BSc in Mathematics

Clustering Algorithms

Clustering Algoritmer

Statistical learning

Jonatan Hauge Steffensen 201905639

Supervised by Lars Nørvang Andersen

June 15, 2022

Abstract

In this project we state and examine three different clustering algorithms; the K-means algorithm, the EM-algorithm for Gaussian mixtures and spectral clustering including two different spectral clustering algorithms. All three algorithms are often used in the field of unsupervised machine learning and for exploratory data analysis. In the study we show that the K-means algorithm converges to a local minimum and find that the K-means algorithm is the limit of a special case of the general EM-algorithm applied to a Gaussian mixture. Before doing so, we introduce the EM-algorithm for Gaussian mixtures in terms maximum likelihood theory and by considering the group assignment as latent stochastic variables.

We introduce a normalized and an unnormalized spectral clustering algorithm through the notion of similarity graphs and properties of three different graph Laplacians. Furthermore, we show that both algorithms find approximate solutions to discrete minimization problems involving the *mincut* of the similarity graph.

Lastly, we define the adjusted Rand index, which we can use to compare two different clusterings. Using this, we compare the algorithms on both a simulated data set and on pictures of handwritten digits from the MNIST database, where we in both cases can compare with the true labels. The K-means algorithm produces good results in most cases and has the clear advantage of working fast even on large data sets. The EM-algorithm for Gaussian mixtures works well if the assumption of data coming from a Gaussian mixture is fair, but on the more complex MNIST data the results of the EM-algorithm are more or less the same as the results of the K-means algorithm. Spectral clustering has many intriguing properties from a theoretical perspective and is able to find more complex structures in data sets such as non-convex clusters. However, even though it in many cases outperforms both the K-means algorithm and the EM-algorithm, the results on the MNIST data show that this is not always the case.

Contents

1 Introduction

In a world where more and more data are being collected the need for good exploratory data analysis tools is becoming increasingly important. A common thing to search for in a data set is groups or *clusters* of similar data points within the data set. It could for instance be a marketing bureau wanting to identify similar consumers to target with a specific add or a medical research team trying to find patterns in the biopsies of lung cancer patients. In this study we define three different clustering algorithms and examine their clustering properties. To do this formally, we need to specify what it means for data points to be similar and as we will see throughout the project, there are several different ways to consider the notion of similarity.
As a starting point, we examine the K-means algorithm that uses euclidean distance as a measure of similarity. The K-means algorithm is both in mathematical complexity and in computational complexity the simplest of the three clustering algorithms we examine, but nevertheless it is a widely used method for many clustering purposes. Furthermore, it also serves as a build up to the more general EM-algorithm (Expectation-Maximization) for Gaussian mixtures, which we will examine in chapter 3.
The EM-algorithm for Gaussian mixtures is a special case of the general EM-algorithm with an assumption of data coming from a normal distribution. To define the EM-algorithm for Gaussian mixtures we formulate the model in terms of a stochastic latent group variable and use this to find the maximum likelihood estimates of the Gaussian mixture model. By using the notion of latent variables, the EM-algorithm can be used as a clustering algorithm when applied to a Gaussian mixture model. Lastly, we show that the K-means algorithm can be derived as the limit of a special case of the EM-algorithm for Gaussian mixtures.
In chapter 4, we examine the method of spectral clustering, a method with roots in the field of spectral graph theory. We define a similarity graph, different graph Laplacians and prove some basic properties of graph Laplacians. From this we state two different spectral clustering algorithms, prove the Courant-Fisher theorem and use this to justify the use of spectral clustering from different perspectives involving graph cut problems and random walks on a graph.
All three clustering algorithms are examples of algorithms for unsupervised machine learning, which in general can be described as trying to learn patterns or retrieve information about unlabelled data. Thus, there is no obvious way of comparing the performance of the methods with each other since we in general do not have a set of *true* labels. In chapter 5, we therefore introduce the Rand index and the adjusted Rand index as a way of comparing two different clusterings of a data set. Using this we compare the performance of the algorithms on the MNIST data set, which is a labelled data set and thereby allows us directly to compare the performance to the true labels through the adjusted Rand index.

For consistency in notation we denote data sets $X_1, X_2, ..., X_n \in \mathbb{R}^p$ with capital letters in all three algorithms, even though the only time we consider random variables is in the EM-algorithm for Gaussian mixtures.

All figures presented throughout the project are made using algorithms that are coded from scratch in Python except the figures presented in section 5.3, where the algorithms from the Python module *scikit-learn* are used due to the size of the data set presented in that section.

2 K-means clustering

In this chapter we take a look at the K-means clustering algorithm, one of the most widely used clustering algorithms. As we will see shortly K-means is an elegant and intuitive way of clustering data points into K different groups. In order to use the K-means algorithm one first has to specify the number of clusters, K, in which we want to partition the data set to. The notation and structure of this section mostly follows chapter 9.1 of [Bis06].

2.1 Setup

Given a data set $X_1, X_2, ..., X_n \in \mathbb{R}^p$ of n observations each with p variables or *features* the aim of the algorithm is to partition the data set into K groups each represented by a *centroid*, $\mu_k \in \mathbb{R}^p$ for $k \in \{1, 2, ..., K\}$. The choice of μ_k as representative of each group is, as we will see later, due to the fact that μ_k actually is the mean of the data points assigned to group k. Rather intuitively, the aim is now to find a set of centroids $\{\mu_1,\ \mu_2,\ ...,\ \mu_K\}$ and an assignment of each data point to a centroid such that the sum of squared Euclidean distances from each point to its associated centroid is minimized.

In order to lighten the notation it is convenient to introduce some notation to describe the assignment of a data point to a specific cluster. For X_i with $i = 1, 2, ...n$, let $r_{ik} \in \{0,\ 1\}$ denote the assignment of X_i such that

$$r_{ik} = \begin{cases} 1, \text{ if } X_i \text{ is assigned to cluster } k \\ 0, \text{ else,} \end{cases}$$

which is also known as the one-in-K encoding. Using this we can define the objective function that we wish to minimize, sometimes denoted the *distortion measure* [Bis06], as

Definition 2.1 (*Distortion measure*)**.** For data points $X_1,\ X_2, ...,\ X_n \in \mathbb{R}^p$, centroids $\{\mu_1,\ \mu_2, ..., \mu_K\}$ and an assignment of the data points $\{r_{ik}\}_{i=1,...,n,\ j=1,...K}$, the distortion measure J is defined as

$$J = \sum_{i=1}^{n} \sum_{j=1}^{K} r_{ij} \cdot ||X_i - \mu_j||^2.$$

Notice that the distortion measure is indeed the sum of squared distances from each point to its assigned centroid. We can minimize J through an iterative process where we successively minimize first over the classifications i.e. $\{r_{ik}\}_{i=1,...,n,\ j=1,...K}$, keeping the centroids $\{\mu_1,\ \mu_2, ..., \mu_K\}$ fixed and then over the centroids keeping the (new) classifications fixed. We will refer to the first step in this minimization as the E step (Expectation) and to the second as the M step (Maximization), because as we show in the next section the K-means algorithm can be derived as a special case of the so-called EM-algorithm for Gaussian mixture models.

The E step is fairly straightforward since J is a linear function in r_{ik} and since each of the n terms in the first sum is functionally independent. Thus, the minimization can be done by for each $i \in \{1, 2, ...n\}$ choosing the $k \in \{1, 2, ..., K\}$ that minimizes $||X_i - \mu_k||^2$, which correspond to choosing the closest centroid for each data point. Formally written that is

$$r_{ik} = \begin{cases} 1, \text{ if } k = argmin_k\ ||X_i - \mu_k||^2 \\ 0, \text{ else.} \end{cases} \tag{2.1}$$

The M step is done by minimizing J over the centroids $\{\mu_1, ..., \mu_K\}$ keeping the classifications fixed. The minimum with respect to μ_k can be found using standard rules for derivatives of vectors [PP12].

$$\frac{\partial}{\partial \mu_k}\ J = \frac{\partial}{\partial \mu_k}\ \sum_{i=1}^{n} \sum_{j=1}^{K} r_{ij} \left(X_i - \mu_j\right)^T \left(X_i - \mu_j\right) = \sum_{i=1}^{n} 2 r_{ik} \left(X_i - \mu_k\right).$$

To find the minimum, we set this equal to zero and find that

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik} X_i}{\sum_{i=1}^{n} r_{ik}}, \tag{2.2}$$

minimizes J when we are keeping the r_{ik}'s fixed.
A closer look at this last expression reveals that this actually corresponds to the mean of the points, that the algorithm at this step assigns to group k. The numerator is a sum over all data points, but only the ones assigned to cluster k are summed due to the r_{ik}'s. The denominator on the other hand is the total number of points assigned to cluster k, thus the μ_k's can be found as the mean of points in the k^{th} cluster, hence the name K-means clustering.

This iterative process of the E and the M step is exactly the K-means algorithm or more formally: the K-means algorithm repeats this process until the classifications do not change between iterations or until J changes less than a certain threshold over an iteration. Finally, the algorithm can also be set to run a fixed number of iterations if the data set is to large to wait for convergence. The convergence of J is equivalent to convergence in the classifications $\{r_{ik}\}_{i=1,...,n,\ j=1,...K}$, since a new classification leads to a decrease in J and the other way around. Since each of the finite number of distinct classifications has a unique minimum of J with respect to $\{\mu_1, ..., \mu_K\}$ and since changes in classification result in a lower value of J, the algorithm will converge after a finite number of iterations. However, we are not sure that J will converge to a global minimum [Bis06]. For instance in figure 1 below we see a toy example of convergence to a local minimum rather than a global one. Even though this example is tailored to illustrate this exact point, we can not exclude the possibility of this happening when using the K-means algorithm on real-life data sets, but by using a random initialization of the centroids $\{\mu_k\}_{j=1,2,...K}$, we can minimize the risk of convergence to a local minimum. Furthermore, it is custom to run the algorithm several times each with a different random initialization and then choosing the best result i.e. the one that minimizes the distortion measure J [GT21].

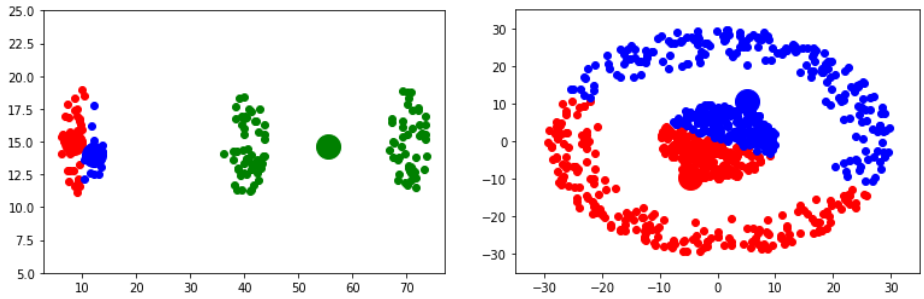

Figure 1: Classifications by the K-means algorithm and centroids. Left: toy example of convergence to a local minimum. Right: example of non-convex clusters that the K-means algorithm cannot detect.

2.2 The K-means algorithm

Letting $C_k = \{X_i \mid 1 \leq i \leq n,\ r_{ik} = 1\}$ denote the set of points assigned to cluster k, the K-means algorithm can be stated as follows:

Algorithm 1 K-means algorithm

Input: $X_1, X_2, ..., X_n \in \mathbb{R}^p$, $K \in \mathbb{N}$ (`Number of desired clusters`)

1: `Initialize` $\mu_j \in \mathbb{R}^p$ `and` $r_{ij} \in \{0, 1\}$ `randomly for` $j = 1, 2, ..., K$, $i = 1, 2, ..., n$, `such that` $\sum_{j=1}^{K} r_{ij} = 1$
2: `E-step: Update the` r_{ij}`'a as in (2.1)` keeping the μ_j's fixed
3: `M-step: Update the` μ_j`'a as in (2.2)` keeping the (new) r_{ij}'s fixed
4: `Repeat step 2 and 3 until there is no change in classification i.e. the` r_{ij}`'s stay unchanged`
5: `Finally, let` $C_j = \{X_i \mid 1 \leq i \leq n,\ r_{ij} = 1\}$ `for` $j = 1, 2, ...K$

Output: Clusters $C_1, C_2, ..., C_K$.

As mentioned the K-means algorithm is one of the most popular clustering algorithms due to its simplicity and due to the fact that the running time of the algorithm is $O(t \cdot n \cdot p \cdot K)$, where t denotes the number of iterations and thereby linear in the number of iteration, data points, dimensions of the data points and the number of clusters we wish to construct. The initialization, the first iteration and the convergence of the K-means algorithm are shown in figure 2 on a toy example with three groups. The K-means algorithm is a fast

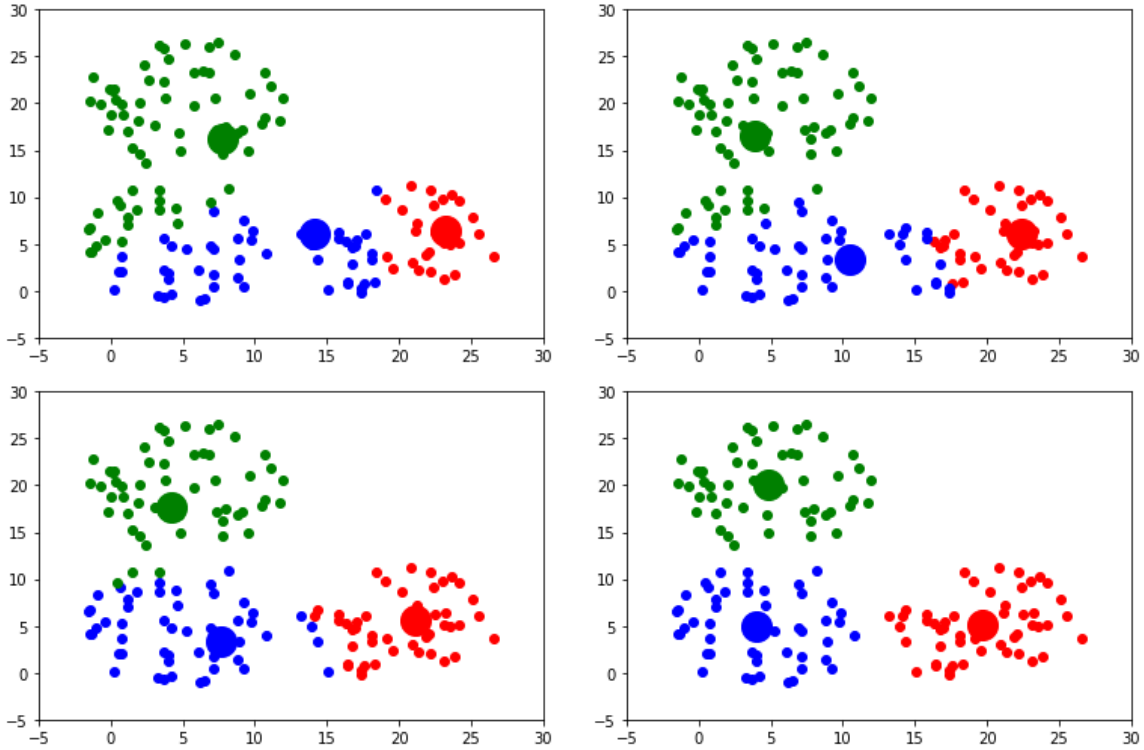

Figure 2: A visualization of the initialization and the iterations of the K-means algorithm. Random initialization of the centroids (Top left), iteration 1 (Top right), iteration 2 (Bottom left) and iteration 5 (Bottom right).

and useful tool for clustering applications, but as a consequence of its simplicity the algorithm is also limited in the sense that the set of points in $\mathbb{R}^p$ where it classifies group k, is always a convex set. Thus, the different classification regions make a partition of $\mathbb{R}^p$ (called a Voronoi diagram) into convex sets, which means that the K-means algorithm cannot correctly classify data sets where the groups form non-convex sets. This restricts the different ways the algorithm can construct clusters and thereby restricts the outcome of the algorithm. A classic example of this is shown in figure 1, where we obviously have two clusters; the circle in the middle and the torus around it. In this case it is easy to see that the K-means algorithm produces a more or less useless clustering, but since we often are dealing with points in a higher dimension than 3 and therefore cannot plot the data, this is not always easy to spot. It is also difficult if not impossible to know whether or not the restriction to convex sets constitutes obvious mistakes in our clustering. We will get back to this problem in chapter 4 about spectral clustering and also see an example in section 5.2 where the K-means algorithm is outperformed by spectral clustering on a data set with non-convex groups.

Another important thing to notice about the K-means algorithm is that it does not distinguish in how certain it is on the classification of a point. If for instance a point lies almost in the middle between two cluster centroids but slightly closer to one of them, it would be useful to know that this point is actually close to another cluster, but this information is completely lost using the K-means algorithm that gives a 'hard' assignment to each data point. In some sense a more probabilistic approach would solve this problem justifying formulations like 'X_i belongs to cluster k with probability p', thereby allowing a 'soft' assignment of the data points. To do this, we need to formulate the clustering problem in the setup of a statistical model making assumptions about the distribution of our data points. This approach is described in the next chapter about the EM-algorithm.

3 Clustering of Gaussian mixtures

We are now interested in the clustering problem from a model based perspective where we are able to apply standard methods such as finding conditional probabilities and calculating maximum likelihood estimates. To do this we need to make some assumptions about the distribution of our data set and furthermore consider the group assignment as a stochastic variable. The group assignment is, at least in the unsupervised learning setup, of course a latent variable since we have no way of observing the true group assignment. But as we will see in this chapter the notion of group assignment as a latent variable paves the way for an algorithmic method of maximizing the likelihood of the observed data in terms of the latent variables. The structure and notation in the following sections follows chapter 9.2 and 9.3 of [Bis06].

3.1 Gaussian mixture models

When dealing with the problem of finding subgroups in a given data set from a model based perspective, we are looking for a model that describes sub-populations in the overall data set and some prior probabilities of belonging to a specific group. Furthermore, we want some way of estimating the prior probabilities and the parameters of the model. Using the notion of a stochastic latent variable, that determines the group assignment, we can estimate the prior probabilities and the parameters in a *Gaussian mixture model*, where we assume that all sub-groups are normal distributed, such that the Gaussian mixture has density:

$$f_X(x) = \sum_{j=1}^{K} \pi_j \mathcal{N}\left(x|\mu_j, \Sigma_j\right), \tag{3.1}$$

with $\mu_j \in \mathbb{R}^p,\ \Sigma_j \in \mathrm{Mat}_{p\times p}(\mathbb{R}),\ 0 < \pi_j \leq 1$ for $j = 1, ..., K$ and $\sum_{j=1}^{K} \pi_j = 1$, where $\mathcal{N}\left(x|\mu_j, \Sigma_j\right)$ denotes the density of a multivariate normal distribution with mean μ_j and covariance Σ_j. Each $\mathcal{N}\left(x|\mu_j, \Sigma_j\right)$ is called a *mixture component* and the π_j's are referred to as the *mixing coefficients*.

For a stochastic variable X with density f_X as above in (3.1) we now want to express the distribution using the notion of a stochastic latent variable. To do so, consider the group assignment i.e. the latent variable, as a stochastic variable $Z \in \{0,1\}^K$ such that Z is a one-in-K encoding meaning that one of the entries in Z is 1 and the rest are 0. Furthermore we let

$$\mathbb{P}\left(Z_j = 1\right) = \pi_j, \quad \forall j \in \{1, 2, ..., K\}, \tag{3.2}$$

where $0 \leq \pi_j \leq 1$ and $\sum_{j=1}^{K} \pi_j = 1$. The stochastic variable Z acts a lot like the r_{ik}'s used in the K-means algorithm, but the key difference here is that Z is stochastic with the π_j's acting as *prior* probabilities of belonging to group j whereas the r_{ik}'s where deterministic. As we will see later this difference is exactly what allows us to specify the probability that X belongs to cluster k.

We want to express the joint distribution $f_{(X,Z)}(x, z)$ of (X, Z) in terms of the marginal distribution of Z and the conditional distribution $f_{X|Z}\left(x|z\right)$ of X given the latent variable Z. The marginal distribution of Z is specified above in terms of the π_j's, but it becomes convenient later on to write this as

$$f_Z(z) = \prod_{j=1}^{K} \pi_j^{z_j}. \tag{3.3}$$

Notice that since z is a one-in-K encoding, all of the exponents will be zero except one, say the k^{th} entry which is a one and therefore the product will exactly be equal to π_k and thereby this formulation is equivalent to (3.2). The conditional distribution of $X|Z$ is given as

$$f_{X|Z}\left(x|z\right) = \prod_{j=1}^{K} \mathcal{N}\left(x|\mu_j, \Sigma_j\right)^{z_j}, \tag{3.4}$$

where we again note that since z is a one-in-K encoding with say $z_k = 1$ we are taking a product over $K - 1$ one's and $\mathcal{N}\left(x|\mu_k, \Sigma_k\right)$, so given the latent variable we know which group x belongs to.

Together the conditional distribution of $X|Z$ and the marginal distribution of Z allow us to find the simultaneous distribution as the product of the two such that

$$f_{(X,Z)}(x,z) = f_Z(z) f_{X|Z}\left(x|z\right) = \prod_{j=1}^{K} \pi_j^{z_j} \mathcal{N}\left(x|\mu_j, \Sigma_j\right)^{z_j}. \tag{3.5}$$

Now since Z is a discrete stochastic variable we can find the marginal distribution of X by summing over all the K different outcomes of Z giving us

$$f_X(x) = \sum_z f_Z(z) f_{X|Z}\left(x|z\right), \tag{3.6}$$

which of course is equal to $\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x|\mu_j, \Sigma_j\right)$, which by assumption is the distribution of X.

At first glance it looks like we have just found an alternative way of formulating the marginal distribution of X, but the fact that we can now express the distribution of X in terms of the latent variable Z and consider the joint distribution of (X, Z), will later on give us the ingredients we need to formulate the EM-algorithm for Gaussian mixtures.

Another important ingredient in the construction of the EM-algorithm is the probability of belonging to group k i.e. $Z_k = 1$ given $X = x$ which we will denote $\gamma(z_k)$. Making use of Bayes' theorem for conditional probabilities we can calculate this as

$$\gamma(Z_k) := \mathbb{P}\left(Z_k = 1|X = x\right) = \frac{\mathbb{P}(Z_k = 1)\,\mathbb{P}\left(X = x|Z_k = 1\right)}{\mathbb{P}(X = x)}, \tag{3.7}$$

which in terms of densities can be written as

$$\gamma(Z_k) \sim f_{Z|X}\left(Z_k = 1|x\right) = \frac{\pi_k \mathcal{N}\left(x|\mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x|\mu_j, \Sigma_j\right)}. \tag{3.8}$$

As mentioned above π_k is considered the prior probability of belonging to group k whereas we can consider $\gamma(Z_k)$ as the *posterior* probability of belonging to group k after observing x which of course also makes sense since $\sum_{j=1}^{K} \gamma\left(Z_j\right) = 1$.

3.2 Maximum likelihood

Now looking at a data set $X_1, X_2, ..., X_n \in \mathbb{R}^p$ we want to apply our knowledge about the conditional and simultaneous distribution given a set of corresponding latent variables $Z_1, Z_2, ..., Z_n \in \{0,1\}^K$ in the framework as above. We therefore assume that $X_1, X_2, ..., X_n$ are drawn independently from a Gaussian mixture as in (3.1). To lighten the notation, we let $\boldsymbol{X} \in \mathrm{Mat}_{n\times p}(\mathbb{R})$ denote the data matrix with the i^{th} row given as X_i^T and $\boldsymbol{Z} \in \mathrm{Mat}_{n\times K}\left(\{0,1\}\right)$ the corresponding matrix of latent variables with the i^{th} row given as Z_i^T. Since the data is i.i.d. outcomes of the Gaussian mixture above we find the likelihood function as

$$L(\boldsymbol{X}; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} f_{X_i}(x_i; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i|\mu_j, \Sigma_j\right) \right), \tag{3.9}$$

where $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_K)$ and $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are representing μ_j and Σ_j for $j = 1, 2, ..., K$. Thus the log-likelihood function is given by

$$\ln\left(L(\boldsymbol{X}; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\right) = \sum_{i=1}^{n} \ln \left(\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i|\mu_j, \Sigma_j\right) \right). \tag{3.10}$$

As usual we want to maximize the log-likelihood function with respect to the parameters in the model.

3.2.1 Maximum likelihood estimates of $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

Finding the maximum likelihood estimates of the model parameters usually leads to a closed form solution of the parameters, but in the case of a Gaussian mixture we are not that lucky. However, as we will see later in this chapter we can estimate a local maximum of the log-likelihood function using the EM-algorithm. To do this we must first find the derivatives of the log-likelihood function with respect to $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and set them equal to zero, and see that we can express the maximum likelihood estimates using the posterior probabilities defined in section 3.1.

First we find the maximum likelihood estimate of $\boldsymbol{\mu}$ by setting the partial derivative of the log-likelihood function with respect to μ_k equal to zero. The partial derivative of the log-likelihood function with respect to μ_k is derived and simplified in the following way:

$$\begin{aligned}
\frac{\partial}{\partial \mu_k} \ln\left(L\left(\boldsymbol{X}; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\right) &= \frac{\partial}{\partial \mu_k} \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i | \mu_j, \Sigma_j\right)\right) \\
&= \sum_{i=1}^{n} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i | \mu_j, \Sigma_j\right)} \cdot \frac{\partial}{\partial \mu_k}\left(\mathcal{N}\left(x_i | \mu_k, \Sigma_k\right)\right) \\
&= \sum_{i=1}^{n} \underbrace{\frac{\pi_k \mathcal{N}\left(x_i | \mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i | \mu_j, \Sigma_j\right)}}_{:= \gamma\left(z_{ik}\right)} \cdot \left(-\Sigma_k^{-1}\left(x_i - \mu_k\right)\right) \\
&= -\sum_{i=1}^{n} \gamma\left(z_{ik}\right) \Sigma_k^{-1}\left(x_i - \mu_k\right),
\end{aligned} \tag{3.11}$$

where it is used in the third step that the partial derivative of the k^{th} mixture component with respect to μ_k can be found as

$$\begin{aligned}
\frac{\partial}{\partial \mu_k} \mathcal{N}\left(x | \mu_k, \Sigma_k\right) &= \frac{\partial}{\partial \mu_k}\left(\frac{1}{(2\pi)^{\frac{p}{2}}} \frac{1}{|\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(x - \mu_k\right)^T \Sigma_k^{-1}\left(x - \mu_k\right)\right)\right) \\
&= \mathcal{N}\left(x | \mu_k, \Sigma_k\right) \cdot \frac{\partial}{\partial \mu_k}\left(-\frac{1}{2}\left(x - \mu_k\right)^T \Sigma_k^{-1}\left(x - \mu_k\right)\right) \\
&= \mathcal{N}\left(x | \mu_k, \Sigma_k\right) \cdot \left(-\frac{1}{2} \cdot 2 \cdot \Sigma_k^{-1}\left(x - \mu_k\right)\right) \\
&= -\mathcal{N}\left(x | \mu_k, \Sigma_k\right) \cdot \Sigma_k^{-1}\left(x - \mu_k\right).
\end{aligned} \tag{3.12}$$

Now letting $N_k = \sum_{i=1}^{n} \gamma\left(z_{ik}\right)$ and setting the derivative of the log-likelihood equal to zero we find

$$\begin{aligned}
0 = -\sum_{i=1}^{n} \gamma\left(z_{ik}\right) \Sigma_k^{-1}\left(x_i - \mu_k\right) &\iff \sum_{i=1}^{n} \gamma\left(z_{ik}\right) \Sigma_k^{-1} \mu_k = \sum_{i=1}^{n} \gamma\left(z_{ik}\right) \Sigma_k^{-1} x_i \\
&\iff N_k \mu_k = \sum_{i=1}^{n} \gamma\left(z_{ik}\right) x_i \\
&\iff \mu_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma\left(z_{ik}\right) x_i.
\end{aligned} \tag{3.13}$$

Since N_k is the sum of all the posterior probabilities of belonging to group k after observing our data, it can be interpreted as the effective number of points we assign to group k. Therefore, the maximum likelihood estimate of μ_k can be viewed as a weighted mean of all the points with weight $\gamma\left(z_{ik}\right)$ for each x_i.

The maximum likelihood estimate of $\boldsymbol{\Sigma}$ is a bit more complicated to derive. We therefore consider the case $p = 1$ to give an overview of the calculations in a simple setting before deriving it for a general p. In the case

$p = 1$ the derivative can be found as

$$\begin{aligned}
\frac{\partial}{\partial \sigma_k^2} \ln\left(L\left(\boldsymbol{X}; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma^2}\right)\right) &= \frac{\partial}{\partial \sigma_k^2} \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{K} \pi_k \mathcal{N}\left(x_i | \mu_k, \sigma_k^2\right)\right) \\
&= \sum_{i=1}^{n} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i | \mu_j, \sigma_j^2\right)} \cdot \frac{\partial}{\partial \sigma_k^2}\left(\mathcal{N}\left(x_i | \mu_k, \sigma_k^2\right)\right) \\
&= \sum_{i=1}^{n} \underbrace{\frac{\pi_k \mathcal{N}\left(x_i | \mu_k, \sigma_k^2\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i | \mu_j, \sigma_j^2\right)}}_{\gamma(z_{ik})} \cdot \left(\frac{1}{2} \frac{1}{\left(\sigma_k^2\right)^2} \left(x_i - \mu_k\right)^2 - \frac{1}{2} \frac{1}{\sigma_k^2}\right) \\
&= \sum_{i=1}^{n} \gamma(z_{ik}) \cdot \left(\frac{1}{2} \frac{1}{\left(\sigma_k^2\right)^2} \left(x_i - \mu_k\right)^2 - \frac{1}{2} \frac{1}{\sigma_k^2}\right),
\end{aligned} \tag{3.14}$$

where it is used in the third step that the partial derivative of the k^{th} mixture component with respect to σ_k^2 can be found as

$$\begin{aligned}
\frac{\partial}{\partial \sigma_k^2} \mathcal{N}\left(x | \mu_k, \sigma_k^2\right) &= \frac{\partial}{\partial \sigma_k^2}\left(\frac{1}{\sqrt{2\pi}} \left(\sigma_k^2\right)^{-1/2} \exp\left(-\frac{\left(x_i - \mu_k\right)^2}{2\sigma_k^2}\right)\right) \\
&= \frac{1}{\sqrt{2\pi}} \frac{-1}{2} \left(\sigma_k^2\right)^{-3/2} \exp\left(-\frac{\left(x_i - \mu_k\right)^2}{2\sigma_k^2}\right) + \frac{1}{\sqrt{2\pi}} \left(\sigma_k^2\right)^{-1/2} \exp\left(-\frac{\left(x_i - \mu_k\right)^2}{2\sigma_k^2}\right) \frac{\left(x_i - \mu_k\right)^2}{2\left(\sigma_k^2\right)^2} \\
&= \mathcal{N}\left(x | \mu_k, \sigma_k^2\right) \left(\frac{1}{2} \frac{1}{\left(\sigma_k^2\right)^2} \left(x_i - \mu_k\right)^2 - \frac{1}{2} \frac{1}{\sigma_k^2}\right).
\end{aligned} \tag{3.15}$$

Now setting the derivative equal to zero and solving for σ_k^2 gives

$$\begin{aligned}
0 &= \sum_{i=1}^{n} \gamma(z_{ik}) \cdot \left(\frac{1}{2} \frac{1}{\left(\sigma_k^2\right)^2} \left(x_i - \mu_k\right)^2 - \frac{1}{2} \frac{1}{\sigma_k^2}\right) = \frac{1}{2} \frac{1}{\left(\sigma_k^2\right)^2} \sum_{i=1}^{n} \gamma(z_{ik}) \left(x_i - \mu_k\right)^2 - \frac{1}{2} \frac{1}{\sigma_k^2} N_k \\
&\iff N_k \sigma_k^2 = \sum_{i=1}^{n} \gamma(z_{ik}) \left(x_i - \mu_k\right)^2 \\
&\iff \sigma_k^2 = \frac{1}{N_k} \sum_{i=1}^{n} \gamma(z_{ik}) \left(x_i - \mu_k\right)^2.
\end{aligned} \tag{3.16}$$

Similar to the maximum likelihood estimate for μ_k it looks like the 'usual' one for a single Gaussian. The only difference is the weighing factor of the $\gamma(z_{ik})$'s and the fact that we divide by N_k instead of the usual number of points, since we do not know the true number of points belonging to group k.

For the case $p > 1$, we need to review some of the standard differentiation rules since we are now finding derivatives with respect to a matrix. The chain rule and the product rule does still apply and are stated in equation (A.1) and (A.2) in the appendix. Other than that we need equation (A.3), which tells us how to find the derivative of the determinant of a matrix with respect to the matrix, since we take the determinant of the covariance in the density of a multivariate normal distribution. Furthermore, we need to find the derivative with respect to Σ_k of $\left(x_i - \mu_k\right)^T \Sigma_k^{-1} \left(x_i - \mu_k\right)$. To do so, we first notice that $\left(x_i - \mu_k\right)^T \Sigma_k^{-1} \left(x_i - \mu_k\right)$ is a real number, so by taking the trace nothing is changed. Second, we use that the trace operator is invariant under cyclic permutations [PP12] to rewrite this as

$$\operatorname{Tr}\left(\left(x_i - \mu_k\right)^T \Sigma_k^{-1} \left(x_i - \mu_k\right)\right) = \operatorname{Tr}\left(\Sigma_k^{-1} \left(x_i - \mu_k\right) \left(x_i - \mu_k\right)^T\right) = \operatorname{Tr}\left(\Sigma_k^{-1} W\right), \tag{3.17}$$

where we define $W := \left(x_i - \mu_k\right) \left(x_i - \mu_k\right)^T$. Thus, we can apply equation (A.4) to find the derivative of $\left(x_i - \mu_k\right)^T \Sigma_k^{-1} \left(x_i - \mu_k\right)$ with respect to Σ_k in this equivalent formulation. Furthermore, we need to be aware

of the fact that the Σ_k's are covariance matrices and thereby especially symmetric and positive semi-definite which therefore restricts a maximum likelihood estimate into also being symmetric and positive semi-definite. But as we will see, applying these rules without restrictions of the result will give us a maximum likelihood estimate which indeed is a symmetric and positive semi-definite matrix. Thereby, we can ignore the restrictions and still be sure that the maximum likelihood estimate is indeed a maximum of the likelihood function *and* a valid covariance matrix.

As we saw in the case $p = 1$, the difficult step is finding the partial derivative of the k^{th} mixture component with respect to Σ_k. Applying the rules for matrix derivatives we can now find this as

$$\begin{aligned}
\frac{\partial}{\partial \Sigma_k}\mathcal{N}\left(x|\mu_k,\Sigma_k\right) &= \frac{\partial}{\partial \Sigma_k}\left((2\pi)^{-p/2}\left|\Sigma_k\right|^{-1/2}\exp\left(-\frac{1}{2}\left(x-\mu_k\right)^T\Sigma_k^{-1}\left(x-\mu_k\right)\right)\right) \\
&= (2\pi)^{-p/2}\,\frac{-1}{2}|\Sigma_k|^{-3/2}|\Sigma_k|\;\Sigma_k^{-1}\exp\left(-\frac{1}{2}\left(x-\mu_k\right)^T\Sigma_k^{-1}\left(x-\mu_k\right)\right) \\
&\quad + (2\pi)^{-p/2}\left|\Sigma_k\right|^{-1/2}\exp\left(-\frac{1}{2}\left(x-\mu_k\right)^T\Sigma_k^{-1}\left(x-\mu_k\right)\right)\frac{-1}{2}\frac{\partial\ \mathrm{Tr}\left(\Sigma_k^{-1}W\right)}{\partial\Sigma_k} \\
&= -\frac{1}{2}\Sigma_k^{-1}\mathcal{N}\left(x|\mu_k,\Sigma_k\right) + \frac{1}{2}\Sigma_k^{-1}W\Sigma_k^{-1}\mathcal{N}\left(x|\mu_k,\Sigma_k\right).
\end{aligned} \tag{3.18}$$

Thus, we can find the derivative of the log-likelihood function with respect to Σ_k as

$$\begin{aligned}
\frac{\partial}{\partial \Sigma_k}\ln\left(L\left(\boldsymbol{X};\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma_k}\right)\right) &= \frac{\partial}{\partial \Sigma_k}\sum_{i=1}^{n}\ln\left(\sum_{j=1}^{K}\pi_k\mathcal{N}\left(x_i|\mu_k,\Sigma_k\right)\right) \\
&= \sum_{i=1}^{n}\frac{\pi_k}{\sum_{j=1}^{K}\pi_j\mathcal{N}\left(x_i|\mu_j,\Sigma_j\right)}\cdot\frac{\partial}{\partial\Sigma_k}\left(\mathcal{N}\left(x_i|\mu_k,\Sigma_k\right)\right) \\
&= \sum_{i=1}^{n}\underbrace{\frac{\pi_k\mathcal{N}\left(x_i|\mu_k,\Sigma_k\right)}{\sum_{j=1}^{K}\pi_j\mathcal{N}\left(x_i|\mu_j,\Sigma_j\right)}}_{\gamma\left(z_{ik}\right)}\cdot\left(\frac{1}{2}\Sigma_k^{-1}W\Sigma_k^{-1}-\frac{1}{2}\Sigma_k^{-1}\right) \\
&= \sum_{i=1}^{n}\gamma\left(z_{ik}\right)\cdot\left(\frac{1}{2}\Sigma_k^{-1}W\Sigma_k^{-1}-\frac{1}{2}\Sigma_k^{-1}\right).
\end{aligned} \tag{3.19}$$

Now setting the derivative equal to zero and solving for Σ_k (by multiplying with Σ_k from the left and the right) gives

$$\begin{aligned}
0 = \sum_{i=1}^{n}\gamma\left(z_{ik}\right)\cdot\left(\frac{1}{2}\Sigma_k^{-1}W\Sigma_k^{-1}-\frac{1}{2}\Sigma_k^{-1}\right) &= \frac{1}{2}\sum_{i=1}^{n}\gamma\left(z_{ik}\right)\Sigma_k^{-1}W\Sigma_k^{-1}-\frac{1}{2}\Sigma_k^{-1}N_k \\
\iff N_k\Sigma_k &= \sum_{i=1}^{n}\gamma\left(z_{ik}\right)W \\
\iff \Sigma_k &= \frac{1}{N_k}\sum_{i=1}^{n}\gamma\left(z_{ik}\right)W = \sum_{i=1}^{n}\gamma\left(z_{ik}\right)\left(x_i-\mu_k\right)\left(x_i-\mu_k\right)^T,
\end{aligned} \tag{3.20}$$

which also looks like the generalization from the one-dimensional case.

Also Σ_k is obviously symmetric since W is and it is positive semi-definite, since we for $v\in\mathbb{R}^p$ with $v\neq 0$ have

$$\begin{aligned}
v^T\Sigma_k v &= \frac{1}{N_k}\sum_{i=1}^{n}\gamma\left(z_{ik}\right)v^T\left(x_i-\mu_k\right)\left(x_i-\mu_k\right)^T v \\
&= \frac{1}{N_k}\sum_{i=1}^{n}\gamma\left(z_{ik}\right)\underbrace{\left|\left|v^T\left(x_i-\mu_k\right)\right|\right|^2}_{\geq 0}\geq 0.
\end{aligned} \tag{3.21}$$

Finally, we need to find the maximum likelihood estimate of $\boldsymbol{\pi}$ under the restriction that $\sum_{j=1}^{K}\pi_j = 1$. We can solve this constrained maximization problem by using the method of Lagrange multipliers which for π_k gives us

the following Lagrangian function [BV09].

$$\mathcal{L}(\pi_k, \lambda) = \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i|\mu_j, \Sigma_j\right)\right) + \lambda\left(\sum_{j=1}^{K} \pi_j - 1\right), \tag{3.22}$$

thus the gradient of the Lagrangian becomes

$$\nabla L(\pi_k, \lambda) = \left(\sum_{i=1}^{n} \frac{\mathcal{N}\left(x_i|\mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i|\mu_j, \Sigma_j\right)} + \lambda,\ \sum_{j=1}^{K} \pi_j - 1\right). \tag{3.23}$$

To maximize $\mathcal{L}$ we set the gradient equal to $(0, 0)$. By assumption $\sum_{j=1}^{K} \pi_j - 1 = 0$, so it is only relevant to consider the first coordinate of the gradient of the Lagrangian. Setting this equal to zero gives

$$\begin{aligned}
0 &= \sum_{i=1}^{n} \frac{\mathcal{N}\left(x_i|\mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i|\mu_j, \Sigma_j\right)} + \lambda \iff \\
0 &= \sum_{i=1}^{n} \underbrace{\frac{\pi_k \mathcal{N}\left(x_i|\mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_i|\mu_j, \Sigma_j\right)}}_{\gamma(z_{ik})} + \pi_k \lambda \iff \\
0 &= \underbrace{\sum_{i=1}^{n} \gamma(z_{ik})}_{N_k} + \pi_k \lambda \iff \\
\pi_k &= \frac{-N_k}{\lambda}.
\end{aligned} \tag{3.24}$$

Now summing over $j \in \{1, 2, ..., K\}$ and using the fact that $\sum_{j=1}^{K} N_j = n$, we find that

$$1 = \sum_{j=1}^{K} \pi_j = \sum_{j=1}^{K} \frac{-N_k}{\lambda} = \frac{-n}{\lambda} \implies \lambda = -n. \tag{3.25}$$

Using this the maximum likelihood estimate of π_k becomes

$$\pi_k = \frac{N_k}{n}. \tag{3.26}$$

Thereby the maximum likelihood estimates for the mixing coefficients are given as the effective number of data points assigned to the group divided by the number of data points. This also fits the interpretation of the mixing coefficients as the weight of each component in the mixture.

3.3 The EM-algorithm for Gaussian mixtures

We now know the maximum likelihood estimates of the model, but since they all depend on the $\gamma(z_{ik})$'s which again in turn as we saw in (3.8) depend on the parameters of the model, we cannot find a closed-form solution to the maximization of the likelihood function. However, using an algorithmic approach we can maximize the likelihood function through iterations by using the co-dependence of the $\gamma(z_{ik})'s$ and the parameters of the models to maximize the likelihood function with respect to the parameters keeping the $\gamma(z_{ik})$'s fixed and then recalculate the $\gamma(z_{ik})$'s in terms of the new parameters. Repeating this process is exactly the *Expectation-Maximization*-algorithm or the EM-algorithm in this special case where we assume our data comes from a Gaussian mixture [Bis06]. These two steps described above are referred to as the E-step and the M-step, where the E-step is calculating the $\gamma(z_{ik})$'s given the current values of the parameters and the M-step is re-estimating the maximum likelihood estimates while keeping the $\gamma(z_{ik})$'s fixed. In chapter 9.4 of [Bis06] it is showed that the general EM-algorithm does in fact converge to a local maximum of the likelihood function, thus the special case assuming a Gaussian mixture model will also converge to a local maximum of the likelihood function. After the algorithm has converged we group our data points into clusters $C_1, ..., C_K$ using the $\gamma(z_{ik})$'s, where

$$C_k = \left\{X_i | 1 \leq i \leq n,\ \text{argmax}_j \gamma\left(z_{ij}\right) = k\right\}. \tag{3.27}$$

Thereby the EM-algorithm for Gaussian mixtures can be stated as follows:

Algorithm 2 The EM-algorithm for Gaussian mixtures

Input: $X_1, X_2, ..., X_n \in \mathbb{R}^p$, $K \in \mathbb{N}$ (`Number of desired clusters`)

1: `Initialize` $\mu_j \in \mathbb{R}^p$, $\Sigma_j \in \text{mat}_{p\times p}(\mathbb{R})$ `and` $\pi_j \in (0,1)$ `randomly for` $j = 1, 2, ..., K$, $i = 1, 2, ..., n$, `such that` $\sum_{j=1}^{K} \pi_j = 1$ `and evaluate the log-likelihood function,`

$$\ln\left(L\left(\boldsymbol{X};\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\right) = \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{K} \pi_j \mathcal{N}\left(X_i|\mu_j,\Sigma_j\right)\right)$$

2: `E-step: Evaluate the` $\gamma(z_{ik})$`'s such that`

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}\left(X_i|\mu_k,\Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(X_i|\mu_j,\Sigma_j\right)}$$

3: `M-step: Re-estimate the parameters of the model using the new` $\gamma(z_{ij})$`'s such that`

$$\mu_k^{\text{new}} = \frac{1}{N_k}\sum_{i=1}^{n} \gamma(z_{ik}) X_i$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k}\sum_{i=1}^{n} \gamma(z_{ik})(X_i - \mu_k^{\text{new}})(X_i - \mu_k^{\text{new}})^T$$

$$\pi_k^{\text{new}} = \frac{N_k}{n}$$

4: `Evaluate the log-likelihood with the new parameters,`

$$\ln\left(L\left(\boldsymbol{X};\boldsymbol{\pi}^{\textbf{new}},\boldsymbol{\mu}^{\textbf{new}},\boldsymbol{\Sigma}^{\textbf{new}}\right)\right) = \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{K} \pi_j^{\text{new}} \mathcal{N}\left(X_i|\mu_j^{\text{new}},\Sigma_j^{\text{new}}\right)\right)$$

5: `Repeat step 2-4 until convergence of the parameters or the log-likelihood function`

6: `Finally, let` $C_j = \left\{X_i | 1 \leq i \leq n,\ \text{argmax}_k \gamma(z_{ik}) = j\right\}$ `for` $j = 1, 2, ...K$

Output: Clusters $C_1, C_2, ..., C_K$.

As shown in figure 3, the convergence of the algorithm can be quite slow even on a simple data set. To speed up the convergence of the algorithm, it is therefore common to initialize the parameters of the model using the K-means algorithm where the parameters of each mixture component are set as the sample mean and covariance of each group found using K-means and the mixing coefficients are set as the fraction of data points assigned to the group by K-means.

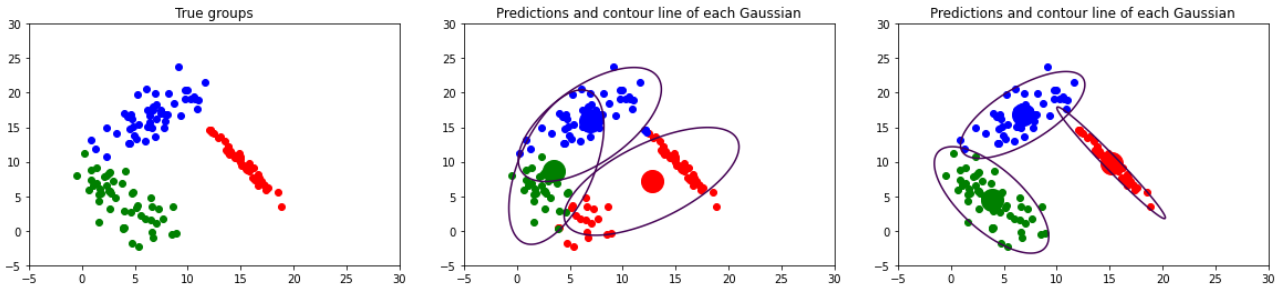

Figure 3: A visualization of the iterations of the EM-algorithm for Gaussian mixtures and contour lines of the density of each mixture component. Groups generated by three bi-variate Gaussians (left), iteration 5 of the EM-algorithm (middle), iteration 25 of the EM-algorithm (right).

3.4 The EM-algorithm for Gaussian mixtures vs. K-means

As discussed in section 2 the K-means algorithm gives a 'hard' assignment of a data point to a group whereas we in the EM-algorithm for Gaussian mixtures can talk about probabilities of belonging to a group in terms of the $\gamma(z_{ik})$'s. But as we will see shortly, we can derive the K-means algorithm as the limit of a special case of the EM-algorithm for Gaussian mixtures. If we assume that our data is i.i.d. outcomes of a mixture of Gaussians with distribution

$$f_X(x) = \sum_{j=1}^{K} \pi_j \mathcal{N}\left(x|\mu_j, \varepsilon I_p\right), \tag{3.28}$$

with $\varepsilon > 0,\ \mu_j \in \mathbb{R}^p,\ 0 < \pi_j \leq 1$ for $j = 1, 2, ..., K$ and $\sum_{j=1}^{K} \pi_j = 1$, such that all mixture components share the same covariance matrix, the features of each component are independent and the variance of each feature is ε. Thereby the distribution of the k^{th} component can written as

$$\begin{aligned} \mathcal{N}\left(x|\mu_k, \varepsilon I_p\right) &= \frac{1}{(2\pi)^{p/2}} \frac{1}{|\varepsilon I_p|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \left(\varepsilon I_p\right)^{-1}(x-\mu_k)\right) \\ &= \frac{1}{(2\pi\varepsilon)^{p/2}} \exp\left(-\frac{1}{2\varepsilon}||x-\mu_k||^2\right). \end{aligned} \tag{3.29}$$

We are interested in what happens to the classification in the EM-algorithm when we let $\varepsilon \to 0$, so we therefore need to look at the $\gamma(z_{ik})$'s. For the i^{th} data point and group k the posterior probability of belonging to group k will in this model be given as

$$\gamma(z_{ik}) = \frac{\pi_k \frac{1}{(2\pi\varepsilon)^{p/2}} \exp\left(-\frac{1}{2\varepsilon}||x_i-\mu_k||^2\right)}{\sum_{j=1}^{K} \pi_j \frac{1}{(2\pi\varepsilon)^{p/2}} \exp\left(-\frac{1}{2\varepsilon}||x_i-\mu_j||^2\right)} = \frac{\pi_k \exp\left(-\frac{1}{2\varepsilon}||x_i-\mu_k||^2\right)}{\sum_{j=1}^{K} \pi_j \exp\left(-\frac{1}{2\varepsilon}||x_i-\mu_j||^2\right)}. \tag{3.30}$$

Dividing with the numerator we can write this as

$$= \frac{1}{1 + \sum_{j\neq k} \frac{\pi_j \exp\left(-\frac{1}{2\varepsilon}||x_i-\mu_j||^2\right)}{\pi_k \exp\left(-\frac{1}{2\varepsilon}||x_i-\mu_k||^2\right)}} = \frac{1}{1 + \sum_{j\neq k} \frac{\pi_j}{\pi_k} \exp\left(\frac{1}{2\varepsilon}\left(||x_i-\mu_k||^2 - ||x_i-\mu_j||^2\right)\right)}. \tag{3.31}$$

Now considering what happens if we let $\varepsilon \to 0$, we see that it suffices to check two cases; whether or not $||x_i - \mu_k||^2$ is the smallest of all $||x_i - \mu_j||^2$ for $j = 1, 2, ..., K$. If $||x_i - \mu_k||^2$ is the smallest, then all the terms in the sum in the denominator will go to 0, since the argument in the exponential function will go to minus infinity. In this case the posterior probability will then go to 1. If on the other hand $||x_i - \mu_k||^2$ is not the smallest, then there exists a $j \neq k$ such that $||x_i - \mu_k||^2 > ||x_i - \mu_j||^2$ and the j^{th} term in the sum will then go to infinity. In this case the posterior probability go to 0. Formally written that is,

$$\lim_{\varepsilon\to 0} \gamma(z_{ik}) = \begin{cases} 1, & \text{if } ||x_i-\mu_k|| < ||x_i-\mu_j||^2,\ \forall j = 1, 2, ..., K, \quad j \neq k \\ 0, & \text{if } \exists j \in \{1, 2, ..., K\},\ j \neq k \ \ s.t.\ ||x_i-\mu_k||^2 > ||x_i-\mu_j||^2. \end{cases} \tag{3.32}$$

Comparing the EM-algorithm in this setup to the K-means algorithm and remembering the definition of the r_{ik}'s we then see that

$$\gamma(z_{ik}) \to r_{ik}, \quad \text{for } \varepsilon \to 0. \tag{3.33}$$

Thus, we can actually consider the K-means algorithm as a special case of the EM-algorithm for Gaussian mixtures. This result of convergence is also showed with a concrete data example in section 5.2.1.
As we will also see in section 5.3, different restrictions to the type of covariance matrix can be applied (If they are fair assumptions) to the EM-algorithm, which then will produce different clusterings.

4 Spectral clustering

Spectral clustering is like K-means a purely algorithmic approach to clustering meaning that we do not assume an underlying distribution of the data set. But in many cases spectral clustering outperforms the K-means algorithm [Lux07], which we will also see in some examples in the next chapter. The mathematical setup of the different spectral clustering algorithms is a bit more complex than for K-means since it requires some graph theory, but luckily the algorithms rely on solving eigenvalue-problems which can be solved efficiently using standard linear algebra software. Before stating two versions of the algorithm, we first need to define a similarity graph, three different graph Laplacians and show some of the properties of these graph Laplacians. The structure of this chapter follows much of the structure of [Lux07].

4.1 Constructing a similarity graph

A key component in spectral clustering is the conversion of a data set into a similarity graph. To do so, we first need to define a graph, some related concepts, a similarity function and a similarity graph. In the literature you can find many different definitions of *graphs*. The one we will use is that *a graph G is a pair $G = (V, E)$ of sets V and E, where V denotes the vertices of the graph and E denotes the edges of the graph where e_{ij} denotes the edges between $v_i, v_j \in V$* [Die05]. We will be using graphs as a way of structuring our data such that each data point X_i is represented in the graph by a vertex v_i. The aim is now to use the graph structure to find a way to partition the data set. We will use some of the standard terminology from graph theory without definitions, but a few important terms concerning subsets of a graph are defined in the following.

Definition 4.1 (*Connected subset, connected component and partition*)**.** Let $G = (V, E)$ be a graph.

1. A subset $A \subseteq V$ is called connected if any two vertices in A can be connected by a path such that all vertices in the path also are in A.
2. A subset $A \subseteq V$ is called a connected component if it is connected and there are no edges between vertices in A and A^c. That is, $\left\{e_{ij} \mid i \in A,\ j \in A^c\right\} = \emptyset$.
3. $A_1, A_2, ..., A_k$ is called a partition of the graph G if $A_i \neq \emptyset$ for $i = 1, 2, ..., k$, $A_i \cap A_j = \emptyset$ for $i \neq j$ and $A_1 \cup A_2 \cup ... \cup A_k = V$.

Furthermore, we will need a way of measuring the similarity between two points in the data set, which we have in the following definition.

Definition 4.2 (*Similarity function on a data set*)**.** A similarity function on a data set $X = \{X_1, X_2, ..., X_n\}$ with $X_i \in \mathbb{R}^p$ is a function

$$\begin{aligned} s:\ \mathbb{R}^p \times \mathbb{R}^p &\to [0, \infty) \\ X_i \times X_j &\to s\left(X_i, X_j\right). \end{aligned}$$

Given a similarity function s we can now construct an *adjacency matrix* $W = \left(w_{ij}\right)_{i,j=1,2,...,n}$ with $w_{ij} = s\left(X_i, X_j\right)$, where each w_{ij} is a *weight* given to the edge e_{ij}, and thereby we can define a similarity graph as follows.

Definition 4.3 (*Similarity graph*)**.** Let $X_1, ..., X_n \in \mathbb{R}^p$ and let s be a similarity function on $\{X_1, X_2, ..., X_n\}$. The similarity graph of $X_1, ..., X_n$ with respect to s is a graph $G = (V, E)$ together with an adjacency matrix $W = \left(w_{ij}\right)_{i,j=1,...,n}$, where $V = \{X_1, ..., X_n\}$ and $e_{ij} \in E \iff w_{ij} > 0$, with $w_{ij} = s\left(X_i, X_j\right)$.

We say that G is *undirected* if W is symmetric. Otherwise we say that G is *directed*.

To lighten the notation we will just write that $G = (V, E)$ is a similarity graph with adjacency matrix W, when we are referring to the similarity graph of $X_1, ..., X_n$ with respect to a similarity function s. Popular choices of similarity functions include the Gaussian similarity function, the $\varepsilon-$neighborhood function and the k-nearest neighbors function, which lead to the following similarity graphs that are visualized in figure 4.

A fully connected graph: Using the Gaussian similarity function $s\left(v_i, v_j\right) = exp\left(\frac{-||X_i - X_j||^2}{2\sigma^2}\right)$ for a $\sigma > 0$, all vertices of G become connected, but the weights on the edges are small if the distance between the points is large and vice versa.

The ε-neighborhood graph: Using the $\varepsilon-$neighborhood function

$$s\left(X_i, X_j\right) = \begin{cases} 1, & \text{if } ||X_i - X_j|| \leq \varepsilon \\ 0, & \text{Otherwise,} \end{cases}$$

for an $\varepsilon > 0$, we obtain the ε-neighborhood graph. This graph is usually considered an unweighted graph, since weighting the edges would not give more information about the data points.

The k-nearest neighbors graph: Using the k-nearest neighbors function where $s\left(X_i, X_j\right) = 1$ if X_j is among the k-nearest neighbors of X_i and 0 otherwise, we construct a directed unweighted similarity graph, since s in this case is not symmetric. A way to make this graph symmetric is either simply ignore the directions (adding the missing edges) or to use *the mutual k-nearest neighbors graph* where $s\left(X_i, X_j\right) = 1$ if both X_j is among the k-nearest neighbors of X_i and the other way around. The difference between the unweighted k-nearest neighbors and the mutual k-nearest neighbors is shown in figure 4.

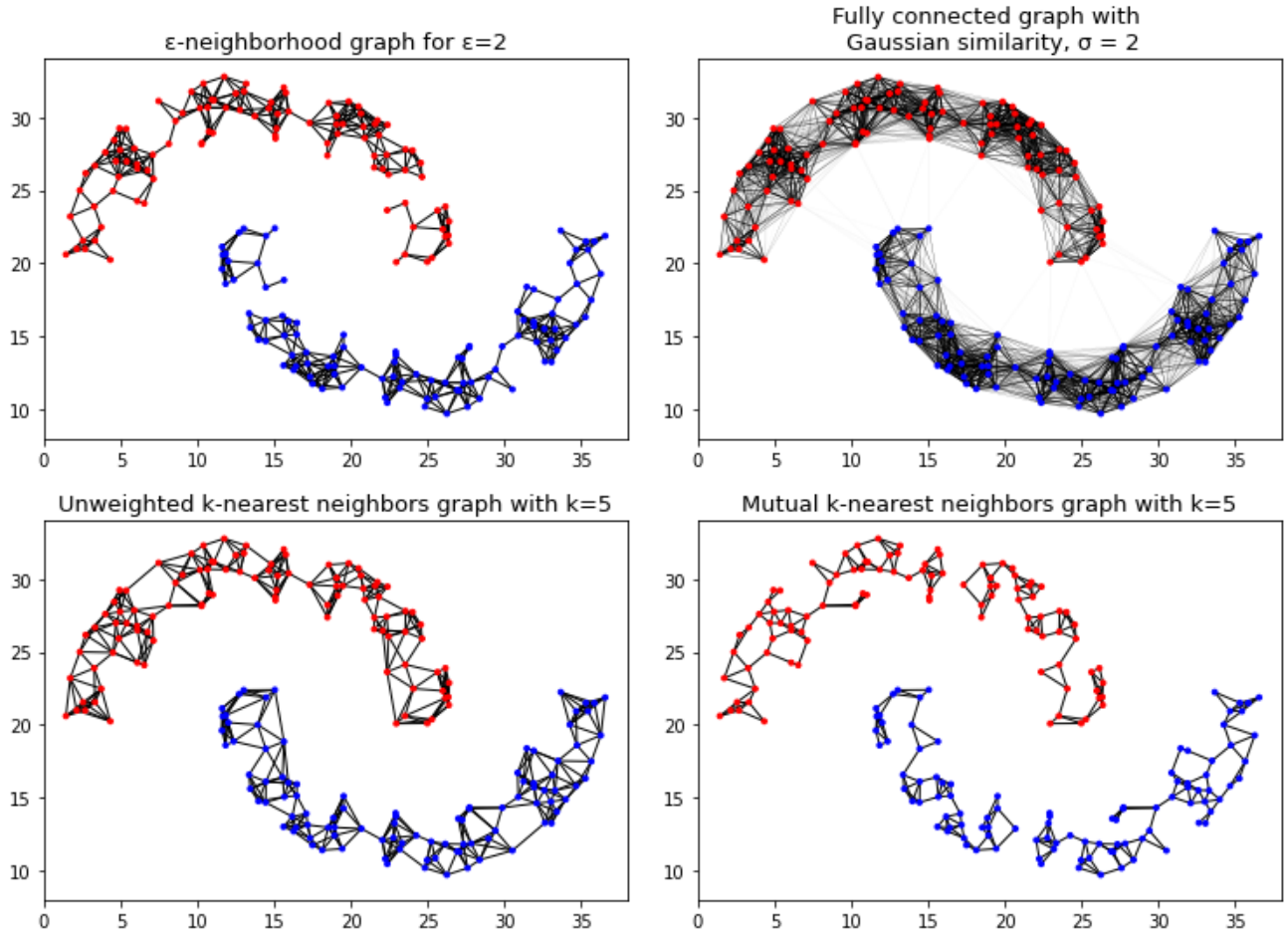

Figure 4: Different similarity graphs as defined above on a simulated data set consisting of two groups. Note that for the fully connected graph the edges are weighted, which is visualized such that the line widths are determined by the weights.

Using a similarity function s and by constructing the similarity graph of the data with respect to s, we can now formulate the goal of the clustering procedure as trying to find a partition of the graph such that edges between

points in different clusters have low weights, while edges within the same cluster have high weights. To further concretize this, we need a few definitions and some short notation that will come in handy later on.

Notation: Let $G = (V, E)$ be a similarity graph with adjacency matrix W.

For $A \subseteq V$ let $i \in A := \{i \mid v_i \in A\}$ and let $\mathbb{1}_A := \left(\mathbb{1}_{\{v_1 \in A\}}, \mathbb{1}_{\{v_2 \in A\}}, ..., \mathbb{1}_{\{v_n \in A\}}\right)^T$.

Definition 4.4 (*Degree of a vertex, degree matrix,*)**.** Let $G = (V, E)$ be a similarity graph with adjacency matrix W. The degree d_i of a vertex $v_i \in V$ is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}.$$

The degree matrix D is defined as

$$D = \operatorname{diag}\{d_1, d_2, ..., d_n\}.$$

We can thus measure the *size* of a subset $A \subseteq V$ in two different ways:

$$|A| := |\{i \mid i \in A\}| = \text{Number of vertices in } A.$$
$$\operatorname{vol}(A) := \sum_{i \in A} d_i = \text{Sum of degrees of vertices in } A.$$

As mentioned above, the intuition of this graph approach is to find a partition of the graph such that the weights of the edges within a cluster are small and the weights of the edges between points in different clusters are large. To formalize this approach, we need to define a way of measuring the weight between different subsets of a similarity graph, this is done in the following definition.

Definition 4.5 (*Weight between subsets of a graph*)**.** Let $G = (V, E)$ be a similarity graph with adjacency matrix W and let $A, B \subseteq V$. The weight between A and B is defined as

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

Now we have the tools we need to define different variants of *graph Laplacians* and show some of their properties which will be important in the spectral clustering algorithm.

4.2 Graph Laplacians

There is no unique way of defining graph Laplacians, but in this section we will define and study some of the properties of three different graph Laplacians. As we will see, these different graph Laplacians lead to slightly different formulations of spectral clustering and thus to different viewpoints when justifying the 'correctness' of the two different algorithms. We define the three different graph Laplacians as follows.

Definition 4.6. Let $G = (V, E)$ be an undirected similarity graph with adjacency matrix W and degree matrix D. We define the graph Laplacians

$$L = D - W$$
$$L_{\text{sym}} = D^{-1/2} L D^{-1/2} = I_n - D^{-1/2} W D^{-1/2}$$
$$L_{\text{rw}} = D^{-1} L = I_n - D^{-1} W,$$

where 'sym' is short for symmetric, since L_{sym} is symmetric and 'rw' is short for random walk, since L_{rw} is correlated with a random walk perspective of the graph partitioning problem [Lux07]. We will refer to L as the unnormalized Laplacian and to L_{sym} and L_{rw} as the normalized Laplacians. In the following we state and prove the properties of the graph Laplacians we need in relation to the spectral clustering algorithms. The properties of the unnormalized Laplacian and the normalized Laplacians are off course closely related, but for completeness we state and prove them in all three cases starting with the unnormalized Laplacian.

4.2.1 Properties of the unnormalized graph Laplacian

Graph Laplacians are used in the field of spectral graph theory and are therefore a well-studied object. In this section we concentrate on the key properties used in relation to spectral clustering.

Proposition 4.1 (Properties of the unnormalized graph Laplacian)**.** *Let $G = (V, E)$ be an undirected similarity graph with adjacency matrix W and degree matrix D. $L = D - W$ satisfies the following:*

1. $\forall u \in \mathbb{R}^n : \ u^T L u = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} \left(u_i - u_j\right)^2$.
2. *L is symmetric and positive semi-definite.*
3. *The smallest eigenvalue of L is 0, with corresponding eigenvector $\mathbb{1} := (1, 1, ..., 1)^T \in \mathbb{R}^n$.*
4. *L has n non-negative, real-valued eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$.*

Proof:
(1): Let $u \in \mathbb{R}^n$.

$$\begin{aligned} u^T L u = u^T D u - u^T W u = \sum_{i=1}^n d_i u_i^2 - u^T \begin{pmatrix} \sum_{i=1}^n w_{1i} u_i \\ \vdots \\ \sum_{i=1}^n w_{ni} u_i \end{pmatrix} = \sum_{i=1}^n d_i u_i^2 - \sum_{i=1}^n \sum_{j=1}^n w_{ij} u_i u_j \\ = \frac{1}{2} \left(\sum_{i=1}^n d_i u_i^2 - 2 \sum_{i=1}^n \sum_{j=1}^n w_{ij} u_i u_j + \sum_{i=1}^n d_i u_i^2 \right) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} \left(u_i - u_j\right)^2 . \end{aligned} \tag{4.1}$$

(2): Symmetry follows from the symmetry of W and D such that $L^T = (D - W)^T = D^T - W^T = D - W = L$. Positive semi-definite follows from (1), since all terms in the sum are non-negative.

(3): By (2) the eigenvalues are non-negative, so it suffices to check that $\mathbb{1}^T L \mathbb{1} = 0$, which is true by (1).

(4): The symmetry implies that L has n real-valued eigenvalues, and in the proof of (3) we argued that the eigenvalues are non-negative.

Besides the basic properties of the unnormalized graph Laplacian, we will need the following proposition which will be important in relation to spectral clustering.

Proposition 4.2 (Number of connected components and the eigenvectors of L)**.** *Let $G = (V, E)$ be an undirected similarity graph with adjacency matrix W and degree matrix D. Let $L = D - W$ be the unnormalized graph Laplacian, then we have the following:*

1. *The multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, A_2, ..., A_k$ in G.*
2. *The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \mathbb{1}_{A_2}, ..., \mathbb{1}_{A_k}$ of those components.*

Proof: We start by proving the proposition in the simple case of $k = 1$, which is the case where G is connected. Assume that $u \in \mathbb{R}^n$ is an eigenvector of L with eigenvalue 0. Using this assumption and (1) from proposition 4.1, we have the equality

$$0 = u^T L u = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} \left(u_i - u_j\right)^2 . \tag{4.2}$$

Since the weights are non-negative and strictly positive if and only if vertex i and vertex j are connected, this implies that $u_i = u_j$, whenever vertex i and j are connected. This further implies that the coordinates of u must be constant for all vertices that can be connected by a path in G. But since G is connected all vertices can be connected by a path and thus u must be on the form $u = c\mathbb{1}$ for some $c \in \mathbb{R}$. This means that the multiplicity of the eigenvalue 0 is 1 and the eigenspace of 0 is spanned by $\mathbb{1}_G$.

Now assume that G has k connected components for a general $k \in \mathbb{N}$ and assume without loss of generality that the vertices are ordered by the connected components they belong to. In this case W will have a block-diagonal form which again implies that L will have a block-diagonal form such that L is on the form

$$L = \begin{pmatrix} L_1 & & \\ & \ddots & \\ & & L_k \end{pmatrix}, \tag{4.3}$$

where each L_i is the graph Laplacian of the sub-graph of the i^{th} connected component of G. Since L is on a block-diagonal form, the eigenvalues of L are the union of the eigenvalues of each L_i and the eigenvectors of L are the eigenvectors of each L_i with zeros on the positions or coordinates of the other blocks [HJ85]. Since each L_i is the graph Laplacian of a connected graph, we know using the case $k = 1$ that the multiplicity of 0 is k (one for each connected component) with eigenvectors $\mathbb{1}_{A_1}, \mathbb{1}_{A_2}, ..., \mathbb{1}_{A_k}$, which concludes the proof.

4.2.2 Properties of the normalized graph Laplacians

Similar to the properties of the unnormalized graph Laplacian the most important properties in relation to spectral clustering of the two normalized graph Laplacians L_{sym} and L_{rw} are stated and proved in the following two propositions.

Proposition 4.3 (Properties of the normalized graph Laplacians)**.** *Let $G = (V, E)$ be an undirected similarity graph with adjacency matrix W and degree matrix D. $L_{sym} = D^{-1/2}LD^{-1/2}$ and $L_{rw} = D^{-1}L$ satisfy the following:*

1. *$\forall u \in \mathbb{R}^n$: $u^T L_{sym} u = \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n w_{ij}\left(\frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_j}}\right)^2$.*
2. *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}u$.*
3. *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigenproblem $Lu = \lambda Du$.*
4. *0 is an eigenvalue of L_{rw} with the constant one-vector $\mathbb{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbb{1}$.*
5. *L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$.*

Proof:
(1) Let $u \in \mathbb{R}^n$.

$$u^T L_{sym} u = u^T D^{-1/2} L D^{-1/2} u = \begin{pmatrix} \frac{u_1}{\sqrt{d_1}} \dots \frac{u_n}{\sqrt{d_n}} \end{pmatrix} L \begin{pmatrix} \frac{u_1}{\sqrt{d_1}} \\ \vdots \\ \frac{u_n}{\sqrt{d_n}} \end{pmatrix} \overset{\text{prop 4.1 (1)}}{=} \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n w_{ij}\left(\frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_j}}\right)^2. \tag{4.4}$$

(2) Let λ be an eigenvalue of L_{rw} with eigenvector u and let $w = D^{1/2}u$.

$$L_{rw}u = \lambda u \iff D^{-1}Lu = \lambda u \iff D^{-1/2}LD^{-1/2}D^{1/2}u = D^{1/2}\lambda u \iff L_{sym}w = \lambda w. \tag{4.5}$$

(3) Let λ be an eigenvalue of L_{rw} with eigenvector u.

$$L_{rw}u = \lambda u \iff D^{-1}Lu = \lambda u \iff Lu = \lambda Du. \tag{4.6}$$

(4) The first statement follows from prop. 4.1 (3), since $L\mathbb{1} = 0$. The second statement follows using (2) above.

(5) By construction L_{sym} is symmetric so it has n real-valued eigenvalues and by (1) it is positive semi-definite so the eigenvalues are non-negative. Now using (2) the result also follows for L_{rw}.

Like in the unnormalized case we will also need an important proposition on the relation between the multiplicity of the eigenvalue 0 and the number of connected components of G.

Proposition 4.4 (Number of connected components and the eigenvectors of L_{rw} and L_{sym})**.** *Let $G = (V, E)$ be an undirected similarity graph with adjacency matrix W and degree matrix D. Let $L_{sym} = D^{-1/2}LD^{-1/2}$ and $L_{rw} = D^{-1}L$ be the normalized graph Laplacians, then we have the following:*

1. *The multiplicity k of the eigenvalue 0 of both L_{rw} and L_{sym} equals the number of connected components $A_1, A_2, ..., A_k$ in G.*
2. *For L_{rw} the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \mathbb{1}_{A_2}, ...\mathbb{1}_{A_k}$ of those components.*
3. *For L_{sym} the eigenspace of eigenvalue 0 is spanned by the vectors $D^{1/2}\mathbb{1}_{A_1}$, $D^{1/2}\mathbb{1}_{A_2}, ..., D^{1/2}\mathbb{1}_{A_k}$.*

Proof: The proof follows the same line of arguments as the proof of proposition 4.2.

4.3 Spectral clustering algorithms

We are now ready to state the two most commonly used spectral clustering algorithms [Lux07], one using the unnormalized graph Laplacian and one for using normalized graph Laplacian L_{rw}. In both cases we assume that s is a symmetric similarity measure.

Algorithm 3 Unnormalized spectral clustering

Input: $X_1, X_2, ..., X_n \in \mathbb{R}^p$, $K \in \mathbb{N}$ (`Number of desired clusters`)

1: `Construct the similarity graph` $G = (V, E)$ `and the adjacency matrix` W `of` $X_1, X_2, ..., X_n$ `with respect to` s
2: `Compute the unnormalized graph Laplacian`
3: `Compute the first` K `eigenvectors` $u_1, u_2, ..., u_K$ `of` L
4: `Let` $U \in \mathrm{mat}_{n\times K}(\mathbb{R})$ `be the matrix with` $u_1, u_2, ..., u_k$ `as columns`
5: `Let` $y_i \in \mathbb{R}^K$ `for` $i = 1, 2, ..., n$ `be the` i^{th} `row of` U
6: `Use the K-means algorithm to cluster the points` $(y_i)_{i=1,...,n} \in \mathbb{R}^K$ `into clusters` $C_1, C_2, ..., C_K$

Output: Clusters $A_1, A_2, ..., A_K$ where $A_i = \{j | y_j \in C_i\}$.

Note that the next *normalized* version of spectral clustering uses the generalized eigenvectors of L, which by proposition 4.3 are the corresponding eigenvectors of L_{rw}. The algorithm is stated as follows:

Algorithm 4 Normalized spectral clustering using L_{rw}

Input: $X_1, X_2, ..., X_n \in \mathbb{R}^p$, $K \in \mathbb{N}$ (`Number of desired clusters`)

1: `Construct the similarity graph` $G = (V, E)$ `and the adjacency matrix` W `of` $X_1, X_2, ..., X_n$ `with respect to` s
2: `Compute the unnormalized graph Laplacian`
3: `Compute the first` K `generalized eigenvectors` $u_1, u_2, ..., u_K$ `of the generalized eigenvalue problem` $Lu = \lambda Du$
4: `Let` $U \in \mathrm{mat}_{n\times K}(\mathbb{R})$ `be the matrix with` $u_1, u_2, ..., u_k$ `as columns`
5: `Let` $y_i \in \mathbb{R}^K$ `for` $i = 1, 2, ..., n$ `be the` i^{th} `row of` U
6: `Use the K-means algorithm to cluster the points` $(y_i)_{i=1,...,n} \in \mathbb{R}^K$ `into clusters` $C_1, C_2, ..., C_K$

Output: Clusters $A_1, A_2, ..., A_K$ where $A_i = \{j | y_j \in C_i\}$.

The algorithms seem quite similar and the only real difference is the choice of graph Laplacian. But as we will see in the following section this choice leads to slightly different interpretations of the clustering when viewed from a graph cut perspective.

4.4 Justifying the use of spectral clustering

From the above stated algorithms it is not at all clear why spectral clustering often outperforms both the K-means algorithm and the EM-algorithm for Gaussian mixtures. Since it is not a model-based algorithm, we need to justify or argue that both algorithms actually are clever ways to find groups in a complex data set. To do so, we need the Courant-Fisher theorem and a related result, which we will state and prove in the following section. Furthermore, the Courant-Fisher theorem plays a key role in principal component analysis (PCA), which we will use later on to visualize the results of the different clustering algorithms on higher dimensional data sets.

4.4.1 The Courant-Fisher theorem and PCA

In the process of justifying spectral clustering as a valid clustering method we are interested in the maximization of something called the *Rayleigh quotient*, which for a vector X in $\mathbb{R}^n$ and an $n \times n$ matrix A is given by

$$\frac{X^T A X}{X^T X}. \tag{4.7}$$

The maximum of the *Rayleigh quotient* can be found as follows:

Theorem 4.1 (The Courant-Fisher theorem)**.** *Let $A \in Mat_{n\times n}(\mathbb{R})$ be a symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$ and corresponding standardized eigenvectors $o_{(i)}$ for $i = 1, 2, ..., n$. Then*

$$\max_{X\in\mathbb{R}^n\setminus\{0\}} \frac{X^T A X}{X^T X} = \max_{||X||=1,\ X\in\mathbb{R}^n} X^T A X = \lambda_1,$$

and for $i = 2, ..., n$

$$\max_{\substack{X\in\mathbb{R}^n\setminus\{0\} \\ o_{(1)}^T X=0,...,o_{(i-1)}^T X=0}} \frac{X^T A X}{X^T X} = \max_{\substack{||X||=1,\ X\in\mathbb{R}^n \\ o_{(1)}^T X=0,...,o_{(i-1)}^T X=0}} X^T A X = \lambda_i.$$

In both cases the maximum value is obtained with $o_{(i)}$ as argument.

The proof follows [And21]. Since A is symmetric it has an eigendecomposition and can therefore be written as

$$A = O \Delta O^T, \tag{4.8}$$

where $\Delta = diag\{\lambda_1, \lambda_2, ..., \lambda_n\}$ is a diagonal matrix of its eigenvalues and $O = \left\{o_{(1)}\ o_{(2)}\ ...\ o_{(n)}\right\}$ is an orthogonal matrix of the corresponding eigenvectors.
Using this, we can rewrite the numerator as

$$X^T A X = X^T O \Delta O^T X = \sum_{i=1}^{n} \lambda_i \left(X^T o_{(i)}\right)\left(o_{(i)}^T X\right) = \sum_{i=1}^{n} \lambda_i ||o_{(i)}^T X||^2, \tag{4.9}$$

and similarly since O is an orthogonal matrix the denominator can be written as

$$X^T X = X^T O O^T X = \sum_{i=1}^{n} ||o_{(i)}^T X||^2. \tag{4.10}$$

Using this, we get the following inequalities

$$\lambda_n X^T X = \sum_{i=1}^{n} \lambda_n ||o_{(i)}^T X||^2 \ \leq\ \sum_{i=1}^{n} \lambda_i ||o_{(i)} X||^2 = X^T A X \ \leq\ \sum_{i=1}^{n} \lambda_1 ||o_{(i)} X||^2 = \lambda_1 X^T X, \tag{4.11}$$

which we can use to bound the *Rayleigh quotient* between the smallest and largest eigenvalue by dividing with $X^T X$, thus finding that

$$\lambda_n \leq \frac{X^T A X}{X^T X} \leq \lambda_1. \tag{4.12}$$

Now noting that for all $i = 1, 2, ..., n$,

$$\frac{o_{(i)}^T A o_{(i)}}{\underbrace{o_{(i)}^T o_{(i)}}_{=1}} = o_{(i)}^T O \Delta O^T o_{(i)} = e_i^T \Delta e_i = \lambda_i, \tag{4.13}$$

where e_i denotes the i^{th} standard unit vector, we find that the maximum of the *Rayleigh quotient* is λ_1 and the value is obtained for $X = o_{(1)}$.
We now prove the second statement in the case $i = 2$ and from this it will be clear that the rest follows using the exact same arguments. We now add the constraint that $o_{(1)}^T X = 0$, which means that

$$X^T A X = \sum_{i=1}^{n} \lambda_i ||o_{(i)}^T X||^2 = \sum_{i=2}^{n} \lambda_i ||o_{(i)}^T X||^2, \tag{4.14}$$

thus the dominating term of the sum is removed. Using this we get the following inequalities

$$\lambda_n X^T X = \sum_{2=1}^{n} \lambda_n ||o_{(i)}^T X||^2 \ \leq \ \sum_{i=2}^{n} \lambda_i ||o_{(i)} X||^2 = X^T A X \ \leq \ \sum_{2=1}^{n} \lambda_1 2 ||o_{(i)} X||^2 = \lambda_2 X^T X, \tag{4.15}$$

which we again can use to bound the *Rayleigh quotient* between the smallest and the second largest eigenvalue, such that

$$\lambda_n \leq \frac{X^T A X}{X^T X} \leq \lambda_2. \tag{4.16}$$

By (4.13) the maximum value λ_2 is obtained with $X = o_{(2)}$, which also satisfies $o_{(1)}^T o_{(2)} = 0$.
The cases with $i > 2$ follows using the same line of arguments using that O is an orthogonal matrix which means that $o_{(i)}^T o_{(j)} = 0$ for $i \neq j$.
Now note that for $\alpha \neq 0$

$$\frac{X^T A X}{X^T X} = \frac{\alpha X^T A \alpha X}{\alpha X^T \alpha X}, \tag{4.17}$$

which means that the constraint $||x|| = 1$ does not affect the result of the maximization, thus finishing the proof. In fact, the same is true for any real constant $c \neq 0$ which will be important in the next section.

Besides the Courant-Fisher theorem, we will also need a closely related result about a trace maximization.

Theorem 4.2 (Trace maximization)**.** *Let $A \in Mat_{n\times n}(\mathbb{R})$ be a symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$ and corresponding eigenvectors $o_{(i)}$ for $i = 1, 2, ..., n$ and let $1 \leq k \leq n$. Then*

$$\max_{\substack{U \in Mat_{n\times k}(\mathbb{R}) \\ U^T U = I_k}} Tr\left(U^T A U\right) = \sum_{i=1}^{k} \lambda_i,$$

and similarly

$$\min_{\substack{U \in Mat_{n\times k}(\mathbb{R}) \\ U^T U = I_k}} Tr\left(U^T A U\right) = \sum_{i=n-k+1}^{n} \lambda_i.$$

The proof of the theorem is outside the scope of this project, but can be found in [HJ85].

Principal component analysis

Before introducing the different intuitions behind the two spectral clustering algorithms we briefly discuss a practical use of the Courant-Fisher theorem namely principal component analysis in short; PCA. This will come in handy in section 5.3 when we want to visualize clusterings of data with 784 features. We therefore introduce principal components from the perspective of dimensionality reduction as trying to minimize the distance of the orthogonal projection of the data onto a lower-dimensional subspace, but as we will also see this corresponds to maximizing the variance of the reduced data. We will only be using PCA to visualize data and will therefore only introduce principal components from an empirical perspective. For a more thorough introduction see [JW07].

Consider n data points $X_1, X_2, ..., X_n \in \mathbb{R}^p$ and assume that the data is standardized (The requirement of standardized data is also elaborated in [JW07]). First, we are interested in finding the one-dimensional subspace U, that represents our data in the best way such that if P denotes the orthogonal projection on U, then P minimizes

$$\sum_{i=1}^{n} ||X_i - PX_i||^2. \tag{4.18}$$

U will be on the form $U = span\,(u)$ for an $u \in \mathbb{R}^p$ and we can therefore assume that $u^T u = 1$ which further implies that $P = uu^T$. The minimization of (4.18) is unaffected by a division with n which we do to get a minimization involving the known loss function *Mean Square Error* or in short; MSE. Letting $Z_i := u^T X_i$, we can thereby reformulate the problem as minimizing

$$\begin{aligned} MSE\,(u) &:= \frac{1}{n}\sum_{i=1}^{n} ||X_i - uu^T X_i||^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} X_i^T X_i - 2X_i^T uu^T X_i + X_i^T uu^T uu^T X_i \\ &= \frac{1}{n}\sum_{i=1}^{n} X_i^T X_i - \left(u^T X_i\right)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} X_i^T X_i - \frac{1}{n}\sum_{i=1}^{n} Z_i^2. \end{aligned} \tag{4.19}$$

From this we see that minimizing $MSE\,(u)$ is equivalent to maximizing $\frac{1}{n}\sum_{i=1}^{n} Z_i^2$. Due to the standardization of the data the mean of Z_i is 0 for all $i = 1, 2, ..., n$ and thereby maximizing $\frac{1}{n}\sum_{i=1}^{n} Z_i^2$ is in fact the same as maximizing the empirical variance of $Z_1, Z_2, ..., Z_n$. We want to use the Courant-Fisher theorem and therefore we introduce the data-matrix $X \in \mathrm{mat}_{n\times p}\,(\mathbb{R})$ with X_i in the i^{th} row. Letting $Z = Xu$ and $\hat{\Sigma} = \frac{X^T X}{n}$, we can rewrite

$$\frac{1}{n}\sum_{i=1}^{n} Z_i^2 = \frac{Z^T Z}{n} = \frac{(Xu)^T (Xu)}{n} = u^T \frac{X^T X}{n} u = u^T \hat{\Sigma} u. \tag{4.20}$$

Now by the Courant-Fisher theorem, we see that (4.20) is maximized by the standardized first eigenvector of $\hat{\Sigma}$, $\hat{o}_1$ with corresponding eigenvalue $\hat{\lambda}_1$. Furthermore, the theorem tells us that if we want to maximize (4.20) with the assumption that $u^T \hat{o}_{(1)} = 0$, we find the maximum with $\hat{o}_{(2)}$ as argument and so on. Finally, we notice that for $i \neq j$ we have $\hat{o}_{(i)}\hat{\Sigma}\hat{o}_{(j)} = \hat{o}_{(i)}\hat{\lambda}_j\hat{o}_{(j)} = \hat{\lambda}_j\hat{o}_{(i)}\hat{o}_{(j)} = 0$, which means that the empirical covariance, $\hat{o}_{(i)}\hat{\Sigma}\hat{o}_{(j)}/n$, between $X\hat{o}_{(i)}$ and $X\hat{o}_{(j)}$ is zero [And21]. From this we can now define the empirical principal components as:

Definition 4.7 (*Empirical principal components*)**.** For $X_1, X_2, ..., X_n \in \mathbb{R}^p$, let X denote the data-matrix with X_i as the i^{th} row.

> The vector Xu_1 that maximizes the empirical variance of Xu_1 with $u_1^T u_1 = 1$ is called the *first* empirical principal component of X.
>
> For $i \geq 2$, the vector Xu_i that maximizes the empirical variance of Xu_i with $u_i^T u_i = 1$ and with the empirical covariance between Xu_i and Xu_j being zero for $i \neq j$, is called the i^{th} empirical principal component of X.

Using this definition we can thereby find the d-dimensional subspace that 'represents' our data in the best way i.e. minimizes the distance of the orthogonal projection of the data onto the subspace or equivalently maximized the variance of the projected data. As we will see in section 5.3, we can thereby visualize high-dimensional data by plotting the data projected onto the two-dimensional subspace spanned by the first two empirical principal components.

4.4.2 Graph cut point of view

With the Courant-Fisher and trace maximization theorems in our toolbox we are now ready to introduce some different intuitions to why the two spectral clustering algorithms actually are meaningful clustering methods. In this section we will consider the clustering problem from a graph cut point of view. The intuition of this is the following: Given a similarity graph we want to find a partition or a cut through the graph such that the edges between points in different groups have low weights and the edges between points within the groups have high weights. This can be formalized as trying to solve the *mincut* problem, which for a given similarity graph $G = (V, E)$ and a number of clusters K can be formulated using definition 4.5 as minimizing

$$cut\left(A_1, A_2, ..., A_K\right) := \frac{1}{2}\sum_{i=1}^{K} W\left(A_i, A_i^c\right), \tag{4.21}$$

over $A_i \subset V$ for $i = 1, 2, ..., K$, where the factor $1/2$ is added as not to count every edge twice.
A problem with this approach is that the solution to the *mincut* problem often is groups with only one point which in many cases is not the desired outcome [Lux07]. A way to prevent this problem is to introduce some kind of weighing regarding the 'size' of the groups. The two most used ways of preventing this problem are to weigh with either the size i.e. $|A|$ of each group or with the volume i.e. $vol\left(A\right)$ of each group. Using this, our objective is now to minimize either

$$\text{RatioCut}\left(A_1, A_2, ..., A_K\right) := \frac{1}{2}\sum_{i=1}^{K} \frac{W\left(A_i, A_i^c\right)}{|A_i|} = \sum_{i=1}^{K} \frac{cut\left(A_i, A_i^c\right)}{|A_i|}, \tag{4.22}$$

or

$$\text{Ncut}\left(A_1, A_2, ..., A_K\right) := \frac{1}{2}\sum_{i=1}^{K} \frac{W\left(A_i, A_i^c\right)}{vol\left(A_i\right)} = \sum_{i=1}^{K} \frac{cut\left(A_i, A_i^c\right)}{vol\left(A_i\right)}. \tag{4.23}$$

Unfortunately, adding this 'normalization' of the groups in both cases make the minimization problem NP-hard to solve [Lux07]. Instead we will try to solve a relaxed version of both minimization problems and as we will see this will lead to the unnormalized spectral clustering algorithm in the case of *RatioCut* and to the normalized spectral clustering algorithm using L_{sym} in the case of *Ncut*. Thus, the assumption when using spectral clustering is that the relaxed solutions of the minimization problems are somewhat close to actual solutions, which in many cases turn out to be true, but in general there is no guarantee to this and it can even be shown that the difference between the two minima can be arbitrarily large [Lux07].

Relaxing RatioCut for $K = 2$

We will start by relaxing RatioCut in the case simple with $K = 2$, since this will allow us to better understand the case of a general $K \geq 2$. In the case of $K = 2$ our goal is to find a solution to

$$\min_{A \subset V} \text{RatioCut}\left(A, A^c\right). \tag{4.24}$$

Given subset $A \subset V$, we want to rewrite this problem. To do so, we define the vector $u \in \mathbb{R}^n$ as

$$u_i = \begin{cases} \sqrt{\frac{|A^c|}{|A|}}, & \text{if } v_i \in A \\ -\sqrt{\frac{|A|}{|A^c|}}, & \text{if } v_i \in A^c \end{cases} \qquad \text{for } i = 1, 2, ..., n. \tag{4.25}$$

Now letting $L = D - W$ be the unnormalized graph Laplacian and using proposition 4.1 and the fact that $u_i - u_j = 0$ if v_i and v_j are in the same connected component, we have the following:

$$\begin{aligned} u^T L u &= \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}\left(u_i - u_j\right)^2 \\ &= \frac{1}{2}\sum_{i\in A, j\in A^c} W_{ij}\left(\sqrt{\frac{|A^c|}{|A|}} + \sqrt{\frac{|A|}{|A^c|}}\right)^2 + \frac{1}{2}\sum_{i\in A^c, j\in A} W_{ij}\left(-\sqrt{\frac{|A|}{|A^c|}} - \sqrt{\frac{|A^c|}{|A|}}\right)^2 \\ &= \frac{1}{2}\left(\sum_{i\in A, j\in A^c} W_{ij} + \sum_{i\in A^c, j\in A} W_{ij}\right)\left(\frac{|A^c|}{|A|} + \frac{|A|}{|A^c|} + 2\right) \\ &= cut\left(A, A^c\right)\left(\frac{|A^c| + |A|}{|A|} + \frac{|A| + |A^c|}{|A^c|}\right) \\ &= |V| \cdot \left(\frac{cut\left(A, A^c\right)}{|A|} + \frac{cut\left(A^c, A\right)}{|A^c|}\right) = n \cdot RatioCut\left(A, A^c\right). \end{aligned} \tag{4.26}$$

Note also that the vector u is orthogonal to the constant one vector, since

$$< u, \mathbb{1} >= \sum_{i=1}^{n} u_i = \sum_{i\in A}\sqrt{\frac{|A^c|}{|A|}} - \sum_{i\in A^c}\sqrt{\frac{|A|}{|A^c|}} = |A| \cdot \sqrt{\frac{|A^c|}{|A|}} - |A^c| \cdot \sqrt{\frac{|A|}{|A^c|}} = 0, \tag{4.27}$$

and that the squared norm of u is equal to the number of data points n, since

$$||u||^2 = \sum_{i=1}^{n} u_i^2 = \sum_{i\in A}\frac{|A^c|}{|A|} + \sum_{i\in A^c}\frac{|A|}{|A^c|} = |A| \cdot \frac{|A^c|}{|A|} + |A^c| \cdot \frac{|A|}{|A^c|} = |A^c| + |A| = |V| = n. \tag{4.28}$$

Using equation (4.27) and (4.28) together with (4.26), we can rewrite the minimization problem in (4.22) in terms of u as

$$\min_{A\subset V} u^T L u, \quad \text{subject to } u \perp \mathbb{1},\ ||u|| = \sqrt{n},\ u \text{ defined as in (4.25).} \tag{4.29}$$

This equivalent discrete minimization problem is of course still NP-hard to solve, but as mentioned we will look for a relaxed version. In this case if we relax the discreteness of u, we will have a minimization problem that we can solve using the Courant-Fisher theorem. This gives us that the eigenvector corresponding to the second smallest eigenvalue of L minimizes $u^T L u$ under the given conditions (Since the smallest eigenvalue is 0 with eigenvector $\mathbb{1}$, which is not orthogonal to it self).
To get a clustering from the real-valued solution, u, to the minimization problem, we need to transform u into a discrete vector. This can be done easily by using the K-means algorithm to cluster the coordinates of u, which are just points in $\mathbb{R}$, into clusters C_1 and C_2 such that the clustering of the data points can be written as

$$\begin{cases} v_i \in A, & \text{if } u_i \in C_1 \\ v_i \in A^c, & \text{if } u_i \in C_2 \end{cases} \qquad \text{for } i = 1, 2, ..., n. \tag{4.30}$$

Looking back at the spectral clustering algorithms in section 4.3, we see that this is the unnormalized spectral clustering algorithm in the case $K = 2$.

Relaxing RatioCut for a general $K \geq 2$

Now for a general $K \geq 2$, we want to minimize (4.22) using a similar technique as in the case $K = 2$. For a given partition of V into K sets $A_1, A_2, ..., A_K$ we define K indicator vectors $h_j = \left(h_{1,j}, h_{2,j}, ..., h_{n,j}\right) \in \mathbb{R}^n$ as

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{|A_j|}}, & \text{if } v_i \in A_j \\ 0, & \text{otherwise} \end{cases} \qquad i = 1, 2, ..., n \quad j = 1, 2, ..., K. \tag{4.31}$$

Similar to the matrix U in the unnormalized spectral clustering algorithm we let $H = \{h_1\ h_2\ ...\ h_K\} \in \text{mat}_{n\times K}$ be the matrix containing the K indicator vectors as columns. We observe that $H^TH = I_n$ i.e. H is orthogonal and like in the case $K = 2$, we are interested in $h_i^T Lh_i$, for which we have the following

$$h_l^T Lh_l = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}\left(h_{i,l} - h_{j,l}\right)^2 = \frac{1}{2}\sum_{i\in A_l, j\in A_l^c} W_{ij}\frac{1}{|A_l|} + \frac{1}{2}\sum_{i\in A_l^c, j\in A_l} W_{ij}\frac{1}{|A_l|} = \frac{cut\left(A_l, A_l^c\right)}{|A_l|}. \tag{4.32}$$

Furthermore, we have that

$$h_l^T Lh_l = \left(H^T LH\right)_{ll}. \tag{4.33}$$

Combining (4.32) and (4.33), we can rewrite RatioCut as

$$RatioCut\left(A_1, A_2, ..., A_K\right) = \sum_{i=1}^{K} h_i^T Lh_i = \sum_{i=1}^{n}\left(H^T LH\right)_{ii} = Tr\left(H^T LH\right). \tag{4.34}$$

Thus, the minimization of RatioCut can be rewritten as

$$\min_{A_1,...,A_k} Tr\left(H^T LH\right), \quad \text{subject to } H^TH = I_n,\ H \text{ defined as in (4.31).} \tag{4.35}$$

Like in the case of $K = 2$, this is a discrete minimization problem that is still NP-hard to solve. But relaxing the discreteness of H leads to a minimization problem that can be solved using the trace maximization theorem by choosing H to be the matrix with the K eigenvectors corresponding to the K smallest eigenvalues of L as columns, which corresponds to the matrix U in the unnormalized spectral clustering algorithm. Like in the case of $K = 2$, we need to convert this real valued solution to a discrete one, which we can do by using the K-means algorithm to cluster the rows of H into clusters $C_1, C_2, ..., C_K$ and then use this to cluster the data points as

$$\begin{cases} v_i \in A_1, & \text{if } h_{i,1} \in C_1 \\ v_i \in A_2, & \text{if } h_{i,2} \in C_2 \\ \vdots & \\ v_i \in A_K, & \text{if } h_{i,K} \in C_K \end{cases} \qquad \text{for } i = 1, 2, ..., n. \tag{4.36}$$

Again, we see that finding a solution to the relaxed version of the minimization of *RatioCut* corresponds exactly to the unnormalized spectral clustering algorithm for a general $K \geq 2$.

Relaxing Ncut for $K = 2$

We will now show that relaxing the minimization of *Ncut* corresponds to spectral clustering in the normalized case using L_{rw}. For simplicity we will again first treat the case $K = 2$ in depth and then go over the case of a general $K \geq 2$ using the same kind of arguments. The line of arguments is similar to those of relaxing *RatioCut*, thus for at given subset $A \subset V$ we define the vector $u \in \mathbb{R}^n$ as

$$u_i = \begin{cases} \sqrt{\frac{vol(A^c)}{vol(A)}}, & \text{if } v_i \in A \\ -\sqrt{\frac{vol(A)}{vol(A^c)}}, & \text{if } v_i \in A^c \end{cases} \qquad \text{for } i = 1, 2, ..., n. \tag{4.37}$$

Using proposition 4.1 and the fact that $u_i - u_j = 0$ if v_i and v_j are in the same connected component with the same line of arguments as in (4.26), we get for $L = D - W$ that

$$u^T Lu = vol\left(V\right) \cdot Ncut\left(A, A^c\right). \tag{4.38}$$

Furthermore, we see that

$$u^T Du = \sum_{i\in A} \frac{vol\left(A^c\right)}{vol\left(A\right)} \cdot d_i + \sum_{i\in A^c} \frac{vol\left(A\right)}{vol\left(A^c\right)} \cdot d_i = vol\left(A^c\right) + vol\left(A\right) = vol\left(V\right), \tag{4.39}$$

and that

$$(Du)^T \mathbb{1} = u^T (D\mathbb{1}) = \sum_{i \in A} \sqrt{\frac{vol(A^c)}{vol(A)}} \cdot d_i - \sum_{i \in A^c} \sqrt{\frac{vol(A)}{vol(A^c)}} \cdot d_i = 0. \tag{4.40}$$

Combining (4.38), (4.39) and (4.40), we can rewrite the minimization of *Ncut* as

$$\min_{A \subset V} u^T L u, \quad \text{subject to } Du \perp \mathbb{1},\ u^T D u = vol(V),\ u \text{ defined as in (4.37).} \tag{4.41}$$

Relaxing the discreteness of u and substituting $g := D^{1/2}u$, the problem can be written as

$$\min_{g \in \mathbb{R}^n} g^T \underbrace{D^{-1/2} L D^{-1/2}}_{L_{sym}} g, \quad \text{subject to } g \perp D^{1/2}\mathbb{1},\ ||g||^2 = vol(V). \tag{4.42}$$

Since $D^{1/2}\mathbb{1}$ is the eigenvector to 0 of L_{sym}, which is the smallest eigenvalue, and $vol(V)$ is constant, the minimization is on the standard form that we can solve using the Courant-Fisher theorem. The solution g is the eigenvector to the second smallest eigenvalue of L_{sym} and we can rewrite this such that $u = D^{-1/2}g$, which by proposition 4.3 is the eigenvector to the second smallest eigenvalue of L_{rw}. Once again we then use the K-means algorithm to cluster the coordinates of u and in the same way as in *RatioCut* use the clustering of coordinates to determine the clustering of the data points.

Relaxing Ncut for a general $K \geq 2$

For a general $K \geq 2$, we use a similar technique as in the case $K = 2$. Given a partition of V into K sets $A_1, A_2, ..., A_K$, we define K indicator vectors $h_j = \left(h_{1,j}, h2, j, ..., h_{n,j}\right) \in \mathbb{R}^n$ as

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}}, & \text{if } v_i \in A_j \\ 0, & \text{otherwise} \end{cases} \qquad i = 1, 2, ..., n \quad j = 1, 2, ..., K. \tag{4.43}$$

We then let $H = \{h_1\ h_2\ ...\ h_K\} \in \text{mat}_{n \times K}$ be the matrix containing the K indicator vectors as columns. We observe that $H^T H = I_n$ i.e. H is an orthogonal matrix and that

$$h_l^T D h_l = \sum_{i \in A_l} \frac{1}{vol(A_l)} \cdot d_i = 1. \tag{4.44}$$

Furthermore, we can use the same line of arguments as in (4.26) to see that

$$h_l^T L h_l = \frac{cut\left(A_l, A_l^c\right)}{vol(A_l)}. \tag{4.45}$$

Thus, we can rewrite the minimization of *Ncut* as

$$\min_{A_1, A_2, ..., A_K} Tr\left(H^T L H\right), \quad \text{subject to } H^T D H = I_n,\ H \text{ defined as in (4.43).} \tag{4.46}$$

By relaxing the discreteness of H and by substituting $S = D^{1/2}H$, we can formulate the relaxed minimization as

$$\min_{S \in \text{mat}_{n \times K}} Tr\left(S^T \underbrace{D^{-1/2} L D^{-1/2}}_{L_{sym}} S\right), \quad \text{subject to } S^T S = I_n, \tag{4.47}$$

which as a consequence of the trace maximization theorem can be solved by the matrix S with the K eigenvectors corresponding to the K smallest eigenvalues of L_{sym} as columns. Again, we can rewrite this such that $H = D^{-1/2}S$, which we by proposition 4.3 see has the K eigenvectors to the K smallest eigenvalues as columns. Now using the K-means algorithm to cluster the rows of H and using this to cluster the data points, we see that the relaxed version of minimizing *Ncut* corresponds to the normalized spectral clustering algorithm using L_{rw}.

4.4.3 Random walk point of view

A different way of considering the clustering problem is to consider a random walk on the similarity graph, where the transition probabilities are proportional to the weights. From this perspective a *good* clustering is a partition of the graph such that the random walk stays within the same cluster as long as possible and only seldom jumps to another cluster. The intuition here is close to the intuition of Ncut, since a balanced partition of the graph with a low cut also means that a random walk does not have many possibilities of jumping to a different cluster and when it is possible the probability should be small. The probability of jumping from v_i to v_j is for a random walk given by $p_{ij} := w_{ij}/d_i$. The transition matrix $P = \left(p_{ij}\right)_{i,j=1,2,...,n}$ of the random walk is then given as

$$P = D^{-1}W. \tag{4.48}$$

Furthermore, if the similarity graph is connected and non-bipartite there exists a unique stationary distribution $\pi = (\pi_1, \pi_2, ..., \pi_n)^T$, where $\pi_i = d_i/vol\,(V)$ [Lux07]. The properties of P are of course closely related to the properties of L_{rw} since $L_{rw} = I - P$. A direct connection between the random walk perspective and Ncut is stated in the following proposition [MS01]:

Proposition 4.5 (*Ncut via transition probabilities*)**.** *Let G be a connected, non-bipartite similarity graph. Assume the $(X_t)_{t\in\mathbb{N}}$ is a random walk on G starting with X_0 in the stationary distribution π. For disjoint subsets $A, B \subset V$, let $\mathbb{P}\left(B|A\right) := \mathbb{P}\left(X_1 \in B|X_0 \in A\right)$. Then*

$$Ncut\,(A^c, A) = \mathbb{P}\left(A^c|A\right) + \mathbb{P}\left(A|A^c\right).$$

Proof: We first observe for $A, B \subset V$ that

$$\mathbb{P}\left(X_0 \in A, X_1 \in B\right) = \sum_{i\in A, j\in B} \pi_i p_{ij} = \sum_{i\in A, j\in B} \frac{d_i}{vol\,(V)} \frac{w_{ij}}{d_i} = \frac{1}{vol\,(V)} \sum_{i\in A, j\in B} w_{ij}. \tag{4.49}$$

From this we see that

$$\mathbb{P}\left(X_1 \in B|X_0 \in A\right) = \frac{\mathbb{P}\left(X_0 \in A, X_1 \in B\right)}{\mathbb{P}\left(X_0 \in A\right)} = \frac{\frac{1}{vol(V)} \sum_{i\in A, j\in B} w_{ij}}{\frac{vol(A)}{vol(V)}} = \frac{2 \cdot cut\,(A, B)}{vol\,(A)}, \tag{4.50}$$

which gives the connection to Ncut as

$$Ncut\,(A, A^c) = \frac{cut\,(A, A^c)}{vol\,(A)} + \frac{cut\,(A^c, A)}{vol\,(A^c)} \propto \mathbb{P}\left(A^c|A\right) + \mathbb{P}\left(A|A^c\right). \tag{4.51}$$

Thereby the minimization of Ncut can also be interpreted as trying to find a cut that minimized the probability of transitioning between clusters when performing a random walk on the similarity graph.

Other ways of trying to justify the use of spectral clustering has been made by for instance considering the commute time distance of a random walk, where we consider the expected time it takes for a random walk to go from v_i to v_j and back, or by considering L as a perturbed version of the ideal case with zero between-cluster similarity. Intuitively, both methods seem as clever ways to consider the clustering problem, but the direct connection to spectral clustering is not as clear as one might have hoped and the existing theory, which is stated but not proved in [Lux07], is beyond the scope of this study.

5 Comparing the methods

In this section we will try to compare the three different clustering algorithms through concrete data examples. In general it will be hard to say that one algorithm is better than the other, since it will depend on the concrete data example and sometimes also the amount of data. Furthermore, clustering algorithms are used in an unsupervised learning setup which means that we cannot compare the output of an algorithm with the *true* clustering. And even if we knew the true clustering we would still need a way to compare two clusterings to be able to compare the outcome of the algorithms. In the literature there are many different ways of doing so, but one of the most popular methods is the *adjusted Rand index* [HA85].

5.1 Rand index and adjusted Rand index

To define the adjusted Rand index, we first define the *Rand index* and motivate the ideas and considerations behind this and use this to define the adjusted Rand index. First, note that given a cluster of points this also tells us which points do not belong to this cluster. Second, note that all points in a data set are of equal importance. Using this, an intuitive way of comparing two clusterings would be to compare how pairs of points are clustered. If a pair of points either is in the same cluster or in different clusters in both of the clusterings, this would mean some kind of similarity of the two clusterings and a dissimilarity in the opposite case [Ran71]. Thus, given a data set $X_1, X_2, ..., X_n \in \mathbb{R}^p$ and two clusterings of the data set $C = \{C_1, C_2, ..., C_{K_1}\}$ and $C' = \left\{C'_1, C'_2, ..., C'_{K_2}\right\}$, we define

$$\alpha_{ij} = \begin{cases} 1, & \text{if } \exists k, k' \text{ } s.t. \text{ } X_i, X_j \in C_k \text{ and } X_i, X_j \in C'_{k'} \\ 0, & \text{otherwise,} \end{cases} \tag{5.1}$$

to indicate if two points are in the same group in both clusterings and

$$\beta_{ij} = \begin{cases} 1, & \text{if } \exists k, k' \text{ } s.t. \text{ } X_i \in C_k, \text{ } X_j \notin C_k \text{ and } X_i \in C'_{k'}, \text{ } X_j \notin C'_{k'} \\ 0, & \text{otherwise,} \end{cases} \tag{5.2}$$

to indicate if two points are in different groups in both clusterings. If either $\alpha_{ij} = 1$ or $\beta_{ij} = 1$, we say that the clusterings C and C' *agree* on X_i and X_j. Using the notion of α and β, we define the Rand index as

$$RI\left(C, C'\right) := \sum_{\substack{i,j=1 \\ i<j}}^{n} \frac{\alpha_{ij} + \beta_{ij}}{\binom{n}{2}}.$$

Notice that $R\left(C, C'\right) \in [0, 1]$, where the value 0 is obtained if the two clusterings do not agree on any pair of points and the value 1 is obtained if they agree on all pairs of points. Even though it is possible for the Rand index of two clusterings to be zero, it is highly unlikely that they would not agree on any pair of points. In some sense we would like it to be zero if a randomly determined clustering is compared to the *true* clustering. To correct for this, the adjusted Rand index is proposed in [HA85] as an alternative way of measuring the similarity of two clusterings, where we take the expected value of the Rand index into account (where the clusterings are chosen at random according to the sizes of the clusters in C and C'). Informally the adjusted Rand index can be defined as

$$ARI\left(C, C'\right) := \frac{RI\left(C, C'\right) - \mathbb{E}\left[RI\left(C, C'\right)\right]}{\underset{C,C'}{max}\ RI\left(C, C'\right) - \mathbb{E}\left[RI\left(C, C'\right)\right]}. \tag{5.3}$$

Here we see that the adjusted Rand index is zero if the Rand index is equal to its expected value, thus for a random clustering the adjusted Rand index is zero. Note that the adjusted Rand index also can be negative if the Rand index is less than the expected value of two random clusterings, meaning that the clusterings are more dissimilar than two random clusterings. A concrete way of calculating the adjusted Rand index is derived in [HA85] using the generalized hypergeometric distribution, where the mean of the Rand index is derived through a result from [Hub77], but this is outside the scope of this project.

5.2 Comparison in general

As we have seen there are both advantages and disadvantages of all three types of clustering algorithms. In the following sections we try to construct an overview of these regarding the different algorithms. To simplify the comparison, we will only be using the unnormalized version of spectral clustering with the k-nearest neighbors graph with different values of k. For a further comparison of the different spectral clustering algorithms and the choice of similarity graphs, see [Lux07]. To illustrate the strengths and weaknesses of the different algorithms, we have plotted their clusterings and corresponding adjusted Rand index compared to the true groups in figure 5. Note that the colors of the groups are irrelevant since the algorithms are unsupervised.

As shown in the first column of the figure the K-means algorithm performs well on data sets with groups that are 'simple' in the sense that the groups are nicely separated with respect to euclidean distance and with 'nice' shapes in the sense that they are convex.

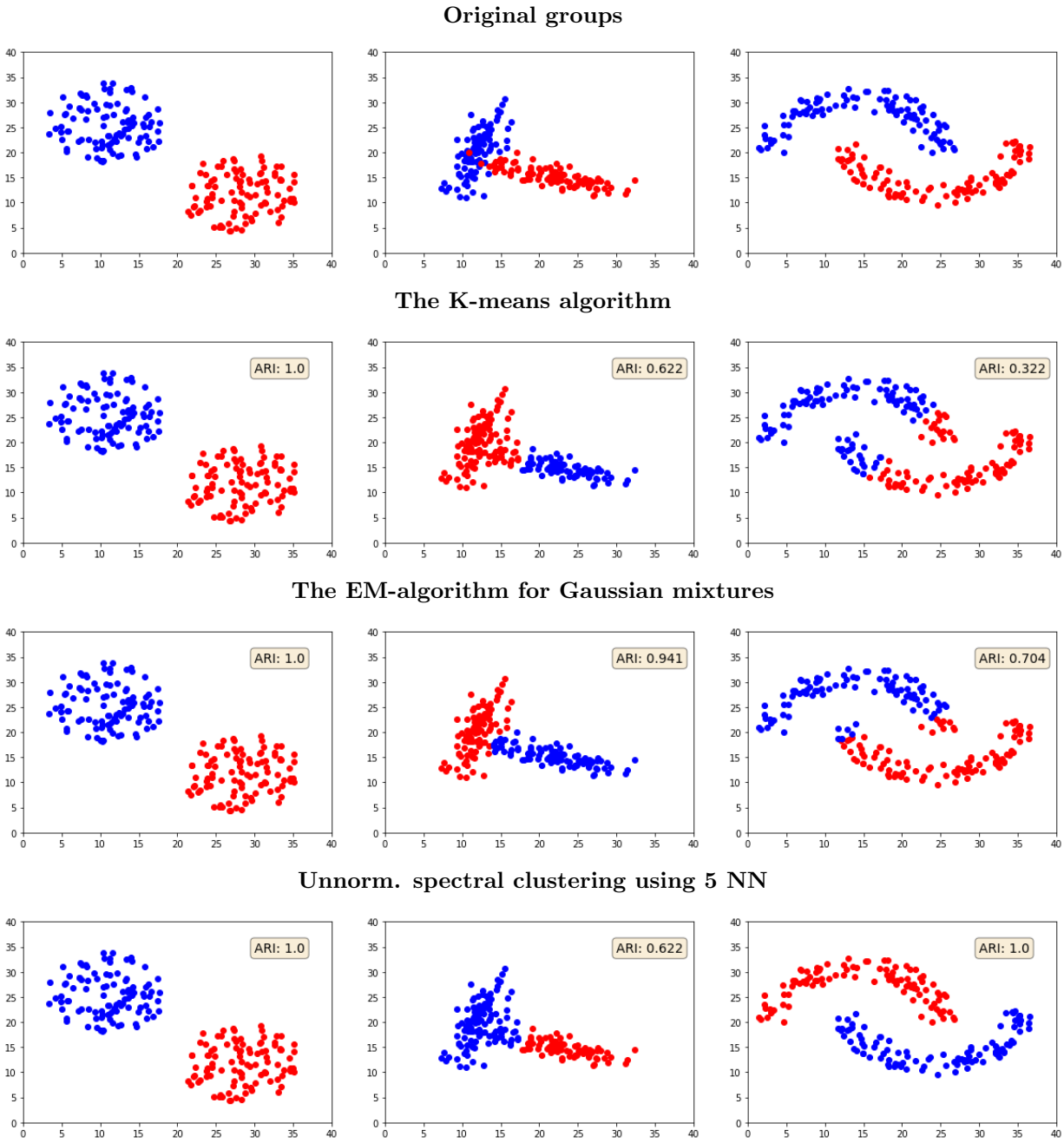

Figure 5: Comparison of all three algorithms using the adjusted Rand index (defined in section 5.1) of the clustering and the original groups from the simulated data.

In this simple setup with two nicely separated groups all three algorithms manages to detect both groups completely. Looking at the second column of figure 5, we see that the K-means algorithm is outperformed by the EM-algorithm for Gaussian mixtures. Here the two groups are samples from two different bi-variate Gaussians with respectively a positive and negative correlation between the components. Since the groups overlap, neither the K-means algorithm nor the unnormalized spectral clustering algorithm manages to classify the points correctly, but the EM-algorithm on the other hand has no problem with this because it tries to model the densities in each group. In the third column of figure 5, we see a classical example of a data set with two non-convex groups, where spectral clustering outperforms both the K-means algorithm and the EM-algorithm. By means of the similarity function, spectral clustering is modelling the local neighborhood of the points which means that shapes of the groups are irrelevant.

As mentioned, it is difficult to say in general that one algorithm will do better than the other on a specific data set. In these examples we were able to compare the results to the original groups using the adjusted Rand index, but since real clustering problems are unsupervised, we cannot compare to original groups or labels. The K-means algorithm produces stable results on *simple* data sets and has the advantage that it works fast even on large data sets. In general, the EM-algorithm for Gaussian mixtures performs well if it is a fair assumption that the data points are samples from a Gaussian mixture, but the convergence of the algorithm can be slow and numerical issues might arise when computing the *log* of really small values. In general, spectral clustering algorithms perform well on many different data sets and are especially well suited on data sets with more complex structures. As long as the similarity graph is sparse, spectral clustering can be implemented efficiently even on large data sets. The disadvantages of spectral clustering are that the choice of a good similarity function is quite complex and can affect the outcome of the algorithm a lot. Furthermore, many similarity functions require a choice of a hyper-parameter which also affects the outcome. Therefore, spectral clustering cannot be used as a 'black box' algorithm for any data set, which we will also see when we compare the algorithms on the MNIST data in the following section, but it is a good clustering tool when used with the right considerations [Lux07].

5.2.1 The K-means algorithm as a special case of the EM-algorithm for Gaussian mixtures

In section 3.4 we showed that the K-means algorithm comes out as a special case of the EM-algorithm namely if we restrict the covariance matrix to be on the form $\varepsilon \cdot I_2$ for a small enough ε. This is illustrated in figure 6 consisting of 300 samples from a Gaussian mixture with three components.

Clustering by the EM-algorithm with covariance restricted to $\boldsymbol{\varepsilon \cdot I_2}$

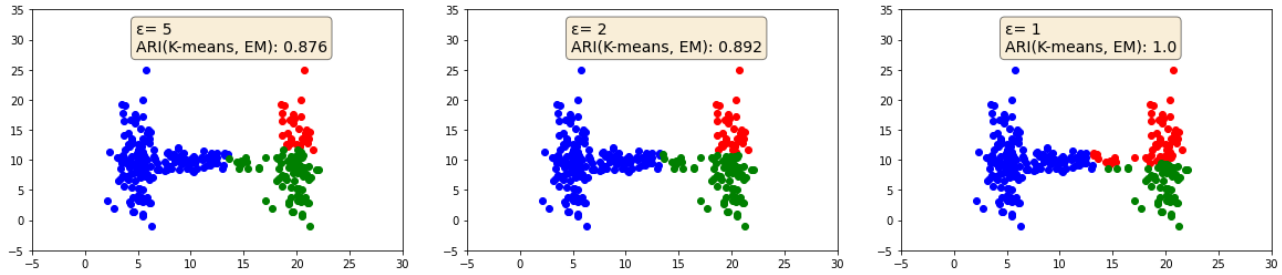

Figure 6: A visualization of the convergence of the EM-algorithm to the K-means algorithm (see section 3.4) with $\varepsilon = 5$ (Left), $\varepsilon = 2$ (Middle) and $\varepsilon = 1$ (Right). The clusterings of the restricted EM-algorithm and the K-means algorithm are compared using the adjusted Rand index.

By comparing the clustering produced by the K-means and the one produced by the modified EM-algorithm using the adjusted Rand index, we see that the clusterings 'converge' when ε gets smaller and since this is a fairly simple data set, we find in this case that $\varepsilon = 1$ is small enough for the clusterings to be identical. Informally; when the covariance matrices vanishes the 'Gaussianity' vanishes and only the means remain and the EM-algorithm is thus equivalent to the K-means algorithm.

5.3 Comparison on the MNIST data set

Using the Adjusted Rand index we will now compare the clusterings produced by the three different algorithms on the MNIST data set. The MNIST data set is a data set containing 60.000 images (28x28 pixels) of handwritten digits between 0 and 9 including their labels and a test data set of 10.000 images including labels [LC10]. Each of the 784 pixels has a value between 0 and 255, that is the color intensity in that pixel. A sample of 36 digits from the data set is shown in figure 7. The data set is widely used in the field of machine learning which is why we choose exactly this data set to compare the algorithms. Furthermore, an advantage of comparing the methods on a labelled data set is that we can compare the results of a clustering with the *true* values i.e. the labels using the adjusted Rand index, which in this case allows us to say that one algorithm did better than another on this concrete data set. We use the theory described in section 4.4.1 to visualize the data by projecting the data points onto the first two empirical principal components. The projection of all the digits 0, 2 and 9 among the first 10 000 digits is shown with their true labels in figure 7.

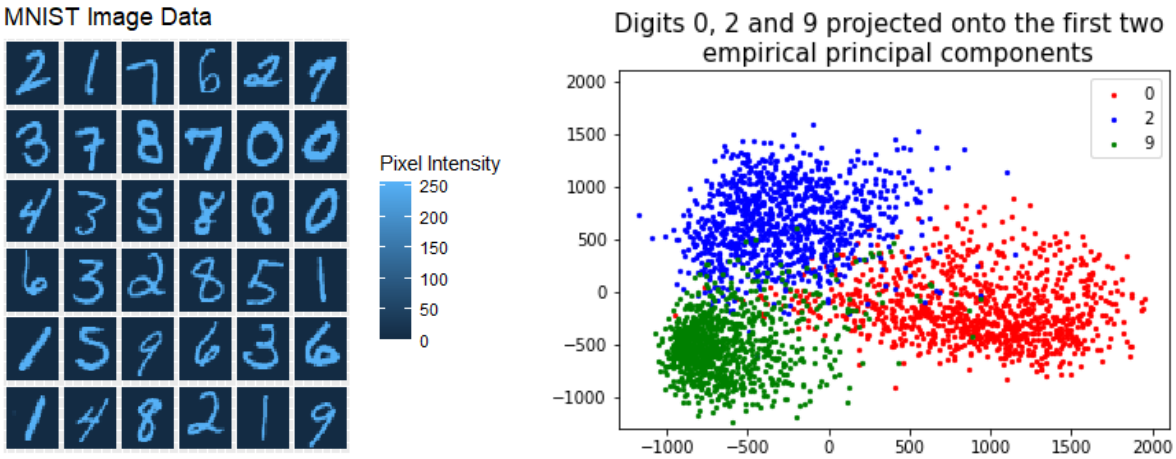

Figure 7: A visualization of a sample of images in the MNIST data set (Left). Projections of the digits 0, 2 and 9 among the first 10 000 data points onto the first two principal components, see section 4.4.1 (Right).

First, we compare the performance of the different algorithms on the first 10 000 digits. Due to the size of the data set (10 000 x 784) we use the algorithms in the Python module *scikit-learn* for comparison. Using this we also need to choose some hyper-parameters for the EM-algorithm and for spectral clustering. In the EM-algorithm in scikit-learn we can choose between different restrictions regarding the covariance matrix and in the spectral clustering algorithm using k-nearest neighbors as similarity graph, we need to choose a specific k. Since we in this case are able to compare the results to the true labels, we evaluate the adjusted Rand index of the clusterings with different values of hyper-parameters to find the best performance of the algorithms. The results are shown in table 1.

Performance of the algorithms on the first 10 000 digits				
K-means algorithm	*EM algorithm for Gaussian mixtures*		*Unnorm. spectral clustering (NN)*	
ARI(Clustering, labels)	Covariance type	ARI(Clustering, labels)	# NN	ARI(Clustering, labels)
0.341	Full	0.316	5	0.587
	Tied	0.130	10	0.570
	Diagonal	0.211	15	0.558
	Spherical	0.066	20	0.543

Table 1: Adjusted Rand index of the clusterings produced by the algorithms and the true labels from MNIST. The spectral clustering algorithm is initialized using the nearest neighbors graph.

The unnormalized spectral clustering algorithm has by far the best results while the K-means algorithm and the EM-algorithm using a 'full' covariance matrix, meaning that each component has its own covariance matrix just like the algorithm in section (3.3), almost get the same results. Next, we consider a subset of the first 10 000 digits namely the subset consisting of all 0, 2 and 9's. Again we consider different choices of hyper-parameters and consider the adjusted Rand index of the clustering and the true labels. The results on this subset are shown in table 2 and the clusterings are visualized in figure 8.

Performance of the algorithms on the digits 0, 2 and 9 (Among the first 10 000 digits)				
K-means algorithm	*EM algorithm for Gaussian mixtures*		*Unnorm. spectral clustering (NN)*	
ARI(Clustering, labels)	Covariance type	ARI(Clustering, labels)	# NN	ARI(Clustering, labels)
0.823	Full	0.635	5	0.961
	Tied	0.883	10	0.957
	Diagonal	0.421	15	0.960
	Spherical	0.343	20	0.963

Table 2: Adjusted Rand index of the clusterings produced by the algorithms and the true labels from MNIST on the digits 0, 2 and 9 among the first 10 000 digits. The spectral clustering algorithm is initialized using the nearest neighbors graph.

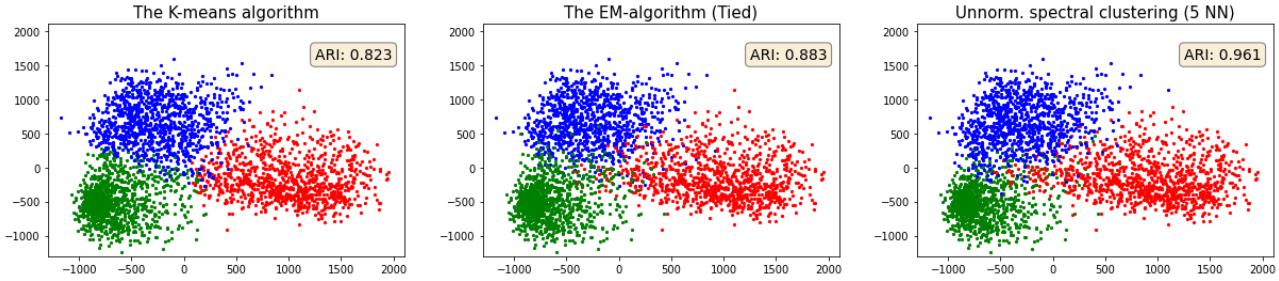

Figure 8: Clustering of all the digits 0, 2 and 9 among the first 10 000 digits by the three different algorithms projected onto the first two empirical principal components. The colors of the groups are set manually according to the labels for an easier visual comparison of the clusterings.

We notice that all three types of algorithms perform remarkably better, which we also would expect since it should be easier to detect three groups compared to detecting 10 groups. Again, the spectral clustering algorithm has the best results, while the K-means this time does a little worse than the best result of the EM-algorithm, that uses a 'tied' covariance matrix, meaning that all the components are restricted to share the same covariance matrix. Lastly, we consider the subset of all 0, 1 and 8's among the first 10 000 digits compared in the same way. The results are shown in table 3 and the clusterings are visualized in figure 9.

Performance of the algorithms on the digits 0, 1 and 8 (Among the first 10 000 digits)				
K-means algorithm	*EM algorithm for Gaussian mixtures*		*Unnorm. spectral clustering (NN)*	
ARI(Clustering, labels)	Covariance type	ARI(Clustering, labels)	# NN	ARI(Clustering, labels)
0.785	Full	0.761	5	0.568
	Tied	0.812	10	0.565
	Diagonal	0.527	15	0.565
	Spherical	0.196	20	0.560

Table 3: Adjusted Rand index of the clusterings produced by the algorithms and the true labels from MNIST on the digits 0, 1 and 8 among the first 10 000 digits. The spectral clustering algorithm is initialized using the nearest neighbors graph.

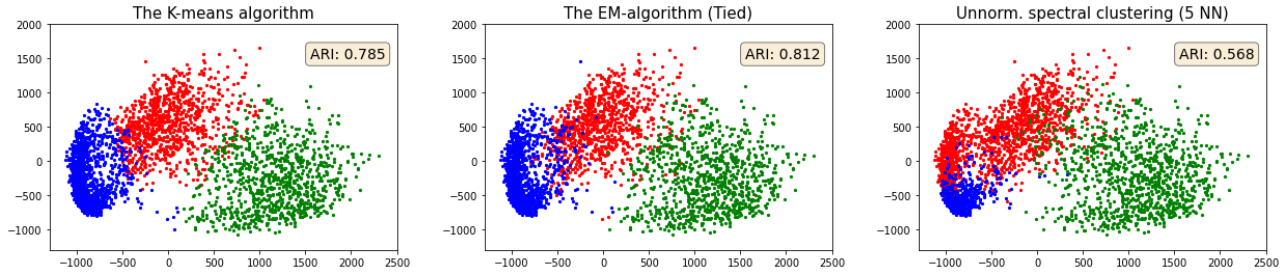

Figure 9: Clustering of all the digits 0, 1 and 8 among the first 10 000 digits by the three different algorithms projected onto the first two empirical principal components. The colors of the groups are set manually according to the labels for an easier visual comparison of the clusterings.

Here we notice that the spectral clustering algorithm is outperformed by both the K-means algorithm and the two best performing versions of the EM-algorithm with covariance matrix types 'full' and 'tied'. Again the EM-algorithm using 'tied' covariance matrices does better than the algorithm described in section (3.3) using 'full' covariance matrices, but this time the results are closer to each other. It is difficult to argue why the 'tied' covariance outperforms the 'full' covariance in the cases with three digits and not in the case with all 10 digits. So to choose one over the other on a data set should be done carefully.
Furthermore, this example also shows that spectral clustering not always does better than the K-means algorithm and the EM-algorithm. The reason of why spectral clustering does remarkably worse on these three digits compared to the digits 0, 2 and 9, is hard if not impossible to find.

From these results we cannot to say that one algorithm in general does better than the other. In many cases spectral clustering outperforms both the K-means and the EM-algorithm, but as we saw above this is not always the case and more importantly, we cannot be sure when this is not the case [Lux07]. Again, it is important to stress the fact that these algorithms are applied to unsupervised learning problems and therefore, we have no way of knowing whether or not an algorithm produces a useful clustering or not.
The results on the different subsets of the MNIST data therefore serve as a good example of why it is essential to know the assumptions and the theoretical background of each algorithm to be able to first, choose an algorithm that fits the data and second, choose the hyper-parameters involved.

6 Conclusion

In this project we have stated three different clustering algorithms and examined their clustering properties from a mathematical perspective. We have showed that the distortion measure in the K-means algorithm converges to a (local) minimum and we have seen examples of data structures where the K-means algorithm cannot correctly classify the groups. We have proved that the general EM-algorithm can be used as a clustering algorithm when applied in a Gaussian mixture setup and stated that this also converges to a local minimum. Furthermore, we have seen examples of data sets drawn from a Gaussian mixture, where the EM-algorithm for Gaussian mixtures outperforms the K-means algorithm, and we have proved and shown in a data example that the K-means algorithm can be derived as a limit of a special case of the EM-algorithm for Gaussian mixtures. Through the notion of similarity graphs we have defined different graph Laplacians and proved some of their key properties regarding spectral clustering algorithms. With the similarity graphs and graph Laplacians at hand, we have stated two different spectral clustering algorithms. To argue of the validity of the spectral clustering algorithms, we showed that both algorithms are solutions to relaxed versions of a discrete minimization of a normalized graph cut of the similarity graph. This minimization problem can be solved using the Courant-Fisher theorem, which we proved and used to define empirical principal components. Furthermore, we have argued that the algorithms also can be derived by finding a graph partition that minimizes the probability of a random walk transitioning between parts of the similarity graph. Lastly, we have defined the adjusted Rand index and used it to compare the algorithms on both simulated data and on different subsets of the digits from the MNIST database, that were visualized by projecting the data onto the first two empirical principal components. We argued that the K-means algorithm works well on simple data sets, that it produces reasonable clusterings on subsets of the MNIST data and that it has the advantage that it works fast even on large data sets. We showed that the EM-algorithms for Gaussian mixtures works well if it is a fair assumption that the data is drawn from a Gaussian mixture, and in section 5.2 we saw that this allowed it to detect overlapping group, which the other algorithms could not detect. But on the larger and more complex MNIST data we saw that the EM-algorithm were outperformed by the K-means algorithm in all three cases, except on the digits 0, 1 and 8 when using a 'tied' covariance matrix. Furthermore, we have seen in section 5.2 that spectral clustering works well even on more complex data structures. we saw that it outperformed both the K-means algorithm and the EM-algorithm on most of the MNIST data, but on the digits 0, 1 and 8 it produced the worst results. From this we concluded, that it is difficult, if not impossible, to say why this is the case and that this served as a good example of why spectral clustering should not be applied blindly to any kind of clustering problem, even though it from a theoretical perspective is an intriguing clustering algorithm.

A Derivatives of matrices

A.1 Matrix-derivatives

Let $\Sigma,\ W \in \mathrm{Mat}_{p\times p}(\mathbb{R})$ be positive semi-definite matrices, let $u = u(\Sigma) \in \mathbb{R},\ v = v(\Sigma) \in \mathbb{R}$, be functions of Σ and let $g:\ \mathbb{R} \to \mathbb{R}$ be a real-valued function. We then have the following identities of derivatives with respect to Σ:

$$\frac{\partial\ (uv)}{\partial \Sigma} = v\frac{\partial u}{\partial \Sigma} + u\frac{\partial v}{\partial \Sigma}. \tag{A.1}$$

$$\frac{\partial g(u)}{\partial \Sigma} = \frac{\partial g(u)}{\partial u} \cdot \frac{\partial u}{\partial \Sigma}. \tag{A.2}$$

$$\frac{\partial |\Sigma|}{\partial \Sigma} = |\Sigma|\Sigma^{-1}. \tag{A.3}$$

$$\frac{\partial\ \mathrm{Tr}\left(\Sigma^{-1}W\right)}{\partial \Sigma} = -\Sigma^{-1}W\Sigma^{-1}. \tag{A.4}$$

The rules are stated in [PP12].

References

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[PP12] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20121115. Nov. 2012. URL: `http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html`.

[GT21] Trevor Hastie Gareth James Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 2021.

[BV09] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009.

[Lux07] Ulrike von Luxburg. "A Tutorial on Spectral Clustering". In: *Statistics and Computing* 17.4 (2007).

[Die05] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. 5. ed. Springer, 2005. ISBN: 3540261826.

[HJ85] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[And21] Lars Nørvang Andersen. *Multivariat Statistisk Analyse II*. Aarhus University press, 2021.

[JW07] Richard Arnold Johnson and Dean W. Wichern. *Applied multivariate statistical analysis*. 6. ed. Upper Saddle River, NJ: Prentice Hall, 2007.

[MS01] Marina Meilă and Jianbo Shi. "A Random Walks View of Spectral Segmentation". In: *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*. Ed. by Thomas S. Richardson and Tommi S. Jaakkola. Vol. R3. Proceedings of Machine Learning Research. Reissued by PMLR on 31 March 2021. PMLR, Apr. 2001, pp. 203–208. URL: `https://proceedings.mlr.press/r3/meila01a.html`.

[HA85] Lawrence Hubert and Phipps Arabie. "Comparing Partitions". In: *Journal of Classification* 2.193-218 (1985).

[Ran71] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* 66.336 (1971).

[Hub77] Lawrence Hubert. "Nominal scale response agreement as a generalized correlation". In: *British Journal of Mathematical Statistical Psychology* 30.1 (1977), pp. 98–103.

[LC10] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: `http://yann.lecun.com/exdb/mnist/`.