UiT THE ARCTIC UNIVERSITY OF NORWAY

Department of Physics and Technology

Joint ranking and clustering based on Markov Chain transition probabilities learned from data

—

Sigurd Løkse

FYS-3921 Master's Thesis in Electrical Engineering
December 2014

Abstract

The focus of this thesis is to develop a Markov Chain based framework for joint ranking and clustering of a dataset without the need for critical user-defined hyper-parameters. Joint ranking and clustering may be useful in several respects, and may give additional insight for the data analyst, as opposed to the traditional separate ranking and clustering procedures. By coupling Markov chain theory with recent advances in kernel methods using the so-called probabilistic cluster kernel, we are able to learn the transition probabilities from the inherent structures in the data in a near parameter-free approach. The theory developed in this thesis is applied to several real world datasets of different types with promising results.

Acknowledgments

First and foremost, I would like to thank my supervisor, Associate Professor Robert Jenssen. Your lectures in signal processing during my bachelor's studies inspired me to get my master's degree in Tromsø, which I have never regretted. Your seemingly endless knowledge and enthusiasm has really been an inspiration to me and helped me stay motivated. Our interesting discussions and your feedback on this thesis have been of great help. Thank you!

To Bård Hansen at IIS, UiT. Your maths lectures during my year in economics made me realize that maths was fun. If it had not been for you, I would never have embarked on this journey.

To the guys at the HPC group at UiT. Thank you for letting me use Stallo for my experiments. It was a great help to me when the size of my matrices were too big for my computer to handle.

To my family and friends. Thank you for understanding my absence during the work on this thesis. I am looking forward to spending more time with you.

Sigurd Løkse

Tromsø, December 2014

Contents

List of Figures

List of Tables

Abbreviations

EM	Expectation Maximization
GHAS	Generalized Hard Algorithmic Scheme
GMM	Gaussian Mixture Model
IID	Independent and Identically Distributed
KPCA	Kernel Principal Component Analysis
MLE	Maximum Likelihood Estimator
PCA	Principal Component Analysis
PCK	Probabilistic Cluster Kernel
PDF	Probability Density Function
PMF	Probability Mass Function
PPR	Personalized PageRank
PR	PageRank
RBF	Radial Basis Function

Chapter 1

Introduction

In our daily lives, we are surrounded by huge amounts of data. We take pictures with our phones and send them to our friends. When working out, we use devices to measure how well we are doing. During a discussion, we use Google to prove that we are right. Everything is quantized and stored. From web usage statistics to driving patterns.

During the past couple of decades, the storage and computational capabilities has advanced drastically. As analyzing large amounts of data has become computationally feasible, the field of data analysis has gained more and more attention. Machine learning is a form of data analysis, where the aim is to implement algorithms which *learns* from the data without human interaction. In this thesis, two methodologies within the machine learning field are considered, namely ranking and clustering.

In recent years, ranking has received a lot of attention, especially in Information Retrieval (IR). The goal of a ranking task is to produce an ordered list of objects. Formally, a ranking model sorts a group of objects according to importance, relevance or preference. Ranking models have been utilized in several applications. For instance search engine query ranking [1, 2, 3], recommender systems using collaborative filtering[1] [4], image ranking [5, 6] and language processing [7, 8].

Within the realm of ranking algorithms, there are both supervised and unsupervised methods. In the supervised methods, ground truth data on listwise or pairwise ordering is provided. Based on this ground truth data, a ranking model is trained to rank new out of sample data. Methods within supervised ranking include the RankNET [9], RankSVM [10] and RankBoost [11]. In the unsupervised setting, there are several methods based on *graph theory* [1, 2, 3, 5, 6, 7, 8]. These rank with a *relational* model. The graph

[1]This is used by Netflix, Spotify and Amazon.

encodes the strength of the relationship between objects (or *vertices*). The importance of an object is then *propagated* through the graph based on the strength of the relationships.

Although the graph theoretic methods mentioned above have specific application areas, similar methods have been proposed for general multi attribute data [12, 13]. These require a similarity measure to define the relationship between the objects. For specific applications, these similarity measures might be easily defined. For general multi attribute data, the similarity measure used often relies on some dataset dependent critical parameter which can influence the result.

The clustering task is to partition the data into subsets, or natural groups, known as clusters. The members of each cluster should be more similar to the members of the same cluster than the members of other clusters. This should be accomplished without any prior knowledge about the data[2]. Clustering is about *discovering* natural groups in the data, rather than *learning* known patterns, which is known as classification. Clustering is normally considered a more difficult task than classification. There are numerous applications of cluster analysis, including image segmentation and computer vision [14, 15, 16], document grouping [17, 18, 19], biology [20, 21, 22, 23] and market segmentation [24, 25] to name a few.

To partition the data, the definition of reasonable partition needs to be clear. This definition is different for each of the clustering algorithms and depends on the structure of the data. Because of this, no single clustering algorithm is appropriate for every dataset available. In some way, all the algorithms are based on *proximity* measures which measure either dissimilarity (for instance Euclidean distance) or similarity (for instance the Gaussian kernel). This could either be proximity between two clusters, proximity between a data point and a cluster or proximity between two data points.

A common obstacle for the machine learning methods requiring similarity measures is the dependence of critical parameters. The result could be very dependent on these parameters which are set prior to the learning process. For supervised learning, this might not be a big problem as we have prior information on groups in the data. Optimal parameter values can then be found by parameter sweeps and evaluating the results on a test set. In unsupervised learning, no ground truth information is known a-priori. Parameters are often set by rules of thumb or heuristic methods, which could leads to sub-optimal results. Recently, methods have been developed, attempting to learn some similarity measure directly from the data. A very recent kernel function called the *Probabilistic Cluster Kernel* (PCK) [26] has been used for

[2]The number of clusters in the data is often required.

semi-supervised learning and clustering with promising results. The PCK is calculated by fitting Gaussian Mixture Models (GMM) to the data with several initial conditions and several number of mixture components using the Expectation Maximization (EM) algorithm. The posterior distributions for the data points are used to calculate pairwise similarities.

In this thesis, ranking and clustering is explored from a graph theoretic point of view with a basis in Markov chain theory. The transition probabilities are generated from the PCK. Since the PCK is learned from structures in the data, the transition probabilities are *learned from the data*. This ensures that no critical parameters needs to be set prior to the learning process. The main contributions of this thesis include:

- A framework for joint ranking and clustering, which enables the data analyst to rank the data, either globally or a versatile local ranking for each cluster. This includes within-cluster-ranking and across-cluster-ranking. Across-cluster-ranking ranks data points in one cluster with respect to the data points in another cluster.

- A way of learning Markov chain transition probabilities from data via the PCK, ensuring that the result is not dependent on critical parameters.

The theory developed in this thesis is applied to several real world datasets with promising results.

1.1 Related work

Very little work has been done on combined frameworks for ranking and clustering. However, some related work has been found.

RankClus [27] is an algorithm which simultaneously ranks and clusters data on information networks. This algorithm iteratively ranks and clusters the data in such a way that the ranking and clustering procedure mutually enhances each other. However, the algorithm is based on *bi-type* information networks like conference-author networks, movie-user networks and newsgroup-author networks. Thus this algorithm does not apply to multi attribute data. *NetClus* [28] is a generalized variant of *RankClus*, where the information network can have more than two types.

RankCompete [29] simultaneously ranks and clusters data in information networks. This algorithm is based on *competing* random walkers on an information network. For a k cluster problem, k random walkers are initialized on unique nodes in the dataset. For each step in the random walk, it is checked which of the k random walkers have the highest probability of being at each node. The random walker with the highest probability of being at a given node owns the node. The random walkers who do not own a node sets the score of the node to zero. The scores are normalized and the steps restarts. The basis for the algorithm is a matrix of pairwise similarities. In this paper, the experiments are based on two types of data. One dataset of images and a research paper dataset. In the image dataset, Scale Invariant Feature Transform (SIFT) based similarities are used. In the research paper dataset, similarities are based on citations. For general multi attribute data, the similarity measure would probably be chosen as an RBF, which in general is sensitive to the width parameter σ.

1.2 Structure of Thesis

This thesis is divided into four parts. Part I contains the background theory needed for the methods used in this thesis. Ch. 2 presents theory on Discrete Markov Chains. This is needed for the relational ranking algorithms presented in Part II. Ch. 3 presents two relational ranking algorithms. The first one is the well known PageRank algorithm by Google. The other is a similar algorithm designed for multi attribute data. Ch. 4 presents theory on Data Representation and Dimensionality Reduction. Two methods are presented. Principal Component Analysis (PCA) and Kernel Principal Component Analysis (KPCA). KPCA is a nonlinear extension of PCA and is essential for the theory in Part II. Ch. 5 presents a few selected clustering methods. There is an overwhelming amount of information on clustering in the literature. For the benefit of the reader, the algorithms presented are selected specifically to build the foundation for the main theory.

Part II contains the main theory. Ch. 6 presents the Probabilistic Cluster Kernel (PCK), which is used as a similarity measure for the methods developed in this thesis. This is a similarity measure which is learned directly from the data and has no strong parameter dependence. In this chapter, a new mathematical connection between the PCK and the consensus matrix as presented in Sec. 5.4 is derived. Ch. 7 introduces a way of learning Markov Chain transition probabilities from the data. This is done using the PCK. In addition to this, a dual form of the stationary distribution of the Markov Chain induced by these transition probabilities in the empirical kernel space is derived. This is extended to the kernel feature space and connected to nonparametric density estimates. Ch. 8 presents the personalized PageRank (PPR) algorithm. This is a generalized version of the PageRank as presented in Sec. 3.1. Previous work on the PPR is presented. Using this theory, an embedding for the data is proposed. It is shown that distances in this embedding has a clear interpretation which makes sense for clustering and that the PPR can be calculated using the embedded data.

In Part III, the theory developed in Part II is applied to several real world datasets of different types. The thesis is concluded with some final words and suggestions to further work in Part IV.

In the figures generated for this thesis, tick labels on the axes have intentionally been removed where this information is redundant. The relevant information is the overall structure of the data, not numerical values.

Part I

Background Theory

Chapter 2

Discrete Markov Chains

A discrete Markov Chain is a statistical model of a process where the process can be modeled by states. For instance, in a weather model, the states might be "raining", "sunny" and "cloudy". With each state, we associate a transition probability to each of the other states in the Markov Chain. That is, the probability that the process transitions from one state to another in one step. The "step" is often a discrete step in time, but it can also be other physical metrics like distance. The Markov Chain theory is vast, and most of the theory is not needed for the methods used in this thesis. This chapter will only present the Markov Chain theory which is relevant for the purposes of this thesis. The theory is needed both for the ranking in Ch. 3 and for the theory presented in Ch. 7 – Ch. 8.

In Machine Learning, the Markov theory is used in several applications. For instance speech recognition with Hidden Markov Models [30], translating hand written text to machine written text with Hidden Markov Models [31, 32], Reinforcement Learning with Markov Decision Processes [33] and web page ranking using stationary distributions [2].

2.1 Definitions

A Markov Chain is a sequence of random variables $\{X_1, X_2, \ldots\}$, where X_n denotes the state of the random process at time n. The transition probabilities between the states satisfies the *Markov Property*:

$$P(X_{n+1} = x_{n+1} | X_n = x_n, \ldots, X_1 = x_1) = P(X_{n+1} = x_{n+1} | X_n = x_n). \quad (2.1)$$

That is, given the current state of the process, the transition probability to the other states is independent on the previous states of the chain. Now, let $p_{ij} = P(X_{n+1} = j | X_n = i)$. Then the matrix $\mathbf{P} = \{p_{ij}\}_{N \times N},\ p_{ij} \geq$

0 is called the *transition probability matrix* of the Markov Chain with N states. It can be shown using the *Chapman Kolmogorov* equations that the transition probabilities in k steps can be calculated easily using the transition probability matrix [34]. Specifically, if $\mathbf{P}^k = \{p_{ij}^{(k)}\}_{N\times N}$, then

$$p_{ij}^{(k)} = P(X_{n+k} = i|X_n = j).$$

That is, we calculate $\mathbf{P}^k$ and extract $P(X_{n+k} = i|X_n = j)$ as the element in row i, column j.

2.1.1 Irreducible Markov Chains

Irreducible Markov Chains is a class of Markov Chains which has some nice properties needed in order to justify the calculation of stationary distributions in later chapters. Two states, i and j, are said to *communicate* if for some $n \geq 0$ and some $m \geq 0$, $p_{ij}^{(n)} > 0$ and $p_{ji}^{(m)} > 0$. This means that it is possible for the process to reach state i from state j and vice versa. Now, state i and state j are said to belong to the same *class* if they communicate. A Markov Chain is said to be irreducible if all of its states belongs to the same class (i.e. there is only one class and all states communicate).

2.1.2 Stochastic Matrices

The transition probability matrix is a special type of matrix called a *stochastic* matrix. There are three types of stochastic matrices: left-, right-, and doubly stochastic matrices. If $\mathbf{P}$ is a matrix with non-negative entries and $\mathbb{1}$ is a vector of ones, then we classify a stochastic matrix as shown in Tab. 2.1. We see that in a left stochastic matrix, each column sums to one. In a right stochastic matrix, each row sums to one. In a doubly stochastic matrix, each row and each column sums to one. It is important to be aware of which type of stochastic matrix we work with, as this dictates where we extract the probabilities. As we will see in later sections, this is also important when calculating stationary distributions.

Example 2.1.1 (Weather model)**.** Suppose that the weather is either ”raining” or ”not raining” and that the weather tomorrow is independent on the weather previous days, given todays weather. Based on historical data of the weather, we have information that if it rains today, the probability that it will rain tomorrow is α. If it does not rain today, the probability that it does not rain tomorrow is β. This can be modeled by a Markov Chain. If the process is in state 1 when it does not rain and in state 2 when it rains, the

Table 2.1: Classification of stochastic matrices.

Requirement	Type
$\mathbb{1}^T\mathbf{P} = \mathbb{1}^T$	Left stochastic
$\mathbf{P}\mathbb{1} = \mathbb{1}$	Right stochastic
$\mathbb{1}^T\mathbf{P} = \mathbb{1}^T$ and $\mathbf{P}\mathbb{1} = \mathbb{1}$	Doubly stochastic

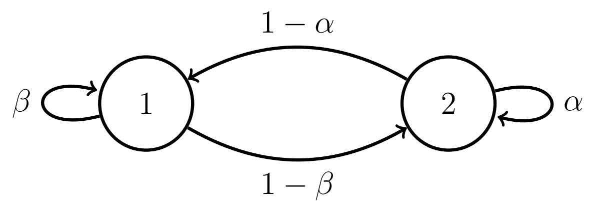


Figure 2.1: State diagram for weather model.

left stochastic transition probability matrix that defines this Markov Chain is given by

$$\mathbf{P} = \begin{pmatrix} \beta & 1-\alpha \\ 1-\beta & \alpha \end{pmatrix}.$$

The process can be visualized by a state diagram as shown in Fig. 2.1.

2.2 The Stationary Distribution

In some Markov Chains, the limit

$$\lim_{k\to\infty} \mathbf{P}^k$$

converges. In this case, the probability

$$\pi_j = \lim_{k\to\infty} p_{ij}^{(k)},$$

exists and is independent on the initial state i. This can be interpreted as the probability for the process to be at state j at any given time n. Thus

$$P(X_n = j) = \pi_j$$

For a N state Markov Chain where the limit converges, the *limiting probability* π_j is the unique non-negative solution of

$$\begin{aligned} \pi_j &= \sum_{i=1}^{N} \pi_i p_{ij} \\ \sum_{j=1}^{N} \pi_j &= 1. \end{aligned} \tag{2.2}$$

Let $\boldsymbol{\pi} = \begin{pmatrix} \pi_1 & \pi_2 & \cdots & \pi_N \end{pmatrix}^T$ be the *stationary distribution* of the Markov Chain and let $\mathbf{P}$ be a right stochastic matrix. From Eq. (2.2) we get

$$\boldsymbol{\pi}^T \mathbf{P} = \boldsymbol{\pi}^T. \tag{2.3}$$

Thus, $\boldsymbol{\pi}$ is the left eigenvector of $\mathbf{P}$ associated with the eigenvalue 1. For a left stochastic matrix, $\boldsymbol{\pi}$ is the right eigenvector of the stochastic matrix associated with the eigenvalue 1.

There are some situations where we know that the stationary distribution exists, but most are outside the scope of this thesis. However, one particular situation is relevant. The type of Markov Chains we will work with are called irreducible *time reversible* Markov Chains. In the next section, we will define time reversibility and conditions that have to be satisfied for a Markov Chain to be time reversible.

2.2.1 Time Reversible Markov Chains

Suppose that we want to investigate what happens if we move through the Markov Chain in reverse order. That is, starting at time k, we consider the state sequence $\{X_k, X_{k-1}, \ldots, X_1\}$. Assume that the limiting probabilities $\pi_i,\ i = 1, 2, \ldots, N$ for the non-reversed Markov Chain exist. This sequence can be shown to be a Markov Chain [34] with transition probabilities

$$p'_{ij} = \frac{\pi_j}{\pi_i} p_{ji}.$$

The Markov Chain is said to be *time reversible* if

$$\begin{aligned} p_{ij} &= p'_{ij} \\ \pi_i p_{ij} &= \pi_j p_{ji}. \end{aligned} \tag{2.4}$$

This is called the *balance equation.* Note that if we can find non-negative numbers y_j, $\sum_{j=1}^{N} y_j = 1$ that satisfies

$$y_i p_{ij} = y_j p_{ji},$$

the Markov Chain is time reversible. Then $\mathbf{y} = \begin{pmatrix} y_1 & y_2 & \cdots & y_N \end{pmatrix}^T$ is the *reversibility distribution.* By summing this equation over i, we get

$$\begin{aligned} \sum_{i=1}^{N} y_i p_{ij} &= \sum_{i=1}^{N} y_j p_{ji} \\ &= y_j \sum_{i=1}^{N} p_{ji} \\ &= y_j. \end{aligned} \tag{2.5}$$

A comparison between this and Eq. (2.2) reveals that the reversibility distribution must be equal to the stationary distribution.

Up until this point, we have assumed that the stationary distribution exists. It turns out that if the Markov Chain is irreducible and the reversibility distribution exists, then the stationary distribution also exists and is equal to the reversibility distribution.

Example 2.2.1 (Time reversible Markov Chain)**.** Consider a Markov Chain with the transition probability matrix

$$\mathbf{P} = \begin{pmatrix} 0 & 1/4 & 0 & 3/4 \\ 1/6 & 3/6 & 2/6 & 0 \\ 0 & 2/6 & 1/6 & 3/6 \\ 3/7 & 0 & 3/7 & 1/7 \end{pmatrix}.$$

The state diagram is shown in Fig. 2.2. As seen in the figure, all states communicate. Thus, the Markov Chain is irreducible. Using the balance equations in Eq. (2.4), it is easily verified that

$$\boldsymbol{\pi} = \begin{pmatrix} 4/23 \\ 6/23 \\ 6/23 \\ 7/23 \end{pmatrix}$$

is a reversibility distribution and that the Markov Chain is time reversible. Since the Markov Chain is irreducible and time reversible, $\boldsymbol{\pi}$ is also the stationary distribution.

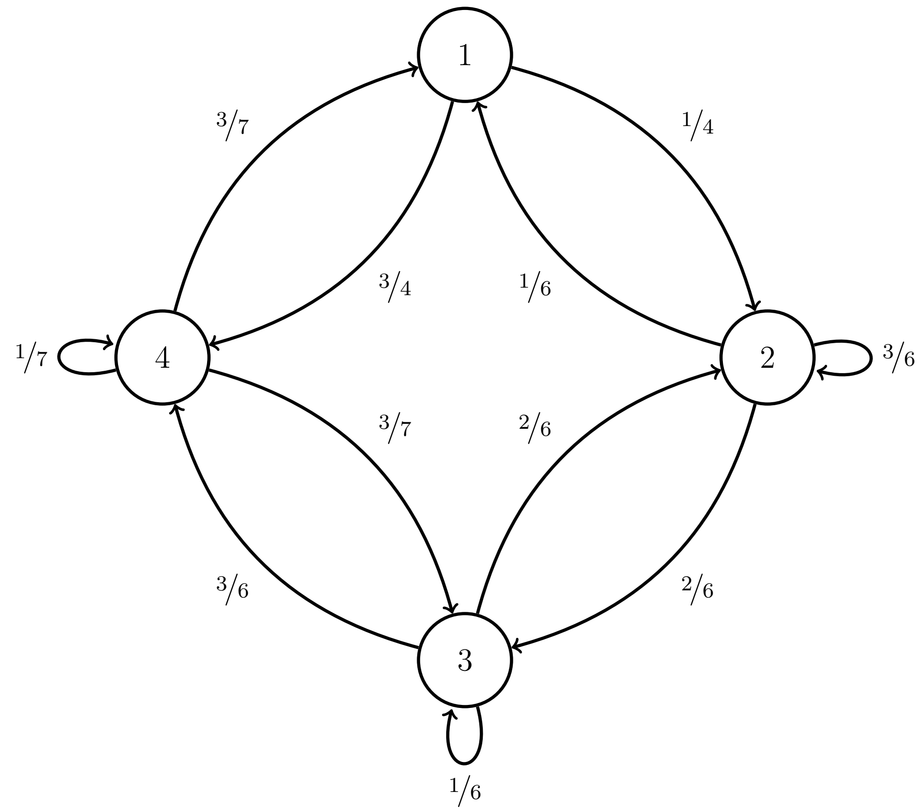


Figure 2.2: State diagram for a time reversible Markov Chain.

Chapter 3

Ranking

This chapter presents unsupervised Markov Chain based ranking algorithms. Specifically, the PageRank (PR) algorithm [2] is presented along with a similar multi attribute version [12]. These algorithms are chosen as the theory in Ch. 7–Ch. 8 have a common background with these algorithms. More information on ranking algorithms can be found in for instance [35, 36] and the references therein.

3.1 The PageRank

In the PageRank algorithm [2], the web is modeled as a directed graph. Each web page is a vertex on the graph and a directed edge from one vertex to the other vertices is determined by the link structures. The key to understanding PR is to think of each hyperlink as a recommendation. Furthermore, if the web page that links to the other web page is itself considered important, the recommendation should have a bigger impact. For an extensive review/analysis of PR, see [37].

Let $G = (V, E, W)$ be a directed graph of web pages with the vertices V, edges E and weights, W. Vertex v_i is connected to vertex v_j with a directed edge if web page i contains a hyperlink to web page j. Let n_i be the number of hyperlinks from web page i. The weight on the edge from vertex v_i to vertex v_j is then set to $w_{ij} = \frac{1}{n_i}$ if web page i has a hyperlink to web page j and $w_{ij} = 0$ otherwise. Furthermore, let π_i be the score of web page i. The score of web page j, π_j, is determined by the score of the pages that link to

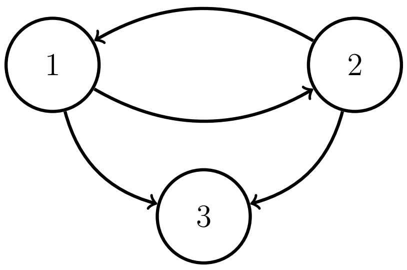


Figure 3.1: A simple web graph. In this example, vertex 3 is a sink.

it. Specifically, the score is defined as

$$\begin{aligned}\pi_j &= \sum_{i=1}^{N} \pi_i w_{ij} \\ &= \sum_{i=1}^{N} \pi_i \frac{1}{n_i},\end{aligned} \tag{3.1}$$

where N is the number of web pages on the graph. By this definition, the score of a web page is equally distributed between every web page it links to. Important web pages (with a large score) that have just a few hyperlinks will have a bigger impact on the scores than a web page with a lot of hyperlinks. By letting $\boldsymbol{\pi} = \begin{pmatrix}\pi_1 & \pi_2 & \dots & \pi_N\end{pmatrix}^T$, Eq. (3.1) can be rewritten as

$$\boldsymbol{\pi}^T = \boldsymbol{\pi}^T \mathbf{P}, \tag{3.2}$$

where $p_{ij} = w_{ij} = \frac{1}{n_i}$. Since $\sum_{j=1}^{N} p_{ij} = 1$ and $\mathbf{P}$ has positive elements, $\mathbf{P}$ is recognized as a right stochastic matrix. Thus, by comparing Eq. (3.2) to Eq. (2.3) we see that $\boldsymbol{\pi}$ is the stationary distribution of a Markov Chain with the transition matrix $\mathbf{P}$. This equation is solved using *power iterations* (see [38, Ch. 7]).

Although this definition of the graph works for connected graphs, there are challenges when working with web pages. For instance, what if a web page with inlinks have no outlinks as seen in Fig. 3.1? These are called *sinks*. What if the graph contain cycles as seen in Fig. 3.2? It turns out that sinks consume all of the scores. Cycles in the graph cause convergence issues during the power iterations. To fix these problems, Page et al. [2] proposed two corrections to the stochastic matrix. If a node is a sink, add a uniform distribution of outlinks to all other web pages. That is, replace $\mathbf{P}$ by

$$\mathbf{S} = \mathbf{P} + \frac{1}{N}\mathbf{a}\mathbb{1}^T,$$

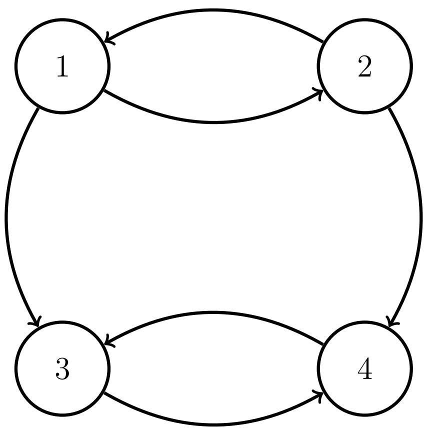


Figure 3.2: In this web graph, we see a cyclic structure between vertex 3 and vertex 4.

where **a** contains ones at every sink and zeros elsewhere and $\mathbb{1}$ is a vector of ones. The other adjustment is to break the cycles by assuming that a random surfer can surf the web without following the link structure. With probability α, a surfer selects a web page with equal probability from every web page available and with probability $1 - \alpha$ the surfer follows the link structure. This can be written mathematically by using the *Google matrix* $\mathbf{G}$ defined as

$$\mathbf{G} = (1-\alpha)\mathbf{S} + \alpha\frac{1}{N}\mathbb{1}\mathbb{1}^T.$$

The PR is the stationary distribution calculated from this stochastic matrix. That is,

$$\begin{aligned}\boldsymbol{\pi}^T &= \boldsymbol{\pi}^T\mathbf{G} \\ &= \boldsymbol{\pi}^T\left((1-\alpha)\mathbf{S} + \alpha\frac{1}{N}\mathbb{1}\mathbb{1}^T\right).\end{aligned} \tag{3.3}$$

It has been reported that Google uses $\alpha = 0.15$ [1, 2][1].

As we have all experienced, this simple model of the web works well for providing users with relevant results when searching for web pages.

[1]Note that the value α in these papers are equivalent to $1 - \alpha$ in this thesis.

3.1.1 A personalized extension to the PageRank algorithm

The well known PageRank algorithm as defined in Sec. 3.1 is a special case of Markov Chain Random Walks with Restarts. Consider the iterative sequence

$$\mathbf{r}_{k+1}{}^T = (1-\alpha)\mathbf{r}_k{}^T\mathbf{P} + \alpha\mathbf{s}^T,\ 0 < \alpha < 1, \tag{3.4}$$

where α is the restart probability, $\mathbf{P}$ is a right stochastic matrix and $\mathbf{s}$ is the *seed* distribution. This is a model for the behaviour of a random walker. At each step, the random walker will proceed to a neighbouring vertex according to the transition probabilities in $\mathbf{P}$ with probability $1-\alpha$. With probability α, the random walker will *restart* according to the seed distribution. The term *personalized* in personalized PageRank comes from the fact that the seed distribution adds the versatility of ranking with respect to the seed. The restart probability, α, decides the importance of the graph versus the seed. This iterative sequence will converge to the stationary distribution of the stochastic matrix

$$\mathbf{P}' = (1-\alpha)\mathbf{P} + \alpha\mathbb{1}\mathbf{s}^T.$$

Letting $\mathbf{s} = \frac{1}{N}\mathbb{1}$ yields

$$\mathbf{P}' = (1-\alpha)\mathbf{P} + \alpha\frac{1}{N}\mathbb{1}\mathbb{1}^T,$$

which is on the same form as the Google matrix. Thus, PR is a special case of the Markov Chain Random Walk with Restarts using a uniform seed distribution.

Authors refer to this procedure by different names, one being the *personalized* PageRank (PPR). This is the name that will be used for the rest of this thesis. Although this is a short section, PPR is essential for the main contributions of this thesis. More theory on this is presented in Ch. 8.

3.2 Ranking multi attribute data

The theory so far in this thesis has revolved around link structures and general stochastic matrices without the need of having real measured data. When working with measured data in machine learning, each data point is represented by a vector. Each vector consists of multiple numerical measurements, or *features*. The dimensionality of the vectors is the same as the number of features. In this thesis, non-random vectors will be denoted by lower case bold letters. For a sample, the data points are numbered. For instance $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ is a sample with N data points. Now that the definition and notation on data is established, an algorithm for ranking on multi attribute data can be presented. This algorithm is defined in the paper *Ranking on Data Manifolds* [12], where the term *manifold* refers to structures in the data. Manifolds in data are often low dimensional structures within a high dimensional space. This specific algorithm is chosen because it is very similar to the basis of the main method developed in this thesis which is defined in Ch. 8. However, it has its limitations with respect to the goal of this thesis. Firstly, the algorithm is based on a similarity measure between data points containing a width parameter which can greatly influence the result. Secondly, the algorithm is solely used for ranking[2]. The goal for the methods in this thesis is to jointly rank and cluster data using a common framework without parameter dependence.

Ranking on Data Manifolds as defined by Zhou et al. [12] is similar to the personalized formulation of the PR as presented in Sec. 3.1.1. The main difference is that this algorithm is based on multi attribute data in an Euclidean space and that the stochastic matrix is exchanged for a symmetrically normalized weight matrix in a weighted connected graph. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ be a sample. Imagine that the goal is to find the data point which are relevant to a specific query. Let $\mathbf{y} = \begin{pmatrix} y_1 & y_2 & \cdots & y_N \end{pmatrix}^T$ be the query, where $y_i = 1$ if x_i is in the query and zero otherwise. The algorithm is summarized in Alg. 1. Intuitively, the scores of the data points are *propagated* through the graph iteratively until convergence. The parameter α decides the importance of the query versus the graph itself. The parameter σ decides the width of the weight function on each data point. The choice of this width parameter could greatly influence the result. Selecting an appropriate value for σ is a challenge in general for methods based on this type of weight function.

[2]Previously, a similar approach has been used for semi supervised learning [39]

Algorithm 1 Ranking on Data Manifolds

```
Input: Data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, query vector $\mathbf{y}$, distance metric $d(\mathbf{x}_i, \mathbf{x}_j)$
    and kernel width $\sigma$.
 1: Calculate pairwise distances between every data point and sort them in
    ascending order.
 2: Construct a connected graph by sequentially connecting two vertices ac-
    cording to the ordered pairwise distances until the graph is connected.
 3: Calculate the affinity matrix, $\mathbf{W}$ by $w_{ij} = \exp\left\{-\frac{d^2(\mathbf{x}_i,\mathbf{x}_j)}{2\sigma^2}\right\}$ if vertex $i$
    and vertex $j$ is connected and zero otherwise. Note that $w_i i = 0$.
 4: Symmetrically normalize $\mathbf{W}$ by $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D} = \mathrm{diag}(d_i) =$
    $\mathrm{diag}(\sum_{j=1}^{N} w_{ij})$.
 5: Calculate the score of each data point by the iterative sequence $\mathbf{f}(t+1) =$
    $\alpha\mathbf{S}\mathbf{f}(t) + (1-\alpha)\mathbf{y}$ until convergence, where $\alpha \in [0, 1\rangle$.
 6: Rank the data points according to their score. The largest score is ranked
    first.
Output: Scores of the data points.
```

Chapter 4

Data representation and dimensionality reduction

As the title implies, the methods in this section are used to reduce the dimensionality of the data. This is often done as a pre-processing step, before analyzing the data using for instance clustering or classification. We do this to avoid dimensionality problems, like the curse of dimensionality[1]. In addition to this, the data often lies on lower dimensional structures, or *manifolds*, in the higher dimensional space. Thus, in practice, the data is lower dimensional but is represented in a high dimensional space. If we are able to preserve the structure in the data using a lower dimensional representation, the data will be easier to analyze.

This chapter starts with a simple statistical motivated approach called Principal Component Analysis (PCA). It will become apparent that PCA is a linear method which is not beneficial when working with nonlinear structures in the data. Thus, a nonlinear extension called Kernel PCA (KPCA) will be presented. KPCA is an essential part of the theory in Ch. 6–Ch. 8.

4.1 Principal Component Analysis

Principal Component Analysis (PCA) [40] is a method used to remove redundant information in the data by decorrelating its components. Furthermore, this is used to represent the data using just a few variables (dimensions) while maintaining most of the *variance*. That is, we want to project the data to a vector space which preserves most of the energy of the data and removes correlation between variables. Note that PCA is a *linear* method. Thus, PCA will not aid when analysing nonlinear data.

[1]Euclidean distances does not make sense in high dimensions.

Historically, PCA has been used in statistical analysis to help interpret the data [41]. In machine learning, PCA is often used for *dimensionality reduction* or *feature extraction*. In other fields, PCA has been proven useful, for instance in signal de-noising and image processing. One could for instance take a look at [42] for a nice introduction to PCA in signal processing. In [42], both image processing (using eigen images) and de-noising speech signals is considered.

4.1.1 Decorrelating the components

Let $\mathbf{X} \in \mathbb{R}^p$ be a random vector with expectation $\boldsymbol{\mu}_\mathbf{X}$ and covariance matrix

$$\boldsymbol{\Sigma}_\mathbf{X} = \mathbb{E}\left[\left(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X}\right)\left(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X}\right)^T\right] \tag{4.1}$$

and let

$$\mathbf{Y} = \mathbf{A}^T\mathbf{X}, \tag{4.2}$$

where $\mathbf{A}$ is some nonrandom transformation matrix. Then, the following holds:

$$\begin{aligned}\boldsymbol{\mu}_\mathbf{Y} &= \mathbb{E}\left[\mathbf{A}^T\mathbf{X}\right] \\ &= \mathbf{A}^T\,\mathbb{E}\left[\mathbf{X}\right] \\ &= \mathbf{A}^T\boldsymbol{\mu}_\mathbf{X}. \end{aligned} \tag{4.3}$$

$$\begin{aligned}\boldsymbol{\Sigma}_\mathbf{Y} &= \mathbb{E}\left[\left(\mathbf{Y} - \boldsymbol{\mu}_\mathbf{Y}\right)\left(\mathbf{Y} - \boldsymbol{\mu}_\mathbf{Y}\right)^T\right] \\ &= \mathbb{E}\left[\left(\mathbf{A}^T\mathbf{X} - \mathbf{A}^T\boldsymbol{\mu}_\mathbf{X}\right)\left(\mathbf{A}^T\mathbf{X} - \mathbf{A}^T\boldsymbol{\mu}_\mathbf{X}\right)^T\right] \\ &= \mathbb{E}\left[\mathbf{A}^T\left(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X}\right)\left(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X}\right)^T\mathbf{A}\right] \\ &= \mathbf{A}^T\,\mathbb{E}\left[\left(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X}\right)\left(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X}\right)^T\right]\mathbf{A} \\ &= \mathbf{A}^T\boldsymbol{\Sigma}_\mathbf{X}\mathbf{A}. \end{aligned} \tag{4.4}$$

To ensure that the variables in $\mathbf{Y}$ are uncorrelated, the covariance matrix $\boldsymbol{\Sigma}_\mathbf{Y}$ needs to be *diagonal*. Now, let $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p$ be the *normalized* eigenvectors of $\boldsymbol{\Sigma}_\mathbf{X}$ with the corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$. Construct the eigenvector matrix $\mathbf{U}$ with the eigenvectors as its columns and the diagonal eigenvalue matrix $\boldsymbol{\Lambda}$ with the corresponding eigenvalues along the main diagonal. That is

$$\mathbf{U} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_p \end{pmatrix}$$
$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_p \end{pmatrix}$$

The normalized eigenvectors of $\mathbf{\Sigma_X}$ are orthogonal since the covariance matrix is symmetric, so $\mathbf{U}$ is orthogonal. This implies that

$$\mathbf{U}^{-1} = \mathbf{U}^T. \tag{4.5}$$

Now, we can *diagonalize* the covariance matrix $\mathbf{\Sigma_X}$ by

$$\mathbf{\Sigma_X} = \mathbf{U\Lambda U}^T. \tag{4.6}$$

Substituting Eq. (4.6) into Eq. (4.4) yields

$$\mathbf{\Sigma_Y} = \mathbf{A}^T\mathbf{U\Lambda U}^T\mathbf{A}. \tag{4.7}$$

The whole point of this operation is to find a transformation matrix $\mathbf{A}$ which makes the covariance matrix $\mathbf{\Sigma_Y}$ diagonal. We see that by letting $\mathbf{A} = \mathbf{U}$ and using Eq. (4.5), Eq. (4.7) is reduced to

$$\mathbf{\Sigma_Y} = \mathbf{\Lambda}, \tag{4.8}$$

which by definition is diagonal. Thus, the components of $\mathbf{Y}$ are uncorrelated. From the covariance matrix, we see that for $\mathbf{Y} = \begin{pmatrix} Y_1 & Y_2 & \cdots & Y_p \end{pmatrix}^T$ we have $\mathrm{Var}(Y_i) = \lambda_i,\ i = 1, 2, \dots, p$. Furthermore, we have that

$$\begin{aligned} \sum_{i=1}^{p} \mathrm{Var}(X_i) &= \mathrm{Trace}(\mathbf{\Sigma_X}) \\ &= \mathrm{Trace}\left(\mathbf{U\Lambda U}^T\right) \\ &= \mathrm{Trace}\left(\mathbf{U}^T\mathbf{U\Lambda}\right) \\ &= \mathrm{Trace}(\mathbf{\Lambda}) \\ &= \sum_{i=1}^{p} \lambda_i \\ &= \sum_{i=1}^{p} \mathrm{Var}(Y_i), \end{aligned}$$

so the total variance is preserved for the transformed data.

Note that $\mathbf{U}^T$ is the change of basis matrix from the standard basis to the eigenbasis of the covariance matrix $\mathbf{\Sigma}$. Since the basis vectors are orthonormal, the linear transformation performs a rotation of the coordinate system until the components of the data are uncorrelated.

4.1.2 PCA on a sample

In real life applications, we work with observed data. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ be a sample of p-dimensional vectors with the *sample* covariance matrix $\mathbf{S}_\mathbf{x}$ and let $\widetilde{\mathbf{u}}_1, \widetilde{\mathbf{u}}_2, \ldots, \widetilde{\mathbf{u}}_p$ be the eigenvectors of $\mathbf{S}_\mathbf{x}$ with the corresponding eigenvalues $\widetilde{\lambda}_1, \widetilde{\lambda}_2, \ldots, \widetilde{\lambda}_p$, where $\widetilde{\lambda}_1 \geq \widetilde{\lambda}_2 \geq \ldots \geq \widetilde{\lambda}_p$. We call the eigenbasis the *principal components* of $\mathbf{S}_\mathbf{x}$, where the principal component is the most dominant eigenvector. The idea now is to represent the data using the ℓ principal components of $\mathbf{S}_\mathbf{x}$. For dimensionality reduction purposes, $\ell < p$. To do this, we construct the *truncated* eigenvector matrix

$$\widetilde{\mathbf{U}} = \begin{pmatrix} \widetilde{\mathbf{u}}_1 & \widetilde{\mathbf{u}}_2 & \cdots & \widetilde{\mathbf{u}}_\ell \end{pmatrix} \tag{4.9}$$

and let

$$\mathbf{y}_i = \widetilde{\mathbf{U}}^T \mathbf{x}_i, \; i = 1, 2, \ldots, N. \tag{4.10}$$

Then $\mathbf{y}_i$ will be ℓ-dimensional vectors. Furthermore, the sample covariance matrix of $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ will be diagonal with $\widetilde{\lambda}_1, \widetilde{\lambda}_2, \ldots, \widetilde{\lambda}_\ell$ along the main diagonal which implies that the components of $\mathbf{y}_i$ $i = 1, 2, \ldots, N$ are uncorrelated. The algorithm is summarized in Alg. 2.

Example 4.1.1 (Bivariate normally distributed data)**.** For multivariate normally distributed data, the eigenvectors of the covariance matrix points in the direction of the semi-axes of the constant density hyperellipsiods of the density function [43]. Fig. 4.1a shows a simulated dataset of bivariate normally distributed data. The curves in the plot are the constant density ellipses of the density function, while the arrows are the eigenvectors of the covariance matrix. Fig. 4.1b shows the decorrelated dataset. It is readily seen that this corresponds to a change of basis to the eigenbasis of the covariance matrix. Fig. 4.1c illustrates how dimensionality reduction using PCA works. The red dots are the projections onto each of the two first principal components. From the estimated density functions (the black lines), we see that the projection onto the eigenvector which points in the direction of maximum variance in the original data has the largest variance of the two projections. Fig. 4.1d shows the projection onto this principal component.

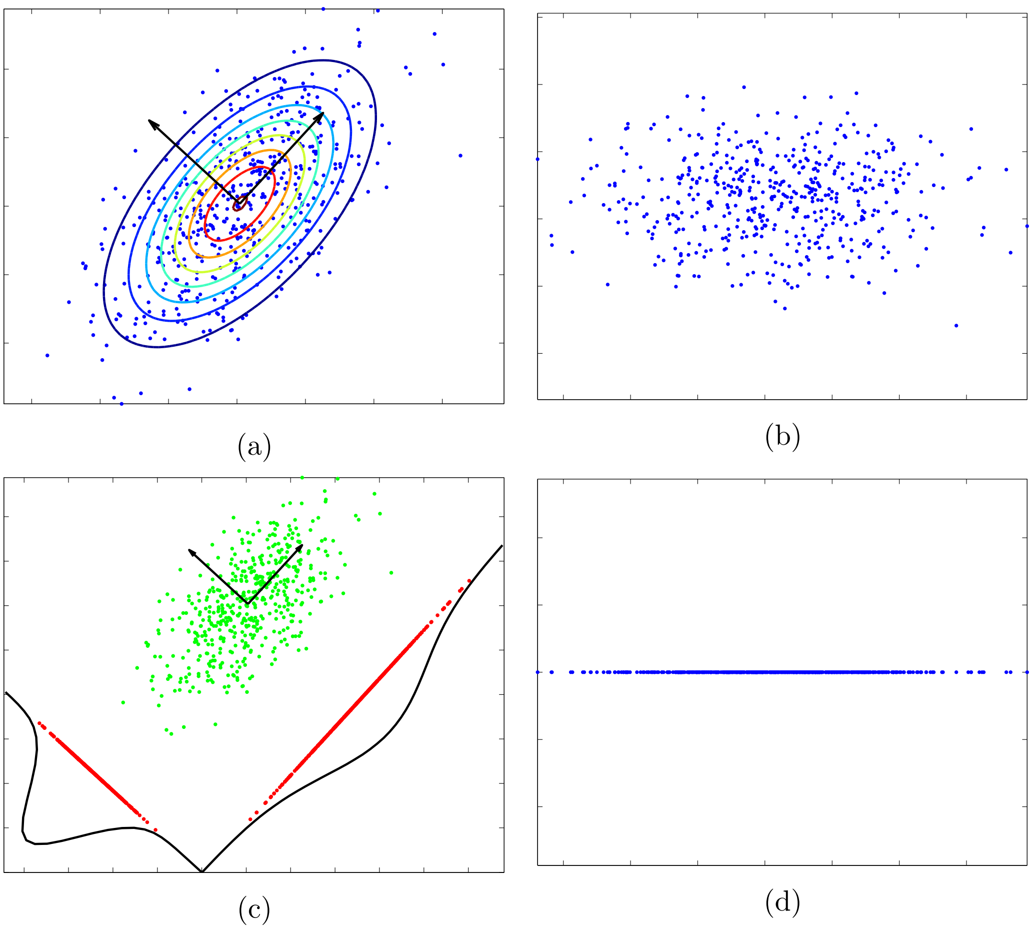


Figure 4.1: (a): Bivariate normally distributed data with constant density lines and eigen-decomposition. (b): Decorrelated data. (c): Projection of the data onto the two principal components and estimated density functions. The red dots are the projections, while the black lines are the estimated density functions. (d): Projection of the data onto the first principal component.

Algorithm 2 Principal Component Analysis

```
Input: Data vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in \mathbb{R}^p$ and the number of principal components $\ell$.
1: Calculate the sample covariance matrix
       $$\mathbf{S}_x = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T,$$
   where
       $$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i.$$
2: Compute the $\ell$ largest eigenvectors $\widetilde{\mathbf{u}}_1, \widetilde{\mathbf{u}}_2, \dots, \widetilde{\mathbf{u}}_\ell$ of $\mathbf{S}_x$ with the corresponding eigenvalues $\widetilde{\lambda}_1, \widetilde{\lambda}_2, \dots, \widetilde{\lambda}_\ell$, where $\widetilde{\lambda}_1 \geq \widetilde{\lambda}_2 \geq \dots \geq \widetilde{\lambda}_\ell$.
3: Construct the truncated eigenvector matrix
       $$\widetilde{\mathbf{U}} = \begin{pmatrix} \widetilde{\mathbf{u}}_1 & \widetilde{\mathbf{u}}_2 & \cdots & \widetilde{\mathbf{u}}_\ell \end{pmatrix}$$
4: Compute
       $$\mathbf{y}_i = \widetilde{\mathbf{U}}^T\mathbf{x}_i, \; i = 1, 2, \dots, N.$$
Output: Data vectors $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N \in \mathbb{R}^\ell$ with uncorrelated components.
```

4.2 Kernel Principal Component Analysis

Kernel Principal Component Analysis (KPCA) [44, 45, 46] is a nonlinear extension of the theory of PCA. The nonlinear property of KPCA is often useful for data with structures which cannot be well represented in a linear subspace. For instance, if we are interested in extracting features as a preprocessing step for classification of nonlinearly separable data, canonical PCA will not be very helpful to aid in discriminating between classes. KPCA, however, may be able to extract features in such a way that the nonlinearly separable data becomes linearly separable.

KPCA uses the theory of *Mercer Kernels* to perform an implicit nonlinear transformation of the data and performs PCA in this (possibly unknown) kernel space. KPCA has been used in several applications. This includes face recognition [47, 48], de-noising [49, 50] and texture classification [51]. One should note that other nonlinear approaches to PCA has been proposed. For instance [52, 53, 54, 55].

This chapter starts with a short introduction to Mercer Kernels. This will then be used in combination with the theory of PCA to derive KPCA. Examples will be presented which compares KPCA and PCA.

4.2.1 Mercer Kernels

Let $\mathbf{x}$ be a data point in the feature space $\mathcal{X}$ and let $\mathcal{H}$ be some (possibly infinitely dimensional) Hilbert space. Define the nonlinear transformation

$$\begin{aligned}\Phi: \quad & \mathcal{X} \to \mathcal{H} \\ & \mathbf{x} \mapsto \Phi(\mathbf{x}).\end{aligned} \tag{4.11}$$

In general, the mapping function $\Phi(\cdot)$ is unknown but there exists a connection between a *Kernel function*

$$\kappa: \quad \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

and inner products in the Hilbert space $\mathcal{H}$. A kernel function satisfies

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}. \tag{4.12}$$

This means that the kernel function $\kappa(\cdot, \cdot)$ calculates inner products in $\mathcal{H}$ without any explicit knowledge of the mapping $\Phi(\cdot)$. Thus, any machine learning algorithm that relies on inner products between the data points could be implemented using Eq. (4.12) as a way to map the data *implicitly* to $\mathcal{H}$.

The existence of a kernel function $\kappa(\cdot, \cdot)$ and the corresponding Hilbert space $\mathcal{H}$ is ensured by the *Moore-Aronszajn* theorem [56] and *Mercer's* theorem [57].

Theorem 1 (Mercer's theorem). *Let $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$. A function $\kappa(\cdot, \cdot)$ is a Mercer kernel if and only if $\kappa(\cdot, \cdot)$ is symmetric and*

$$\sum_i \sum_j a_i a_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0,$$

where $a_k \in \mathbb{R}$. Also, there exists a Hilbert space $\mathcal{H}$ and some mapping function

$$\Phi: \quad \mathcal{X} \to \mathcal{H}$$

such that

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$$

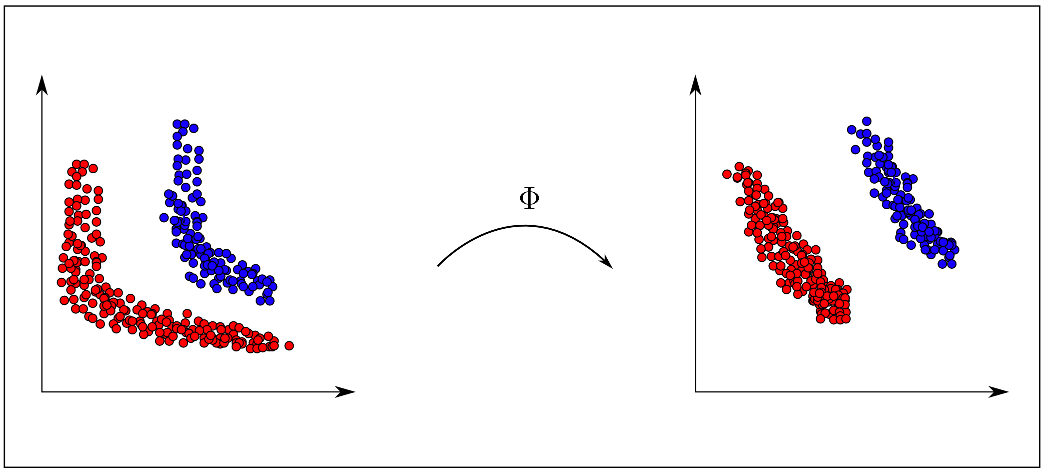


Figure 4.2: Nonlinear mapping Φ.

By defining the matrix

$$\mathbf{K} = \{\kappa(\mathbf{x}_i, \mathbf{x}_j)\}, \quad i, j = 1, 2, \ldots, n, \tag{4.13}$$

we see that Mercer's theorem implies that **K** is a positive semidefinite matrix.

There exists numerous valid Mercer kernels [58]. The one that is probably the most known and used in practice is the *Gaussian* kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}, \tag{4.14}$$

where σ controls the width of the kernel.

Why use Mercer Kernels?

There are numerous reasons why the kernel methods have become popular in machine learning. Many of the methods in machine learning are originally linear. If the structures in the data are nonlinear, the linear methods could fail. However, if the mathematical foundation of the methods can be expressed solely as a function of inner products between feature vectors, a Mercer kernel could perform an implicit mapping to a possibly infinitely dimensional space where the data might be linearly separable. The Support Vector Machine [59, 60] is a popular classification method where this is possible. An illustration of this mapping is shown in Fig. 4.2. The benefit of using an implicit mapping and not an explicit mapping is best shown with an example.

Example 4.2.1 (Polynomial kernel.)**.** The polynomial kernel [61] is defined as

$$\kappa(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^T\mathbf{y})^d. \tag{4.15}$$

Considering a low dimensional feature space with $\mathbf{x},\mathbf{y}\in\mathbb{R}^2$ and a quadratic kernel with $d=2$, we get

$$\begin{aligned}\kappa(\mathbf{x},\mathbf{y}) &= (1+\mathbf{x}^T\mathbf{y})^2 \\ &= 1^2 + 2\mathbf{x}^T\mathbf{y} + \left(\mathbf{x}^T\mathbf{y}\right)^2 \\ &= 1 + 2\sum_{i=1}^{2} x_iy_i + \sum_{i=1}^{2} x_iy_i \sum_{j=1}^{2} x_jy_j \\ &= 1 + \sqrt{2}x_1\sqrt{2}y_1 + \sqrt{2}x_2\sqrt{2}y_2 + x_1^2y_1^2 + x_2^2y_2^2 + \sqrt{2}x_1x_2\sqrt{2}y_1y_2 \\ &= \langle\Phi(\mathbf{x}),\Phi(\mathbf{y})\rangle,\end{aligned}$$

where

$$\Phi(\mathbf{x}) = \begin{pmatrix}1 & \sqrt{2}x_1 & \sqrt{2}x_2 & \sqrt{2}x_1x_2 & x_1^2 & x_2^2\end{pmatrix}^T.$$

Here, we were able to find an explicit mapping Φ such that $\kappa(\mathbf{x},\mathbf{y}) = \langle\Phi(\mathbf{x}),\Phi(\mathbf{y})\rangle$. If we were to first do this mapping and then compute the inner product in the kernel feature space, we see that both the computational complexity and the memory complexity are increased in comparison to just evaluating the kernel function. Also, if we increase d or increase the dimensionality of the data in the feature space, the dimensionality of the mapped data increases drastically. So if we only need the inner product in the kernel feature space, it is unnecessary to map the data first and then compute the inner product when it is easy to compute via the kernel function.

In addition to the simplicity of computing inner products implicitly using the kernel function, we do not always know the explicit mapping Φ. For instance with the Gaussian kernel, the kernel feature space is *infinitely dimensional* [62]. And thus, an explicit mapping is not possible.

In some situations, it is not necessarily the nonlinear property of the kernel methods which is the most appealing. As seen in for instance [62], it is possible to construct kernel matrices for nonvectorial data like documents and DNA. This allows us to use the kernel methods on types of data which regular methods are not compatible with.

4.2.2 Derivation of KPCA

Let $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N\in\mathcal{X}$ be a sample and let $\Phi(\mathbf{x})$ be a nonlinear transformation as defined in (4.11). Consider the correlation matrix

$$\boldsymbol{\mathcal{R}} = \mathbb{E}\left[\Phi(\mathbf{x})\Phi(\mathbf{x})^T\right],$$

which is approximated by

$$\mathbf{R} = \frac{1}{N}\sum_{i=1}^{N}\Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^T. \tag{4.16}$$

Let $\mathbf{v}$ be an eigenvector of $\mathbf{R}$ with the corresponding eigenvalue λ. Then

$$\begin{aligned}\mathbf{R}\mathbf{v} &= \lambda\mathbf{v}\\ \frac{1}{N}\sum_{i=1}^{N}\Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^T\mathbf{v} &= \lambda\mathbf{v}.\end{aligned} \tag{4.17}$$

Now, define

$$a(i) = \frac{1}{\lambda N}\Phi(\mathbf{x}_i)^T\mathbf{v}. \tag{4.18}$$

By substitution, we have

$$\mathbf{v} = \sum_{i=1}^{N} a(i)\Phi(\mathbf{x}_i). \tag{4.19}$$

From this, we see that $\mathbf{v} \in \text{Span}\{\Phi(\mathbf{x}_i),\ i = 1, 2, \ldots, N\}$. By left multiplying Eq. (4.17) with $\Phi(\mathbf{x}_k)$ and defining the kernel matrix

$$\mathbf{K} = \{\kappa(\mathbf{x}_i, \mathbf{x}_j)\}_{N\times N}$$

and the vector $\mathbf{a} = \begin{pmatrix} a(1) & a(2) & \cdots & a(N)\end{pmatrix}^T$, we can show that

$$\mathbf{K}\mathbf{a} = N\lambda\mathbf{a}. \tag{4.20}$$

That is, $\mathbf{a}$ is an eigenvector of $\mathbf{K}$ with the corresponding eigenvalue $\lambda^* = N\lambda$. But how is this used for PCA in kernel space? Recall that the projection of $\Phi(\mathbf{x}_k)$ onto the principal component of $\mathbf{R}$ is given by

$$\begin{aligned}\mathbf{v}^T\Phi(\mathbf{x}_k) &= \sum_{i=1}^{N} a(i)\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_k)\\ &= \sum_{i=1}^{N} a(i)\kappa(\mathbf{x}_i, \mathbf{x}_k).\end{aligned} \tag{4.21}$$

From this, we see that the projection is completely defined by the kernel matrix and its eigenvectors. From the theory on principal component analysis, we know that we need the eigenvectors of the covariance matrix to be normalized. That is

$$\begin{aligned}
\mathbf{v}^T\mathbf{v} &= 1 \\
\sum_{i=1}^{N} a(i)\Phi(\mathbf{x}_i)^T \sum_{j=1}^{N} a(j)\Phi(\mathbf{x}_j) &= 1 \\
\sum_{i=1}^{N}\sum_{j=1}^{N} a(i)a(j)\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j) &= 1 \\
\sum_{i=1}^{N}\sum_{j=1}^{N} a(i)a(j)\kappa(\mathbf{x}_i,\mathbf{x}_j) &= 1 \\
\mathbf{a}^T\mathbf{K}\mathbf{a} &= 1 \\
\lambda^*\mathbf{a}^T\mathbf{a} &= 1 \\
\|\mathbf{a}\|^2 &= \frac{1}{\lambda^*} \\
\|\mathbf{a}\| &= \frac{1}{\sqrt{\lambda^*}}.
\end{aligned}$$

So, we can ensure that $\mathbf{v}$ is normalized by scaling $\mathbf{a}$. Note that $\lambda^* = N\lambda$ is the corresponding eigenvalue of $\mathbf{a}$, while λ is the corresponding eigenvalue of $\mathbf{v}$. Now, let $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_\ell$ be the ℓ dominant eigenvectors of $\mathbf{K}$, with $\lambda_1^* \geq \lambda_2^* \geq \ldots \geq \lambda_\ell^*$. Using (4.21), the projection of $\Phi(\mathbf{x}_k)$ onto the ℓ principal components of $\mathbf{R}$ is given by

$$\begin{pmatrix} \mathbf{v}_1^T\Phi(\mathbf{x}_k) \\ \mathbf{v}_2^T\Phi(\mathbf{x}_k) \\ \vdots \\ \mathbf{v}_\ell^T\Phi(\mathbf{x}_k) \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{\lambda_1^*}}\sum_{i=1}^{N} a_1(i)\kappa(\mathbf{x}_i,\mathbf{x}_k) \\ \frac{1}{\sqrt{\lambda_2^*}}\sum_{i=1}^{N} a_2(i)\kappa(\mathbf{x}_i,\mathbf{x}_k) \\ \vdots \\ \frac{1}{\sqrt{\lambda_\ell^*}}\sum_{i=1}^{N} a_\ell(i)\kappa(\mathbf{x}_i,\mathbf{x}_k) \end{pmatrix}. \tag{4.22}$$

The algorithm is summarized in Alg. 3.

Centering

In the previous discussion, we derived KPCA without any assumptions on the data. If we assume that the data is centered, i.e.

$$\mathbb{E}\left[\Phi(\mathbf{x})\right] = \mathbf{0},$$

we see that the correlation matrix and the covariance matrix are equal. We are able to force the data in kernel space to be centered by modifying the kernel matrix. By instead considering the covariance matrix of

Algorithm 3 Kernel Principal Component Analysis

```
Input: Data vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^p$, the number of principal components $\ell$ and a valid kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$.
1: Construct the kernel matrix $\mathbf{K} = \{k_{ij}\}_{N \times N}$, where $k_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.
2: Compute the $\ell$ largest eigenvectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_\ell$ of $\mathbf{K}$ and the corresponding eigenvalues $\lambda_1^*, \lambda_2^*, \ldots, \lambda_\ell^*$, where $\lambda_1^* \geq \lambda_2^* \geq \ldots \geq \lambda_\ell^*$.
3: Calculate the projection of $\Phi(\mathbf{x}_k)$ onto the $\ell$ principal components in kernel space
   $$\mathbf{z}_k = \begin{pmatrix} \frac{1}{\sqrt{\lambda_1^*}} \sum_{i=1}^N a_1(i)\kappa(\mathbf{x}_i, \mathbf{x}_k) \\ \frac{1}{\sqrt{\lambda_2^*}} \sum_{i=1}^N a_2(i)\kappa(\mathbf{x}_i, \mathbf{x}_k) \\ \vdots \\ \frac{1}{\sqrt{\lambda_\ell^*}} \sum_{i=1}^N a_\ell(i)\kappa(\mathbf{x}_i, \mathbf{x}_k) \end{pmatrix},$$
   for $k = 1, 2, \ldots, N$.
Output: Data vectors $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N \in \mathbb{R}^\ell$ with uncorrelated components.
```

$\Phi(\mathbf{x}_i) - \frac{1}{N}\sum_{j=1}^N \Phi(\mathbf{x}_j)$, it is possible to show that we can use the same methodology by modifying the kernel matrix. Instead of using $\mathbf{K}$ as defined in Eq. (4.13), we use

$$\widetilde{\mathbf{K}} = \mathbf{K} - \mathbb{1}_N\mathbf{K} - \mathbf{K}\mathbb{1}_N + \mathbb{1}_N\mathbf{K}\mathbb{1}_N. \tag{4.23}$$

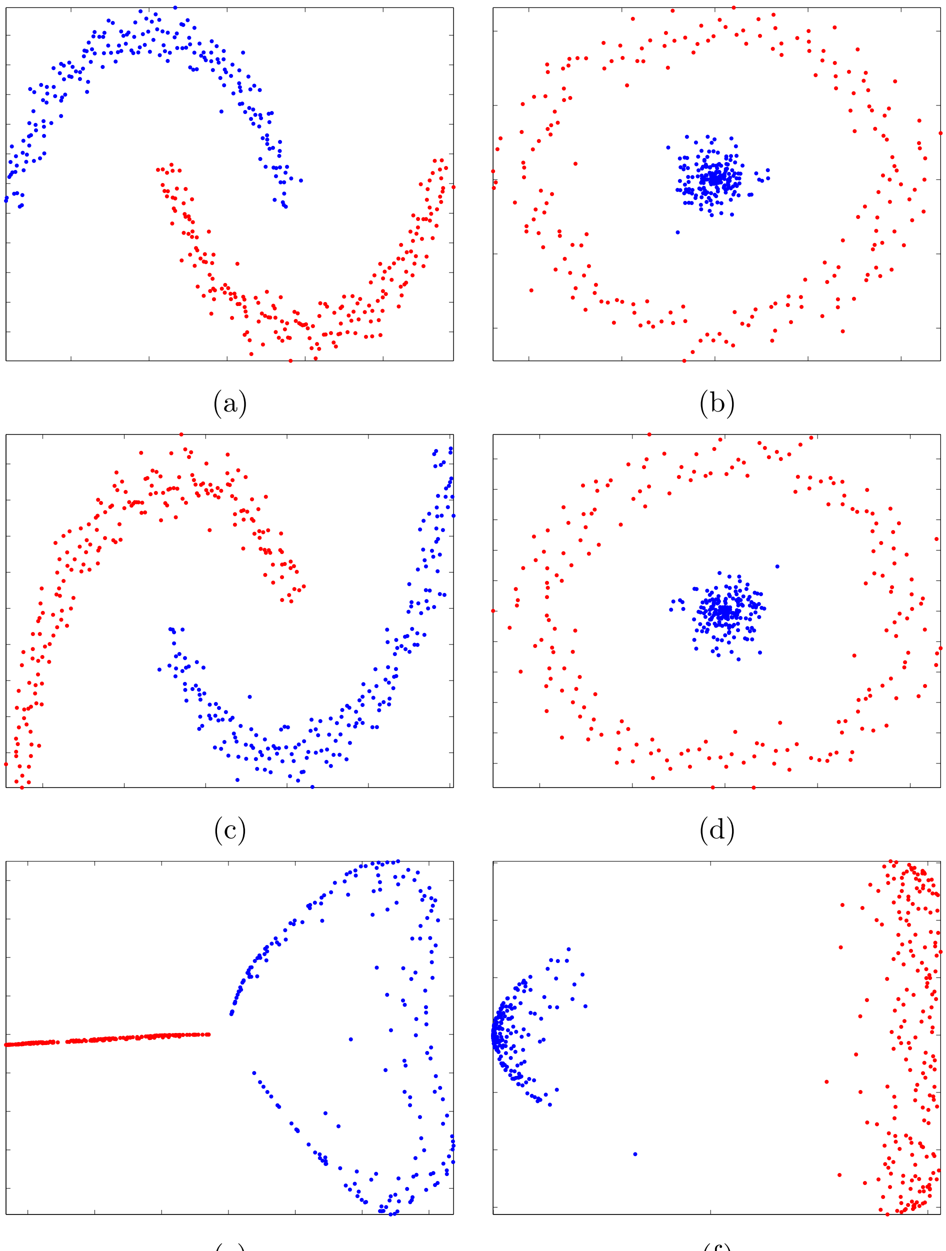
Here, $\mathbb{1}_N = \left\{\frac{1}{N}\right\}_{N \times N}$. For details, see [44].

Now, the question is whether to center the data or not. This question will not be answered here. The interested reader is pointed towards [63] for a recent review on this topic. However, we will state that there are methods where centering does not make sense. For instance Kernel Entropy Component Analysis (KECA) [64]. As we will see in Ch. 7 and Ch. 8, centering the kernel matrix does not make sense in these Markov Chain based methods.

The Empirical Kernel Space

As stated earlier, the kernel space might be infinite dimensional. When using KPCA, the data is said to be embedded in the *empirical kernel space*. This is because it is embedded to a *finite* dimensional space *preserving the inner products* in the kernel space.

Example 4.2.2 (Comparison of KPCA and PCA)**.** Fig. 4.3 compares KPCA to PCA for two toy datasets. We see that the canonical PCA just rotates the coordinate system until the variables are uncorrelated. The effect of the implicit nonlinear transformation is readily seen in the KPCA plots. The original data was nonlinearly separable. After applying KPCA, the data is linearly separable which might be beneficial for further analysis. Both kernel matrices were constructed using a Gaussian kernel with $\sigma = 0.1$ and $\sigma = 5$ for the crescent moon data and the circle data respectively. This shows that the value of this parameter is very dependent on the situation. Note that in this example, the dimensionality of the output is the same as the dimensionality of the input. We could have high dimensional input data and use KPCA or PCA to extract just a few features. This might reveal otherwise unknown structures in the data which could aid in further analysis. In the case of KPCA, we could also have low dimensional data and project these to a higher dimensional kernel feature space.



(a) (b) (c) (d) (e) (f)

Figure 4.3: (a-b): Original data. (c-d): PCA projection. (e-f): KPCA projection using a Gaussian kernel with $\sigma = 0.1$ in (e) and $\sigma = 5$ in (f).

4.2.3 In-sample KPCA

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^p$ be a sample and let $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N \in \mathbb{R}^\ell$ be the projections of the sample onto the ℓ principal components in kernel feature space. Let $\mathbf{K}$ be a kernel matrix with the ℓ most dominant eigenvectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_\ell$ and the corresponding eigenvalues $\lambda_1^*, \lambda_2^*, \ldots, \lambda_\ell^*$. It is easy to see that if we define the eigenvector matrix

$$\mathbf{E} = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_\ell \end{pmatrix}$$

and the diagonal eigenvalue matrix

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1^* & 0 & \cdots & 0 \\ 0 & \lambda_2^* & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_\ell^* \end{pmatrix},$$

a combined calculation of all the projections is done with

$$\begin{aligned} \mathbf{Z} &= \begin{pmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_N^T \end{pmatrix} \\ &= \mathbf{K}^T \mathbf{E} \mathbf{\Lambda}^{-\frac{1}{2}}. \end{aligned} \tag{4.24}$$

Note that $\mathbf{K}$ is symmetrical, so $\mathbf{K}^T = \mathbf{K}$. Furthermore, $\mathbf{E}$ is orthogonal. This implies that $\mathbf{E}^{-1} = \mathbf{E}^T$. Thus, the kernel matrix can be diagonalized by $\mathbf{K} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$. Using these properties in Eq. (4.24) yields

$$\begin{aligned} \mathbf{Z} &= \mathbf{K}^T \mathbf{E} \mathbf{\Lambda}^{-\frac{1}{2}} \\ &= \mathbf{K} \mathbf{E} \mathbf{\Lambda}^{-\frac{1}{2}} \\ &= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T \mathbf{E} \mathbf{\Lambda}^{-\frac{1}{2}} \\ &= \mathbf{E}\mathbf{\Lambda}^{\frac{1}{2}}. \end{aligned} \tag{4.25}$$

Thus, all the projections can be done in one matrix operation after constructing the eigenvector matrix and the eigenvalue matrix. We also see that the projection only depends on the eigenvalues and eigenvectors of the kernel matrix, not the kernel function itself. If KPCA is used in combination with for instance classification or regression, we have both training data and test data. In this case, we would need to use the general expression in Eq. (4.22) for the *test* data since the systems are trained based on the projection of the training data. In clustering, however, we can use the expression in Eq. (4.25).

Chapter 5

Clustering

This chapter presents the clustering algorithms needed to build the foundation for the main theory in this thesis. A more in-depth presentation of available clustering algorithms can be found in [58, 65, 66, 67, 68].

5.1 Generalized Hard Algorithmic Scheme

The Generalized Hard Algorithmic Scheme (GHAS) [58] is a subset of clustering methods based on minimizing a specific cost funtion. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be the data vectors we want to cluster and let $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ be their associated membership functions. Assume we want to cluster our data into k clusters, $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$. For the GHAS, we assume *hard* membership functions. That is, for

$$\mathbf{u}_i = \begin{bmatrix} u_{i1} \\ u_{i2} \\ \vdots \\ u_{ik} \end{bmatrix},$$

we have

$$u_{ij} \in \{0, 1\} \tag{5.1}$$

$$\sum_{j=1}^{k} u_{ij} = 1. \tag{5.2}$$

This means that $u_{ij} = 1$ if $\mathbf{x}_i$ is assigned to $\mathcal{C}_j$ and zero otherwise. Now, assume we can represent the cluster $\mathcal{C}_j$ by a single vector, $\boldsymbol{\theta}_j, \quad j = 1, 2, \ldots, k$.

This vector is called the *cluster representative* for $\mathcal{C}_j$. Then, the cost function for the GHAS is defined as

$$J(\mathbf{X},\mathbf{U},\mathbf{\Theta}) = \sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij} d(\mathbf{x}_i, \boldsymbol{\theta}_j), \tag{5.3}$$

where $d(\mathbf{x}_i, \boldsymbol{\theta}_j)$ is some dissimilarity measure between the data point, $\mathbf{x}_i$ and the cluster representative $\boldsymbol{\theta}_j$. The arguments in the cost function are matrices containing the data points, membership functions and cluster representatives.

The intuition behind the cost function is that since $u_{ij} = 1$ if and only if $\mathbf{x}_i$ is assigned to $\mathcal{C}_j$, we obtain the minimum of Eq. (5.3) when *all* the data vectors are assigned to the cluster with the *least* dissimilar cluster representative. We also need to update the cluster representatives to minimize the dissimilarity between the data vectors and the cluster representatives. This ensures that the *within cluster* dissimilarity is minimized.

Optimizing a cost function is normally done by differentiating it with respect to the unknown variables and equating the result to zero. In this case, the unknown variables are the membership functions and the cluster representatives. However, we see that the membership functions are not differentiable because of their hard values. We will solve this by minimizing Eq. (5.3) in an iterative manner. There are two stages for each iteration:

1. Fix $\mathbf{\Theta}$ and set $u_{ij} = 1$ if $d(\mathbf{x}_i, \boldsymbol{\theta}_j) = \min_{m=1,2,\ldots,k} d(\mathbf{x}_i, \boldsymbol{\theta}_m)$. Set $u_{ij} = 0$ otherwise. Do this for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$.

2. Fix $\mathbf{U}$ and minimize Eq. (5.3) with regards to the cluster representatives, $\boldsymbol{\theta}_j, \quad j = 1, 2, \ldots, k$.

For the cluster representative, $\boldsymbol{\theta}_z$, we get

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}_z} J(\mathbf{X},\mathbf{U},\mathbf{\Theta}) &= \frac{\partial}{\partial \boldsymbol{\theta}_z} \sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij} d(\mathbf{x}_i, \boldsymbol{\theta}_j) \\ &= \sum_{i=1}^{n} u_{iz} \frac{\partial}{\partial \boldsymbol{\theta}_z} d(\mathbf{x}_i, \boldsymbol{\theta}_z).\end{aligned}$$

Thus, the $\boldsymbol{\theta}_z$ which minimizes Eq. (5.3) has to satisfy

$$\sum_{i=1}^{n} u_{iz} \frac{\partial}{\partial \boldsymbol{\theta}_z} d(\mathbf{x}_i, \boldsymbol{\theta}_z) = \mathbf{0}. \tag{5.4}$$

Naturally, this derivative depends entirely on which dissimilarity measure we choose.

5.1.1 *k*-means Clustering

The k-means clustering algorithm [69] is a classical method used extensively because of its simplicity in implementation and its cost efficiency in terms of computing power [66]. It shows up as a special case of the GHAS when we use the squared Euclidean distance as our dissimilarity measure. That is,

$$d(\mathbf{x}_i, \boldsymbol{\theta}_j) = \|\mathbf{x}_i - \boldsymbol{\theta}_j\|_2^2 = (\mathbf{x}_i - \boldsymbol{\theta}_j)^T(\mathbf{x}_i - \boldsymbol{\theta}_j). \tag{5.5}$$

Differentiating Eq. (5.5) with respect to $\boldsymbol{\theta}_j$ yields

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}_j} d(\mathbf{x}_i, \boldsymbol{\theta}_j) &= \frac{\partial}{\partial \boldsymbol{\theta}_j} (\mathbf{x}_i - \boldsymbol{\theta}_j)^T(\mathbf{x}_i - \boldsymbol{\theta}_j) \\ &= -2(\mathbf{x}_i - \boldsymbol{\theta}_j). \qquad (5.6) \end{aligned}$$

Substituting Eq. (5.6) into Eq. (5.4) gives us

$$\begin{aligned} \sum_{i=1}^{n} u_{ij} \frac{\partial}{\partial \boldsymbol{\theta}_j} d(\mathbf{x}_i, \boldsymbol{\theta}_j) &= \mathbf{0} \\ \sum_{i=1}^{n} u_{ij}(-2(\mathbf{x}_i - \boldsymbol{\theta}_j)) &= \mathbf{0} \\ \sum_{i=1}^{n} u_{ij}\mathbf{x}_i &= \sum_{i=1}^{n} u_{ij}\boldsymbol{\theta}_j \\ \boldsymbol{\theta}_j &= \frac{\sum_{i=1}^{n} u_{ij}\mathbf{x}_i}{\sum_{i=1}^{n} u_{ij}}. \qquad (5.7) \end{aligned}$$

We see that the optimal $\boldsymbol{\theta}_j$ is just the sample mean vector of all the data vectors which are assigned to $\mathcal{C}_j$. Thus, the k-means algorithm is summarized as follows

1. Initialize the cluster representatives $\boldsymbol{\theta}_j,\ j = 1, 2, \ldots, k$ randomly.

2. Assign each data vector $\mathbf{x}_i,\ i = 1, 2, \ldots, n$ to the cluster with the cluster representative which is closest in terms of Euclidean distance.

3. Update the cluster representatives $\boldsymbol{\theta}_j,\ j = 1, 2, \ldots, k$ according to Eq. (5.7).

4. Check for convergence. If the algorithm has not converged, move to step 2.

We see that we need to provide the number of clusters, k. Choosing this is not necessarily trivial. This would require either some prior knowledge of the structure of the data or that we run the algorithm multiple times with different values of k. In the latter case, people familiar with the type of data might be able to interpret which of the k values makes sense based on the output partitions.

In step 1 it is common either to choose a random sample of size k from the data vectors as the initial cluster representatives or to sample k random vectors within the bounds of the data [1]. There are also several ways to check if the algorithm has converged. For instance if no cluster representatives changes when they are updated or if $\Delta J < \varepsilon$, where ε is some small constant. The cost function is guaranteed not to increase during the iterations [70], but the k-means algorithm is susceptible to converging to local minima [66].

A consequence of using the squared Euclidean distance as the dissimilarity measure is that the k-means algorithm is *linear*. It is easy to show that the set of points equidistant to two cluster representatives is constricted to a linear function. Let $\boldsymbol{\theta}_j$ and $\boldsymbol{\theta}_z$ be two cluster representatives and let $\mathbf{x}$ be a vector in feature space. The vectors $\mathbf{x}$ which are equidistant to $\boldsymbol{\theta}_j$ and $\boldsymbol{\theta}_z$ has to satisfy

$$\begin{aligned}(\boldsymbol{\theta}_j - \mathbf{x})^T(\boldsymbol{\theta}_j - \mathbf{x}) &= (\boldsymbol{\theta}_z - \mathbf{x})^T(\boldsymbol{\theta}_z - \mathbf{x}) \\ \boldsymbol{\theta}_j^T\boldsymbol{\theta}_j - 2\boldsymbol{\theta}_j^T\mathbf{x} + \mathbf{x}^T\mathbf{x} &= \boldsymbol{\theta}_z^T\boldsymbol{\theta}_z - 2\boldsymbol{\theta}_z^T\mathbf{x} + \mathbf{x}^T\mathbf{x} \\ 2(\boldsymbol{\theta}_z - \boldsymbol{\theta}_j)^T\mathbf{x} + \boldsymbol{\theta}_j^T\boldsymbol{\theta}_j - \boldsymbol{\theta}_z^T\boldsymbol{\theta}_z = 0,\end{aligned}$$

which is a linear function in $\mathbf{x}$ of the form $\mathbf{w}^T\mathbf{x}+w_0 = 0$. This implies that the k-means clustering algorithm will not be able to capture the cluster structure for nonlinearly separable data. It can also struggle with weirdly shaped clusters that are linearly separable or if the clusters have vastly different scales. This is shown in Fig. 5.1. Because of the intrinsic properties of the Euclidean distance metric, the algorithm works best for "hypersphere" shaped clusters as the set of points equidistant to a cluster representative forms a hypersphere.

Example 5.1.1 (k-means for Gaussian data)**.** Fig. 5.2 and Fig. 5.3 shows a toy dataset with three clusters of bivariate normally distributed data. In Fig. 5.2, the data has been clustered using $k = 3$. We see that k-means correctly identifies the clusters. In Fig. 5.3, the data has been clustered using $k = 4$, which is incorrect. This forces the k-means algorithm to divide one of the natural clusters into two clusters.

[1]For instance the smallest possible hypercube containing the data.

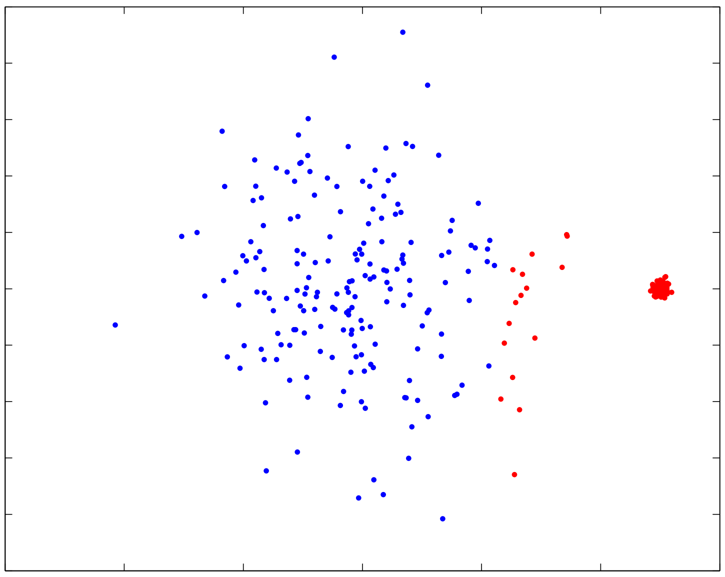


Figure 5.1: Two clusters of bivariate normal data with different scales. We see that the k-means algorithm fails to capture the natural cluster structure.

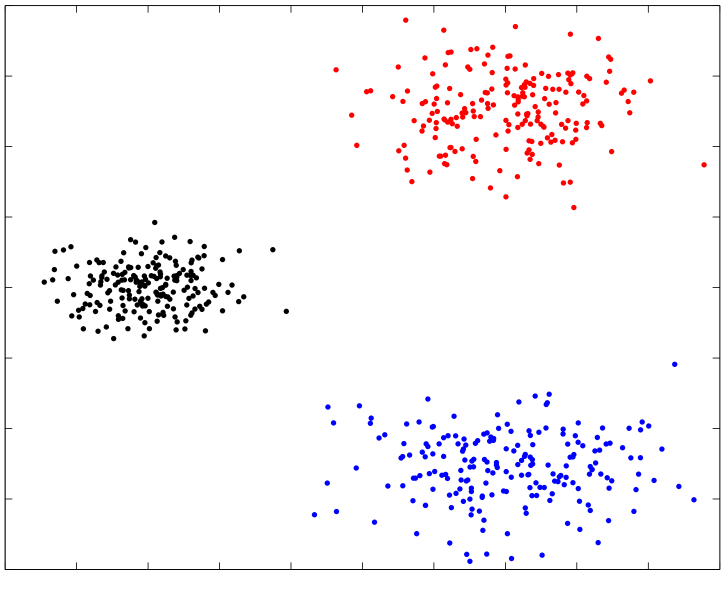


Figure 5.2: Three well separated clusters of bivariate normally distributed data. The k-means clustering algorithm correctly identifies the clusters with $k = 3$.

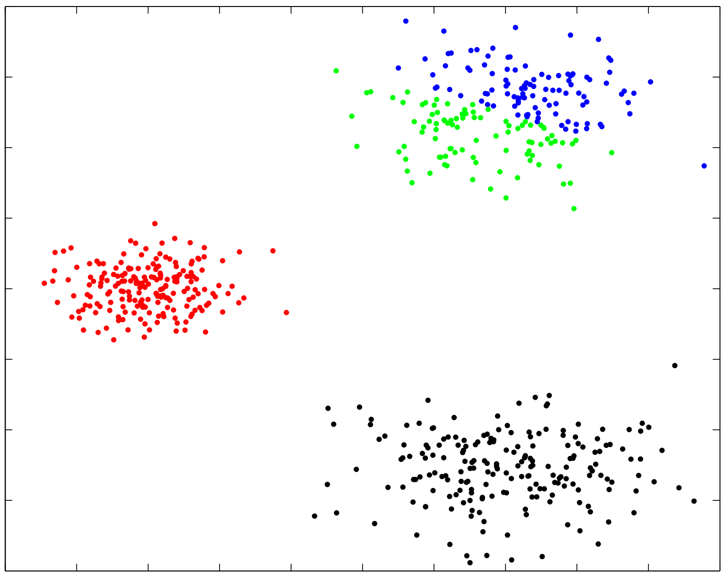


Figure 5.3: The same data as Fig. 5.2, but with $k = 4$. We see that the algorithm is forced to find four clusters, even though the data naturally consists of only three.

5.2 Clustering with Gaussian Mixture Models

This section presents a clustering algorithm which assumes a model for the data. More specifically, we assume that the Probability Density Function (PDF) for the data can be expressed as a mixture of Gaussians. To arrive at an algorithm which can be used for clustering, we need to take a detour and establish some statistical theory.

In the end of this section, it will be shown that the k-means clustering algorithm is a special case of clustering using Gaussian Mixture Models (GMM). If the clusters in the data differs significantly from hyperspherical structures, but is still compact, the GMM will capture the structure better than k-means. The GMM also considers the covariance structure of the data. This enables us to cluster data where the clusters have different scales.

5.2.1 Mixture Models

Mixture models [71] are used to estimate the PDF for some data. We assume that the density function can be expressed as a linear combination of PDF. That is, we define the PDF of the data to be

$$f_{\mathbf{X}}(\mathbf{x}) = \sum_{k=1}^{K} P_k f_k(\mathbf{x}|\boldsymbol{\theta}_k), \tag{5.8}$$

where P_k are the *mixing coefficients*, $f_k(\mathbf{x}|\boldsymbol{\theta}_k)$ are the *mixture components* and $\boldsymbol{\theta}_k$ are the parameters of component k. Note that $f_k(\mathbf{x}|\boldsymbol{\theta}_k)$ are density functions. If we require

$$0 \leq P_k \leq 1 \tag{5.9}$$

$$\sum_{k=1}^{K} P_k = 1, \tag{5.10}$$

we have

$$\begin{aligned}\int_{\mathcal{X}} f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x} &= \int_{\mathcal{X}} \sum_{k=1}^{K} P_k f_k(\mathbf{x}|\boldsymbol{\theta}_k)\,\mathrm{d}\mathbf{x} \\ &= \sum_{k=1}^{K} P_k \int_{\mathcal{X}} f_k(\mathbf{x}|\boldsymbol{\theta}_k)\,\mathrm{d}\mathbf{x} \\ &= \sum_{k=1}^{K} P_k \\ &= 1.\end{aligned}$$

Thus, $f_{\mathbf{X}}(\mathbf{x})$ is a valid PDF and P_k could be interpreted as *prior probabilities* of component k.

5.2.2 Mixture of Gaussians

A very important special case of the mixture model in Eq. (5.8) arises when we choose the Normal (Gaussian) PDF as our mixture components. That is

$$f_k(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = (2\pi)^{-\frac{p}{2}}|\boldsymbol{\Sigma}_k|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}, \quad (5.11)$$

where p is the dimensionality of the data, $\boldsymbol{\mu}_k$ is the mean vector and $\boldsymbol{\Sigma}_k$ is the covariance matrix. We note that $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ from Eq. (5.11) and P_k from Eq. (5.8) are unknown. We need to estimate these to fit the model to the data. For the GMM, this is normally done by *Maximum Likelihood Estimation* (MLE) via the *Expectation Maximization* (EM) algorithm [72]. To apply this to the GMM, we first need to establish some basic theory behind these methods.

The Maximum Likelihood Estimator

The MLE is used to estimate unknown parameters for a specific model based on the observed data. Let $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ be independent and identically distributed (IID) from $f_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\theta})$. Now, define the likelihood function as the joint density

$$\begin{aligned}L(\boldsymbol{\theta}) &= f_{\mathbf{X}_1,\mathbf{X}_2,\ldots,\mathbf{X}_N}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N|\boldsymbol{\theta}) \\ &\overset{\text{IID}}{=} \prod_{i=1}^{N} f_{\mathbf{X}}(\mathbf{x}_i|\boldsymbol{\theta}).\end{aligned} \quad (5.12)$$

Notice that if $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are observed, Eq. (5.12) is a function of the parameter $\boldsymbol{\theta}$. The likelihood function can then be maximized with respect to $\boldsymbol{\theta}$ to obtain the parameter which is most likely given the sample.

Often, it is convenient to use the *log-likelihood*

$$\begin{aligned} l(\boldsymbol{\theta}) &= \ln\left(L(\boldsymbol{\theta})\right) \\ &= \ln\left(\prod_{i=1}^{N} f_{\mathbf{X}}(\mathbf{x}_i|\boldsymbol{\theta})\right) \\ &= \sum_{i=1}^{N} \ln\left(f_{\mathbf{X}}(\mathbf{x}_i|\boldsymbol{\theta})\right) \end{aligned}$$

instead of the normal likelihood function to make computations easier. The logarithm is a monotone function, so maximizing the log-likelihood is equivalent to maximizing the likelihood. This is especially useful if the assumed distribution of the data comes from an exponential family, like the normal distribution.

The MLE is often preferred to other methods (like the method of moments [73, Ch. 7]) because it has some nice properties [73, 41, 74]. Under some conditions, we have:

- The MLE converges almost surely to the true parameter.
- The MLE is asymptotically unbiased.
- The MLE is asymptotically normally distributed with a variance given by the Cramér-Rao lower bound.
- The MLE of a function of the parameter is the function applied to the MLE of the parameter. That is for $\gamma = g(\theta)$,

$$\widehat{\gamma}_{\mathrm{ML}} = g(\widehat{\theta}_{\mathrm{ML}}).$$

It is often convenient to use the MLE to estimate parameters, although in some cases it is not possible to find an analytical solution. In these cases we need to find an estimate of the MLE by other means (like the EM algorithm).

Example 5.2.1 (Multivariate normally distributed data)**.** Let $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ be independent and identically distributed from a p-variate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

$$\mathbf{X}_i \sim \mathcal{N}_p\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$$

with the PDF

$$f_{\mathbf{X}}(\mathbf{x}_i|\boldsymbol{\mu},\boldsymbol{\Sigma}) = (2\pi)^{-\frac{p}{2}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu})\right\}.$$

To ease the computations, we will use the log-likelihood

$$\begin{aligned} l(\boldsymbol{\mu},\boldsymbol{\Sigma}) &= \sum_{i=1}^{N} \ln\left(f_{\mathbf{X}}(\mathbf{X}_i|\boldsymbol{\mu},\boldsymbol{\Sigma})\right) \\ &= \sum_{i=1}^{N} \ln\left((2\pi)^{-\frac{p}{2}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{X}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{X}_i-\boldsymbol{\mu})\right\}\right) \\ &= -\frac{Np}{2}\ln(2\pi) - \frac{N}{2}\ln\left(|\boldsymbol{\Sigma}|\right) - \frac{1}{2}\sum_{i=1}^{N}(\mathbf{X}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{X}_i-\boldsymbol{\mu}). \end{aligned}$$

By taking the partial derivative of this with respect to $\boldsymbol{\mu}$ and equating it to $\mathbf{0}$, we get

$$\begin{aligned} \boldsymbol{\Sigma}^{-1}\sum_{i=1}^{N}(\mathbf{X}_i-\boldsymbol{\mu}) &= \mathbf{0} \\ \boldsymbol{\mu} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i \\ &\Downarrow \\ \widehat{\boldsymbol{\mu}}_{\text{ML}} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i \\ &= \overline{\mathbf{X}}, \end{aligned}$$

under the assumption that $\boldsymbol{\Sigma}$ is non-singular. This is just the sample mean, which is not a surprising result. Maximizing the likelihood function with respect to $\boldsymbol{\Sigma}$ is a bit more cumbersome, but by some trace manipulations and applying a variant of the Cauchy-Schwartz inequality [41, Ch. 4], the MLE of $\boldsymbol{\Sigma}$ becomes

$$\widehat{\boldsymbol{\Sigma}}_{\text{ML}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{X}_i-\overline{\mathbf{X}}\right)\left(\mathbf{X}_i-\overline{\mathbf{X}}\right)^T.$$

Note that this is not an unbiased estimator since

$$\begin{aligned}\mathbb{E}\left(\widehat{\boldsymbol{\Sigma}}_{\mathrm{ML}}\right) &= \mathbb{E}\left(\frac{N-1}{N}\mathbf{S}\right)\\ &= \frac{N-1}{N}\mathbb{E}\left(\mathbf{S}\right)\\ &= \frac{N-1}{N}\boldsymbol{\Sigma}\\ &\neq \boldsymbol{\Sigma},\end{aligned}$$

where $\mathbf{S}$ is the sample covariance matrix which is unbiased. However, the MLE of the covariance matrix is asymptotically unbiased.

The Expectation Maximization Algorithm

The EM algorithm was originally proposed by Dempster et al. [72] as a way to solve difficult likelihood optimization problems by an iterative method which converges to the solution of the original problem. The idea is to augment the observed data $\mathbf{X} = \begin{pmatrix} X_1 & X_2 & \cdots & X_N \end{pmatrix}^T$ with some missing/hidden variables $\mathbf{Z} = \begin{pmatrix} Z_1 & Z_2 & \cdots & Z_M \end{pmatrix}^T$ and use the likelihood function of the complete dataset to estimate the parameters iteratively. Assuming IID variables $X_1, X_2, \dots, X_N$ from $f_X(x|\theta)$, the maximum likelihood estimator for θ is given by

$$\begin{aligned}\widehat{\theta}_{\mathrm{ML}} &= \arg\max_{\theta} L(\theta|\mathbf{X})\\ &= \arg\max_{\theta} \prod_{i=1}^{N} f_X(X_i|\theta),\end{aligned} \tag{5.13}$$

where both the observed values, X_i, and the parameter, θ, could be vectors. We now construct the complete data set by augmenting the observed data with the missing/hidden data. We assume the joint distribution

$$\mathbf{X}, \mathbf{Z} \sim f_{\mathbf{X},\mathbf{Z}}(\mathbf{x}, \mathbf{z}|\theta). \tag{5.14}$$

By conditioning on the observed data and assuming, we get

$$\begin{aligned}f_{\mathbf{Z}|\mathbf{X}}(\mathbf{z}|\mathbf{x}, \theta) &= \frac{f_{\mathbf{X},\mathbf{Z}}(\mathbf{x}, \mathbf{z}|\theta)}{f_{\mathbf{X}}(\mathbf{x}|\theta)}\\ f_{\mathbf{Z}|\mathbf{X}}(\mathbf{z}|\mathbf{x}, \theta) &= \frac{L^c(\theta|\mathbf{x}, \mathbf{z})}{L(\theta|\mathbf{x})},\end{aligned} \tag{5.15}$$

where $L^c(\theta|\mathbf{x}, \mathbf{z})$ denotes the likelihood function for the complete dataset and $L(\theta|\mathbf{x})$ denotes the likelihood function for the observed data. It can be shown

[72] that given the estimate $\widehat{\theta}^{(i)}$ and the function

$$\begin{aligned} Q\left(\theta, \widehat{\theta}^{(i)}\right) &= \int_{\mathcal{Z}} l^c(\theta|\mathbf{x},\mathbf{z}))f_{\mathbf{Z}|\mathbf{X}}\left(\mathbf{z}|\mathbf{x},\widehat{\theta}^{(i)}\right)\,\mathrm{d}\mathbf{z} \\ &= \mathbb{E}_{\mathbf{Z}}\left[l^c(\theta|\mathbf{x},\mathbf{z})\middle|\mathbf{X},\widehat{\theta}^{(i)}\right], \end{aligned} \tag{5.16}$$

maximizing the log-likelihood of the original data can be done by an iterative algorithm:

E-step:

Calculate $Q\left(\theta, \widehat{\theta}^{(i)}\right)$, where $\widehat{\theta}^{(i)}$ is the current estimate of the parameter.

M-step:

Maximize Q with respect to θ

$$\widehat{\theta}^{(i+1)} = \arg\max_{\theta} Q\left(\theta, \widehat{\theta}^{(i)}\right) \tag{5.17}$$

These steps are repeated for each iteration until the likelihood function converges. One thing we notice is that to kickstart the algorithm, we need an initial guess for the parameter, $\widehat{\theta}^{(0)}$. The choice of this initial guess will depend on the situation.

The likelihood function is guaranteed to be non-decreasing for each iteration and the solution is guaranteed to converge to a *stationary point* of the likelihood function [75, Ch. 5]. However, for multimodal likelihood functions this stationary point could be a local maximum.

MLE and EM for the Gaussian Mixture Model

We are now ready to formulate an algorithm to estimate maximum likelihood estimators for the GMM. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ be the observed data we want to fit the model to. Recall the mixture model in Eq. (5.8) and the mixture components in Eq. (5.11). Sampling from Eq. (5.8) is equivalent to sampling from $f_k(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with probability P_k, $k = 1, 2, \ldots, K$. To avoid confusion, we will use $\mathcal{N}_p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to denote a p-variate normal PDF with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ evaluated at $\mathbf{x}$. Now, define $\mathbf{z}_i = \begin{pmatrix} z_{i1} & z_{i2} & \cdots & z_{iK} \end{pmatrix}^T$ to be the *indicator* vector for $\mathbf{x}_i$, $i = 1, 2, \ldots, N$. That is

$$z_{ik} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is drawn from mixture component } k \\ 0 & \text{otherwise} \end{cases}. \tag{5.18}$$

We see that

$$\begin{aligned} &z_{ik} \in \{0,1\} \\ &\sum_{k=1}^{K} z_{ik} = 1. \end{aligned} \tag{5.19}$$

We will now use the indicator vectors as our hidden variables in the EM algorithm. The probability mass function (PMF) of $\mathbf{z}_i$ puts mass P_k on $\mathbf{z}_i$ if and only if $z_{ik} = 1$. Because of Eq. (5.18), we see that the PMF will be

$$f(\mathbf{z}_i) = \prod_{k=1}^{K} P_k^{z_{ik}}. \tag{5.20}$$

We also need to define the conditional distribution of a data point given the indicator vector. Recall that if $z_{ik} = 1$, the data point $\mathbf{x}_i$ is drawn from a p-variate normal distribution with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. Thus, we have

$$f(\mathbf{x}_i|\mathbf{z}_i) = \prod_{k=1}^{K} \left[\mathcal{N}_p(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]^{z_{ik}}. \tag{5.21}$$

From this, we see that the joint distribution of the indicator vector $\mathbf{z}_i$ and the data $\mathbf{x}_i$ will be

$$\begin{aligned} f(\mathbf{x}_i, \mathbf{z}_i) &= f(\mathbf{x}_i|\mathbf{z}_i) f(\mathbf{z}_i) \\ &= \prod_{k=1}^{K} \left[\mathcal{N}_p(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]^{z_{ik}} \prod_{k=1}^{K} P_k^{z_{ik}} \\ &= \prod_{k=1}^{K} \left[P_k \mathcal{N}_p(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]^{z_{ik}}. \end{aligned} \tag{5.22}$$

Using Eq. (5.13), we can then construct the log-likelihood of the complete data

$$\begin{aligned} l^c(\boldsymbol{\Theta}|\mathbf{X}, \mathbf{Z}) &= \sum_{i=1}^{N} \ln\left[f(\mathbf{x}_i, \mathbf{z}_i)\right] \\ &= \sum_{i=1}^{N} \ln\left[\prod_{k=1}^{K} \left[P_k \mathcal{N}_p(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]^{z_{ik}}\right] \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \ln\left[P_k \mathcal{N}_p(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right], \end{aligned} \tag{5.23}$$

where $\boldsymbol{\Theta}$ is the set of all the parameters, $\mathbf{X}$ is the set of all data vectors and $\mathbf{Z}$ is the set of all indicator vectors. Now, we need to find $Q\left(\boldsymbol{\Theta}, \widehat{\boldsymbol{\Theta}}^{(l)}\right)$ as defined in Eq. (5.16). This is the expectation of the complete log-likelihood with respect to $\mathbf{Z}$ given the data $\mathbf{X}$ and the current estimates of the parameters, $\widehat{\boldsymbol{\Theta}}^{(l)}$. The only stochastic variables in Eq. (5.23) are the z_{ik}'s since the data is assumed observed. Thus, we only need to find the expectation of these given the data and parameters. We have that

$$\begin{aligned}
\mathbb{E}\left(z_{ik}\middle|\mathbf{x}_i, \widehat{\boldsymbol{\Theta}}^{(l)}\right) &= 0 \cdot P\left(z_{ik} = 0\middle|\mathbf{x}_i, \widehat{\boldsymbol{\Theta}}^{(l)}\right) + 1 \cdot P\left(z_{ik} = 1\middle|\mathbf{x}_i, \widehat{\boldsymbol{\Theta}}^{(l)}\right) \\
&= P\left(z_{ik} = 1\middle|\mathbf{x}_i, \widehat{\boldsymbol{\Theta}}^{(l)}\right) \\
&= \frac{P\left(z_{ik} = 1\middle|\widehat{\boldsymbol{\Theta}}^{(l)}\right) f\left(\mathbf{x}_i\middle|z_{ik} = 1, \widehat{\boldsymbol{\Theta}}^{(l)}\right)}{\sum_{j=1}^{K} P\left(z_{ij} = 1\middle|\widehat{\boldsymbol{\Theta}}^{(l)}\right) f\left(\mathbf{x}_i\middle|z_{ij} = 1, \widehat{\boldsymbol{\Theta}}^{(l)}\right)} \\
&= \frac{\widehat{P}_k^{(l)} \mathcal{N}_p(\mathbf{x}_i|\widehat{\boldsymbol{\mu}}_k^{(l)}, \widehat{\boldsymbol{\Sigma}}_k^{(l)})}{\sum_{j=1}^{K} \widehat{P}_j^{(l)} \mathcal{N}_p(\mathbf{x}_i|\widehat{\boldsymbol{\mu}}_j^{(l)}, \widehat{\boldsymbol{\Sigma}}_j^{(l)})} \\
&= \widehat{\gamma}_{ik}^{(l)}.
\end{aligned} \tag{5.24}$$

Because of Eq. (5.18), we see that this can be interpreted as the *posterior* probability of the data point $\mathbf{x}_i$ belonging to component k. By using Eq. (5.11) and expanding the products inside the logarithm, we get

$$\begin{aligned}
Q\left(\boldsymbol{\Theta}, \widehat{\boldsymbol{\Theta}}^{(l)}\right) &= \sum_{i=1}^{N} \sum_{k=1}^{K} \widehat{\gamma}_{ik}^{(l)} \ln\left[P_k \mathcal{N}_p(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} \widehat{\gamma}_{ik}^{(l)} \Big[\ln P_k - \frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| \\
&\quad - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k)\Big].
\end{aligned} \tag{5.25}$$

From Eq. (5.17) we see that we need to maximize this function with respect to the unknown parameters. By using known formulas for matrix and vector derivatives [76], we can obtain the partial derivatives of Eq. (5.25) with respect to $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. Equating these to $\mathbf{0}$ gives us the estimates

$$\widehat{\boldsymbol{\mu}}_k^{(l)} = \frac{1}{\sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)}} \sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)} \mathbf{x}_i \tag{5.26}$$

$$\widehat{\boldsymbol{\Sigma}}_k^{(l)} = \frac{1}{\sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)}} \sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)} \left(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k^{(l)}\right) \left(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k^{(l)}\right)^T. \tag{5.27}$$

Notice that the prior probabilities P_k needs to be optimized under the constraint in Eq. (5.10). Thus, we use Lagrange optimization by constructing the Lagrange function

$$\mathcal{L}\left(\boldsymbol{\Theta}, \widehat{\boldsymbol{\Theta}}^{(l)}, \lambda\right) = Q\left(\boldsymbol{\Theta}, \widehat{\boldsymbol{\Theta}}^{(l)}\right) - \lambda\left(\sum_{k=1}^{K} P_k - 1\right).$$

Differentiating this with respect to P_k and equating it to zero yields

$$P_k = \frac{\sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)}}{\lambda}. \tag{5.28}$$

Substituting Eq. (5.28) into Eq. (5.10) and solving for λ gives us $\lambda = N$. Thus,

$$\widehat{P}_k^{(l)} = \frac{\sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)}}{N}. \tag{5.29}$$

By comparing this to the frequentist approach to probability, we see that the numerator could be interpreted as the *effective* number of data points belonging to component k.

Alg. 4 summarizes the steps needed to estimate the parameters for the GMM. The convergence criterion is either based on evaluating the log-likelihood of the observed data or by checking the parameters.

For the initialization step, it is common to first use the k-means clustering algorithm to partition the data into K klusters and using the resulting cluster representatives as the initial mean vectors. The initial covariance matrices are estimated by the sample covariance matrix of the data points assigned to each cluster. The initial mixing coefficients are estimated by counting the number of data points assigned to a cluster and dividing this by the total number of data points.

In some situations, one or several covariance matrices may become singular during the iterations. This often happens when the number of mixture components are larger than necessary (wrong model). However, there are situations where we actually want to have a lot of mixture components. To circumvent this problem, it is possible to *regularize* the covariance matrices [77]. This is done by adding a small constant to the main diagonal of the matrix, which will make sure that the covariance matrix is positive definite (and non-singular). Doing this will introduce a bias in the estimate, but this bias could be small if the size of the constant is carefully chosen. The size of this constant will depend on the data and the number of components. It should be as small as possible without making the covariance matrices singular to avoid that it affects the variances in the model.

Algorithm 4 Expectation Maximization for a Gaussian Mixture Model.

```
Input: Data points $\mathbf{x}_i,\ i = 1, 2, \ldots, N$ and the number of components $K$.
 1: Initialize $\widehat{\boldsymbol{\mu}}_k^{(0)}$, $\widehat{\boldsymbol{\Sigma}}_k^{(0)}$ and $\widehat{P}_k^{(0)}$ for $k = 1, 2, \ldots, K$.
 2: repeat
 3:     E-step. Based on the current parameter estimates and the observed
        data, calculate the posterior probabilities
            $\widehat{\gamma}_{ik}^{(l)} = \frac{\widehat{P}_k^{(l)} \mathcal{N}_p(\mathbf{x}_i | \widehat{\boldsymbol{\mu}}_k^{(l)}, \widehat{\boldsymbol{\Sigma}}_k^{(l)})}{\sum_{j=1}^{K} \widehat{P}_j^{(l)} \mathcal{N}_p(\mathbf{x}_i | \widehat{\boldsymbol{\mu}}_j^{(l)}, \widehat{\boldsymbol{\Sigma}}_j^{(l)})}$
        for $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, K$.
 4:     M-step. Adjust the parameters according to
            $\widehat{\boldsymbol{\mu}}_k^{(l+1)} = \frac{1}{\sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)}} \sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)} \mathbf{x}_i$
            $\widehat{\boldsymbol{\Sigma}}_k^{(l+1)} = \frac{1}{\sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)}} \sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)} \left(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k^{(l+1)}\right) \left(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k^{(l+1)}\right)^T$
            $\widehat{P}_k^{(l+1)} = \frac{\sum_{i=1}^{N} \widehat{\gamma}_{ik}^{(l)}}{N}.$
 5: until convergence.
Output: Estimate of parameters in the GMM.
```

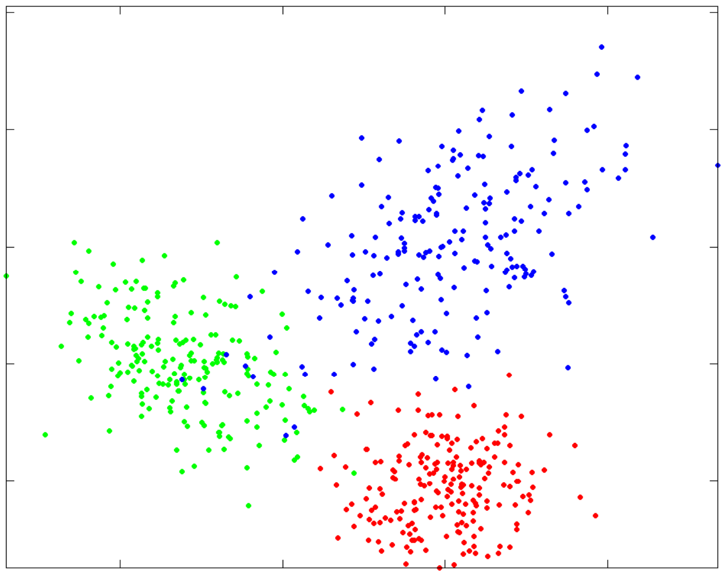


Figure 5.4: Simulated dataset from a GMM with three components.

Table 5.1: Parameters for simulated mixture of Gaussians.

Component	μ_1	μ_2	σ^2	ρ
Red	2	0	0.025	0
Green	1.2	0.5	0.05	-0.5
Blue	2	1	0.1	0.5

Example 5.2.2 (Gaussian Mixture Model for Gaussian data)**.** Fig. 5.4 shows a visualization of a dataset constructed by simulating 200 data points each from three bivariate Gaussian distributions with different parameters. The parameters are summarized in Tab. 5.1, where

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$$

$$\boldsymbol{\Sigma} = \sigma^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}.$$

Fig. 5.5 shows the data along with constant density lines of a GMM PDF fitted using the EM-algorithm which converged after 30 iterations. Comparing this to Fig. 5.6 which shows the contour lines of the theoretical PDF constructed using the parameters in Tab. 5.1, we see that the estimated PDF and the theoretical PDF are very similar.

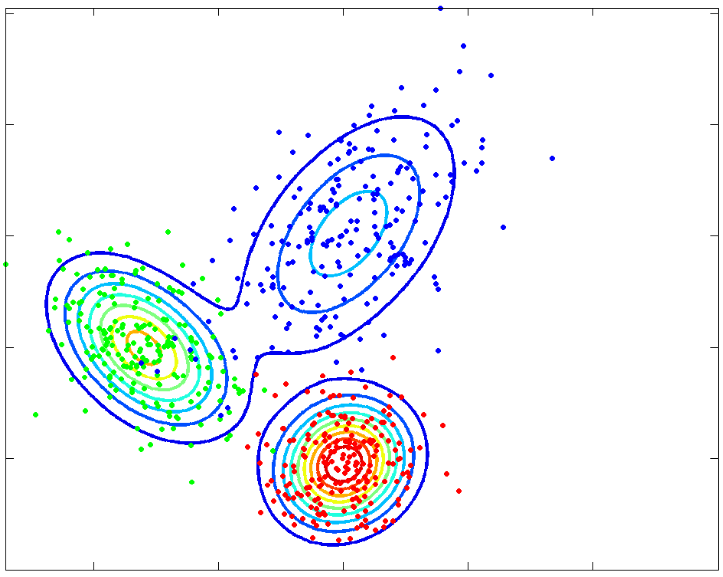


Figure 5.5: Simulated dataset from a GMM with three components. The constant density lines are from the PDF estimated using the EM-algorithm.

Clustering using GMM and EM

After fitting a GMM to the data, it is possible to use the posterior probabilities to partition the data in K clusters. Let $\boldsymbol{\gamma}_i = \begin{pmatrix}\gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{iK}\end{pmatrix}^T$, $i = 1, 2, \ldots, N$ be the posterior probabilities for data point i. We will now consider these to be the membership functions for the clustering algorithm. We see that the GMM outputs *fuzzy* membership functions. This indicates that we are able to measure the *degree* a data point belongs to a cluster. Some data points may have a strong membership to a certain cluster, while others may be more uncertain (especially when it lies in the boundary between two or more clusters). Assigning a data point to a single cluster is a matter of thresholding the membership function. That is, we set the largest value to one and the rest to zero.

Example 5.2.3 (GMM Clustering)**.** Fig. 5.7 shows the simulated dataset we used previously. The colors of the data points show the degree of membership to a certain cluster. Saturated colors indicate a strong membership, while less saturated colors are more uncertain. Fig. 5.8 shows the cluster memberships after applying a threshold to the membership functions. By comparing this to Fig. 5.4, we see that there are some data points that are assigned to the wrong cluster. This is because we have overlapping data, which cannot be clustered 100% correctly.

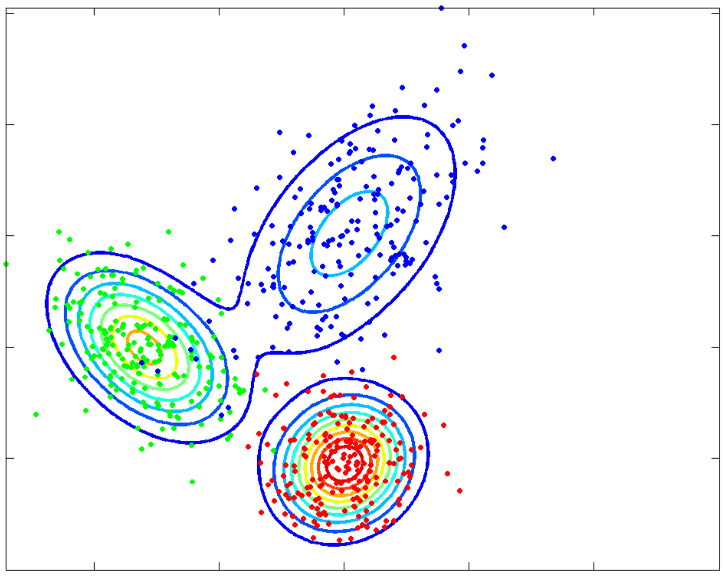


Figure 5.6: Simulated dataset from a GMM with three components. The constant density lines are from the theoretical PDF constructed using the parameters in Tab. 5.1.

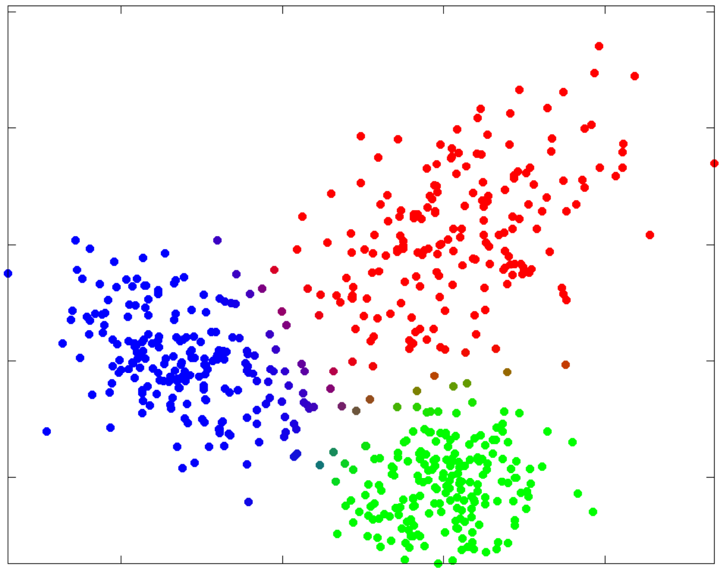


Figure 5.7: Cluster membership of a dataset from a GMM with three components. Saturated colors indicate a strong membership to the cluster.

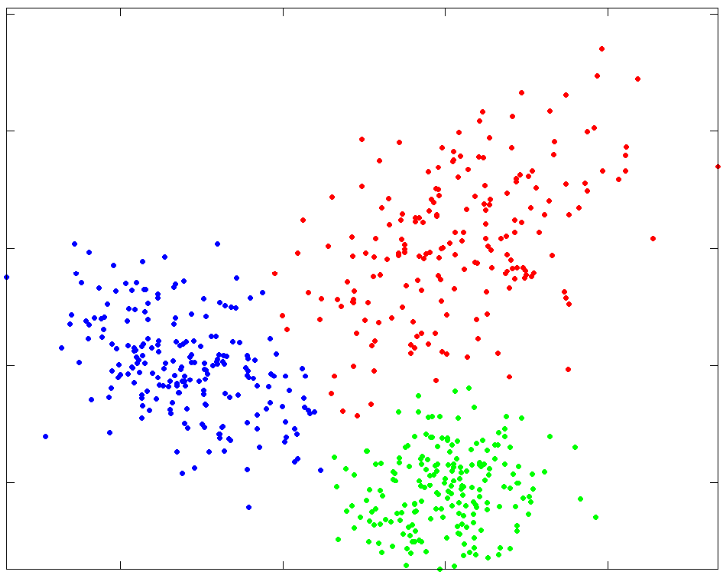


Figure 5.8: Cluster membership of a dataset from a GMM with three components after thresholding the fuzzy membership functions.

Relation to k-means

The GMM clustering algorithm is closely related to the k-means algorithm. More specifically, k-means arises as a special case of the GMM. Now, assume that the covariance matrices in the mixture components are equal and diagonal. That is,

$$\mathbf{\Sigma}_k = \varepsilon \mathbf{I},\ k = 1, 2, \dots, K, \tag{5.30}$$

where $\mathbf{I}$ is the identity matrix and $\varepsilon > 0$. Inserting Eq. (5.30) into Eq. (5.24) gives us the posterior probabilities

$$\gamma_{ik} = \frac{P_k \exp\left\{-\frac{1}{2\varepsilon}\|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2\right\}}{\sum_{j=1}^{K} P_j \exp\left\{-\frac{1}{2\varepsilon}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2\right\}}, \tag{5.31}$$

where the estimate- and iteration step notation has been omitted for readability. Let $\boldsymbol{\mu}_{j'}$ denote the mean vector which is closest to $\mathbf{x}_i$. If we consider the limit $\varepsilon \to 0$, we see that $\gamma_{ik} \to 0$ for all k such that $\boldsymbol{\mu}_k \neq \boldsymbol{\mu}_{j'}$. Otherwise, $\gamma_{ik} \to 1$. This reduces to the same hard membership functions as the k-means algorithm outputs. By inserting these limits in Eq. (5.26), we see that the mean vectors will be the same as the mean vectors in the k-means algorithm.

5.3 Spectral Clustering

Spectral clustering refers to a family of clustering methods which exploits the eigenvalues and eigenvectors (the spectrum) of a similarity matrix to partition the data. In recent years, these methods have become increasingly popular since they often outperform other simple clustering algorithms like the k-means algorithm. While the algorithms are often easy to implement, the theory is a bit involved.

In this section, we will establish some of the theory behind the spectral methods. First from a graph cut point of view, where we construct a similarity graph and try to partition the data based on this graph. The graph cut view is closely related to Markov Chain random walks [78]. The second part is about the connection to kernel methods. For a more in-depth introduction to spectral clustering, the interested reader could take a look at [68].

5.3.1 Graph Cut

The goal of the clustering methods is to partition a dataset in K clusters, where the datapoints in the same cluster are *similar*, while datapoints from different clusters are *dissimilar.* A nice way of representing similarity between data points is a similarity graph.

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ be a set of data points we want to partition into K clusters and let $s_{ij} > 0$ be some similarity measure between $\mathbf{x}_i$ and $\mathbf{x}_j$. The similarity graph $G = (V, E)$ consists of vertices and edges. Each vertex v_i represents a data point $\mathbf{x}_i$. Each edge is weighted by w_{ij}. The weight, $w_{ij} = s_{ij}$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected in the graph and $w_{ij} = 0$ otherwise. The weights of the graph can be stored in the *weight matrix* $\mathbf{W} = \{w_{ij}\}_{i,j=1,2,\ldots,N}$. For an undirected graph, we have $w_{ij} = w_{ji}$, so $\mathbf{W}$ is symmetric.

There are three types of similarity graphs that are commonly used:

1. *ε-neighborhood graph.* In a ε-neighborhood graph, we connect the vertices v_i and v_j with an edge if $\|\mathbf{x}_i - \mathbf{x}_j\| < \varepsilon$, $\varepsilon > 0$. That is, if the Euclidean distance between the data points are lower than some threshold, they are connected in the graph.

2. *k-nearest neighbor (knn) graph.* A k-nearest neighbor graph is constructed by considering the k-nearest neighbors of a data point $\mathbf{x}_i$. Notice that if we connect vertex v_i to v_j by an edge based solely on the k-nearest neighbors of $\mathbf{x}_i$, the graph will become *directed.* To make the graph undirected, we connect v_i to v_j if $\mathbf{x}_j$ is among the k-nearest neighbors of $\mathbf{x}_i$ *or* $\mathbf{x}_i$ is among the k-nearest neighbors of $\mathbf{x}_j$. Fig. 5.9 shows a dataset. The two red points are selected to illustrate a k-nearest neighbor graph. The three nearest neighbors of the points lies within the circle centered at the point. We see that the bottom point is among the three nearest neighbors of the top point. The opposite, however, is not true.

3. *Fully connected graph.* In a fully connected graph, all points are connected to each other. This is a useful representation if the similarity measure models local neighborhoods.

Now that we know how to construct a graph, we can move on to defining the *Graph Laplacian* and the *Normalized Cut.* This will lead us to the spectral clustering algorithms.

The Graph Laplacian

Given the *degree matrix,* $\mathbf{D} = \operatorname{diag}(d_i)_{i=1,2,\ldots,N}$, where $d_i = \sum_{j=1}^{N} w_{ij}$. The graph Laplacian is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{5.32}$$

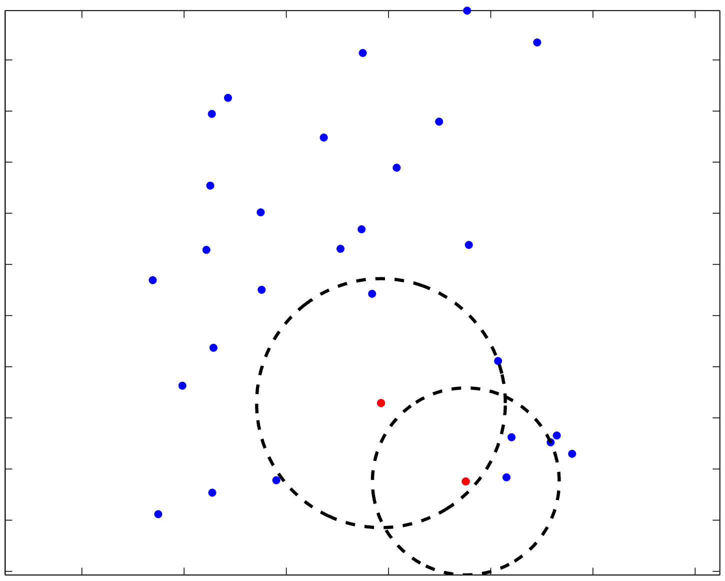


Figure 5.9: Illustration of constructing a k-NN graph. The three nearest neighbors of the two red points lies within the circles.

The graph Laplacian has some nice properties. For an undirected graph, consider the quadratic form

$$
\begin{aligned}
\mathbf{y}^T\mathbf{L}\mathbf{y} &= \mathbf{y}^T(\mathbf{D}-\mathbf{W})\mathbf{y} \\
&= \mathbf{y}^T\mathbf{D}\mathbf{y} - \mathbf{y}^T\mathbf{W}\mathbf{y} \\
&= \sum_{i=1}^{N} y_i^2 d_i - \sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j w_{ij} \\
&= \frac{1}{2}\left(\sum_{i=1}^{N} y_i^2 d_i - 2\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j w_{ij} + \sum_{j=1}^{N} y_j^2 d_j\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{N} y_i^2 \sum_{j=1}^{N} w_{ij} - \sum_{i=1}^{N}\sum_{j=1}^{N} 2 y_i y_j w_{ij} + \sum_{j=1}^{N} y_j^2 \sum_{i=1}^{N} w_{ji}\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} y_i^2 w_{ij} - \sum_{i=1}^{N}\sum_{j=1}^{N} 2 y_i y_j w_{ij} + \sum_{j=1}^{N}\sum_{i=1}^{N} y_j^2 w_{ji}\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} y_i^2 w_{ij} - \sum_{i=1}^{N}\sum_{j=1}^{N} 2 y_i y_j w_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{N} y_j^2 w_{ij}\right) \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \left(y_i^2 - 2 y_i y_j + y_j^2\right) w_{ij} \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \left(y_i - y_j\right)^2 w_{ij}.
\end{aligned}
\tag{5.33}
$$

Since $w_{ij} \geq 0$, we see that $\mathbf{y}^T\mathbf{L}\mathbf{y} \geq 0 \ \forall \ \mathbf{y} \in \mathbb{R}^N$. Therefore, the graph Laplacian is positive semi-definite. From this, it follows that the eigenvalues of $\mathbf{L}$ are nonnegative. It is also easy to show that any constant vector $c \cdot \mathbb{1}$ lies in the null space of $\mathbf{L}$. So the smallest eigenvalue of $\mathbf{L}$ is 0 with the associated eigenvector $c \cdot \mathbb{1}$.

The Normalized Cut

Clustering on similarity graphs is often done by minimizing *graph cuts*. That is, we want find a partition of the graph such that the between-cluster weights are as low as possible while the within-cluster weights are high. That means that vertices in different clusters are dissimilar, while vertices within the same cluster are similar. We will consider the situation where we want to partition the dataset into two clusters and then state a generalized algorithm.

For two complementary sets A and B, define the graph cut

$$\mathrm{cut}(A,B) = \sum_{i \in A} \sum_{j \in B} w_{ij}. \tag{5.34}$$

The graph cut is the sum of the weights from every vertex in A to every vertex in B. It makes sense intuitively to try to minimize this. If the graph cut is low, the between-cluster similarity low. However, minimizing the graph cut will often lead to a partition with one single vertex in one cluster and all the other vertices in the other cluster. To compensate for this, we will define the *volume* of the set A:

$$\mathrm{vol}(A) = \sum_{i \in A} d_i. \tag{5.35}$$

If A consists of a single vertex, the volume will be the sum of all the weights out of that vertex which is small compared to a partition where A contains more vertices. This leads us to the *normalized cut*:

$$\mathrm{Ncut}(A,B) = \mathrm{cut}(A,B) \left[\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)} \right]. \tag{5.36}$$

We see that the normalized cut will be minimized if $\mathrm{cut}(A,B)$ is low while at the same time $\mathrm{vol}(A)$ and $\mathrm{vol}(B)$ are high. This ensures that the between-cluster similarity is low and that the within-cluster similarity is high.

Fig. 5.10 shows an illustration of a fully connected graph. The circles are vertices in the graph and the black lines represents the edges. The width of the lines corresponds to the weight of the edge. Using the normalized cut, the edges crossing the green line are removed.

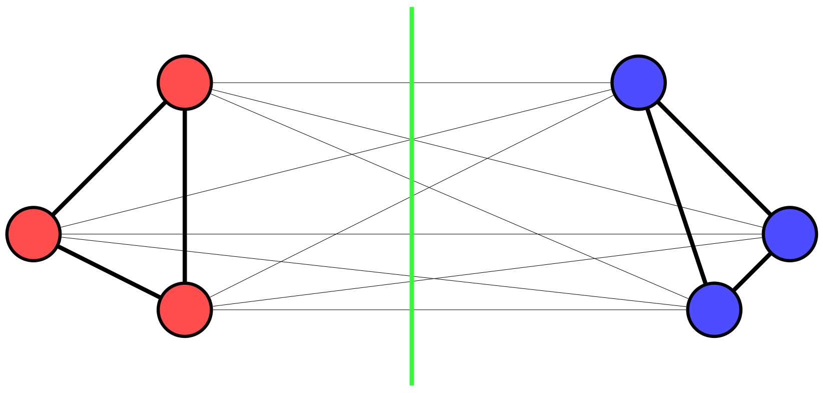


Figure 5.10: An illustration of the normalized cut.

An approximation to the Normalized Cut

Minimizing the normalized cut is not trivial. However, we are able to approximate it by using the graph Laplacian. Now, consider the label vector $\mathbf{y} = \left(y_1, \quad y_2, \quad \dots, \quad y_N\right)^T$, where

$$y_i = \begin{cases} \frac{1}{\mathrm{vol}(A)} & \text{if } \mathbf{x}_i \in A \\ -\frac{1}{\mathrm{vol}(B)} & \text{if } \mathbf{x}_i \in B \end{cases}. \tag{5.37}$$

If $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned to the same cluster, we have $y_i - y_j = 0$. From this, it follows that

$$\begin{aligned}\mathbf{y}^T\mathbf{L}\mathbf{y} &= \frac{1}{2}\sum_{i\in A}\sum_{j\in B}\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)^2 w_{ij} \\ &= \frac{1}{2}\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)^2 \sum_{i\in A}\sum_{j\in B} w_{ij} \\ &= \frac{1}{2}\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)^2 \mathrm{cut}(A, B).\end{aligned}$$

We also have

$$\begin{aligned}\mathbf{y}^T\mathbf{D}\mathbf{y} &= \sum_{i\in A} y_i^2 d_i + \sum_{j\in B} y_j^2 d_j \\ &= \sum_{i\in A}\frac{1}{\mathrm{vol}^2(A)} d_i + \sum_{j\in B}\frac{1}{\mathrm{vol}^2(B)} d_j \\ &= \frac{1}{\mathrm{vol}^2(A)}\sum_{i\in A} d_i + \frac{1}{\mathrm{vol}^2(B)}\sum_{j\in B} d_j \\ &= \frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}.\end{aligned}$$

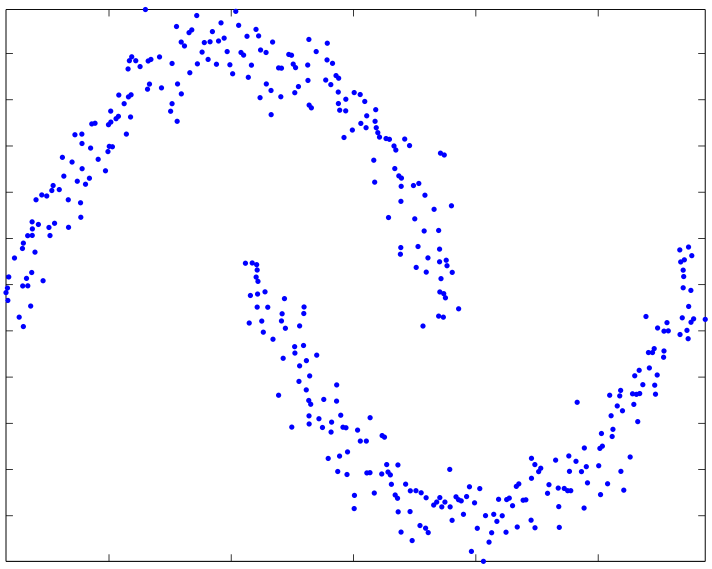


Figure 5.11: Toy data with nonlinearly separable clusters. Each cluster consists of 200 data points.

So

$$\frac{\mathbf{y}^T\mathbf{L}\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}} = \frac{1}{2}\frac{\left(\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}\right)^2 \mathrm{cut}(A,B)}{\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}}$$
$$\propto \mathrm{Ncut}(A,B).$$

So minimizing the normalized cut can be done by

$$\min_{\mathbf{y}} \frac{\mathbf{y}^T\mathbf{L}\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}. \tag{5.38}$$

By relaxing the constraints on $\mathbf{y}$ in Eq. (5.37) so that we allow any value, it is possible to show that an approximate minimization of the normalized cut can be obtained by the generalized eigenvalue problem

$$\mathbf{L}\mathbf{y} = \lambda\mathbf{D}\mathbf{y}. \tag{5.39}$$

Here, λ is the *second* smallest eigenvalue since the smallest is the trivial solution where $\lambda = 0$. The idea is to compute $\mathbf{y}$ as an eigenvector of $\mathbf{D}^{-1}\mathbf{L}$ and then threshold it to get the cluster labels. Note that $\mathbf{D}$ is a diagonal matrix, which is easy to invert as long as all the diagonal elements are nonzero.

Example 5.3.1 (Crescent moon)**.** Fig. 5.11 shows a typical two-cluster toy dataset with nonlinearly separable clusters. Each cluster contains 200 data

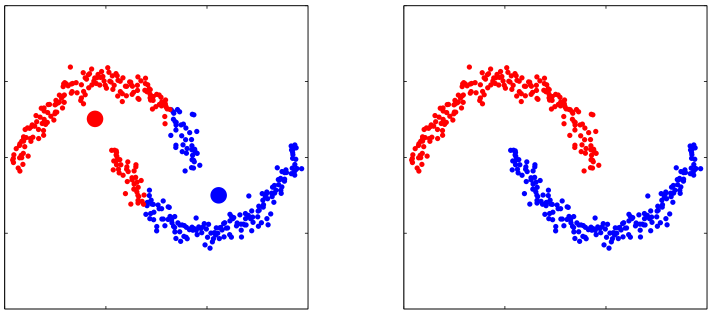


Figure 5.12: Clustering solution of the toy data. The k-means solution (left) fails to correctly cluster the data, while the normalized cut approximation (right) succeeds. The large circles in the k-means plot are the cluster representatives for each of the two clusters.

points. Other algorithms, like k-means, fail to partition the data correctly as seen in Fig. 5.12 along with the solution of the normalized cut approximation which is clustered correctly. The similarity graph was constructed using a fully connected graph with the Gaussian kernel

$$s_{ij} = s(\mathbf{x}_i, \mathbf{x}_j) = \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right\} \tag{5.40}$$

as the similarity measure. The parameter σ controls the width of the kernel and was set to $\sigma = 0.1$. Fig. 5.13 shows a plot of the second smallest eigenvector, which is the approximate solution which minimizes the normalized cut. We see that we can clearly apply a threshold to this eigenvector to generate labels for our data.

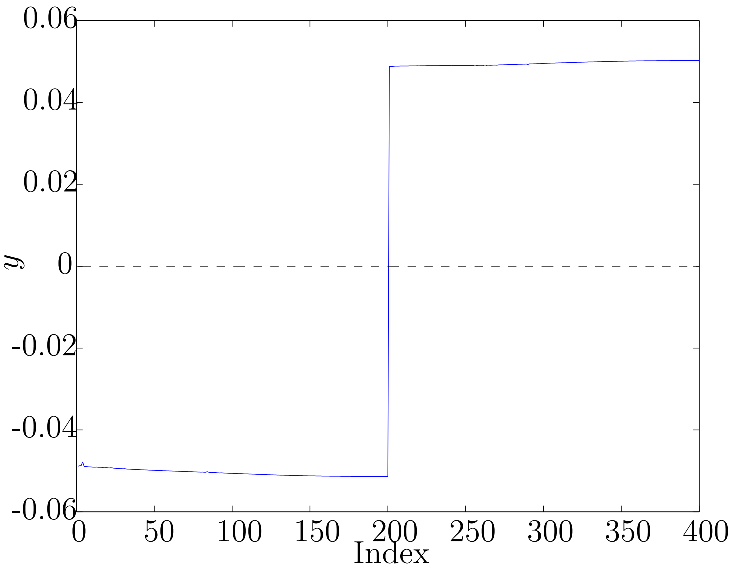


Figure 5.13: Plot of the second smallest eigenvector of $\mathbf{D}^{-1}\mathbf{L}$. We see that the labels can be obtained by thresholding this vector as the values are clearly separable with a straight line indicated by the dashed line.

General algorithms

In the previous discussion, we assumed that the data could be partitioned into two clusters. However, the theory is extendable to multiple clusters. Now, define the *normalized* graph Laplacians

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} \tag{5.41}$$

$$\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L}. \tag{5.42}$$

The first normalized graph Laplacian is a symmetric matrix, while the second one is related to random walks. Hence the subscripts. Recall the generalized eigenvalue problem

$$\mathbf{L}\mathbf{y} = \lambda\mathbf{D}\mathbf{y}. \tag{5.43}$$

By left multiplying Eq. (5.43) with $\mathbf{D}^{-1}$, we get

$$\begin{aligned} \mathbf{D}^{-1}\mathbf{L}\mathbf{y} &= \lambda\mathbf{y} \\ \mathbf{L}_{\text{rw}}\mathbf{y} &= \lambda\mathbf{y}. \end{aligned} \tag{5.44}$$

Thus, the solution of the generalized eigenvalue problem in Eq. (5.43) is a solution of the regular eigenvalue problem using $\mathbf{L}_{\text{rw}}$. Now, let

$$\mathbf{y} = \mathbf{D}^{-\frac{1}{2}}\mathbf{z}. \tag{5.45}$$

Algorithm 5 Spectral clustering using $\mathbf{L}_{\text{sym}}$

```
Input: Similarity matrix $\mathbf{S}$ and the number of clusters $k$.
1: Construct a similarity graph with the weight matrix $\mathbf{W}$.
2: Compute the symmetric normalized graph Laplacian $\mathbf{L}_{\text{sym}}$.
3: Compute the first $k$ eigenvectors of $\mathbf{L}_{\text{sym}}$ corresponding to the $k$ smallest eigenvalues and form the matrix $\mathbf{U}$ containing the eigenvectors as columns.
4: Normalize the rows of $\mathbf{U}$ to unit length.
5: Let $\mathbf{y}_i,\ i = 1, 2, \dots, N$ be the rows of the row-normalized matrix $\mathbf{U}$.
6: Cluster the data points $\mathbf{y}_i$ into $k$ clusters using the $k$-means algorithm.
Output: Cluster membership functions
```

Substituting Eq. (5.45) into Eq. (5.43) yields

$$\begin{aligned}
\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{z} &= \lambda\mathbf{D}\mathbf{D}^{-\frac{1}{2}}\mathbf{z} \\
\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{z} &= \lambda\mathbf{D}^{\frac{1}{2}}\mathbf{z} \\
\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{z} &= \lambda\mathbf{z} \\
\mathbf{L}_{\text{sym}}\mathbf{z} &= \lambda\mathbf{z}.
\end{aligned} \tag{5.46}$$

We see that if $\mathbf{y}$ is an eigenvector of $\mathbf{L}_{\text{rw}}$ with the corresponding eigenvalue λ, then $\mathbf{z}$ is an eigenvector of $\mathbf{L}_{\text{sym}}$ with the corresponding eigenvalue λ. We also have that $\mathbf{y} = \mathbf{D}^{-\frac{1}{2}}\mathbf{z}$. An algorithm using $\mathbf{L}_{\text{sym}}$ is stated in Alg. 5 [79]. Alg. 6 [16] uses $\mathbf{L}_{\text{rw}}$. von Luxburg [68] suggests that using Alg. 6 is better than using Alg. 5 because multiplying the eigenvectors with $\mathbf{D}^{-\frac{1}{2}}$ could lead to undesired artifacts and the row normalization in Alg. 5 could reduce the discriminating properties of the eigenvectors. The justification the row-normalization is beyond the scope of this thesis. Two different views of this can be found in [79] and [68].

A connection to Laplacian Eigenmaps

Laplacian Eigenmaps [80] is a dimensionality reduction method which assumes that the data lies within a lower dimensional manifold in a higher dimensional space. It attempts to preserve the relationship between data points when mapping the data to a lower-dimensional space. More specifically, if two data points are close in the high dimensional space, they should also be close in the lower-dimensional space.

Let $G = (V, E)$ be an undirected graph with the data points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ as its vertices. The edges are weighted by w_{ij}. Now, consider a mapping to

Algorithm 6 Spectral clustering using $\mathbf{L}_{\text{rw}}$

```
Input: Similarity matrix $\mathbf{S}$ and the number of clusters $k$.
1: Construct a similarity graph with the weight matrix $\mathbf{W}$.
2: Compute the normalized graph Laplacian $\mathbf{L}_{\text{rw}}$.
3: Compute the first $k$ eigenvectors of $\mathbf{L}_{\text{rw}}$ corresponding to the $k$ smallest eigenvalues and form the matrix $\mathbf{U}$ containing the eigenvectors as columns.
4: Let $\mathbf{y}_i,\ i = 1, 2, \ldots, N$ be the rows of the matrix $\mathbf{U}$.
5: Cluster the data points $\mathbf{y}_i$ into $k$ clusters using the $k$-means algorithm.
Output: Cluster membership functions
```

a one dimensional space. An intuitive cost function which preserves the relationship between data points is

$$J(\mathbf{y}, \mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} (y_i - y_j)^2 w_{ij}, \tag{5.47}$$

where $y_k,\ k = 1, 2, \ldots, N$ is the one-dimensional mapping of the high dimensional data point $\mathbf{x}_k$. Since the weight w_{ij} is a similarity measure in the high dimensional space, this is large if the data points are similar and small if they are dissimilar. Looking at the cost function, we see that if w_{ij} is large (similar data points) then $y_i - y_j$ must be small to minimize the cost function. Similarly, if w_{ij} is small (dissimilar data points), then $y_i - y_j$ could be large without affecting the value of the cost function. From Eq. (5.33), we see that

$$J(\mathbf{y}, \mathbf{W}) \propto \mathbf{y}^T \mathbf{L} \mathbf{y},$$

which we know is minimized by the second lowest eigenvector of generalized eigenvalue problem in Eq. (5.43). For an ℓ-dimensional mapping, the cost function is modified to

$$J(\mathbf{Y}, \mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \|\mathbf{y}_i - \mathbf{y}_j\|^2 w_{ij} = \text{Trace}(\mathbf{Y}^T \mathbf{L} \mathbf{Y}),$$

where

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1^T \\ \mathbf{y}_2^T \\ \vdots \\ \mathbf{y}_N^T \end{pmatrix}$$

and $\mathbf{y}_i \in \mathbb{R}^\ell,\ i = 1, 2, \ldots, N$. This is minimized by the same generalized eigenvalue problem, but using ℓ eigenvectors.

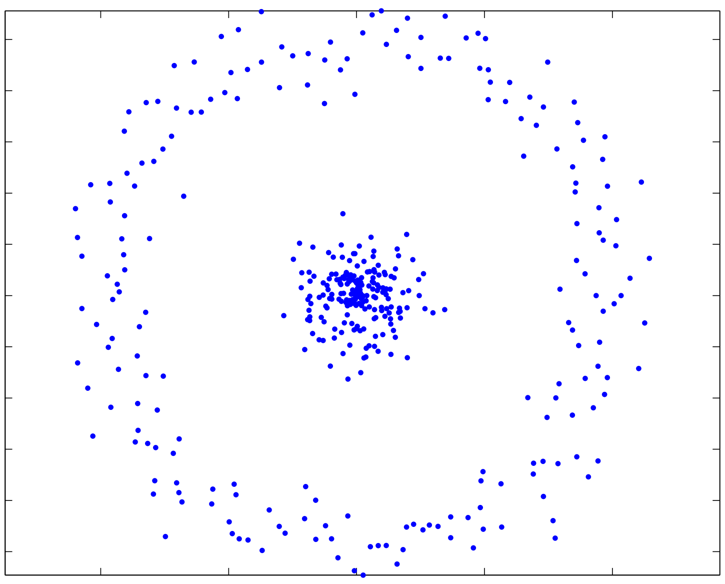


Figure 5.14: Circle data.

We see a close connection between Alg. 6 and Laplacian Eigenmaps. The algorithm uses Laplacian Eigenmaps for feature extraction and performs k-means clustering on this new representation.

5.3.2 KPCA Spectral Clustering

The idea of Alg. 6, where we use Laplacian Eigenmaps for feature extraction and cluster this new representation can be extended to include kernel methods. As seen in Sec. 4.2.2, a nonlinear transformation could make nonlinearly separable data linearly separable. This can be exploited by performing KPCA on the data using the in-sample version in Eq. (4.25) and cluster this new representation of the data with a simple clustering algorithm like k-means. Since KPCA exploits the eigenvectors and eigenvalues of a kernel matrix, this is a spectral method. A connection between the Laplacian Eigenmap embedding and the kernel PCA projections is provided in [81, 82].

Example 5.3.2 (Circle data)**.** Fig. 5.14 shows the familiar circle data with 200 data points in each of the two circles. A kernel matrix was constructed using a Gaussian kernel with $\sigma = 5$ on the data. The data was projected to the $\ell = 2$ principal components in kernel space using KPCA and canonical k-means was applied to this representation of the data using $k = 2$. The results of this is shown in Fig. 5.15a and Fig. 5.15b. The algorithm discovers the two clusters perfectly. Note that the exact same matrix could be used as a weight matrix in for instance Alg. 6. An ε-neighborhood graph was constructed using the same kernel matrix. The Laplacian eigenmap representation of the data in three dimensions is shown in Fig. 5.15c, while the KPCA projection in three dimensions is shown in Fig. 5.15d. We see that these two representations of the data are very similar. However, for this specific dataset, the Laplacian representation seems to be more compact.

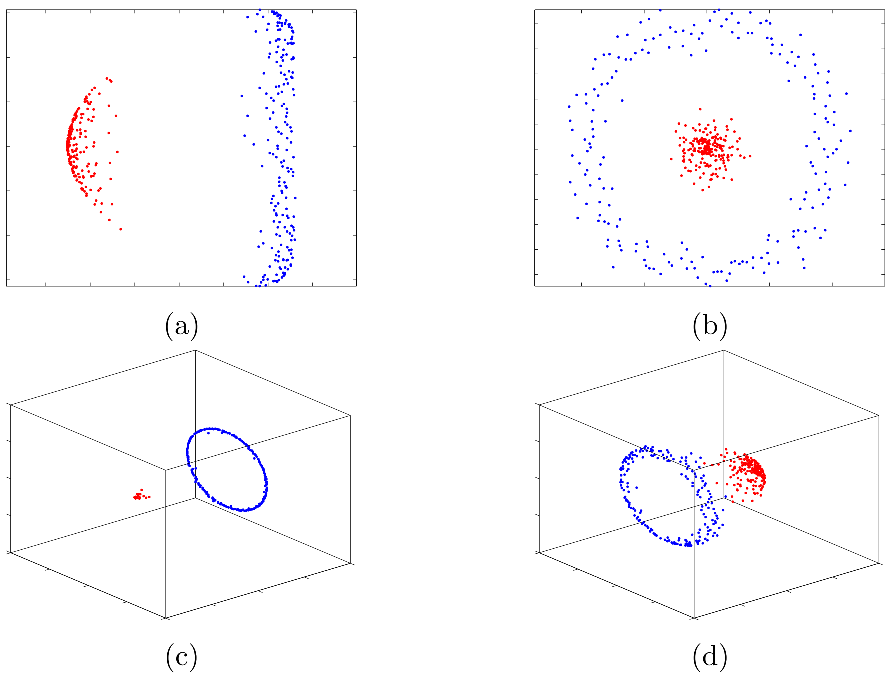


Figure 5.15: (a): PCA projection in kernel space with labels assigned by the k-means algorithm. (b): Original representation of the data. The data points are colored according to the labels in (a). (c): 3-dimensional Laplacian eigenmap of the circle data. (d): 3-dimensional PCA projection in kernel space.

5.4 Consensus Clustering

Consensus clustering (also called ensemble clustering) is a relatively new methodology which has emerged over the last decade or so. Even though the amount of clustering algorithms is vast, there are no clustering algorithms which will be appropriate to use for every dataset and different algorithms might produce different partitions for the same dataset. Even when applying one clustering algorithm several times to the same dataset with different initial conditions, ambiguous results might arise when we compare the outputs of the different trials. This might make the interpretation of the clustering results a challenge. The idea of consensus clustering is to combine the results of several clustering trials to obtain a better partition than each individual trial. This is often done by constructing a similarity matrix called a *consensus-*, *co-association-* or *ensemble* matrix. Different authors use different names for the same type of matrix. In this thesis, the term consensus matrix will be used. The theory in this section is considered as a basis for the discussion in Sec. 6.2, where we will provide a connection between the consensus matrix and *cluster kernels*.

There are several proposed algorithms to combine clustering results. Fred and Jain [83, 84] suggests using the k-means clustering algorithm several times with random initial conditions. In each of the clustering trials, the number of clusters, k, is either fixed or chosen randomly in the range $k \in [k_{\min}, k_{\max}]$. The resulting partitions are then used to *vote* in a sense. A $N \times N$ consensus matrix $\mathbf{S} = \{s_{ij}\}_{N\times N}$ is constructed by counting the number of times the points $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned to the same cluster in the M different partitions. Each time these data points are clustered together, it counts as one *vote*. They call this voting process *evidence accumulation*. The elements of $\mathbf{S}$ are then calculated by

$$s_{ij} = \frac{n_{ij}}{M},$$

where n_{ij} is the number of times $\mathbf{x}_i$ and $\mathbf{x}_j$ has been assigned to the same cluster. Since $n_{ij} = n_{ji}$, we see that $\mathbf{S}$ will be symmetrical.

In the ideal case, we should have

$$s_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ belong to the same cluster} \\ 0 & \text{otherwise} \end{cases}.$$

This happens when $\mathbf{x}_i$ and $\mathbf{x}_j$ are clustered together in all of the k-means trials. We see that if the data points are ordered according to their final cluster assignment, the consensus matrix will be a *block diagonal* matrix. That is, there exists a permutation matrix $\mathbf{Q}$ such that

$$\mathbf{C} = \mathbf{Q}^T \mathbf{S} \mathbf{Q},$$

where **C** is a block diagonal matrix. In this case, detecting the cluster structure of the data is trivial. However, in real applications there will be some non-zero matrix $\boldsymbol{\varepsilon}$ in the off-diagonal elements of this matrix and thus, **C** is *block diagonally dominant*.

The consensus matrix is a type of similarity matrix. If two data points are clustered together in many of the different clustering solutions, they are considered more similar than two data points that are not clustered together as often. This similarity matrix is used to obtain a final partition. Fred and Jain [84] suggests a hierarchical algorithm like the single link or average link for this purpose.

In [84], an optimality criteria based on information theory is defined[2]. This is used to choose the number of clusters in the final result. The theory behind this is a bit involved and will not be presented in this thesis. However, empirical results in the original paper suggests that the same number of clusters is chosen when using the longest lifetime based on the dendrogram of the hierarchical clustering algorithm.

While this is the basic idea, other similar approaches has been devised. Monti et al. [86] includes resampling techniques (like Bootstrapping [87]) to simulate a perturbation of the original dataset. Strehl and Ghosh [85] use (amongst other things) hypergraph partitioning to obtain a clustering solution. Hore et al. [88, 89] use centroid based consensus clustering to reduce memory complexity for large datasets. Nascimento et al. [90] use spectral clustering theory. Meyer and Wessell [91] employ a stochastic (random walk) approach.

[2]The same criterion was also proposed by Strehl and Ghosh [85].

Part II

Joint ranking and clustering based on Markov Chain transition probabilities learned from data

Chapter 6

The Probabilistic Cluster Kernel

The Probabilistic Cluster Kernel (PCK) is a new and exciting data generated kernel function. The goal of this kernel function is to distance ourselves from strong parameter dependence that other kernel functions often suffer severely from. For instance the Gaussian kernel, which is dependent on a width parameter σ. The thought behind the PCK is to break free from this strong parameter dependence by learning the kernel function from the data.

The idea behind cluster kernels in general can be traced back to Weston et al. [92] where they used hard cluster membership functions to calculate a weight function in semi supervised learning on protein data. The cluster kernel is calculated by clustering the data several times with different number of clusters and count the number of times two data points are clustered together. It is easy to show that this counting process can be expressed as a sum of inner products between the hard cluster membership functions. Thus, a matrix with these values as its elements is a valid kernel matrix. This is used in a Support Vector Machine as a weighting function together with a domain specific kernel.

This idea was adopted by Tuia and Camps-Valls [93] for semi supervised learning in remote sensing. The cluster kernel was further developed to include the use of fuzzy membership functions in [94] using Gaussian Mixture Models and the EM algorithm. Recently, it has been used for spectral clustering in [26], showing promising results.

This chapter starts by defining the PCK mathematically. A motivational example is then presented. The chapter is concluded with a new connection between the PCK and the consensus clustering methodology.

6.1 Definition

The PCK is calculated by averaging inner products between posterior distributions found by fitting Gaussian Mixture Models (GMM) to the data with different initial conditions and different number of mixture components. The intuition is simple. If two data points are considered similar if they have a high probability of being drawn from the same mixture components.

Let G be the maximum number of clusters and let Q be the number of initial conditions. The PCK is generated by fitting GMMs with $g = 2, 3, \ldots, G$ mixture components, Q times using GMM-EM. This outputs the posterior distributions $\boldsymbol{\gamma}_i(q, g) \in \mathbb{R}^g$ of data point $i = 1, 2, \ldots, N$ with the initial condition $q = 1, 2, \ldots, Q$ and $g = 2, 3, \ldots, G$ mixture components. The PCK is then defined as

$$K_{\text{PCK}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{Z} \sum_{q=1}^{Q} \sum_{g=2}^{G} \boldsymbol{\gamma}_i(q, g)^T \boldsymbol{\gamma}_j(q, g), \tag{6.1}$$

where Z is a normalizing constant.

We should note that the PCK is able to capture both local- and global scale similarities if the number of mixture components used is sufficiently high. Imagine that we have a large number of mixture components. Each mixture component will then attempt to estimate the density of a small region in the input space. On the other hand, if the number of mixture components is small, each of the mixture components will estimate the density of a large region. Thus, a large number of mixture components will capture similarities on a local scale, while a small number of mixture components will capture similarities on a global scale. An illustration of this concept is shown in Fig. 6.1.

Izquierdo-Verdiguier et al. [26] shows that the matrix formed by the PCK, $\mathbf{K}_{\text{PCK}}$, is a valid Mercer kernel and provide the actual mapping function Φ by concatenating the posterior probability vectors. With other kernel functions, this explicit mapping might be unknown. Empirical results in this paper suggests that the eigenvectors of the PCK has discriminative properties with regards to groups in the data. These properties can be exploited in cluster analysis as suggested in Sec. 5.3.2.

In implementations of the PCK, we are required to let G be sufficiently large to capture the local scale similarities in the data. For a large number of mixture components, the covariance matrices for the mixture distributions might become singular while running the EM-algorithm. If this situation arises, the covariance matrices needs to be regularized as described in Sec. 5.2.

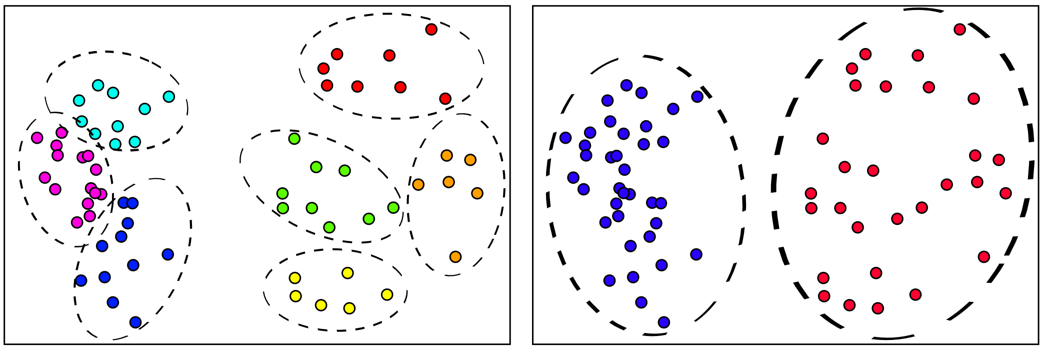


Figure 6.1: (a): Local structure with 7 mixture components. (b): Global structure with two mixture components.

Example 6.1.1 (Motivation)**.** Let g denote the number of mixture components used. The posterior probability vector for the data point $\mathbf{x}_i$, is $\boldsymbol{\gamma}_i = \begin{pmatrix}\gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ig}\end{pmatrix}^T$. Here, γ_{ij} $j = 1, 2, \ldots, g$ is the probability that $\mathbf{x}_i$ is drawn from mixture component j as seen in Eq. (5.24). If two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ have a high probability of being drawn from mixture component k, we see that the inner product between the posterior probability vectors will be large. For instance if $g = 3$, $\boldsymbol{\gamma}_i = \begin{pmatrix}0.80 & 0.15 & 0.05\end{pmatrix}^T$ and $\boldsymbol{\gamma}_j = \begin{pmatrix}0.91 & 0.05 & 0.04\end{pmatrix}$, we have

$$\begin{aligned}\boldsymbol{\gamma}_i^T\boldsymbol{\gamma}_j &= \begin{pmatrix}0.80 & 0.15 & 0.05\end{pmatrix}\begin{pmatrix}0.91\\0.05\\0.04\end{pmatrix}\\ &= 0.80 \cdot 0.91 + 0.15 \cdot 0.05 + 0.05 \cdot 0.04\\ &= 0.7375.\end{aligned}$$

Now, if we swap the first two components in $\boldsymbol{\gamma}_j$, we get

$$\begin{aligned}\boldsymbol{\gamma}_i^T\boldsymbol{\gamma}_j &= \begin{pmatrix}0.80 & 0.15 & 0.05\end{pmatrix}\begin{pmatrix}0.05\\0.91\\0.04\end{pmatrix}\\ &= 0.80 \cdot 0.05 + 0.15 \cdot 0.91 + 0.05 \cdot 0.04\\ &= 0.1785.\end{aligned}$$

This corresponds to a situation where $\mathbf{x}_i$ and $\mathbf{x}_j$ are drawn from different mixture components. We see that this inner product is much smaller than the one where they were drawn from the same mixture component. Thus, the inner product between the posterior vectors is some sort of similarity measure between the data points for a given GMM.

6.2 A connection to Consensus Clustering

There is a clear connection between the consensus clustering methodology and cluster kernels. Recall from Sec. 5.4 that the consensus matrix is calculated by clustering the data several times with different numbers of clusters and counting the number of times pairwise data points has been clustered together. The normalized consensus matrix is defined as $\mathbf{S} = \left\{\frac{n_{ij}}{M}\right\}_{N\times N}$, where n_{ij} is the number of times two data points, $\mathbf{x}_i$ and $\mathbf{x}_j$, are clustered together and M is the number of clusterings. The cluster kernels as defined by Weston et al. [92] is performed in the same way. The data is clusted several times with different number of clusters. The output of the clustering procedures is a hard membership function. This is a vector $\mathbf{m}_i = \begin{pmatrix} m_{i1} & m_{i2} & \cdots & m_{iK}\end{pmatrix}^T$, where $m_{ik} = 1$ if data point $\mathbf{x}_i$ is assigned to cluster k and zero otherwise. When performing M clusterings, element (i, j) of the cluster kernel matrix is defined as

$$k_{ij}^{\text{CK}} = \frac{1}{M}\sum_{m=1}^{M} \mathbf{m}_i^{(\text{m})^T}\mathbf{m}_j^{(\text{m})}, \tag{6.2}$$

where $\mathbf{m}_i^{(m)}$ is the cluster membership function of data point $\mathbf{x}_i$ in clustering trial m. Then $\mathbf{m}_i^{(\text{m})^T}\mathbf{m}_j^{(\text{m})} = 1$ if data point $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned to the same cluster and zero otherwise. Thus, Eq. (6.2) calculates the average number of times two data points are clustered together. This is exactly the same calculation that is performed in consensus clustering. Although these matrices are identical, they were developed independently for different purposes. As far as the author knows, the researchers working on consensus clustering has not mentioned the kernel aspect of the matrix and vice versa. The rest of this section is devoted to deriving a new connection between the more general PCK and the consensus matrix.

Assume that the number of mixture components, $\mathcal{G} \in \{2, 3, \ldots, G\}$ is random. Let $\mathcal{Y}_i = y$ if data point i is drawn from mixture component y. The posterior probabilities from the Gaussian Mixture Models are interpreted as

$$\boldsymbol{\gamma}_i(q, g) = \begin{pmatrix} P(\mathcal{Y}_i = 1|\mathcal{Q} = q, \mathcal{G} = g) \\ P(\mathcal{Y}_i = 2|\mathcal{Q} = q, \mathcal{G} = g) \\ \vdots \\ P(\mathcal{Y}_i = g-1|\mathcal{Q} = q, \mathcal{G} = g) \end{pmatrix}.$$

This is justified by the fact that in the EM-algorithm, we condition on the parameters. The parameters contain information on the number of mixture components. In addition to this, the EM-algorithm calculates a deterministic sequence. If the EM-algorithm is applied several times with the same initial condition, the parameters will converge to the same values for the different trials. Thus, we implicitly condition on the initial condition. Then,

$$\boldsymbol{\gamma}_i(q,g)^T\boldsymbol{\gamma}_j(q,g) = \sum_{y=1}^{g-1} P(\mathcal{Y}_i = y|\mathcal{Q} = q, \mathcal{G} = g)P(\mathcal{Y}_j = y|\mathcal{Q} = q, \mathcal{G} = g).$$

Assuming that $\mathbf{x}_i$ and $\mathbf{x}_j$ are drawn independently from one another, we see that

$$\begin{aligned}\boldsymbol{\gamma}_i(q,g)^T\boldsymbol{\gamma}_j(q,g) &= \sum_{y=1}^{g-1} P(\mathcal{Y}_i = y, \mathcal{Y}_j = y|\mathcal{Q} = q, \mathcal{G} = g)\\ &= P(\mathcal{Y}_i = \mathcal{Y}_j|\mathcal{Q} = q, \mathcal{G} = g).\end{aligned}$$

Now, assume that $\mathcal{G}$ is distributed by $\mathcal{G} \sim P(\mathcal{G} = g)$. Then

$$\begin{aligned}P(\mathcal{Y}_i = \mathcal{Y}_j|\mathcal{Q} = q) &= \sum_g P(\mathcal{G} = g)P(\mathcal{Y}_i = \mathcal{Y}_j|\mathcal{Q} = q, \mathcal{G} = g)\\ &= \sum_g P(\mathcal{G} = g)\boldsymbol{\gamma}_i(q,g)^T\boldsymbol{\gamma}_j(q,g).\end{aligned}$$

If $\mathcal{G}$ is uniformly distributed, i.e.

$$P(\mathcal{G} = g) = \frac{1}{G-1},\ g = 2, 3, \ldots, G,$$

then

$$P(\mathcal{Y}_i = \mathcal{Y}_j|\mathcal{Q} = q) = \sum_{g=2}^{G} \frac{1}{G-1}\boldsymbol{\gamma}_i(q,g)^T\boldsymbol{\gamma}_j(q,g).$$

From this, we see that if we assume that $\mathcal{Q}$ is uniformly drawn from Q different random initial conditions and let $Z = \frac{1}{Q(G-1)}$, the PCK becomes

$$\begin{aligned}K_{\text{PCK}}(\mathbf{x}_i, \mathbf{x}_j) &= \frac{1}{Q(G-1)}\sum_{q=1}^{Q}\sum_{g=2}^{G}\boldsymbol{\gamma}_i(q,g)^T\boldsymbol{\gamma}_j(q,g)\\ &= \frac{1}{Q}\sum_{q=1}^{Q}\sum_{g=2}^{G}\frac{1}{G-1}\boldsymbol{\gamma}_i(q,g)^T\boldsymbol{\gamma}_j(q,g)\\ &= \frac{1}{Q}\sum_{q=1}^{Q} P(\mathcal{Y}_i = \mathcal{Y}_j|\mathcal{Q} = q)\\ &= \sum_{q=1}^{Q}\frac{1}{Q}P(\mathcal{Y}_i = \mathcal{Y}_j|\mathcal{Q} = q)\\ &= P(\mathcal{Y}_i = \mathcal{Y}_j),\end{aligned}$$

which is an estimate of the probability that data point $\mathbf{x}_i$ is drawn from the same mixture component as $\mathbf{x}_j$. This is the same as the probability that two data points belong to the same cluster[1].

As described in Sec. 5.4, the consensus matrix contains the number of times two data points are clustered together using a range of the number of clusters for different clustering algorithms or several runs of the same algorithm. Let Q be the number of different clustering algorithms (or the number of runs for one algorithm) and let G be the maximum number of clusters used. Then we have a total of $Q(G-1)$ partitions/clusterings. If we denote the number of times $\mathbf{x}_i$ and $\mathbf{x}_j$ are co-clustered as n_{ij}, the elements of the normalized consensus matrix are $\frac{n_{ij}}{Q(G-1)}$. From a frequentist point of view, this is the probability of the two data points being clustered together. This is the same interpretation as the elements of the PCK.

[1]Note that since we assumed independent drawing, the interpretation breaks down on the main diagonal of the matrix. Ideally, all these elements should be 1. This is not necessarily the case, although the diagonal elements tend to be very close to 1.

Chapter 7

Learning transition probabilities using the Probabilistic Cluster Kernel

In this chapter, the PCK is used to define transition probabilities in a Markov Chain. These transition probabilities are stored in a stochastic matrix $\mathbf{P}$. The properties of $\mathbf{P}$ yields interesting results from a theoretical point of view when evaluating the stationary distribution of the Markov Chain. It is shown that the stationary distribution of the Markov Chain calculates a linear ranking function in the empirical kernel space defined by the inner products in the kernel function. This is also true for kernel feature space. If the kernel function is an RBF, the stationary distribution calculates nonparametric density estimates using Parzen windowing. To the author's best knowledge, these interpretations of the stationary distribution are new.

The theory in this chapter is derived on general form, assuming that the stochastic matrix is generated from a kernel matrix with positive entries. As the goal of this thesis is to use transition probabilities *learned from data*, the PCK will be used in the end.

The chapter is concluded with an algorithm for ranking on multi attribute data using Markov Chain transition probabilities learned from data. This ranking algorithm produces a *global* rank, not a local rank. For the goal of developing *joint* ranking and clustering, we need the versatility of a local rank. Thus, the theory in this chapter is considered as background theory for Ch. 8.

7.1 Generating a stochastic matrix from a kernel function

Let $G = (V, E)$ be a connected undirected graph where the edge weights are stored in $\mathbf{K}$. Here, $\mathbf{K}$ is a kernel matrix with positive entries. Let $\mathbf{D} = \operatorname{diag}(d_i)$ denote the degree matrix, where degree is defined by $d_i = \sum_{j=1}^{N} k_{ij}$. Then the matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$ is a right stochastic matrix.

Proof. As defined in Tab. 2.1, a right stochastic matrix has positive entries and each row sums to one. Since $\mathbf{K}$ has positive entries, $\mathbf{P}$ will have positive entries. Each row sums to one if $\mathbb{1}$, a vector of ones, is a right eigenvector of $\mathbf{P}$ associated with the eigenvalue 1. Let $\mathbf{d} = \begin{pmatrix} d_1 & d_2 & \cdots & d_N \end{pmatrix}^T$. Since $\mathbf{K}\mathbb{1} = \mathbf{d}$ and $\mathbf{D}^{-1} = \operatorname{diag}(\frac{1}{d_i})$, we have

$$\begin{aligned} \mathbf{P}\mathbb{1} &= \mathbf{D}^{-1}\mathbf{K}\mathbb{1} \\ &= \mathbf{D}^{-1}\mathbf{d} \\ &= \mathbb{1}. \end{aligned}$$

Thus, $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$ is a right stochastic matrix. □

This implies that for a weighted graph G, we can construct a Markov Chain where the transition probabilities are induced by the weights. If the weight between two vertices in the graph is large compared to the other vertices, the transition probability will be large. Thus, the "flow" between vertices is larger for large weights.

7.1.1 Properties of the Stochastic Matrix

Let $\mathbf{P}$ be a right stochastic matrix as defined above. Then $\mathbf{P}$ has the following properties:

1. If $\mathbf{x}$ is a right eigenvector of $\mathbf{P}$ with the corresponding eigenvalue λ, then $\mathbf{D}\mathbf{x}$ is a right eigenvector of $\mathbf{P}^T$ with the same eigenvalue.

   Proof. If $\mathbf{x}$ is a right eigenvector of $\mathbf{P}$ with the eigenvalue λ, then we have

$$\begin{aligned} \mathbf{P}\mathbf{x} &= \lambda\mathbf{x} \\ \mathbf{D}^{-1}\mathbf{K}\mathbf{x} &= \lambda\mathbf{x} \\ \mathbf{K}\mathbf{D}^{-1}\mathbf{D}\mathbf{x} &= \lambda\mathbf{D}\mathbf{x} \\ \mathbf{P}^T(\mathbf{D}\mathbf{x}) &= \lambda(\mathbf{D}\mathbf{x}). \end{aligned} \tag{7.1}$$

   □

2. The stochastic matrix $\mathbf{P}$ has real, positive eigenvalues where the largest eigenvalue is 1.

   Proof. Since $\mathbf{P}$ is a stochastic matrix, we know that its eigenvalues lies within the unit circle in the complex plane with the largest eigenvalue being 1 [95]. If we let $\mathbf{K}_{\mathrm{N}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{K}\mathbf{D}^{-\frac{1}{2}}$, then $\mathbf{D}^{\frac{1}{2}}\mathbf{P}\mathbf{D}^{-\frac{1}{2}} = \mathbf{K}_{\mathrm{N}}$ so $\mathbf{P}$ and $\mathbf{K}_{\mathrm{N}}$ are similar and have the same eigenspectrum. We know that $\mathbf{K}$ is a kernel matrix with the entries $k_{ij} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle$. From the definition of $\mathbf{K}_{\mathrm{N}}$, we see that its elements are $\frac{1}{\sqrt{d_i}\sqrt{d_j}}k_{ij}$, which can be written as inner products since

   $$\begin{aligned} \frac{1}{\sqrt{d_i}\sqrt{d_j}}k_{ij} &= \frac{1}{\sqrt{d_i}\sqrt{d_j}}\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle \\ &= \langle \frac{1}{\sqrt{d_i}}\Phi(\mathbf{x}_i), \frac{1}{\sqrt{d_j}}\Phi(\mathbf{x}_j)\rangle. \end{aligned} \tag{7.2}$$

   Thus, $\mathbf{K}_{\mathrm{N}}$ is a kernel matrix and positive semidefinite. Since $\mathbf{K}_{\mathrm{N}}$ is symmetric and positive semidefinite, it has real, positive eigenvalues. Furthermore, $\mathbf{P}$ is stochastic and has 1 as its largest eigenvalue. These matrices are similar and thus, the result follows. □

3. The stationary distribution of the Markov Chain is given by $\boldsymbol{\pi} \propto \mathbf{D}\mathbb{1}$, where $\mathbb{1}$ is a vector of ones.

   Proof. The graph G is connected and thus the Markov Chain associated with $\mathbf{D}^{-1}\mathbf{K}$ is irreducible. The stationary distribution of the Markov Chain is a left eigenvector of $\mathbf{P}$ corresponding to the eigenvalue 1 (or equivalently, a right eigenvector of $\mathbf{P}^T$). Since $\mathbf{P}$ is a right stochastic matrix, we have that $\mathbb{1}$ is a right eigenvector of $\mathbf{P}$ corresponding to the eigenvalue 1. The stationary distribution $\boldsymbol{\pi} \propto \mathbf{D}\mathbb{1}$ follows directly from the first property. This is indeed a valid stationary distribution as it is also a reversibility distribution. The balance equation is satisfied if

   $$\begin{gathered} p_{ij}\pi_i = p_{ji}\pi_j \\ \frac{k_{ij}}{d_i}d_i = \frac{k_{ji}}{d_j}d_j \\ \Updownarrow \\ k_{ij} = k_{ji}. \end{gathered} \tag{7.3}$$

   This is satisfied as $\mathbf{K}$ is symmetric. □

7.2 The stationary distribution

Property 3 in Sec. 7.1.1 states that the stationary distribution of the Markov Chain induced by the transition probability matrix can be expressed in explicit form as $\boldsymbol{\pi} \propto \mathbf{D}\mathbb{1}$. In this section, the stationary distribution is investigated from a theoretical point of view. The theory is derived in Sec. 7.2.1–Sec. 7.2.3 and discussed in Sec. 7.2.4. To the author's best knowledge, the connections obtained in this section are unknown prior to this thesis.

7.2.1 Projections in the empirical kernel space

In this section, the stationary distribution is investigated from the Empirical Kernel Space. From Eq. (4.25), it is seen that $\mathbf{z}_i \in \mathbb{R}^N$, the embedding of $\mathbf{x}_i$ to the Empirical Kernel Space of $\mathbf{K}$, is given by

$$\begin{aligned}\mathbf{Z} &= \begin{pmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_N^T \end{pmatrix} \\ &= \mathbf{E}\boldsymbol{\Lambda}^{\frac{1}{2}},\end{aligned}$$

where $\mathbf{E}$ is the orthogonal eigenvector matrix of $\mathbf{K}$ and $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix of $\mathbf{K}$. Property 3 in Sec. 7.1.1 states that the stationary distribution of the induced Markov Chain is given by

$$\begin{aligned}\boldsymbol{\pi} &\propto \mathbf{D}\mathbb{1} \\ &= \mathbf{K}\mathbb{1} \\ &= \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T\mathbb{1} \\ &= \mathbf{E}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{E}^T\mathbb{1} \\ &= \left(\mathbf{E}\boldsymbol{\Lambda}^{\frac{1}{2}}\right)\left(\mathbf{E}\boldsymbol{\Lambda}^{\frac{1}{2}}\right)^T\mathbb{1} \\ &= \mathbf{Z}\mathbf{Z}^T\mathbb{1} \\ &= \mathbf{Z}\sum_{i=1}^{N}\mathbf{z}_i \\ &\propto \mathbf{Z}\mathbf{m}, \end{aligned} \tag{7.4}$$

where $\mathbf{m} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{z}_i$ is the sample mean in the Empirical Kernel Space. Thus, the limiting probabilities are given by $\pi_i \propto \mathbf{z}_i^T\mathbf{m} = \mathbf{m}^T\mathbf{z}_i$, $i = 1, 2, \ldots, N$, which is proportional to a projection of $\mathbf{z}_i$ onto the sample mean.

7.2.2 Projections in kernel feature space

As seen in Sec. 7.2.1, the limiting probabilities can be interpreted as a projection on the sample mean in the empirical kernel space. Since KPCA preserves inner products, this is also valid for kernel feature space. This is easily seen, as

$$\begin{aligned}\mathbf{Z}\mathbf{Z}^T &= \mathbf{E}\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{E}^T \\ &= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T \\ &= \mathbf{K}.\end{aligned}$$

Let $\Phi(\cdot)$ be defined as in Sec. 4.2.1. Since the matrix $\mathbf{K}$ is a kernel matrix, its elements are inner products. Thus

$$\begin{aligned}\pi_i &\propto d_i \\ &= \sum_{j=1}^{N} k_{ij} \\ &= \sum_{j=1}^{N} \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \\ &\propto \mathbf{m}_\Phi^T \Phi(\mathbf{x}_i),\end{aligned} \tag{7.5}$$

where $\mathbf{m}_\Phi$ is the sample mean in kernel space and $\Phi(\mathbf{x}_i),\ i = 1, 2, \ldots, N$ is the mapping of $\mathbf{x}_i$.

7.2.3 Nonparametric density estimates

There is a connection between the stationary distribution of the induced Markov Chain and nonparametric density estimates if the kernel function used is an RBF. Let $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2}$, where $\mathbf{x}_i \in \mathbb{R}^p,\ i = 1, 2, \ldots, N$. Then

$$\begin{aligned}\widehat{f}_{\mathcal{X}}(\mathbf{x}_i) &= \frac{1}{(2\pi\sigma^2)^{\frac{p}{2}} N} \sum_{j=1}^{N} \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{p}{2}} N} \sum_{j=1}^{N} e^{-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2}\end{aligned}$$

is a nonparametric estimate of the density function evaluated at $\mathbf{x}_i$ using Parzen windowing [96]. Comparing this to Eq. (7.5), we see that

$$\widehat{f}_{\mathcal{X}}(\mathbf{x}_i) \propto \mathbf{m}_\Phi^T \Phi(\mathbf{x}_i),$$

which is proportional to the limiting probability π_i.

7.2.4 Key observations

In this section, some key observations on the previous results are discussed.

Centering in Kernel Space

In Sec. 4.2, we introduced the concept of centering in kernel space. Centered data implies that $\mathbf{m} = \mathbf{0}$. By Eq. (7.4) and Eq. (7.5), a mean of zero implies that all the rows of $\mathbf{K}$ should sum to zero. For positive kernel functions as assumed in the Markov Chain methodology, *this is never satisfied.* Thus, centering the data when working with Markov Chain methods is not appropriate. This makes sense as negative probabilities do not exist.

Ranking with the stationary distribution

Eq. (7.4) and Eq. (7.5) states that the score of the data points is a linear function. Ranking functions used in the literature for supervised ranking are on the same form, for instance [62, Ch. 8]. Thus, the stationary distribution as a ranking function fits well within the general ranking methodology. Note that this is a linear function in kernel space. It is a nonlinear function in input space.

Nonparametric density estimates

As a special case scenario, it is shown that when the kernel matrix is an RBF, the stationary distribution calculates nonparametric density estimates using Parzen windowing. In this setting, the data points are ranked according to *density* in input space. Similar connections have appeared in other circumstances when investigating connections between information theory, graph theory and Mercer kernels [97].

Are cluster centroids representative in spectral clustering?

As the stationary distribution is proportional to a projection onto the sample mean in the empirical kernel space, data points with the highest degree of connectivity will generally be positioned far away from the origin. The centroid in centroid based clustering is often used as a cluster representative. That is, the centroid should be representative for the data points in the cluster. The score obtained from the stationary distribution should represent importance. The important data points in the sense of a Markov Chain are embedded at the edge of the cluster. Thus, the centroid may not be the most useful alternative as a cluster representative in spectral clustering.

7.3 Learning transition probabilities from data

In the previous sections, it was assumed that the transition probability matrix $\mathbf{P}$ was generated from a kernel matrix with positive entries, $\mathbf{K}$. These kernel matrices are generated from kernel functions. Well known kernel functions, like the RBF, require the choice of a critical width parameter, σ. The choice of this parameter greatly influences the result and is very dependent on the dataset used.

By a coupling of the PCK and Markov chain theory, we are in this work able to learn the transition probabilities from the inherent structures in the data since the PCK is learned from the data and satisfies the properties in Sec. 7.1. We achieve this by generating the matrix $\mathbf{P}$ of transition probabilities from the probabilistic cluster kernel matrix $\mathbf{K}_{\text{PCK}}$, by

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}_{\text{PCK}}.$$

7.4 A global ranking algorithm

In the previous sections, we have shown that given a kernel matrix with positive entries, $\mathbf{K}$, and Markov Chain with the transition probability matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$, $\mathbf{D} = \text{diag}(d_i)$, $i = 1, 2, \ldots, N$, the stationary distribution of the Markov Chain calculates a linear ranking function in the empirical kernel space. By using the PCK such that $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}_{\text{PCK}}$, the transition probababilities are learned from the data and not dependent on critical parameters.

Let

$$\boldsymbol{\pi} = \frac{\mathbf{D}\mathbb{1}}{\mathbb{1}^T\mathbf{D}\mathbb{1}} = \begin{pmatrix}\pi_1 & \pi_2 & \cdots & \pi_N\end{pmatrix}^T$$

be the normalized stationary distribution of the induced Markov Chain. Based on the analysis of the stationary distribution in the previous sections, we propose using the elements of $\boldsymbol{\pi}$ as ranking scores and ordering the data points according to this. That is

$$\mathbf{x}_i \preceq \mathbf{x}_j \Leftrightarrow \pi_i \leq \pi_j, \; i, j = 1, 2, \ldots, N,$$

resolving ties arbitrarily. Here, $\mathbf{x}_i \preceq \mathbf{x}_j$ means that $\mathbf{x}_j$ is preferred to $\mathbf{x}_i$. Note that this ranking is not dependent on the normalization constant $\mathbb{1}^T\mathbf{D}\mathbb{1}$. Thus, ranking using $\boldsymbol{\pi} \propto \mathbf{D}\mathbb{1}$ is equivalent to ranking using the normalized stationary distribution. The algorithm is summarized in Alg. 7

We should note that this is the second algorithm implementing the PCK for unsupervised learning. Although this ranking algorithm can be used for a *global* ranking, joint ranking and clustering requires the versatility of *local*

Algorithm 7 Markov Chain Ranking using the PCK.

```
Input: Data matrix $\mathbf{X}$, number of initial contitions $Q$ and maximum number of clusters $G$.
1: Compute the PCK $\mathbf{K}_{\text{PCK}}$ as described in Ch. 6.
2: Compute the degree (score) of each data point as $d_i = \sum_{j=1}^{N} k_{ij}$.
Output: Score of each data point.
```

ranking. The experiments in this thesis is focused on the theory in Ch. 8. Results obtained using Alg. 7 will be presented at a later point in time.

Chapter 8

Joint ranking and clustering

In the previous chapter, it was shown that a transition probability matrix can be generated from a positive kernel function **K**. When using the PCK, the transition probabilities are *learned* from the data as the PCK is learned from the inherent structures in the data. This chapter is focused on developing a *joint* ranking and clustering algorithm for multi attribute data using the learned transition probabilities. The basis for the algorithm is the personalized PageRank (PPR) as presented in Sec. 3.1.1. Normally, this is only used for ranking.

Previous work by Chung and Zhao [98] forms the basis for the theory in this chapter. We will define an embedding for the data to a kernel space. This embedding preserves a generalized effective resistance between vertices in the graph as squared Euclidean distance, which will be exploited for clustering. Furthermore, the PPR is expressed as projections in this vector space. Because of the versatility of the PPR in terms of the seed distribution, this enables us to combine ranking and clustering to output an ordered list of data points both within and across clusters for the first time in this framework. This is helpful for the analysis of the clustering result.

This chapter starts with a review on work done by Chung and Zhao [98], which is extended to propose an embedding of the data. The embedding will be analyzed in terms of distances between the data points. Based on this analysis, the embedding is proposed to be used for clustering. An illustrative example on synthetic data is presented in order to highlight properties of the algorithm.

8.1 Previous work

This section is based on the work by Chung and Zhao [98]. In this paper, the authors study the relationship between the PPR and graph invariants often considered when dealing with random walks and electrical networks. The PPR is for instance used to approximate effective resistance.

8.1.1 A Symmetrical PageRank

Let $G = (V, E)$ be a connected graph, where the edges are weighted by $\mathbf{K}$. Recall from Sec. 3.1.1 that the PPR is calculated from the iterative sequence

$$\mathbf{r}_{k+1}{}^T = (1-\alpha)\mathbf{r}_k{}^T\mathbf{P} + \alpha\mathbf{s}^T,\ 0 < \alpha < 1, \tag{8.1}$$

where $\mathbf{P}$ is a right stochastic matrix, $\mathbf{s}$ is the seed distribution and α is the restart probability. This difference equation converges to the stationary distribution of

$$\mathbf{P}' = (1-\alpha)\mathbf{P} + \alpha\mathbb{1}\mathbf{s}^T.$$

Let $\boldsymbol{\pi}(\alpha, \mathbf{s})$ be the solution of Eq. (8.1). Then

$$\boldsymbol{\pi}(\alpha, \mathbf{s})^T = (1-\alpha)\boldsymbol{\pi}(\alpha, \mathbf{s})^T\mathbf{P} + \alpha\mathbf{s}^T, \tag{8.2}$$

where $\boldsymbol{\pi}(\alpha, \mathbf{s})$ is cosidered unknown. Solving Eq. (8.2) for $\boldsymbol{\pi}(\alpha, \mathbf{s})$ yields

$$\begin{aligned}
\boldsymbol{\pi}(\alpha, \mathbf{s})^T &= (1-\alpha)\boldsymbol{\pi}(\alpha, \mathbf{s})^T\mathbf{P} + \alpha\mathbf{s}^T \\
\boldsymbol{\pi}(\alpha, \mathbf{s})^T(\mathbf{I} - (1-\alpha)\mathbf{P}) &= \alpha\mathbf{s}^T \\
\boldsymbol{\pi}(\alpha, \mathbf{s})^T(\alpha\mathbf{I} + (1-\alpha)(\mathbf{I} - \mathbf{P})) &= \alpha\mathbf{s}^T \\
\frac{1-\alpha}{\alpha}\boldsymbol{\pi}(\alpha, \mathbf{s})^T\left(\frac{\alpha}{1-\alpha}\mathbf{I} + \mathbf{I} - \mathbf{P}\right) &= \mathbf{s}^T \\
\frac{1}{\beta}\boldsymbol{\pi}(\alpha, \mathbf{s})^T(\beta\mathbf{I} + \mathbf{I} - \mathbf{P}) &= \mathbf{s}^T,
\end{aligned}$$

where $\beta = \frac{\alpha}{1-\alpha}$[1]. If the stochastic matrix $\mathbf{P}$ is on the form $\mathbf{D}^{-1}\mathbf{K}$, where $\mathbf{K}$ is symmetric with positive entries we get

$$\begin{aligned}
\frac{1}{\beta}\boldsymbol{\pi}(\alpha, \mathbf{s})^T(\beta\mathbf{I} + \mathbf{I} - \mathbf{D}^{-1}\mathbf{K}) &= \mathbf{s}^T \\
\frac{1}{\beta}\boldsymbol{\pi}(\alpha, \mathbf{s})^T\mathbf{D}^{-1}(\beta\mathbf{D} + \mathbf{D} - \mathbf{K}) &= \mathbf{s}^T \\
\frac{1}{\beta}\boldsymbol{\pi}(\alpha, \mathbf{s})^T\mathbf{D}^{-1}(\beta\mathbf{D} + \mathbf{L}) &= \mathbf{s}^T \\
\frac{1}{\beta}\boldsymbol{\pi}(\alpha, \mathbf{s})^T\mathbf{D}^{-1}\mathbf{L}_\beta &= \mathbf{s}^T, \qquad (8.3)
\end{aligned}$$

where we define $\mathbf{L}_\beta = \beta\mathbf{D} + \mathbf{L}$ as the β-adjusted Laplacian. Here, $\mathbf{L} = \mathbf{D} - \mathbf{K}$ is the regular Laplacian matrix. This can be solved using a Green's function. In our case, we use a discrete Green's function [99]. A discrete Green's function $\mathbf{G}_\beta$ for $\mathbf{L}_\beta$ satisfies

$$\mathbf{G}_\beta\mathbf{L}_\beta = \mathbf{L}_\beta\mathbf{G}_\beta = \mathbf{I}.$$

We see that since $\mathbf{L}_\beta$ is symmetric, we have

$$\mathbf{L}_\beta = \sum_{i=1}^{N} \lambda_i \mathbf{e}_i\mathbf{e}_i^T = \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T,$$

where λ_i are the eigenvalues of $\mathbf{L}_\beta$ and $\mathbf{e}_i$ are the corresponding orthonormal eigenvectors. Clearly, we have

$$\mathbf{G}_\beta = \sum_{i=1}^{N} \frac{1}{\lambda_i}\mathbf{e}_i\mathbf{e}_i^T = \mathbf{E}\boldsymbol{\Lambda}^{-1}\mathbf{E}^T, \qquad (8.4)$$

under the assumption that $\lambda_i > 0 \ \forall \ i \in \{1, 2, \ldots, N\}$. The author has not seen a proof in the literature that this is satisfied. However, the proof is simple.

Proof. To show that this assumption is satisfied, we need to show that $\mathbf{L}_\beta$ is positive definite. That is, $\lambda_i > 0 \ \forall \ i \in \{1, 2, \ldots, N\}$ iff $\mathbf{y}^T\mathbf{L}_\beta\mathbf{y} > 0 \ \forall \ \mathbf{y} \in \mathbb{R}^N \setminus \{\mathbf{0}\}$. Consider the *$\beta$-normalized* Laplacian matrix

$$\boldsymbol{\mathcal{L}}_\beta = \beta\mathbf{I} + \boldsymbol{\mathcal{L}},$$

[1]The observant reader will notice that Chung and Zhao [98] use $\beta = \frac{2\alpha}{1-\alpha}$. This is because they introduce a *lazy* random walk, which is not needed when using the PCK.

where $\boldsymbol{\mathcal{L}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$ is the symmetric normalized Laplacian. The β-normalized Laplacian has the same eigenvectors as $\boldsymbol{\mathcal{L}}$, but its eigenvalues are shifted by β. Thus,

$$\boldsymbol{\mathcal{L}}_\beta = \sum_{i=1}^{N}(\lambda_i^* + \beta)\boldsymbol{\phi}_i\boldsymbol{\phi}_i^T,$$

where $0 = \lambda_1^* < \lambda_2^* \leq \ldots \leq \lambda_N^*$ are the eigenvalues of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\phi}_i$ are the corresponding orthonormal eigenvectors. Then, we have

$$\begin{aligned}
\mathbf{y}^T\mathbf{L}_\beta\mathbf{y} &= \mathbf{y}^T\mathbf{D}^{\frac{1}{2}}\boldsymbol{\mathcal{L}}_\beta\mathbf{D}^{\frac{1}{2}}\mathbf{y} \\
&= (\mathbf{D}^{\frac{1}{2}}\mathbf{y})^T\boldsymbol{\mathcal{L}}_\beta(\mathbf{D}^{\frac{1}{2}}\mathbf{y}) \\
&= \mathbf{z}^T\boldsymbol{\mathcal{L}}_\beta\mathbf{z} && (\mathbf{z} = \mathbf{D}^{\frac{1}{2}}\mathbf{y}) \\
&= \mathbf{z}^T\sum_{i=1}^{N}(\lambda_i^* + \beta)\boldsymbol{\phi}_i\boldsymbol{\phi}_i^T\mathbf{z} \\
&= \sum_{i=1}^{N}(\lambda_i^* + \beta)(\mathbf{z}^T\boldsymbol{\phi}_i)(\boldsymbol{\phi}_i^T\mathbf{z}) \\
&= \sum_{i=1}^{N}(\lambda_i^* + \beta)x_i^2 && (x_i = \boldsymbol{\phi}_i^T\mathbf{z} = \mathbf{z}^T\boldsymbol{\phi}_i) \\
&= \beta x_1^2 + (\lambda_2^* + \beta)x_2^2 + \ldots + (\lambda_N^* + \beta)x_N^2 \\
&> 0,
\end{aligned}$$

as $\boldsymbol{\phi}_i,\ i = 1, 2, \ldots, N$ forms an orthonormal basis in $\mathbb{R}^N$. Thus, $\mathbf{L}_\beta$ is positive definite and has strictly positive eigenvalues. □

Now, from Eq. (8.3) we get

$$\begin{aligned}
\frac{1}{\beta}\boldsymbol{\pi}(\alpha, \mathbf{s})^T\mathbf{D}^{-1}\mathbf{L}_\beta &= \mathbf{s}^T \\
\boldsymbol{\pi}(\alpha, \mathbf{s})^T &= \beta\mathbf{s}^T\mathbf{G}_\beta\mathbf{D} \\
\boldsymbol{\pi}(\alpha, \mathbf{s}) &= \beta\mathbf{D}\mathbf{G}_\beta\mathbf{s}. \qquad (8.5)
\end{aligned}$$

We see that up to a change of basis, $\mathbf{G}_\beta$ is a symmetrical form of the PPR.

8.1.2 Generalized Effective Resistance

For a weighted graph, G, we can interpret the weights on the edges as electrical conductance of an electrical network. The effective resistance [100] between vertex i and vertex j in the graph is found by injecting a unit current at vertex i, extracting it at vertex j and using Kirchhoff's laws. This has been shown to be related to commute time in Markov Chains and is a well known graph metric used to measure dissimilarity. The effective resistance can be calculated using a Green's function of the graph Laplacian. Specifically, if we let $\mathbf{G}$ be the Green's function of the graph Laplacian $\mathbf{L}$ and let $\boldsymbol{\chi}_i$ be an indicator vector where every element is 0 except element i which is 1, the effective resistance is given by

$$R(i,j) = (\boldsymbol{\chi}_i - \boldsymbol{\chi}_j)^T \mathbf{G} (\boldsymbol{\chi}_i - \boldsymbol{\chi}_j). \tag{8.6}$$

Chung and Zhao [98] defines the *generalized* effective resistance as

$$R_\alpha(i,j) = \beta(\boldsymbol{\chi}_i - \boldsymbol{\chi}_j)^T \mathbf{G}_\beta (\boldsymbol{\chi}_i - \boldsymbol{\chi}_j). \tag{8.7}$$

An upper bound for the difference between the effective resistance and the generalized effective resistance is provided. This upper bound is given by

$$|\beta R(i,j) - R_\alpha(i,j)| \leq \frac{\beta^2}{\lambda_1^2}\left(\frac{1}{d_i} + \frac{1}{d_j}\right), \tag{8.8}$$

where λ_1 is the smallest nontrivial eigenvalue of $\mathbf{L}$.

The generalized effective resistance is used in Sec. 8.3.1 as an interpretation of squared Euclidean distances in the embedding proposed in Sec. 8.2.

8.2 Proposed embedding

A key point in this thesis is to notice that the Green's function, $\mathbf{G}_\beta$, is positive definite. Thus, it is a kernel matrix with elements corresponding to inner products in some Reproducing Kernel Hilbert Space (RKHS). This will exploited for joint ranking and clustering by embedding the data to the empirical kernel space[2].

Now that this has been established, the next step is to recognize that

$$\begin{aligned}\mathbf{G}_\beta &= \mathbf{E}\boldsymbol{\Lambda}^{-1}\mathbf{E}^T \\ &= \mathbf{E}\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{E}^T \\ &= (\mathbf{E}\boldsymbol{\Lambda}^{-\frac{1}{2}})(\mathbf{E}\boldsymbol{\Lambda}^{-\frac{1}{2}})^T,\end{aligned} \tag{8.9}$$

[2]In the latter stages of this work, the author noticed that [101] also noticed the positive semidefiniteness of the Green's function, however, [101] used this fact in a completely different setting for estimating the Green's function through random projections.

where $\mathbf{E}$ is the orthonormal eigenvector matrix of $\mathbf{L}_\beta$ and $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix of $\mathbf{L}_\beta$. Let $\mathrm{Vol}(G) = \sum_{i=1}^N d_i = \mathbb{1}^T\mathbf{D}\mathbb{1}$. From Eq. (8.5), we get

$$\begin{aligned}
\boldsymbol{\pi}(\alpha, \mathbf{s}) &= \beta\mathbf{D}\mathbf{G}_\beta\mathbf{s} \\
&= \beta\mathbf{D}(\mathbf{E}\boldsymbol{\Lambda}^{-\frac{1}{2}})(\mathbf{E}\boldsymbol{\Lambda}^{-\frac{1}{2}})^T\mathbf{s} \\
&= \frac{1}{\mathrm{Vol}(G)}\mathbf{D}(\sqrt{\beta}\sqrt{\mathrm{Vol}(G)}\,\mathbf{E}\boldsymbol{\Lambda}^{-\frac{1}{2}})(\sqrt{\beta}\sqrt{\mathrm{Vol}(G)}\,\mathbf{E}\boldsymbol{\Lambda}^{-\frac{1}{2}})^T\mathbf{s} \qquad (8.10) \\
&= \frac{1}{\mathrm{Vol}(G)}\mathbf{D}\mathbf{Z}_\beta\mathbf{Z}_\beta^T\mathbf{s}, \qquad (8.11)
\end{aligned}$$

where

$$\mathbf{Z}_\beta = \begin{pmatrix} \mathbf{z}_1(\beta)^T \\ \mathbf{z}_2(\beta)^T \\ \vdots \\ \mathbf{z}_N(\beta)^T \end{pmatrix} = \sqrt{\beta}\sqrt{\mathrm{Vol}(G)}\,\mathbf{E}\boldsymbol{\Lambda}^{-\frac{1}{2}}$$

is an embedding in of the data preserving the inner products in $\mathbf{G}'_\beta = \beta\,\mathrm{Vol}(G)\,\mathbf{G}_\beta$. For the rest of this thesis, the matrix $\beta\,\mathrm{Vol}(G)\,\mathbf{G}_\beta$ is denoted by $\mathbf{G}'_\beta$ to improve readability. The eigenvector and eigenvalue matrix is on the form

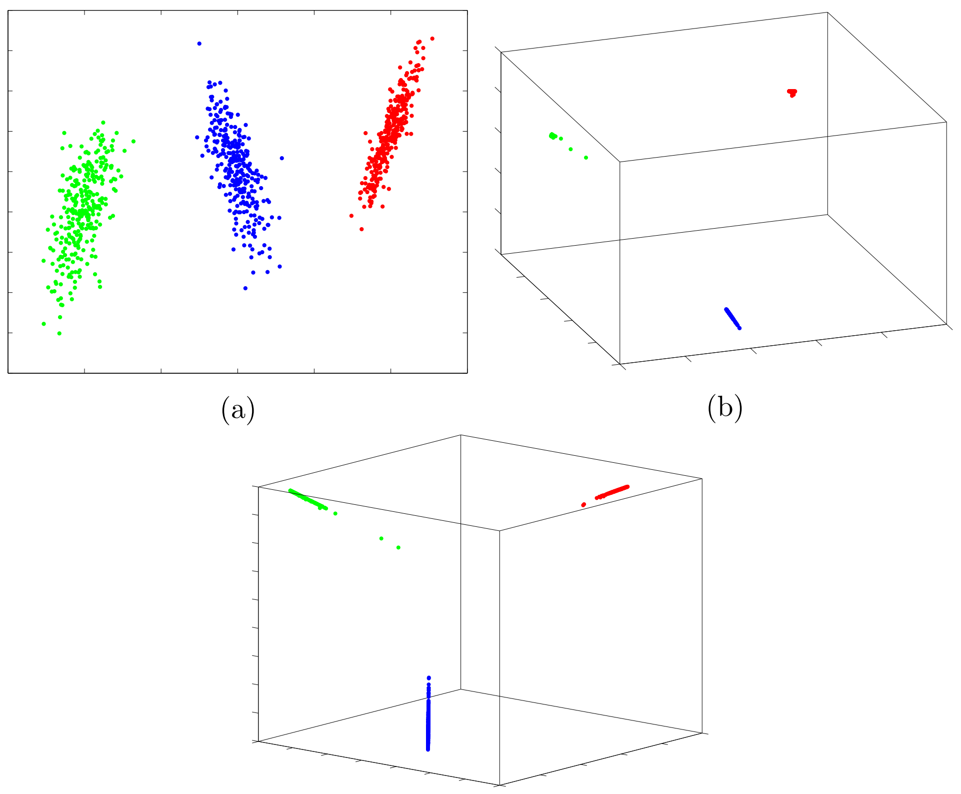
$$\mathbf{E} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_N \end{pmatrix}$$

and

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_N \end{pmatrix},$$

where $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N$ are the orthonormal eigenvectors of $\mathbf{L}_\beta$ and $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$ are the corresponding eigenvalues. We see from Eq. (4.25) that this is equivalent to the in-sample form of KPCA on the kernel matrix $\mathbf{G}'_\beta$. Notice that the eigenvalues are shown in *ascending* order. This is because $\mathbf{G}_\beta$ has the same eigenvectors as $\mathbf{L}_\beta$ but with *inverted* eigenvalues. Since KPCA requires eigenvalues in descending order in the eigenvalue matrix, the eigenvalues of $\mathbf{L}_\beta$ needs to be in ascending order. The smallest eigenvalues of $\mathbf{L}_\beta$ will then introduce the largest eigenvalues in $\mathbf{G}_\beta$. The volume term $\mathrm{Vol}(G)$ is introduced to keep the scale approximately equal for different sized datasets.

Example 8.2.1 (Embedding)**.** Fig. 8.1a shows a dataset containing three clusters of bivariate normally distributed data, each with $N = 300$ data



(a) (b) (c)

Figure 8.1: (a): Three clusters of bivariate normally distributed data. (b): Embedding of the data in (a) using the PCK and the embedding proposed in this section with $\alpha = 0.15$. (c): KPCA of the data using the PCK.

points. A PCK, $\mathbf{K}_{\text{PCK}}$, was trained on this dataset with $Q = G = 20$. Using the kernel matrix and $\alpha = 0.15$, the data was embedded to a 3-dimensional space with the embedding proposed in Sec. 8.2. This is shown in Fig. 8.1b. We see that the embedding successfully discriminates between the three clusters. Fig. 8.1c shows an embedding using KPCA directly on the PCK. In this dataset, the PageRank embedding seems to contain more compact clusters than the one using KPCA on the PCK.

8.3 Analysis

8.3.1 Distances in kernel feature space

In Sec. 8.1.2, we presented a generalized effective resistance as defined by Chung and Zhao [98]. If we expand Eq. (8.7) and use the fact that $\mathbf{G}_\beta$ is symmetric, we get

$$\begin{aligned} R_\alpha(i,j) &= \beta(\boldsymbol{\chi}_i^T \mathbf{G}_\beta \boldsymbol{\chi}_i - 2\boldsymbol{\chi}_i^T \mathbf{G}_\beta \boldsymbol{\chi}_j + \boldsymbol{\chi}_j^T \mathbf{G}_\beta \boldsymbol{\chi}_j) \\ &= \frac{1}{\mathrm{Vol}(G)} \left(\beta \,\mathrm{Vol}(G)\, G_\beta(i,i) - 2\beta \,\mathrm{Vol}(G)\, G_\beta(i,j) + \beta \,\mathrm{Vol}(G)\, G_\beta(j,j)\right) \\ &= \frac{1}{\mathrm{Vol}(G)} \left(G'_\beta(i,i) - 2G'_\beta(i,j) + G'_\beta(j,j)\right), \end{aligned} \tag{8.12}$$

where $G'_\beta(i,j)$ is found in row i, column j of the kernel matrix $\mathbf{G}'_\beta = \{\langle \Psi(\mathbf{x}_i), \Psi(\mathbf{x}_j)\rangle\}_{N\times N}$. Substituting this into Eq. (8.12) yields

$$\begin{aligned} \mathrm{Vol}(G)\, R_\alpha(i,j) &= \langle \Psi(\mathbf{x}_i), \Psi(\mathbf{x}_i)\rangle - 2\langle \Psi(\mathbf{x}_i), \Psi(\mathbf{x}_j)\rangle + \langle \Psi(\mathbf{x}_j), \Psi(\mathbf{x}_j)\rangle \\ &= \|\Psi(\mathbf{x}_i) - \Psi(\mathbf{x}_j)\|^2. \end{aligned} \tag{8.13}$$

Thus, an embedding of the data points to the Empirical Kernel Space defined by $\mathbf{G}'_\beta$ ensures that the squared Euclidean distances between the data points is proportional to the generalized effective resistances between vertices in the graph.

8.3.2 The PPR as projections

From (8.11) we see that

$$\begin{aligned} \boldsymbol{\pi}(\alpha, \mathbf{s}) &= \frac{1}{\mathrm{Vol}(G)} \mathbf{D}\mathbf{Z}_\beta \sum_{j=1}^{N} s_j \mathbf{z}_j(\beta) \\ &= \frac{1}{\mathrm{Vol}(G)} \mathbf{D}\mathbf{Z}_\beta \mathbf{m}_\mathbf{s}, \end{aligned} \tag{8.14}$$

so

$$\pi_i(\alpha, \mathbf{s}) = \frac{d_i}{\mathrm{Vol}(G)} \mathbf{z}_i(\beta)^T \mathbf{m}_\mathbf{s}, \tag{8.15}$$

where $\mathbf{m}_s = \sum_{j=1}^N s_j \mathbf{z}_j(\beta)$ is a weighted mean. The weights are determined by the seed distribution $\mathbf{s}$. If we let π_i be the score determined by the stationary distribution of a regular random walk on $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$, we have seen in Sec. 7.2 that $\pi_i = \frac{d_i}{\mathbb{1}^T\mathbf{D}\mathbb{1}} = \frac{d_i}{\mathrm{Vol}(G)}$. Then, the PPR becomes

$$\pi_i(\alpha, \mathbf{s}) = \pi_i \mathbf{z}_i(\beta)^T \mathbf{m}_\mathbf{s}. \tag{8.16}$$

From a random walk on $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$, we obtain the *base score*, π_i. Eq. (8.16) states that there are two factors contributing to the score obtained by the PPR: the base score and a factor proportional to the projection onto a weighted mean vector. We see that the effects of the restart in the random walk comes from the projection. The restart probability, α, is encoded in the embedding and the seed distribution is encoded in the weighted mean.

Using a uniform seed distribution

In this section we will investigate what happens if the seed distribution is uniform over a given set of data points. This will result in an interpretation of the PPR with respect to the k-means cluster centroids.

Let V be a set of data points. If $\mathbf{s}$ is uniform over V, then

$$\begin{aligned}\mathbf{m}_\mathbf{s} &= \sum_{j=1}^{N} s_j \mathbf{z}_j(\beta) \\ &= \frac{1}{|V|} \sum_{\mathbf{j} \in V} \mathbf{z}_j(\beta),\end{aligned}$$

which is the sample mean of the data points in V. When clustering with k-means, the algorithm outputs cluster centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$. These are the sample means of the data points assigned to the clusters. Thus, ranking the data using a uniform seed distribution over the data points in a particular cluster discovered by k-means is equivalent to inserting its cluster centroid in Eq. (8.14) or Eq. (8.16). This is useful, either for within cluster ranking or ranking data points outside a cluster with respect to the data points in a cluster.

8.4 Putting it all together

Now that the theory has been presented, we are ready to observe key points on the theory in this chapter:

- From Eq. (8.13), we see that the squared Euclidean distances in the embedding are proportional to the generalized effective resistance in Eq. (8.7). Although not equivalent, this distance metric is similar to the effective resistance on graphs. The squared Euclidean distances between data points and the cluster centroids is a key factor in the cost function of the k-means algorithm. Minimizing the cost function of k-means in this embedding corresponds to minimizing generalized effective resistances. The experiments in this thesis reveals that this distance metric is able to preserve group structures in the data.

- From Eq. (8.16), we see that the score obtained when ranking with PPR has a geometrical interpretation. Up to a base score factor, it is proportional to a projection onto a weighted mean. If this weighted mean is the sample mean of a cluster centroid obtained using k-means, this corresponds to a uniform seed distribution over the data points in the cluster. Thus, when using the cluster centroids as the weighted means, the seed distribution is in some way *learned* by clustering the data.

The theory in this chapter has been derived without mentioning which matrix $\mathbf{K}$ to use when generating the transition probability matrix $\mathbf{P}$. By the arguments in Ch. 7, a suitable choice is the PCK as the transition probabilities are learned from the data. As the experiments in this thesis suggests, this kernel function works well without any modifications to parameters across different types of data. One more factor that has not been mentioned is the restart probability α. As suggested in [2, 1], $\alpha = 0.15$ is a good choice. This α value is used in all the experiments in this thesis. Although no mathematical justification for this value is provided, empirical results in the experiments suggests that this value is an appropriate choice.

Stages of the procedure

The joint framework for ranking and clustering consists of three stages.

1. The data is embedded using Alg. 8. This includes training a PCK on the data to learn transition probabilities.

2. This embedding is used for clustering using Alg. 9.

3. The data is ranked using Alg. 10. The seed distribution is chosen based on the purpose of the analysis. To rank within and across clusters, uniform distributions based on the cluster assignments are good choices. As noted above, this corresponds to projecting the data onto the cluster centroids learned by k-means.

Algorithm 8 Embedding using PCK.

```
Input: Data matrix $\mathbf{X}$, number of initial contitions $Q$, maximum number of clusters $G$ and restart probability $\alpha$.
1: Compute the Probabilistic Cluster Kernel $\mathbf{K}_{\text{PCK}}$ as described in Ch. 6.
2: Compute the $\beta$-adjusted Laplacian $\mathbf{L}_\beta = \beta\mathbf{D} + \mathbf{L}$, where $\mathbf{L} = \mathbf{D} - \mathbf{K}_{\text{PCK}}$, $\mathbf{D} = \text{diag}(d_i) = \text{diag}(\sum_{j=1}^{N} k_{ij})$ and $\beta = \frac{\alpha}{1-\alpha}$.
3: Compute the orthogonal eigenvector matrix $\mathbf{E}$ and the diagonal eigenvalue matrix $\mathbf{\Lambda}$ of $\mathbf{L}_\beta$, where the eigenvalues are ordered in ascending order.
4: Compute the embedding as $\mathbf{Z}_\beta = \sqrt{\beta}\sqrt{\text{Vol}(G)}\,\mathbf{E}\mathbf{\Lambda}^{-\frac{1}{2}}$, where $\text{Vol}(G) = \sum_{i=1}^{N} d_i$.
Output: Embedding of the data.
```

Algorithm 9 Clustering the embedded data.

```
Input: Embedded data $\mathbf{Z}_\beta$ and the number of clusters $k$.
1: Cluster the data in $\mathbf{Z}_\beta$ using $k$-means with $k$ dimensions of the embedding.
Output: Cluster assignments and cluster centroids.
```

Algorithm 10 Ranking of the embedded data.

```
Input: Embedded data $\mathbf{Z}_\beta$, degree matrix $\mathbf{D}$ and seed distribution $\mathbf{s}$.
1: Compute the weighted mean $\mathbf{m}_\mathbf{s} = \sum_{i=1}^{N} s_i \mathbf{z}_i(\beta)$, where $\mathbf{z}_i(\beta)^T$ is located in row $i$ of $\mathbf{Z}_\beta$.
2: Compute the scores $\boldsymbol{\pi}(\alpha, \mathbf{s}) = \mathbf{D}\mathbf{Z}_\beta\mathbf{m}_\mathbf{s}$.
3: Sort the scores in descending order.
Output: Ordered list of data points according to the sorted scores.
```

Part III

Experiments

Chapter 9

Experiment setup

9.1 Parameter Choice

9.1.1 Probabilistic Cluster Kernel

A key feature of the Probabilistic Cluster Kernel is that it is not strongly dependent on the value of any parameters to get good results. The two parameters we need to set is the number of initial conditions, Q, and the maximum number of clusters, G. As long as these are sufficiently high, the results are stable over a range of parameter values. In all the experiments, these parameters are held fixed at $Q = G = 30$. By experimentation and testing, this value is found to be an appropriate choice, independent of the dataset used. Previous experience indicates that in some datasets, a value of 20–25 might be a bit low. A value higher than 30 did not yield different results than using 30. Thus, to emphasize the robustness of the PCK, these are fixed at $Q = G = 30$ for all the experiments in this thesis. Note that the results were always good when using $Q = G = 20$. In some cases, especially with high dimensional data like the Frey Faces, the PCK performed better with $Q = G = 30$.

9.1.2 The restart probability

The restart probability used in the personalized PageRank dictates how often the random walk should restart according to the seed distribution. Google has reported that they use $\alpha = 0.15$ [1, 2]. In all the experiments of this thesis, the restart probability is held fixed at $\alpha = 0.15$ as this seems to be a good trade off between cluster discrimination and mixing in the Markov Chain for ranking purposes.

9.2 Clustering procedure

Every clustering is performed by clustering the embedded data with k-means 100 times. The final partition is chosen by evaluating the cost function and choosing the partition with the lowest cost function value. This ensures that no human interaction can affect the results.

Chapter 10

Results

This chapter presents the experiments and the results obtained. The thought behind selecting these particular datasets is that the results can be visualized in print. Only one of the datasets have ground truth information on group structure. For the other datasets, the challenge is that there is no guarantee that there are group structures in the data. Thus, choosing the number of clusters is a challenge and numerical comparisons of accuracies with other methods are not relevant. Some aspects of the method may be compared to other methods where appropriate.

In this chapter, the names for the methods used will be abbreviated. Thus, we need to establish some notation. KPCA using the Probabilistic Cluster Kernel is called *KPCA-PCK*. Embedding the data using Alg. 8 is called *JRC-EMBED-PCK*. Embedding the data using Alg. 8 but with an RBF instead of the PCK is called *JRC-EMBED-RBF*. Ranking with Alg. 10 is called *JRC-RANK-PCK*.

10.1 Cloud Screening

The task in this experiment is to decide which pixels in an optical multispectral satellite image is a part of a cloud and which pixels are not. In Remote Sensing, it is important to classify cloud pixels with high accuracy as the clouds covers the ground areas we want to analyze. The image is taken over an area in Spain in 2003 with the Medium Resolution Imaging Spectrometer (MERIS) on the Evironmental Satellite (ENVISAT). An RGB composite of the image is shown in Fig. 10.1a. Each pixel of the 1153 px $\times$ 1153 px image is considered as a data point. Thus, there are $1153^2 = 1329409$ data points in the dataset. Each pixel consists of 6 physically inspired features [102] and 13 spectral bands. In total, there are 19 features. In addition to the image

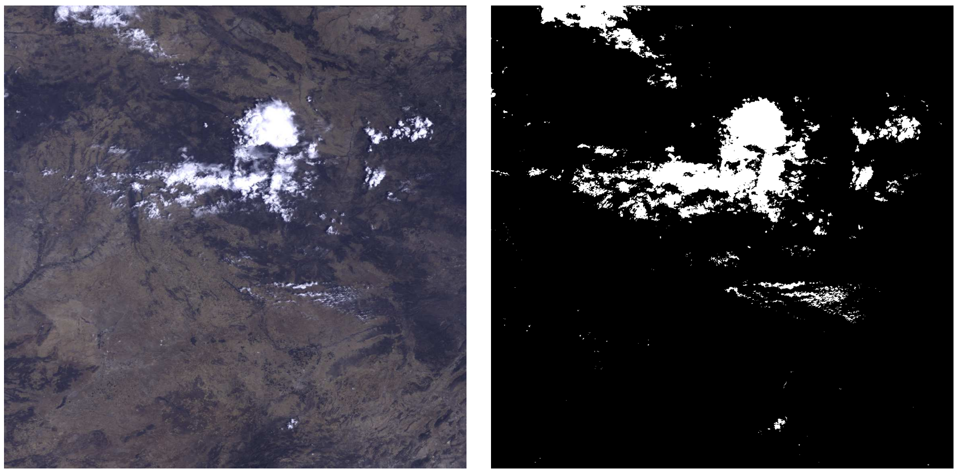

(a) (b)

Figure 10.1: (a): RGB composite of the satellite image. (b): Ground truth labels for the pixels. The white pixels represents clouds. Black pixels represents ground.

itself, we have ground truth data. Experts have labeled the pixels of the image as "cloud" or "ground". These labels are shown in Fig. 10.1b. This particular image has been use previously in [103] for cloud screening using Kernel ECA Spectral Clustering [64] and an Information Theoretic K-NN clustering approach [104] with good results.

10.1.1 Clustering and classification

Since this is a large dataset, a similar approach to Vikjord and Jenssen [104] is used instead of clustering the whole image. They draw a fixed number of pixels from each class at random, cluster these using their Information Theoretic K-NN Clustering method and use the result as training labels with a 1-NN classifier to classify the remaining pixels in the image.

Based on the ground truth labels, a sample of 200 cloud pixels and 200 ground pixels are drawn from the image. This sample is embedded using *JRC-EMBED-PCK*. The data is clustered with k-means using $k = 3$. In this dataset, the value of k is chosen by investigating the eigenvalues of $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}_{\text{PCK}}$. The theory behind this is not presented here, but the idea is that for a similarity matrix $\mathbf{K}$ with dominating diagonal blocks, the number of eigenvalues approximately equal to 1 in the stochastic matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$ should be equal to the number of diagonal blocks. For data containing clusters, the number of diagonal blocks in $\mathbf{K}_{\text{PCK}}$ should be equal to

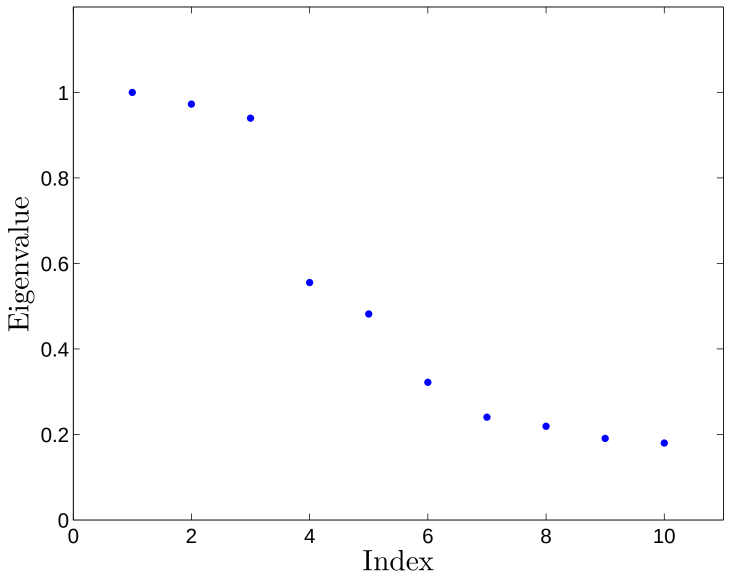


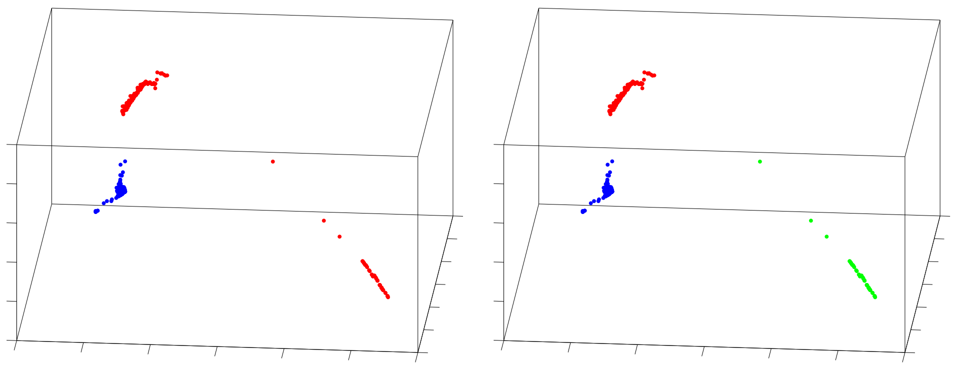
Figure 10.2: Eigenvalues of $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}_{\mathrm{PCK}}$. These indicates 3 clusters in the data.

the number of clusters in the data. These eigenvalues are shown in Fig. 10.2[1]. There are three eigenvalues close to 1 with a gap to the fourth eigenvalue. Thus, we expect three clusters in the data.

The embedded data is shown in Fig. 10.3a. The blue data points are ground pixels, while the red data points are cloud pixels. We see that there are indeed three clusters in the data, as indicated by Fig. 10.2. Although this was suspected prior to embedding the data, this is a surprising result. Other spectral methods applied to this dataset have assumed $k = 2$ by default. Since the inherent properties of the PCK allows it to learn the structures in the data, surprising, but more realistic, results like this might appear.

Fig. 10.3a reveals that the cloud class can be subdivided into two groups. The result of the clustering procedure is shown in Fig. 10.3b. Fig. 10.3c shows the data embedded using *KPCA-PCK*. The data points are colored using the same colors as Fig. 10.3b. The three clusters from *JRC-EMBED-PCK* is apparent in this representation too. This is not surprising, as the kernel matrix is used as a basis for both embeddings. *JRC-EMBED-PCK* seems to increase the distance between the green cluster and the two other compared to *KPCA-PCK*. This might be advantageous for better separability between clusters in some circumstances.

[1]The kernel matrix is thresheld at the value $t = 0.05$ to improve the diagonal blocks prior to generating the stochastic matrix.


(a) (b)

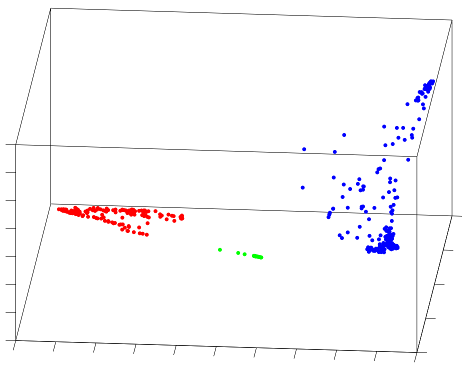

(c)

Figure 10.3: (a): Embedded data using *JRC-EMBED-PCK* with ground truth labels. (b): Embedded data using *JRC-EMBED-PCK*, clustered with $k = 3$. (c): Embedded data using *KPCA-PCK*. The data points are colored according to the colors in (b).

A 1-NN classifier is trained using the labels of the three classes in Fig. 10.3b. The remaining pixels in the image is classified using this classifier. The classification map is shown in Fig. 10.4a. The black, grey and white pixels corresponds to the blue, red and green class respectively in Fig. 10.3b. Notice that the white pixels seems to be in the center of the clouds. These might correspond to dense clouds. As the classifier was trained using three classes, the results cannot be compared to the results in [103] or to the ground truth labels. However, since the green class consists of cloud pixels, merging the green and red class is a possibility. A new 1-NN classifier is trained using the labels obtained by merging the red and green class. The remaining pixels are classified with a classification accuracy of 99.13%. This is comparable to the best result obtained in [103] of 99.41%. In [103], other methods like k-means and kernel k-means were tested on this image with worse results.

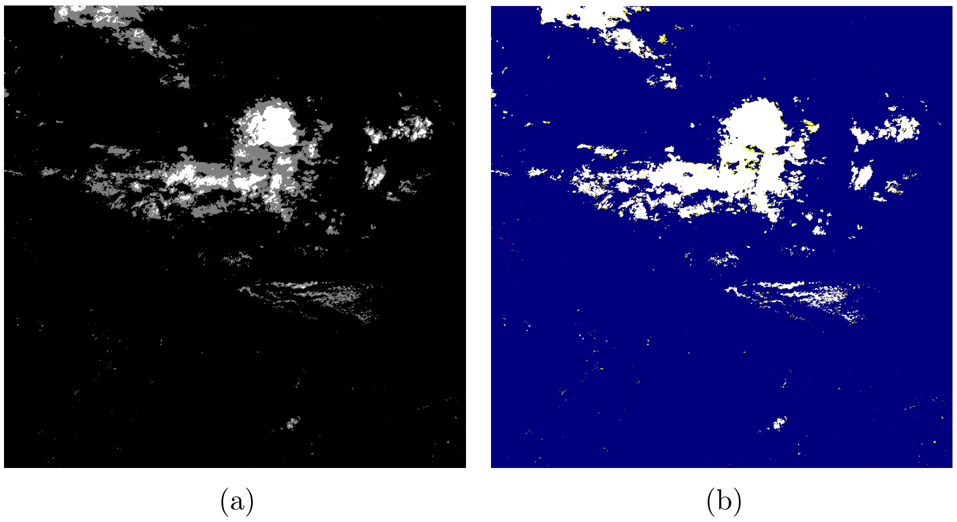


Figure 10.4: (a): Classification map with three classes. Black pixels indicate ground, while grey and white pixels indicate cloud. The white pixels might correspond to dense clouds. (b): Classification map with merged cloud classes. Blue- and white colors indicate correctly classified ground and cloud pixels. Red color indicates cloud classified as ground. Yellow indicates ground classified as cloud.

The classification map is shown in Fig. 10.4b. Most of the error comes from ground pixels classified as cloud. It would be worse if the errors came from cloud pixels classified as ground since the goal is to eliminate the cloud pixels prior to analyzing the ground pixels.

10.1.2 Using ranking information to adjust the training set

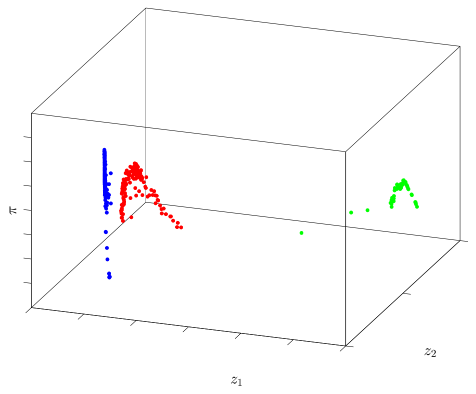
In Sec. 10.1.1, a classifier was trained based on cluster assignments. It was shown that *JRC-EMBED-PCK* can be used to discriminate between cloud and ground pixels in the image. However, experiences show that the classification accuracy is dependent on the initial randomly sampled training data. It is suspected that the cloud pixels which are the most similar to the ground pixels could introduce noise in the training set. Especially since the 1-NN classifier is very simple. If for instance a test point is approximately equidistant to both a cloud and ground pixel in the training set, the classification assignment would be questionable. For more robust classifiers, like the Support Vector Machine, this might not be a problem. The idea now is to use ranking information to reduce the training set to only contain the data points which are important for the structure of the data set. This is motivated by Fig. 10.5a, where the global score of the data points is plotted as a function of the two first components in *JRC-EMBED-PCK*. It is clear that the data points near the boundary between ground and cloud pixels are ranked lower than the other pixels and thus they are not very important for the overall structure.

Since this framework for joint ranking and clustering allows for it, this section is dedicated to investigating the use of ranking information to adjust the training set. This is done by drawing random samples from the classes and using the top ranked data points from each class as the training set. The labels used in the classifier are found by clustering the sample.

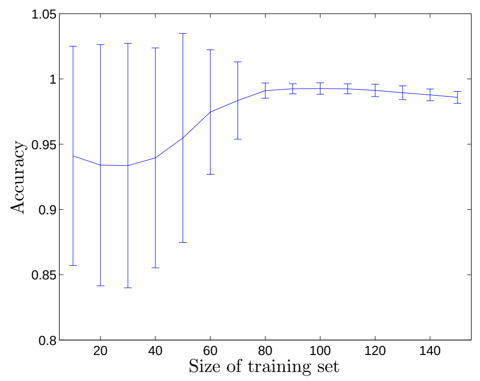
A total of $m = 200$ random samples a of size $n = 200 \times 2$ data points are drawn from the image with balanced classes (cloud/ground). Each sample is embedded using *JRC-EMBED-PCK* and clustered. The global ranking score is calculated using *JRC-RANK-PCK* with a uniform seed distribution over the whole sample. A subset of $n_{\text{top}} \in \{10, 20, 30, \ldots, 150\}$ of the n_{top} top ranked data points from each class assigned by the clustering procedure are used to train a 1-NN classifier. The remaining pixels in the image is classified and the overall accuracy is calculated. Fig. 10.5b shows the mean overall accuracy with standard deviations for the different sample sizes used. The maximum accuracy of 99.68% is found with a subset of 110 data points per class. Using the whole training set on this run, an accuracy of 99.02% is obtained.

For each of the sample sizes, a 95% confidence interval for the difference between the accuracy of the subset and the accuracy of the whole sample is calculated. This is shown in Fig. 10.5c. By using 80–130 data points per class in the training set the overall accuracy is significantly better than the accuracy when using the whole sample. The result is summarized in Tab. 10.1.

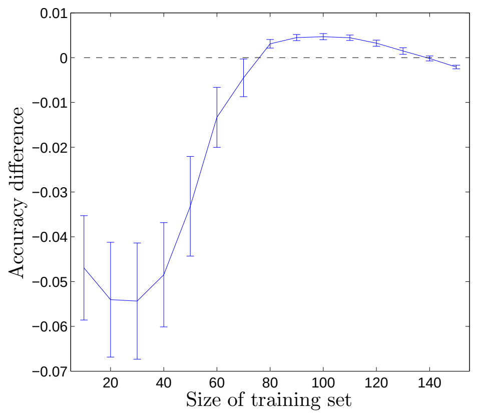
This section is concluded with a few words on the methods used in this experiment. The clustering results in this experiment differ from results of other methods. In particular, the PCK finds structures in the data indicating that the cloud class can be divided into two groups. The clustering method used by Vikjord and Jenssen [104] found two clusters. One for cloud pixels and one for ground pixels. Using the PCK, the clustering result had to be investigated manually to verify that the unexpected third cluster belonged to the cloud class before training the classifier. Thus, this method is not entirely unsupervised. In addition to this, there seems to be a sweet spot for the number of top ranked data points used in the training set to obtain the best classification accuracy. This sweet spot probably differs for different datasets. It is not known if this method of adjusting the training set is



(a)



(b)



(c)

Figure 10.5: (a): Global ranking score as a function of the two first components of *JRC-EMBED-PCK*. (b): Mean accuracy for the different training set sizes. The error bars indicate the standard deviation of the accuracy. (c): Confidence intervals of the mean difference between the global accuracy and the accuracy using a reduced training set.

Table 10.1: Confidence interval and p-value for the difference between the mean accuracy for the reduced training set and the mean accuracy of the total training set. The null hypothesis is that the means are equal, while the alternative hypothesis is that the means are not equal.

Sample size	Confidence Interval		p-value
	θ_{lower}	θ_{upper}	
10	-0.05864	-0.03521	< 0.0001
20	-0.06692	-0.04114	< 0.0001
30	-0.06740	-0.04128	< 0.0001
40	-0.06021	-0.03672	< 0.0001
50	-0.04435	-0.02201	< 0.0001
60	-0.01999	-0.006652	0.0001015
70	-0.00865	-0.0003526	0.03354
80	0.002146	0.004032	< 0.0001
90	0.003767	0.005207	< 0.0001
100	0.003900	0.005448	< 0.0001
110	0.003744	0.005160	< 0.0001
120	0.002427	0.004052	< 0.0001
130	0.0006063	0.002362	0.0009720
140	-0.0009745	0.0006007	0.6410
150	-0.002887	-0.001302	< 0.0001

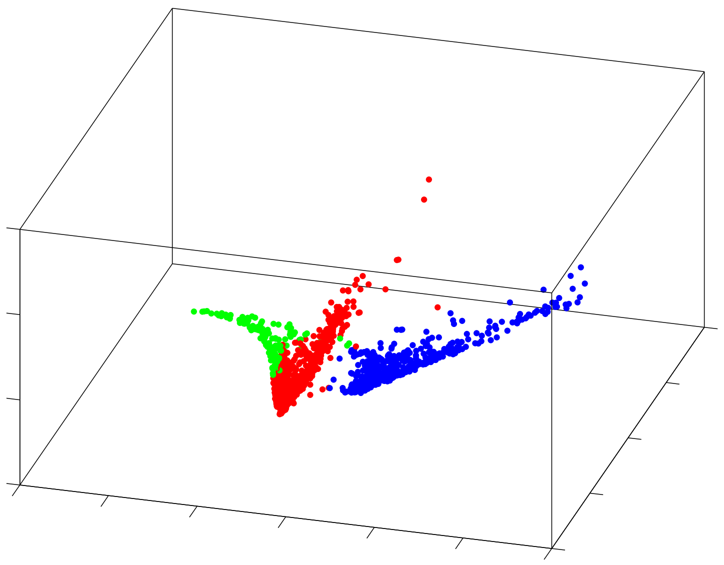
beneficial for other datasets or other classifiers. This has to be investigated further in the future.

10.2 Frey Faces

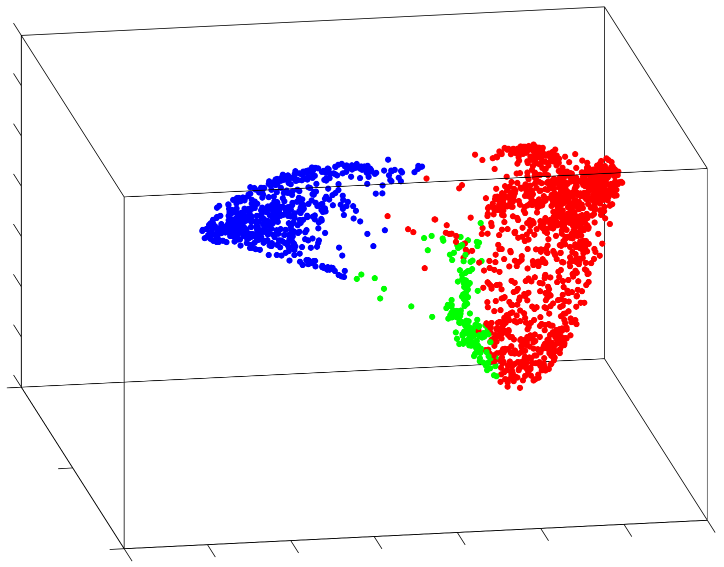
The Frey Faces dataset originates from Brendan Frey who recorded a video of himself while changing his facial expression. The dataset consists of 1965 28px × 20px grayscale images that have been extracted from the video. These images are reshaped into 560 dimensional vectors. This is an interesting dataset for clustering because there is no guarantee that the data contains clusters since it is just a video of his face.

Using *JRC-EMBED-PCK*, the data is embedded into a 3-dimensional space and clustered using k-means with $k = 3$. The embedding and clustering result is shown in the scatterplot in Fig. 10.6a. There are two separable clusters in the data. The structure of one of them seems to separate into two branches. A reasonable explanation to this is that the images in the two branches are somewhat similar to each other and have a common relative. Here, relative means that his facial expression in the video changes from one branch to the other via the relative. Thus, the images in the branches might be different enough from each other to be separable, but because of their relative, they are still connected in the structure of the data. Fig. 10.6b shows the embedding of the data using *KPCA-PCK*. The data points in the scatter plot are colored according to the clustering partition shown in Fig. 10.6a. The cluster with branches when using *JRC-EMBED-PCK* does not show a similar structure when using *KPCA-PCK*.

Fig. 10.7 shows 50 randomly drawn images from each of the three clusters. In the blue cluster, he seems to be smiling. In the other two, he is not. The red and green cluster seems to be different from each other in the sense that in the red cluster, he looks straight ahead or to the left. In the green cluster, he looks to the right. The facial expressions are also less apathic in the green cluster than in the red. Ranking will now be incorporated in the experiment by showing the highest ranked images within each cluster. The ranking is performed using *JRC-RANK-PCK*. Both with a global uniform seed distribution and with uniform seed distributions within each cluster. This should in theory give us the images which are the most representative for each cluster and the images from the other clusters which are the most similar to the images in the cluster we rank with respect to.

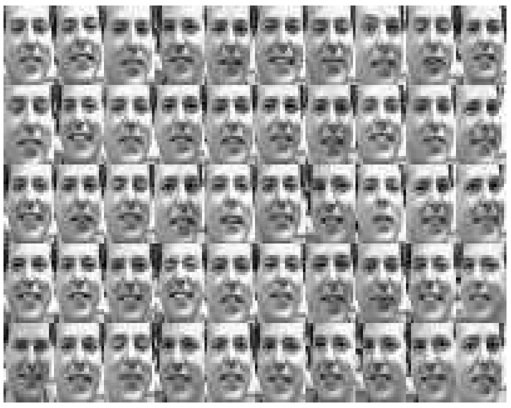


(a)

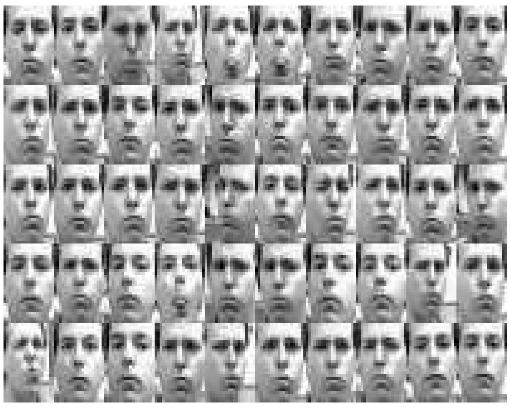


(b)

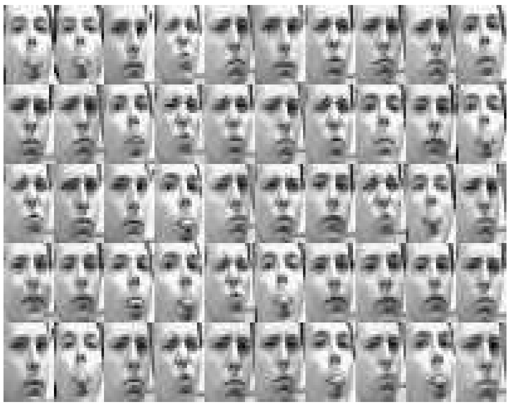
Figure 10.6: Embedding of the Frey Face dataset using: (a) *JRC-EMBED-PCK*(b) *KPCA-PCK*. Data points have the same color in both embeddings.



(a)



(b)



(c)

Figure 10.7: 50 randomly drawn images from: (a) The blue cluster (b) The red cluster (c) The green cluster.

10.2.1 Ranking the result

Fig. 10.8 shows rankings of the data using several different seed distributions. For each ranking, the top 10 data points of each cluster are shown as columns. The different rankings are displayed as rows in the figure. In the top row, the ranking is calculated using a global uniform seed. The following rows are ranked using a uniform seed distribution over the blue, red and green cluster respectively.

From the global ranking (Fig. 10.8, top row), we can clearly see differences between the three clusters. Within each cluster, all of the top ranked images are very similar. In the blue cluster, he is smiling. In the red cluster, he is frowning and looking straight ahead. In the green cluster, the frowning and looking to the right. Notice that the frowning in the green cluster is more prominant than in the red cluster. Since the red and green cluster is a part of the same structure in the empirical kernel space, is expected that even though the top ranked images are different, they contain similar images to one another. This is seen in the relative rankings.

The images in the second row of Fig. 10.8 shows the top ranked images with respect to the smiling cluster. In the top ranked image of the red cluster, it looks like he is on the verge to smile. This is not seen in any of the other top ranked images of this cluster. In the green cluster, there are several images where he smiles. In these images, he is still looking to the right.

In the third row, the images are ranked according to the red cluster. This does not seem to provide us with any more information within the other clusters other than that the green cluster might not contain any images where he does not look to the right.

As we see in the last row, when ranking with respect to the green cluster, he looks to the right in all the top ranked images in all of the clusters. In the blue cluster, he is still smiling, but has turned his head to the right. In the red cluster, the images are very similar to the images we have previously seen in the green cluster. This was expected, as these belong to the same structure in the embedding.

10.2.2 Changing the kernel function

To show off the validity of using the Probabilistic Cluster Kernel as the similarity measure for this dataset, the embedding has been performed using an RBF for comparison. Since the Probabilistic Cluster Kernel does not have any critical parameters, rules of thumb for the width parameter σ has to be used for comparable methodologies. Shi and Malik [16] suggests using 10–20% of the total range of pairwise distances. Jenssen [64] suggests using

(a)

(b)

(c)

(d)

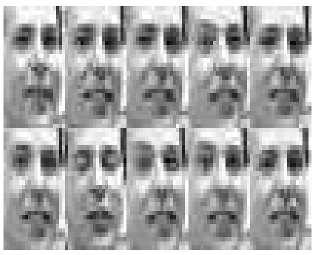
(e)

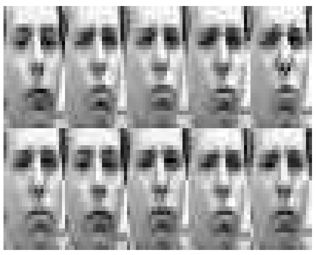
(f)

(g)

(h)

(i)


(j)


(k)

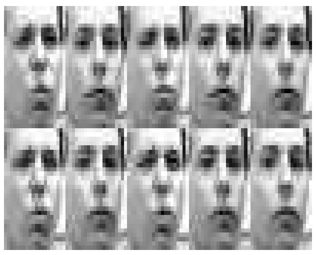

(l)

Figure 10.8: Top ranked images in each of the three clusters. (a, d, g, j): Blue cluster. (b, e, h, k): Red cluster. (c, f, i, l): Green cluster. First row: Global uniform seed. Row 2, 3 and 4: Uniform seed over the Blue, Red and Green cluster respectively.

10–20% of the median or mean range of pairwise distances. The latter is used in this experiment with σ being 15% of the median range of pairwise Euclidean distances. This yields $\sigma \approx 130$.

Using $\alpha = 0.15$ and $\sigma = 130$, the data is embedded using *JRC-EMBED-RBF*. The embedded data is shown in Fig. 10.9. Most of the data is embedded close to the origin, while four data points are embedded along the axes. During the experimentation with the embedding algorithm, the author has experienced that data points with a low degree of connectivity (i.e. low ranked points according to Ch. 7) might be severely penalized in terms of the generalized effective resistance (i.e. distance in the embedding). These could be interpreted as outliers. If the four data points with the lowest degree of connectivity are removed, the situation shown in Fig. 10.10a is obtained. This might allow the clustering procedure to work. The data points are embedded onto two axes. Because of this angular difference between two groups, the data is clustered using a variant of k-means using a cosine distance. The clustering result is shown in Fig. 10.10b. 50 random data points drawn from each of the two clusters are shown in Fig. 10.11. Both clusters seems to contain both smiling and frowning faces. The blue cluster shown in Fig. 10.11a does seem to have a larger proportion of smiling faces, while the red cluster in Fig. 10.11b seems to have a larger proportion of frowning faces. Even though we do get results using *JRC-EMBED-RBF*, they are a bit more ambiguous than with *JRC-EMBED-PCK*.

We would like the reader to note that even though *JRC-EMBED-PCK* worked out of the box, we had to make an effort to produce results using *JRC-EMBED-RBF*. Both by removing low degree data points and by changing the dissimilarity measure used in the k-means cost function. This is probably because the width parameter is sub-optimal. It is not straight forward to find the optimal width parameter. Especially when no ground truth data on groups is available. Thus, a kernel function with no critical parameters is advantageous.

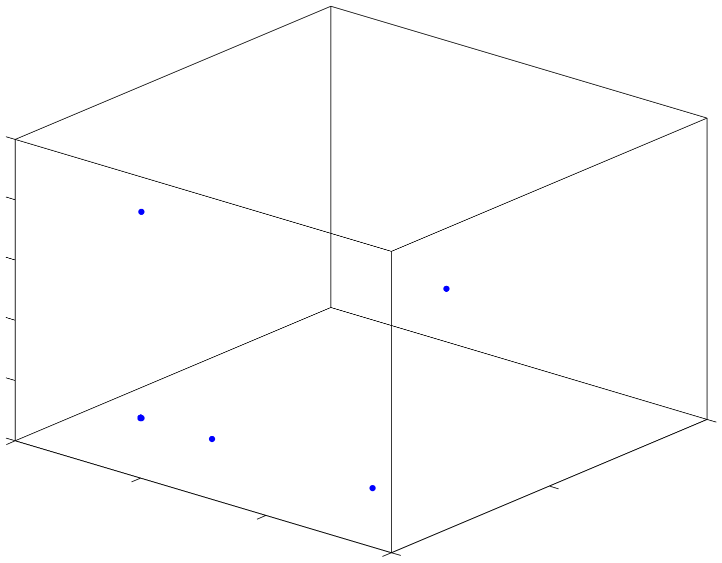
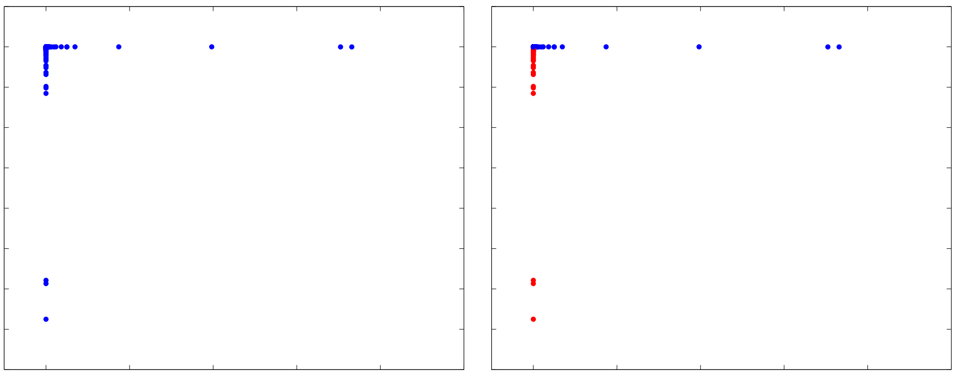
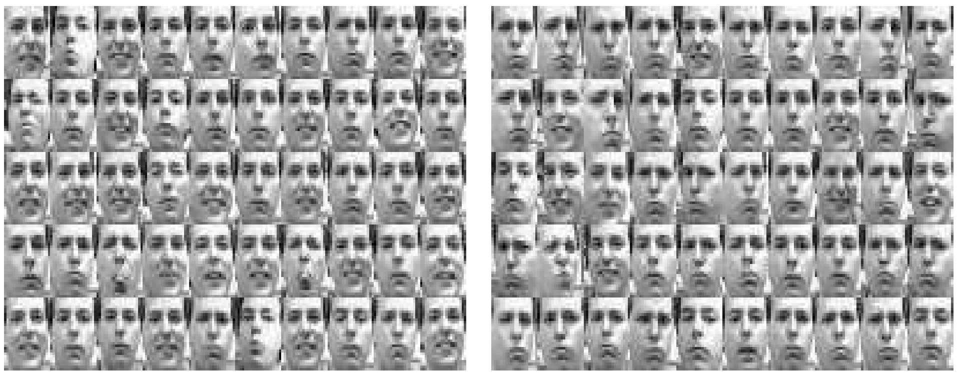


Figure 10.9: Frey Face data embedded using *JRC-EMBED-RBF* with $\alpha = 0.15$.



(a) (b)

Figure 10.10: Frey Face data embedded using *JRC-EMBED-RBF* with $\alpha = 0.15$. The four lowest ranked data points according to Alg. 7 are removed. (a): Embedded data. (b): Embedded data with cluster labels.



(a) (b)

Figure 10.11: 50 randomly drawn images from each of the two groups found using an RBF as the similarity measure.

10.3 NIPS Conference Dataset

The NIPS dataset[2] is generated using a bag-of-words model and includes papers from the NIPS conference in the pre electronic era from 1987–1999. In total, there are $N = 1740$ papers by 2037 authors. There are $d = 13649$ unique words in the vocabulary. In the bag-of-words model, the frequency of each unique word in the paper is used as features. Each paper is regarded as a data point in this experiment.

In this dataset, the number of data points is much lower than the number of features. Thus, the rank of the covariance matrices which are calculated during the EM-iterations for the Probabilistic Cluster Kernel is less than d. This implies that the covariance matrices are singular and thus, non-invertible. Since we are required to invert these matrices in the EM-iterations, it is not possible to calculate the Probabilistic Cluster Kernel for the original dataset. However, by reducing the dimensionality of the data as a preprocessing step this problem is avoided. In this thesis, Principal Component Analysis and Kernel Principal Component Analysis has been presented for this purpose. Neither will be used here. Instead, a generalization of PCA called Singular Value Decomposition (SVD) will be used. This is frequently used for Latent Semantic Indexing which is based on bag-of-words models [105]. The theory behind the SVD will not be presented here, but the interested reader may look at [58, Ch. 6.4].

Motivated by the scree plot in Fig. 10.12, $d' = 20$ is chosen as the dimensionality of the dimensionality reduced dataset. The data is embedded using *JRC-EMBED-PCK* and clustered with k-means using $k = 4$. Here, $k = 4$ is chosen with the hope of discovering four themes in the papers. It is not expected that this dataset contains separable clusters. However, it is expected that data points embedded within the same area are somewhat similar. A 3-dimensional visualization of this is shown in Fig. 10.13. The clusters contains 522, 621, 413 and 184 papers for the blue, red, green and black cluster respectively.

To help the interpretation of the clustering result, the data is ranked using *JRC-RANK-PCK* with a uniform seed distribution over each of the clusters. The idea is that the top ranked papers are the ones which are the most representative for the clusters. The titles of the top 20 ranked papers within each cluster are listed in Tab. 10.2. Each column contains papers in one cluster. The rows contains the paper titles.

It is interesting to see that each cluster seems to have its own theme. In the blue cluster, the papers are either related to Neural Networks or other

[2]http://cs.nyu.edu/∼roweis/data.html

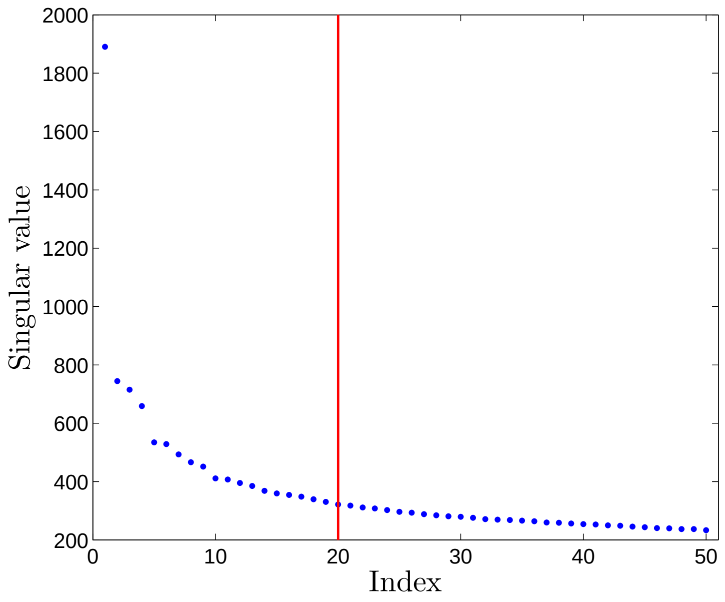


Figure 10.12: Scree plot from the singular value decomposition of the NIPS data.

network architectures. The top ranked papers are application specific. The papers in the red cluster seems to be statistically motivated, and theoretical results. In the green cluster, the papers seems to have a biological theme. Many of the papers have words related to the brain or the eyes in the title. Some of the papers even mentions specific animal species. On the front page of the NIPS web page [106], it says:

> The Neural Information Processing Systems (NIPS) Foundation is a non-profit corporation whose purpose is to foster the exchange of research on neural information processing systems in their biological, technological, mathematical, and theoretical aspects. Neural information processing is a field which benefits from a combined view of biological, physical, mathematical, and computational sciences.

With emphasis on *biological* in the quote, this is a natural theme for papers at the NIPS conference. In the black cluster, nearly every paper within the top 20 ranked papers have Reinforcement Learning or Q-Learning[3] in the title. There are three papers (including the top ranked paper) that does not. These papers have been investigated manually. All three of them are related to Reinforcement Learning.

[3]Q-Learning is a variant of Reinforcement Learning.

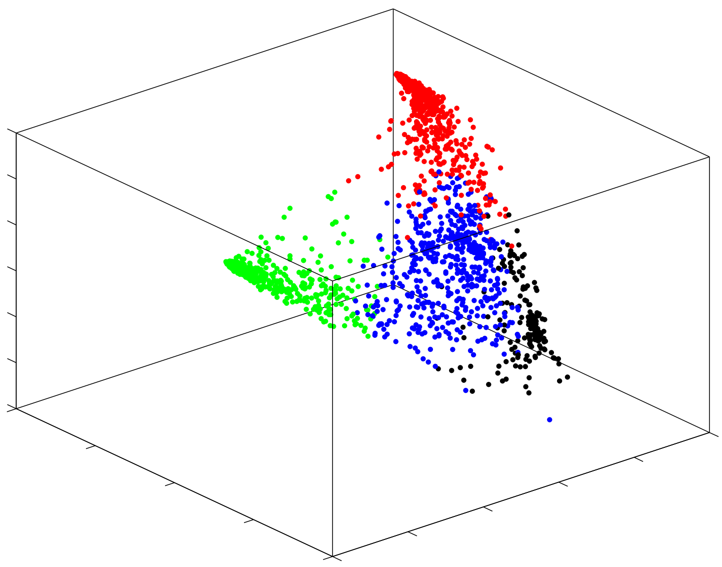


Figure 10.13: Embedding and cluster solution for the NIPS dataset.

Table 10.2: Top 20 ranked papers within each of the four clusters.

Rank	Blue	Red	Green	Black
1.	Integrated Segmentation and Recognition of Hand-Printed Numerals	Monotonicity Hints	A Computer Modeling Approach to Understanding the Inferior Olive and Its Relationships to the Cerebellar Cortex in Rats	Convergence of Indirect Adaptive Asynchronous Value Iteration Algorithms
2.	Applications of Error Back-Propagation to Phonetic Classification	The Error Coding and Substitution PaCTs	The Computation of Sound Source Elevation in the Barn Owl	REINFORCEMENT LEARNING METHODS FOR CONTINUOUS-TIME MARKOV DECISION PROBLEMS
3.	A Short-Term Memory Architecture for the Learning of Morphophonemic Rules	Probabilistic Methods for Support Vector Machines	Computer Simulation of Oscillatory Behavior in Cerebral Cortical Networks	Risk Sensitive Reinforcement Learning
4.	Signature Verification Using a ”Siamese” Time Delay Neural Network	Discovering Hidden Features with Gaussian Processes Regression	Interaction Among Ocularity, Retinotopy and On-center/Off-center Pathways	Memory-based Reinforcement Learning: Efficient Computation with Prioritized Sweeping
5.	A CONNECIIONIST TECHNIQUE FOR ACCELERATED TEXTUAL INPUT: LETTING A NETWORK DO THE TYPING	General Bounds on Bayes Errors for Regression with Gaussian Processes	A Computationally Robust Anatomical Model for Retinal Directional Selectivity	Scheduling Straight-Line Code Using Reinforcement Learning and Rollouts
6.	A Self-Organizing Integrated Segmentation and Recognition Neural Net	Learning Curves for Gaussian Processes	Retinogeniculate Development: The Role of Competition and Correlated Retinal Activity	Finite-Sample Convergence Rates for Q-Learning and Indirect Algorithms
7.	Generalization Performance in PARSEC–A Structured Connectionist Parsing Architecture	Extensions of a Theory of Networks for Approximation and Learning	Visual Cortex Circuitry and Orientation Tuning	Reinforcement Learning for Continuous Stochastic Control Problems
8.	Multi-Digit Recognition Using a Space Displacement Neural Network	FROM DATA DISTRIBUTIONS TO REGULARIZATION IN INVARIANT LEARNING	Orientation Contrast Sensitivity from Long-range Interactions in Visual Cortex	Low Power Wireless Communication via Reinforcement Learning
9.	Dimensionality Reduction and Prior Knowledge in E-Set Recognition	Geometry of Early Stopping in Linear Networks	Neural Network Simulation of Somatosensory Representational Plasticity	Adaptive Choice of Grid and Time in Reinforcement Learning
10.	Tonal Music as a Componential Code: Learning Temporal Relationships between and within Pitch and Timing Components	Practical Confidence and Prediction Intervals	Analog Computation at a Critical Point	The Asymptotic Convergence-Rate of Q-learning
11.	A Constructive RBF Network for Writer Adaptation	NeuroScale: Novel Topographic Feature Extraction using RBF Networks	An Architectural Mechanism for Direction-tuned Comical Simple Cells: The Role of Mutual Inhibition	Automated Aircraft Recovery via Reinforcement Learning: Initial Experiments
12.	A Large-Scale Neural Network Which Recognizes Handwritten Kanji Characters	ASYMPTOTICS OF GRADIENT-BASED NEURAL NETWORK TRAINING ALGORITHMS	Feedback Synapse to Cone and Light Adaptation	Enhancing Q-Learning for Optimal Asset Allocation
13.	Incremental Parsing by Modular Recurrent Connectionist Networks	BAYESIAN QUERY CONSTRUCTION FOR NEURAL NETWORK MODELS	Instabilities in Eye Movement Control: A Model of Periodic Alternating Nystagmus	Learning Macro-Actions in Reinforcement Learning
14.	Connectionist Music Composition Based on Melodic and Stylistic Constraints	Generalization Properties of Radial Basis Functions	Mapping Between Neural and Physical Activities of the Lobster Gastric Mill	Optimal Asset Allocation Using Adaptive Dynamic Programming
15.	A CONVOLUTIONAL NEURAL NETWORK HAND TRACKER	Predictive Approaches for Choosing Hyperparameters in Gaussian Processes	Lower Boundaries of Motoneuron Desynchronization via Renshaw Interneurons	Stable Fitted Reinforcement Learning
16.	Recognizing Overlapping Hand-Printed Characters by Centered-Object Integrated Segmentation and Recognition	The Generalisation Cost of RAMnets	Development and Regeneration of Eye-Brain Maps: A Computational Model	Exploring Unknown Environments with Real-71me Search or Reinforcement Learning
17.	Learning to See Where and What: Training a Net to Make Saccades and Recognize Handwritten Characters	Bayesian Transduction	Effects of Spike Timing Underlying Binocular Integration and Rivalry in a Neural Model of Early Visual Cortex	Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding
18.	Handwritten Digit Recognition with a Back-Propagation Network	An Information-theoretic Learning Algorithm for Neural Network Classification	THE ELECTRONIC TRANSFORMATION: A TOOL FOR RELATING NEURONAL FORM TO FUNCTION	Convergence of Stochastic Iterative Dynamic Programming Algorithms
19.	Natural Dolphin Echo Recognition Using an Integrator Gateway Network	Radial Basis Functions: A Bayesian Treatment	The Role of Activity in Synaptic Competition at the Neuromuscular Junction	Robust Reinforcement Learning in Motion Planning
20.	SEXNET: A Neural Network Identifies Sex From Human Faces	Ordered Classes and Incomplete Examples in Classification	An Analog VLSI Model of Central Pattern Generation in the Leech	Integrated Modeling and Control Based on Reinforcement Learning

10.4 Queries

In this section, ranking using *JRC-RANK-PCK* is explored for image queries. This is done to show that the Probabilistic Cluster Kernel is able to learn similarities between data points which can be used with the personalized PageRank for ranking purposes. The dataset consists of a subset of 400×2 randomly drawn 16px$\times$16px images of handwritten fours and nines from the USPS handwritten digits dataset[4].

The data is embedded using *JRC-EMBED-PCK* and ranked using *JRC-RANK-PCK* with randomly drawn query images. The results are shown in Fig. 10.14–Fig. 10.17. The left panels show the ranking results with *JRC-RANK-PCK*. The right panels shows the ranking results using Euclidean distances from the query images. The query images are shown in the top left corner of the panels.

In Fig. 10.14, the query image is a closed four. Thus, it is similar to a nine with straight edges. The Euclidean distance based ranking has 5 closed fours within the top ranked images. However, it also presents 7 nines in the list. Using *JRC-RANK-PCK*, the result is a bit different. Only two closed fours are shown in the list. These are also ranked lower than when using the Euclidean distance. However, the *concept* of a four seems to be better preserved in this list.

The results are not as prominent for the other queries tested. In Fig. 10.15, the query is a four with a curved left edge. This left edge resembles the left edge of a nine. Still, *JRC-RANK-PCK* only finds one nine in the list. This nine is low ranked. The Euclidean distance based method finds 8 nines. For the query in Fig. 10.16, both methods find only fours. For the query in Fig. 10.17, both methods finds a lot of fours, even though the query is a nine. However, this nine seems to have a very prominent slanting right edge. Since the Probabilistic Cluster Kernel is learned from the data, this slanted edge might be different enough from the norm of the concept of nines and fours that the kernel considers images with a similar edge as a group.

[4]http://cs.nyu.edu/~roweis/data.html

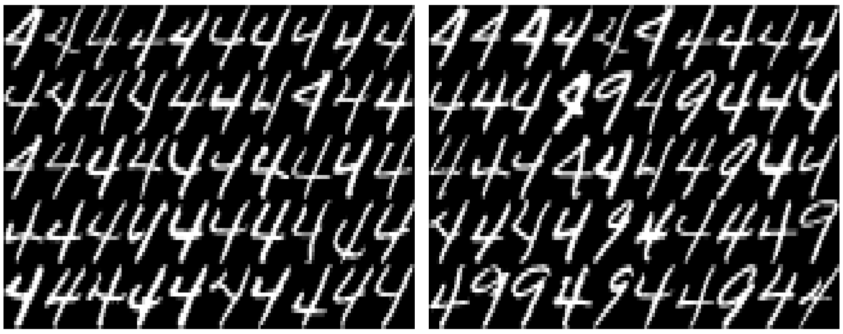


Figure 10.14: Ranked with *JRC-RANK-PCK*(left) and Euclidean distances (right)

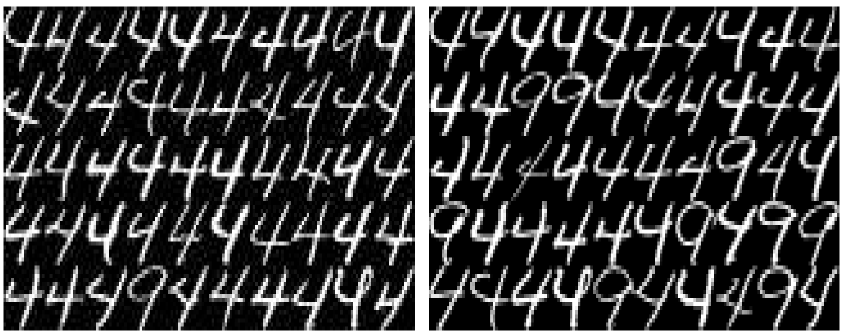


Figure 10.15: Query of a curly four.

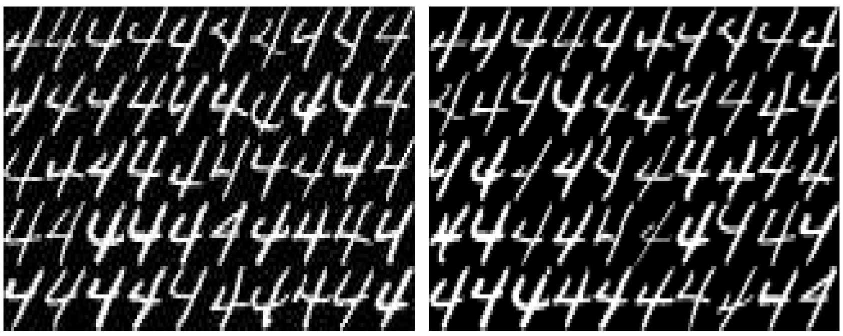


Figure 10.16: Query where both methods succeed.

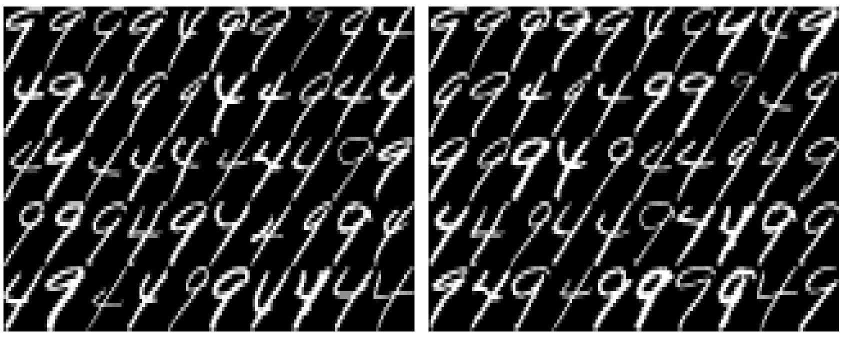


Figure 10.17: Query where both methods fail.

10.5 Barro-Lee Educational Attainment Dataset

The Barro-Lee Educational Attainment Dataset [107] consists of data on education level in countries of the world for different age groups in the period 1950–2010. Within the dataset, it is possible to use a population of either males, females or both. The features in the dataset are listed in Tab. 10.3. In this experiment, the dataset is limited to males and females in the age group of people over 25. The data is from 2010.

The data is embedded using *JRC-EMBED-PCK* and clustered using k-means with $k = 3$. Although other k-values might be appropriate, $k = 3$ is chosen to get a general overview of the educational attainment. The data is ranked using *JRC-RANK-PCK* with a global uniform seed distribution. The embedded data with cluster labels is shown in Fig. 10.18. There is no clear separable cluster structure in the data. Just by looking at this plot, the clustering result is a bit hard to interpret. However, the data points corresponds to countries. A world map containing the clustering and ranking result is shown in Fig. 10.19. Each of the three groups (in general) consists of countries in special regions of the world. The red cluster seems to dominate Africa and spread to South Asia through the Middle East. From Mexico and southwards through South America, the countries are mainly assigned to the blue cluster. The green cluster consists of countries from Europe, Central- and North Asia, North America and Oseania. Note that the data itself has no information about the location of the countries. Still, the result of the clustering is very region specific.

Fig. 10.21 shows a boxplot for the original features in each of the three clusters. The labels on the first axis corresponds to the features as described in Tab. 10.3. The blue and the green cluster has a stable low percentage of people not attending school. The green cluster seems to have a higher level

Table 10.3: Features of the Barro-Lee Educational Attainment Dataset. The percentages are based on the *highest* attained schooling.

Feature	Description
lu	Percentage of no schooling attained in population
lp	Percentage of primary schooling attained in population
lpc	Percentage of completed primary schooling attained in population
ls	Percentage of secondary schooling attained in population
lsc	Percentage of completed primary schooling attained in population
lh	Percentage of tertiary schooling attained in population
lhc	Percentage of completed tertiary schooling attained in population
yr_sch	Average years of schooling attained
yr_sch_pri	Average years of primary schooling attained
yr_sch_sec	Average years of secondary schooling attained
yr_sch_ter	Average years of tertiary schooling attained

of educational attainment than the blue. However, there are some outliers (the red plus signs). These countries are low ranked within the cluster. This is not unexpected as the ranking procedure is based on similarities.

The red cluster has a wide range of percentages in this feature. From 3.9%–81.37% not having any schooling, but with no outliers in this feature. The red cluster also has the widest range of percentages where starting primary school (but not completing it) is the highest level of education achieved. As the level of education increases, the percentage of people achieving this level decreases. Thus, these countries are expected to be poorly educated. Note that some countries which is famous for brilliantly educated people (like India), is within this cluster. Although there are many people which are well educated and well known in the academic world, the country has millions of inhabitants which lives in poverty. Thus, the overall educational attainment is low.

The rank of the countries should not be interpreted as the ones with the "best" education. As this is a relational ranking algorithm, the result depends on the relationship between the data points. In this case, it is based on similarities of educational attainment. The top ranked countries will then be countries which are similar to a lot of countries. These are the big red dots in Africa. This might be because the low level of education in these countries is common around the world. This shows off the point of joint ranking and clustering. Does it make sense for a relational ranking to rank across very different groups? It would probably make more sense to rank within a group of similar data points. Tab. 10.4–Tab. 10.6 shows the top 15 ranked countries within each of the three clusters.

It should be noted that the dataset has been clustered using k-means on the input data. This is shown in Fig. 10.20. Although the result is not very different from the one obtained using *JRC-EMBED-PCK*, there are some subtle differences. For instance, Finland, New Zealand and Iceland are assigned to a different cluster than Norway. The question of this being a correct decision by the algorithm is not easy to answer. However, subtle differences like this might be important when choosing which clustering algorithm to use in a specific application.

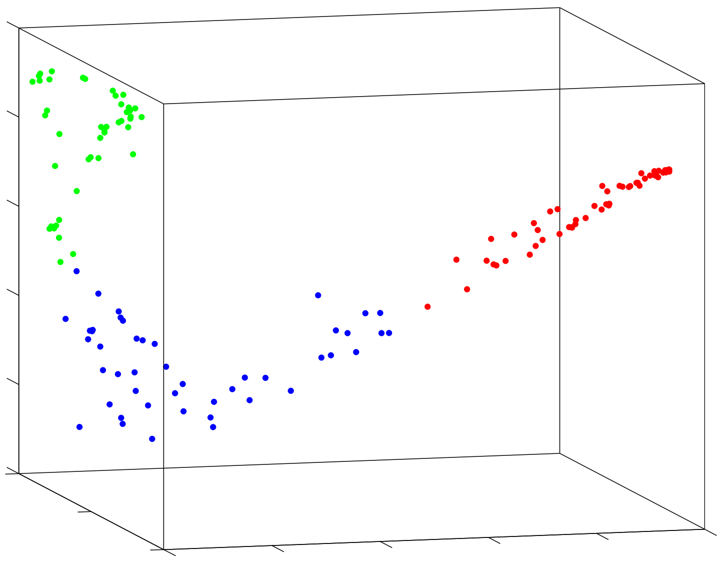


Figure 10.18: Embedded data with cluster labels.

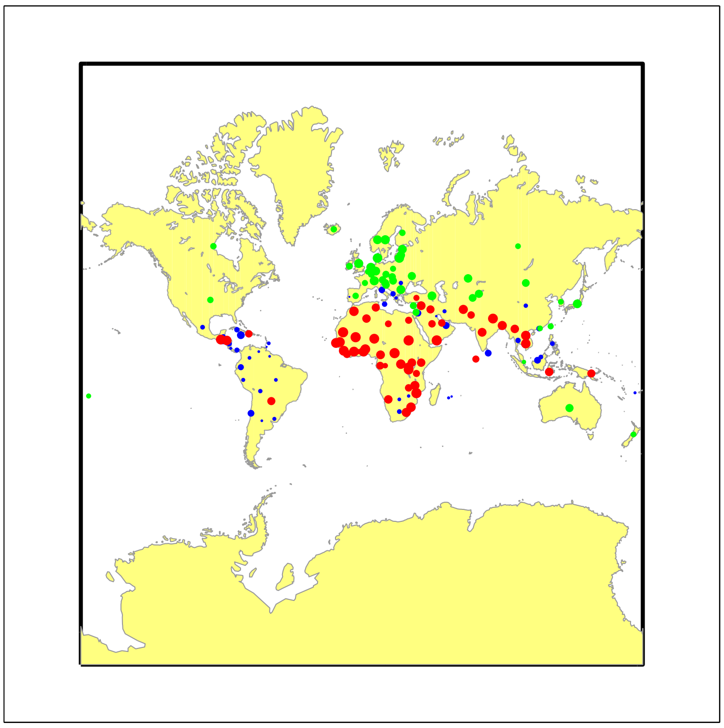


Figure 10.19: World map. The data points represents countries. They are colored according to the clustering labels. The size of the points are set using ranking information. Larger points are ranked higher.

Table 10.4: Top 10 ranked countries in the blue cluster.

Country	Region
Jordan	Middle East and North Africa
Chile	Latin America and the Caribbean
Sri Lanka	South Asia
Malaysia	East Asia and the Pacific
Guyana	Latin America and the Caribbean
China	East Asia and the Pacific
Malta	Middle East and North Africa
Albania	Europe and Central Asia
Cuba	Latin America and the Caribbean
South Africa	Sub-Saharan Africa
⋮	⋮

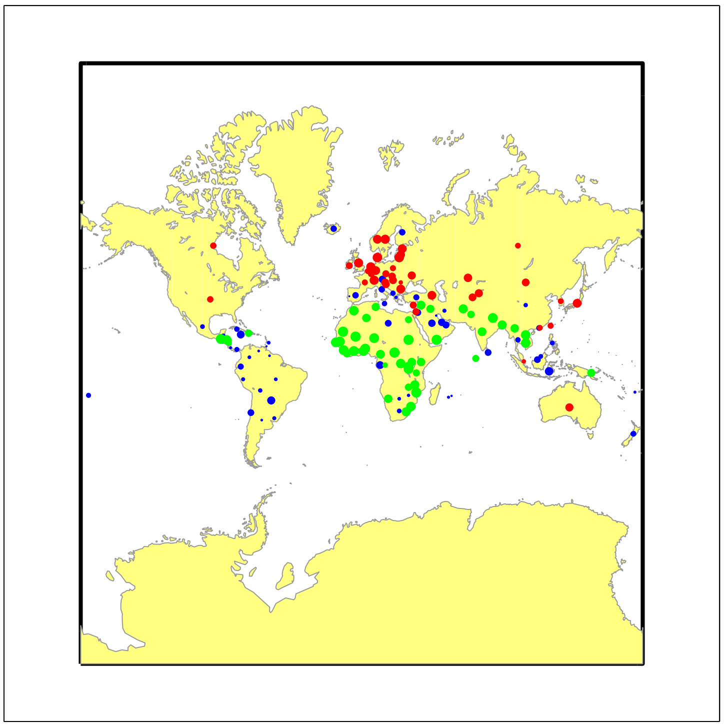
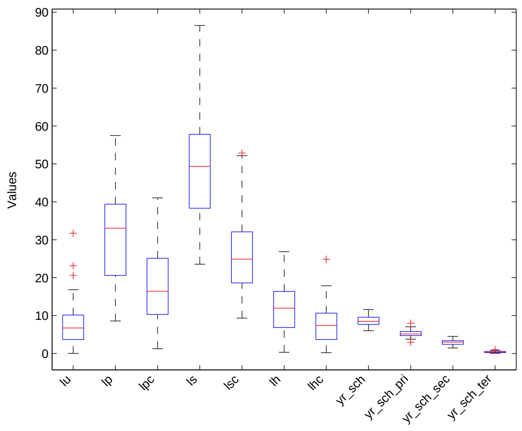


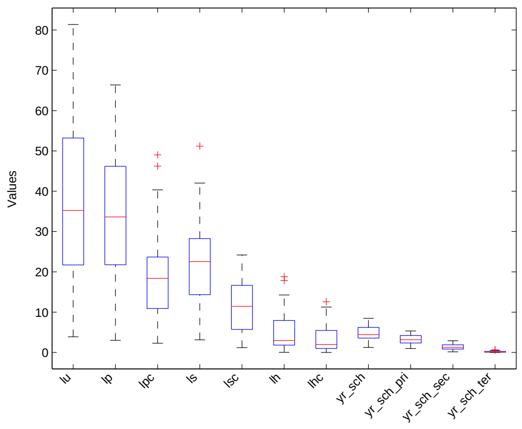
Figure 10.20: Clustering of the countries using k-means on the input data. We see there are subtle differences, like Finland, New Zealand and Iceland.

Table 10.5: Top 10 ranked countries in the red cluster.

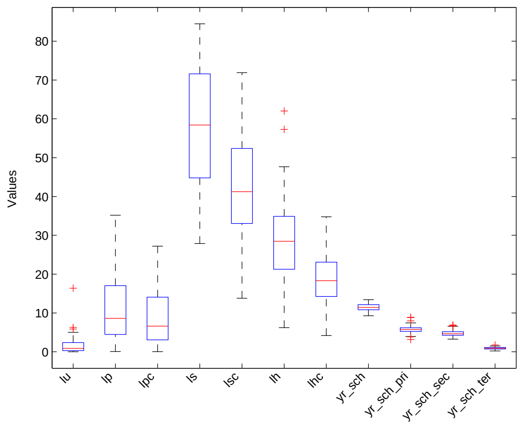
Country	Region
Central African Republic	Sub-Saharan Africa
Mali	Sub-Saharan Africa
Mozambique	Sub-Saharan Africa
Niger	Sub-Saharan Africa
Mauritania	Sub-Saharan Africa
Yemen	Middle East and North Africa
Senegal	Sub-Saharan Africa
Sudan	Sub-Saharan Africa
Burundi	Sub-Saharan Africa
Sierra Leone	Sub-Saharan Africa
⋮	⋮



(a)



(b)



(c)

Figure 10.21: Box plot of the features in (a): The blue cluster. (b): The red cluster. (c): The green cluster.

Table 10.6: Top 10 ranked countries in the green cluster.

Country	Region
Denmark	Advanced Economies
Switzerland	Advanced Economies
Lithuania	Europe and Central Asia
Sweden	Advanced Economies
United Kingdom	Advanced Economies
Netherlands	Advanced Economies
Estonia	Europe and Central Asia
Japan	Advanced Economies
Bulgaria	Europe and Central Asia
Germany	Advanced Economies
⋮	⋮

10.5.1 Ranking with respect to Norway

Using *JRC-RANK-PCK*, it is possible to rank with respect to one specific data point. In this section, the countries are ranked using *JRC-RANK-PCK* with Norway as the seed. One would expect the top ranked countries to be similar to Norway in educational attainment. The top ranked countries are shown in Tab. 10.7. The top 6 results are not unexpected results. Getting Lithuania, Bulgaria and Estonia within the top ranked countries with respect to Norway was unexpected for the author. An OECD report from 2012[5] claims that the attainment levels in Estonia is among the highest among OECD countries. Also, a report from the European Centre for the Development of Vocational Training from 2012[6] states that Lithuania has one of the highest educational attainments in Europe. Although similar reports on Bulgaria has not been found, this indicates that the ranking does make sense.

[5]http://www.oecd.org/estonia/estonia50.pdf

[6]http://libserver.cedefop.europa.eu/vetelib/2012/2012_CR_LT.pdf

Table 10.7: Top 10 ranked countries with respect to Norway.

Country	Region
Denmark	Advanced Economies
Netherlands	Advanced Economies
Sweden	Advanced Economies
United Kingdom	Advanced Economies
Switzerland	Advanced Economies
Japan	Advanced Economies
Lithuania	Europe and Central Asia
Bulgaria	Europe and Central Asia
Estonia	Europe and Central Asia
Germany	Advanced Economies
⋮	⋮

Part IV

Chapter 11

Conclusion

In this thesis, we have proposed a framework for joint ranking and clustering on multi attribute data based on Markov chain theory with transition probabilities learned from the data via the PCK. In the experiments, *all parameters were held fixed*. This suggests that the methods developed does not depend on critical parameters. This is an important point where many other methods fail.

Theoretical results in this thesis include a new connection between the PCK and the consensus clustering methodology. We have also shown that the stationary distribution of a Markov chain with a transition probability matrix on the form $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$ calculates a projection in the empirical kernel space. This has been connected to Mercer kernel theory and nonparametric density estimates.

The experiments in this thesis have mostly been focused on datasets where there is no ground truth data. We have shown that the proposed embedding is able to conserve group structures in the data when combined with the PCK. It has also been suggested that ranking data using the clustered data in this framework enables us to gain a better understanding of the clustering result. Although some ranking results were unexpected, the results were verified by investigating the data further.

Lastly, we have used the embedding in a specific application, namely cloud screening. Here, the PCK found structures in the data which methods in previously published papers did not find. This is one of the benefits of having a function which is learned from the data. Other functions where we have to use the same parameters for the whole input space might miss these structures. In the cloud screening application, the ranking information was used to increase the prediction accuracy. Although it is unknown if this applies to other datasets or other classifiers, using the top ranked data points of the randomly sampled initial training set as a new training set for the classifier gained significantly better results.

11.1 Further work

There are numerous questions with regards to the methods proposed in this thesis. For instance, are there other applications than ranking and clustering for the embedding? In Sec. 10.2.2, it was barely mentioned that low degree data points may be penalized in in terms of distance in the embedding. Experiments have shown that a way of "forcing" this effect is to increase the restart probability α. This seems to have the effect of increasing the distance from the low degree data points to the rest of the data points. Although the embedding has not been tested for this purpose, we suspect this can be used for outlier detection.

The iterative form of the personalized PageRank has been used previously for semi supervised learning [39]. In many semi supervised learning algorithms, the data is embedded using a nonlinear embedding, a linear classifier is trained on the few labels available and the unknown data points are then classified using this classifier [108, 109]. As the embedding proposed in this thesis seems to be able to capture group structures, this approach might be an option for semi supervised learning.

In this thesis, the α parameter was set to $\alpha = 0.15$ by the compelling argument that "Google recommends it". The experiments indicate that this is a good choice. However, we have no solid evidence that this is optimal. Thus, a possible research area could be to investigate if there is such a thing as an optimal α.

In the cloud screening experiment, we carefully suggested that the ranking information could be used to improve the classification result by only including the top ranked data points in the training set. Is this an appropriate approach in other datasets or with other classifiers than the simple 1-NN classifier? Currently, this is an unanswered question which has to be investigated further.

Lastly, there is always the big question of choosing the number of clusters in the clustering algorithm. Many approaches have been proposed in the literature. One interesting approach based on *iterative* Consensus Clustering and Markov chain theory exploits the eigenvalues of a transition probability matrix [110]. This seems to fit well within this framework. Thus, it would be interesting to investigate if this method could be applied.

Appendix A

One-step random walks and the Personalized PageRank

In this chapter, it will be shown that the matrix used to calculate the personalized PageRank, $\mathbf{P}_\beta = \beta\mathbf{D}\mathbf{G}_\beta$, is a left stochastic matrix. The personalized PageRank is calculated by a one-step random walk using this matrix. The stationary distribution of the associated Markov Chain is the same as the stationary distribution of the Markov Chain associated with $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$.

Recall that the β-adjusted Laplacian is defined as $\mathbf{L}_\beta = \beta\mathbf{D} + \mathbf{L}$, where $\mathbf{L} = \mathbf{D} - \mathbf{K}$ is the Laplacian, $\mathbf{D} = \mathrm{diag}(d_i)$ is the degree matrix with the degree defined as $d_i = \sum_{j=1}^{N} k_{ij}$. The β-normalized Laplacian is defined as $\boldsymbol{\mathcal{L}}_\beta = \beta\mathbf{I} + \boldsymbol{\mathcal{L}}$, where $\boldsymbol{\mathcal{L}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized Laplacian. The Green's function $\mathbf{G}_\beta$ of $\mathbf{L}_\beta$ satisfies $\mathbf{G}_\beta\mathbf{L}_\beta = \mathbf{L}_\beta\mathbf{G}_\beta = \mathbf{I}$. If $\boldsymbol{\mathcal{G}}_\beta$ is the Green's function of $\boldsymbol{\mathcal{L}}_\beta$ then

$$\begin{aligned} \boldsymbol{\mathcal{G}}_\beta\boldsymbol{\mathcal{L}}_\beta &= \mathbf{I} \\ \boldsymbol{\mathcal{G}}_\beta\left(\beta\mathbf{I} + \boldsymbol{\mathcal{L}}\right) &= \mathbf{I} \\ \boldsymbol{\mathcal{G}}_\beta\mathbf{D}^{-\frac{1}{2}}\left(\beta\mathbf{D} + \mathbf{L}\right)\mathbf{D}^{-\frac{1}{2}} &= \mathbf{I} \\ \mathbf{D}^{-\frac{1}{2}}\boldsymbol{\mathcal{G}}_\beta\mathbf{D}^{-\frac{1}{2}}\mathbf{L}_\beta &= \mathbf{I}. \end{aligned} \tag{A.1}$$

The Green's function (or inverse) is unique. Thus, $\mathbf{G}_\beta = \mathbf{D}^{-\frac{1}{2}}\boldsymbol{\mathcal{G}}_\beta\mathbf{D}^{-\frac{1}{2}}$ is the Green's function for $\mathbf{L}_\beta$.

Let $\boldsymbol{\pi} = \frac{1}{\mathrm{Vol}(G)}\mathbf{D}\mathbb{1}$ be the stationary distribution of the Markov Chain associated with the transition probability matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$. Then

$$\begin{aligned}
\mathbf{P}_\beta \boldsymbol{\pi} &= \beta \mathbf{D}\mathbf{G}_\beta \boldsymbol{\pi} \\
&= \frac{\beta}{\mathrm{Vol}(G)} \mathbf{D}\mathbf{G}_\beta \mathbf{D}\mathbb{1} \\
&= \frac{\beta}{\mathrm{Vol}(G)} \mathbf{D}\mathbf{D}^{-\frac{1}{2}} \boldsymbol{\mathcal{G}}_\beta \mathbf{D}^{-\frac{1}{2}} \mathbf{D}\mathbb{1} \\
&= \frac{\beta}{\mathrm{Vol}(G)} \mathbf{D}^{\frac{1}{2}} \boldsymbol{\mathcal{G}}_\beta (\mathbf{D}^{\frac{1}{2}}\mathbb{1}).
\end{aligned}$$

It is known that $\mathbf{D}^{\frac{1}{2}}\mathbb{1}$ is an eigenvector of $\boldsymbol{\mathcal{L}}$ associated with the eigenvalue 0. Since $\boldsymbol{\mathcal{L}}_\beta = \beta\mathbf{I} + \boldsymbol{\mathcal{L}}$ has the same eigenvectors as $\boldsymbol{\mathcal{L}}$, but with β shifted eigenvalues, $\mathbf{D}^{\frac{1}{2}}\mathbb{1}$ is an eigenvector of $\boldsymbol{\mathcal{L}}_\beta$ with the associated eigenvalue β. Thus, $\mathbf{D}^{\frac{1}{2}}\mathbb{1}$ is an eigenvector of $\boldsymbol{\mathcal{G}}_\beta$ with the associated eigenvalue $\frac{1}{\beta}$. This yields

$$\begin{aligned}
\mathbf{P}_\beta \boldsymbol{\pi} &= \frac{\beta}{\mathrm{Vol}(G)} \mathbf{D}^{\frac{1}{2}} \frac{1}{\beta} \mathbf{D}^{\frac{1}{2}}\mathbb{1} \\
&= \frac{1}{\mathrm{Vol}(G)} \mathbf{D}\mathbb{1} \\
&= \boldsymbol{\pi}.
\end{aligned}$$

This result implies that $\boldsymbol{\pi}$ is a right eigenvector of $\mathbf{P}_\beta$ with the associated eigenvalue 1. Furthermore,

$$\begin{aligned}
\mathbb{1}^T \mathbf{P}_\beta &= \mathbb{1}^T \beta \mathbf{D}\mathbf{G}_\beta \\
&= \mathbb{1}^T \beta \mathbf{D}\mathbf{D}^{-\frac{1}{2}} \boldsymbol{\mathcal{G}}_\beta \mathbf{D}^{-\frac{1}{2}} \\
&= \mathbb{1}^T \beta \mathbf{D}^{\frac{1}{2}} \boldsymbol{\mathcal{G}}_\beta \mathbf{D}^{-\frac{1}{2}} \\
&= \beta (\mathbf{D}^{\frac{1}{2}}\mathbb{1})^T \boldsymbol{\mathcal{G}}_\beta \mathbf{D}^{-\frac{1}{2}}.
\end{aligned}$$

By the same arguments as earlier,

$$(\mathbf{D}^{\frac{1}{2}}\mathbb{1})^T \boldsymbol{\mathcal{G}}_\beta = \frac{1}{\beta} (\mathbf{D}^{\frac{1}{2}}\mathbb{1})^T,$$

since $\boldsymbol{\mathcal{G}}_\beta$ is symmetric. This yields

$$\begin{aligned}
\mathbb{1}^T \mathbf{P}_\beta &= \mathbb{1}^T \beta \mathbf{D}\mathbf{G}_\beta \\
&= \beta \frac{1}{\beta} (\mathbf{D}^{\frac{1}{2}}\mathbb{1})^T \mathbf{D}^{-\frac{1}{2}} \\
&= \mathbb{1}^T.
\end{aligned}$$

Thus, $\mathbb{1}^T$ is a left eigenvector of $\mathbf{P}_\beta$ with the associated eigenvalue 1. These two results are suspiciously similar to properties of a left stochastic matrix. To show that $\mathbf{P}_\beta$ is indeed a left stochastic matrix, we need to show that $\mathbf{G}_\beta$ has non-negative elements. The β-adjusted Laplacian $\mathbf{L}_\beta$ is a non-singular M-matrix [111, Def. 1.1]. By [111, Thm. 2.2] its inverse (i.e. $\mathbf{G}_\beta$) has non-negative elements. Thus $\mathbf{P}_\beta = \beta\mathbf{D}\mathbf{G}_\beta$ has non-negative elements and each column sums to 1. So $\mathbf{P}_\beta$ is a left stochastic matrix. It is easily verified that this Markov Chain is time reversible with $\boldsymbol{\pi}$ as its reversibility distribution. Thus, the associated Markov Chain has the same stationary distribution as a random walk on the Markov Chain associated with $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$. Furthermore, we recognize from Eq. (8.5) that the personalized PageRank can be interpreted as a one-step random walk on the Markov Chain associated with $\mathbf{P}_\beta$ with the initial distribution $\mathbf{s}$.

Bibliography

[1] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, Apr. 1998.

[2] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web." Tech. Rep., 1999.

[3] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, Sep. 1999.

[4] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," *2008 Eighth IEEE International Conference on Data Mining*, pp. 263–272, Dec. 2008.

[5] Y. Jing and S. Baluja, "VisualRank: applying PageRank to large-scale image search." *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 11, pp. 1877–90, Nov. 2008.

[6] X. He, W.-Y. Ma, and H. Zhang, "ImageRank: Spectral Techniques for Structural Analysis of Image Database," in *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, 2003, pp. 25–28.

[7] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," *Association for Computational Linguistics*, 2004.

[8] R. Mihalcea, "Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization," *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, 2004.

[9] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," *Proceedings of the 22nd international conference on Machine Learning*, pp. 89–96, 2005.

[10] T. Joachims, "Optimizing search engines using clickthrough data," *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*, pp. 133–142, 2002.

[11] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An Efficient Boosting Algorithm for Combining Preferences," *The Journal of Machine Learning Research*, vol. 4, pp. 933–969, 2003.

[12] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf, "Ranking on Data Manifolds," *Advances in Neural Information Processing Systems*, vol. 16, pp. 169–176, 2004.

[13] X.-Q. Cheng, P. Du, J. Guo, X. Zhu, and Y. Chen, "Ranking on Data Manifold with Sink Points," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 177–191, 2013.

[14] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 450–465, 1999.

[15] A. K. Jain and P. J. Flynn, "Image segmentation using clustering," *Advances in Image Understanding, IEEE Computer Society Press*, pp. 65–83, 1996.

[16] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[17] M. Iwayama and T. Tokunaga, "Cluster-based Text Categorization: A Comparison of Category Search Strategies," in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '95. ACM, 1995, pp. 273–280.

[18] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," 2000.

[19] O. Zamir and O. Etzioni, "Web document clustering: A feasibility demonstration," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 1998, pp. 46–54.

[20] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences*, vol. 96, no. 12, pp. 6745–6750, 1999.

[21] B.-D. Amir, R. Shamir, and Z. Yakhini, "Clustering Gene Expression Patterns," *Journal of Computational Biology*, vol. 6, no. 3–4, pp. 281–297, 1999.

[22] P. Baldi and G. W. Hatfield, *DNA Microarray and Gene Expression.* Cambridge University Press, 2002.

[23] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, pp. 14 863–14 868, 1998.

[24] G. Arimond and A. Elfessi, "A clustering method for categorical data in tourism market segmentation research," *Journal of Travel Research*, vol. 39, no. 4, pp. 391–397, 2001.

[25] H. Hruschka, "Market definition and segmentation using fuzzy clustering methods," *International Journal of Research in Marketing*, vol. 3, no. 2, pp. 117–134, 1986.

[26] E. Izquierdo-Verdiguier, R. Jenssen, L. Gómez-Chova, and G. Camps-Valls, "Spectral clustering with the probabilistic cluster kernel," *Neurocomputing*, Sep. 2014.

[27] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu, "RankClus: Integrating Clustering with Ranking for Heterogeneous Information Network Analysis," *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pp. 565–576, 2009.

[28] Y. Sun, Y. Yu, and J. Han, "Ranking-Based Clustering of Heterogeneous Information Networks with Star Network Schema Categories and Subject Descriptors," *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 797–806, 2009.

[29] L. Cao, X. Jin, Z. Yin, A. Del Pozo, J. Luo, J. Han, and T. S. Huang, "RankCompete: Simultaneous ranking and clustering of information networks," *Neurocomputing*, vol. 95, pp. 98–104, Oct. 2012.

[30] M. Gales and S. Young, "The Application of Hidden Markov Models in Speech Recognition," *Foundations and Trends in Signal Processing*, vol. 1, no. 3, pp. 195–304, 2008.

[31] M.-Y. Chen, A. Kundu, and J. Zhou, "Off-Line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 481–496, May 1994.

[32] J. Hu, M. K. Brown, and W. Turin, "HMM Based On-Line Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 1039–1045, 1996.

[33] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning : A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[34] S. M. Ross, *Introduction to Probability Models*, 10th ed. Elsevier Inc., 2010.

[35] H. Li, "A Short Introduction to Learning to Rank," *IEICE Transactions on Information and Systems*, vol. 94, no. 10, pp. 1854–1862, 2011.

[36] T.-Y. Liu, "Learning to Rank for Information Retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[37] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond.* Princeton University Press, 2006.

[38] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. The John Hopkins University Press, 1996, vol. 208-209.

[39] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with Local and Global Consistency," *Advances in Neural Information Processing Systems*, vol. 16, no. 16, pp. 321–328, 2004.

[40] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, Sep. 1933.

[41] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, 6th ed., R. A. Johnson and D. W. Wichern, Eds. Pearson Education Limited, 2013.

[42] S. V. Vaseghl, *Multimedia Signal Processing: Theory and Applications in Speech, Music and Communications.* John Wiley & Sons, Ltd., 2008.

[43] N. H. Timm, *Applied Multivariate Analysis.* Springer, 2002.

[44] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," Max-Planck-Institut für biologische Kybernetik, Tech. Rep. 44, 1996.

[45] ——, "Kernel principal component analysis," in *Artificial Neural Networks - ICANN'97*, ser. Lecture Notes in Computer Science, 1997, vol. 1327, pp. 583–588.

[46] ——, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[47] K. I. Kim, K. Jung, and H. J. Kim, "Face recognition using kernel principal component analysis," *Signal Processing Letters, IEEE*, vol. 9, no. 2, pp. 40–42, Feb. 2002.

[48] M.-H. Yang, N. Ahuja, and D. Kriegman, "Face recognition using kernel eigenfaces," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 1, 2000, pp. 37–40.

[49] A. M. Jade, B. Srikanth, V. K. Jayaraman, B. D. Kulkarni, J. P. Jog, and L. Priya, "Feature extraction and denoising using kernel {PCA}," *Chemical Engineering Science*, vol. 58, no. 19, pp. 4441–4448, 2003.

[50] T. Takahashi and T. Kurita, "Robust De-noising by Kernel PCA," in *Artificial Neural Networks – ICANN 2002*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2415, pp. 739–744.

[51] K. I. Kim, S. H. Park, and H. J. Kim, "Kernel principal component analysis for texture classification," *Signal Processing Letters, IEEE*, vol. 8, no. 2, pp. 39–41, Feb. 2001.

[52] J. Karhunen and J. Joutsensalo, "Representation and separation of signals using nonlinear {PCA} type learning," *Neural Networks*, vol. 7, no. 1, pp. 113–127, 1994.

[53] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[54] N. Lawrence, “Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models,” *J. Mach. Learn. Res.*, vol. 6, pp. 1783–1816, 2005.

[55] M. Scholz and R. Vigário, “Nonlinear PCA: a new hierarchical approach.” in *ESANN*, 2002, pp. 439–444.

[56] N. Aronszajn, “Theory of Reproducing Kernels,” *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

[57] J. Mercer, “Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations,” *Philosophical Transactions of The Royal Society*, vol. A, no. 209, pp. 415–446, 1909.

[58] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*, 4th ed. Academic Press, 2008.

[59] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the fifth annual workshop on Computational learning theory.* ACM, 1992, pp. 144–152.

[60] C. Cortes and V. N. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[61] Y.-Q. Zhang and J. C. Rajapakse, *Machine Learning in Bioinformatics.* John Wiley & Sons, Inc., 2009.

[62] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis.* Cambridge university press, 2004.

[63] P. Honeine, “An eigenanalysis of data centering in machine learning,” p. 14, Jul. 2014. [Online]. Available: http://arxiv.org/abs/1407.2904

[64] R. Jenssen, “Kernel Entropy Component Analysis,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 5, pp. 847–860, 2010.

[65] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, “A survey of kernel and spectral methods for clustering,” *Pattern Recognition*, vol. 41, no. 1, pp. 176–190, 2008.

[66] A. K. Jain, “50 years beyond K-means,” *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[67] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data Clustering: A Review,” *ACM Computing Surveys*, vol. 31, no. 3, pp. 265–323, Sep. 1999.

[68] U. von Luxburg, “A tutorial on spectral clustering,” *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[69] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 281–297, 1967, p. 14.

[70] S. Z. Selim and M. A. Ismail, “K-Means-Type Algorithm: A Generalized Convergence Theorem and Characterization of Local Optimality,” *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 1, pp. 81–87, 1984.

[71] A. R. Webb and K. D. Copsey, *Statistical Pattern Recognition.* John Wiley & Sons, Ltd., 2011.

[72] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[73] G. Casella and R. L. Berger, *Statistical Inference International Edition*, 2nd ed. Brooks/Cole, 2008.

[74] E. L. Lehmann and G. Casella, *Theory of Point Estimation*, 2nd ed., G. Casella, S. Fienberg, and I. Olkin, Eds. Springer, 1998.

[75] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, 2nd ed. Springer, 2004.

[76] K. B. Petersen and M. S. Pedersen, “The Matrix Cookbook,” *Matrix*, vol. M, no. 1, pp. 1–71, 2008.

[77] M. Pourahmadi, *High-Dimensional Covariance Estimation.* John Wiley & Sons, Inc., 2013.

[78] M. Meila and J. Shi, “A Random Walks View of Spectral Segmentation,” in *A random walks view of spectral segmentation.*, 2001.

[79] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems*, pp. 849–856, 2001.

[80] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[81] Y. Bengio, P. Vincent, J.-F. Paiement, O. Delalleau, M. Ouimet, and N. L. Roux, "Spectral Clustering and Kernel PCA are Learning Eigenfunctions," Tech. Rep., 2003.

[82] Y. Bengio, O. Delalleau, N. L. Roux, J.-F. Paiement, P. Vincent, and M. Ouimet, "Learning Eigenfunctions Links Spectral Embedding and Kernel PCA," *Neural Computation*, vol. 16, no. 10, pp. 2197–2219, 2004.

[83] A. L. N. Fred and A. K. Jain, "Data clustering using evidence accumulation," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4. IEEE, 2002, pp. 276–280.

[84] ——, "Combining multiple clusterings using evidence accumulation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 6, pp. 835–850, 2005.

[85] A. Strehl and J. Ghosh, "Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[86] S. Monti, P. Tamayo, J. Mesirov, and T. Golub, "Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data," *Machine Learning*, vol. 52, no. 1-2, pp. 91–118, 2003.

[87] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap.* Chapman and Hall/CRC, 1994.

[88] P. Hore, L. Hall, and D. Goldgof, "A cluster ensemble framework for large data sets," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, vol. 4. IEEE, 2006, pp. 3342–3347.

[89] P. Hore, L. O. Hall, and D. B. Goldgof, "A scalable framework for cluster ensembles," *Pattern recognition*, vol. 42, no. 5, pp. 676–688, 2009.

[90] M. C. V. Nascimento, F. M. B. de Toledo, and A. C. P. L. F. Carvalho, "Consensus Clustering Using Spectral Theory," in *Advances in Neuro-Information Processing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5506, pp. 461–468.

[91] C. Meyer and C. Wessell, "Stochastic Data Clustering," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1214–1236, 2012.

[92] J. Weston, C. Leslie, E. Ie, D. Zhou, A. Elisseeff, and W. S. Noble, "Semi-supervised protein classification using cluster kernels." *Bioinformatics (Oxford, England)*, vol. 21, no. 15, pp. 3241–7, Aug. 2005.

[93] D. Tuia and G. Camps-Valls, "Semisupervised Remote Sensing Image Classification With Cluster Kernels," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 2, pp. 224–228, Apr. 2009.

[94] E. Izquierdo-Verdiguier, L. Gomez-Chova, L. Bruzzone, and G. Camps-Valls, "Semisupervised nonlinear feature extraction for image classification," *2012 IEEE International Geoscience and Remote Sensing Symposium*, pp. 1525–1528, Jul. 2012.

[95] H. J. Landau and A. M. Odlyzko, "Bounds for Elgenvalues of Certain Stochastic Matrlces," *Linear Algebra and its Applications*, vol. 38, pp. 5–15, 1981.

[96] E. Parzen, "On Estimation of a Probability Density Function and Mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[97] R. Jenssen, J. C. Principe, D. Erdogmus, and T. Eltoft, "The Cauchy—Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels," *Journal of the Franklin Institute*, vol. 343, no. 6, pp. 614–629, Sep. 2006.

[98] F. Chung and W. Zhao, "Pagerank and random walks on graphs," *Fete of combinatorics and computer science*, pp. 1–16, 2010.

[99] F. Chung and S.-T. Yau, "Discrete Green's Functions," *Journal of Combinatorial Theory, Series A*, vol. 91, no. 1-2, pp. 191–214, Jul. 2000.

[100] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks.* Washington: Mathematical Association of America, Jan. 1984, vol. 10.

[101] P. Sarkar and G. J. Gordon, "Random Walks with Random Projections," *Workshop on Analyzing Networks and Learning with Graphs NIPS 2009*, p. 7, 2009.

[102] L. Gómez-Chova, G. Camps-Valls, J. Calpe-Maravilla, L. Guanter, and J. Moreno, "Cloud-Screening Algorithm for ENVISAT / MERIS Multispectral Images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 4105–4118, 2007.

[103] L. Gómez-Chova, R. Jenssen, and G. Camps-Valls, "Kernel Entropy Component Analysis for Remote Sensing Image Clustering," *Geoscience and Remote Sensing Letters, IEEE*, vol. 9, no. 2, pp. 312–316, 2011.

[104] V. V. Vikjord and R. Jenssen, "Information theoretic clustering using a k-nearest neighbors approach," *Pattern Recognition*, vol. 47, no. 9, pp. 3070–3081, Sep. 2014.

[105] M. Berry, S. T. Dumais, and G. W. O'Brien, "Using Linear Algebra for Intelligent Information Retrieval," *SIAM Review*, vol. 37, no. 4, pp. 573–595, 1995.

[106] "NIPS Web Page." [Online]. Available: http://nips.cc/

[107] R. J. Barro and J. W. Lee, "A new data set of educational attainment in the world, 1950–2010," *Journal of Development Economics*, vol. 104, pp. 184–198, Sep. 2013.

[108] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster Kernels for Semi-Supervised Learning," *Advances in Neural Information Processing Systems*, pp. 585–592, 2002.

[109] J. N. Myhre and R. Jenssen, "Mixture weight influence on Kernel Entropy Component Analysis and Semi-Supervised Learning using the Lasso," *Machine Learning for Signal Processing (MLSP), 2012 IEEE International Workshop on*, p. 6, 2012.

[110] C. D. Meyer, S. Race, and K. Valakuzhy, "Determining the Number of Clusters via Iterative Consensus Clustering," in *SDM*. SIAM, 2013, pp. 94–102.

[111] J. M. Peña, "M-Matrices Whose Inverses Are Totally Positive," *Linear Algebra and its Applications*, vol. 221, pp. 189–193, May 1995.