proposed in [8] is referred to as projected power method. It is proposed for an optimization problem of the form of equation (3.2) for a general semi-definite matrix $\mathbf{L}$. The idea is that they consider the constraint $\mathbf{Bx} = \mathbf{c}$ as a hyperplane which intersects with the sphere with $\|\mathbf{x}\| = 1$. The solution is computed in an iterative scheme. In each iteration, the vector $\mathbf{x}_j$ starting from the origin and ending on the intersection of the hyperplane and the sphere is considered as a valid solution. In order to update the solution in each iteration, the previous solution is multiplied with the matrix $\mathbf{L}$, then projected to the hyperplane $\mathbf{Bx} = \mathbf{c}$. After some normalization, the vector is added to $\mathbf{n_0}$ which is "*the vector from the origin to its projection on the hyperplane* $\mathbf{Bv} = \mathbf{c}$" according to [8]. This will produce a new solution for $\mathbf{x}$ which is greater than the previous one.

It is proven in [8] that this algorithm converges, and the solution is upper bounded by the maximum eigenvector of $\mathbf{L}$. As this projection method was originally intended for solving the maximization problem of the form

$$\max_{\mathbf{x}} \mathbf{x}^T \mathbf{L} \mathbf{x},$$

in paper [8] it is recommended to define a large α so that our original minimization problem is equivalent to solving the maximization problem

$$\max_{\mathbf{x}} \quad \mathbf{x}^T (\alpha \mathbf{I} - \mathbf{L}) \mathbf{x}.$$

In our work, we set α to the value of the maximum eigenvalue of the matrix $\mathbf{L}$.

This way, it is also shown that the solution of the minimization problem is basically the minimum eigenvector of the matrix $\mathbf{L}$. For more details regarding geometrical point of view of the algorithm refer to [8].

All steps of this algorithm are summarized in Algorithm 3. The matrices $\mathbf{L}$, $\mathbf{B}$ and $\mathbf{C}$ are the Laplacian matrix, sampling matrix and codebook respectively. This algorithm shall be applied in every iteration of the algorithm 1 (introduced in section 3.1.1) to compute a vector of the matrix $\mathbf{X}$.

Algorithm 3: solve $\min_{\mathbf{x}} \ \mathbf{x}^T\mathbf{L}\mathbf{x}$ s.t. $\|\mathbf{x}\| = 1, \mathbf{B}\mathbf{x} = \mathbf{c}$

Input: matrices $\mathbf{L}, \mathbf{B}, \mathbf{C}$

1. Calculate maximum eigenvalue of $\mathbf{L} \Rightarrow \alpha$
2. $\mathbf{P} = \mathbf{I}$ - $\mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B}$, $j = 0$
3. $\mathbf{n_0} = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{c}$
4. $\gamma = \sqrt{1 - \|\mathbf{n_0}\|^2}$
5. $\mathbf{x_0} = \gamma \frac{\mathbf{P}\mathbf{L}\mathbf{n}_0}{\|\mathbf{P}\mathbf{L}\mathbf{n}_0\|} + \mathbf{n_0}$
6. ***repeat***
   $\mathbf{u_{j+1}} = \gamma \frac{\mathbf{P}\mathbf{L}\mathbf{x_j}}{\|\mathbf{P}\mathbf{L}\mathbf{x_j}\|}$
   $\mathbf{x_{j+1}} = \mathbf{u_{j+1}} + \mathbf{n_0}$
   $j = j + 1$
7. ***until*** $\mathbf{x}$ converges

Output: column vector $\mathbf{x}$

The idea of power method is also applied in [39] for estimating eigenvectors in order to overcome numerical complexities.

3.3.2 CVX Toolbox

As an alternative way to solve the optimization problem (3.2) is to apply convex optimization toolbox of the MATLAB (CVX). According to [40], *"CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard MATLAB expression syntax"*. We need only to write down the problem we have, including the constraints exactly as it is shown in algorithm 4.

Algorithm 4: solve $\min_{\mathbf{x}} \ \mathbf{x}^T\mathbf{L}\mathbf{x}$ s.t. $\|\mathbf{x}\| = 1, \mathbf{B}\mathbf{x} = \mathbf{c}$

Input: $\mathbf{L}, \mathbf{B}, \mathbf{c}$

```
cvx-begin
variable x
minimize(x' * L * x)
subject to
B * x == c
cvx-end
```

Output: column vector $\mathbf{x}$

The advantage of applying CVX toolbox is the simplicity of this method. We

don't need to implement those iterations in algorithm 1. There is an engine which is picking the reasonable optimizer once we run that. However, the simulation time is significantly higher than implementation of algorithm 3. That's the reason we perform the implementation based on *"Projected Power Method"* in our simulations.

3.4 Implementation Steps on a Toy Example

In order to understand the idea of error correcting approach, a simple example including all steps will be explained in this part. As in our work, we generate the graph by SBM, the graph which shall be clustered is totally random and we have a redundancy in cluster labels. We need to specify probability values for SBM model, as well as number of known labels. For example, consider a graph with 200 nodes, 4 clusters and 5 known labels in each cluster. Also assume that following setting is defined for the SBM model:

- Probability of having edges between nodes within the same cluster is $p = 0.65$
- Probability of having edges between nodes of two different clusters is $q = 0.35$

We also need to specify class membership vector as an input to generate a graph with SBM model. For simplicity, we assume that 4 clusters have the same size. That means the class membership vector $\mathbf{c}_0$ will be a 200 by 1 vector as

$$\mathbf{c}_0 = \Big(\underbrace{1 \ \dots \ 1}_{50\times1} \quad \underbrace{2 \ \dots \ 2}_{50\times1} \quad \underbrace{3 \ \dots \ 3}_{50\times1} \quad \underbrace{4 \ \dots \ 4}_{50\times1} \Big)^T .$$

Our goal is to label the rest of the nodes with having miss-classified nodes as least as possible. First step is to encode cluster labels and create a code-book matrix. In this work, all simulations are implemented in MATLAB.

As we have 4 clusters in this example, we can apply Hamming code (7,4) [16]. We perform two simulations, first based on one-hot encoding with 4 bits and in the second simulation we apply Hamming code encoding to create the code-book with 7 bits for each cluster label. If we apply HDCE, the output of the encoding will be (following the design criteria explained in section 4.4)

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} cluster1 \\ cluster2 \\ cluster3 \\ cluster4 \end{pmatrix},$$

which will have a Hamming distance of maximum 4 in the Hamming encoded code-book. As a next step we create the selector matrix $\mathbf{B}$ based on the position and the number of known labels in each cluster in order to define a linear constraint $\mathbf{Bx} = \mathbf{c}$ in our optimization problem. We assume that we know the first 5 labels

of each cluster. That makes the 200 by 4 (number of bits) matrix

$$\left[\begin{array}{cccc}
-1 & -1 & -1 & 1 \\
\vdots & \vdots & \vdots & \vdots \\
-1 & -1 & -1 & 1 \\
\cdots & & \cdots & \\
\cdots & & \cdots & \\
-1 & -1 & 1 & -1 \\
\vdots & \vdots & \vdots & \vdots \\
-1 & -1 & 1 & -1 \\
\cdots & & \cdots & \\
\cdots & & \cdots & \\
-1 & 1 & 1 & 1 \\
\vdots & \vdots & \vdots & \vdots \\
-1 & 1 & 1 & 1 \\
\cdots & & \cdots & \\
\cdots & & \cdots & \\
1 & -1 & -1 & 1 \\
\vdots & \vdots & \vdots & \vdots \\
1 & -1 & -1 & 1 \\
\cdots & & \cdots & \\
\cdots & & \cdots &
\end{array}\right]
\begin{array}{l}
\} 5 \times known \\
\} 45 \times unknown \\
\} 5 \times known \\
\} 45 \times unknown \\
\} 5 \times known \\
\} 45 \times unknown \\
\} 5 \times known \\
\} 45 \times unknown
\end{array}$$

as the matrix $\mathbf{X}$. At this point, it is the task of semi-supervised clustering method to calculate those unknown values. For this example, we applied algorithm 1 from linear constrained optimization method as a function in MATLAB. It calculates the matrix $\mathbf{X}$ such that all the unknown nodes will be a real value between -1 and 1.

In the next step, those calculated values will be rounded to +1 and -1. Then we map them to the real code-words by using minimum distance decoder. This is our soft-decoding step. Finally, we need to convert this soft-decoded output to discrete values (from 1 to 4) and compare it to the class membership vector $\mathbf{c}_0$. Then we count the number of miss-classified nodes. In this work, we applied the formula

$$\text{AE} = \frac{\text{number of miss-classified nodes}}{\text{total number of simulations}} \tag{3.6}$$

to derive Average Error (AE). For this particular example, if we do the simulation 10 times, it gives AE = 43 with one-hot encoding.

Now if we want to apply Hamming code encoding on top of one-hot encoding, all the steps are the same except the first step. As mentioned, we applied Hamming code (7,4). Therefore, the code-book matrix $\mathbf{C}$ would be as

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{pmatrix} cluster1 \\ cluster2 \\ cluster3 \\ cluster4 \end{pmatrix}.$$

After applying the same steps as before and running the simulation for 10 times, it produces the AE = 24, which shows a great performance of error correcting codes. In the next chapter, simulation results for graphs with large number of nodes will be presented.

Chapter 4

Results

Numerical experiments from different perspectives are performed and the results are presented in this chapter. Hamming code is applied as error correcting code. The results of following experiments will be shown in this chapter:

- Ordinary one-hot encoding
- Hamming code on top of one-hot encoding
- HDCE
- Hamming code on top of HDCE
- Comparison of different minimum Hamming distances
- Increasing number of clusters for the same bits
- Performance analysis over real data-set

These experiments are done for different number of nodes and clusters, different SBM settings and applied on different optimization methods. All simulations are done in MATLAB R2016b on a PC with 1.8 GHz CPU and 8 GB RAM. For SBM model, MATLAB function from [33] is used.

4.1 Constrained Optimization Problem

As a first experiment, clustering algorithm 1 is applied with one-hot encoding and Hamming code on top of that. The performance is compared based on Empirical Cumulative Distribution Function (ECDF) plots for 4 and 10 clusters. Also the AE against wide range of SBM settings and number of nodes is studied (for the case of HDCE and Hamming code on top). The range of p and q values in SBM settings is defined such that the error rate is between 5 and 25 percent.

4.1.1 Ordinary One-Hot Encoding

We consider a graph generated with 1000 nodes. The total number of simulations is $M = 500$. Number of known cluster labels in each cluster is assumed to be 5 and 10 for the case of 4 and 10 clusters respectively. The SBM settings are defined as

- p = 0.55 , q = 0.35 for $k = 4$ clusters
- and p = 0.55 , q = 0.20 for $k = 10$.

It can be seen in figures 4.1 that ECDF curve of the number of errors for both cases one-hot and Hamming code encoding is roughly the same, so there is no improvement by Hamming code in this case. The reason of this behavior is that there is only Hamming distance of 2 between codewords for ordinary one-hot encoding scheme (as presented in section 3.2.1). However, one main design criterion when applying error correcting code is having row separation in codewords as much as possible according to paper [14]. Another point is the error correction capability of Hamming codes. As discussed in section 2.4, there is a general rule which states: for the minimum Hamming distance of d, up to $\lfloor \frac{d-1}{2} \rfloor$ bits can be corrected with Hamming codes. Considering this general rule, it makes sense that ordinary one-hot encoding with $d = 2$ doesn't have error correcting capability. Nevertheless, from figures 4.1 and 4.2 it is obvious that the performance with error correcting codes doesn't get worse than one-hot encoding. Specially from figure 4.2 it can be seen that even in case of ordinary one hot encoding for 10 clusters there is still a little improvement with Hamming code encoding. There are two reasons for having an improvement for the case of 10 clusters:

1. For 10 clusters, we have defined more optimum values for SBM setting (q value) in order to keep the error rate in a range of 5 to 25 percent.

2. For the case of 10 clusters, there are 10 known cluster labels in each cluster which means 100 known labels in total, whereas for the case of 4 clusters there are 5 known labels in each cluster or 20 known labels in total.

In general, it is better to apply the Hamming code with minimum number of bits, because of computational complexity of clustering algorithms for higher bits. However, for the case of 10 clusters, it was required to apply the Hamming code (15,11) as it is not possible to encode 10 clusters with 7 bits (with one-hot encoding).

It should be noted that, as we are dealing with random graphs, by increasing the number of clusters and keeping the SBM settings unchanged, the algorithm is prone to produce more errors. Therefore, in order to have a trade off between number of clusters and SBM settings, for the case of 4 clusters we set the SBM settings to $p = 0.55$ and $q = 0.35$, then we update the settings to $p = 0.55$ and $q = 0.20$ for 10 clusters. This point will be studied in section 4.5, where we increase the number of clusters by keeping both SBM settings and the number of bits unchanged.

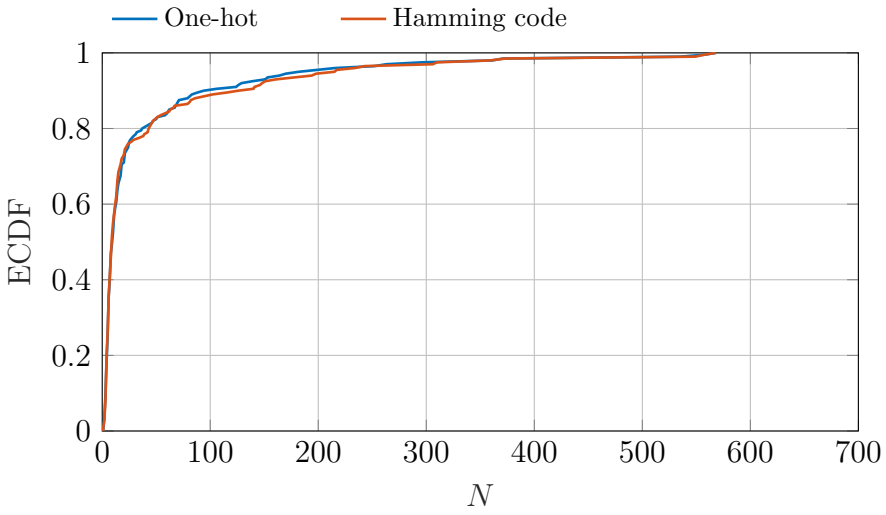


Figure 4.1: ECDF curve of the number of errors (N) for ordinary one-hot encoding and Hamming code on top for k=4 clusters.

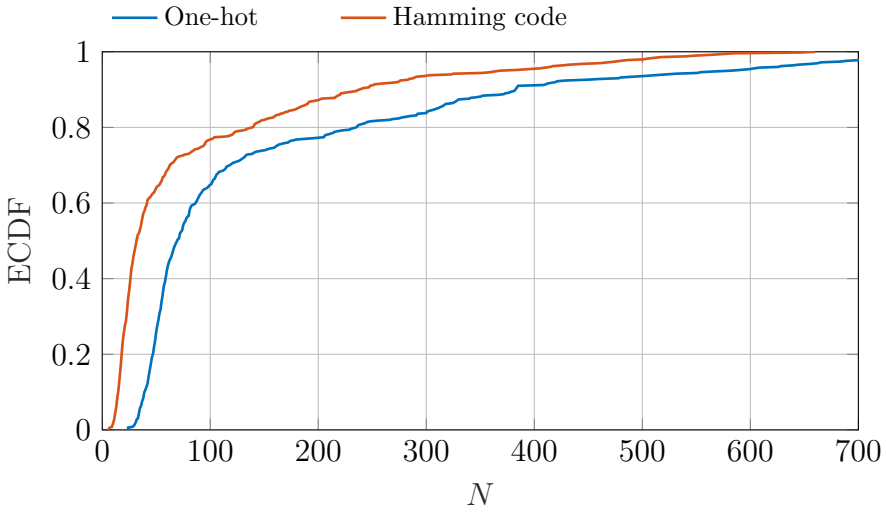


Figure 4.2: ECDF curve of the number of errors (N) for ordinary one-hot encoding and Hamming code on top for k=10 clusters.

4.1.2 Higher Distance Compact Encoding

By applying HDCE, minimum Hamming distances of 4 could be achieved in Hamming code (7,4) encoded clusters. This way, we could achieve error detection of up to 2 bits and error correction of 1 bit. Therefore, it is expected to have a better performance in terms of ECDF curves by applying Hamming code on top of of HDCE. Figures 4.3 and 4.4 proves this improvement. The settings of the experiments are set the same as previous section in order to have a meaningful comparison.

Since the graphs are generated randomly with SBM model, it is interesting to analyse the performance over different SBM settings. The results are presented for $k = 4$ clusters. First we experiment the behavior in terms of AE against p value of the SBM settings. To do so, we keep the $q = 0.35$ unchanged and perform the simulations for a range of p values. All other experiment settings are the

same as before. As it is expected, the AE is decreasing by increasing p values. The complementary experiment is performed for a range of q values by keeping p unchanged. For this case, AE is increased by increasing q. Nevertheless, it can be seen in figures 4.5 and 4.6 that there is a boundary for each SBM settings for which the algorithms perform well. If we fix the probability of having edge between clusters to $q = 0.35$, then according to figure 4.5, p value in SBM setting shouldn't be less than 0.50. For the second case, where we fix $p = 0.55$, the q value shouldn't be larger that 0.40. It can be concluded that there is a good flexibility in terms of SBM settings for the case of clustering with algorithm 1.

As a next step, the performance is studied with respect to the number of the nodes. For this case, Node Error Rate (NER) is considered, so the number of errors is divided by the total number of the nodes. This experiment is performed for 4 clusters and SBM settings as $p = 0.55$ and $q = 0.35$ and $M = 500$ simulations. Number of the known cluster labels is 5 in each cluster. As it can be seen in figure 4.7, the error rate is decreasing. In general The reason for this behavior is that we are dealing with random graphs generated by SBM model. It shows that with this SBM settings, for smaller number of nodes there is higher uncertainty in graph generation with specified number of clusters, whereas for the bigger graphs, the results are less dependent on SBM settings, therefore it has less uncertainty. However, this decreasing starts at some point depending on the clustering algorithm applied for the experiment. As it is illustrated in figure 4.12, where algorithm 2 is applied, there won't be error rate decreasing behavior until the end, where it just starts showing flat behavior.

It is worth mentioning that according to figure 4.7 error correcting code outperforms HDCE for all number of the nodes, which is the expected behavior.

Comparing figures 4.4 and 4.2, there is an improvement with Hamming code on top of both encoding schemes (one-hot and HDCE), however with HDCE, the number of errors is substantially lower.

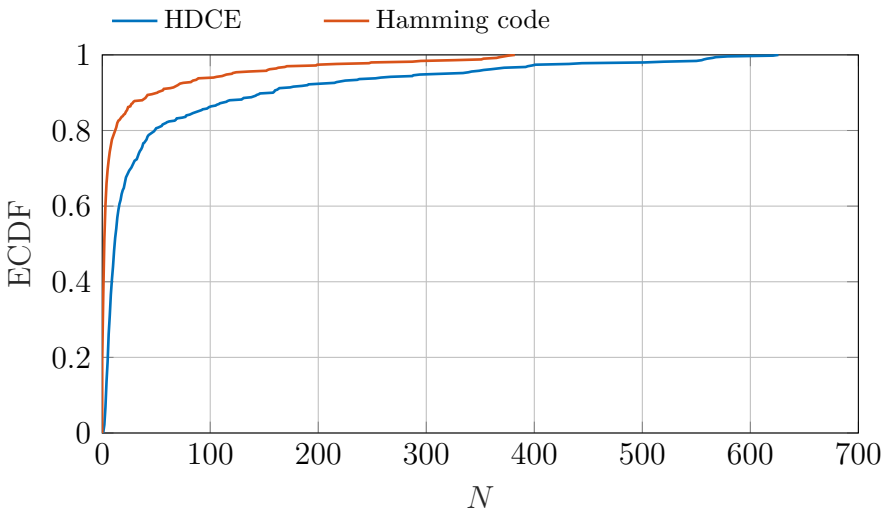


Figure 4.3: ECDF curve of the number of errors (N) for HDCE and Hamming code on top for k=4 clusters.

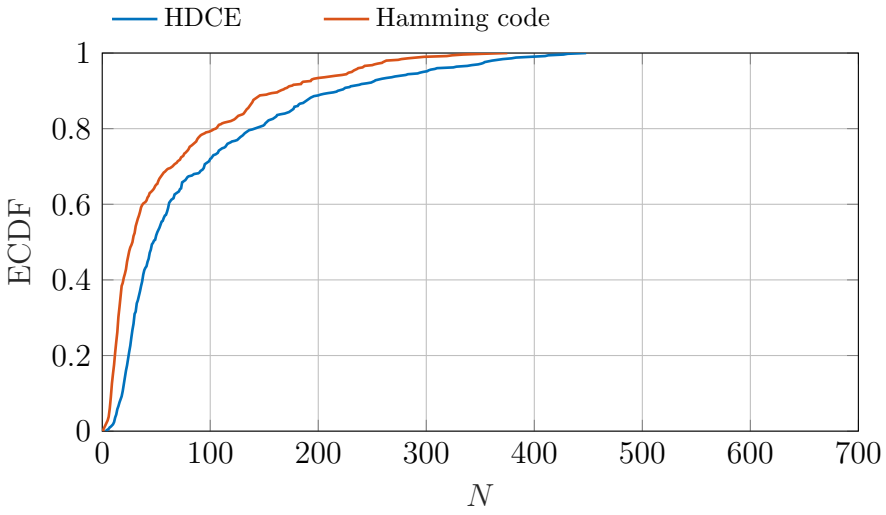


Figure 4.4: ECDF curve of the number of errors (N) for HDCE and Hamming code on top for k=10 clusters.

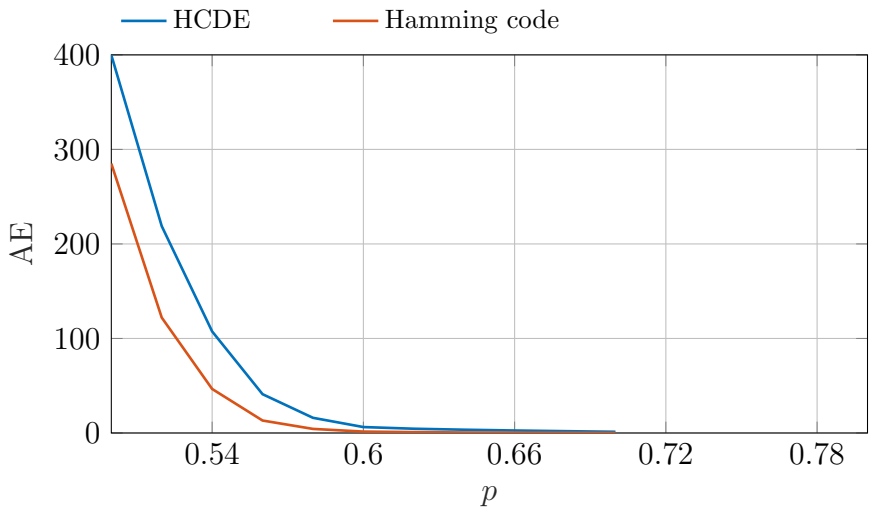


Figure 4.5: Comparison of HDCE and Hamming code in terms of AE against the probability of having edge within the clusters (p). Probability q is fixed to 0.35 and the graph is generated with 1000 nodes and k=4 clusters.

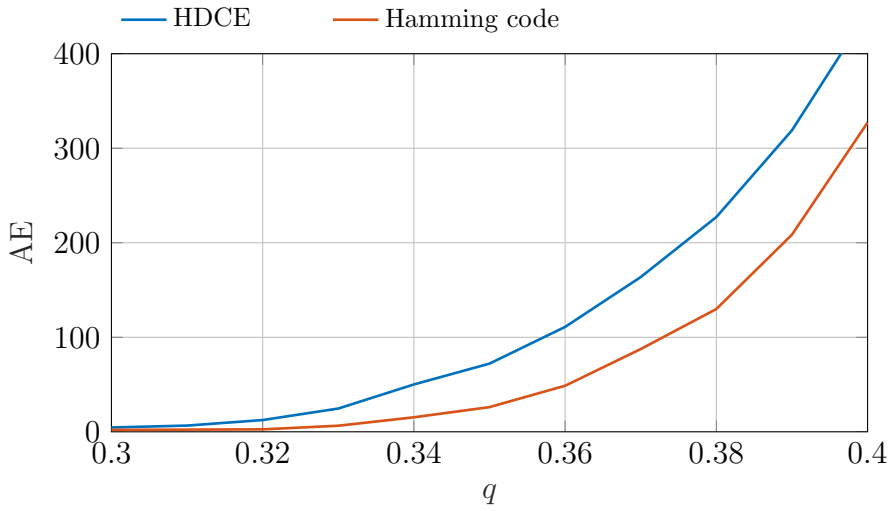


Figure 4.6: Comparison of HDCE and Hamming code in terms of AE against the probability of having edge between the clusters (q). Probability p is fixed to 0.55 and the graph is generated with 1000 nodes and k=4 clusters.

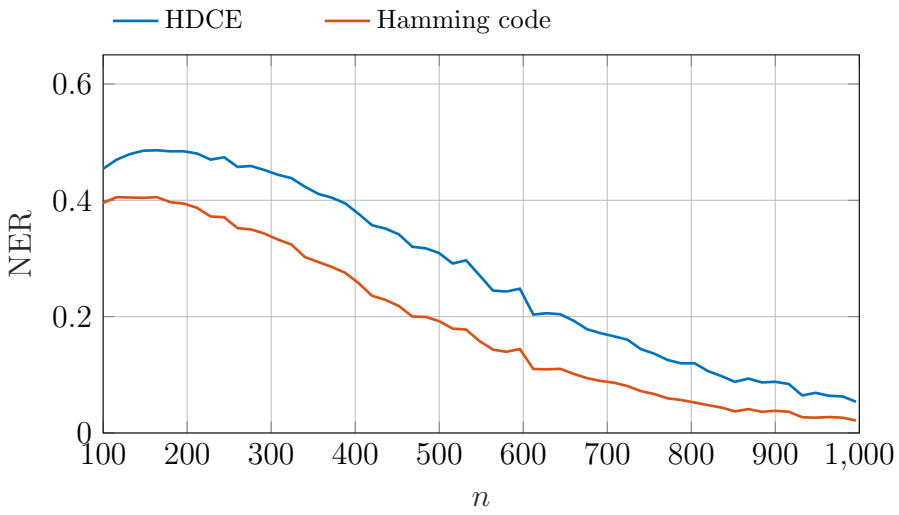


Figure 4.7: Comparison of HDCE and Hamming code in terms of NER against the number of nodes for k=4 clusters. SBM setting: p=0.55 and q=0.35. Clustering algorithm 1 is applied.

4.2 Harmonic Function Solution

Implementation of the error correcting algorithm based on clustering with harmonic function is illustrated in this section. Similar to previous case, with ordinary one hot encoding there is not much improvement comparing to the HDCE. Therefore, only the results of HDCE and Hamming code on top of that are presented.

The graphs are created with 1000 nodes and total number of simulations was $M = 500$. Similar to previous case, number of known cluster labels in each cluster was 5 and 10 for $k = 4$ and $k = 10$ clusters respectively. As it is shown in figures 4.8(a) and 4.9(a), with the same SBM settings as applied for algorithm 1, the behavior is not optimum and the number of errors are not in a reasonable range (range of having error rates between 5 and 25 percent), although Hamming code improves the performance. The reason is that with clustering algorithm 2, there is no iteration to improve the solution of the optimization problem. This is the reason for having weak behavior for clustering based on algorithm 2 comparing to algorithm 1. Therefore, in figures 4.8(b) and 4.9(b), the SBM settings are improved in order to obtain the same behavior as algorithm 1.

The performance over different SBM settings for this clustering algorithm is also studied. The results are presented for 4 clusters. Similar to the clustering algorithm 1, the results with error correcting codes over HDCE is promising from all perspectives. The figures are plotted for 200 simulations. Other experiment settings remained unchanged.

The performance against number of nodes in figure 4.12 shows the fact that clustering based on harmonic function is very sensitive to SBM settings. As explained in section 4.1.2, the error rate curve in figure 4.12 is flattening at the end, so the shape is similar to the figure 4.7 in general, but the difference is that the error rate is not decreased at the beginning of the curve like figure 4.7.

Based on the experiments, algorithm 1 and algorithm 2 can be compared from two perspectives: boundaries and computational complexity.

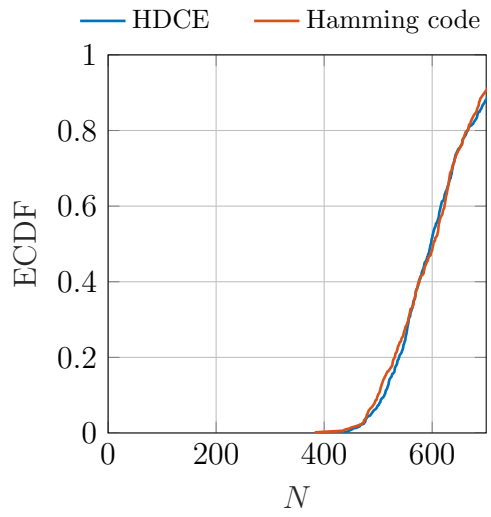
Boundaries

Boundaries are compared in terms of SBM settings for which the algorithm works without producing high amount of errors. Comparing figures 4.5 and 4.10, it can be seen that algorithm 1 runs into trouble for probability values p less than 0.50, whereas for algorithm 2 this threshold is 0.70, which means there is more flexibility for p values of SBM settings with algorithm 1. Performing the same comparison between figures 4.6 and 4.11 for q values shows the same conclusion, because the q value must be upper bounded by 0.40 for algorithm 1 and by 0.30 for algorithm 2.

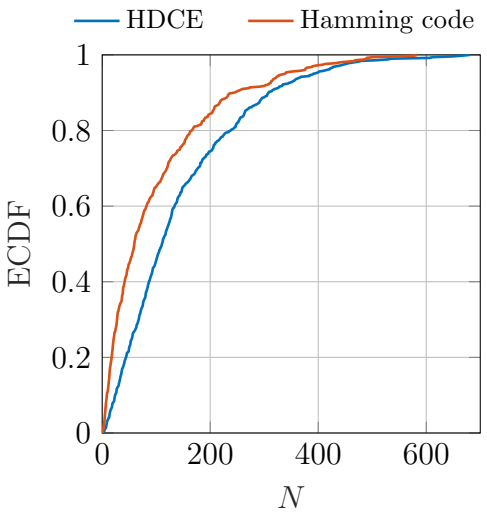
Computational complexity

According to [8], algorithm 1 has complexity of $O(n^3)$ for computing each column vector of the matrix $\mathbf{X}$. Assuming the number of bits for the codewords equal to m, the computational complexity of algorithm 1 is $O(mn^3)$, without considering the bound on the number of iterations.

The computational complexity of algorithm 2 involves matrix inversion which is performed on m column vectors of the matrix X. That means the complexity is $O(mn^3)$. However, this algorithm runs much faster than algorithm 1. The reason is that the algorithms for Harmonic Function Solution are implemented in C which is much faster than an implementation in MATLAB.

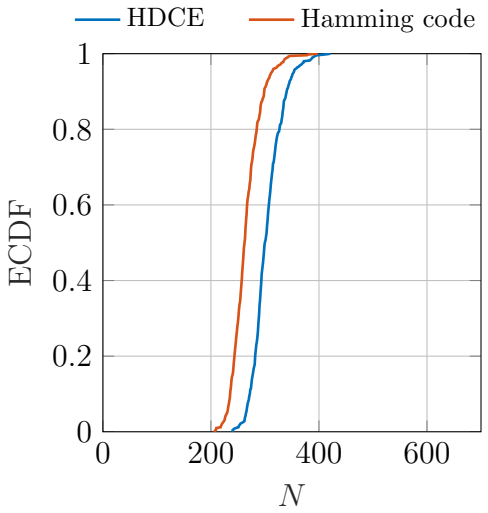


(a) SBM setting: p=0.55 , q=0.35

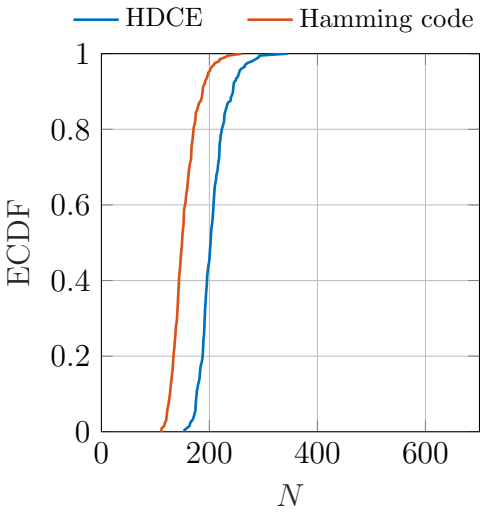


(b) SBM setting: p=0.80 , q=0.20

Figure 4.8: k=4 clusters performance. Comparison of the performance for clustering algorithm 2 (based on harmonic function) for two different SBM settings with respect to ECDF curve for number of errors (N).



(a) SBM setting: p=0.55 , q=0.20



(b) SBM setting: p=0.75 , q=0.25

Figure 4.9: k=10 clusters performance. Comparison of the performance for clustering algorithm 2 (based on harmonic function) for two different SBM settings with respect to ECDF curve for number of errors (N).

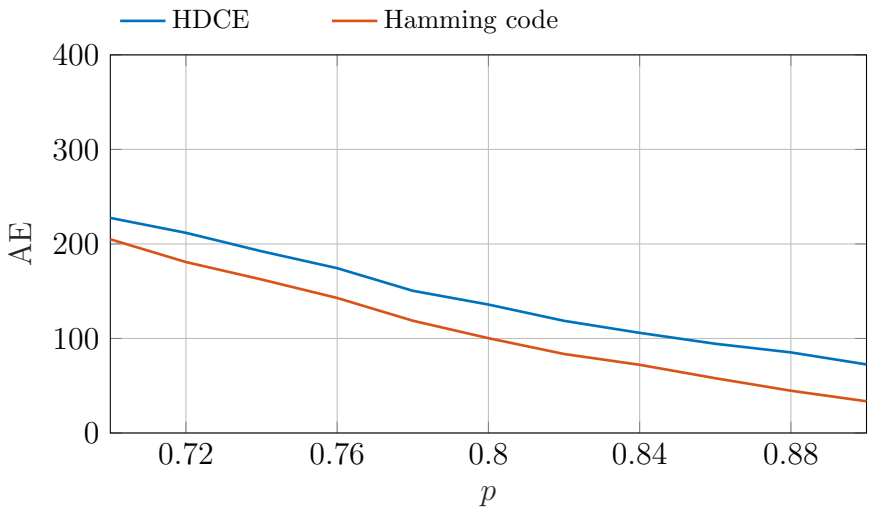


Figure 4.10: Comparison of HDCE and Hamming code in terms of AE versus probability of having edges within the same clusters (p) for k=4 and q=0.20. Clustering is based on harmonic function (Algorithm 2).

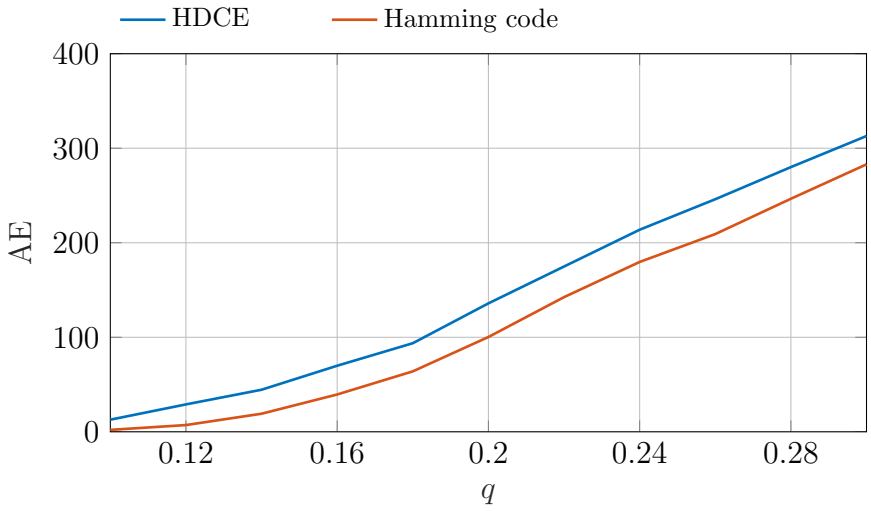


Figure 4.11: Comparison of HDCE and Hamming code in terms of AE versus probability of having edges between clusters (q) for k=4 clusters and p=0.80. Clustering is based on harmonic function (Algorithm 2).

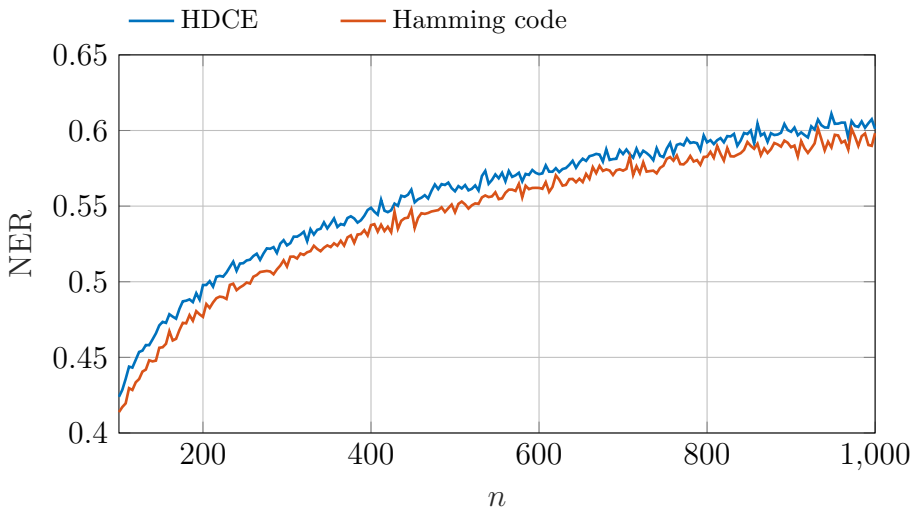


Figure 4.12: Comparison of HDCE and Hamming code in terms of error rate vs. number of nodes for 4 clusters and SBM settings p=0.55 and q=0.35. Clustering algorithm 2 is applied.

4.3 Comparing the Applied Encoding Schemes

In this section the comparison of four presented encoding methods is shown. Figure 4.13 illustrates the behavior of algorithm 1 in section 4.1. Performance of algorithm 2 in section 4.2 is also shown in figure 4.14. The graphs are generated from 500 nodes and 4 clusters, with 5 known labels in each cluster. SBM settings are $p = 0.55$ and $q = 0.35$ for 4 clusters and $p = 0.80$ and $q = 0.20$ for 10 clusters . The total number of simulations is 500 (all the settings are the same as experiments of sections 4.1 and 4.2).

Both figures 4.13 and 4.14 show that the proposed error correcting approach with Hamming code on top of HDCE scheme outperforms other encoding schemes for both clustering algorithms 1 and 2. For clustering based on algorithm 2, it can be seen that HDCE-based encoding performance is better than one-hot and Hamming code on top of one-hot encoding, however the error rate is lower than clustering based on algorithm 1. Finally, from both figures it is obvious that encoding with higher minimum Hamming distance improves the performance of clustering.

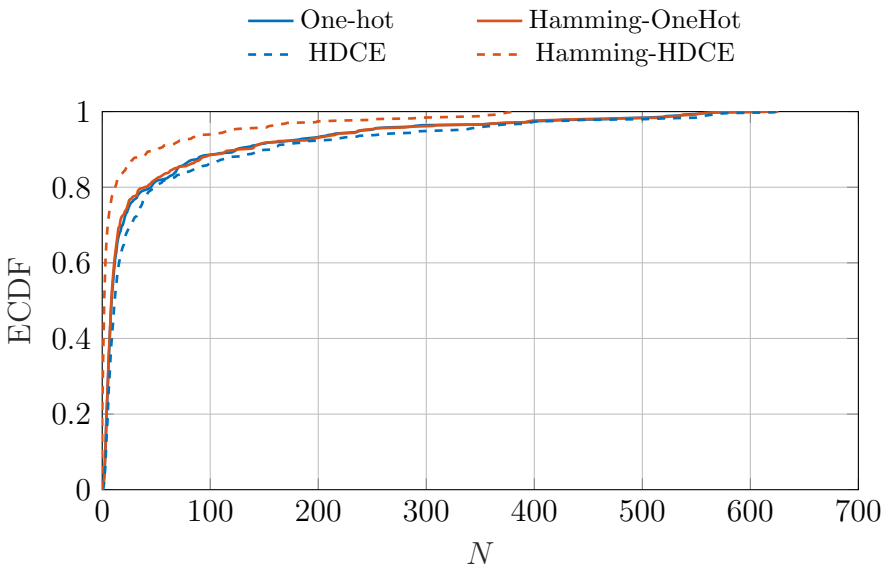


Figure 4.13: Comparing the applied encoding schemes in terms of ECDF curves of number of errors (N) for clustering based on algorithm 1

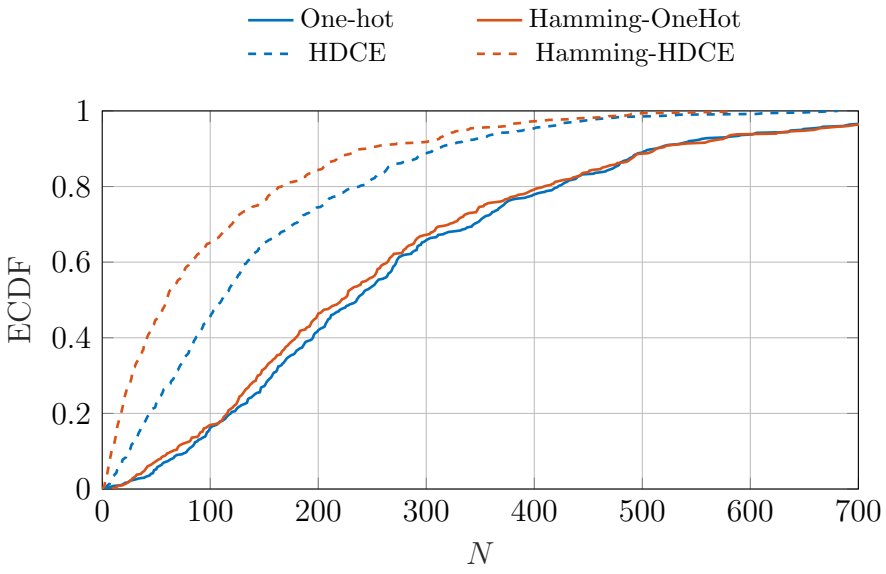


Figure 4.14: Comparing the applied encoding schemes in terms of ECDF curves of number of errors (N) for clustering based on algorithm 2

4.4 Comparison of Different Minimum Hamming Distances

The results that are shown so far for higher Hamming distances, are related to the minimum Hamming distance of 4 for Hamming code encoded code-words with 7 bits (for the case of 4 clusters). In this section, the performance for lower minimum Hamming distance is studied for the graphs generated with 1000 nodes, 4 clusters and 5 known labels in each cluster. SBM settings are set to $p = 0.55$ and $q = 0.35$ in order to be consistent with the experiments of the section 4.1.2.

As it is illustrated in figure 4.15, in case of encoding with Hamming code (7,4), it is required to use the code-words in HDCE (4bits) which produces Hamming code-words (7bits) of minimum Hamming distance equal to 4 (figure 4.15a). Of course there would be a lot of possibilities of codewords combinations for having these minimum Hamming distances. In this case, with Hamming code encoding of (7,4), there is 16 codewords in total. For 4 clusters we need a combination of 4 codewords, which means there will be

$$\binom{16}{4} = \frac{16!}{4!(16-4)!} = 1820$$

combinations in total. As mentioned, Hamming code (7,4) is applied, therefore all codewords have 7 bits with maximum minimum Hamming distance of either 3 or 4. As it is not feasible to experiment all these combinations, the figure 4.15(b) is plotted for a randomly chosen of the codewords combination with minimum Hamming distance of 3. However, we can employ the codebook design criterion which is suggested in papers [29] and [14] (also shortly discussed in section 4.1.1). In these papers, it has been proven that there are two main requirements to design the codebood. First requirement which is already discussed in section 4.1.1 has to do with maximizing the Hamming distance between different codewords (rows of the codebook). In order for this condition to be fulfilled, the codebook with maximum minimum Hamming distance of 4 has to be chosen. The second requirement is related to the column separation of the codebook. That means, the columns of the codebook must be as uncorrelated as possible. According to [29], when considering this condition, two points have to be taken care of:

- The columns shouldn't be complementary of each other, because in case of wrong bit position at the output, both will be affected.
- Columns of all zeros or all ones are not recommended, because of the row dependency.

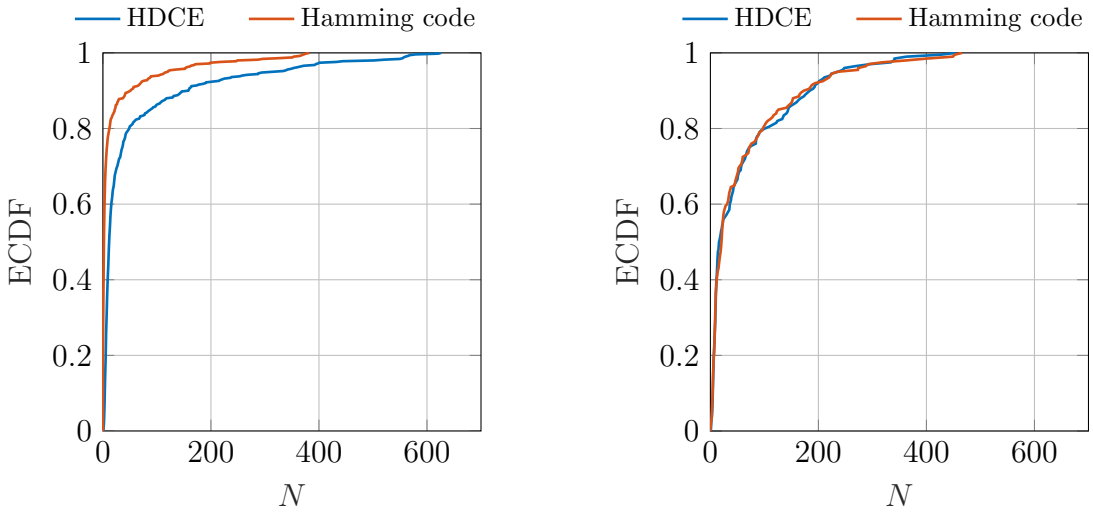
We can incorporate these conditions to justify the figure 4.15(a). The codebook that is applied for this experiment has the structure as

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

As it can be observed, this codebook fulfills all the conditions related to rows and columns and has a minimum Hamming distance, which achieves the maximum of 4. however, consider another codebook as

$$\mathbf{C}' = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

This codebook also has the maximum minimum Hamming distance of 4 as the previous one. However as it can be seen, it violates the second condition, because



(a) minimum Hamming distance=4 (b) minimum Hamming distance=3

Figure 4.15: Performance analysis for 4 clusters with two different minimum Hamming distances in terms of ECDF curve for number of errors (N).

the second column has all zeros. Also all other columns are complementary of each other.

This codebook design criteria holds true for the general case of codebook design for K clusters.

4.5 Increasing Number of Clusters

Experimental result of the section 3.2.4 is illustrated in this section. As already explained, the goal is to estimate the maximum number of clusters that can be packed in the fixed number of bits (4 bits in our experiment). To do so, the number of clusters with four bits is increased and the performance is analysed for both algorithm 1 and algorithm 2 clustering methods. Both experiments are done for the graphs with 500 nodes and 20 known cluster labels. Total number of simulations is M=500. SBM settings for algorithm 1 is defined as $p = 70$ and $q = 0.35$ and for the algorithm 2 the values are set to $p = 80$ and $q = 0.20$. As it is not possible to apply one-hot encoding for more than 4 clusters with 4 bits, the experiments are done only for the case of Hamming code encoding.

As it is shown in figures 4.16 and 4.17, from 9 clusters, the AE with 4 bits is more than half of the total number of the nodes, which is equivalent to random guess for cluster labels. For both cases one can see a jump in the average errors from 8 to 9 clusters. It can be concluded that for 4 bits with this specific experiment settings, up to twice the number of bits can be clustered with having an error rate of less than 0.50.

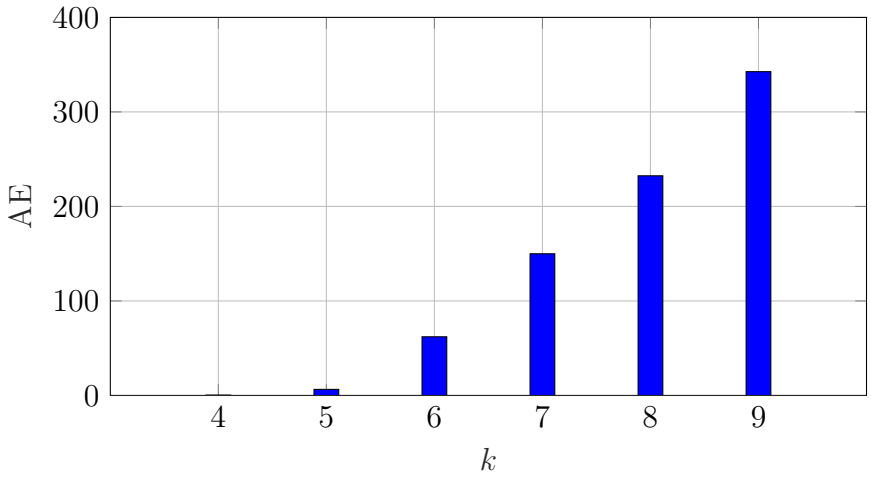


Figure 4.16: Comparison of different number of clusters (k) for encoding with 4 bits in terms of AE bars for clustering algorithm 1.

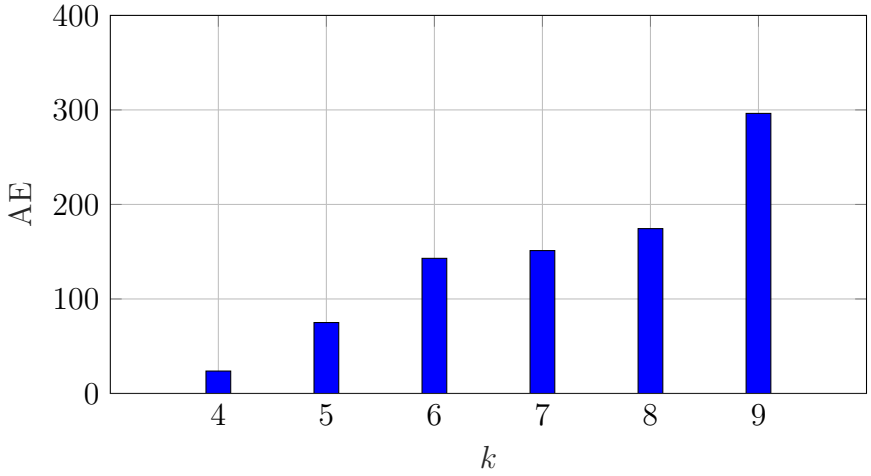


Figure 4.17: Comparison of different number of clusters (k) for encoding with 4 bits in terms of AE bars for clustering algorithm 2.

4.6 Performance Analysis on Real Data Set

As a final experiment, error correcting approach is applied on real data set. Stanford Network Analysis Project (SNAP) library from [41] is used as a source of large data sets. Since most of the datasets in SNAP had a huge number of nodes, a dataset with 1005 nodes was chosen from this library so that was possible to perform simulations on a conventional machine. The dataset was related to the email network with 42 clusters. Each cluster is an indication of one department of an institute. The graph is generated from department membership labels and Email communication links between members of the institution. According to [41] "*There is an edge (u, v) in the network if person u sent person v at least one email*". Number of simulations was 200 and number of the known cluster labels in each cluster assumed to be L=26.

As expected, the performance is different comparing to the previous experiments as the graph is now deterministic and the number of clusters is much higher than our previous experiments on random graphs. One problem is that when the number of known labels in each cluster is specified by L, in some clusters there is less than L nodes. To overcome this issue, two different modifications are applied to have a good performance with such graphs:

1. Clusters that have less than L nodes get sampled completely and reduce the total number of labels (figure 4.18)

2. Clusters that have less than L nodes are removed from the graph and by that reducing the total number of nodes (figure 4.19)

As we are comparing two curves with different number of nodes, it makes sense to consider error rate instead of number of the errors. Also clustering algorithm 1 is applied for both cases. For this experiment, we are using a (15,11)-Hamming code with HDCE and a resulting minimum Hamming distance of 2.

It can be seen from figures 4.18 and 4.19 that both modifications performs well in terms of error rate. The second modification shows a slightly better performance regarding Hamming code, but both are almost in the same range.

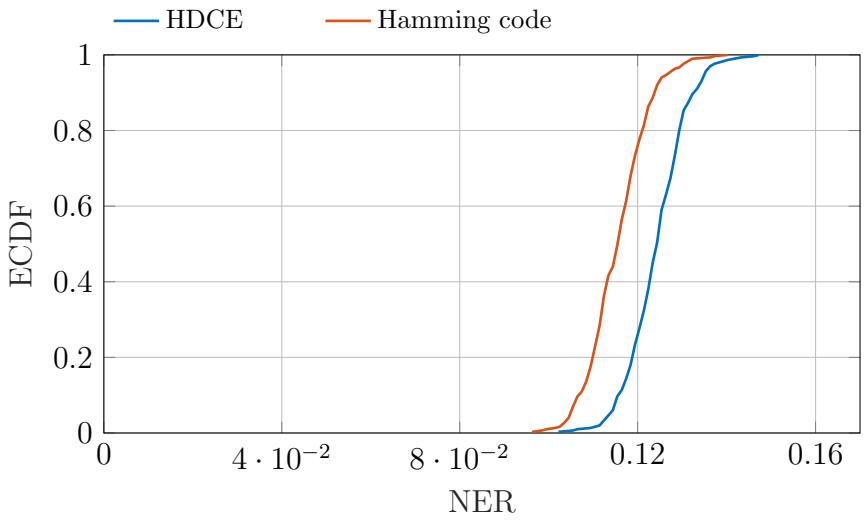


Figure 4.18: ECDF plot of error rate for HDCE and Hamming code encoding for the case of first modification, which keeps the number of clusters (42) and nodes (1005) unchanged. The number of known labels in each cluster is 26.

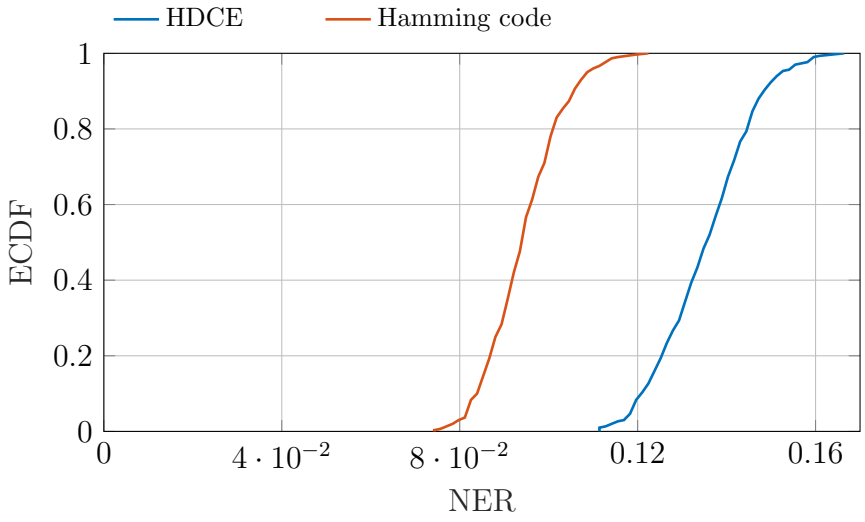


Figure 4.19: ECDF plot of error rate for HDCE and Hamming code encoding for the case of second modification, which reduces the number of clusters and nodes. This modification led to 727 nodes and 15 clusters for 26 known labels in each cluster.

Chapter 5

Conclusion and Outlook

In this thesis, clustering perspective of the big data analysis in the field of GSP has been studied. The main focus was on semi-supervised clustering task for graphs with known number of clusters. We developed a framework to incorporate prior information regarding cluster labels more efficiently. To do so, error correcting codes has been applied to represent cluster labels. Unlike previous methods that apply error correcting codes on supervised clustering for two clusters, our algorithm is able to work with semi-supervised multi-class clustering methods. We extend the idea of one-hot encoding, which is a method to represent categorical data. The goal was to reduce clustering errors by exploiting error correction capability of such encoding.

For the experiments, we applied SBM in order to generate random graphs. We experimented our idea on two different semi-supervised clustering methods based on spectral clustering and harmonic functions. For each clustering method, our algorithm has been applied over different number of clusters and a wide range of SBM settings and number of the nodes. We also experimented the algorithm on a real dataset in order to analyse the performance over deterministic graphs. Hamming code encoding scheme has been applied to represent cluster labels.

As the first experiment, we applied the algorithm on top of one-hot encoding method, but there was no improvement for the general number of clusters (only for 10 clusters). It turned out that the algorithm works well only with certain codebook design criteria. Therefore, we made some modifications in codebook representation of one-hot encoding, then applied Hamming code on top of that. With this modification, we could achieve a great improvement by applying error correcting codes. Based on the experimental results, we concluded that our algorithm outperforms the existing algorithms on semi-supervised multi class clustering for both random and real datasets.

5.1 Outlook

Although the experiments lead to desired results, the algorithm can be improved further. One possibility is related to the real graphs with heterogeneous structure and node number of hundreds of thousands. In such cases, other linear error correcting codes like Low-Density Parity Check (LDPC) or Turbo codes can be checked.

Regarding the boundaries of the SBM setting, there could be other semi-

supervised clustering methods on top of which our algorithm gains more flexibility in choosing probability values of SBM model.

Last point is related to the codebook design criteria. The requirement regarding independent rows and columns is limiting the choices and requires more computational effort in order to find the best codebook which suits the problem. Future works can be performed in this field.

Appendix A

Symbols and Terms

Symbols

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$	Graph with node set $\mathcal{V}$, edge set $\mathcal{E}$ and adjacency matrix $\mathbf{W}$
$\mathrm{C}(A_1, A_2)$	Graph cut between two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$
$\mathrm{RC}(A_1, A_2)$	Ratio cut between two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$
$\mathbf{f} \in \mathbb{R}^N$	Vector notation (lower case, boldface letter)
$\mathbf{W} \in \mathbb{R}^{N \times N}$	Matrix notation (upper case, boldface letter)
$\mathbf{I}$	Identity matrix
$\mathbf{W}_{:j}$	j^{th} column of the matrix $\mathbf{W}$
$\mathbf{W}_{j:}$	j^{th} row of the matrix $\mathbf{W}$

Operators

$(.)^T$	Transpose of a matrix or vector
$\mathrm{Tr}(\mathbf{W})$	Trace of the matrix $\mathbf{W}$
$\|.\|$	norm operator
$\triangle$	Laplacian
$\mathrm{diag}(\mathbf{f})$	Square diagonal matrix with the elements of vector $\mathbf{f}$ in the main diagonal
$\binom{n}{k}$	Binomial coefficient – n choose k

Acronyms

SBM	Stochastic Block Model
GSP	Graph Signal Processing
ECOC	Error Correcting Output Code
AE	Average Error
ECDF	Empirical Cumulative Distribution Function
NER	Node Error Rate
SNAP	Stanford Network Analysis Project
RC	Ratio Cut
SVM	Support Vector Machine
LDPC	Low-Density Parity Check
HDCE	Higher Distance Compact Encoding

Bibliography

[1] Y. Fan. *Creating and Analysing Facebook Friend Network Graphs Using Python*. 2019. URL: https://www.databentobox.com/2019/07/28/facebook-friend-graph/.

[2] D. Gilbert. *InMaps Visualises Professional Networks*. 2011. URL: https://randomwire.com/linkedin-inmaps-visualises-professional-connections/.

[3] A. Sandryhaila and J. M. F. Moura. "Big Data Analysis with Signal Processing on Graphs: Representation and processing of massive data sets with irregular structure". In: *IEEE Signal Processing Magazine* 31.5 (2014), pp. 80–90.

[4] Z. Yan, F. K. Soong, and R. Wang. "Word Graph Based Feature Enhancement for Noisy Speech Recognition". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Vol. 4. 2007, pp. IV-373-IV–376.

[5] L. Gao et al. "A Graph-Based Resource Sharing and Admission Control for Vehicular Networks". In: *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. 2019, pp. 1–6. DOI: 10.1109/WCNCW.2019.8902755.

[6] A. Furno et al. "A Graph-Based Framework for Real-Time Vulnerability Assessment of Road Networks". In: *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2018, pp. 234–241. DOI: 10.1109/SMARTCOMP.2018.00096.

[7] P. Berger, G. Hannak, and G. Matz. "Graph Signal Recovery via Primal-Dual Algorithms for Total Variation Minimization". In: *IEEE Journal of Selected Topics in Signal Processing* 11.6 (2017), pp. 842–855.

[8] L. Xu, W. Li, and D. Schuurmans. "Fast normalized cut with linear constraints". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2009), pp. 2866–2873.

[9] X. Zhu, Z. Ghahramani, and J. Lafferty. "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions". In: *Proc. Int. Conf. Machine learning (ICML-03)* (2003), pp. 912–919.

[10] M. Belkin, I. Matveeva, and P. Niyogi. "Regularization and Semi-supervised Learning on Large Graphs". In: *Learning Theory*. Ed. by J. Shawe-Taylor and Y. Singer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 624–638. ISBN: 978-3-540-27819-1.

[11] W. Liu, J. Wang, and S. Chang. "Robust and Scalable Graph-Based Semisupervised Learning". In: *Proceedings of the IEEE* 100.9 (2012), pp. 2624–2638.

[12] A. Blum and S. Chawla. "Learning from Labeled and Unlabeled Data Using Graph Mincuts". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 19–26. ISBN: 1558607781.

[13] K. Avrachenkov, P. Chebotarev, and A. Mishenin. "Semi-supervised learning with regularized Laplacian". In: *Optimization Methods and Software* 32.2 (2017), pp. 222–236.

[14] T. G. Dietterich and G. Bakiri. "Solving multiclass learning problems via error-correcting output codes". In: *Journal of Artificial Intelligence Research* 2 (1995), pp. 263–286.

[15] J. Brownlee. *Why One-Hot Encode Data in Machine Learning?* 2020. URL: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.

[16] Y. Tsuruda and A. Tsuneda. "Study on Error-Correcting Using Output Level of Correlation Receivers and Hamming Codes in CDMA Communications". In: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. 2018, pp. 234–236.

[17] U. Von Luxburg. "A tutorial on spectral clustering". In: *Statistics and computing* 17.4 (2007), pp. 395–416.

[18] W. Zhu, F. Nie, and X. Li. "Fast Spectral Clustering with efficient large graph construction". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 2492–2496. DOI: 10.1109/ICASSP.2017.7952605.

[19] B. Chen et al. "A spectral clustering algorithm for automatically determining clusters number". In: *Proceeding of the 11th World Congress on Intelligent Control and Automation*. 2014, pp. 3723–3728. DOI: 10.1109/WCICA.2014.7053336.

[20] N. Sapkota et al. "Data Summarization Using Clustering and Classification: Spectral Clustering Combined with k-Means Using NFPH". In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. 2019, pp. 146–151. DOI: 10.1109/COMITCon.2019.8862218.

[21] P. Berger, T. Dittrich, and G. Hannak & G. Matz. "Semi-Supervised Multiclass Clustering Based on Signed Total Variation". In: *ICASSP* (2019), pp. 4953–4957.

[22] D. Wagner and F. Wagner. "Between min cut and graph bisection". In: *18th International Symposium on Mathematical Foundations of Computer Science (MFCS)* (1993), pp. 744–750.

[23] L. Hagen and A. B. Kahng. "New spectral methods for ratio cut partitioning and clustering". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11.9 (1992), pp. 1074–1085. DOI: 10.1109/43.159993.

[24] H. Lütkepohl. *Handbook of Matrices. Chichester: Wiley.* 1997.

[25] P. Berger, T. Dittrich, and G. Matz. "SEMI-SUPERVISED CLUSTERING BASED ON SIGNED TOTAL VARIATION". In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP).* 2018, pp. 793–797.

[26] L. Xiao-feng, Z. Xue-ying, and D. Ji-kang. "Speech Recognition Based on Support Vector Machine and Error Correcting Output Codes". In: *2010 First International Conference on Pervasive Computing, Signal Processing and Applications.* 2010, pp. 336–339.

[27] C. Cardie. "Using Decision Trees to Improve Case-Based Learning". In: *Proceedings of the Tenth International Conference on Machine Learning, Morgan Kaufmann* (1993), pp. 25–32.

[28] L. J. Cao and F. E. H. Tay. "Support vector machine with adaptive parameters in financial time series forecasting". In: *IEEE Transactions on Neural Networks* 14.6 (2003), pp. 1506–1518.

[29] F. Masulli1 and G. Valentini. "Effectiveness of Error Correcting Output Coding methods in ensemble and monolithic learning machines". In: *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings* (1993), p. 2000.

[30] P. Erdos and A. Rényi. "On random graphs". In: *I. Publicationes Mathematicae (Debrecen)* 6 (1959), pp. 290–297.

[31] E. Abbe, A. S. Bandeira, and G. Hall. "Exact Recovery in the Stochastic Block Model". In: *IEEE Transactions on Information Theory* 62 (2016), pp. 471–478.

[32] K. S. Xu. "Stochastic Block Transition Models for Dynamic Networks". In: *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)* 38 (2015), pp. 1079–1087.

[33] K. S. Xu. *Dynamic-Stochastic-Block-Model.* 2018. URL: `https://github.com/IdeasLabUT/Dynamic-Stochastic-Block-Model`.

[34] H. Hu et al. "Multi-Class Constrained Normalized Cut With Hard, Soft, Unary and Pairwise Priors and its Applications to Object Segmentation". In: *IEEE Transactions on Image Processing* 22.11 (2013), pp. 4328–4340.

[35] Z. Li, J. Liu, and X. Tang. "Constrained clustering via spectral regularization". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 2009, pp. 421–428.

[36] X. Wang and I. Davidson. "Flexible constrained spectral clustering". In: July 2010, pp. 563–572. DOI: `10.1145/1835804.1835877`.

[37] A. P. Eriksson, C. Olsson, and F. Kahl. "Normalized Cuts Revisited: A Reformulation for Segmentation with Linear Grouping Constraints". In: *2007 IEEE 11th International Conference on Computer Vision.* 2007, pp. 1–8.

[38] M. Subhransu, V. Nisheeth, and M. Jitendra. "Biased normalized cuts". In: June 2011, pp. 2057–2064. DOI: `10.1109/CVPR.2011.5995630`.

[39] C. Reiter. "Easy Algorithms for Finding Eigenvalues". In: *Mathematics Magazine* 63.3 (1990), pp. 173–178.

[40] S. Boyd. *CVX: Matlab Software for Disciplined Convex Programming*. 2020. URL: http://cvxr.com/cvx/.

[41] J. Leskovec and A. Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. http://snap.stanford.edu/data. June 2014.

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 Iman Rezaei