UNIVERSITY OF T A R T U
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

**Roman Tekhov**

# Model-Based Clustering of Geographical Data

**Master's thesis (30 ECTS)**

Supervisor: Sven Laur, DSc

Author: ..... "....." ..... 2012

Supervisor: ..... "....." ..... 2012

Approved for defence

Professor: ..... "....." ..... 2012

TARTU 2012

# Contents

# Chapter 1

# Introduction

This work was motivated by the Virtual Situation Room project [4]. Its purpose is to predict the occurrence of some destructive events based on existing data. The goal of this thesis was to design and implement algorithms useful for analyzing geo-tagged events.

We focus on two different kinds of data. The first type is data distributed normally around central point sources which is a very common model in literature and in practice. The second type and the main focus of this work is data distributed along line segment sources. For example traffic accidents in rural area are likely to be distributed along highways which can be modeled as sequences of segments. In each case the goal is to identify regions where the events of the same type are likely to occur in the future. We accomplish this by applying model-based clustering techniques to our existing data observations.

We will start by providing basic background statistical theory needed to bootstrap the subsequent chapters. At the heart of this work is the well-known Expectation-Maximization algorithm. We will provide its formal definition and describe one of its most common applications - clustering of normally distributed data. We will also describe the standard routine for hard clustering of normal data - the k-means algorithm. Then we will suggest algorithms for both hard clustering and Expectation-Maximization fitting of line segment source data. These segment related algorithms are the main contribution of this thesis. Routines for dealing with noisy data will also be presented. Lastly we will introduce the implementation of algorithms and present the results of practical experiments.

# Chapter 2

# Probability distributions

In this chapter, we will explain some basic statistical definitions. This is required for understanding more advanced techniques described in the following chapters.

## 2.1 Basic definitions

Random variable $X$ is a variable whose possible values are the numeric outcomes of some phenomenon measurements. Random variables can be either discrete or continuous. Discrete variables can only take values from a fixed set of possible exact values. Continuous random variables can take any numeric value from some known intervals. We will be working with continuous random variables in this thesis.

Probability density function $\mathrm{p}(x)$ of a continuous random variable $X$ describes its behavior. The probability of $X$ to take values in range $[a, b]$ is equal to the integral of its density in that range:

$$\Pr[a \leqslant X \leqslant b] = \int_a^b \mathrm{p}(x)\,dx \ .$$

Density is subject to the following constraints. It is non-negative:

$$\mathrm{p}(x) > 0 \qquad \forall x \ ,$$

and the total probability is one:

$$\int_{-\infty}^{\infty} \mathrm{p}(x)dx = 1 \ .$$

Density can also depend on a number of parameters $\boldsymbol{\theta}$ which will further be referred to as the distribution parameters. The corresponding density will be denoted as $\mathrm{p}(x \mid \boldsymbol{\theta})$.

The expected value $\mathbf{E}(X)$ is the average of all possible values of $X$. For a continuous random variable it is defined as

$$\mathbf{E}(X) = \int_{-\infty}^{\infty} x\mathrm{p}(x)dx \ .$$

Variance $\mathbf{Var}(X)$ is a measure of how far from $\mathbf{E}(X)$ the values of $X$ are located

in general. It is defined as

$$\mathbf{Var}(X) = \mathbf{E}\left((X - \mathbf{E}(X))^2\right) \ .$$

Covariance $\mathbf{Cov}(X, Y)$ of two random variables $X$ and $Y$ is a measure of how these variable influence each other. It is defined as

$$\mathbf{Cov}(X, Y) = \mathbf{E}\left((X - \mathbf{E}(X))(Y - \mathbf{E}(Y))\right) \ .$$

Covariance is zero if the variables are independent. If $Y$ tends to increase as $X$ increases (or vice versa) then the covariance is positive. If $Y$ tends to decrease as $X$ increases (or vice versa) then the covariance is negative.

The $\alpha$-th percentile $p_\alpha$ of $X$ is the value below which a random observation falls with probability $\alpha$:

$$\mathrm{p}(X \leqslant p_\alpha) = \alpha \ .$$

The upper $\alpha$-th percentile $q_\alpha$ is the value below which a random observation falls with probability $1 - \alpha$:

$$\mathrm{p}(X \leqslant q_\alpha) = 1 - \alpha \ . \tag{2.1}$$

Sample $\boldsymbol{s}$ is a set of independent, identically distributed random observations that is used to make assumptions about the studied phenomenon.

### 2.1.1 Multivariate distributions

Suppose that a random variable $\boldsymbol{X}$ is a $d$-dimensional random vector, i.e., $\boldsymbol{X} = (X_1, \cdots, X_d)$. Each observation $\boldsymbol{x}_j$ of sample $\boldsymbol{s}$ is in this case also a $d$-dimensional vector, i.e., $\boldsymbol{x}_j = (x_{j1}, \cdots, x_{jd})$. The distributions of such vectors are called multivariate distributions. We will be working with 2-dimensional multivariate distributions in this thesis.

## 2.2 Important distributions

### 2.2.1 Uniform distribution

If a random variable $X$ can take arbitrary values in range $[a,\ b]$ and nowhere else then it is said to be uniformly distributed. The distribution is denoted as $X \sim U(a,\ b)$ and has the following density:

$$\mathrm{p}(x) = \begin{cases} \frac{1}{|b-a|} & x \in [a,\ b] \ , \\ 0 & x \notin [a,\ b] \ . \end{cases} \tag{2.2}$$

Density is constant inside the range (hence the name) and is proportional to its length, elsewhere the density is zero.

### 2.2.2 Normal distribution

A normally (or Gaussian) distributed one-dimensional random variable $X$ is denoted as $X \sim N(\mu,\, \sigma^2)$. Here $\mu$ is the mean (central point of mass of the distribution) and $\sigma^2$ is the variance. The density function of the univariate normal distribution is

$$\mathrm{p}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad . \tag{2.3}$$

The density is therefore proportional to the difference between the observation and the mean being highest at the mean. Variance controls the rate at which the density lowers as the difference increases.

Normal random variable with zero mean and unit variance is called a standard normal random variable, lets denote it as

$$Z \sim N(0,\, 1) \quad . \tag{2.4}$$

Suppose that $X$ is a non-standard normal random variable. It is possible to construct a standard one from it via standardization:

$$\frac{X-\mu}{\sigma} \sim N(0,\, 1) \quad . \tag{2.5}$$

Any linear transformation of $X$ is also a normal random variable:

$$aX + b \sim N\left(a\mu + b,\, a^2\sigma^2\right) \qquad \forall a \in \mathbb{R},\, \forall b \in \mathbb{R} \quad . \tag{2.6}$$

#### 2.2.2.1 Multivariate normal distribution

Random vector $\boldsymbol{X}$ is normally distributed if any linear combination of $\boldsymbol{X}$ has a univariate normal distribution:

$$\boldsymbol{A}^T\boldsymbol{X} = \sum_{k=1}^{d} a_k X_k \sim N \qquad \forall A \in \mathbb{R}^d \quad . \tag{2.7}$$

Multivariate normal distribution is denoted as $\boldsymbol{X} \sim N_d(\boldsymbol{\mu},\, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu}$ is the $d$-dimensional mean vector of $\boldsymbol{X}$ and $\boldsymbol{\Sigma} = (\Sigma_{ab})$ is its covariance matrix of size $d \times d$:

$$\boldsymbol{\mu} = (\mathbf{E}(X_1), \cdots, \mathbf{E}(X_d)) \quad ,$$

$$\boldsymbol{\Sigma}_{ab} = \mathbf{Cov}(X_a, X_b) \quad .$$

Probability density function of the multivariate normal distribution is defined as

$$\mathrm{p}(\boldsymbol{x} \mid \boldsymbol{\mu},\, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{\mathrm{d}}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.8}$$

where $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix and $\boldsymbol{\Sigma}^{-1}$ is its inverse matrix.

## 2.3 Parameter estimation

Suppose we have a sample $\boldsymbol{s} = (x_1, \cdots, x_n)$ of $n$ independently selected and identically distributed observations. We know the density equation but the parameter $\boldsymbol{\theta}$ values on which it depends are unknown. To get a complete description of data it is required to estimate the parameters based on $\boldsymbol{s}$.

Maximum Likelihood Estimation (MLE) method is a common way of unknown parameter estimation in case of known density equation [7]. Joint density of the sample characterizes the likelihood of obtaining $\boldsymbol{s}$ from $X$. Since all $x_j \in \boldsymbol{s}$ are independent, the joint density is the product of individual observation densities:

$$\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{s}) = \mathrm{p}(\boldsymbol{s} \mid \boldsymbol{\theta}) = \prod_{j=1}^{n} \mathrm{p}(x_j \mid \boldsymbol{\theta}) \quad . \tag{2.9}$$

Therefore the optimal choice of $\boldsymbol{\theta}$ must produce the maximal possible value of likelihood (2.9):

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\operatorname{argmax}} \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{s})$$

where $\boldsymbol{\Theta}$ represents the entire parameter space. To simplify computation the log likelihood is often used instead:

$$\ell(\boldsymbol{\theta} \mid \boldsymbol{s}) = \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{s}) = \sum_{j=1}^{n} \log \mathrm{p}(x_j \mid \boldsymbol{\theta}) \tag{2.10}$$

which has the same maximum points as (2.9). Therefore the goal is to find

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\operatorname{argmax}} \ell(\boldsymbol{\theta} \mid \boldsymbol{s}) \quad . \tag{2.11}$$

There is a standard way of obtaining closed form MLE. First, find the partial derivatives of (2.10) with respect to each parameter $\psi \in \boldsymbol{\theta}$. Then set this derivatives to zero and solve the resulting system of equations

$$\frac{\partial \ell(\boldsymbol{\theta} \mid \boldsymbol{s})}{\partial \psi} = 0 \qquad \psi \in \boldsymbol{\theta} \quad .$$

# Chapter 3

# Mixture distributions

## 3.1 Introduction

Suppose we are dealing with a mixture probabilistic model, i.e., the model in which the overall population consists of $M$ sub-components, each having its own distribution with specific parameters $\boldsymbol{\theta}_i$.

Let $\mathrm{p}(x \mid \boldsymbol{\theta}_i)$ denote the density of the $i$-th component. Let $\omega_i$ denote the weight of the $i$-th component in the mixture, i.e., the probability that a random observation is originated from $i$. Due to the law of total probability

$$\sum_{i=1}^{M} \omega_i = 1 \ . \tag{3.1}$$

Let $\boldsymbol{\omega}$ denote the vector of all mixture weights and let $\boldsymbol{\Theta}$ denote the vector of all mixture parameters:

$$\boldsymbol{\omega} = (\omega_1, \cdots, \omega_M) \ ,$$

$$\boldsymbol{\Theta} = \boldsymbol{\omega} \cup \boldsymbol{\theta} \ .$$

Then the density of the entire mixture is defined as the convex combination of individual component densities:

$$\mathrm{p}(x \mid \boldsymbol{\Theta}) = \sum_{i=1}^{M} \omega_i \mathrm{p}(x \mid \boldsymbol{\theta}_i) \ . \tag{3.2}$$

## 3.2 Mixture parameter estimation

Suppose we have a sample $\boldsymbol{s}$ selected from the mixture population. We have made assumptions about the distributions of mixture components, i.e., we known the density equations but no actual $\boldsymbol{\Theta}$ values. All observations $x_j$ are unlabeled, i.e., it is unknown from which component each observation originates from. The goal is to determine the unknown parameters $\boldsymbol{\Theta}$ based on the sample.

### 3.2.1 Expectation maximization algorithm

Direct Maximum Likelihood estimation is hard since the sample contains unknown data - component membership of observations. One common way of estimating parameters in case of hidden data is the Expectation Maximization (EM) algorithm which estimates parameters iteratively [6]. The idea is that the current estimations can be used for making assumptions about observation origins. And those in turn can be used to find new and better parameter estimations. The process continuous until convergence, i.e., until each subsequent iteration doesn't give significantly better estimations than the previous one.

Let $\boldsymbol{z}$ denote the unobserved vector of sample origins of length $n$, i.e., $z_j = i$ if sample element $x_j$ comes from the $i$-th component. According to the law of total probability and (2.10)

$$\ell\left(\boldsymbol{\Theta} \mid \boldsymbol{s}\right) = \sum_{j=1}^{n} \log\left(\sum_{i=1}^{M} \mathrm{p}\left(x_j,\, z_j = i \mid \boldsymbol{\Theta}\right)\right) \quad . \tag{3.3}$$

Suppose we have the current parameter estimates $\boldsymbol{\Theta}^{(m)}$ obtained as a result of $m$-th iteration. If each $j$-th element of (3.3) is multiplied and divided by $\mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right)$ then the equation becomes

$$\ell\left(\boldsymbol{\Theta},\, \boldsymbol{\Theta}^{(m)} \mid \boldsymbol{s}\right) = \sum_{j=1}^{n} \log\left(\sum_{i=1}^{M} \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \frac{\mathrm{p}\left(x_j,\, z_j = i \mid \boldsymbol{\Theta}\right)}{\mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right)}\right)$$

where

$$\sum_{i=1}^{M} \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) = 1 \qquad \forall j \in (1, \cdots, n) \quad .$$

Then each component satisfies the conditions for Jensen's inequality [14] with $\mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right)$ being the weight and logarithm being the convex function:

$$\ell\left(\boldsymbol{\Theta},\, \boldsymbol{\Theta}^{(m)} \mid \boldsymbol{s}\right) \geqslant Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right)$$

where

$$\begin{aligned} Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) &= \sum_{j=1}^{n}\sum_{i=1}^{M} \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \log \frac{\mathrm{p}\left(x_j,\, z_j = i \mid \boldsymbol{\Theta}\right)}{\mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right)} \\ &= \sum_{j=1}^{n}\sum_{i=1}^{M} \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \log \mathrm{p}\left(x_j,\, z_j = i \mid \boldsymbol{\Theta}\right) \\ &\quad - \sum_{j=1}^{n}\sum_{i=1}^{M} \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \log \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \quad . \end{aligned}$$

Lets denote the two components of this function as $Q'$ and $Q''$:

$$Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) = Q'\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) - Q''\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right)$$

where

$$Q'\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) = \sum_{j=1}^{n}\sum_{i=1}^{M} \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \log \mathrm{p}\left(x_j,\, z_j = i \mid \boldsymbol{\Theta}\right) \quad ,$$

$$Q''\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) = \sum_{j=1}^{n}\sum_{i=1}^{M} \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \log \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \quad .$$

The goal is to find new estimates of $\boldsymbol{\Theta}$ which maximize the outcome of $Q$ given the sample and the current estimates:

$$\boldsymbol{\Theta}^{(m+1)} = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}\, Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) \quad .$$

Since $Q''$ doesn't depend on $\boldsymbol{\Theta}$ it can be ignored during maximization. Therefore the problem boils down to maximizing $Q'$:

$$\boldsymbol{\Theta}^{(m+1)} = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}\, Q'\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) \quad .$$

This is the maximization step of the EM algorithm. For simplicity we will denote $Q'$ as simply $Q$ and it will be referred to as the Q-function:

$$Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) = \sum_{j=1}^{n}\sum_{i=1}^{M} \mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right) \log \mathrm{p}\left(x_j,\, z_j = i \mid \boldsymbol{\Theta}\right) \quad .$$

In case of a mixture model $\mathrm{p}\left(x_j,\, z_j = i \mid \boldsymbol{\Theta}\right)$ is the density of the $i$-th component at $x_j$:

$$\mathrm{p}\left(x_j,\, z_j = i \mid \boldsymbol{\Theta}\right) = \mathrm{p}\left(z_j = i \mid \boldsymbol{\Theta}\right) \mathrm{p}\left(x_j \mid \boldsymbol{\theta}_i\right) = \omega_i \mathrm{p}\left(x_j \mid \boldsymbol{\theta}_i\right)$$

where $\omega_i$ is the weight of the $i$-th component and $\boldsymbol{\theta}_i$ are its distribution parameters.

In the Expectation step we use the current parameter estimates $\boldsymbol{\Theta}^{(m)}$ to compute the posterior membership probabilities $\mathrm{p}\left(z_j = i \mid x_j,\, \boldsymbol{\Theta}^{(m)}\right)$. Lets denote these probabilities as $\boldsymbol{\tau}_{ij}^{(m)}$. Each $\boldsymbol{\tau}_{ij}^{(m)}$ is the probability of $x_j$ belonging to the $i$-th component based on the parameter estimates we have after the $m$-th iteration:

$$\boldsymbol{\tau}_{ij}^{(m)} = \frac{\omega_i^{(m)} \mathrm{p}\left(x_j \mid \boldsymbol{\theta}_i{}^{(m)}\right)}{\sum_{h=1}^{M} \omega_h^{(m)} \mathrm{p}\left(x_j \mid \boldsymbol{\theta}_h^{(m)}\right)} \quad . \tag{3.4}$$

The full form of the Q-function in case of mixture model is then

$$Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) = \sum_{j=1}^{n}\sum_{i=1}^{M} \boldsymbol{\tau}_{ij}^{(m)}(\log \omega_i^{(m)} + \log \mathrm{p}\left(x_j \mid \boldsymbol{\theta}_i\right)) \ . \tag{3.5}$$

The algorithm uses the current parameter estimates in order to calculate the posterior probabilities $\boldsymbol{\tau}_{ij}^{(m)}$ in the E-step and they, in turn, are used in order to find new (and better) estimates in the M-step. The process stops when parameter estimations have converged to a value close enough to a local maximum, i.e.,

$$\left|Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m+1)}\right) - Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right)\right| \leqslant \varepsilon_{EM} \tag{3.6}$$

where $\varepsilon_{EM}$ is some user-defined convergence threshold. This threshold is an absolute number which depends on the model being analyzed. A more universal way is to express the threshold in terms of percents. We decide that the algorithm has converged if the difference between $Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m+1)}\right)$ and $Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right)$ is $\varepsilon_{EM}$ percents or less. That way we can reuse the same $\varepsilon_{EM}$ value for different settings.

In order to ease maximization the Q-function can be decomposed into a sum of individual components which are independent from each other in terms of parametrization:

$$Q\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) = \sum_{i=1}^{M} Q_i\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) + Q_0\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right)$$

where

$$\begin{aligned} Q_i\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) &= \sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \log \mathrm{p}\left(x_j \mid \boldsymbol{\theta}_i\right) \ , \\ Q_0\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) &= \sum_{i=1}^{M}\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \log \omega_i \ . \end{aligned} \tag{3.7}$$

The main task can thus be split into multiple sub-tasks of maximizing individual $Q_i$. The component weight estimator is universal, i.e., it is the same for all distributions and can be obtained by maximizing $Q_0\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right)$ with respect to $\omega_i$:

$$\hat{\omega}_i^{(m+1)} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \tag{3.8}$$

which is simply the average of each components membership probabilities.

Sometimes it is also possible to obtain the exact formulas of parameter estimators in closed form. In cases where such solution is not available it is sometimes possible to estimate the values using other techniques, e.g., iteratively.

#### 3.2.1.1 Generalized Expectation-Maximization algorithm

Despite the name it is actually not strictly necessary to maximize the expectation. It is sufficient to just increase the value of the objective function after each iteration.

Although such procedure produces slower convergence it is still guaranteed to reach local maximum eventually. This technique is known as the Generalized Expectation Maximization [6].

#### 3.2.1.2 Local vs global maximum

Note that EM is guaranteed to reach its local maximum which is not necessarily the global maximum. It can converge to different results if we execute it more that once on the same data set and with different initial settings. It is a common practice to run the entire fitting procedure multiple times and select the model with the highest likelihood (3.3).

## 3.3 Constrained EM algorithm

Sometimes we want to control the way parameters are estimated. In particular, we want to control the weight of components. If left uncontrolled some components might disappear and some might grow too large. To prevent this we can define weight constraints and specify maximum or minimum allowed weight values. This technique is described for example in [12]. Lets denote the constraints as $\omega_{max}$ and $\omega_{min}$ correspondingly. Suppose we have estimated the weights after some iteration. Let $\mathcal{V}_{min}$ be the set of components whose weight is smaller than $\omega_{min}$. Let $\mathcal{V}_{max}$ be the set of components whose weight is greater than $\omega_{max}$. We set weights of all $\mathcal{V}_{min}$ components to $\omega_{min}$ and weights of all $\mathcal{V}_{max}$ components to $\omega_{max}$. Due to total weight constraint (3.1) we also have to normalize estimated weights of other components. Let $\mathcal{I}$ be the set of components whose initial weight did not violate any constraints, i.e., $\mathcal{I} = (1, \cdots, M) \setminus (\mathcal{V}_{min} \cup \mathcal{V}_{max})$. The sum of all weights is

$$S = \omega_{min} \mid \mathcal{V}_{min} \mid + \omega_{max} \mid \mathcal{V}_{max} \mid + \sum_{i \in I} \omega_i \ .$$

In order to make this sum equal to one we can multiply each $\omega_i$ by a normalization factor $f$:

$$\omega_{min} \mid \mathcal{V}_{min} \mid + \omega_{max} \mid \mathcal{V}_{max} \mid + \sum_{i \in I} \omega_i f = 1 \ .$$

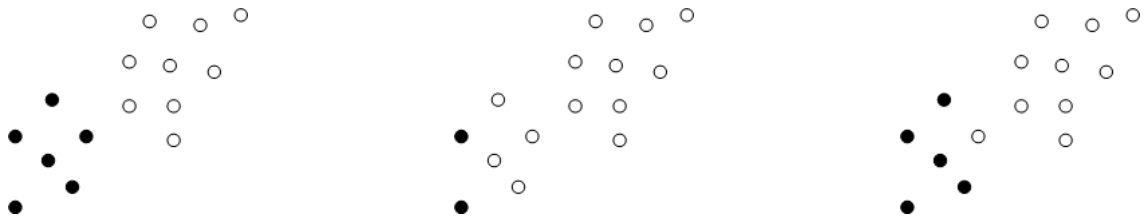
From this we can derive the equation for $f$ itself:

$$f = \frac{1 - \omega_{min} \mid \mathcal{V}_{min} \mid - \omega_{max} \mid \mathcal{V}_{max} \mid}{\sum_{i \in I} \omega_i} \ .$$

As a result each component $i$ will have $\omega_{min} \leqslant \omega_i \leqslant \omega_{max}$ at any stage while (3.1) is still respected.

Figure (3.1) illustrates constrained vs non-constrained fitting. Color of the point indicates the component that this point is assigned to with larger probability. In the non-constrained case one of the components becomes very small in favor of the other.

It can only get smaller in the next iterations, up until disappearing. In the constrained case we control its weight and therefore it cannot get smaller than allowed.



(a) Initial (b) Without constraint (c) With constraint

Figure 3.1: Constrained vs non-constrained EM

## 3.4 EM algorithm initialization

The simplest way to start the first iteration of EM is to provide some randomly generated initial values of model parameters and hope that it will eventually reach its maximum. This approach, however, might result in algorithm converging to values which are far from being optimal. It also means that the algorithm will need to perform more iterations until convergence. A better solution is to obtain the initial estimates by using a hard clustering procedure suitable for the underlying model. Then the EM algorithm receives input that is already close to optimal and it only needs to adjust the parameters for improved sample fitting. Hard clustering is described in detail in Chapter (4).

This gives as the initial membership probabilities $\boldsymbol{\tau}^{(-1)}$. We use them to bootstrap the EM routine, i.e., parameters $\boldsymbol{\Theta}^{(0)}$ can be computed with the usual equations of the maximization step.

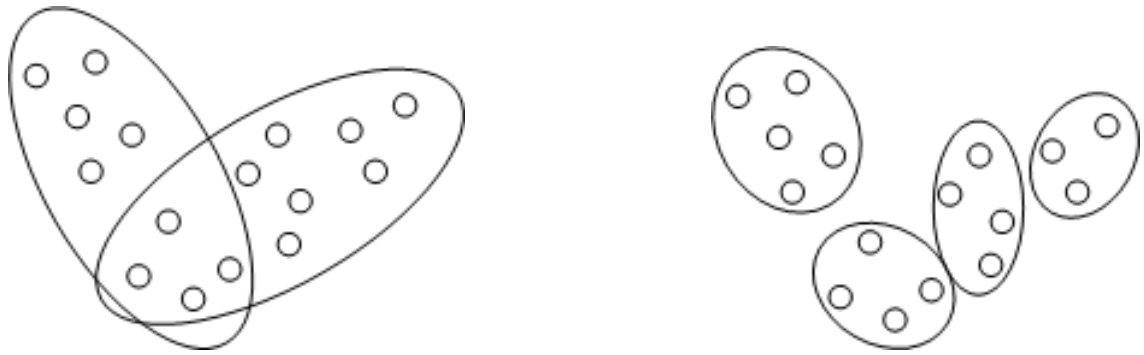
A generalized version of the suitable hard clustering algorithm looks like this. We find $\boldsymbol{\tau}^{(-1)}$ by clustering sample $\boldsymbol{s}$ into $M$ components. First, $M$ initial centers are chosen. Then at each iteration each sample point is assigned to the cluster of its closest center. After that each clusters center is re-calculated by fitting to the observations assigned to that cluster. The process continues until convergence, i.e., until no more re-assignments are performed. Initial membership estimates are then obtained from hard clustering result as follows:

$$\boldsymbol{\tau}_{ij}^{(-1)} = \begin{cases} 1 & \text{if } x_j \text{ belongs to cluster } i \ , \\ 0 & \text{otherwise } . \end{cases}$$

Such hard clustering algorithms are typically non-deterministic and can converge to different results if we start them from different initial positions. We can execute the algorithm multiple times and select the best result, i.e., the one that minimizes the sum of squared distances from sample points to their cluster centers (see Section (4.1.2.5)).

## 3.5 Estimating the number of components

An important limitation of the described approach is that the number of components $M$ has to be given a priori. If those values are unknown then it is required to choose the model that best fits to the given sample from a set of candidate models. Problem of overfitting also arises since models with more parameters tend to have bigger likelihood. The reason is that they can be more closely fitted to the sample without generalizing from it. Consider Figure (3.2). The second model has more parameters than the first one which allows it to fit more closely to the data. Therefore it has higher likelihood. But it doesn't mean that the second model describes population in a better way since it might be overfitted to this concrete sample.



(a) Less parameters (b) More parameters

Figure 3.2: Number of parameters and overfitting

One possible way is to use the Bayesian Information Criterion (BIC) measure [20]. For model $M$ its BIC value is obtained by using the equation

$$BIC_M = 2\ell\left(\boldsymbol{s} \mid \hat{\boldsymbol{\Theta}}_M\right) - \mid \boldsymbol{\Theta}_M \mid \log n \tag{3.9}$$

where $\hat{\boldsymbol{\Theta}}_M$ is the final estimation of parameters at which the EM has converged and $\mid \boldsymbol{\Theta}_M \mid$ is the total number of parameters. This criterion penalizes models with larger number of parameters thus reducing the risk of overfitting. E.g., in case of Figure (3.2) the first model would probably have higher BIC value than the second overfitted one.

To find the best model we have to choose the minimum and maximum allowed number of components $M_{min}$ and $M_{max}$. Then we fit our sample to each $M \in [M_{min}, \cdots, M_{max}]$. This produces $M_{max} - M_{min} + 1$ mixture estimations. The optimal mixture model is the one that has the largest BIC value.

# Chapter 4

# Model based clustering

Suppose we have a data set of objects and the goal is to partition these objects into groups based on some meaningful similarity measure. This problem is known as clustering and several different approaches have been developed for it. E.g., one way is to assign each object into exactly one cluster (also known as hard clustering). Another way is to calculate the probabilities of each object belonging to each cluster (soft or fuzzy clustering).

The initial situation for the clustering problem, however, is very similar to the probability mixture parameter estimation problem described in the previous chapter. There we also have a set of unlabeled observations initially and the goal is to estimate model parameters based on them. If we make assumptions about the probabilistic distribution of the data then the clustering problem can be transformed into the mixture parameter estimation problem. The objects are sample observations and probabilistic mixture components are clusters. Inference of parameters leads to the inference of likelihood of each object belonging to each cluster which means that the objects can be assigned to clusters based on the probabilistic likelihood measure. This approach is known as model based clustering.

In this chapter, we will introduce some data models that we worked with. Then we will describe both hard clustering and soft clustering routines that can be applied to these models.

## 4.1 Data models

In the next sections, we are going to consider data representing some events annotated with point geographical coordinates. This kind of data is in general case represented by a tuple $\boldsymbol{x} = (x_1, x_2) =$ (latitude, longitude). We will analyze three different kinds of data distribution specific to geo-tagged events.

### 4.1.1 Point source events

Events such as traffic accidents in large cities are likely to be distributed normally around city centers (see Figure (4.1)). Therefore we can model them as multivariate Gaussians, see Section (2.2.2.1).

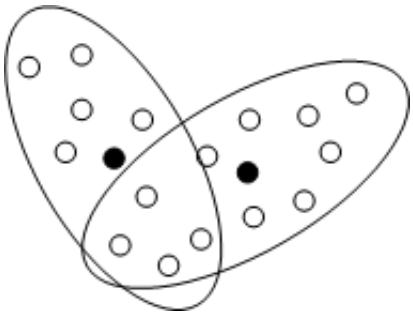


Figure 4.1: Point source events

### 4.1.2 Line segment source events

Some events are likely to be distributed along line segments rather than single central points. E.g., rural area traffic accidents are probably distributed along highways which can be modeled as line segments (see Figure (4.2)). Next we will describe this model in more detail.

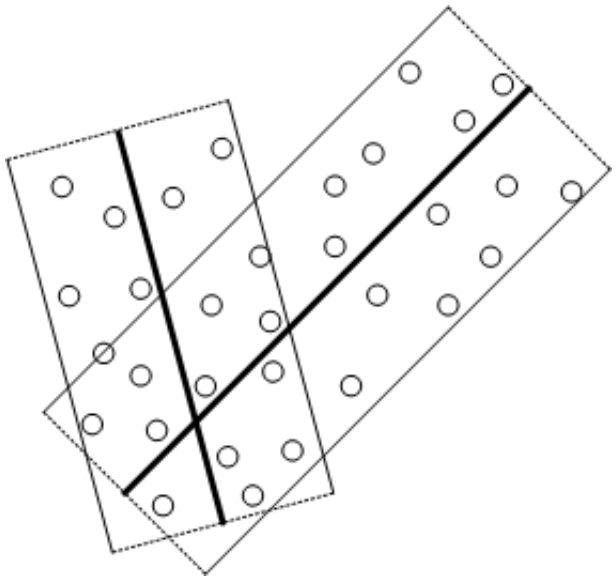


Figure 4.2: Line segment source events

#### 4.1.2.1 Distance distribution

Suppose we have a line on a 2-dimensional plane parametrized by a vector of coefficients $\boldsymbol{\beta} = (\beta_0,\ \beta_1,\ \beta_2)$ where $\beta_1$ and $\beta_2$ determine direction and $\beta_0$ corresponds to bias of the line (see Figure (4.3)):

$$L_{\boldsymbol{\beta}} = \{\boldsymbol{y} = (y_1,\ y_2) \mid \beta_1 y_1 + \beta_2 y_2 = \beta_0\} \quad . \tag{4.1}$$

Lets also require an additional constraint for $\boldsymbol{\beta}$:

$$\beta_1^2 + \beta_2^2 = 1 \quad . \tag{4.2}$$

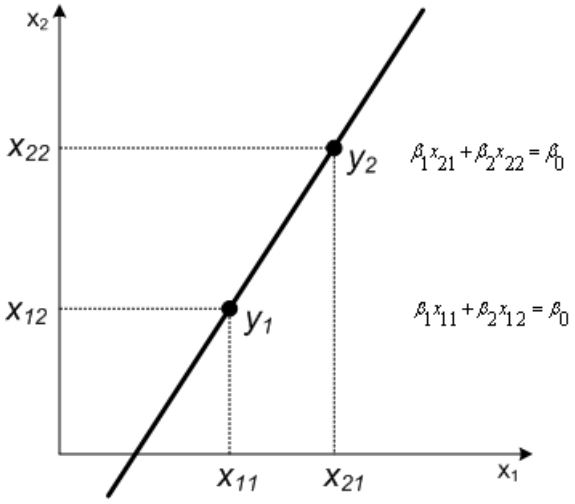



Figure 4.3: Line

The distance between some arbitrary point $\boldsymbol{x} = (x_1,\, x_2)$ and $L_{\boldsymbol{\beta}}$ is the shortest perpendicular distance between $\boldsymbol{x}$ and $\boldsymbol{y} \in L_{\boldsymbol{\beta}}$. Its equation is

$$d\left(\boldsymbol{x},\, L_{\boldsymbol{\beta}}\right) = \left|\beta_1 x_1 + \beta_2 x_2 - \beta_0\right| \quad . \tag{4.3}$$

Suppose $\boldsymbol{X}$ is a random vector such that its values are distributed along $L_\beta$. Lets assume that distances $d\left(\boldsymbol{X},\, L_{\boldsymbol{\beta}}\right)$ are distributed normally with zero mean and some variance $\sigma^2$:

$$d\left(\boldsymbol{X},\, L_{\boldsymbol{\beta}}\right) \sim N\left(0,\, \sigma^2\right) \quad .$$

The density of distance distribution is parametrized by $\boldsymbol{\beta}$ and $\sigma^2$. It is the normal distribution density of distances. According to (2.3) it takes the form

$$\mathrm{p}\left(\boldsymbol{x} \mid \boldsymbol{\beta},\, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(\beta_1 x_1 + \beta_2 x_2 - \beta_0\right)^2}{2\sigma^2}\right) \quad . \tag{4.4}$$

#### 4.1.2.2 Projection distribution

Let $q_{\boldsymbol{\beta}}\left(\boldsymbol{x}\right)$ denote the projection of point $\boldsymbol{x}$ onto line $L_{\boldsymbol{\beta}}$:

$$q_{\boldsymbol{\beta}}\left(\boldsymbol{x}\right) \;=\; \underset{\boldsymbol{y} \in L_{\boldsymbol{\beta}}}{\operatorname{argmin}}\, d\left(\boldsymbol{x},\, \boldsymbol{y}\right) = \left(z_1\left(\boldsymbol{x}\right),\, z_2\left(\boldsymbol{x}\right)\right) \tag{4.5}$$

where

$$z_1\left(\boldsymbol{x}\right) = x_1 - \beta_1\left(\beta_1 x_1 + \beta_2 x_2 - \beta_0\right) \quad ,$$

$$z_2\left(\boldsymbol{x}\right) = x_2 - \beta_2\left(\beta_1 x_1 + \beta_2 x_2 - \beta_0\right) \quad .$$

Lets assume that projections are distributed uniformly in some segment $[\boldsymbol{a},\, \boldsymbol{b}]$ of

$L_{\boldsymbol{\beta}}$ as in Figure (4.4). According to (2.2) the density of projection distribution is

$$\mathrm{p}\left(\boldsymbol{x} \mid \boldsymbol{\beta},\, \boldsymbol{a},\, \boldsymbol{b}\right) = \begin{cases} \frac{1}{d(\boldsymbol{a},\, \boldsymbol{b})} & q_{\boldsymbol{\beta}}\left(\boldsymbol{x}_j\right) \in [\boldsymbol{a},\, \boldsymbol{b}] \ , \\ 0 & q_{\boldsymbol{\beta}}\left(\boldsymbol{x}_j\right) \notin [\boldsymbol{a},\, \boldsymbol{b}] \end{cases} \tag{4.6}$$

where

$$d\left(\boldsymbol{a},\, \boldsymbol{b}\right) > 0 \ ,$$

i.e., $\boldsymbol{a}$ and $\boldsymbol{b}$ have to be different points.

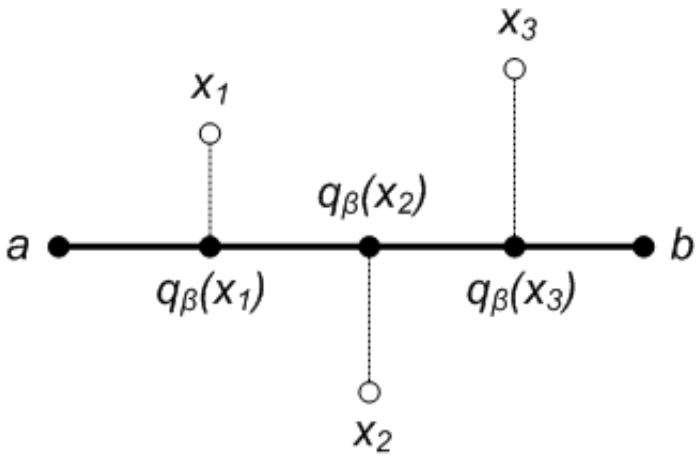



Figure 4.4: Projections

#### 4.1.2.3 Total distribution

Since the projection of a point and distance from it are assumed to be independent the entire line distribution density is

$$\mathrm{p}\left(\boldsymbol{x} \mid \boldsymbol{\beta},\, \sigma^2,\, \boldsymbol{a},\, \boldsymbol{b}\right) \;=\; \mathrm{p}_1\left(\boldsymbol{x} \mid \boldsymbol{\beta},\, \sigma^2\right) \mathrm{p}_2\left(\boldsymbol{x} \mid \boldsymbol{\beta},\, \boldsymbol{a},\, \boldsymbol{b}\right) \tag{4.7}$$

where $p_1$ is the distance distribution density (4.4) and $p_2$ is the projection distribution density (4.6).

#### 4.1.2.4 Distance between a point and a segment

Intuitively, the distance between a point $\boldsymbol{x}$ and a segment $[\boldsymbol{a},\, \boldsymbol{b}]$ can be expressed as

$$d\left(\boldsymbol{x},\, \boldsymbol{a},\, \boldsymbol{b}\right) = \begin{cases} d\left(\boldsymbol{x},\, L_{\boldsymbol{\beta}}\right) & q_{\boldsymbol{\beta}}\left(\boldsymbol{x}\right) \in [\boldsymbol{a},\, \boldsymbol{b}] \ , \\ min\left(d\left(\boldsymbol{x},\, \boldsymbol{a}\right),\, d\left(\boldsymbol{x},\, \boldsymbol{b}\right)\right) & \text{otherwise} \ . \end{cases} \tag{4.8}$$

In case $\boldsymbol{x}$ projects onto the segment like $\boldsymbol{x}_1$ in Figure (4.5) we use the diagonal distance between $\boldsymbol{x}$ and the line. Otherwise it is the distance from $\boldsymbol{x}$ to its closest segment endpoint like in case of points $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ in Figure (4.5).

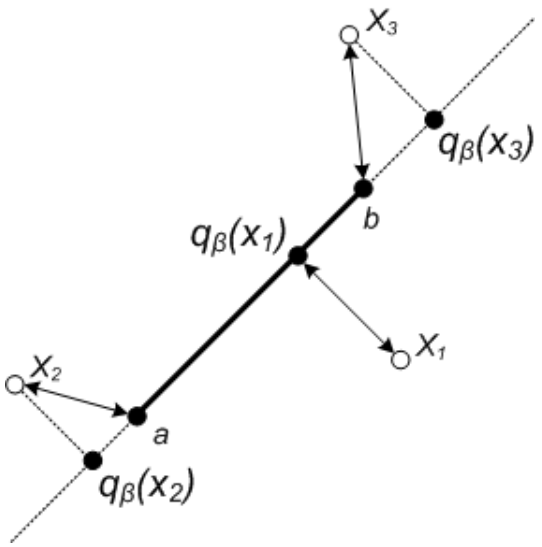



Figure 4.5: Distance between points and a segment

#### 4.1.2.5 Line coefficient fitting

In both hard and soft clustering we have to deal with estimating line orientation coefficients $\boldsymbol{\beta}$ based on sample. Consider the following function $S$:

$$S(\boldsymbol{\beta}) = \sum_{j=1}^{n} w_j d(\boldsymbol{x}_j,\ L_{\boldsymbol{\beta}})^2 \tag{4.9}$$

where

$$0 \leqslant w_j \leqslant 1 \qquad j \in (1, \cdots, n) \ .$$

It is the sum of squared distances between observations $\boldsymbol{x}_j$ and line $L_{\boldsymbol{\beta}}$. Distances are weighted by $w_j$ that defines the contribution of observations to the total sum. The goal is to find

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, S(\boldsymbol{\beta}) \ .$$

which is known as the Total Least Squares problem [10].

In Section (4.3.2.1) it will be shown that bias term $\beta_0$ can be expressed as

$$\beta_0 = \beta_1 \overline{x}_1 + \beta_2 \overline{x}_2 \tag{4.10}$$

where $\overline{x}_1$ and $\overline{x}_2$ denote the weighted sample mean of the corresponding dimension values:

$$\overline{x}_1 = \frac{\sum_{j=1}^{n} w_j x_{j1}}{\sum_{j=1}^{n} w_j} \ ,$$

$$\overline{x}_2 = \frac{\sum_{j=1}^{n} w_j x_{j2}}{\sum_{j=1}^{n} w_j} \quad .$$

If we take equation (4.10) into account then (4.9) can be expressed as

$$\begin{aligned} S(\boldsymbol{\beta}) &= \sum_{j=1}^{n} w_j d(\boldsymbol{x}_j,\ L_{\boldsymbol{\beta}})^2 \\ &= \sum_{j=1}^{n} w_j (\beta_1 x_{j1} + \beta_2 x_{j2} - \beta_0)^2 \\ &= \sum_{j=1}^{n} w_j (\beta_1 x_{j1} + \beta_2 x_{j2} - \beta_1 \overline{x}_1 - \beta_2 \overline{x}_2)^2 \\ &= \sum_{j=1}^{n} w_j (\beta_1 (x_{j1} - \overline{x}_1) + \beta_2 (x_{j2} - \overline{x}_2))^2 \quad . \end{aligned}$$

The same equation in matrix form is

$$S(\boldsymbol{\beta}) = (\boldsymbol{U}\boldsymbol{\beta})^T \boldsymbol{W} (\boldsymbol{U}\beta) \tag{4.11}$$

where $\boldsymbol{U}$ is a $n \times 2$ matrix such that

$$\begin{aligned} \boldsymbol{U}_{j1} &= x_{j1} - \overline{x}_1 \ , \\ \boldsymbol{U}_{j2} &= x_{j2} - \overline{x}_2 \end{aligned}$$

and $\boldsymbol{W}$ is a $n \times n$ diagonal matrix of weights such that $\boldsymbol{W}_{jj} = w_j$. We minimize this function by taking its partial derivative with respect to $\boldsymbol{\beta}$ and setting it to 0:

$$\frac{\partial S}{\partial \boldsymbol{\beta}} = 2\boldsymbol{U}^T \boldsymbol{W} \boldsymbol{U} \boldsymbol{\beta} = 0$$

which is subject to $\beta_1^2 + \beta_2^2 = 1$ due to (4.2).

One common way of solving this equation is the Singular Value Decomposition (SVD) method [9]. The idea is to decompose the matrix $\boldsymbol{U}^T \boldsymbol{W} \boldsymbol{U}$ into the product of three components:

$$\boldsymbol{U}^T \boldsymbol{W} \boldsymbol{U} = \boldsymbol{L} \boldsymbol{V} \boldsymbol{R} \tag{4.12}$$

where each matrix $\boldsymbol{L}$, $\boldsymbol{V}$ and $\boldsymbol{R}$ is a $2 \times 2$ matrix. $\boldsymbol{V}$ is a diagonal matrix which values are known as the singular values of the original matrix. Columns of $\boldsymbol{L}$ are known as the left singular vectors of the original matrix and the columns of $\boldsymbol{R}$ are known as the right singular vectors. The total least squares estimates of $\beta_1$ and $\beta_2$ are

$$\left(\hat{\beta}_1,\ \hat{\beta}_2\right) = \boldsymbol{R}_k \tag{4.13}$$

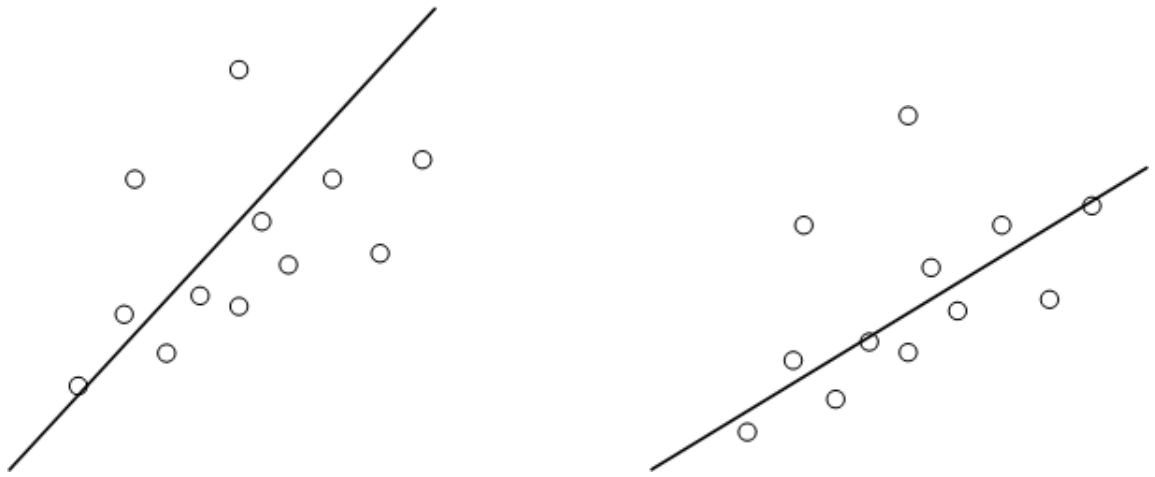
such that

$$k = \underset{k \in (1,\,2)}{\operatorname{argmin}} \boldsymbol{V}_{kk} \ .$$

I.e., the estimate is the right singular vector that corresponds to the smallest singular value.

#### 4.1.2.6 Robust coefficient fitting

Maximization of coefficients in its basic form requires to fit them to all observations. This is dangerous, however, since least squares estimation is extremely sensitive to outliers, i.e., the observations which are located significantly further from the line than the majority of observations. It gives excessive weight to each distance causing the outliers to influence the total sum significantly. Robust fitting techniques aim at reducing this influence and fit only to the most probable observations. The difference between non-robust and robust fitting is shown in Figure (4.6). In case of non-robust fitting the few outliers shift the line away from its optimal orientation. In robust fitting case this influence is eliminated.



(a) Non-robust (b) Robust

Figure 4.6: Non-robust vs robust fitting

A common way of making the fitting procedure robust to outliers is to use the Iteratively Re-weighted Total Least Squares approach (IRTLS) [13]. We re-fit the elements by iteratively re-weighting them so that the weight is proportional to the current distance from the line. The points that are further from the current line estimate have smaller influence on calculating the next estimate. The process stops when the parameters have converged to optimal values.

Suppose we have obtained $\hat{\beta}_1$ and $\hat{\beta}_2$ from (4.13). We also calculate $\hat{\beta}_0$ based on these values using (4.10). We use these coefficients $\hat{\boldsymbol{\beta}}$ to bootstrap the iterative re-

fitting procedure:

$$\boldsymbol{\beta}^{(0)} = \hat{\boldsymbol{\beta}} \ .$$

At each iteration $r$ the goal is to minimize the value of the objective function

$$S_{irtls}\left(\boldsymbol{\beta}^{(r)}\right) = \sum_{j=1}^{n} w_j \psi\left(d_j^{(r)}\right)\left(d_j^{(r)}\right)^2$$

where

$$d_j^{(r)} = \mid \beta_1^{(r)}\left(x_{j1} - \overline{x}_1\right) + \beta_2^{(r)}\left(x_{j2} - \overline{x}_2\right) \mid$$

is the distance from $\boldsymbol{x}_j$ to the current line estimation and $\psi(d)$ is some outlier weighting function. The new IRTLS coefficient estimates are once again the values that minimize the objective function:

$$\boldsymbol{\beta}^{(r+1)} = \underset{\boldsymbol{\beta}^{(r)}}{\operatorname{argmin}}\, S_{irtls}\left(\boldsymbol{\beta}^{(r)}\right)$$

subject to $\left(\beta_1^{(r+1)}\right)^2 + \left(\beta_2^{(r+1)}\right)^2 = 1$. They can once again be found by using SVD where the weight matrix is

$$\boldsymbol{W}_{jj}^{(r)} = w_j \psi\left(d_j^{(r)}\right) \ .$$

The process terminates when the coefficients have converged, i.e., when

$$\left| S\left(\boldsymbol{\beta}^{(r+1)}\right) - S\left(\boldsymbol{\beta}^{(r)}\right)\right| \leqslant \varepsilon_{irtls} \tag{4.14}$$

where $\varepsilon_{irtls}$ is a user defined threshold. To make this value more universal we can also use percentage instead of absolute number just like we did for EM convergence threshold (3.6). I.e., we decide that the process has converged if the difference between $S\left(\boldsymbol{\beta}^{(r+1)}\right)$ and $S\left(\boldsymbol{\beta}^{(r)}\right)$ is $\varepsilon_{irtls}$ percents or less.

**Tukey's bisquare estimator** One of the best known distance weighting functions is Tukey's bisquare estimator [19]:

$$\psi_{TB}(d) = \begin{cases} \left(1 - \left(\frac{d}{C\hat{s}}\right)^2\right)^2 & \frac{d}{\hat{s}} \leqslant C \ , \\ 0 & \frac{d}{\hat{s}} > C \end{cases} \tag{4.15}$$

where $C$ is a tuning constant commonly chosen to be 4.685 and $\hat{s}$ is the robust estimation of distance standard deviation, e.g., $\hat{s} = \frac{MAD}{0.6745}$ where $MAD$ is the weighted median absolute deviation [11] of distances:

$$MAD = median\left(\mid d^{(r)} - median\left(d^{(r)}\right) \mid\right) \ .$$

We have also used (4.15) for our experiments.

#### 4.1.2.7 Finding segment end points

Once we have estimated the orientation of the line (i.e., $\boldsymbol{\beta}$) we have to estimate the appropriate line segment endpoints $\boldsymbol{a}$ and $\boldsymbol{b}$. We can do it by analyzing the projections of our sample on the estimated line.

Suppose we have a fixed unbounded line $L_{\boldsymbol{\beta}}$ and each observation $\boldsymbol{x}_j$ projects onto it producing $q_{\boldsymbol{\beta}}(\boldsymbol{x}_j)$. Each observation also has a weight $w_j$ proportional to its significance. Only part of observations really belongs to the segment component so it's required to distinguish "true" elements from the "false" ones. To do that lets assume some requirements that the observations must satisfy:

1. They must be located significantly close to $L_{\boldsymbol{\beta}}$.
2. Their weight must be significantly high.

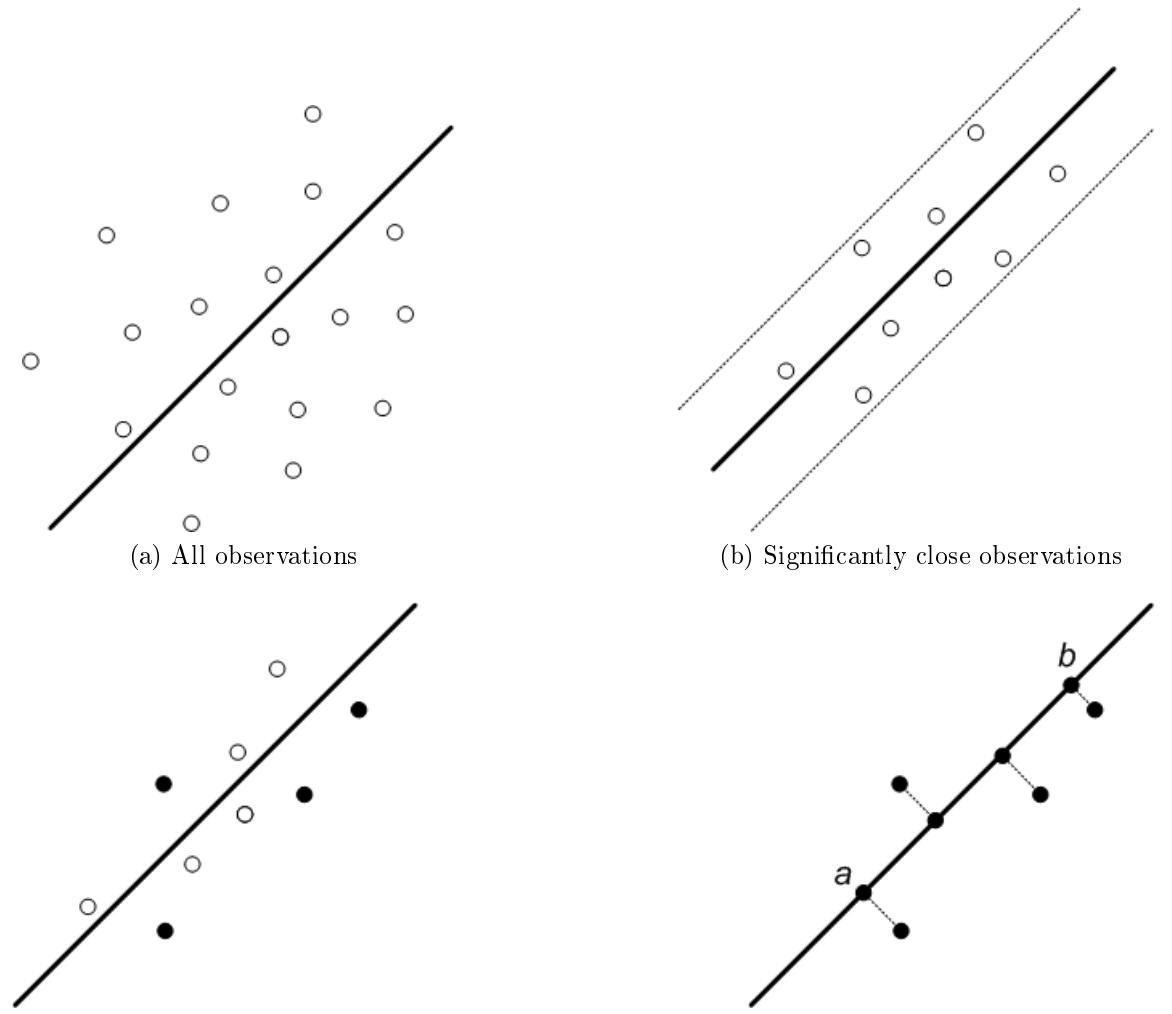
Compliance to the first requirement can once again be measured with the distance weighting function $\psi(d)$ described previously. We set the significance threshold $\varepsilon_d$ for $\psi(d)$ values. All $\boldsymbol{x}_j$ that have $\psi(d_j) \geqslant \varepsilon_d$ can be considered significantly close to $L_{\boldsymbol{\beta}}$. For the second requirement we can use the weight threshold $\varepsilon_w$. All $\boldsymbol{x}_j$ that have $w_j \geqslant \varepsilon_w$ can be considered significant in terms of weight. Let $\mathcal{S}$ denote the set of significant observations (Figures (4.7b) and (4.7c)):

$$\mathcal{S} = \{\boldsymbol{x}_j \mid \psi(d_j) \geqslant \varepsilon_d,\ w_j \geqslant \varepsilon_w\} \ .$$

The corresponding significant projections are:

$$q_{\boldsymbol{\beta}}(\mathcal{S}) = \{q_{\boldsymbol{\beta}}(\boldsymbol{x}_j) \mid \boldsymbol{x}_j \in \mathcal{S}\} \ .$$

We must estimate segment endpoints $\boldsymbol{a}$ and $\boldsymbol{b}$ such that all projections of $q_{\boldsymbol{\beta}}(\mathcal{S})$ fall between them and distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ is minimal. The boundary points of $q_{\boldsymbol{\beta}}(\mathcal{S})$ are the estimations that satisfy these requirements (Figure (4.7d)).




(a) All observations

(b) Significantly close observations

(c) Observations with significant weight

(d) Projections and segment endpoints

Figure 4.7: Finding segment endpoints

### 4.1.3 Noise events

The basic models assume that observation originates from one of the base model components. It might, however, be the case that some elements are outliers and don't really belong to any base component. Fitting such elements would result in distorted parameter estimations. Therefore, we introduce an additional data model, the outlier (or noise) component. It has a uniform distribution with constant probability density

$$\mathrm{p}\left(x \mid A\right) = \frac{1}{A}$$

where $A$ denotes the area of the sample space (the area inside the convex hull of $\boldsymbol{s}$).

## 4.2 Hard clustering

### 4.2.1 The k-means algorithm

The k-means algorithm [17] is a popular algorithm for clustering Gaussian mixture data. We start by selecting $M$ initial cluster centers as described in Section (4.2.1.1). Then at each iteration we assign observations to their closest cluster centers based on the Euclidean distance measure. Then we re-calculate the cluster centers to be the centroids of their assigned elements. The algorithm stops when there are no more re-assignments of observations.

#### 4.2.1.1 Initial center selection for k-means

One option is to randomly select centers for the first iteration. This simple approach might lead to slower convergence and not-optimal solutions. Another option is to use more intelligent ways to select initial centers. One popular initialization algorithm is k-means++ [5].

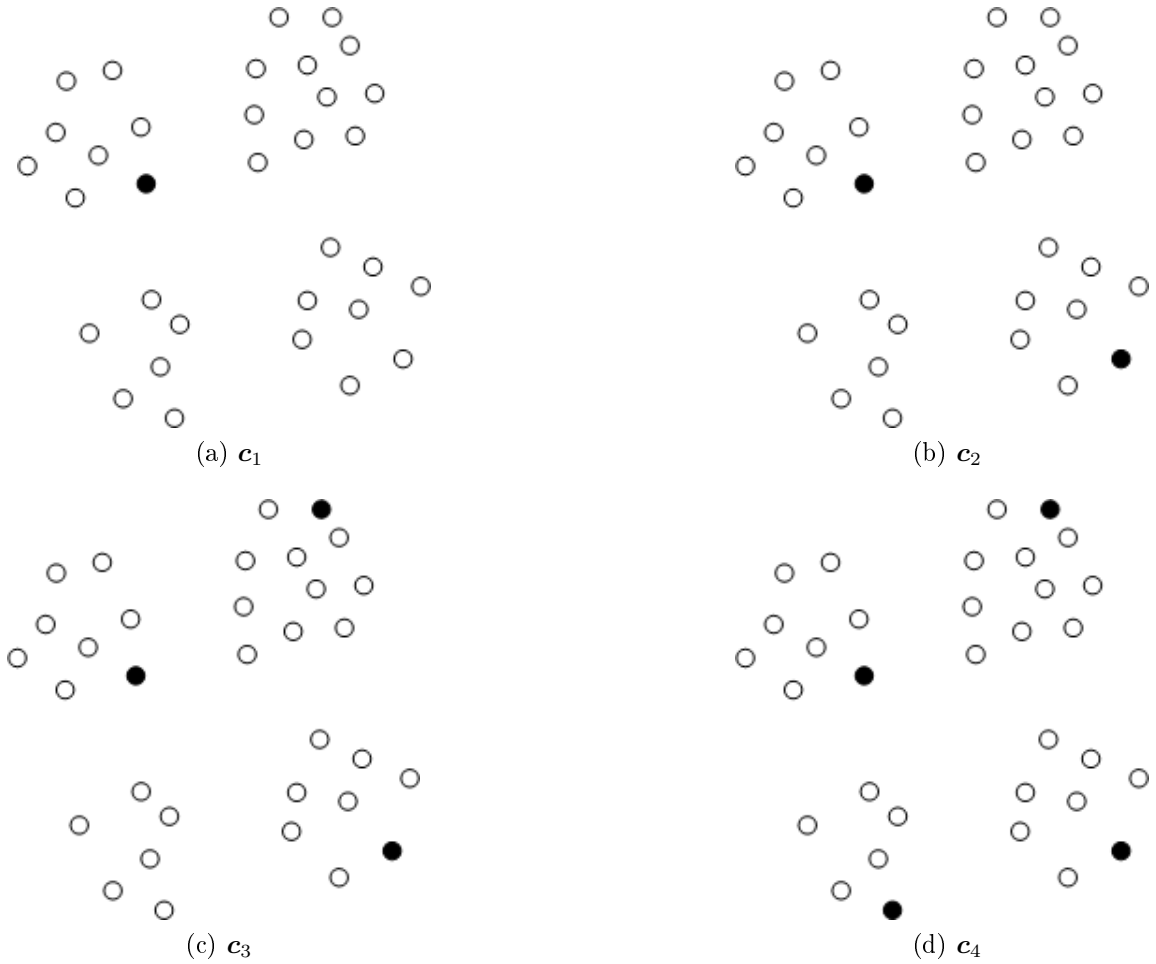
$M$ centers are selected from sample one by one. Let $\mathcal{C}$ be the set of centers known so far. Let $D_j$ be the distance between observation $\boldsymbol{x}_j$ and its closest center:

$$D_j = \min_{\boldsymbol{c}_i \in \mathcal{C}} d\left(\boldsymbol{x}_j,\ \boldsymbol{c}_i\right) \quad .$$

The first center $\boldsymbol{c}_1$ is selected randomly as one of the observations. When selecting centers $\boldsymbol{c}_2, \cdots, \boldsymbol{c}_M$ each observation can be selected with probability

$$p_j = \frac{D_j^2}{\sum_{k=1}^{n} D_k^2} \quad .$$

I.e., observations that are located far from current centers have higher chance of becoming the next center. Therefore k-means++ finds initial centers that are maximally apart from each other. The process of selecting 4 cluster centers is depicted in Figure (4.8).



(a) $\boldsymbol{c}_1$ (b) $\boldsymbol{c}_2$ (c) $\boldsymbol{c}_3$ (d) $\boldsymbol{c}_4$

Figure 4.8: K-means++ with 4 cluster centers

### 4.2.2 The k-segments algorithm

In case of line segment source model we can cluster observations around central segments. This clustering algorithm will further be referred to as k-segments. First, we select $M$ initial segments as described in Section (4.2.2.1). Then at each iteration we assign observations to their closest segments based on the distance measure (4.8). Then we re-fit the segments to their assigned elements in the following way.

Let $C_i$ denote the set of observations assigned to the $i$-th cluster. We need to find the $\boldsymbol{\beta}_i$ estimates of the cluster line, i.e., its orientation. For that we apply IRTLS algorithm (see Section (4.1.2.6)) giving $C_i$ as input. There are no observation weights here so we take $w_j = 1$ for all $j$.

Next step is to find segment endpoints $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$. For that we find projections of $C_i$ onto $L_{\boldsymbol{\beta}}$ and take their boundary points.

The algorithm stops when there are no more re-assignments of observations.

#### 4.2.2.1 Initial segment selection

Hard clustering relies strongly on the quality of initial segment selection. So this is a crucial step for the entire procedure. For our experiments we used the extended

version of k-means++ algorithm described in (4.2.1.1). Not surprisingly we will refer to this algorithm as k-segments++.

$M$ center-segments are selected from sample one by one. Let $\mathcal{C}$ be the set of centers known so far. Let $D_j$ be the distance between observation $\boldsymbol{x}_j$ and its closest center-segment $\boldsymbol{c}_i = (\boldsymbol{a}_i,\ \boldsymbol{b}_i)$:

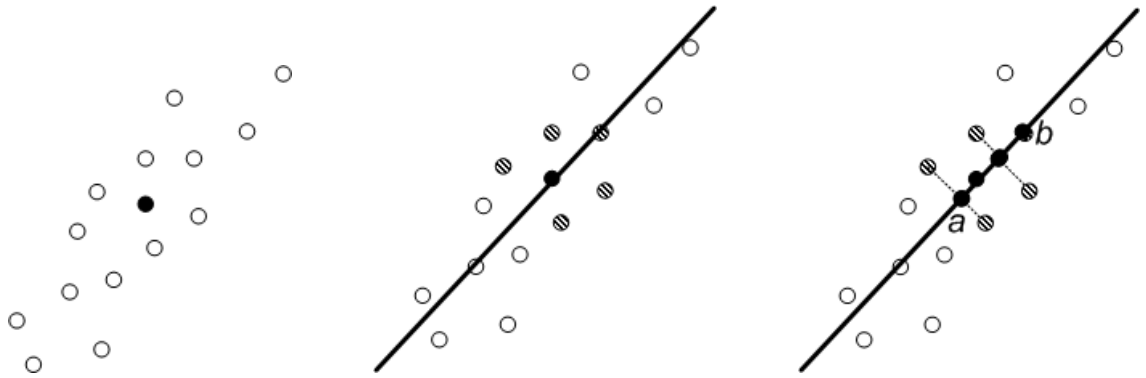
$$D_j = \min_{\boldsymbol{c}_i \in \mathcal{C}} d\left(\boldsymbol{x}_j,\ \boldsymbol{a}_i,\ \boldsymbol{b}_i\right) \quad .$$

First, we need to select an observation $\boldsymbol{y}_i$ that will be the seed of the next center-segment (Figure (4.9a)). The first seed $\boldsymbol{y}_1$ is selected randomly. When selecting seeds $\boldsymbol{y}_2, \cdots, \boldsymbol{y}_M$ each observation can be selected with probability

$$p_j = \frac{D_j^2}{\sum_{k=1}^{n} D_k^2} \quad .$$

First, we need to find initial estimation of line coefficients $\boldsymbol{\beta}_i$ which gives us the orientation of the line. The $k_{segment}$ nearest neighbors of $\boldsymbol{y}_i$ (lets denote them as $\mathcal{Y}_{ik}$) must probably belong to the same cluster as $\boldsymbol{y}_i$ itself. We apply the IRTLS procedure described in Section (4.1.2.6) giving $\mathcal{Y}_{ik}$ as input and using weight $w_j = 1$ for all $\boldsymbol{x}_j \in \mathcal{Y}_{ik}$ (Figure (4.9b)). Here $k_{segment}$ is a user defined parameter.

Next step is to find the new center-segment endpoints $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$. For that we find projections of $\mathcal{Y}_{ik}$ and take their boundary points (Figure (4.9c)).



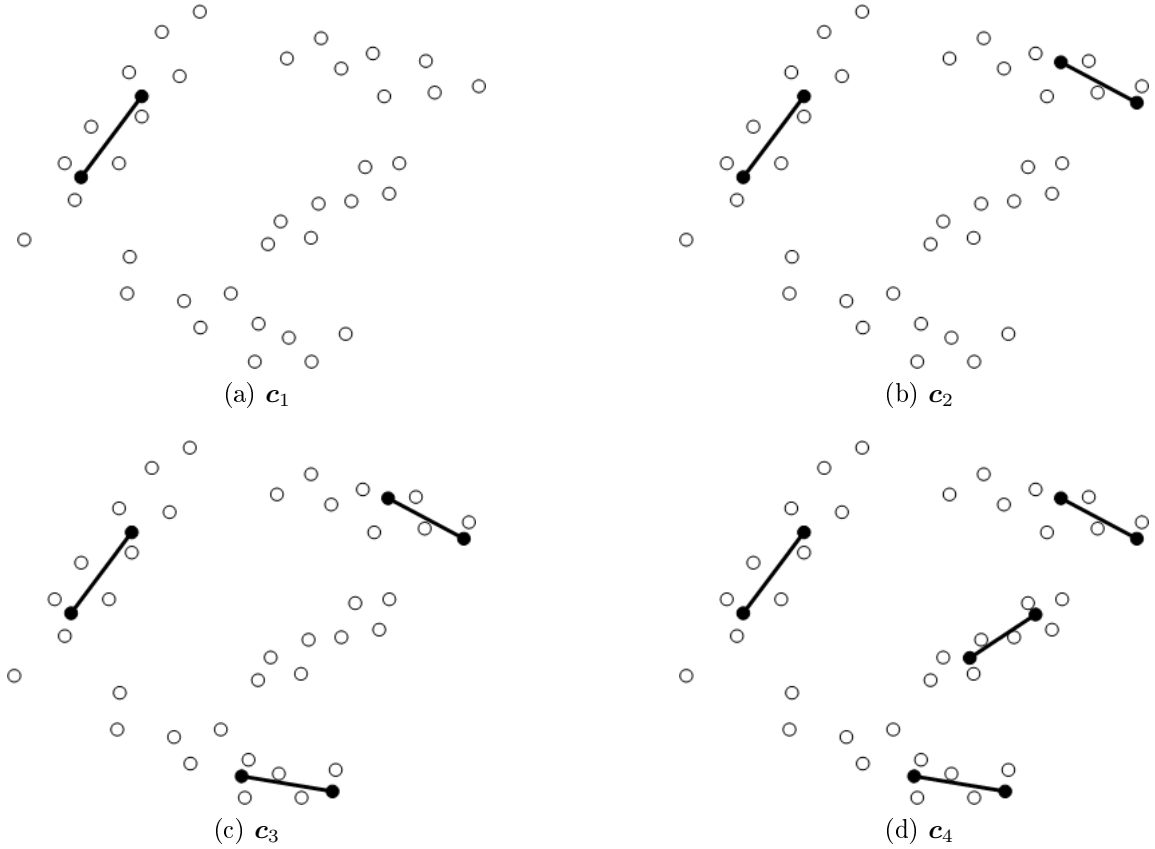

(a) Select seed $\boldsymbol{y}_i$

(b) Fit line to $k$ nearest neighbors of $\boldsymbol{y}_i$

(c) Find segment endpoints

Figure 4.9: Finding a center-segment

K-segments++ also tries to find initial centers that are maximally apart from each other. The process of selecting 4 cluster centers is depicted in Figure (4.10).



(a) $\boldsymbol{c}_1$ (b) $\boldsymbol{c}_2$ (c) $\boldsymbol{c}_3$ (d) $\boldsymbol{c}_4$

Figure 4.10: K-segments++ with 4 cluster centers

## 4.3 Soft clustering using expectation maximization algorithm

Expectation-maximization algorithm can be applied for soft clustering of data. After running EM on our sample we have the mixture parameter estimations $\hat{\boldsymbol{\theta}}_i$ as well as the final membership probabilities $\boldsymbol{\tau}_{ij}$, see (3.7) and (3.4). Parameters describe probabilistic components while membership probabilities represent grouping of observations across clusters. Each observation is assigned to each cluster with certain probability, therefore EM solves the soft clustering problem.

### 4.3.1 EM fitting of Gaussian models

Q-function of a single Gaussian component depends on its mean and covariance matrix:

$$Q_i\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) = \sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \log \mathrm{p}\left(\boldsymbol{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \quad . \tag{4.16}$$

Closed form equations for these parameter estimation can be obtained via maximization (2.11) [15]:

$$\hat{\boldsymbol{\mu}}_i^{(m+1)} = \frac{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \boldsymbol{x}_j}{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)}} \ , \tag{4.17}$$

$$\hat{\boldsymbol{\Sigma}}_i^{(m+1)} = \frac{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \left(\boldsymbol{x}_j - \boldsymbol{\mu}_i^{(m+1)}\right) \left(\boldsymbol{x}_j - \boldsymbol{\mu}_i^{(m+1)}\right)^T}{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)}} \ . \tag{4.18}$$

EM is basically a generalization of k-means and therefore they are very similar in nature. The difference is that EM uses Mahalanobis distance (see Section (4.4.1)) as its distance measure while k-means uses Euclidean distance. Therefore k-means can only fit spherical Gaussians while EM can handle Gaussians of arbitrary shape.

### 4.3.2 EM fitting of line segment models

The Q-function component corresponding to total density (4.7) is

$$\begin{aligned} Q_i\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) &= \sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \log \mathrm{p}\left(\boldsymbol{x} \mid \boldsymbol{\beta}_i,\, \sigma_i^2,\, \boldsymbol{a}_i,\, \boldsymbol{b}_i\right) \\ &= -\frac{1}{2}\log\left(2\pi\right) \sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \\ &\quad -\frac{1}{2}\log\sigma_i^2 \sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \\ &\quad -\frac{1}{2\sigma_i^2} \sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} d\left(\boldsymbol{x}_j,\, L_{\boldsymbol{\beta}}\right)^2 \\ &\quad -\sum_{x_j \in \mathcal{P}_i} \boldsymbol{\tau}_{ij}^{(m)} \log\, d\left(\boldsymbol{a}_i,\, \boldsymbol{b}_i\right) \end{aligned}$$

where

$$\mathcal{P}_i = \left\{\boldsymbol{x}_j \mid q_{\boldsymbol{\beta_i}}\left(\boldsymbol{x}_j\right) \in \left[\boldsymbol{a}_i,\, \boldsymbol{b}_i\right]\right\}$$

We can ignore the first component since it doesn't depend on any parameters. For convenience lets decompose the remaining part into separate logical components:

$$Q_i\left(\boldsymbol{\Theta} \mid \boldsymbol{s},\, \boldsymbol{\Theta}^{(m)}\right) = -\frac{1}{2}\log\sigma_i^2 \sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} - \frac{1}{2\sigma_i^2} Q_{i1} - Q_{i2} \tag{4.19}$$

where

$$Q_{i1} = \sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} d\left(\boldsymbol{x}_j,\, L_{\boldsymbol{\beta}}\right)^2 \tag{4.20}$$

defines the orientation of the line and

$$Q_{i2} \;=\; \log\, d\left(\boldsymbol{a}_i,\, \boldsymbol{b}_i\right) \sum_{x_j \in \mathcal{P}_i} \boldsymbol{\tau}_{ij}^{(m)} \;\;, \tag{4.21}$$

defines the segment of the line.

#### 4.3.2.1 Parameter estimation

Maximum Likelihood estimators of $\beta_{i0}^{(m+1)}$and distance variance $\sigma_i^{2(m+1)}$ can be obtained in closed form by setting the corresponding partial derivatives to zero. The estimators obtained this way are

$$\hat{\sigma_i^2}^{(m+1)} = \frac{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} \left(\beta_{i1}^{(m+1)} x_{j1} + \beta_{i2}^{(m+1)} x_{j2} - \beta_{i0}^{(m+1)}\right)^2}{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)}} \tag{4.22}$$

and

$$\hat{\beta}_{i0}^{(m+1)} = \beta_{i1}^{(m+1)} \overline{x}_{i1} + \beta_{i2}^{(m+1)} \overline{x}_{i2} \tag{4.23}$$

where $\overline{x}_{i1}$ and $\overline{x}_{i2}$ denote the weighted sample mean of the corresponding dimension values:

$$\overline{x}_{i1} = \frac{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} x_{j1}}{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)}} \;\;,$$

$$\overline{x}_{i2} = \frac{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)} x_{j2}}{\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(m)}} \;\;.$$

$Q_{i1}$ is identical to the Total Least Squares objective function (4.9). The weights $w_j$ are in this case the membership probabilities $\boldsymbol{\tau}_{ij}$. We choose the robust fitting way and therefore adjust each estimation using IRTLS described in Section (4.1.2.6). It means that we don't necessarily maximize the likelihood but only increase it in order to get robust estimations. Thus, we apply the Generalized Expectation-Maximization

technique (see Section (3.2.1.1)).

We also use the Generalized EM for estimating segment end points $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$. For that we apply heuristics described in Section (4.1.2.7). Here we can also use membership probabilities $\boldsymbol{\tau}_{ij}$ as the weights $w_j$.

## 4.4 Prediction regions

Once we have estimated component parameters we can use them to calculate prediction regions. These are the regions that contain observations with some given probability.

### 4.4.1 Prediction ellipsoids of Gaussian models

The value $(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})$ in the density equation (2.8) is the so-called squared Mahalanobis distance [18] which measures how close a point $\boldsymbol{x}$ is to the mean. When compared to standard Euclidean distance this measure is more appropriate for distributions which are not spherically symmetrical since it takes into account also the correlations between variables $X_1$ and $X_2$.

All points with equal squared Mahalanobis distance also have the same probability density. Therefore the set of all such points is called a constant probability density contour. It has the form of an ellipsoid and is centered at $\boldsymbol{\mu}$. Contour axes have the same directions as eigenvectors $\boldsymbol{e}_k$ of $\boldsymbol{\Sigma}^{-1}$. Their half-lengths are equal to $\sqrt{\lambda_k C}$ where $\lambda_k$ is the eigenvalue corresponding to $\boldsymbol{e}_k$ and $C$ is the squared Mahalanobis distance from contour points to $\boldsymbol{\mu}$ [15]. All points which are located inside the contour have distance less than $C$.

Another distribution that must be mentioned is the chi-squared distribution [8]. The sum of squares of $n$ independent standard normal random variables has chi-squared distribution with $n$ degrees of freedom:

$$\sum_{k=1}^{n} Z_k^2 \sim \chi_n^2, \qquad Z \sim N(0,\, 1) \quad . \tag{4.24}$$

In [15] it is shown that the squared Mahalanobis distance has the chi-squared distribution with 2 degrees of freedom in case of 2-dimensional data:

$$(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu}) \sim \chi_2^2 \quad .$$

Let $\chi_2^2(\alpha)$ denote the upper $\alpha$-th percentile of the chi-squared distribution with 2 degrees of freedom. According to (2.1)

$$\mathrm{p}\left((\boldsymbol{X}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{X}-\boldsymbol{\mu}) \leqslant \chi_2^2(\alpha)\right) = 1-\alpha \quad .$$

In other words, if we take a contour with squared Mahalanobis distance $\chi_2^2(\alpha)$ then there is a $1-\alpha$ probability that a random observation falls inside that contour. This contour is called a $1-\alpha$ prediction ellipsoid, the half-lengths of its axes are equal

to $\sqrt{\lambda_k \chi_2^2(\alpha)}$. This contour is the region that contains observations with some given probability $1-\alpha$. It can be used for visualization purposes like in Figure (4.11) .

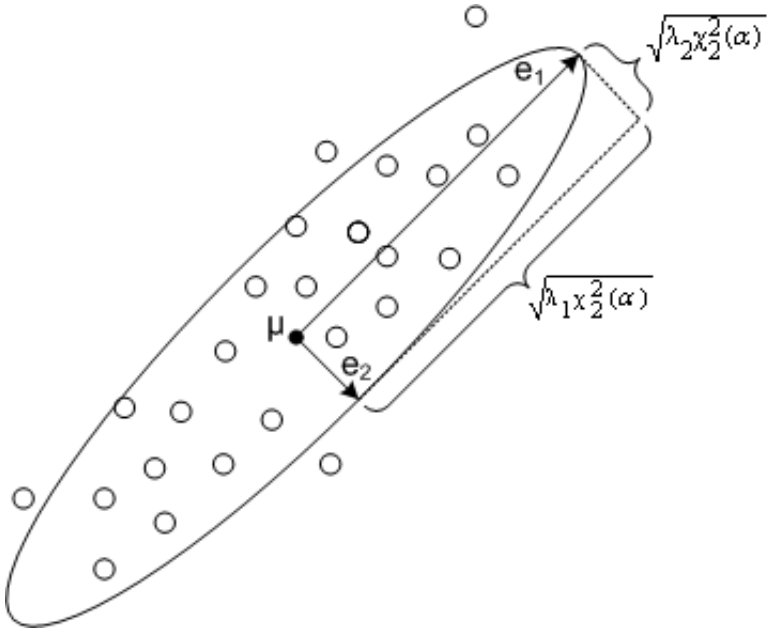



Figure 4.11: Prediction ellipse

### 4.4.2 Prediction bands of segment models

From density equation (4.7) it is clear that the density is equal for points $\boldsymbol{x}$ such that $q_{\boldsymbol{\beta}}(\boldsymbol{x}) \in [\boldsymbol{a}, \boldsymbol{b}]$ and $\frac{1}{\sigma^2}(\beta_1 x_1 + \beta_2 x_2 - \beta_0)^2 = C$ where $C$ is some constant. This corresponds to 2 symmetrical lines $L1$ and $L2$ which are parallel to $L$ and located at the distance of $\sigma\sqrt{C}$ in both directions from it. Segment $[\boldsymbol{a}, \boldsymbol{b}]$ is projected on $L1$ and $L2$ producing $[\boldsymbol{a}_1, \boldsymbol{b}_1]$ and $[\boldsymbol{a}_2, \boldsymbol{b}_2]$ respectively. Density is equal for all $\boldsymbol{x} \in [\boldsymbol{a}_1, \boldsymbol{b}_1] \cup [\boldsymbol{a}_2, \boldsymbol{b}_2]$. All points $\boldsymbol{y}$ which are located between these segments have $\frac{1}{\sigma^2}(\beta_1 y_1 + \beta_2 y_2 - \beta_0)^2 < C$. We can refer to such prediction regions as prediction bands (see Figure (4.12)).

Recall that since $\boldsymbol{X} = (X_1, X_2)$ is a multivariate random variable any linear combination of $X_1$ and $X_2$ is an univariate random variable, see (2.7). Therefore $\beta_1 X_1 + \beta_2 X_2 \sim N$. Recall that any linear transformation of a normal random variable is also a normal random variable, see (2.6). Therefore $|\beta_1 X_1 + \beta_2 X_2 - \beta_0| \sim N$. This is the exact equation (4.3) of distance between a point and a line and it has distribution

$$|\beta_1 X_1 + \beta_2 X_2 - \beta_0| \sim N\left(0,\, \sigma^2\right) \quad .$$

According to normal random variable standardization (2.5)

$$\frac{1}{\sigma}|\beta_1 X_1 + \beta_2 X_2 - \beta_0| \sim N(0,\, 1) \quad .$$

Recall that the sum of squares of $n$ independent standard normal random variables has

chi-squared distribution with $n$ degrees of freedom, see (4.24). Therefore

$$\frac{1}{\sigma^2}\left(\beta_1 X_1 + \beta_2 X_2 - \beta_0\right)^2 \sim \chi_1^2 \ .$$

Let $\chi_1^2(\alpha)$ denote the upper $\alpha$-th percentile of a chi-squared distribution with 1 degree of freedom. According to (2.1)

$$\begin{aligned} \mathrm{p}\left(|\beta_1 X_1 + \beta_2 X_2 - \beta_0| \leqslant \sigma\sqrt{\chi_1^2(\alpha)}\right) &= \\ \mathrm{p}\left(d\left(\boldsymbol{X},\, L_{\boldsymbol{\beta}}\right) \leqslant \sigma\sqrt{\chi_1^2(\alpha)}\right) &= 1-\alpha \ . \end{aligned}$$

This can be interpreted as a band with $C = \chi_1^2(\alpha)$ and $1-\alpha$ probability that a random observation falls inside that band. Therefore we can call it a $1-\alpha$ prediction band where $[\boldsymbol{a}_1,\, \boldsymbol{b}_1]$ and $[\boldsymbol{a}_2,\, \boldsymbol{b}_2]$ are located at $\sigma\sqrt{\chi_1^2(\alpha)}$ distance from $[\boldsymbol{a},\, \boldsymbol{b}]$ (see Figure (4.12)).

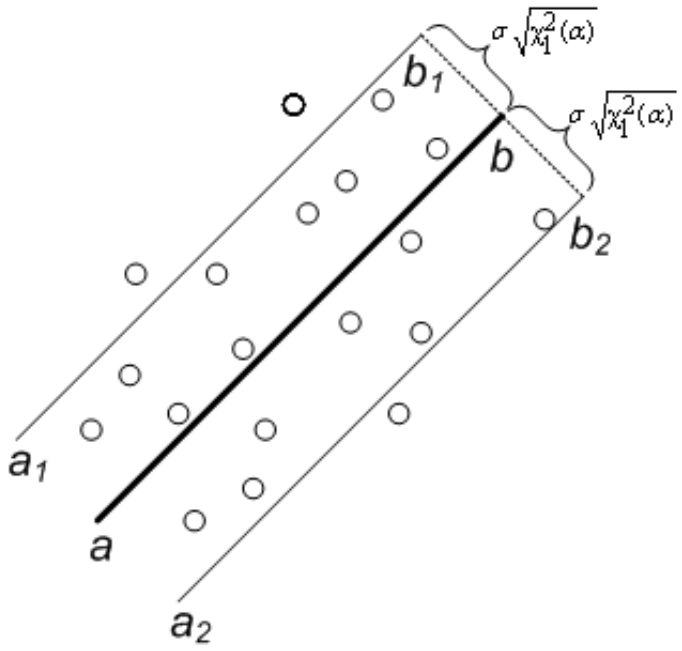



Figure 4.12: Prediction band

## 4.5 Modeling background noise

Recall from Section (4.1.3) that we model noise as a separate uniformly distributed component of the mixture. Lets denote it as the 0-th component. Let $\mathcal{N}$ denote the elements which are considered to be outliers and let $\omega_0$ denote the fraction (weight) of outliers in the population. The estimation of weight $\omega_0$ is then $\hat{\omega}_0 = \frac{|\mathcal{N}|}{n}$. Initial membership probabilities for the noise component can also be found based on $\mathcal{N}$:

$$\boldsymbol{\tau}_{0j}^{(-1)} = \begin{cases} 1 & \text{if } \boldsymbol{x}_j \in \mathcal{N} \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

The remaining inlier elements $\boldsymbol{x}_j \notin \mathcal{N}$ can then be clustered into $M$ components by using the usual hard clustering routine for finding $\boldsymbol{\tau}_{ij}^{(-1)}$.

### 4.5.1 Outlier detection with LoOP algorithm

One way to find outliers is by using the Local Outlier Probabilities (LoOP) density based method [16]. The main idea of this approach is that outlier observations typically have lower local density than their nearest neighbors while cluster elements have approximately the same local density as their nearest neighbors.

Let $K_{\boldsymbol{x}}$ denote the set of $k_{LoOP}$ nearest neighbors of element $\boldsymbol{x}$ where $k_{LoOP}$ is a user defined number. The standard distance from $\boldsymbol{y} \in K_{\boldsymbol{x}}$ to $\boldsymbol{x}$ is an estimation of density around $\boldsymbol{x}$, i.e., the average distance from $\boldsymbol{x}$ to its neighbor element:

$$s\left(\boldsymbol{x}\right) = \sqrt{\frac{\sum\limits_{\boldsymbol{y} \in K_{\boldsymbol{x}}} d\left(\boldsymbol{x}, \boldsymbol{y}\right)^2}{k_{LoOP}}} \ .$$

The Probabilistic Local Outlier Factor (PLOF) of $\boldsymbol{x}$ is a measure of how close the standard distance of $\boldsymbol{x}$ is to the average standard distance of its neighbors.

$$\mathsf{plof}\left(\boldsymbol{x}\right) = \frac{s\left(\boldsymbol{x}\right)}{[\sum\limits_{y \in K_{\boldsymbol{x}}} s\left(\boldsymbol{y}\right)]/\mathrm{k_{LoOP}}} - 1 \ .$$

$\mathsf{plof}(\boldsymbol{x}) \leqslant 0$ indicates that $\boldsymbol{x}$ is not an outlier while higher values indicate an increasing chance of $\boldsymbol{x}$ being an outlier. This measure, however, is not normalized and its value depends on the model being analyzed. Therefore we need to normalize the measure and present the result in the form of probability. Let $\boldsymbol{\sigma}_{\mathsf{plof}}$ denote the standard deviation of $PLOF$ values:

$$\boldsymbol{\sigma}_{\mathsf{plof}} = \sqrt{\frac{\sum\limits_{j=1}^{n} \mathsf{plof}\left(\boldsymbol{x}_j\right)^2}{n-1}} \ .$$

The probability of element $\boldsymbol{x}$ being an outlier is then referred to as the Local Outlier Probability (LoOP) and is defined as

$$LoOP\left(\boldsymbol{x}\right) = max\left(0\,, \mathrm{erf}\left(\frac{\mathsf{plof}\left(\boldsymbol{x}\right)}{\boldsymbol{\sigma}_{\mathsf{plof}}\sqrt{2}}\right)\right)$$

where erf is the Gaussian Error Function:

$$erf\left(a\right) = \frac{2}{\sqrt{\pi}} \int_0^a e^{-t^2} dt \ . \tag{4.25}$$

We can then define a LoOP threshold $\varepsilon_{LoOP}$ to classify observations as outliers:

$$\mathcal{N} = \{\boldsymbol{x} \in \boldsymbol{s} \mid LoOP(\boldsymbol{x}) \geqslant \varepsilon_{LoOP}\} \ . \tag{4.26}$$

### 4.5.2 Noise weight constraint

In our experiments we have used a separate maximum weight constraint $\omega^{\mathcal{N}}_{max}$ for the noise component as described in Section (3.3). This prevents the noise component from growing beyond acceptable limits.

## 4.6 Final algorithm for fitting with background noise

We start by detecting outliers according to (4.5.1).

Recall from Section (3.5) that we typically don't know the exact number of components $M$. Therefore we have to try fitting all numbers of mixture components $M \in [M_{min}, \cdots, M_{max}]$.

For each $M$ we first partition the inlier observations into $M$ clusters using hard clustering routines as described in Section (3.4). This procedure is executed multiple times and the best clustering result is used to bootstrap the EM routine. The entire fitting of $M$ components is also repeated multiple times and the model with the highest likelihood value as selected (see Section (3.2.1.2)). For that model we calculate its corresponding BIC value according to (3.9). The final result is the model that has the maximum BIC value.

Finally we calculate prediction regions of interest for all resulting mixture components as described in Section (4.4). This regions as well as model parameters can be used for visualization and final output.

### 4.6.1 Configuration parameters

Our overall algorithm is a composition of multiple sub-routines. Each one of them depends on one or multiple configuration parameters that we need to define. See Table (4.1) for a list of all these parameters and their descriptions.

| Parameter | Description | Reference |
|---|---|---|
| $M_{min}$, $M_{max}$ | Minimum and maximum number of mixture models to fit. | (3.5) |
| $\varepsilon_{EM}$ | EM convergence threshold. | (3.6) |
| EM trials | Number of EM trials per one number of mixture components. | (3.2.1.2) |
| Hard clustering trials | Number of hard clustering executions per one EM trial. | (3.4) |
| $\omega_{min}$ | Minimum allowed weight of a non-noise mixture component. | (3.3) |
| $\omega_{max}^{\mathcal{N}}$ | Maximum allowed weight of a noise component. | (4.5.2) |
| $k_{LoOP}$ | Number of nearest neighbors for LoOP computation. | (4.5.1) |
| $\varepsilon_{LoOP}$ | LoOP threshold. | (4.26) |
| $\varepsilon_{irtls}$ | IRTLS convergence threshold. | (4.14) |
| $\varepsilon_d$, $\varepsilon_w$ | Distance and weight thresholds for segment fitting. | (4.1.2.7) |
| $k_{segment}$ | Number of nearest neighbors for k-segments++ seeds. | (4.2.2) |

Table 4.1: All parameters

# Chapter 5

# Implementation and experimental results

In this chapter, we are going to describe the actual implementation of the algorithms. We will also describe the settings of practical experiments and present their results.

## 5.1 Implementation

The algorithm was implemented in Python, see Appendix A. We have used a number of third-party libraries for various purposes:

- NumPy [2] for working with arrays and matrices.
- SciPy [3] for calculating the value of Gaussian Error Function (4.25), Singular Value Decomposition (4.12) and finding chi-squared percentiles (4.4.2).
- Matplotlib [1] for visualization.

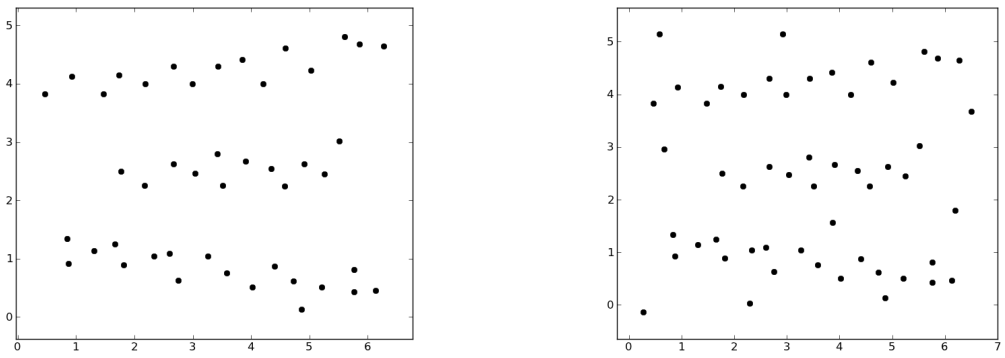
## 5.2 Experimental results

The implementation was evaluated using both generated synthetic data and a real world data set. Main goal was to test the line segment fitting routine. Therefore, we shall only present segment fitting results in this work. Point source data fitting using EM of Gaussian mixture models is well known and provides little practical challenge.

### 5.2.1 Synthetic data

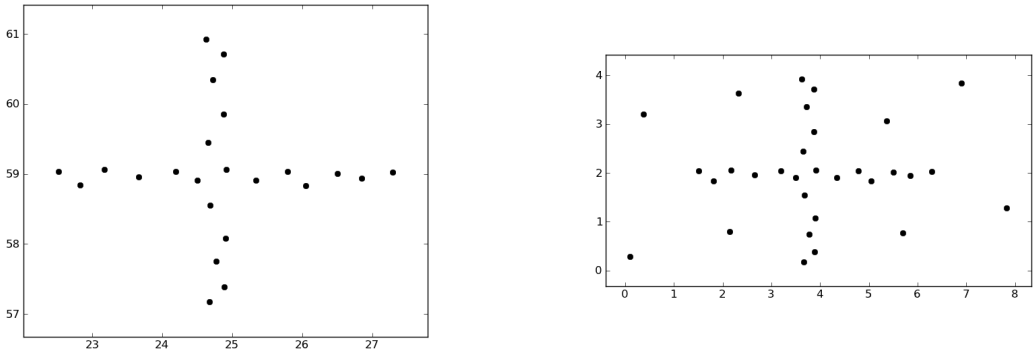
We have tested the algorithm on three synthetic data sets. The first is the "simple" data set which contains points aligned to three segments. These segments don't intersect or connect in any way. The second is the "cross" data set which contains points aligned to two intersecting segments. And the third is the "circular" data set which contains points aligned to four segments with matching endpoints. Each data

set also has a second form with added random noise points. All data sets are depicted on Figures (5.1), (5.2) and (5.3).



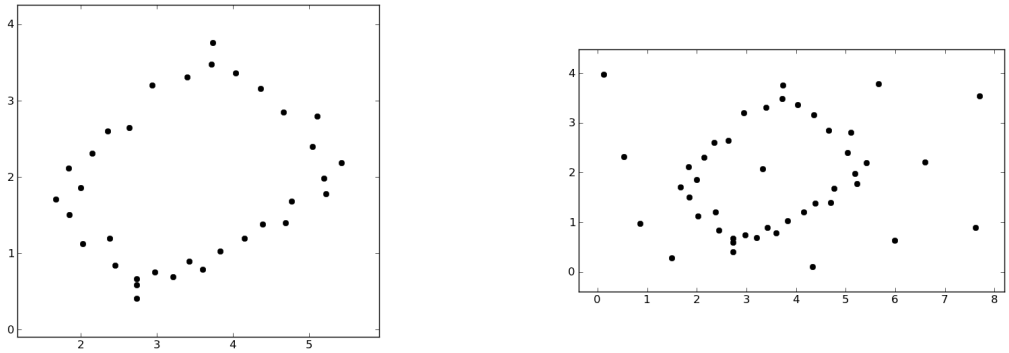
(a) Without noise (b) With noise

Figure 5.1: “Simple” data set



(a) Without noise (b) With noise

Figure 5.2: “Cross” data set

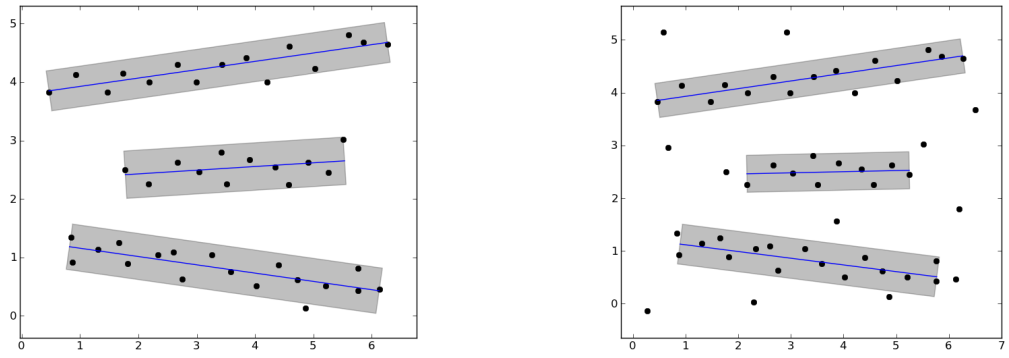


(a) Without noise (b) With noise

Figure 5.3: “Circular” data set

Both EM and IRTLS convergence thresholds were equal to 0.00005% during experiments ($\varepsilon_{EM} = 0.00005$, $\varepsilon_{irtls} = 0.00005$). Minimum allowed weight of a segment component was 0.05 as well as the maximum allowed weight of a noise component ($\omega_{min} = 0.05$, $\omega^{\mathcal{N}}_{max} = 0.05$). We used 10 trials for both the k-segment routine and the entire EM routine. LoOP outlier probability threshold $\varepsilon_{LoOP}$ was 60% for fitting with noise and 100% for fitting without noise, i.e., we didn't allow any noise points while fitting such data sets. Other common parameters: $\varepsilon_d = 0.1$, $\varepsilon_w = 0.1$, $k_{LoOP} = 5$, $k_{segment} = 5$.

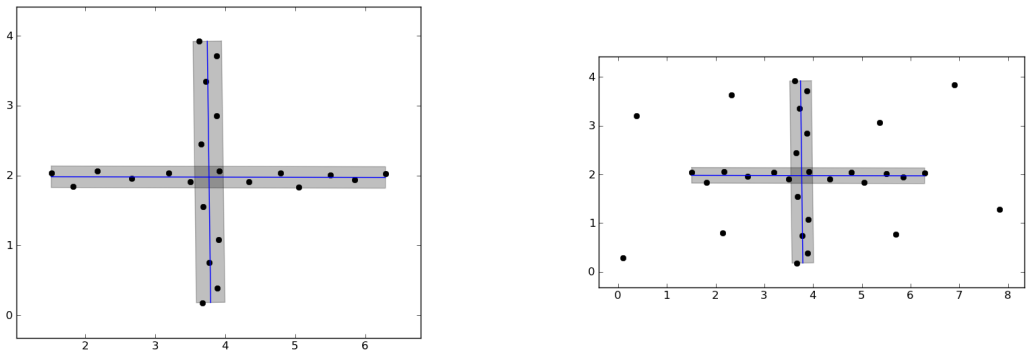
For "simple" data set we used $M_{min} = 2$, $M_{max} = 4$. For "cross" data set $M_{min} = 1$, $M_{max} = 3$. For "circular" data set $M_{min} = 2$, $M_{max} = 4$. In all cases our algorithm successfully selected the expected models based on BIC values. I.e., 4 components for "simple" and "circular" data sets and 2 components for "cross" data set. See Figures (5.4), (5.5) and (5.6) for graphical visualization of results. All figures illustrate $1-0.01$ prediction bands.



(a) Without noise

(b) With noise

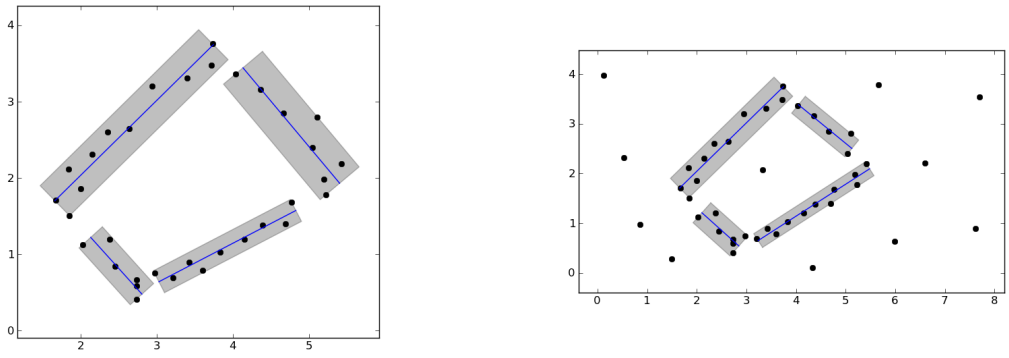
Figure 5.4: "Simple" data set results



(a) Without noise

(b) With noise

Figure 5.5: "Cross" data set results



(a) Without noise

(b) With noise

Figure 5.6: "Circular" data set results

### 5.2.2 Real world data

We used the sample of 400 Tallinn addresses. We assumed that they should be grouped according to major city regions and aligned to major streets. Therefore, it is reasonable to apply the line segment model based clustering routine to such data. The addresses sample is depicted on Figure (5.7).

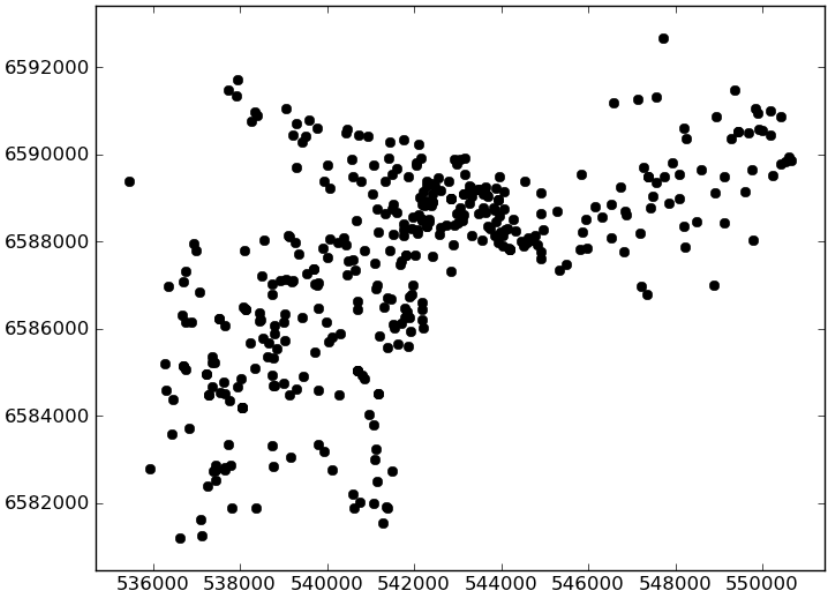
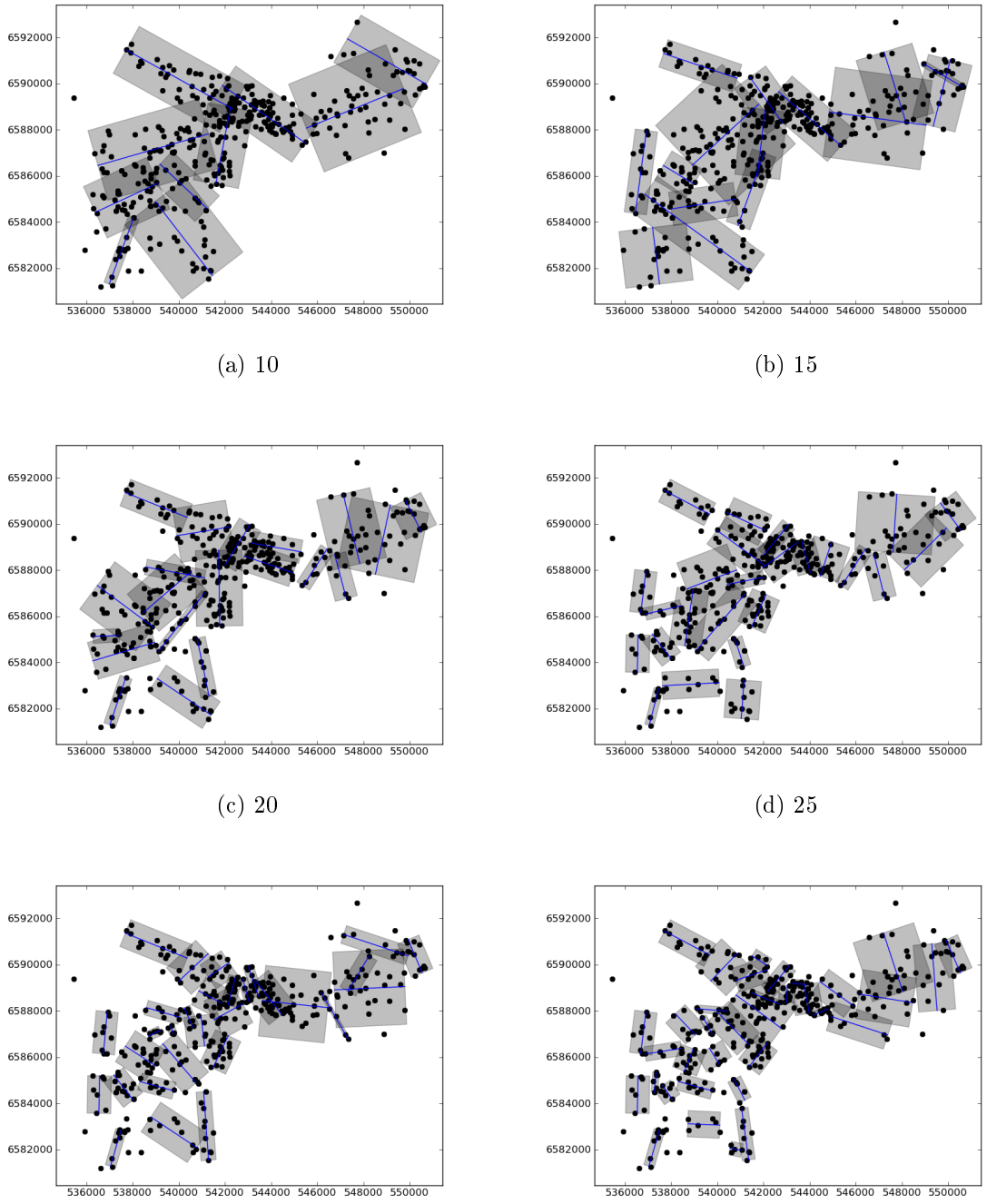



Figure 5.7: Tallinn addresses data set

Most of the parameter values are the same as in case of synthetic data experiments. However, we chose larger outlier probability threshold $\varepsilon_{LoOP}$ , i.e., 90%. The reason is that the sample clearly doesn't contain much outliers so we should only filter out the most extreme ones. Total number of outliers detected using this threshold was 32.

First we fitted our sample to 10, 15, 20, 25, 30 and 35 components to observe the trends in output and overall behavior of algorithm. Corresponding $1 - 0.01$ prediction bands are depicted on Figure (5.8).




(a) 10

(b) 15

(c) 20

(d) 25

(e) 30

(f) 35

Figure 5.8: Addresses clustering results

We can conclude that the algorithm identifies segment regions quite reliably. As we increase the number of components segments tend to become shorter and more accurately fitted to corresponding data groups. This demonstrates effectiveness of

existing data clustering but it also is a sign of overfitting in terms of prediction regions. Recall that for identifying most likely model we select the one with maximum BIC value. We fitted our sample to 3 to 9 components ($M_{min} = 3$, $M_{max} = 9$). According to BIC value the most likely number of components is 6 which roughly corresponds to the number of major city parts. Resulting $1 - 0.01$ prediction band is depicted on Figure (5.9).

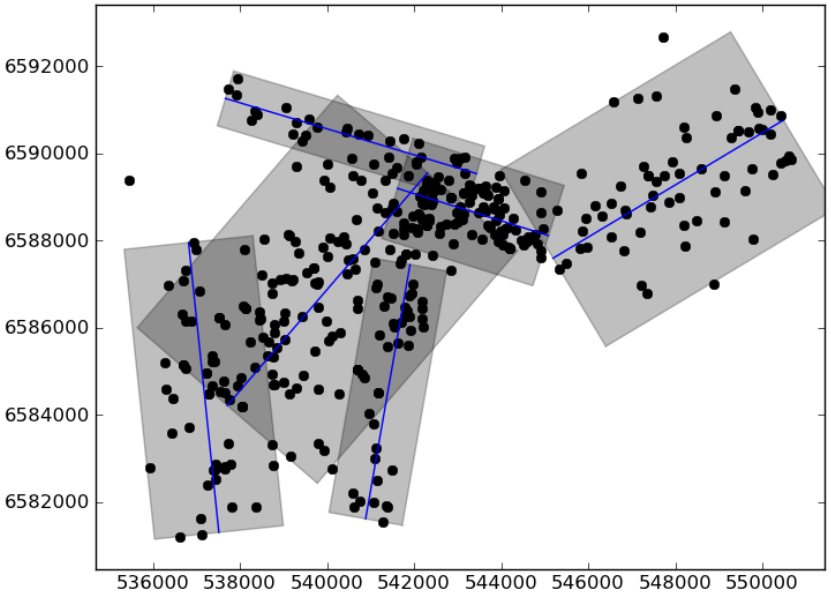



Figure 5.9: Largest BIC value - 6 components

The log likelihood and BIC values of mixtures of 3 to 9 components are listed in Table (5.1).

| M | Log likelihood | BIC |
|---|---|---|
| 3 | -7248.12 | -14664.02 |
| 4 | -7221.84 | -14665.38 |
| 5 | -7195.84 | -14667.28 |
| 6 | -7164.48 | -14658.50 |
| 7 | -7158.81 | -14701.07 |
| 8 | -7152.93 | -14743.24 |
| 9 | -7147.12 | -14785.53 |

Table 5.1: Log likelihood and BIC of mixtures of 3 to 9 components

If we increase the number of components then the likelihood also increases since the sample can be fitted more accurately. BIC value, however, reaches its maximum at 6 components and starts decreasing with each new component.

# Chapter 6

# Summary

In this thesis we have presented algorithms for model-based clustering of two-dimensional geo-tagged events of two types: distributed normally around central points and distributed along line segments. Routines have been presented for hard clustering as well as soft clustering using the Expectation-Maximization algorithm. We have also described techniques for finding prediction regions and for dealing with sample outliers.

Normal data related model-based clustering routines are quite standard and well known. Segment related techniques, however, have been designed specifically for this thesis and are the main contribution of it. We have implemented the described algorithms and conducted practical experiments using both synthetic and real world data to confirm algorithm reliability.

One possible direction of future research is developing routines for fitting data distributed along curved lines. The current approach allows modeling curves as a sequence of multiple straight segments and fitting data to them. Fitting to actual curves, however, would produce much more accurate and realistic results.

Another problem is selecting values for the large number of algorithm parameters. The current work doesn't contain any guidelines for doing that. We used trial and error approach during experiments which is not very practical. More intelligent strategies could be investigated.

# Geograafiliste andmete mudelipõhine rühmitamine

## Magistritöö (30 EAP)


## Roman Tekhov


## Resümee


Töö eesmärk oli disainida ja realiseerida algoritme teatud tüübiga andmete analüüsimiseks. Tegu on geograafiliste koordinaatidega annoteeritud sündmustega (pikkus ja laius). Algoritmide põhiülesanne on ennustada geograafilisi piirkondi, kus sama tüübiga sündmused toimuvad suure tõenäosusega ka tulevikus.

Me vaatlesime kahte tüüpi andmeid. Esimene tüüp on kahemõõtmelise normaaljaotusega sündmused, mis paiknevad ümber keskset punktallikat. Teine tüüp on sündmused, mis on jaotatud mööda keskset joonelõiku. Näiteks liiklusõnnetuste toimumiskohad paiknevad mööda maanteesid, mida saab modelleerida sirgete lõikude kogumina. Esimene andmetüüp on laialt esitatud kirjanduses, ning antud töös kirjeldatakse hästi tuntud sellekohased algoritmid. Teise andmetüübi analüüsimiseks sobivate algoritmide arendus ja esitamine oli aga selle töö peamine panus.

Iga andmetüübi puhul me vaatlesime kahte mudelipõhist klasterdamise viisi. Esimene viis on tavaline rühmitamine, kus iga olemasolev sündmus määratakse täpselt ühte klastri. Punktallikatega sündmuste puhul sobib selle ülesanne jaoks populaarne k-means algoritm, mis on antud töös kirjeldatud. Lõiguallikatega sündmuste jaoks sobilik k-segments algoritm on tuletatud lähtudes k-means algoritmi printsiibist ning arvestades mudeli eripäradega.

Teine viis on pehme rühmitamine, kus iga sündmuse kohta leitakse tema kõikidesse klastritesse kuuluvuse tõenäosused. Antud ülesannet saab lahendada esitades sündmusi tõenäosusliku segumudelina ning rakendades tuntud *Expectation-Maximization* algoritmi. Selle tulemuseks on nii sündmuste kuuluvuse tõenäosused, kui ka sündmuste geograafilise jaotuse parameetrite hinnangud, mis on ühtlasi ka põhiülesanne lahenduseks. Antud töös on kirjeldatud klassikaline protseduur EM algoritmi rakendamiseks normaaljaotusega komponentidest koosneva segumudeli korral. Pärast seda on välja pakutud hinnangufunktsioonid lõiguallikatega komponentidest koosneva segumudeli parameetritele.

Mõlema andmemudeli puhul on toodud algoritmid ennustuspiirkondade arvutamiseks, ehk on kirjeldatud viisid kuidas leida ja visualiseerida piirkondi, mis sisal-

davad tuleviku sündmusi mingi etteantud tõenäosusega. Lõpus on pakutud viis, kuidas saab rühmitada andmeid mis sisaldavad müra, ehk juhuslikke elemente, mis ei kuulu põhikomponentidesse.

Kõik kirjeldatud algoritmid on realiseeritud programmina kasutades Python keelt. Antud töös on esitatud ka programmi abil tehtud eksperimentide kirjeldus ning tulemused.

# Bibliography


[1] Matplotlib. URL `http://matplotlib.sourceforge.net/`.

[2] Numpy. URL `http://numpy.scipy.org/`.

[3] Scipy. URL `http://www.scipy.org/`.

[4] Virtual simulation room project. URL `http://www.ria.ee/vsr/`.

[5] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM, 2007.

[6] A. P. Dempster, N. M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[7] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London*, Series A 222:309–368, 1922.

[8] R A Fisher. On a distribution yielding the error functions of several well known statistics. 1924.

[9] G. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14:403–420, 1970.

[10] G. H. Golub and C. F. Van Loan. An analysis of the total least squares problem. *SIAM J. Numer. Anal.*, 17:883–893, 1980.

[11] Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

[12] R. J. Hathaway. A constrained EM-algorithm for univariate normal mixtures. *Journal of Statistical Computation and Simulation*, 23(3), 1986.

[13] P. J. Huber. *Robust Statistics*. Wiley, 1981.

[14] J. Jensen. Sur les fonctions convexes et les inegalites entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906. ISSN 0001-5962.

[15] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis*. 1998.

[16] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. LoOP: local outlier probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 1649–1652. ACM, 2009.

[17] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, Berkeley, Califonia, 1967. University of California Press.

[18] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, 1936.

[19] F. Mosteller and J. W. Tukey. *Data Analysis and Regression: A Second Course in Statistics*. 1977.

[20] G. Schwarz. Estimating the dimension of a model. *Ann. Stat.*, 14, 1978.

# Appendix A

# Implementation

The implementation code and instructions can be found at `https://bitbucket.org/romantek/model-based-clustering`.