Application of Mixture Models to Threshold Anomaly Scores

MSc Mathematical Sciences

Student: W.N. van Veluw
Studentno.: 6278957
Supervisors: C.W. Oosterlee, J. M. C. Tas, L.A. Souto Arias, J. S. van Balken, A. K. Smit
Second Reader: I.V. Kryven

Abstract

Identifying anomalous observations in data has become more important with the availability of large data sets. Where statisticians were primarily interested in filtering the data for anomalies, in modern applications the anomalies themselves are of interest as well: they could indicate security breaks, potential disease outbreaks or fraudulent transactions. Many anomaly detection algorithms have been developed for this purpose. These algorithms assign an anomaly score to every observation, which increase in the level of anomalousness. However, to formulate conclusive answers to the question which observations are anomalies, thresholding the anomaly scores is necessary. The state-of-the-art method is to select a certain number of observations, which is often motivated by the capacity of processing all selected anomalies. In this thesis, mixture models are applied to determine a threshold in a data-driven way. Being a popular clustering method, mixture models of different compositions are tested on benchmark anomaly detection data sets as well as on a real-world financial data set. In addition, we study the stability of the thresholds through time. We show that mixture models can be applied as truly unsupervised thresholding methods, matching the performance of the state-of-the-art method, but are highly dependent on the exact form of the mixture.

Contents

1 Background

1.1 Introducing Anomalies

The term "anomaly" has many synonyms in the literature, of which "outlier" may be the best known. In this thesis, the two terms will be used interchangeably. The definition of an anomaly or outlier is not set in stone. In fact, numerous scientists have formed their own definitions (table 1 in [2]), specified for the field of science they are working in. One of the earliest and most cited definitions of an anomaly is that of Hawkins [10], who defined an anomaly as follows.

> *"An outlier is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism."*

Although all definitions deviate slightly, the general idea is that anomalies are data points that are significantly dissimilar or deviating in comparison with other data points within the same data set. Figure 1 gives an illustration of anomalies in a two-dimensional data set. Besides one big cluster of observations close to each other, two observations $(-2.5, 3.5)$ and $(6, 0)$ lie relatively far from other observations and are seen as outliers.
Statisticians were already interested in identifying anomalies in data sets since the end of the 19th century ([7]). The interest for identifying anomalies has only increased with the development of fields as machine learning and AI, as anomalies can play a disrupting role in the estimation of models. This is reflected in the definition of Hawkins: in machine learning, one tries to model the mechanism(s) that generated the data set at hand. As anomalies are suspective of being generated by a different mechanism, the estimation of the parameters of a machine learning model can be substantially distorted when these anomalies are included ([16]).
Besides the more theoretical motivation, identifying anomalies also has many real-world applications. In financial data, anomalies are associated with fraudulent customers, market risks or business opportunities. In communication between computers, anomalies might indicate intrusions or security breaks. Detection of potential disease outbreaks is an example of anomaly detection applied in public health. Hence, there are numerous practical examples which justify the interest in the ability to detect anomalies.

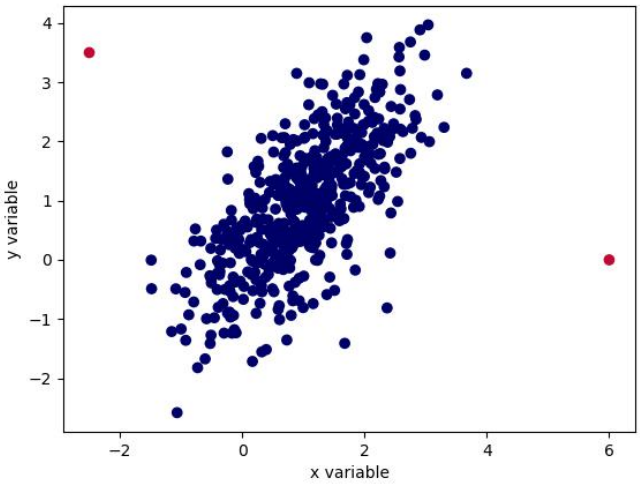


Figure 1: Illustration of anomalies in a two dimensional data set.

1.2 Taxonomy of Anomaly Detection Algorithms

Identifying anomalies in a data set as in Figure 1 can be done visually. This is, however, infeasible for data sets with many dimensions. To detect anomalies in such high-dimensional data sets many anomaly detection algorithms (ADAs) have been developed ([4], [9]) and, undoubtedly, many more will be published in the coming years. The Python module PyOD ([21]) contains many benchmark ADAs as well as complex and innovative algorithms to identify anomalies. The documentation of this package therefore gives also a nice overview of all available ADAs, see [22]. The module has, as well as many surveys on anomaly detection, created its own taxonomy to characterise ADAs. This thesis follows mostly the line of the PyOD taxonomy, but characterises ADAs by the *way of learning*, the *formalisation of dissimilarity* and whether or not an algorithm is *(non)parametric*.

Way of Learning

The way of learning of an ADA mostly depends on the availability of labeled data. If all the observations are labeled, one could use these labels to learn patterns in the data. This type of learning is known as *supervised* learning. The detection of anomalies is then similar to a binary classification problem, one class being the inliers and the other being the outliers.
When the labels of the observations are not available, the models learn the patterns in an *unsupervised* manner. In this case, anomaly detection resembles clustering problems. Assuming that inliers show similar characteristics, the most similar data instances are clustered together forming the cluster of inliers. The observations that fall outside this cluster are considered anomalies. Note that we are not necessarily bounded to modeling two clusters, as many ADAs can work with several clusters of inliers and/or outliers.
If the labels are only partly available, the type of learning is a mixture of supervised and unsupervised learning known as *semi-supervised* learning. Within this type of learning, the ideas from supervised and unsupervised models are combined in order to learn from all available data.

The ADAs learning in an unsupervised manner are often useful in practical applications as obtaining the labels is almost never easy. Anomalies are rare events and do not occur often. Getting enough labelled data to perceive patterns might be too hard. In addition, labeling data instances as anomalies is often done by domain experts in practice. This can be time-consuming, resulting in too few labeled points to apply supervised learning. Detecting anomalies may lie therefore closer to the nature of unsupervised or semi-supervised methods.

Definition of Anomalousness

In the definition by Hawkins, it is stated that anomalies should deviate much from other observations in the same data set. However, it is not clear in what sense observations should deviate. All ADAs formalise this notion of "deviation" or *anomalousness* in their own way.
The most used formalisation of anomalousness is in terms of the (Euclidian) distance to other observations in the data set. Indeed, it is reasonable to assume that observations that lie far away from each other might originate from two different mechanisms. This insight forms the foundation of many state-of-the art ADAs, such as the k-Nearest Neighbour (kNN) and Local Outlier Factor (LOF) algorithms. Although formalising anomalousness by distance seems to be a natural way, the distance-based algorithms have one backfall. It is known that the (Euclidian) distance measure suffers from *the curse of dimensionality* ([11], section 6.4) and, therefore, kNN and LOF do as well. This implies that the performance of these algorithms decreases when the dimensionality of the data set increases.
The probabilistic models in the PyOD module formalise anomalousness by means of *density*, i.e. anomalous observations occur in areas of low density. Generally a probabilistic model is fitted to the data and the density according to every observation is computed. The Gaussian Mixture Model (GMM) algorithm is an example of a probabilistic ADA.
A different and rather new formalisation of anomalousness is the concept of *isolation*. This formalisation builds upon the data set being randomly split. An anomalous observation would be isolated after just a few

splits, as it is assumed that this observation lies far from other observations. In contrast, isolating a normal observation would require many random splits. One of the most basic algorithms that work with the concept of isolation is the isolation tree. Due to the probabilistic aspect introduced by random splitting, the effect of randomness is mitigated by combining many isolation trees into an isolation forest (iForest).
The last group of ADAs also computes distances, but not between observations. Instead, the observations are projected into a subspace with lower dimensionality or onto a multivariate Gaussian distribution. The former is known as Principal Components Analysis (PCA), which computes anomaly scores as the sum of weighted distances of an observation to the principal components. The latter algorithm is known as Minimum Covariance Determinant (MCD). For MCD, the parameters of a multivariate Gaussian distribution are estimated in a way that is robust to anomalies. Then, the Mahalanobis distance is used as anomaly score. PyOD groups these methods under the header "linear models", as does this thesis.
Besides ADAs that work with just one formalisation of anomalousness, in recent years attempts to combine different formalisations into one algorithm have been made. For example, the Analytic Isolation and Distance-Based Anomaly Detection (AIDA) algorithm ([1]) combines the concepts of distance and isolation.

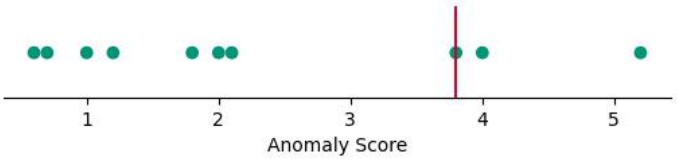
In summary, four formalisations of anomalousness have been introduced together with seven ADAs. For more information on kNN, LOF, GMM, PCA and MCD, we refer to the documentation of PyOD and the benchmarking paper [9].

Parametric or Nonparametric

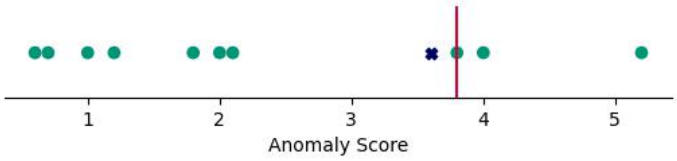
The term *nonparametric* refers to models that are fully specified by their hyperparameters, as opposed to *parametric* models which involve parameters to be estimated from the data. The kNN is a classic example of a nonparametric model, as it is specified whenever the value of k is given. On the contrary, GMMs are examples of parametric models as they involve estimating the parameters of the Gaussian distribution.
The difference between these two types of models is mainly in the assumptions underlying them. In parametric models, assumptions are often made about the form of a distribution. This is also the case with GMMs, as it is assumed that the data is (high-dimensional) Gaussian distributed. Assumptions about the form of a distribution are not needed in nonparametric models.
In general, it holds that parametric models outperform nonparametric models if the assumptions made are correct. If the assumptions were made falsely, the parametric models perform worse than their nonparametric relatives.

1.3 Thresholding Anomaly Scores

Despite their differences in the way of learning, definition of anomalousness and the use of parameters, the ADAs described in Section 1.2 have one thing in common: they output a set of real numbers called the anomaly scores. The anomaly scores are values that describe the level of suspiciousness. In general, it holds that a higher anomaly score strengthens the belief that the observation in question is an anomaly.
The anomaly scores of the different ADAs come with two challenges. The first challenge concerns the interpretability of the scores. Indeed, in general it holds that a higher anomaly score makes an observation more suspicious, but the output of different ADAs may differ in meaning and scale. For example, the anomaly scores outputted by kNN are the distances between observations and their k-th nearest neighbour. However, the output of an iForest is the average number of splits required to isolate the observation. The anomaly scores outputted by kNN and iForest may therefore be similar in value, but differ in terms of meaning. In addition, it is also not the case that an observation is twice as suspicious when its anomaly score is twice as high compared to other observations. Lastly, there exist ADAs which assign low scores to anomalous observations, such as GMM. Therefore, lower anomaly scores indicate anomalous observations instead of higher scores. In that case, the anomaly scores from GMM should therefore also be interpreted differently.
The second challenge comes with the fact that anomaly scores do not give conclusive answers on which observations can be considered anomalies. The general rule "the higher, the more suspicious" is useful, but is indecisive on the point where the anomaly score is "too high". To assign the labels "inlier" and "outlier"



(a) Data set on which threshold is computed.



(b) New observations recorded (the blue cross).

Figure 2: Illustration of the problem when the threshold is set equal to an anomaly score.

to observations, the anomaly scores should be thresholded. In case the anomaly score of an observation exceeds this threshold, the observation can be considered an anomaly. Due to the difference in meaning and scale, there is not a universal method to threshold the anomaly scores.

In practical applications, putting some thought into the value of the threshold is not a waste of time. Indeed, similar to hypothesis testing the type 1 and type 2 error may play an important role. For example, in fraud detection the anomalous observations are considered fraudulent customers. However, there is a risk in accusing these customers of fraud. Considering the null hypothesis as "the customer is fraudulent", the type 1 error is interpreted as letting a fraudulent customer go free, whereas the type 2 error stands for falsely accusing a customer of fraud. The tendency is to assure the type 2 error is as low as possible, as false accusations should be avoided. These types of considerations should be taken into account when setting a threshold.
Other desirable properties of a threshold might be the following.

- A data-driven threshold. The threshold should be motivated by the data instead of solely practical considerations.
- A stable threshold. The threshold should not change heavily if the set of anomaly scores changes only slightly.
- An in-between threshold. Suppose the observations in a data set are assigned anomaly scores as in Figure 2a. If the threshold is set equal to the second highest anomaly score, two observations would be marked as anomaly. Now suppose a new observation occurs and gets assigned an anomaly score close to this second highest anomaly score, see Figure 2b. As this new observation has an anomaly score way closer to the threshold than to the highest inlier, it is questionable that this observation is - according to the threshold - marked as inlier. Ideally, the threshold should take into account large gaps between inliers and outliers and should compute a threshold somewhere in between.

1.4 Related Work

Research on how to threshold a set of anomaly scores $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ has become more popular during the last decades. In [20], a small study of applied thresholding techniques is done, resulting in the histogram of Figure 3. For the study, the researchers searched for the term 'outlier detection' on Google scholar. From the 38,900 hits within a two year period (2016-2018) 100 papers were randomly sampled.

According to Figure 3, the most popular thresholding technique is called "top-N". This technique corresponds to choosing the threshold as the observation with the N-th highest anomaly score. In a data set of size n, this naturally results in N outliers or an outlier percentage of $\frac{N}{n} \cdot 100\%$. A similar technique, although phrased differently, is not to specify the number of outliers but this outlier percentage, also called contamination level η. In that case, the threshold is chosen such that (at least) $\eta \cdot 100\%$ of the observations is labelled as anomaly. The default thresholding technique for ADAs implemented in the PyOD module is also by means of this contamination level.
It may be clear that this is not a natural, data-driven way of setting a threshold, but is rather motivated

by practical applications. In practical applications, it is often the procedure that the selected outliers are investigated in more detail by domain experts. The (team of) domain experts have a maximal capacity of processing these anomalies, therefore choosing the number of anomalies N or the contamination level η in such a way that the observations marked as outlier can all be processed. In other words, the threshold is not determined based on patterns in the data, but rather on processing capacity. This could result in missing anomalies, if the number of anomalies is greater than the processing capacity, or marking too many observations as anomalies, if there are fewer anomalies than the processing capacity. In case of the latter, the domain experts spend time investigating observations that are inliers in the first place and, therefore, wasting time. Determining the threshold in a data-driven way, i.e. based on patterns in the data, is the optimal method in the opinion of the author.

The aim of [18] is to estimate the contamination level by fitting a Bayesian Gaussian mixture model. The proposed method there is to estimate the distribution of the contamination level, i.e. the posterior distribution, and then setting the contamination level to the mean of this posterior distribution. This estimated contamination level is then used in the Top-N strategy.

Besides the method of specifying the number or the contamination level of outliers, the (manual) specification of a threshold parameter is the second manual thresholding technique according to Figure 3. This technique might be feasible for domain experts working directly on data sets with relatively low dimensions. However, this technique of setting a threshold is not data-driven and can be very hard in high-dimensional data. In addition, interpreting the anomaly scores outputted by an ADA in terms of the features of the data set is very hard, making it too complex to determine a threshold by hand.
As automatic thresholding techniques, Figure 3 shows the methods of Standard Deviation (SD), Median Absolute Deviation (MAD) and Inter Quartile Range (IQR). The threshold T is defined, respectively, as

$$\begin{aligned} T_{SD} &= mean + c_{SD} \cdot SD; \\ T_{MAD} &= median + c_{MAD} \cdot MAD; \\ T_{IQR} &= Q_3 + c_{IQR} \cdot IQR, \end{aligned}$$

with $mean$ as the mean of $\mathcal{S}$, SD as the (sample) standard deviation of $\mathcal{S}$, $median$ as the median of $\mathcal{S}$, MAD as

$$MAD = b \cdot median(\ \{\ |\ s - median(\mathcal{S})\ |\ ; \forall s \in \mathcal{S}\}\ )$$

with $b = 1.482$ as suggested in [19], Q_3 as the third quartile or 75-th percentile of $\mathcal{S}$ and IQR as $IQR = Q_3 - Q_1$ with Q_1 as the first quartile or 25-th percentile of $\mathcal{S}$. The three automatic thresholding techniques described above require specification of a hyperparameter c. In general, the values of c_{SD} and c_{MAD} are set to 3, motivated by the tail probabilities of the Gaussian distribution. The value of c_{IQR} is often set to 1.5. The problem with these methods of thresholding is the choice of hyperparameter c. Although the specified values for c are motivated by the characteristics of the Gaussian distribution, they are still somewhat arbitrary. There is not really a conclusive way to choose between $c = 3$ and $c = 3.5$. In addition, it might not even make sense to threshold anomaly scores by a method that is motivated by the Gaussian distribution. For instance, using kNN as ADA the outputted anomaly scores are distances and therefore never negative. To find a data-driven threshold, one would ideally tune hyperparameter c. Hence, one would optimise a certain performance measure, for example accuracy, F1-score, AUC or even more complex measures. It is important to note that these measures require the true labels of the observations. Hence, when considering supervised learning, tuning hyperparameter c is feasible. In unsupervised learning, it is infeasible to choose the value of c in a data-driven way.

A different approach is to transform the anomaly scores into the range [0,1]. In [12], several methods have been proposed to transform the anomaly scores. Examples of transformations are the minmax-conversion or linear normalisation, Gaussian scaling or Gamma scaling. After transformation, it is stated that the anomaly scores can be interpreted as probabilities of being an outlier. However, this is not enough to determine a

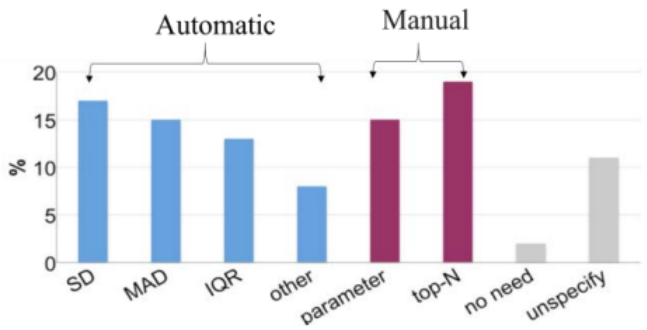


Figure 3: Frequency of used thresholding techniques. *Source [20].*

threshold, as the question at which point a probability is high enough to conclude that an observation is anomalous remains. In other words, transforming the anomaly scores to the range [0,1] could be a useful step, but the problem of thresholding the anomaly scores/probabilities is not solved.

In [8], the anomaly scores are also transformed into probabilities by means of the sigmoid function and a mixture model. The former is a well-known function in machine learning, often used in logistic regression for binary classification. The usage of this function makes sense in that way, as the two classes are the inliers and the outliers. The latter is, opposed to the work in [18], not a Gaussian mixture model. Instead, a mixture of a Gaussian and an exponential distribution is proposed to transform the output of the kNN algorithm into probabilities. Both methods have the advantage of not only modelling the probability of being an outlier, but also the probability of being an inlier. With these two probabilities modeled, the authors propose a Bayesian risk model to infer the threshold.

Lastly, the Python module PyThresh ([13]) contains 29 thresholding techniques, including the above mentioned techniques among others. One of the thresholding techniques is the Gaussian mixture model, but PyThresh does not include mixtures with other types of distributions. The creators also provided some benchmarking of the thresholding techniques by considering the anomaly scores outputted by six ADAs evaluated on 16 data sets. Although the Gaussian mixture model is implemented via the Bayesian approach from [18], PyThresh does not include mixtures consisting of any other distributions than the Gaussian. The results in [8] give reason that including other distributions in the mixture might increase the performance of the mixture.

1.5 Research Questions and Aim

The aim of this thesis is to investigate methods to calibrate a threshold on a set of anomaly scores. Important requirements for the method are the following.

- The method should be data-drive, i.e. not something like the Top-N strategy;
- The method should not involve any hyperparameters to choose/tune, i.e. not something like the SD, MAD or IQR methods;
- The method should preferably produce an in-between threshold, see Section 1.3.

Based on the literature, mixture models are methods that satisfy all three requirements. In this work the performance of mixture models-based threshold calibration is studied. For that purpose, the research questions are formulated as follows.

Question 1: What mixture to use?
When mixture models are applied, the mixture of choice is often a Gaussian mixture model (GMM), involving two Gaussian/normal distributions. This can readily be seen from the fact that popular machine learning modules in Python - for instance Scikit-learn and PyOD - have the GMM implemented. The PyThresh module even has the Bayesian GMM from [18] implemented. However, [8] argues that an exponential-normal mixture might be more appropriate when transforming anomaly scores to probabilities. In addition, several papers in the financial domain use a lognormal-Pareto mixture ([3]).

Question 2: How does the mixture model-based threshold compare with other thresholders?
To answer this question, the benchmarking work of PyThresh forms the most important comparison material. The results of the mixture model-based thresholders are compared to the thresholding methods in PyThresh.

Question 3: Are the mixture model-based thresholds stable?
The thresholds resulting from the mixture models should not be affected greatly when the training data changes slightly. If this would be the case, applying mixture model-based threshold in the financial domain would be infeasible.

One remark should be made before the focus is put on mixture models. A famous saying in machine learning is that "there is no such thing as a free lunch". Although mixture models check all of the boxes on the list of desired properties, e.g. it is a data-driven method that does not require tuning/choosing hyperparameters, they do come with a cost. In the case of mixture models, the cost is payed in assuming which distributions to include in the mixture. Indeed, when the anomaly scores follow a mixture consisting of a Gamma and an uniform distribution, fitting a normal-normal mixture will certainly result in worse performance. Therefore, even mixtures can not guarantee free lunch.

2 Methodology

In Section 1.4, the choice of using mixture models to set thresholds on anomaly scores was motivated. This chapter discusses the way of fitting these mixture models and explains how to determine a threshold from this fit. Lastly, some notes are made on the code that implements the methodology and an illustrative example of the algorithms is given.

2.1 Foundations of Mixture Models

In general, a mixture model consists of a combination of K distributions, also called *components*. The density of a random variable X is then modeled as

$$f(x) = \prod_{k=1}^{K} \omega_k f_k(x \mid \boldsymbol{\theta}_k), \tag{1}$$

where ω_k is called the mixing proportion or prior probability of component k and $\boldsymbol{\theta}_k$ are the distribution-specific parameters of component k. An important condition is that $\sum_k \omega_k = 1$. Mixture models are often used as a classification or clustering method, such that every component models one class or cluster.
In this thesis, we want to separate the outliers from the inliers, which could be thought of as modeling two classes. Hence, the mixture models considered in this thesis will consist of two components.

Two important random variables used in the mixture are the anomaly score S and the class assignment T, also called the label. Hence, $T = 0$ is associated with the class of inliers and $T = 1$ with the class of outliers. Important to note is that the variable S can be observed, or computed by an ADA, whereas variable T can not be observed and is a so-called *latent* variable. The assumptions that form the basis of the mixture model are the following.

1) The observations are independent and identically distributed.
2) The inliers follow a distribution F_0 with density $f_0(s \mid \boldsymbol{\theta}_0)$, alternatively denoted as $f_{S|T=0}(s \mid \boldsymbol{\theta}_0)$.
3) The outliers follow a distribution F_1 with density $f_1(s \mid \boldsymbol{\theta}_1)$, alternatively denoted as $f_{S|T=1}(s \mid \boldsymbol{\theta}_1)$.
4) The prior probability of a data instance being an outlier is ω, i.e. $P(T = 1) = \omega$, from which it follows that $P(T = 0) = 1 - \omega$.

With the two conditional densities for S given T and the marginal (prior) density of T described above, we can infer all other densities that can be made by S and T. These densities are the marginal density of S, the joint density of S and T and, lastly, the conditional density of T given S. In the derivation of these densities, the law of total probability and Bayes' rule play an important role ([6], chapter 3).

The marginal density of the anomaly score S can be found by applying the rule of total probability and Bayes' rule. This yields

$$\begin{aligned} f_S(s) &= f_{S,T}(s, T = 0) + f_{S,T}(s, T = 1) \\ &= P(T = 0) f_0(s \mid \boldsymbol{\theta}_0) + P(T = 1) f_1(s \mid \boldsymbol{\theta}_1) \\ &= (1 - \omega) f_0(s \mid \boldsymbol{\theta}_0) + \omega f_1(s \mid \boldsymbol{\theta}_1) \end{aligned} \tag{2}$$

The density described in (2) is referred to as *the mixture*. It has the same form as (1) with $K = 2$.

For the joint density of S and T, we apply Bayes' rule again to obtain

$$\begin{aligned} f_{S,T}(S = s, T = t) &= P(T = t) f_{S|T}(S = s \mid T = t) \\ &= \begin{cases} (1 - \omega) f_0(s \mid \boldsymbol{\theta}_0) & \text{if } t = 0; \\ \omega f_1(s \mid \boldsymbol{\theta}_1) & \text{if } t = 1; \end{cases} \end{aligned}$$

There are several ways in which this density can be written in one line. For example

$$f_{S,T}(S = s, T = t) = \mathbb{I}_{\{t=0\}}(1-\omega)f_0(s \mid \boldsymbol{\theta}_0) + \mathbb{I}_{\{t=1\}}\omega f_1(s \mid \boldsymbol{\theta}_1),$$

with $\mathbb{I}_{\{\}}$ denoting the indicator function, or

$$\begin{aligned} f_{S,T}(S = s, T = t) &= (1-t)(1-\omega)f_0(s \mid \boldsymbol{\theta}_0) + t\omega f_1(s \mid \boldsymbol{\theta}_1), \text{ or} \\ f_{S,T}(S = s, T = t) &= \Big((1-\omega)f_0(s \mid \boldsymbol{\theta}_0)\Big)^{1-t}\Big(\omega f_1(s \mid \boldsymbol{\theta}_1)\Big)^{t} \end{aligned} \tag{3}$$

For the remainder of this thesis we will use (3), as it allows for a concise expression of the log-likelihoods.

Lastly, the conditional density of T given S can be found by applying Bayes' rule once more and substituting (2) and (3).

$$\begin{aligned} f_{T|S}(T = t \mid S = s) &= \frac{f_{S,T}(S = s, T = t)}{f_S(S = s)}, \\ &= \frac{\Big((1-\omega)f_0(s \mid \boldsymbol{\theta}_0)\Big)^{1-t}\Big(\omega f_1(s \mid \boldsymbol{\theta}_1)\Big)^{t}}{(1-\omega)f_0(s \mid \boldsymbol{\theta}_0) + \omega f_1(s \mid \boldsymbol{\theta}_1)}. \end{aligned} \tag{4}$$

Note that if t is equal to 0 or 1, the density described above simplifies to respectively

$$\begin{aligned} f_{T|S}(T = 0 \mid S = s) &= \frac{(1-\omega)f_0(s \mid \boldsymbol{\theta}_0)}{(1-\omega)f_0(s \mid \boldsymbol{\theta}_0) + \omega f_1(s \mid \boldsymbol{\theta}_1)}, \\ f_{T|S}(T = 1 \mid S = s) &= \frac{\omega f_1(s \mid \boldsymbol{\theta}_1)}{(1-\omega)f_0(s \mid \boldsymbol{\theta}_0) + \omega f_1(s \mid \boldsymbol{\theta}_1)}. \end{aligned}$$

The conditional density of T given S is often referred to as *the posterior density*.

Note that all densities described above are implicitly defined given the parameters $\omega, \boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$. For notational purposes, this condition is often discarded.

2.2 Fitting Mixture Models and Inferring the Threshold

In order to fit a mixture model, we need to estimate the parameters $\omega, \boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ from a set of anomaly scores $\mathcal{S} = \{s_1, s_2, ..., s_n\}$. Let us denote $\Theta = (\omega, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$ as the vector of parameters. The method of maximum likelihood is a well-known and popular method to estimate parameters of a distribution. In this section, we consider two ways of applying the concept of maximum likelihood estimation, which differ in the way of handling the latent class label.
As the mixture defines a density stated in (2), the first method to estimate Θ is by just applying maximum likelihood directly. This way, the latent class labels are ignored. The second method is a well-known estimation procedure that handles latent data: the Expectation-Maximisation algorithm.

2.2.1 Direct Maximum Likelihood Estimation

The direct maximum likelihood method (DML) maximises the *observed likelihood*. This likelihood only includes the observed data $\mathcal{S}$ and the density described in (2). Hence, the observed likelihood equals

$$\begin{aligned} L_{obs}(\Theta \mid \mathcal{S}) &= \prod_{i=1}^{n} f_S(S = s_i \mid \Theta), \\ &= \prod_{i=1}^{n} (1-\omega)f_0(s_i \mid \boldsymbol{\theta}_0) + \omega f_1(s_i \mid \boldsymbol{\theta}_1), \end{aligned}$$

and therefore the observed log-likelihood equals

$$\ell_{obs}(\Theta \mid \mathcal{S}) = \sum_{i=1}^{n} \log\Big((1-\omega) f_0(s_i \mid \boldsymbol{\theta}_0) + \omega f_1(s_i \mid \boldsymbol{\theta}_1)\Big). \tag{5}$$

The parameters will then be estimated as

$$\hat{\Theta}_{DML} = (\hat{\omega}, \hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\theta}}_1)_{DML} = \underset{\omega, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1}{\arg\max}\, \ell_{obs}(\omega, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1 \mid \mathcal{S}). \tag{6}$$

We will refer to (6) as *direct maximum likelihood* (DML) estimation, as it does not include the latent variable T. Instead, the estimates of $\omega, \boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ can be directly derived from the observed data. Due to its form, (6) can not be solved analytically and we rely on numerical optimisation. Many optimisation algorithms, such as BFGS, Nelder-Mead, Newton-CG or any other Newton's-type algorithms, are implemented in the package scipy.optimize. The results presented in this thesis rely on this package as well. In this thesis, the L-BFGS-B method is used in DML estimation, as it is an algorithm allowing for bounds to be set. This property is necessary, for example, to make sure that outputted estimates of variance parameter are non-zero. To start the algorithm, an initial guess for the parameters $\hat{\Theta}^{(0)}$ is required. More on forming this initial guess in section 2.2.4.

2.2.2 Expectation-Maximisation Algorithm

When the latent class labels are included in the estimation method, the *complete likelihood* is maximised. This likelihood models the joint density of the observed and latent variables described in (3). Hence, the complete likelihood yields

$$\begin{aligned} L_{com}(\Theta \mid \mathcal{S}, \mathcal{T}) &= \prod_{i=1}^{n} f_{S,T}(s_i, t_i \mid \Theta), \\ &= \prod_{i=1}^{n} \Big((1-\omega) f_0(s_i \mid \boldsymbol{\theta}_0)\Big)^{1-t_i} \Big(\omega f_1(s_i \mid \boldsymbol{\theta}_1)\Big)^{t_i}. \end{aligned}$$

The complete log-likelihood is the natural logarithm of the complete likelihood, i.e.

$$\begin{aligned} \ell_{com}(\Theta \mid \mathcal{S}, \mathcal{T}) &= \log(L_{com}(\Theta \mid \mathcal{S}, \mathcal{T})) \\ &= \sum_{i=1}^{n} \log\Big(\Big((1-\omega) f_0(s_i \mid \boldsymbol{\theta}_0)\Big)^{1-t_i} \Big(\omega f_1(s_i \mid \boldsymbol{\theta}_1)\Big)^{t_i}\Big). \end{aligned}$$

This can be rewritten as

$$\ell_{com}(\Theta \mid \mathcal{S}, \mathcal{T}) = \sum_{i=1}^{n} (1-t_i)\Big(\log(1-\omega) + \log(f_0(s_i \mid \boldsymbol{\theta}_0))\Big) + t_i\Big(\log(\omega) + \log(f_1(s_i \mid \boldsymbol{\theta}_1))\Big). \tag{7}$$

Due to the inclusion of the latent data set $\mathcal{T}$, (7) can not be maximised directly. A well-known and famous algorithm that copes with latent data is the Expectation-Maximisation (EM) algorithm ([15]). In short, the EM-algorithm produces a sequence of parameter estimates $\hat{\Theta}^{(0)}, \hat{\Theta}^{(1)}, ...$ and label estimates $\hat{\mathcal{T}}^{(0)}, \hat{\mathcal{T}}^{(1)}, ...$, until the parameter estimates converge, i.e. for some k it holds that $\mid \hat{\Theta}^{(k)} - \hat{\Theta}^{(k+1)} \mid \leq \varepsilon$ elementwise. Then, the output is given by

$$\hat{\Theta}_{EM} = (\hat{\omega}, \hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\theta}}_1)_{EM} = \hat{\Theta}^{(k)}.$$

To update the set of parameters, i.e. to compute $\hat{\Theta}^{(k+1)}$ and $\hat{\mathcal{T}}^{(k+1)}$ from $\hat{\Theta}^{(k)}$ and $\hat{\mathcal{T}}^{(k)}$, the algorithm performs the Expectation (E) step and the Maximisation (M) step. Both steps are elaborated on below. To start the EM-algorithm, the scientist has to provide initial guesses $\hat{\Theta}^{(0)}$ for the parameters to estimate.

More on the initialisation in Section 2.2.4.

In the E-step, the expected complete log-likelihood given the observed data $\mathcal{S}$ and previous estimates $\hat{\Theta}^{(k)}$ is computed, with respect to the posterior distribution of T. Mathematically speaking, this yields

$$\begin{aligned} Q(\Theta, \hat{\Theta}^{(k)}) :&= E_{T|S}\big[\ell_{com}(\Theta \mid \mathcal{S}, \mathcal{T}) \mid \mathcal{S}, \hat{\Theta}^{(k)}\big], \\ &= E_{T|S}\Big[\sum_{i=1}^{n}(1-t_i)\Big(\log(1-\omega) + \log(f_0(s_i \mid \boldsymbol{\theta}_0))\Big), \\ &\qquad + t_i\Big(\log(\omega) + \log(f_1(s_i \mid \boldsymbol{\theta}_1))\Big) \mid \mathcal{S}, \hat{\Theta}^{(k)}\Big]. \end{aligned}$$

As the terms with ω and the densities f_0 and f_1 do not include t_i's, we have

$$\begin{aligned} &= \sum_{i=1}^{n}\Big(1 - E_{T|S}[t_i \mid s_i, \hat{\Theta}^{(k)}]\Big)\Big(\log(1-\omega) + \log(f_0(s_i \mid \boldsymbol{\theta}_0))\Big), \\ &\qquad + E_{T|S}[t_i \mid s_i, \hat{\Theta}^{(k)}]\Big(\log(\omega) + \log(f_1(s_i \mid \boldsymbol{\theta}_1))\Big). \end{aligned} \tag{8}$$

For the expectation of the class label t_i we use (4) to find

$$\begin{aligned} \hat{t}_i^{(k)} :&= E_{T|S}[t_i \mid s_i, \hat{\Theta}^{(k)}] \\ &= 0 \cdot f_{T|S}(t_i = 0 \mid s_i, \hat{\Theta}^{(k)}) + 1 \cdot f_{T|S}(t_i = 1 \mid s_i, \hat{\Theta}^{(k)}) \\ &= \frac{\hat{\omega}^{(k)} f_1(s_i \mid \hat{\boldsymbol{\theta}}_1^{(k)})}{(1-\hat{\omega}^{(k)}) f_0(s_i \mid \hat{\boldsymbol{\theta}}_0^{(k)}) + \hat{\omega}^{(k)} f_1(s_i \mid \hat{\boldsymbol{\theta}}_1^{(k)})}. \end{aligned} \tag{9}$$

Note that, since there are probabilities and densities involved, these estimated class labels are not necessarily equal to 0 or 1, but may be decimal values as well. Substituting this into (8) gives

$$\begin{aligned} Q(\Theta, \hat{\Theta}^{(k)}) = \sum_{i=1}^{n}&(1-\hat{t}_i^{(k)})\Big(\log(1-\omega) + \log(f_0(s_i \mid \boldsymbol{\theta}_0))\Big) \\ &+ \hat{t}_i^{(k)}\Big(\log(\omega) + \log(f_1(s_i \mid \boldsymbol{\theta}_1))\Big). \end{aligned} \tag{10}$$

In the M-step, the expected complete log-likelihood described in (10) is maximised with substitution of the updated class label estimates $\hat{\mathcal{T}}^{(k)}$. This way, we obtain new parameter estimates $\hat{\Theta}^{(k+1)}$:

$$\hat{\Theta}^{(k+1)} = \arg\max_{\Theta} Q(\Theta, \hat{\Theta}^{(k)}).$$

The EM-algorithm is listed in Algorithm 1. Appendix A constructs the update rules for several distribution-specific parameters in the M-step. The update rule for ω does not depend on the chosen distributions, as the terms in (10) involving ω do not include any other parameters. Therefore, the update rule for ω can be found by taking the partial derivative w.r.t. ω and finding the root. The partial derivative equals

$$\frac{\partial}{\partial \omega} Q = \sum_{i=1}^{n} -\frac{1-\hat{t}_i^{(k)}}{1-\omega} + \frac{\hat{t}_i^{(k)}}{\omega}.$$

Equating to zero and solving for ω gives

$$\omega = \frac{1}{n}\sum_{i=1}^{n} \hat{t}_i^{(k)}.$$

No matter what distributions are chosen, the update rule for ω in the M-step will always be

$$\hat{\omega}^{(k+1)} = \frac{1}{n}\sum_{i=1}^{n} \hat{t}_i^{(k)}.$$

Algorithm 1: EM-algorithm

Input: Set of anomaly scores $\mathcal{S}$, convergence threshold ε, intial guesses $\hat{\Theta}^{(0)}$.
Output: Parameter estimates $\hat{\Theta}$.

1) E-step: construct label estimates $\hat{\mathcal{T}}^{(k+1)}$ by (9).
2) M-step: construct new parameter estimates as $\hat{\Theta}^{(k+1)} = \arg\max_{\Theta} Q(\Theta, \hat{\Theta}^{(k)})$.
3) If $\mid \hat{\Theta}^{(k)} - \hat{\Theta}^{(k+1)} \mid \leq \varepsilon$ holds elementwise, return $\hat{\Theta} = \hat{\Theta}^{(k)}$. Otherwise, go back to 2).

2.2.3 Which Fitting Method to Use?

In sections 2.2.1 and 2.2.2, two ways of estimating the parameters of a mixture have been discussed. In this subsection, the (dis)advantages of both fitting methods are discussed from a theoretical point of view. In addition, a numerical comparison is made in section 3.1.

An advantage of the EM algorithm over the DML by means of Newton's-type algorithms - such as the L-BFGS-B method - is the assurance of increasing the likelihood at every step, converging to a (local) maximum. This is in contrast with Newton's-type algorithms, which have the probability to 'overshoot' the (local) maximum ([14]). In addition, the EM-algorithm fits nicely in the story of maximum likelihood estimation, as the update rules in the M-step are very similar to well-known maximum likelihood estimators of familiar distributions. On the contrary, the update rules of Newton's-type are not always that clear, due to the need to numerically approximate the Jacobian or Hessian of the (observed) log-likelihood. The power of the EM-algorithm lies in its simple outline and closed-form update rules.
The major advantage of Newton's-type algorithm is in its general applicability. The EM algorithm is relatively fast when the distributions to fit have closed-form update rules. However, when these closed-form update rules do not exist, such as with the gamma or beta distributions, the execution time of the M-step increases. This is due to the execution of a numerical solver in the M-step. In that case, the EM algorithm loses its power. Instead of executing a numerical solver every M-step, one could better run an optimisation algorithm - the DML algorithm - once.

In summary, the EM-algorithm is favoured by some, as it assures the log-likelihood to increase in every iteration, it has well-known and familiar update rules and is relatively simple. If time consumption is an important aspect of the research, the DML method might be preferable, especially when distributions that lack closed-form maximum likelihood estimates are included in the mixture.

2.2.4 Forming an Initial Guess

Both the DML and the EM-algorithms require initial guesses $\hat{\Theta}^{(0)} = (\hat{\omega}, \hat{\boldsymbol{\theta}}_0^{(0)}, \hat{\boldsymbol{\theta}}_1^{(0)})$ to start the maximisation. As the observed and complete log-likelihoods are not convex in general, there can be local maxima. As a consequence, the results of the algorithms are depending on this initial guess $\hat{\Theta}^{(0)}$. Therefore, it is useful to put some thought in forming this initial guess.
It is not straightforward to guess the value of parameters in the mixture, as the mixtures can get quite complex and the dependencies are not always clear. As the range of the estimated class labels $\hat{t}_i^{(0)}$ is known, these are guessed first. Indeed, the estimated class labels are always in the range [0,1]. With the guessed values of $\hat{t}_i^{(0)}$, the initial guesses for the parameters can be computed by the updating rules following from

the M-step of the EM-algorithm. Hence, when estimating the parameters by DML, the initial guesses will also be computed by the results from the M-step.
Three types of guessing the estimated class labels are considered. Recall that these estimates do not necessarily equal zero or one, but may be decimal values as well.

The first method to guess the estimated class labels is by drawing from from the $Bernoulli(p)$ distribution for every observation. This results in every $\hat{t}_i^{(0)}$ being equal to zero or one. The parameter p can be set to 0.5 to simulate purely random guessing, resulting in approximately evenly many inliers and outliers. The value of p can be altered to reflect beliefs in the contamination level, as decreasing p results in less estimated class labels of one and, therefore, less observations marked as outlier. This way of initialising the estimated class labels is referred to as *p-random initialisation*.
As increasing anomaly scores reflect stronger belief in observations being anomalies, the observation with the highest anomaly score is the most suspicious. Based on this insight, the estimated class labels are initialised linearly. Hence, the anomaly scores are first sorted and the observation with the i-th anomaly score gets estimated class label $\hat{t}_i^{(0)} = \frac{i-1}{n-1}$. This *linear initialisation* ensures that the class label of the observation with the lowest anomaly score is estimated as zero, whereas the observation with the highest anomaly score has an estimated class label of one. Note that other monotonically increasing initialisation are possible, but in this thesis the most simple one is chosen for no particular reason.
The third method of guessing the class labels is by running a clustering algorithm on the data first. Setting the number of clusters equal to two results in every observation getting a label of zero of one. These labels can then be used to compute the initial guess of the parameters of the mixture. The benchmark clustering algorithm used for this purpose is k-means (with $k = 2$), which is the reason that this method is called 2-*means initialisation*. Note that k-means also requires initial guesses, for which random guessing is used in general.

Figure 4 shows the results of the different initialisation methods. As mentioned, decreasing the value of p in p-random initialisation results in fewer observations having a guessed estimated class label of one. The 2-means results in the first 82 observations being clustered as inliers and, therefore, are assigned a guessed estimated class label of zero.

2.2.5 Computing the Threshold

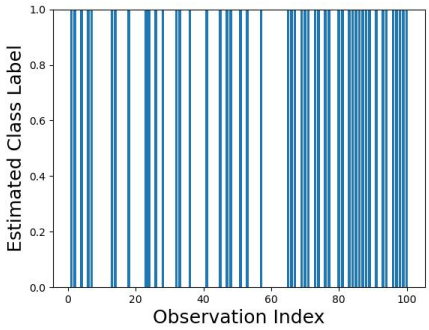
When the parameters $\omega, \boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ are estimated, all the densities described in section 2.1 are completely determined. To compute a threshold that can be used to separate the inliers from the outliers, the density ratio $R(s) = \frac{f_1(s|\boldsymbol{\theta}_1)}{f_0(s|\boldsymbol{\theta}_1)}$ plays an important role. The idea is to equate $R(s)$ to a value γ such that the threshold is defined as the solution of $R(s) = \gamma$. Three values of γ are proposed, leading to a threshold based on *likelihood, posterior density* or *cost weights*.

The idea behind the threshold of **equal likelihood** - or likelihood-threshold in short - is that, given the parameter estimates, the threshold equals the value of s that is equally likely under $T = 0$ as under $T = 1$. In other words, the threshold τ is value of s that solves

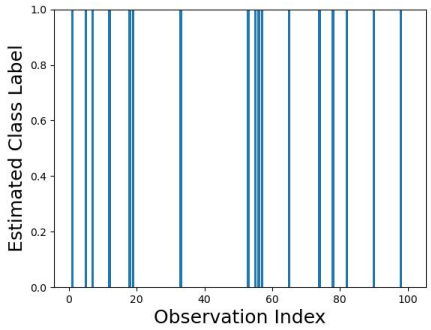
$$\begin{aligned} f_{S|T}(S = s \mid T = 0) = f_{S|T}(S = s \mid T = 1) \quad &\Rightarrow \quad f_0(s \mid \boldsymbol{\theta}_0) = f_1(s \mid \boldsymbol{\theta}_1) \\ &\Rightarrow \quad \frac{f_1(s \mid \boldsymbol{\theta}_1)}{f_0(s \mid \boldsymbol{\theta}_0)} = 1. \end{aligned}$$

Hence, this threshold, denoted as $\tau_{likelihood}$, can be defined as

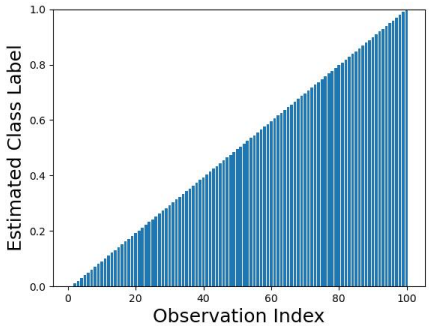
$$\tau_{likelihood} = s^* \text{ such that } R(s^*) = 1. \tag{11}$$



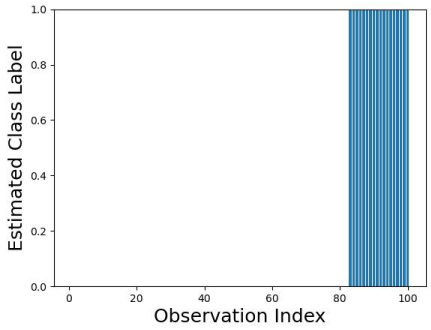
(a) 0.5-random initialisation.



(b) 0.2-random initialisation.



(c) Linear initialisation.



(d) 2-means initialisation.

Figure 4: Different initialisation methods for the estimated class labels. The 100 observations are first ordered by their anomaly score.

The **posterior probability**-threshold or simply posterior-threshold compares the posterior probabilities of the class labels. Given the parameter estimates, the threshold is computed as the value of s for which the posteriors are equal, i.e.

$$\begin{aligned} P(T=0 \mid S=s) = P(T=1 \mid S=s) \quad &\Rightarrow \quad \frac{(1-\omega) f_0(s \mid \boldsymbol{\theta}_0)}{(1-\omega) f_0(s \mid \boldsymbol{\theta}_0) + \omega f_1(s \mid \boldsymbol{\theta}_1)} = \frac{\omega f_1(s \mid \boldsymbol{\theta}_1)}{(1-\omega) f_0(s \mid \boldsymbol{\theta}_0) + \omega f_1(s \mid \boldsymbol{\theta}_1)} \\ &\Rightarrow \quad \frac{f_1(s \mid \boldsymbol{\theta}_1)}{f_0(s \mid \boldsymbol{\theta}_0)} = \frac{1-\omega}{\omega}. \end{aligned}$$

This threshold is denoted as $\tau_{posterior}$ and is defined as

$$\tau_{posterior} = s^* \text{ such that } R(s^*) = \frac{1-\omega}{\omega}. \tag{12}$$

Threshold $\tau_{likelihood}$ is a special case of $\tau_{posterior}$ for $\omega = 0.5$.

The inclusion of **cost weights** is proposed by [8] and essentially follows from a Bayesian risk model. It requires the user to specify a matrix of misclassification costs

$$C = \begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix},$$

where c_{ij} indicates the cost of assigning class i to an observation that belongs to class j. With this cost

matrix, the threshold can be computed as the value of s that solves

$$(c_{10} - c_{00})P(T = 0 \mid S = s) = (c_{01} - c_{11})P(T = 1 \mid S = s)$$
$$\Rightarrow \frac{f_1(s \mid \boldsymbol{\theta}_1)}{f_0(s \mid \boldsymbol{\theta}_0)} = \frac{c_{10} - c_{00}}{c_{01} - c_{11}} \cdot \frac{1 - \omega}{\omega}.$$

The threshold including the cost matrix is denoted as $\tau_{weigthed}$ and can be defined as

$$\tau_{weighted} = s^* \text{ such that } R(s^*) = \frac{c_{10} - c_{00}}{c_{01} - c_{11}} \cdot \frac{1 - \omega}{\omega}. \tag{13}$$

If the zero-one loss is used, i.e.

$$c_{ij} = \begin{cases} 0 & \text{if } i = j; \\ 1 & \text{if } i \neq j; \end{cases}$$

then $\tau_{weighted} = \tau_{posterior}$.
The usage of a cost matrix becomes important for different applications of the thresholding techniques. In section 1.3, the risk of falsely accusing a customer to be fraudulent was mentioned. This risk can be modelled by defining the cost matrix. For example, if we want to avoid false accusations, we can increase the value of c_{10}. This results in the threshold to shift to a higher value, marking less observations as anomalies and therefore decreases the risk of false accusations.

2.2.6 Short Summary

In this section, we have discussed two methods of estimating parameters $\omega, \boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$. These methods are called direct maximum likelihood estimation and EM estimation. When the parameters are estimated, the most generic way to compute the threshold is $\tau_{weighted}$, which yields $\tau_{posterior}$ and $\tau_{likelihood}$ for particular values of $c_{ij}; i, j = 0, 1$ and ω.

2.3 Distributions with Bounded Support

In appendix A, the update rules of some distributions are derived. However, there are some distributions that do not work well with the default method of optimising the expected complete log-likelihood, in this section also referred to as the log-likelihood. The difficulties with this default method are due to some distributions having a bounded support. In this subsection, we discuss the uniform and Pareto distribution in more detail and propose a workaround.
Note that, similar to the work in Appendix A, in deriving the update rules we assume, without loss of generality, that the distributions model the outliers. Indeed, if the distribution models the inliers, the label estimates $1 - \hat{t}_i^{(k)}$ instead of $\hat{t}_i^{(k)}$ should be used. For more information, see Appendix A.

2.3.1 The Uniform Distribution

The uniform distribution is often defined in terms of a lower bound a and an upper bound b. Denoted as $Uniform(a, b)$, the density of an uniform distribution is given by

$$f(x \mid a, b) = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a, b]; \\ 0 & \text{otherwise.} \end{cases}$$

The distribution could also be parametrised by the mean μ and the standard deviation σ. The density of this distribution $Uniform(\mu, \sigma)$ is given by

$$f(x \mid \mu, \sigma) = \begin{cases} \frac{1}{2\sigma\sqrt{3}} & \text{if } \mid x - \mu \mid \leq \sigma\sqrt{3} \\ 0 & \text{otherwise.} \end{cases}$$

Switching between the parametrisation can be done by the relations

$$a = \mu - \sigma\sqrt{3}; \qquad b = \mu + \sigma\sqrt{3}. \tag{14}$$

During the M-step of the EM-algorithm, the parameters of the uniform distribution are estimated by maximising the term

$$Q(\Theta, \hat{\Theta}^{(k-1)}) = \sum_{i=1}^{n} \hat{t}_i^{(k-1)} \log(f(s_i \mid \boldsymbol{\theta}_1)), \tag{15}$$

with either $\boldsymbol{\theta}_1 = (a, b)$ or $\boldsymbol{\theta}_1 = (\mu, \sigma)$, depending on the applied parametrisation. Note that the expected complete log-likelihood defined in (10) evaluates to (15), as all other terms do not include any parameters of the uniform distribution and are, therefore, left out of the maximisation.

Optimisation of a and b. When using the parametrisation with a and b, we encounter some difficulties during the M-step of the EM-algorithm. Using $s_{(i)}$ as the order statistics, i.e. $s_{(1)}$ is the smallest and $s_{(n)}$ is the largest anomaly score, we note the following.

1) Whenever we pick $a > s_{(1)}$ or $b < s_{(n)}$, there will be an observation (having the smallest respectively largest anomaly score) with density $f(s \mid a, b) = 0$. Since the logarithm of 0 equals $-\infty$, this implies that the value of (15) yields $-\infty$ as well. For these cases, the log-likelihood will clearly not be maximised. Hence, the MLEs of a and b require at least $a \leq s_{(1)}$ and $b \geq s_{(n)}$.

2) When constraint 1) above is satisfied, the log-likelihood equals

$$Q(\Theta, \hat{\Theta}^{(k-1)}) = -\log(b - a) \sum_{i=1}^{n} t_i.$$

The log-likelihood is increasing in a and decreasing in b. Hence, to obtain the maximum (log)likelihood, we would like to pick a as large as possible and b as small as possible.

Combining the two points above, we conclude that the MLEs of a and b are

$$\hat{a} = s_{(1)} = \min(s_1, ..., s_n); \qquad \hat{b} = s_{(n)} = \max(s_1, ..., s_n)$$

Although this may seem like a nice result, it essentially means that the estimates of a and b are equal throughout the EM-algorithm. The estimates are not updated and the distribution will converge after one iteration. This is not really a problem, but the problem is that the density resulting from EM will span the entire data set, as shown in Figure 5a. In other words, the density for observations with a low anomaly score is equal to that of observations with a high anomaly score. When the uniform distribution is used to model the outliers, this implies that an anomaly having a low score is equally likely to an anomaly having a high score. Hence, the distribution for the outliers does not distinguish outliers from inliers, which is not very useful.

Optimisation of μ and σ. If we use $Uniform(\mu, \sigma)$ as parametrisation and let $s_{(i)}$ be the order statistics again, we note the following.

1) If we pick μ and σ such that $\sigma\sqrt{3} < \mid s_{(1)} - \mu \mid$, then $f(s_{(1)} \mid \mu, \sigma) = 0$. Since $\log(0) = -\infty$, the value of (15) will yield $-\infty$ as well. Similarly, if $\sigma\sqrt{3} < \mid s_{(n)} - \mu \mid$, the log-likelihood will equal $-\infty$ again. In these cases, we can be sure that the log-likelihood is not maximised. Hence, to maximise the log-likelihood, we should at least have

$$\sigma \geq \frac{1}{\sqrt{3}} \max\{ \mid s_{(1)} - \mu \mid , \mid s_{(n)} - \mu \mid \}$$

2) When the constraint in 1) above is satisfied, the expected complete log-likelihood evaluates to

$$Q(\Theta, \hat{\Theta}^{(k-1)}) = -\log(2\sigma\sqrt{3}) \sum_{i=1}^{n} \hat{t}_i^{(k-1)}.$$

This is a decreasing function for σ. Hence, to optimise it, we would like to pick σ as small as possible in order to maximise the loglikelihood.

From the two remarks above, we conclude that we should estimate σ in the M-step as

$$\hat{\sigma}^{(k)} = \frac{1}{\sqrt{3}} \max\{ \mid x_{(1)} - \hat{\mu}^{(k)} \mid , \mid x_{(n)} - \hat{\mu}^{(k)} \mid \},$$

with μ estimated as the weighted mean, i.e.

$$\hat{\mu}^{(k)} = \frac{\sum_{i=1}^{n} \hat{t}_i^{(k-1)} s_i}{\sum_{i=1}^{n} \hat{t}_i^{(k-1)}}. \tag{16}$$

With the estimates $\hat{\mu}^{(k)}$ and $\hat{\sigma}^{(k)}$, one can infer the values of a and b by (14).

As opposed to the results of $Uniform(a, b)$, the parameters of the uniform distribution will be updated throughout the EM-algorithm. It seems like this solved the problem, but the outputted result is however even worse. Indeed, the fitted uniform distribution now not only spans the complete data set, but also extends to either side. See Figure 5b. Hence, it is suffering from the same problem described in the previous paragraph: according to the fitted distribution, anomalies having low scores are evenly likely as anomalies having high scores.

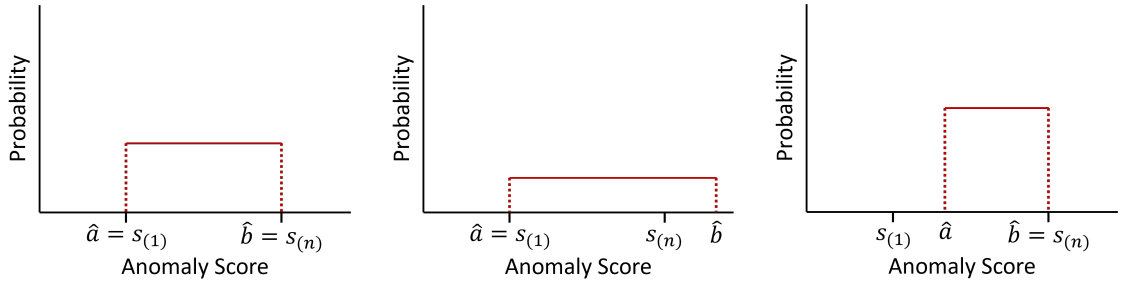
Fixing One Bound As shown, the default maximisation of the expected complete loglikelihood has two backfalls. The parametrisation with a and b suffers from 1) the estimated distribution not getting updated after one iteration of the EM-algorithm and 2) the eventual distribution spanning the entire data set. Parametrising the uniform distribution by its mean and standard deviation solves the first problem, but worsens the second.
To solve both problems, it is proposed to fix one bound of the uniform distribution. Hence, if the uniform distribution is used to model the scores of the outliers, the upper bound b gets fixed to the largest anomaly score $s_{(n)}$, i.e. $\hat{b} = s_{(n)}$. When the uniform distribution is used to model the inliers, the lower bound is fixed to the smallest anomaly score, i.e. $\hat{a} = s_{(1)}$. The non-fixed parameter is estimated by using the weighted mean again, i.e. if the upper bound b is fixed, the estimate for the lower bound a is

$$\begin{aligned} \hat{a}^{(k)} &= \hat{\mu}^{(k)} - (\hat{b} - \hat{\mu}^{(k-1)}) \\ &= 2\hat{\mu}^{(k)} - \hat{b} \\ &= 2\hat{\mu}^{(k)} - s_{(n)} \end{aligned}$$

Similarly, if the lower bound is fixed, estimate the upper bound as

$$\hat{b}^{(k)} = 2 \cdot \hat{\mu}^{(k-1)} - \hat{a} = 2 \cdot \hat{\mu}^{(k)} - s_{(1)},$$

with $\hat{\mu}^{(k)}$ as in (16). This results in the distribution getting narrower every iteration, moving towards the fixed bound. The result of this maximisation approach is shown in Figure 5c.



(a) $Uniform(a, b)$. (b) $Uniform(\mu, \sigma)$. (c) Fixed b.

Figure 5: Output of EM-algorithm for three different maximisation approaches of the Uniform distribution, when used to model the outliers.

2.3.2 The Pareto Distribution

The density of the $Pareto(x_m, \alpha)$ distribution is defined as

$$f(x \mid x_m, \alpha) = \begin{cases} \frac{\alpha x_m^\alpha}{x^{\alpha+1}} & \text{if } x \geq x_m; \\ 0 & \text{otherwise.} \end{cases}$$

For the default optimisation, we consider the terms of the expected complete log-likelihood in (10) that contain x_m or α. This yields

$$Q(\hat{\Theta}, \hat{\Theta}^{(k-1)}) = \sum_{i=1}^{n} \hat{t}_i^{(k-1)} \log(f(s_i \mid x_m, \alpha)).$$

Similar to the Uniform distribution, we note that $f(s_i \mid x_m, \alpha) = 0$ if $s_i < x_m$, resulting in the expected complete log-likelihood to equal $-\infty$. Therefore, the value of x_m should satisfy $x_m \leq s_{(1)}$, the minimal anomaly score.
When the condition $x_m \leq s_{(1)}$ is satisfied, the expected complete log-likelihood can be written as

$$\begin{aligned} Q(\hat{\Theta}, \hat{\Theta}^{(k-1)}) &= \sum_{i=1}^{n} \hat{t}_i^{(k-1)} \log\left(\frac{\alpha x_m^\alpha}{x^{\alpha+1}}\right); \\ &= \sum_{i=1}^{n} \hat{t}_i^{(k-1)} \log(\alpha) + \alpha \hat{t}_i^{(k-1)} \log(x_m) - (\alpha+1)\hat{t}_i^{(k-1)} \log(s_i), \\ &= \log(\alpha) \sum_{i=1}^{n} \hat{t}_i^{(k-1)} + \alpha \log(x_m) \sum_{i=1}^{n} \hat{t}_i^{(k-1)} - (\alpha+1) \sum_{i=1}^{n} \hat{t}_i^{(k-1)} \log(s_i). \end{aligned}$$

This function is increasing in x_m, which is the reason that the expected complete log-likelihood is maximised for x_m to be as high as possible. Given the constraint $x_m \leq s_{(1)}$, this results in the estimate

$$\hat{x}_m^{(k)} = s_{(1)}.$$

We are left with estimating the value of α. To find the value of α that maximises the expected complete log-likelihood, we equate the derivative w.r.t. α to zero. For the derivative we have

$$\frac{\partial}{\partial \alpha} Q = \frac{1}{\alpha} \sum_{i=1}^{n} \hat{t}_i^{(k-1)} + \log(x_m) \sum_{i=1}^{n} \hat{t}_i^{(k-1)} - \sum_{i=1}^{n} \hat{t}_i^{(k-1)} \log(s_i).$$

Equating this derivative to zero gives

$$
\begin{aligned}
0 &= \frac{1}{\alpha}\sum_{i=1}^{n}\hat{t}_i^{(k-1)} + \log(x_m)\sum_{i=1}^{n}\hat{t}_i^{(k-1)} - \sum_{i=1}^{n}\hat{t}_i^{(k-1)}\log(s_i),\\
0 &= \frac{1}{\alpha} + \log(x_m) - \overline{\log(s)}^{(k-1)},\\
\alpha &= \left(\overline{\log(s)}^{(k-1)} - \log(x_m)\right)^{-1},
\end{aligned}
$$

with $\overline{\log(s)}^{(k-1)}$ the weighted mean of the logarithm of the anomaly scores. Hence, the default optimisation results in updating rules

$$
\begin{aligned}
\hat{x}_m^{(k)} &= s_{(1)},\\
\hat{\alpha}^{(k)} &= \left(\overline{\log(s)}^{(k-1)} - \log(x_m)\right)^{-1}.
\end{aligned}
$$

It may be clear that the estimate of x_m is not updated throughout the EM-algorithm: it is set equal to the lowest anomaly score. Therefore, the fitted Pareto distribution spans the entire data set again. That is no problem if the Pareto distribution is used to model the inliers, but it is not desired if the Pareto distribution models the outliers. In case of the latter, it means that observations with a low anomaly score are much more likely than observations with a high anomaly score, which is exactly the opposite of what we are trying to establish.

It is very common in financial data to model transactions with a mixture. The body of the data is modelled by a log-normal distribution and the Pareto distribution is often chosen to model the tail. The goal of fitting this mixture is to find the value of x_m, i.e. the value at which the tail starts. This reflects the problem with the results of the default maximisation: no matter which class is modelled by the Pareto distribution, x_m is always estimated as the lowest anomaly score. When the Pareto distribution is used to model the outliers, or the tail of the anomaly scores, the estimate of x_m is certainly inaccurate.
In [3], a better method to estimate x_m is proposed. The proposed procedure is to iteratively fix the value of x_m to decreasing observation values, i.e. $x_m = s_{(n)}, x_m = s_{(n-1)}, ..., x_m = s_{(1)}$. For every fixed x_m, the other parameters in the mixture model are estimated and the log-likelihood of the mixture is computed. Eventually, the outputted parameter estimates are the ones that returned the maximal log-likelihood.
A remark should be made here that with looping over n data points, the execution time of the algorithm increases with an order n. As the data sets increase in size, the EM algorithm can take minutes to find an optimal solution. An early stopping rule is added to the algorithm: if the decrease in log-likelihood is larger than a specified threshold, stop the loop and return the best solution so far. The pseudo-code of this approach is shown in Algorithm 2.

Algorithm 2: Estimation with Pareto Distribution.

Input: Set of anomaly scores $\mathcal{S}$, convergence threshold ε, intial guesses $\hat{\Theta}^{(0)}$, stopping threshold η.
Output: Parameter estimates $\hat{\Theta}$.

Construct $\mathcal{S}^-$, the anomaly scores sorted in descending order.
For x_m in $\mathcal{S}^-$:
    Estimate other parameters of the mixture by the DML- or EM-algorithm.
    Compute the log-likelihood of the data.
    If the decrease in log-likelihood compared with the best so far is larger than $\eta \cdot 100\%$, break.
Return the parameter estimates with the highest log-likelihood.

2.4 Illustration of the EM-algorithm

In this section, the mechanism of the EM-algorithm is illustrated by fitting a mixture of an exponential and a normal distribution. Hence, the parameters to estimate are $\Theta = (\omega, \lambda, \mu, \sigma^2)$ and we assume that

- the inliers follow an exponential distribution with rate λ, i.e. $f_0(s \mid \lambda) = \lambda e^{-\lambda s}$;
- the outliers follow a normal distribution with mean μ and variance σ^2, i.e.

$$f_1(s \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2\sigma^2}(s-\mu)^2);$$

- the prior probabilities of the class label are given by $P(T=1) = \omega$ and $P(T=0) = 1-\omega$.

From the results in Appendix A, the update rules for the parameters in the M-step are given by

$$\begin{aligned}
\hat{\omega}^{(k+1)} &= \frac{1}{n}\sum_{i=1}^{n} \hat{t}_i^{(k)} \\
\hat{\lambda}^{(k+1)} &= \frac{\sum_{i=1}^{n}(1-\hat{t}_i^{(k)})}{\sum_{i=1}^{n}(1-\hat{t}_i^{(k)})s_i} \\
\hat{\mu}^{(k+1)} &= \frac{\sum_{i=1}^{n} \hat{t}_i^{(k)} s_i}{\sum_{i=1}^{n} \hat{t}_i^{(k)}} \\
\hat{\sigma}^{(k+1)} &= \Big(\frac{\sum_{i=1}^{n} \hat{t}_i^{(k)}(s_i - \hat{\mu}^{(k+1)})^2}{\sum_{i=1}^{n} \hat{t}_i^{(k)}}\Big)^{\frac{1}{2}}.
\end{aligned}$$

Note that the exponential distribution is used to model the inliers, so $\hat{t}_i^{(k)}$ is replaced by $(1-\hat{t}_i^{(k)})$.
To illustrate the EM-algorithm, a data set is drawn from a mixture with parameters

$$\begin{aligned}
\omega_{true} &= 0.2; \qquad & \lambda_{true} &= 0.7; \\
\mu_{true} &= 15; \qquad & \sigma_{true} &= 3,
\end{aligned}$$

and we will investigate how the EM-algorithm approaches these true parameters. The data set is depicted in Figure 6 and is drawn by setting a random seed of 302, sampling 160 times from the exponential distribution (the inliers) and 40 times from the normal distribution (the outliers).
To fit the mixture model, the convergence threshold has been set to $\varepsilon = 10^{-5}$. The convergence results are shown in Figure 7. After 34 iterations, all parameters have converged. The output of the EM-algorithm is quite close to the true values:

$$\begin{aligned}
\hat{\omega}_{EM} &= 0.1997; \qquad & \hat{\lambda}_{EM} &= 0.7589; \\
\hat{\mu}_{EM} &= 14.6119; \qquad & \hat{\sigma}_{EM} &= 3.1673.
\end{aligned}$$

The resulting threshold $\tau_{posterior}$ equals 7.5091, selecting 38 observations as anomalies.

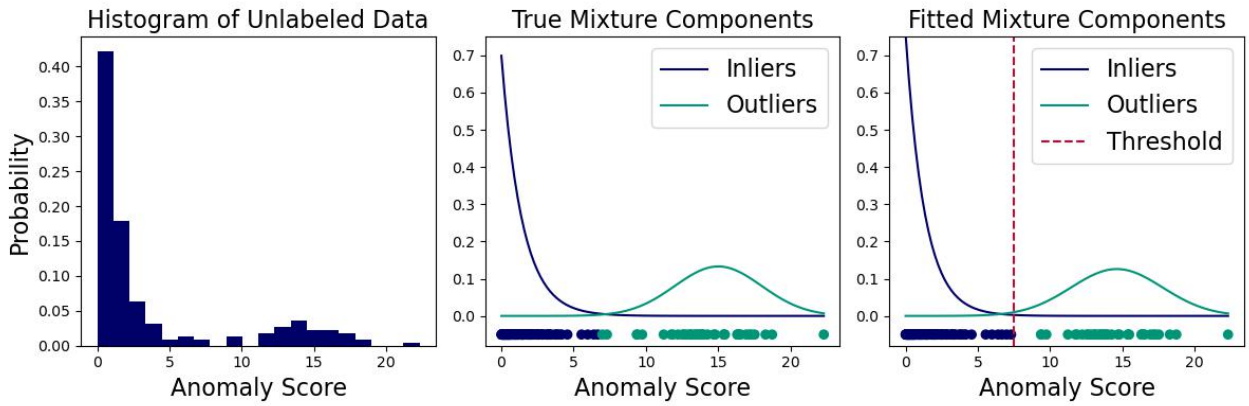


Figure 6: Description of the generated data. On the left is a histogram of the unlabeled data, the input of the EM-algorithm. In the middle, the distribution that is used to generate the data set is plotted. On the right the result of the fit is shown.

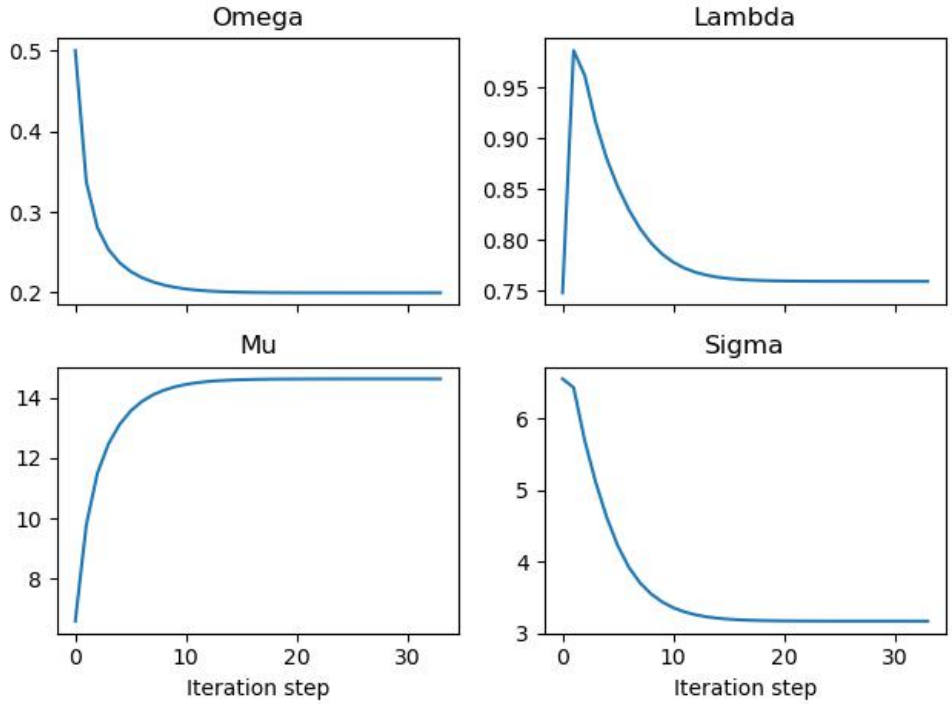


Figure 7: Convergence of the parameters of the mixture. After iteration 34 all parameters have converged.

3 Numerical Results

To test the performance of the mixture models described in Section 2, three numerical experiments are executed. For each experiment, the setup will be discussed in three parts - which data set(s) is/are used, what to compare, which performance measure to use - after which the numerical results are discussed. Lastly, a short conclusion on the numerical results is given.
The experiments are used to answer different subquestions of the research aim, but also worsen in the lack of information. In the first experiment, the true labels as well as the true parameters are known. The second experiment only includes information on the labels. The last experiment starts from zero: there is no information on the parameter values or the labels whatsoever.

3.1 Comparison of EM and DML Algorithms

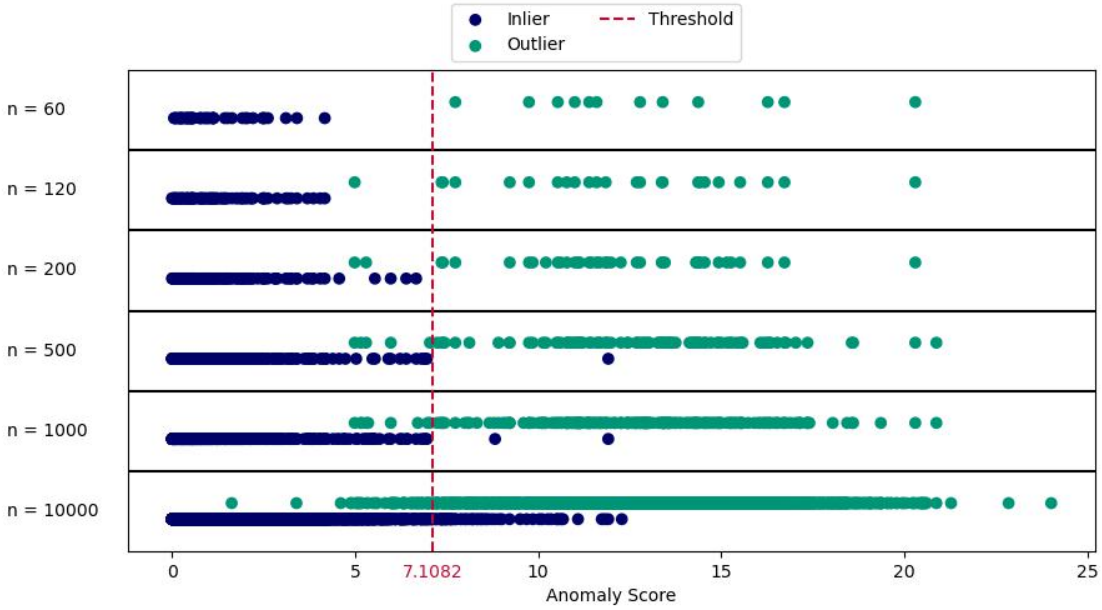
In the first experiment, we compare the performance of the EM and DML algorithms and formulate conditions on which algorithm to use in a particular experiment. The comparison is made based on three aspects. Firstly, the sensitivity of the two estimation algorithms to the input settings - the number of observations and three convergence settings - is considered. The settings and their default value are listed in Table 1. In this experiment, we vary the values of one of the settings and keep the others equal to their default. We study the effect of the settings to the output of the algorithm. Note that the estimation is cut off if it has run for 1000 iterations and no convergence has occurred.
Secondly, we compare the EM and DML algorithms in terms of correctness, i.e. whether or not the outputted results are close to the true parameter values. For that purpose, we generate data sets from a known mixture and aim to retrieve that mixture.
Lastly, a short experiment is executed to gain insight on the execution times for both algorithms.

Setting	Default Value	Values
Number of restarts	10	$\{1, 5, 10, 20, 50, 100\}$
Initialisation	0.5-random	linear and p-random with $p \in \{0.8, 0.5, 0.1, 0.05\}$
Tolerance ε	10^{-5}	$\{10^{-3}, 10^{-5}, 10^{-7}, 10^{-10}\}$

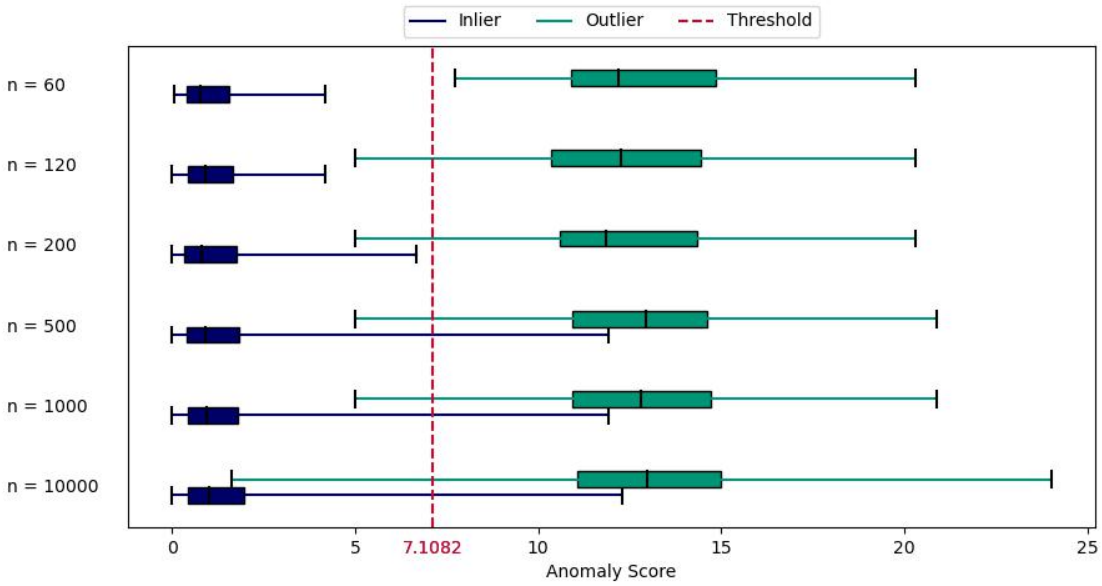
Table 1: Default settings and considered values for DML and EM algorithms.

Experimental Setup

The data sets used in this experiment are drawn from an exponential-normal mixture, varying in the number of observations n. The parameters of the mixture are fixed at $\omega = 0.2, \lambda = 0.7, \mu = 13$ and $\sigma = 3$. The value of n is taken from the set $\{60, 120, 200, 500, 1000, 10000\}$. The data set of $n = 60$, for instance, is generated by drawing 48 times from the exponential distribution and 12 times from the normal distribution, assuring 20% outliers (and thus $\omega = 0.2$) For every data set, the random seed is set to 302.
Increasing the value of n has two considerable effects. Firstly, the number of observations to process increases, which play an important role in the execution time of the algorithm. Secondly, the anomaly scores of inliers and outliers are getting mixed, although the regions with high density are kept in location. This can be seen from the plotted data sets in Figure 8a and the boxplots in Figure 8. For the case $n = 60$, the inliers and outliers are clearly separated. However, as the number of observations increases, the anomaly scores of the inliers and outliers overlap.
In general, a machine learning model yields better performance if the amount of data increases (and the risk of overfitting is ignored). On the contrary, we might argue that as the observations are mixed, the mixture might get "confused". In that case, the threshold that perfectly separates inliers and outliers in case of $n = 60$ could not be retrieved for the case $n = 10000$.



(a) Data sets used to compare DML and EM algorithms.



(b) Boxplot of the data sets used to compare DML and EM algorithms.

Figure 8: Description of the data sets used to compare the performance of the DML and EM algorithms.

In this experiment, the performances of the EM and DML algorithms is compared. Central to this performance is the sensitivity of the output of both algorithms to the value of the settings described in Table 1. For threshold computation, the default threshold based on equal posterior probability is used.
The execution time of the estimation algorithms is not only tested by fitting an exponential-normal mixture on the data sets depicted in Figure 8a. In addition, a cross-mixture comparison is made by also fitting a normal-Pareto and gamma-uniform mixture on the data sets of $n = 60, 200, 500, 1000$, with default settings. The idea behind using these three mixtures is that the EM algorithm takes a different approach to estimate the parameters. As shown in Appendix A, the exponential and normal distributions have closed-form update rules. The Pareto distribution also has a closed-form update rule for α, but to estimate x_m a loop over all anomaly scores is executed. Lastly, the Gamma distribution has no closed-form update rule and a root finding algorithm is executed in every M-step. Therefore, we expect the execution time of the EM algorithm to increase for fitting mixtures with Pareto or Gamma distributions involved, whereas the DML algorithm should more or less stay equal. Note that we are only interested in the execution time, so no correctness is computed.

The sensitivity and correctness of the EM and DML algorithms is measured in terms of the log-error of the estimated threshold compared to the "true threshold". The true threshold is the threshold computed by using the true parameter values in the mixtures. Note that the value of this true threshold does not depend on the size of the data set, but solely on the parameter estimates. As can be seen in Figure 8a, the true threshold (rounded to four decimals) equals 7.1082.
The correctness of the output of the estimation algorithms is directly related to the log-error: the lower the log-error, the better is the output. For the sensitivity, the variance of the log-error is of interest. If the log-errors have low variance, the algorithms find equal results and the setting is not affecting the algorithm strongly. However, if the outputs do vary, it is concluded that the algorithm is sensitive to the setting.
In addition to the log-error, the sensitivity is also measured in terms of execution time.

Results

The three sensitivity plots - one for each setting `n_starts`, `tolerance` and `initialisation` - looked very much alike, which is the reason to only show the plot for `n_starts` in Figure 9.
As far the correctness goes, we see that the mixture performance does not worsen as the amount of data is increased. Instead, the algorithms get closer to the true threshold. This follows from the fact that the log-error of the computed threshold is lower for higher amounts of data.
From Figure 9, it can be concluded that the DML algorithm is a bit more sensitive to the settings than the EM algorithm. This can be seen from the fact that, for the different values of `n_starts`, the log-error of the thresholds for the EM algorithm are equal. However, the log-error of the DML outputs does vary, most apparently for just one restart.
Regarding the execution time as result of varying a setting, the results were quite different. For setting `n_starts`, the results are shown in Figure 10. A logical result of having more restarts and a larger data set is that the execution time increases, but the DML algorithm seems to be more sensitive to this setting compared to EM, as the dashed line has a steeper increase than the solid lines.

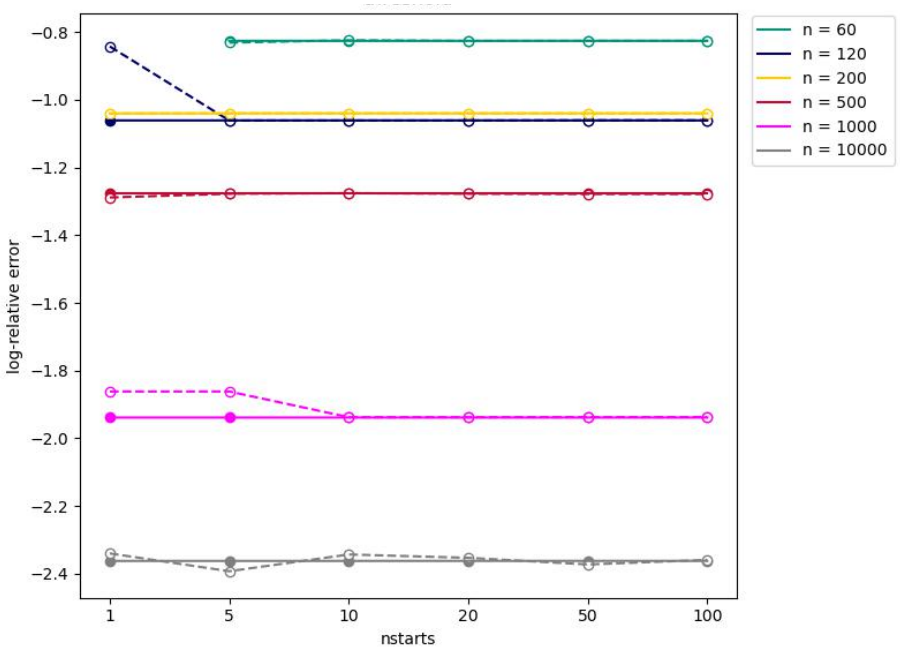


Figure 9: Logarithm of the relative error for different values of `n_starts`. Solid lines are the results of EM, dashed lines are the results of DML.

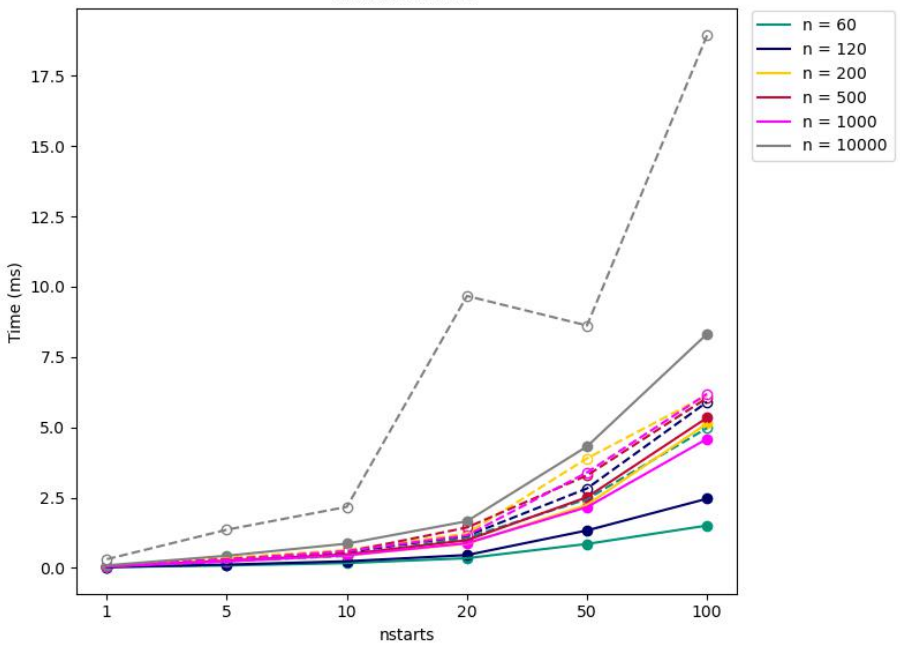


Figure 10: Execution times for different values of `n_starts`. Solid lines are the results of EM, dashed lines are the results of DML.

The cross-mixture test for execution times resulted in Table 2 and Table 3. A general trend of increasing execution times for larger amounts of data can be seen here. In addition, a significant change in the execution times of the EM algorithm is recorded for the different mixtures. As expected, the EM algorithm needs more time for all the calculations involved in case of mixtures including Pareto or gamma distributions. This is in contrast with the DML algorithm, for which the execution times stay more or less equal for the different mixtures. Furthermore, the DML algorithm is faster in fitting mixtures with the Pareto or gamma distributions, as it does not have to do a loop over all observation nor executing a numerical solver in every M-step.

Mixture	n = 60	n = 200	n = 500	n = 1000
Exponential-Normal	0.151	0.355	0.412	0.464
Normal-Pareto	0.599	1.213	1.673	2.380
Gamma-Uniform	0.201	0.482	0.565	0.587

Table 2: Execution time in seconds of the EM algorithm.

Mixture	n = 60	n = 200	n = 500	n = 1000
Exponential-Normal	0.322	0.413	0.460	0.502
Normal-Pareto	0.328	0.419	0.403	0.470
Gamma-Uniform	0.324	0.320	0.356	0.400

Table 3: Execution time in seconds of the DML algorithm.

Conclusion

The conclusion to draw based on the performed experiments is that the settings do not matter that much and the default values from Table 1 will do just fine. The variability of the DML algorithm is a little bit higher compared to the EM algorithm. This is in line with the possibility of DML to "overshoot" the maximum - see Section 2.2.3 - as this overshooting may result in different parameter estimates.
Considering the execution times of the algorithms, it seems that the DML algorithm is more sensitive to the number of restarts. For distributions that do not have closed-form update rules - such as the gamma distribution - or looping over the observations in the data set is needed, e.g. the Pareto distribution, the DML algorithm is faster than the EM algorithm. On the contrary, if the EM algorithm is used to estimate parameters of distributions with closed-form update rules, it is faster than the DML algorithm.
Even though there are minor differences described above, both algorithms are not much different from each other.

3.2 Performance on Real-World Data Sets

In the second experiment, we apply the mixtures to real-world (labeled) data sets from the Outlier Detection DataSets (ODDS) database. The results of the mixtures on these data sets will be compared to each other and to the results of the PyThresh module. To make fair comparison with the results of PyThresh possible, the experimental setup will be similar.

Experimental Setup

From the ODDS database, 10 data sets were used in this experiment. Included are some of the data sets used in the benchmarking of PyThresh. The data sets vary in size and number of outliers, see Table 4.
The 10 data sets are put through 7 different ADAs, similar to the PyThresh benchmark. This means that all ADAs are fitted with their default value, e.g. kNN is fitted with $k = 5$ and LOF with $k = 10$. For all default values of the ADAs, see the documentation of the PyOD module. In addition to the 7 ADAs, an 8-th set of anomaly scores is constructed by taking the mean of the output of the ADAs. This set is called the *ensemble* ADA. After this procedure, we end up with 80 data sets of anomaly scores. The used ADAs are depicted in Figure 13. Before fitting the mixtures on the sets of anomaly scores, a minmax-conversion is applied to transform the anomaly scores into the range [0,1]. That way, the highest anomaly score is mapped to 1 and the lowest anomaly score is mapped to 0.
One remark should be made about the sets of anomaly scores: if the ADA performs poorly, we can not expect the mixture to find a good threshold. For example, in Figure 11 the output of kNN on data set `ionosphere` is plotted. Here, the anomaly scores of the inliers and outliers are mixed, but kNN does pick up on the general trend: most outliers are assigned a high anomaly score. We expect that computing thresholds based on the mixtures fitted to this data results in thresholds of high performance. On the contrary, the output of LOF on the data set `annthyroid` in Figure 12. The ADA does not pick up on the patterns in the data, assigning inliers and outliers more or less equal anomaly scores. In fact, the highest anomaly score are assigned to observations that are actually inliers. Due to the low performance of the ADA, we can not expect the mixture to find a good threshold in this case.

Data set	No. of observations	No. of dimensions	No. of outliers	Outlier Proportion (%)
Annthyroid	7200	6	534	7.42
Arrhythmia	452	274	66	15.00
Breastw	683	9	3511	7.00
Cardio	1831	21	176	9.60
Glass	214	9	9	4.20
Ionosphere	351	33	126	36.00
Lympho	148	18	6	4.10
Musk	3062	166	97	3.20
Pima	768	8	268	35.00
WBC	278	30	21	5.60

Table 4: Description of data sets used in the experimental study.

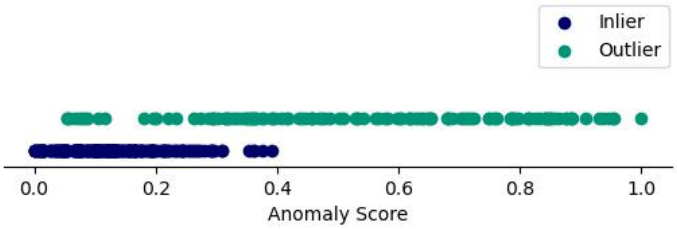


Figure 11: Output of kNN for the data set `ionosphere`.

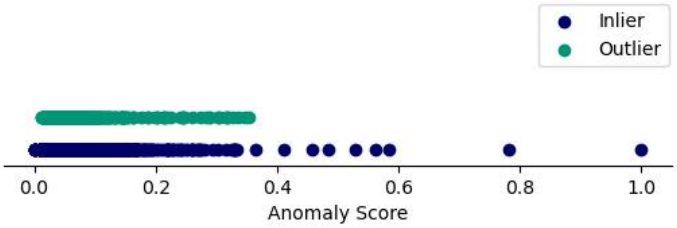


Figure 12: Output of LOF for the data set `annthyroid`.

In this experiment, different mixtures were compared. In total, eight distributions were combined to construct 42 mixtures, see Figure 13 and Appendix B for an extensive list of the used mixtures. Note that the half-normal distribution is only used to model the inliers, whereas the uniform and Pareto distributions are only applied to model the outliers. Important to remark here is that, due to time constraints, the mixtures including a gamma or beta distribution are only fitted by DML.
Besides comparing the mixtures to each other, we also compare the mixtures to the state-of-the-art *Top-N* thresholder. Since the data sets are labelled, the number of outliers is known and the Top-N thresholder is feasible. To illustrate how the Top-N thresholder works, see Figure 14. This figure shows the anomaly scores outputted by the iForest detection algorithm, applied on the data set `glass`. As this data set contains 9 outliers, the observations with the 9 highest anomaly scores are marked as outliers. In other words, the threshold will be set to the 9th highest anomaly score, in this case 0.68.
Lastly, the mixtures will be compared to the thresholding techniques discussed in the PyThresh benchmarking. The results of this benchmarking are shown in Figure 23. On the horizontal axis, the different thresholding techniques implemented in the module are listed. On the vertical axis, the distance to the MCC of the Top-N strategy is plotted.

To measure the performance of the different mixtures and Top-N thresholder, the Matthew's Correlation Coefficient (MCC, [5]) and the Silhouette Score (SSC, [17]) are used.
The MCC is a supervised measure - as it is computed from the confusion matrix such as depicted in Table 5 - defined as

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FN) \cdot (TN + FP)}}.$$

It is an alternative to the accuracy in case of analysing imbalanced data sets ([5]). Indeed, when considering the accuracy as performance measure, it is hard to beat the *majority class* strategy, which marks every observation as an inlier. This way, the accuracy will equal $(100 - p)$, with p the percentage of outliers. The MCC maps the confusion matrix to the range [-1, 1], where an MCC of 1 indicates perfect prediction and -1 an inverted perfect prediction. The majority class strategy, as well as random guessing, yields an MCC of 0.
The SSC is a well-known measure to validate the quality of a clustering solution, based on the concepts of cohesion and separation. The former measures the average (Euclidian) distance between observations within one cluster, whereas the latter measures this average distance between (representative points of)

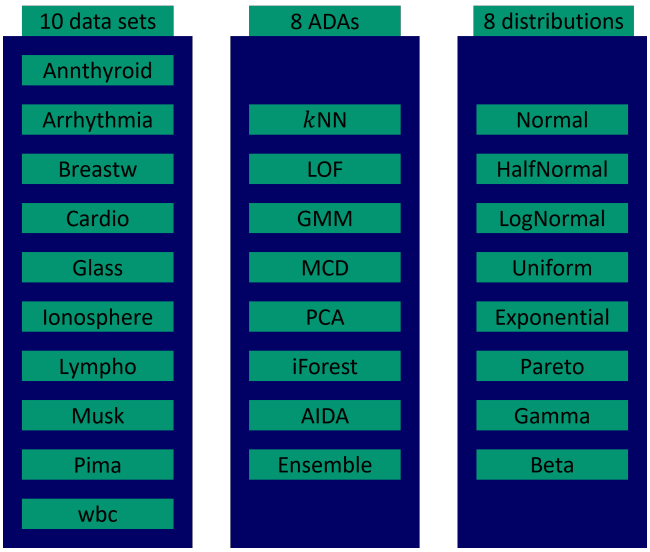


Figure 13: Setup of the experiment on real-world data sets.

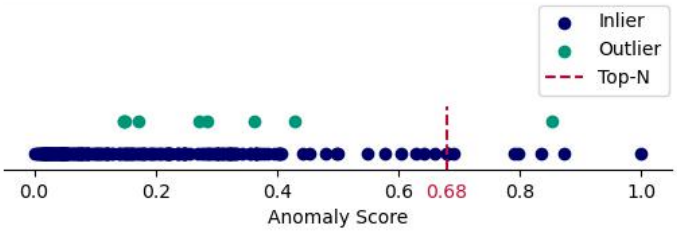


Figure 14: Output of ADA iForest on data set `glass`.

clusters. Both the cohesion and separation are computed in an unsupervised manner - as they do not use true labels, but only predicted labels - and are included in this thesis as well in order to simulate real-world analysis. As the true labels are often inaccessible in applications, having an unsupervised measure to discriminate on model performances is necessary. The SSC maps predicted labels to the range [-1,1], where an SSC of 1 indicates perfect separation and cohesion. Note that the SSC does not exist if only one cluster is formed.

One remark should be made here. The described measures are often applied for evaluation of classification and clustering solutions. However, this does not necessarily reflect the quality of the threshold. For example, if the results of Figure 14 are thresholded, the quality of a threshold $\tau = 0.95$ will equal the quality of a threshold $\tau = 1$. Indeed, both will mark only one observations as outlier, resulting in equivalent classification/clustering. Unfortunately, the author was not aware of any measures for evaluation of the threshold directly.

		Predicted labels	
		Negative	Positive
True labels	Negative	TN	FP
	Positive	FN	TP

Table 5: Confusion matrix. In this paper, "negative" and "positive" correspond to inliers and outliers respectively.

Results

Let's first consider the performance of the ADAs on the different data sets. The performance of the ADAs on the different data sets is measured by the Area Under the ROC Curve (AUC). To compute this measure, for each possible threshold the true positive rate and false positive rate are plotted. The AUC then equals the area under this curve. The AUC is a value between 0.5 and 1, where a random guessing strategy yields an AUC of 0.5. In case of a perfect prediction, i.e. the ADA distinctively separates the inliers and outliers, the AUC will equal 1.

Figure 15 displays the AUC values of the ADAs on the data sets. The last column contains the mean AUC of the ADAs over all data sets. From this last column, it can be seen that all ADAs - expect the LOF - perform equally well on average, as the mean AUCs are within 0.06 range of each other. The LOF is the only exception as its mean AUC is only 0.712. The top three performing ADAs are MCD, iForest and the Ensemble.

When specific ADA-data set combinations are considered, we see that five ADAs - MCD, PCA, iForest, AIDA and Ensemble - perform (almost) perfectly on the data set `musk` (as the AUC values are rounded to three decimal values). The same five ADAs are close to 1 for some other data sets as well. In Figure 16, we can see that these ADAs have (an almost) perfect separation of the inliers and outliers, whereas the KNN and LOF do not. On the contrary, the three worse performances were by LOF.

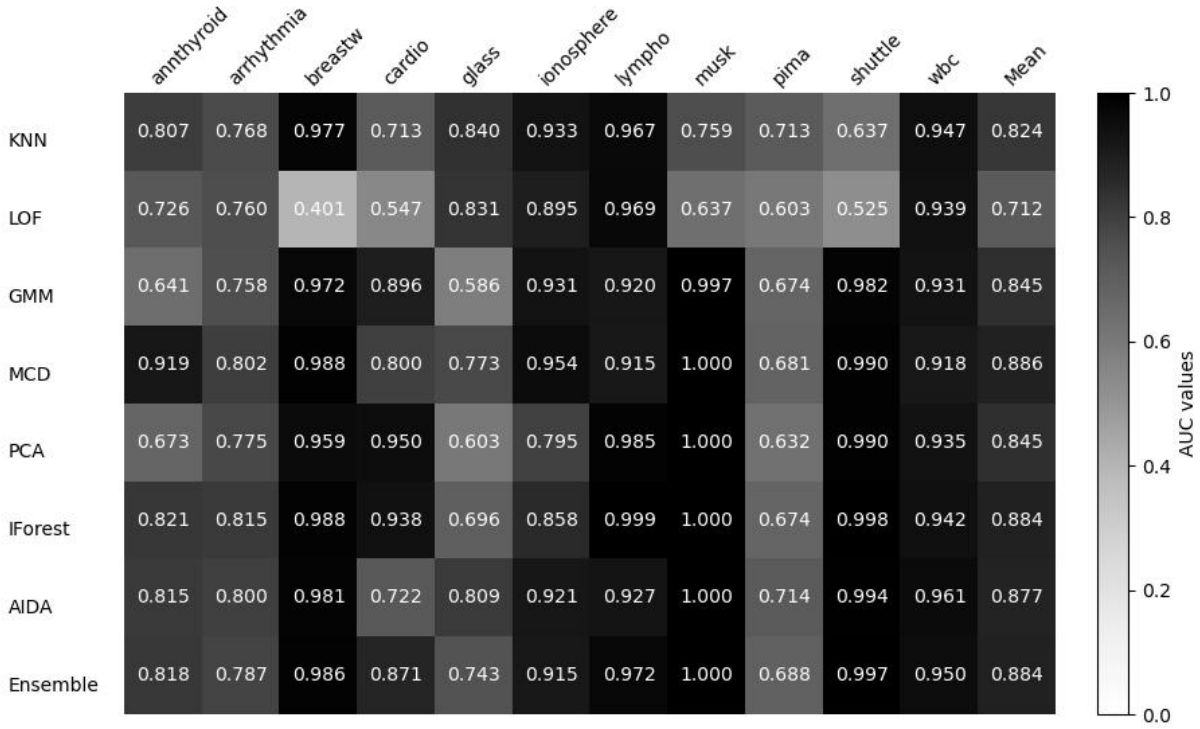


Figure 15: AUC values of all ADA-data set combinations.

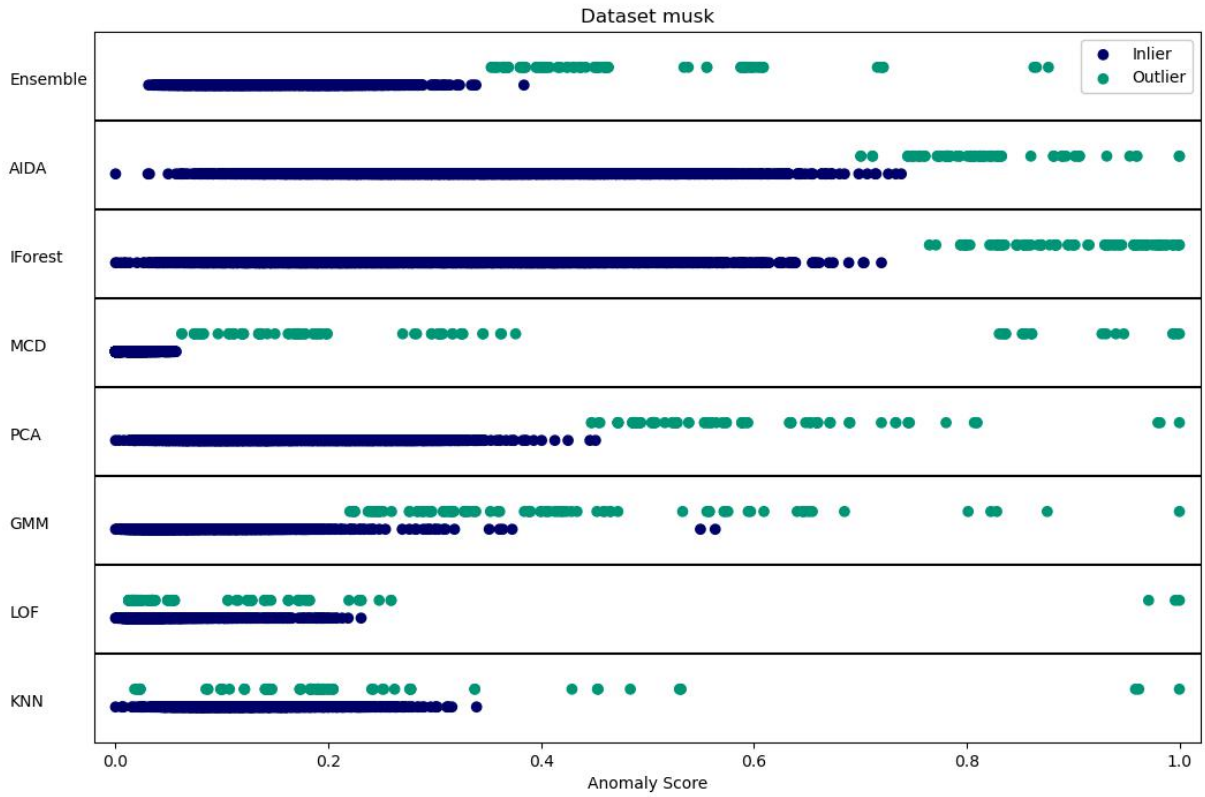
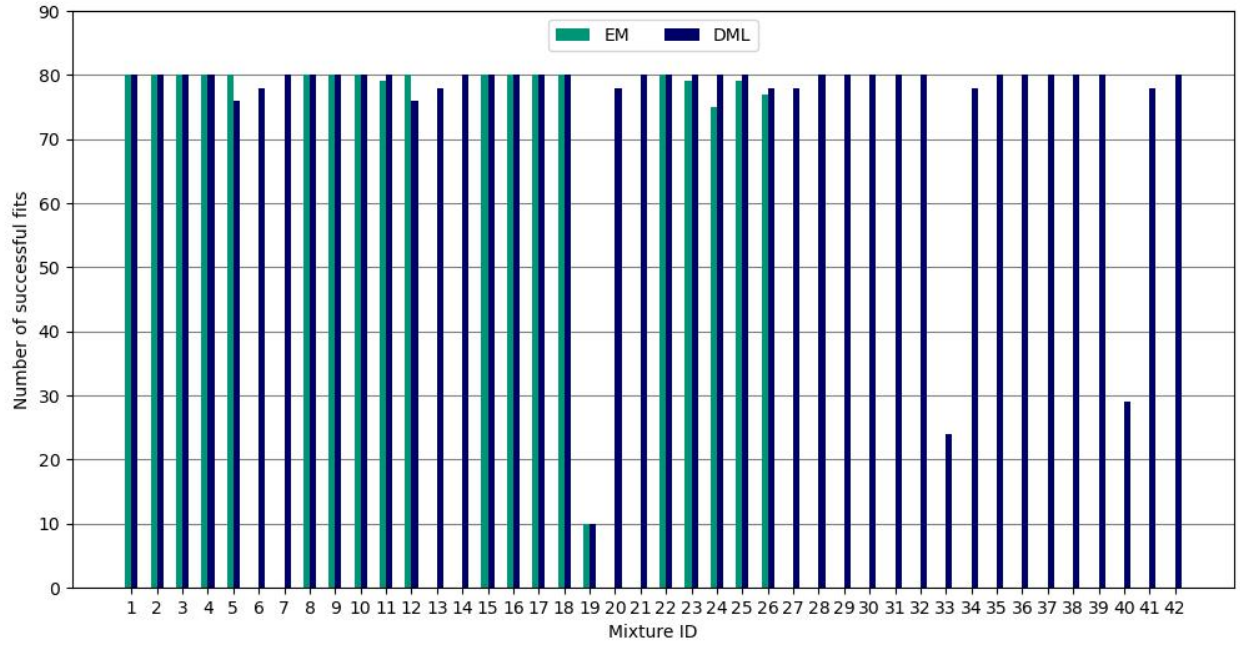
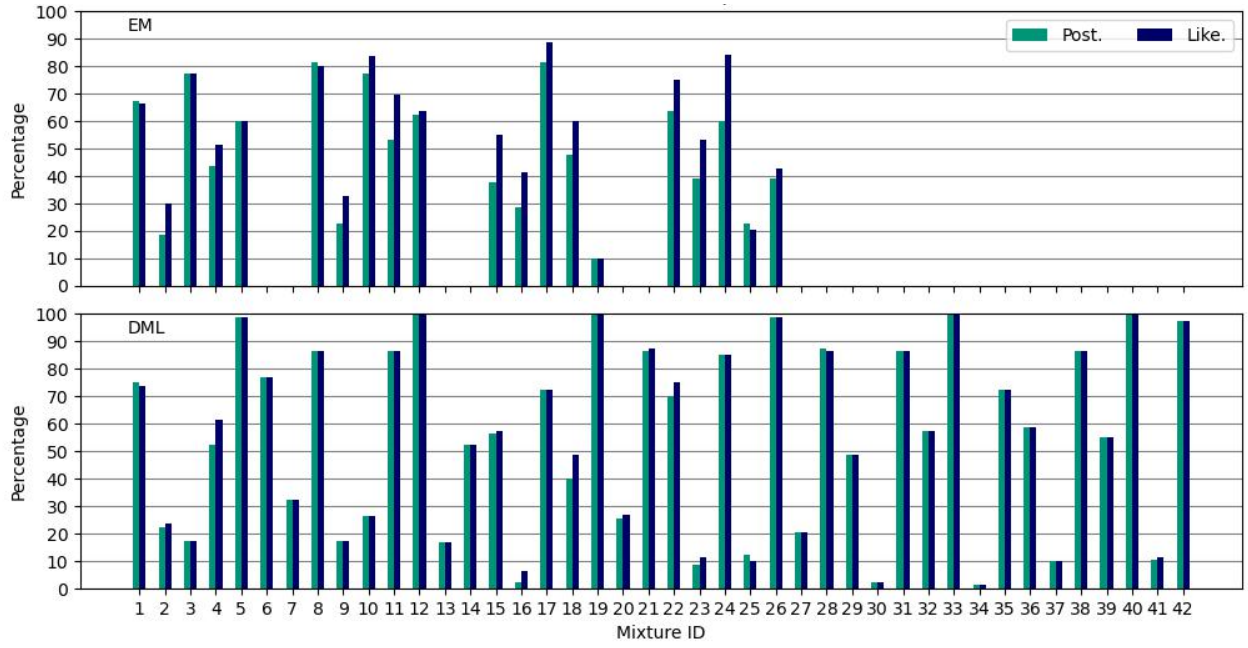


Figure 16: Anomaly scores computed for data set `musk`.

Figure 17a shows the number of times the estimation algorithms successfully estimated the mixture parameters. Note that all mixtures including a gamma or beta distribution are only fitted by DML, i.e. mixtures 6-7, 13-14, 20-21 and 27-42 are not fitted by EM. The figure shows overall good results as the number of successful fits is 80 almost everywhere. The algorithms seem to have only trouble with estimating mixture 19 (log-normal - Pareto), as the number of successful fits is only 10. The DML algorithm also has trouble with estimating the parameters of mixture 33 (gamma-Pareto) and mixture 40 (beta-Pareto).
Figure 17b shows the percentage of cases that the threshold could be found after a successful parameter estimation. It can be seen that finding a threshold from the results of EM causes trouble for every mixture, as no mixture reaches 100%. It can also be seen that the likelihood-threshold is found more often than the posterior-threshold.
The barchart for DML in Figure 17b shows better results, as finding a threshold in 100% of the cases succeeded for approximately seven mixtures. Two of these mixtures are mixture 33 (gamma-Pareto) and mixture 42 (beta-Pareto), the two mixtures with which the DML algorithm had trouble to estimate. That means that five mixtures perform really well in terms of calibrating a threshold successfully. These mixtures are mixture 5, mixture 12, mixture 19, mixture 26 and mixture 40, all of which apply the Pareto distribution to model the outliers. The difference between likelihood- and posterior-thresholds is less present for the results of DML.



(a) Number of times the estimation algorithms successfully estimated the mixture parameters.



(b) Number of cases the threshold could be found after successful parameter estimation. The barchart on top shows the results for EM, the bottom displays the results for DML.

Figure 17: Results on the number of thresholds found.

In the remainder of this subsection, we will zoom in on a poorly performing ADA, an ADA that performs excellent and the overall performance of the ADAs, keeping in mind the number of successful fits. The combination GMM-`annthyroid` - with an AUC of 0.641 - is considered as the poor case, the iForest-`musk` as the excellent case (since its AUC value is approximately 1). The output of the GMM algorithm on the `annthyroid` data set is plotted in Figure 18. It can be seen that the outliers are not very well separated from the inliers, as there are lot of outliers having a low anomaly score and, in addition, the three highest anomaly scores are assigned to inliers. We refer back to Figure 16 for the anomaly scores of iForest-`musk`.

The results of the GMM-`annthyroid` fit - the poorly performing ADA - are shown in Figure 19. In Figure 20, the results of the fit on the iForest anomaly scores computed for the `musk` data set - the excellent ADA - are displayed. On the axes, the MCC and SSC are plotted. For the GMM-`annthyroid` results, the logarithm of the two measures is taken to magnify differences. In addition, the axes are
The hypothesis that finding a good mixture based on a poor ADA is confirmed by the figures. This conclusion can be drawn from the fact that, for the poor ADA, the maximal log-MCC is approximately -0.827. This log-MCC is scored by the likelihood-threshold calibrated from the results of mixture 22 (exponential-normal) fitted by DML. The log-MCC of -0.827 corresponds with an MCC of $10^{-0.827} = 0.149$, which is relatively low compared with the performances of the mixtures on the excellent ADA.
Regarding the Top-N strategy, the figures give rise to different conclusions. The Top-N strategy is outperformed by several mixtures for the poor ADA, as there are mixtures that have higher MCC or SSC. In fact, the likelihood-threshold computed on the results of EM outperforms the Top-N strategy in both the MCC and the SSC. The opposite is true for the good ADA, as the Top-N strategy is only outperformed by mixture 5 (Normal-Pareto) in terms of SSC and is approximately matched in performance by mixture 1 (Normal-Normal), if the EM algorithm is used to estimate mixture parameters.
Secondly, there are some mixtures that perform really well in terms of SSC, but very poor in terms of MCC. For example, mixture 17 (LogNormal-Uniform) in Figure 19a has an almost perfect and the highest SSC, whereas the MCC score is the lowest of all mixtures. Mixture 10 (HalfNormal-Uniform) in Figure 20a shows similar behaviour.

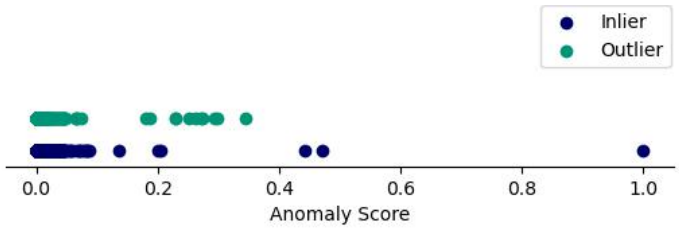
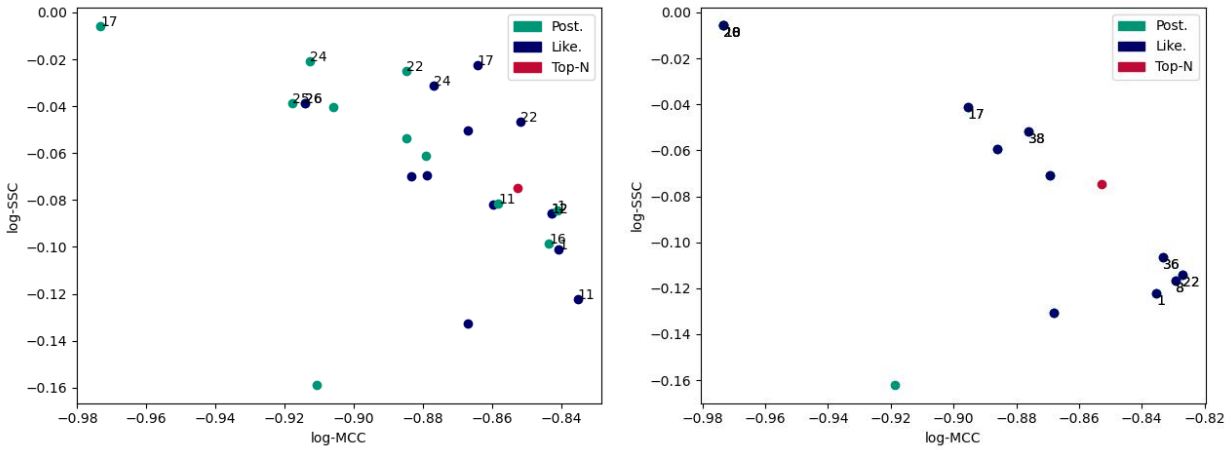
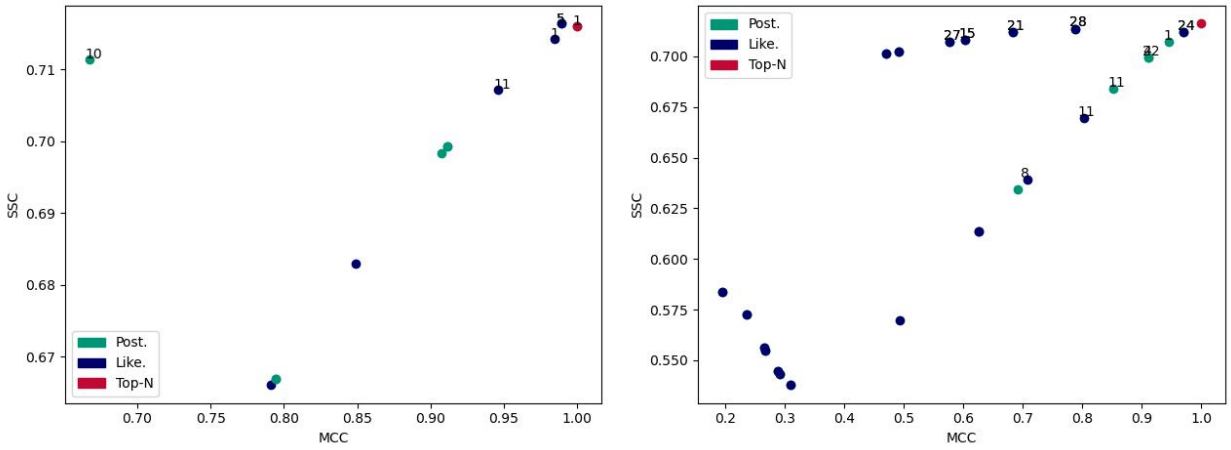


Figure 18: Output of GMM on the data set `annthyroid`.



(a) Results of EM. (b) Result of DML.

Figure 19: Performance of the mixtures on output of GMM on data set `annthyroid` (the poor ADA). For the mixture names, see the list in Appendix B.



(a) Results of EM. (b) Result of DML.

Figure 20: Performance of the mixtures on output of iForest on data set `musk` (the excellemt ADA). For the mixture names, see the list in Appendix B.

Regarding the average performance of the mixtures, Figure 21 and Figure 22 have been constructed. Every mixture is fitted a total of 160 times - 10 data sets, 8 ADAs, 2 estimation algorithms - except for the mixtures including the gamma or beta distributions. The latter ones are only fitted by DML, which results in 80 fits. For each fit, the posterior- and likelihood-threshold is calibrated, after which the MCC and SSC are computed. This is also done for the Top-N strategy described in the experimental setup. With all mixtures and the Top-N performances evaluated, the difference of the mixture performance and Top-N strategy is computed, the so-called MCC and SSC deterioration. Hence, if a mixture performed better than the Top-N, this deterioration will be positive. If the mixture performed worse, the deterioration will be below zero. Then, for each mixture, the mean and standard deviation of the MCC and SSC deterioration are computed and plotted. The uncertainty bound is constructed by adding and subtracting 2 times the standard deviation of the measure from the mean.

It is remarkable that in both figures mixture 30 (gamma - log-normal) comes out on top in the performed experiment. In terms of MCC, it is even the only mixture outperforming the Top-N strategy on average and it has the lowest variance of all mixtures. However, from Figure 17b, we learn that mixture 30 has very few thresholds found. More precisely, in only four cases a threshold could be found, which make the results of mixture 30 unreliable. Therefore ignoring mixture 30, no mixture was able to outperform the Top-N strategy on average in terms of MCC, but in terms of SSC multiple mixtures were performing better than the Top-N strategy.
Both performance measures give different results in the sense that the ranking of the mixtures is completely different. For example, mixture 1 (normal-normal) is among the best performers in terms of MCC, but is found back on the 30th place in the ranking based on the SSC. The only mixtures that perform quite well for both performance measures are mixture 9 (half-normal - log-normal) - ranked 6th for MCC and 10th for SSC - and mixture 10 (half-normal - uniform) - ranked 10th for MCC and 11th for SSC -.

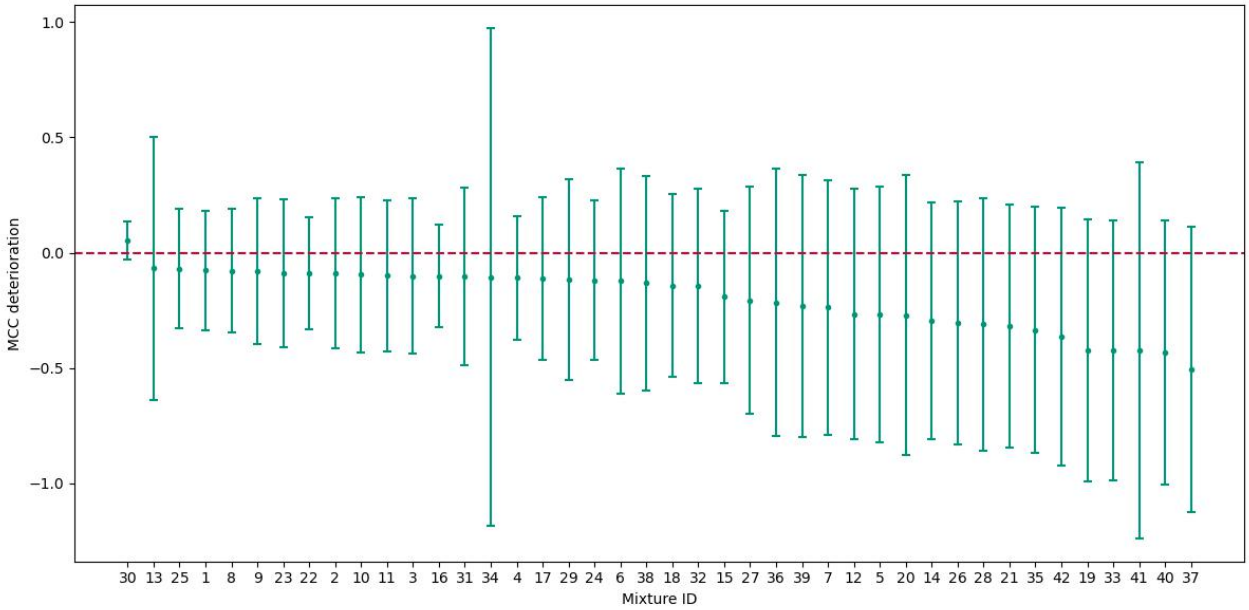


Figure 21: Average Matthew's Correllation Coefficient deterioration.

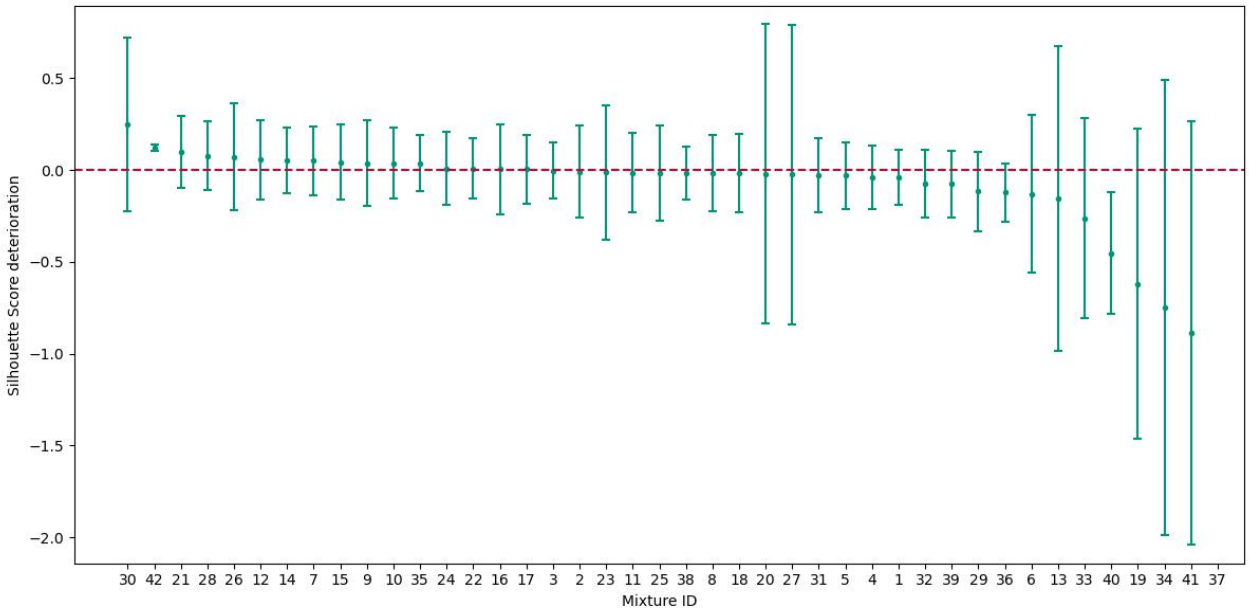


Figure 22: Average Silhouette Score deterioration.

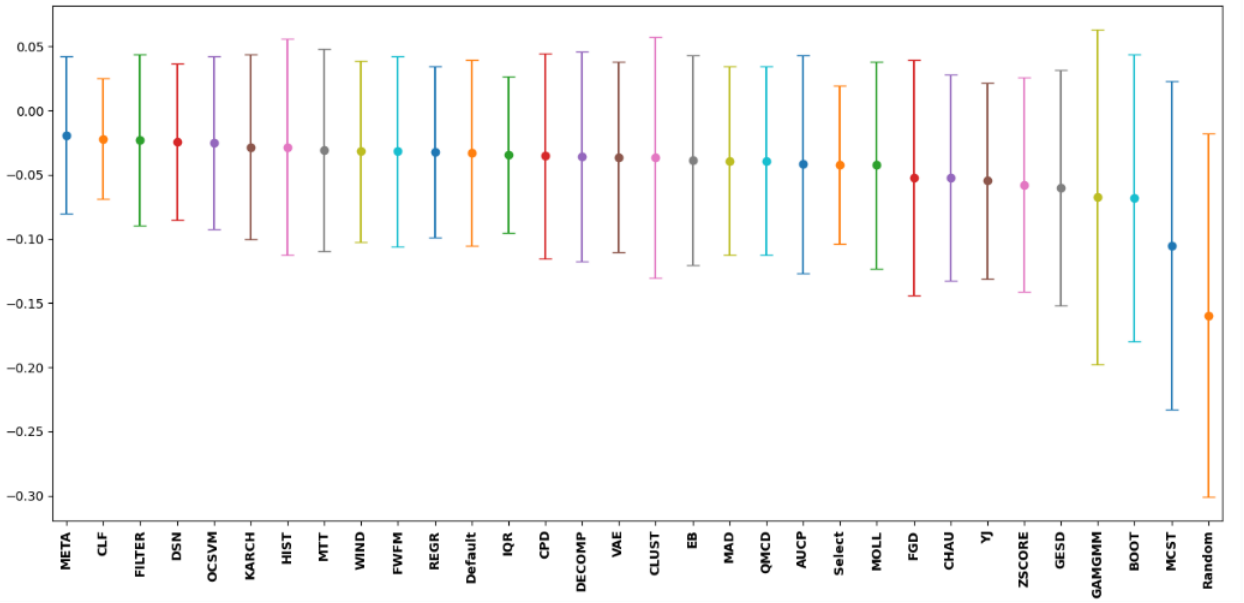


Figure 23: Results of the benchmarking of PyThresh. The figure is directly copied from the presented results. The y-axis shows the MCC deterioration, equivalent to Figure 21.

Conclusion

The first conclusion to draw is that calibrating a threshold on mixtures fitted by EM is less reliable than on mixtures fitted by DML, as can be seen from Figure 17b.

From the results plotted in Figure 19 and Figure 20, we can draw the conclusion that the performance of the mixtures depends on the quality of the output of the applied ADA. Furthermore, the Top-N strategy could not be outperformed by the mixtures in case of a well-performing ADA. However, there are mixtures that are close to or even matching the Top-N strategy. The likelihood-threshold computed for mixture 1 (normal-normal) fitted by EM matches the Top-N strategy, whereas the likelihood-threshold computed for mixture 5 (normal-Pareto) is close to the Top-N strategy in terms of MCC and has a better SSC. This is a positive result, as the mixtures perform as good as the Top-N strategy while using no prior information on the number of thresholds.
In case of a poor ADA, there are multiple mixtures that outperform the Top-N strategy. One of those is mixture 22 (exponential-normal), outperforming the Top-N strategy in both MCC and SSC.
Regarding the average performance of the mixtures, other mixtures than the mentioned mixtures 1, 5 and 22 come out as top performers. Again, the Top-N strategy could not be outperformed, but some mixtures are able to come close to the performance of the Top-N strategy in terms of the MCC. For the unsupervised SSC, the Top-N is outperformed by about 11 mixtures. Mixture 9 (half-normal - log-normal) and mixture 10 (half-normal - uniform) can be consider the best here, as they are close to the Top-N strategy in terms of MCC and outperform the Top-N strategy in terms of SSC.
Based on the same figures, the MCC and SSC do not rank the mixtures in a similar way. Indeed, the top performers in terms of MCC are totally different compared to the top performers in terms of SSC. This is unfortunate, as the MCC measures accurate prediction but is infeasible in a unsupervised setting.

A similar conclusion is drawn from the results of the benchmarking of PyThresh. In Figure 23, there is no thresholding technique outperforming the Top-N strategy in terms of MCC. This can be seen from the fact that no average is higher than 0.
Comparing the PyThresh benchmark with the results of the mixtures, (1) the variance of the PyThresh thresholder is much lower than the variances of the mixtures and (2) the PyThresh thresholders are much closer to the Top-N strategy. To draw the latter conclusion, note that the scale of the y-axis is different.

3.3 Testing on Financial Data

In the last experiment, the mixture models are tested in a real unsupervised environment. The notion of "correctness" from experiment 1, the Top-N strategy and the MCC of experiment 2 are not applicable anymore, as the information of the true labels is not available. Therefore, this experiment is not focused on correctness or whether or not the mixtures outperform the Top-N, but the main quantity of interest is the stability through time.

Before explaining the experimental setup, there is one remark to make. With the monitoring through time comes another challenge. The aim of machine learning models is to retrieve the mechanism that generated a data set of interest. It is implicitly assumed that this mechanism is stable over time, therefore being able to produce similar data as before. In practice, this assumption of a stable mechanism does not always hold. Due to disruptions - such as the covid-19 pandemic - systems change and data generating mechanisms change with it, producing data sets that differ from the ones on which the machine learning model is trained. This concept of evolving data generating mechanisms is known as *data drift*. There exist several hypothesis tests that detect these data drifts, from which the two-sample Kolmogorov-Smirnov (KS) test might be the best known. The test is based on the empirical CDFs of the two samples, computing the test-statistic as the largest difference between the empirical CDFs. The two-sample KS test is implemented in the scipy.stats module and in this thesis the KS test is performed with a significance level of 0.05.
When a threshold shows jumps or behaves volatile through time, it may be that the mixture produces

instable results. However, it is important to keep in mind that volatile behaviour might be due to data drift, as fitting a mixture on a completely different data set is prone to result in different thresholds. Therefore, instead of looking at the stability through time, we will be looking at the stability of the threshold for the periods between the points in time at which the KS test gives the conclusion that data drift has occurred.

Experimental Setup

The data sets used in this experiment are provided by Triodos Bank and are, therefore, confidential. The sets include transactional data, collected during certain short periods of time. These short periods of time are referred to as *timestamps* and 6 timestamps are combined into one *period*. By a moving window approach, 156 of such periods are extracted. To reduce information on the bank's customers, but allow for inspecting the behaviour of the mixture models through time, the thresholds and number of observations in the 156 periods are indexed at the start.
The pre-processing steps includes three steps. Firstly, the data is log-transformed. Secondly, the data set shows that many transactions are made with round amounts, for instance €500 or €1000, causing spikes to occur in the data. The data is smoothed by adding $Normal(0, 0.1)$-random noise, drawn with a seed of 302. Lastly, the data is scaled into the range [0,1] by applying a minmax-conversion.
The number of observations in each of these periods is plotted in the histogram of Figure 24. It can be seen that the number of observations per period fluctuates, with periods 90-156 having more than four times the amount of data compared to the first period. On the contrary, periods 20-90 have half the amount of data. As we have seen in Section 3.1, having more data to work with increases the performance of the threshold. Therefore, it is expected that finding thresholds for periods 20-90 is harder than for periods 90-156.
The red line on the bottom of Figure 24 indicate the points in time at which data drift was detected by the KS test. For a precise description of the way these changepoints are computed, see Algorithm 3.

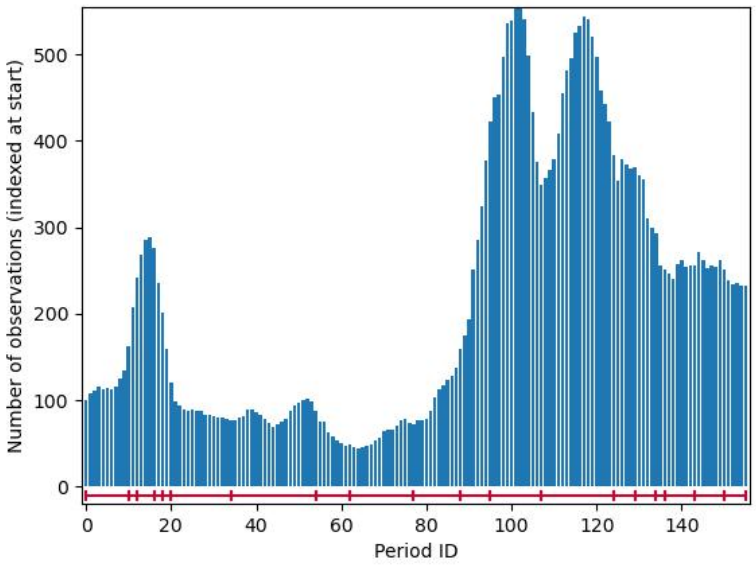


Figure 24: Number of observations for each of the 156 periods.

Algorithm 3: Computation of changepoints.

Input: The 156 data sets of Triodos Bank $\mathcal{D} = \{d_0, d_1, ..., d_{155}\}$.
Output: List of period IDs at which data drift is detected.

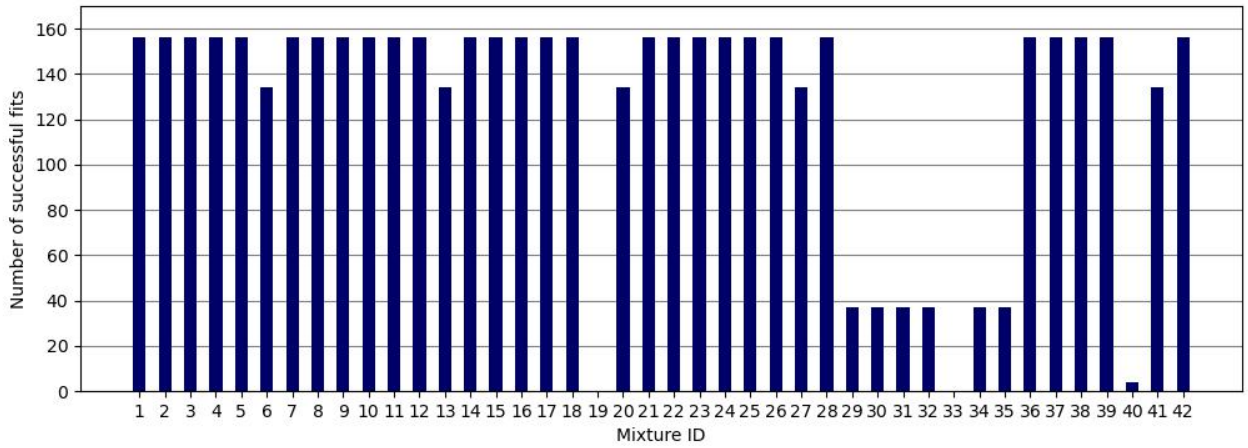
changepoints, anchor_idx = [ ], 0
for i in $\{1, 2, ..., 155\}$:
    if ks_pvalue(d_{anchor_idx}, d_i) < 0.05:
        Add i to list of changepoints, anchor_idx = i
Return changepoints.

In this section, the same 42 mixtures as in Section 3.2 are compared see Appendix B. For that purpose, the data of each of the 156 periods are considered as anomaly scores and form the basis of the fitting procedure. Important to note is that we only use the DML algorithm to fit mixtures on the data, as the results of Section 3.2 show that the DML algorithm is more promising than the EM algorithm.

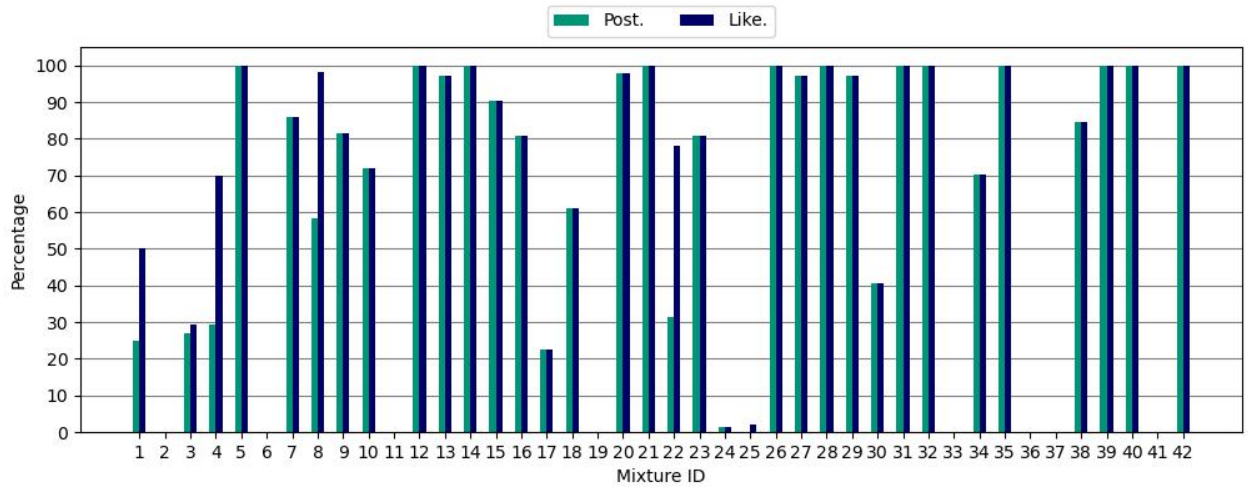
As this experiment is executed in a truly unsupervised environment, the MCC can not be used as performance measure. Therefore, we rely on the Calinsky-Harabasz Score (CHS, [17]) and the SSC from experiment 2 as measures of performance. The CHS is similar to the SSC in the sense that it is an unsupervised measure, a higher CHS is considered a better performance and it is also based on cohesion and separation. Unlike the SSC, it is unbounded from above.
The stability of the threshold is measured in terms of the variance of the threshold, taking into account the changepoints. Hence, for each of the 19 distinct intervals in Figure 24, we compute the variance. A low variance would indicate a stable threshold.

Results

Let us first inspect the number of successful fits and thresholds found. Figure 25a shows the number of successful fits per mixture. It can be seen that estimating the parameters of most mixtures is not a problem for DML. However, the DML algorithm has more trouble with the mixtures involving the gamma distribution. Indeed, the number of successful fits is 133 for mixtures which apply the gamma distribution to model the outliers (mixtures 6, 13, 20, 27, 41) and 38 successes for mixtures including the gamma distribution to model the inliers (mixtures 29 - 35). Furthermore, some of the mixtures involving a Pareto distribution are not fitted at all or only in a few cases, as can be seen from the bars of mixture 19 (log-normal - Pareto), mixture 33 (gamma-Pareto) and mixture 40 (beta-Pareto).
Figure 25b shows that both types of thresholds could be computed an equal number of times, with exception of mixtures 1, 3, 4, 8 and 22. It seems that it has something to do with the normal distribution, as mixtures 1 is the normal-normal mixture, mixtures 3 and 4 use the normal distribution to model the inliers and mixtures 8 and 22 use the normal distribution to model the outliers. In addition, (almost) no thresholds could be found for mixtures 2, 6, 11, 24, 25, 36, 37. There seems to be no particular pattern in these mixtures, as they involve different distributions.



(a) Number of times the DML algorithm successfully estimates the mixture parameters.



(b) Number of cases the threshold could be found after the mixture is successfully fitted.

Figure 25: Results on the number of thresholds found.

The average scores of all the mixtures are plotted in Figure 26. To magnify differences in performances, the logarithm is taken. The results in Figure 26 show that the likelihood-thresholds perform better in terms of CHS than the posterior-thresholds. In addition, we see that some of the mixtures scoring very good in terms of SSC, but not in terms of CHS, for instance mixture 22 combined with a posterior-threshold. The best performer in terms of SSC is the posterior-thresholds resulting from the estimation of mixture 4 (normal-exponential), as its log-SSC is approximately -0.21. This yields an SSC of $10^{-0.21} = 0.617$. Compared with the results from Section 3.2, this is relatively low. Hence, finding a threshold with high cohesion and separation is more difficult for the financial data sets than for the benchmark data sets.
Mixture 25 (exponential-exponential) could be considered as the best performer here, as it has one of the highest performances in terms of both CHS and SSC. However, Figure 25b shows that the likelihood-threshold could only be computed in a few cases. The average score of mixture 25 is therefore unreliable, as it might have been a lucky shot/some lucky shots.

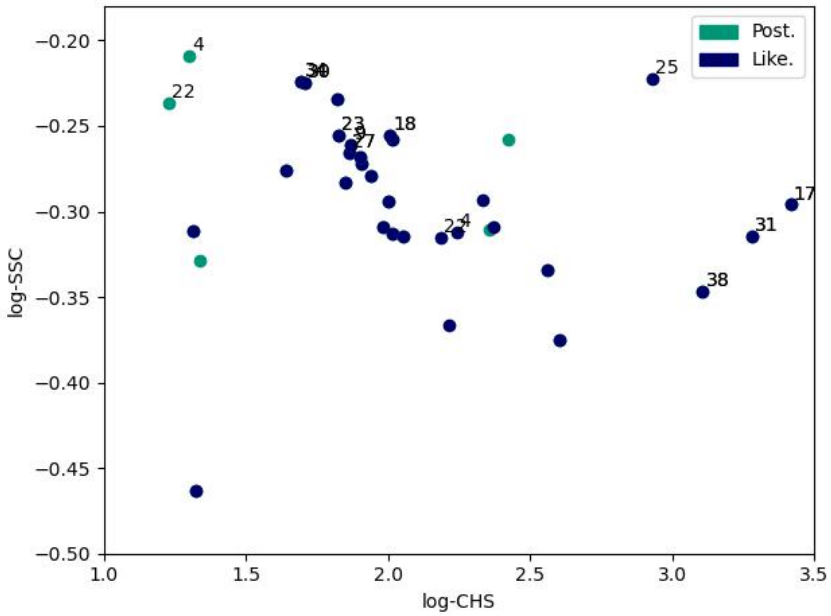


Figure 26: Performance of the mixtures on the 156 periods of the Triodos Bank data set. For the mixture names, see the list in Appendix B.

Considering the average CHS and SSC in combination with the number of successfully found thresholds, it is evident that no mixture-threshold combination can be considered as *the best* performer. The top performers often have only few thresholds found and, therefore, these results are not reliable. Mixtures for which thresholds could be computed in at least 70% of the number of successful fits tend to have lower MCCs and SSCs. Five mixtures - in combination with the likelihood-threshold - have been identified as being moderate on average, based on two conditions. The first condition is that a high number of thresholds should be found, i.e. the mixture should be fitted on all 156 data sets and in at least 70% of the cases a threshold should be found. Secondly, the mixture should be among the top performers in terms of one of the performance measures or the mixture should score moderately in terms of both performance measures. The five mixtures are mixture 4 (normal-exponential), mixture 9 (half-normal - log-normal), mixture 22 (exponential-normal), mixture 23 (exponential - log-normal), mixture 38 (beta-uniform).

For the five mixtures stated above, the stability is investigated in more detail. As shown in Figure 24, the entire timeline can be divided into 19 intervals. Based on the KS test, the data from the periods that fall inside such an interval can be considered similar. A stable threshold should have a low variance within such

	Periods	Number of periods
Interval 1	0 - 9	10
Interval 6	20 - 33	14
Interval 7	34 - 53	20
Interval 9	62 - 76	15
Interval 10	77 - 87	11
Interval 12	95 - 106	12
Interval 13	107 - 124	17

Table 6: Intervals containing at least ten periods.

an interval, indicating that the calibrated threshold does not differ much from period to period. As some intervals only contain a few periods, we continue with intervals having at least ten periods. These intervals are stated in Table 6.

Besides the variance of the threshold itself, the variance in number of outliers selected and the variance of the outlier percentage are computed. From these three variances, Figure 27 is constructed. The way this figure should be read is that, for example, the log-variance of the number of outliers of mixture 4 for interval 1 is approximately 0.5, which implies the variance itself to be approximately $10^{0.5} = 3.162$. Hence, the variance of the number of outliers selected by the likelihood-threshold of mixture 4 is approximately 3.162. Recall that the thresholds are computed on minmax-scaled data and are, therefore, values in the range [0,1] as well. From Figure 27, we see that the five selected mixtures have trouble with computing thresholds for the periods in interval 9. This follows from the observation that there is no graph plotted for that interval, indicating that there were no or just one threshold to compute a variance over. This corresponds with the note made on the data set sizes, as the periods in interval 9 contain the least amount of observations.
Regarding the stability of the mixtures, mixture 38 is considered as the most stable in terms of the threshold itself. Indeed, mixture 38 has the lowest variance for all intervals (except interval 9). On the contrary, mixture 38 has the highest variance in terms of outlier percentage and number of outliers. For those two graphs, mixtures 9 and 23 are considered the most stable.

Conclusion

From Figure 25, it can be concluded that mixtures including the gamma distribution do not perform well, as the number of successful fits is lower than for other mixtures. Combining the number of successes with the average performance resulted in a shortlist of five mixtures. If the stability of the threshold itself is important, then mixture 38 (beta-uniform) can be considered the best performer. If the stability of the amount of observations marked as outlier is of more importance, mixtures 9 (half-normal - log-normal) and 23 (exponential - log-normal) can be considered the top performers.

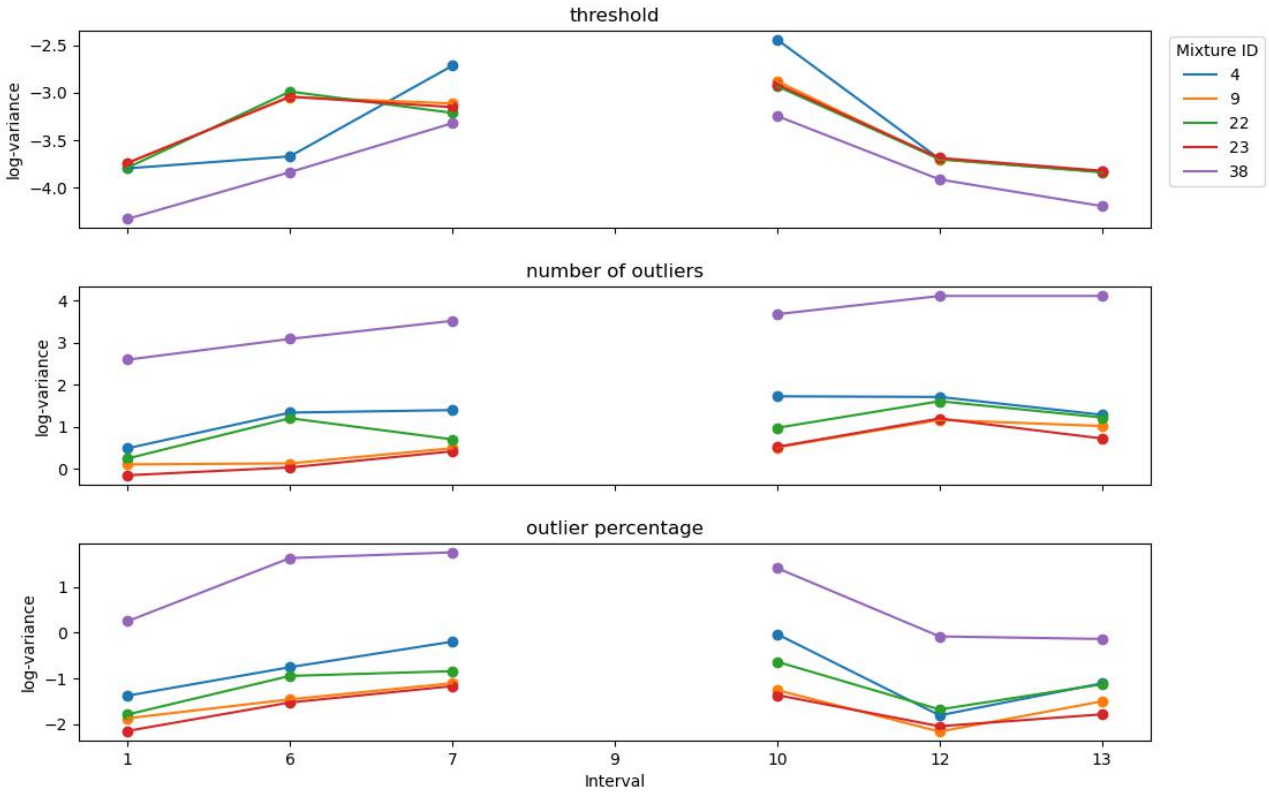


Figure 27: Logarithm of the variance for each interval.

4 Conclusion

4.1 Research Questions

The main goal of this research was to investigate mixture models as basis of calibration of thresholds on anomaly scores. Three research question have been posed, which will be answered in this section.

Question 1: What mixture to use? First, the subquestion of which estimation algorithm is answered. In experiment 1 (Section 3.1 it was concluded that, on a perfect data set, the EM algorithm produces better results than the DML algorithm. Note that with "a perfect data set" we mean that the data originates exactly from the mixture that is fitted. In addition, the DML algorithm was a little bit more sensitive to the settings of the algorithm.
However, in experiment 2 (Section 3.2) we saw the EM algorithm having trouble to find a threshold on a non-perfect data set. Furthermore, the execution time of the EM algorithm increases when no closed-form update rules exist. Based on these observations, we conclude that the DML algorithm is the estimation method to go for, as it seems to be more reliable than the EM algorithm.

To answer the question if there is added value in altering the mixture components, we refer to Figure 21, Figure 22 and Figure 26. In terms of the MCC of Figure 21, the normal-normal mixture (mixture 1) is only outperformed by mixtures that have few thresholds found. As these results might therefore unreliable, it can be concluded that the normal-normal mixture is the best performer on average.
However, regarding Figure 22 and Figure 26, the normal-normal mixture is not among the top performers for the unsupervised measures. This indicates that the normal-normal mixture performs worse in terms of separation and cohesion than some other mixtures. Hence, there could be a gain in performance if the components of the mixture are altered.

Question 2: How does the mixture model-based threshold compare with other thresholders?
The answer to this question is two-fold. On average, it seems like the mixture do not perform better than other thresholding techniques implemented in PyThresh. This can be seen from Figure 21 and Figure 23, as the average MCC of the methods in PyThresh is closer to zero and have a lower variance.
When comparing the mixture model-based threshold to the state-of-the-art Top-N strategy, the mixture model-thresholds definitely have an added value. In case of a good quality ADA, Figure 20 shows that the mixture model-based thresholds perform almost as good as the top-N strategy, while using no prior information on the number of thresholds. This is definitely an advantage of the mixture model-based thresholds, as such information is not accessible in real world applications. In addition, if the quality of the ADA is bad, Figure 19 shows that the Top-N strategy is outperformed by many mixtures.
Hence, the conclusion is that the mixture model-based thresholds perform better on average than the Top-N strategy, but perform worse than other thresholding techniques implemented in PyThresh.

Question 3: Are the mixture model-based thresholds stable?
From Figure 27, the likelihood-thresholds of the most stable thresholds have a variance of around 10^{-4}, indicating that the thresholds are within a range of 0.04 of each other. The outlier percentage resulting from the likelihood-thresholds is about $10^{-1.5} \approx 0.0316$, indicating that the outlier percentage is within a range of 1% of each other. From these observations, it can be concluded that the mixtures mark approximately evenly many observations as outlier, indicating that they are stable.
From Figure 27, it could also be seen that different mixture come out as most stable when the stability importance is associated differently. In case of the stability being associated with the threshold itself, mixture 38 (beta-uniform) was considered most stable. In case of a stable number/percentage of observations marked as outlier, which would resemble the top-N strategy, mixture 23 (exponential - log-normal) was considered the most stable.
The answer to this question is affirmative, with a remark that the number of observations should be large enough. Otherwise, fitting the mixtures will result in a threshold only rarely.

4.2 Discussion

The aim of this thesis is to investigate if mixture models are reliable and stable methods to set thresholds on anomaly scores. A popular way of fitting a mixture model is the EM algorithm and it is said to be a valid method as well. However, in this thesis finding a threshold based on the fit of the EM algorithm was troublesome, making it hard to compare the performance of the EM and DML algorithms thoroughly. As far as the author knows, the code that generated the results was correct. Therefore, the trouble of finding thresholds might be due to poor estimation of the parameters. More research could have been done about why the EM algorithm shows a low amount of successfully found thresholds.

The failure of threshold computation might be related to the scaling applied to the anomaly scores. This scaling ensured the anomaly scores to be mapped in the range [0,1]. However, in some mixtures the update rules of EM prescribed to take a logarithm of the anomaly scores or divide by it. Therefore, the scaling might not have been appropriate as the lowest anomaly scores could not be used for these mixtures, since taking the logarithm of or dividing by zero is of course infeasible. The mixtures could generate better results if this scaling was not applied. In addition, a log-transformation was applied to the financial data.

In the estimation algorithms, some choices have been made. In the EM algorithm these choices concerned the estimation of mixtures including the uniform, Pareto, gamma or beta distributions. For the former two, parameters were fixed at a certain value, while the latter two relied on numerical root-finding in the M step. For the DML algorithm, the L-BFGS-B method was chosen as optimisation algorithm. Maybe other maximisation algorithms were more appropriate to use.

For construction of the sets of anomaly scores in Section 3.2, all ADAs were fitted with default hyperparameters. Especially in the case of kNN and LOF, the value of k might have a great impact on the performance. Ideally, the ADAs should have been fitted for different values of the hyperparameters and included in this thesis.

4.3 Future Work

An important aspect of the experiment in an unsupervised setting was that the SSC could be used as performance measure. However, from the results of Section 3.2 we concluded that the SSC yielded very different results compared to the MCC. As the MCC measures correct prediction, and the ranking of the mixtures was different, the SSC is not reliable in terms of correct predictions. Ideally, we would have an unsupervised measure yielding equivalent results as the MCC, such that we are sure that the unsupervised measure measures correct prediction. In that case, evaluating the performance of the mixtures in terms of this unsupervised measure ensures that we draw conclusions about the correctness of the mixtures. Finding such an unsupervised measure could be the subject of future work.

Regarding the execution time of fitting all mixtures on data sets of relatively small size (the largest data set had 7200 observations), this execution time could explode in case of really big data sets. Then, fitting all 42 mixtures becomes infeasible. An important question is how to identify which mixture to use from the data, in case visual inspection is not feasible. Constructing (the framework of) a search method that selects the (most promising) mixture(s) could be the subject of future work. One could for example build a framework upon nested models - as the exponential distribution is a special case of the gamma distribution - or distributions from the exponential family, which include many of the distributions used in this thesis.

In the last experiment, the stability of the mixtures in combination with the threshold type was studied. Here, the mixtures were compared to each other, but not to a state-of-the-art thresholding strategy. To really form a conclusive answer, the top-N strategy or any other thresholding method from PyThresh should be monitored over time as well. A more thorough study of the stability of the proposed thresholding strategy by mixture models could be the subject of future work.

Lastly, one of the desired properties of a thresholder was formulated as being able to produce an in-between threshold. Although it could be seen that the produce threshold was indeed in between observed anomaly scores, there was no further testing if this was any good. Ideally, a train/test split should have been made and the computed threshold should be examined on the test set. If true anomalies in the test set were more often marked as anomalies by the in-between threshold, it could have been concluded that the in-between requirement was useful. This could be the subject of future work.

Acknowledgements

I would like to thank all my supervisors for the discussions, ideas and valuable suggestions during our (bi-)weekly meetings. In addition, I would like to give a special thanks to Triodos Bank for providing me with an opportunity to undergo an internship and for suggesting an interesting project. I am also appreciative of the bank for allowing me to be part of the E-crime team for seven months, a period of time during which I learned a lot an gained much experience. I will certainly look back with joy to this internship.

References

[**1**] Luis Antonio Souto Arias, Cornelis W Oosterlee, and Pasquale Cirillo. Aida: Analytic isolation and distance-based anomaly detection algorithm. *Pattern Recognition*, 141:109607, 2023.

[**2**] Aya Ayadi, Oussama Ghorbel, Abdulfattah M Obeid, and Mohamed Abid. Outlier detection approaches for wireless sensor networks: A survey. *Computer Networks*, 129:319–333, 2017.

[**3**] Marco Bee. On discriminating between lognormal and pareto tail: an unsupervised mixture-based approach. *Advances in Data Analysis and Classification*, pages 1–19, 2022.

[**4**] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[**5**] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13, 2020.

[**6**] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *A Modern Introduction to Probability and Statistics: Understanding why and how*, volume 488. Springer, 2005.

[**7**] Francis Ysidro Edgeworth. Xli. on discordant observations. *The london, edinburgh, and dublin philosophical magazine and journal of science*, 23(143):364–375, 1887.

[**8**] Jing Gao and Pang-Ning Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 212–221. IEEE, 2006.

[**9**] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 35:32142–32159, 2022.

[**10**] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[**11**] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. *An introduction to statistical learning*, volume 112. Springer, 2013.

[**12**] Hans-Peter Kriegel, Peer Kroger, Erich Schubert, and Arthur Zimek. Interpreting and unifying outlier scores. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 13–24. SIAM, 2011.

[**13**] Daniel M Kulik. Pythresh 0.3.3 documentation. https://pythresh.readthedocs.io/en/latest/index.html, 2022.

[**14**] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[**15**] Daniel Oberski. Mixture models: Latent profile and latent class analysis. *Modern statistical methods for HCI*, pages 275–287, 2016.

[**16**] Jason W Osborne and Amy Overbay. The power of outliers (and why researchers should always check for them). *Practical Assessment, Research, and Evaluation*, 9(1):6, 2004.

[**17**] Julio-Omar Palacio-Niño and Fernando Berzal. Evaluation metrics for unsupervised learning algorithms. *arXiv preprint arXiv:1905.05667*, 2019.

[**18**] Lorenzo Perini, Paul Buerkner, and Arto Klami. Estimating the contamination factor's distribution in unsupervised anomaly detection. *arXiv preprint arXiv:2210.10487*, 2022.

[**19**] Peter J Rousseeuw and Christophe Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical association*, 88(424):1273–1283, 1993.

[**20**] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. Outlier detection: how to threshold outlier scores? In *Proceedings of the international conference on artificial intelligence, information processing and cloud computing*, pages 1–6, 2019.

[**21**] Yue Zhao. Pyod 1.1.0 documentation. https://pyod.readthedocs.io/en/latest/index.html, 2022.

[**22**] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

Appendices

Appendix A: Estimation of Some Distributions

In iteration k of the M-step of the EM-algorithm the parameters of the mixture model are estimated by maximising the expected complete loglikelihood given the incomplete dataset $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ of anomaly scores and the estimated class assignments of the previous E-step $\mathcal{T}^{(k-1)} = \{t_1^{(k-1)}, t_2^{(k-1)}, ..., t_n^{(k-1)}\}$. That is, we want to maximise the function

$$\begin{aligned} Q(\Theta, \hat{\Theta}^{(k-1)}) &= E\Big[\ell(\omega, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1 \mid \mathcal{S}, \mathcal{T}^{(k-1)}) \mid \mathcal{S}, \hat{\Theta}^{(k-1)}\Big] \\ &= \sum_{i=1}^{n} (1 - \hat{t}_i^{(k-1)}) \log(1-\omega) + (1 - \hat{t}_i^{(k-1)}) \log\big(f_0(s_i \mid \boldsymbol{\theta}_0)\big) \\ &\qquad + \hat{t}_i^{(k-1)} \log(\omega) + \hat{t}_i^{(k-1)} \log\big(f_1(s_i \mid \boldsymbol{\theta}_1)\big). \end{aligned} \tag{17}$$

In this appendix we will derive the estimators in the M-step for several distributions. Hence, we will take derivatives, equate them to zero and solve for the parameter(s) of interest. When we estimate the parameters of the inlier-distribution, we ignore all the terms except for $(1-\hat{t}_i)\log(f_0(s_i \mid \boldsymbol{\theta}_0))$. Similarly, when estimating the parameters of the outlier-distribution, we only consider $\hat{t}_i \log(f_1(s_i \mid \boldsymbol{\theta}_1))$. In the following we will consider the distributions to model the outliers and derive the estimates. These estimates can also be used when the distribution models the inliers: just replace all $\hat{t}_i$'s by $1 - \hat{t}_i$.
The considered distributions are listed below.

- Normal distribution;
- Exponential distribution;
- Uniform distribution (discussed in Section 2.3);
- Lognormal distribution;
- Halfnormal distribution;
- Pareto distribution (discussed in Section 2.3);
- Gamma distribution;
- Beta distribution.

Normal Distribution

The Normal distribution has parameters μ and σ^2, respectively the mean and the variance. The probability density function (PDF) of the Normal distribution is

$$f(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big).$$

Then to find the estimates of μ and σ^2 in the M-step, we want to maximise

$$\begin{aligned} Q(\Theta, \hat{\Theta}^{(k-1)}) &= \sum_{i=1} \hat{t}_i^{(k-1)} \log\big(f(s_i \mid \mu, \sigma)\big) \\ &= \sum_{i=1}^{n} -\hat{t}_i^{(k-1)} \log(\sqrt{2\pi}) - \frac{1}{2}\hat{t}_i^{(k-1)} \log(\sigma^2) - \frac{\hat{t}_i^{(k-1)}(s_i-\mu)^2}{2\sigma^2} \end{aligned}$$

The maximisers are computed by equating partial derivatives to zero and solving for the parameters. It follows for μ that

$$\begin{aligned}\frac{\partial}{\partial\mu}Q &= \sum_{i=1}^{n}\frac{\hat{t}_i^{(k-1)}(s_i-\mu)}{\sigma^2}\\ &= \frac{1}{\sigma^2}\sum_{i=1}^{n}\hat{t}_i^{(k-1)}s_i - \frac{1}{\sigma^2}\mu\sum_{i=1}^{n}\hat{t}_i^{(k-1)}\end{aligned}$$

Then equating to zero and solving for μ yields

$$\mu = \frac{\sum_{i=1}^{n}\hat{t}_i^{(k-1)}s_i}{\sum_{i=1}^{n}\hat{t}_i^{(k-1)}}.$$

For the estimation of σ^2, we have

$$\frac{\partial}{\partial\sigma^2}Q = -\frac{1}{\sigma^2}\sum_{i=1}^{n}\hat{t}_i^{(k-1)} + \frac{1}{\sigma^4}\sum_{i=1}^{n}\hat{t}_i^{(k-1)}(s_i-\mu)^2$$

Equating this to zero and solving for σ^2 yields

$$\sigma^2 = \frac{\sum_{i=1}^{n}\hat{t}_i^{(k-1)}(s_i-\mu)^2}{\sum_{i=1}^{n}\hat{t}_i^{(k-1)}}.$$

With the expressions above, the update rules of the parameters of the Normal distribution in the M-step are

$$\begin{aligned}\hat{\mu}^{(k)} &= \frac{\sum_{i=1}^{n}\hat{t}_i^{(k-1)}s_i}{\sum_{i=1}^{n}\hat{t}_i^{(k-1)}}\\ \hat{\sigma}^{(k)} &= \Big(\frac{\sum_{i=1}^{n}\hat{t}_i^{(k-1)}(s_i-\hat{\mu}^{(k)})^2}{\sum_{i=1}^{n}\hat{t}_i^{(k-1)}}\Big)^{\frac{1}{2}}\end{aligned} \tag{18}$$

Exponential Distribution

The Exponential distribution has one parameter λ, which is called the rate. The probability density is

$$f(x \mid \lambda) = \lambda\exp(-\lambda x).$$

To find the estimate for λ in the M-step, we want to maximise

$$\begin{aligned}Q(\Theta,\hat{\Theta}^{(k-1)}) &= \sum_{i=1}\hat{t}_i^{(k-1)}\log\big(f(s_i \mid \mu,\sigma)\big)\\ &= \sum_{i=1}^{n}\hat{t}_i^{(k-1)}\log(\lambda) - \hat{t}_i^{(k-1)}\lambda s_i\end{aligned}$$

The partial derivative w.r.t. λ then yields

$$\begin{aligned}\frac{\partial}{\partial\lambda}Q &= \sum_{i=1}^{n}\frac{\hat{t}_i^{(k-1)}}{\lambda} - \hat{t}_i^{(k-1)}s_i\\ &= \frac{1}{\lambda}\sum_{i=1}^{n}\hat{t}_i^{(k-1)} - \sum_{i=1}^{n}\hat{t}_i^{(k-1)}s_i.\end{aligned}$$

Equation to zero and solving for λ then gives

$$\lambda = \frac{\sum_{i=1}^{n} \hat{t}_i^{(k-1)}}{\sum_{i=1}^{n} \hat{t}_i^{(k-1)} s_i},$$

which is the reciprocal of the weighted mean. Hence, the update rule for λ in the M-step is

$$\hat{\lambda}^{(k)} = \frac{\sum_{i=1}^{n} \hat{t}_i^{(k-1)}}{\sum_{i=1}^{n} \hat{t}_i^{(k-1)} s_i} = \left(\hat{\mu}^{(k)}\right)^{-1},$$

with $\hat{\mu}^{(k)}$ as given in (18).

Lognormal Distribution

Similar to the Normal distribution, the Lognormal distribution has parameters for the mean μ and variance σ^2 as well. The probability density is given as

$$f(x \mid \mu, \sigma) = \frac{1}{x\sqrt{2\pi\sigma^2}} \exp\Big(- \frac{(\log(x) - \mu)^2}{2\sigma^2}\Big).$$

To find the estimators of μ and σ^2, we want to maximise

$$\begin{aligned} Q(\Theta, \hat{\Theta}^{(k-1)}) &= \sum_{i=1} \hat{t}_i^{(k-1)} \log\big(f(s_i \mid \mu, \sigma)\big) \\ &= \sum_{i=1}^{n} -\hat{t}_i^{(k-1)} \log(s_i) - \hat{t}_i^{(k-1)} \log(\sqrt{2\pi}) - \frac{1}{2}\hat{t}_i^{(k-1)} \log(\sigma^2) - \hat{t}_i^{(k-1)} \frac{(\log(s_i) - \mu)^2}{2\sigma^2} \end{aligned}$$

Leaving out terms that do not include μ of σ^2 gives

$$Q(\Theta, \hat{\Theta}^{(k-1)}) = \sum_{i=1}^{n} -\frac{1}{2}\hat{t}_i^{(k-1)} \log(\sigma^2) - \hat{t}_i^{(k-1)} \frac{(\log(s_i) - \mu)^2}{2\sigma^2}$$

For μ, we want to find the root of

$$\frac{\partial}{\partial \mu} Q = \frac{1}{\sigma^2} \sum_{i=1}^{n} \hat{t}_i^{(k-1)} (\log(s_i) - \mu)$$

which yields

$$\mu = \frac{\sum_{i=1}^{n} \hat{t}_i^{(k-1)} \log(s_i)}{\sum_{i=1}^{n} \hat{t}_i^{(k-1)}}.$$

For σ^2, we want to find the root of

$$\frac{\partial}{\partial \sigma^2} Q = -\frac{1}{\sigma^2} \sum_{i=1}^{n} \hat{t}_i^{(k-1)} + \frac{1}{\sigma^4} \sum_{i=1}^{n} \hat{t}_i^{(k-1)} (\log(s_i) - \mu)^2$$

which yields

$$\sigma^2 = \frac{\sum_{i=1}^{n} \hat{t}_i^{(k-1)} (\log(s_i) - \mu)^2}{\sum_{i=1}^{n} \hat{t}_i^{(k-1)}}.$$

Hence, the update rules for the parameters of the lognormal distribution in the M-step are

$$\hat{\mu}^{(k)} = \frac{\sum_{i=1}^{n} \hat{t}_i^{(k-1)} \log(s_i)}{\sum_{i=1}^{n} \hat{t}_i^{(k-1)}}$$
$$\hat{\sigma}^{(k)} = \Big(\frac{\sum_{i=1}^{n} \hat{t}_i^{(k-1)} (\log(s_i) - \hat{\mu}^{(k)})^2}{\sum_{i=1}^{n} \hat{t}_i^{(k-1)}} \Big)^{\frac{1}{2}}$$

Gamma Distribution

The $Gamma(\alpha, \beta)$ distribution has density

$$f(x \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x).$$

Hence, to find estimators for α and β we want to optimise

$$\begin{aligned} Q(\Theta, \hat{\Theta}^{(k)}) &= \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(f(s_i \mid \alpha, \beta)) \\ &= \sum_{i=1}^{n} \alpha \hat{t}_i^{(k)} \log(\beta) - \hat{t}_i^{(k)} \log(\Gamma(\alpha)) + \hat{t}_i^{(k)} (\alpha - 1) \log(s_i) - \hat{t}_i^{(k)} \beta s_i. \end{aligned}$$

Taking the partial derivative w.r.t. β gives

$$\frac{\partial}{\partial \beta} Q = \sum_{i=1}^{n} \frac{\alpha \hat{t}_i^{(k)}}{\beta} - \sum_{i=1}^{n} \hat{t}_i^{(k)} s_i.$$

Setting this expression equal to zero and solving for β gives

$$\beta = \frac{\alpha \sum_{i=1}^{n} \hat{t}_i^{(k)}}{\sum_{i=1}^{n} \hat{t}_i^{(k)} s_i} = \frac{\alpha}{\bar{s}^{(k)}},$$

with $\bar{s}^{(k)}$ the weighted mean of the anomaly scores. Substituting this back into the expected complete likelihood gives

$$\begin{aligned} Q(\Theta, \hat{\Theta}^{(k)}) &= \sum_{i=1}^{n} \alpha \hat{t}_i^{(k)} \log(\frac{\alpha}{\bar{s}^{(k)}}) - \hat{t}_i^{(k)} \log(\Gamma(\alpha)) + \hat{t}_i^{(k)} (\alpha - 1) \log(s_i) - \hat{t}_i^{(k)} \frac{\alpha}{\bar{s}^{(k)}} s_i \\ &= \sum_{i=1}^{n} \alpha \hat{t}_i^{(k)} \log(\alpha) - \alpha \hat{t}_i^{(k)} \log(\bar{s}^{(k)}) - \hat{t}_i^{(k)} \log(\Gamma(\alpha)) + \hat{t}_i^{(k)} (\alpha - 1) \log(s_i) - \hat{t}_i^{(k)} \frac{\alpha}{\bar{s}^{(k)}} s_i. \end{aligned}$$

Taking the derivative w.r.t. α yields

$$\begin{aligned} \frac{\partial}{\partial \alpha} Q &= \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(\alpha) + \hat{t}_i^{(k)} - \hat{t}_i^{(k)} \log(\bar{s}^{(k)}) - \hat{t}_i^{(k)} \psi(\alpha) + \hat{t}_i^{(k)} \log(s_i) - \frac{\hat{t}_i^{(k)} s_i}{\bar{s}^{(k)}} \\ &= \log(\alpha) \sum_{i=1}^{n} \hat{t}_i^{(k)} + \sum_{i=1}^{n} \hat{t}_i^{(k)} - \log(\bar{s}^{(k)}) \sum_{i=1}^{n} \hat{t}_i^{(k)} - \psi(\alpha) \sum_{i=1}^{n} \hat{t}_i^{(k)} + \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(s_i) - \frac{1}{\bar{s}^{(k)}} \sum_{i=1}^{n} \hat{t}_i^{(k)} s_i \end{aligned}$$

where we have used that $\frac{d}{dx} \log(\Gamma(x)) = \psi(x)$, with $\psi(x)$ the digamma-function. Setting this partial derivative equal to zero gives

$$\begin{aligned} 0 &= \frac{\partial}{\partial \alpha} Q \\ 0 &= \log(\alpha) \sum_{i=1}^{n} \hat{t}_i^{(k)} + \sum_{i=1}^{n} \hat{t}_i^{(k)} - \log(\bar{s}^{(k)}) \sum_{i=1}^{n} \hat{t}_i^{(k)} - \psi(\alpha) \sum_{i=1}^{n} \hat{t}_i^{(k)} + \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(s_i) - \frac{1}{\bar{s}^{(k)}} \sum_{i=1}^{n} \hat{t}_i^{(k)} s_i \end{aligned}$$

Dividing by $\sum_{i=1}^n \hat{t}_i^{(k)}$ gives

$$
\begin{aligned}
0 &= \log(\alpha) + 1 - \log(\bar{s}^{(k)}) - \psi(\alpha) + \frac{\sum_{i=1}^n \hat{t}_i^{(k)} \log(s_i)}{\sum_{i=1}^n \hat{t}_i^{(k)}} - \frac{1}{\bar{s}^{(k)}} \cdot \frac{\sum_{i=1}^n \hat{t}_i^{(k)} s_i}{\sum_{i=1}^n \hat{t}_i^{(k)}} \\
0 &= \log(\alpha) + 1 - \log(\bar{s}^{(k)}) - \psi(\alpha) + \overline{\log(s)}^{(k)} - 1 \\
0 &= \log(\alpha) - \log(\bar{s}^{(k)}) - \psi(\alpha) + \overline{\log(s)}^{(k)}.
\end{aligned}
$$

From here we rely on numerical solvers to find α.
One possible root-finding xbar = algorithm is the Newton-Rhapson method, which requires specification of the first derivative and an initial guess. As we would like to find the root of $\frac{\partial}{\partial \alpha} Q$, the derivative used by Newton-Rhapson is

$$
\begin{aligned}
\frac{\partial^2}{\partial \alpha^2} Q &= \frac{\partial}{\partial \alpha}\Big(\frac{\partial}{\partial \alpha} Q\Big) \\
&= \frac{\partial}{\partial \alpha}\Big(\log(\alpha) - \log(\bar{s}^{(k)}) - \psi(\alpha) + \overline{\log(s)}^{(k)}\Big) \\
&= \frac{1}{\alpha} - \psi^{(1)}(\alpha),
\end{aligned}
$$

with $\psi^{(1)}$ as the polygamma function of order 1.
For the initial guess one could use the coefficient of variation, which is defined as the expectation of the distribution divided by the standard deviation. For the Gamma distribution it is known that

$$
\mu = \frac{\alpha}{\beta}; \quad \sigma = \frac{\sqrt{\alpha}}{\beta}.
$$

Hence, the coefficient of variation will equal

$$
\frac{\mu}{\sigma} = \frac{\alpha \backslash \beta}{\sqrt{\alpha} \backslash \beta} = \sqrt{\alpha}.
$$

Therefore, the initial guess for the numerical solver could be

$$
\alpha = \Big(\frac{\bar{s}^{(k)}}{\sigma_s^{(k)}}\Big)^2, \text{ with } \sigma_s^{(k)} = \Big(\frac{\sum_{i=1}^n \hat{t}_i^{(k)} (s_i - \bar{s}^{(k)})^2}{\sum_{i=1}^n \hat{t}_i^{(k)}}\Big)^{\frac{1}{2}}.
$$

In summary, to update the estimates of the parameters of the Gamma distribution in the M-step we

1) Find $\hat{\alpha}^{(k+1)}$ such that $\log(\hat{\alpha}^{(k+1)}) - \log(\bar{s}^{(k)}) - \psi(\hat{\alpha}^{(k+1)}) + \overline{\log(s)}^{(k)} = 0$, for instance with the Newton-Rhapson algorithm;
2) Compute $\hat{\beta}^{(k+1)} = \frac{\hat{\alpha}^{(k+1)}}{\bar{s}^{(k)}}$.

Beta Distribution

The density of a (α, β) function is given as

$$
f(x \mid \alpha, \beta) = x^{\alpha - 1}(1 - x)^{\beta - 1} \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}.
$$

With this density, the function to optimise in the M-step becomes

$$\begin{aligned} Q(\Theta, \hat{\Theta}^{(k)}) &= \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(f(s_i \mid \alpha, \beta)) \\ &= \sum_{i=1}^{n} \hat{t}_i^{(k)} (\alpha - 1) \log(s_i) + \hat{t}_i^{(k)} (\beta - 1) \log(1 - s_i) + \hat{t}_i^{(k)} \log(\Gamma(\alpha + \beta)) \\ &\qquad - \hat{t}_i^{(k)} \log(\Gamma(\alpha)) - \hat{t}_i^{(k)} \log(\Gamma(\beta)) \end{aligned}$$

To optimise these functions, we set the partial derivatives equal to zero. The partial derivatives are given by

$$\begin{aligned} \frac{\partial}{\partial \alpha} Q &= \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(s_i) + \hat{t}_i^{(k)} \psi(\alpha + \beta) - \hat{t}_i^{(k)} \psi(\alpha) \\ &= \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(s_i) + \psi(\alpha + \beta) \sum_{i=1}^{n} \hat{t}_i^{(k)} - \psi(\alpha) \sum_{i=1}^{n} \hat{t}_i^{(k)} \end{aligned}$$

and similarily

$$\frac{\partial}{\partial \beta} Q = \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(1 - s_i) + \psi(\alpha + \beta) \sum_{i=1}^{n} \hat{t}_i^{(k)} - \psi(\beta) \sum_{i=1}^{n} \hat{t}_i^{(k)}$$

Setting these derivatives equal to zero gives a system of equations.

$$\begin{cases} 0 = & \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(s_i) + \psi(\alpha + \beta) \sum_{i=1}^{n} \hat{t}_i^{(k)} - \psi(\alpha) \sum_{i=1}^{n} \hat{t}_i^{(k)} \\ 0 = & \sum_{i=1}^{n} \hat{t}_i^{(k)} \log(1 - s_i) + \psi(\alpha + \beta) \sum_{i=1}^{n} \hat{t}_i^{(k)} - \psi(\beta) \sum_{i=1}^{n} \hat{t}_i^{(k)} \end{cases}$$

Which can be rewritten as

$$\begin{cases} 0 = & \overline{\log(s)}^{(k)} + \psi(\alpha + \beta) - \psi(\alpha) \\ 0 = & \overline{\log(1 - s)}^{(k)} + \psi(\alpha + \beta) - \psi(\beta) \end{cases} \tag{19}$$

Hence, the update rule for the parameters of the Beta Distribution is to solve system (19) numerically. Again the initial guesses can be computed from the moments of the Beta Distribution

$$\mu = \frac{\alpha}{\alpha + \beta}; \quad \sigma^2 = \frac{\alpha\beta}{(\alpha + \beta^2)(\alpha + \beta + 1)},$$

by means of

$$\alpha = \mu\Big(\frac{\mu(1 - \mu)}{\sigma^2} - 1\Big); \quad \beta = (1 - \mu)\Big(\frac{\mu(1 - \mu)}{\sigma^2} - 1\Big).$$

Therefore, the initial guesses for the numerical solver could be

$$\begin{aligned} \alpha_0 &= \bar{s}^{(k)}\Big(\frac{\bar{s}^{(k)}(1 - \bar{s}^{(k)})}{\sigma^{2(k)}} - 1\Big) \\ \beta_0 &= (1 - \bar{s}^{(k)})\Big(\frac{\bar{s}^{(k)}(1 - \bar{s}^{(k)})}{\sigma^{2(k)}} - 1\Big), \end{aligned}$$

with

$$\bar{s}^{(k)} = \frac{\sum_{i=1}^{n} \hat{t}_i^{(k)} s_i}{\sum_{i=1}^{n} \hat{t}_i^{(k)}}; \quad \sigma^{2(k)} = \frac{\sum_{i=1}^{n} \hat{t}_i^{(k)} (s_i - \bar{s}^{(k)})^2}{\sum_{i=1}^{n} \hat{t}_i^{(k)}}.$$

Appendix B: Overview of Considered Mixtures

On the next page is an extensive list of the mixtures considered in the numerical experiments of Section 3.2 and Section 3.3.

Mixture ID	Inlier Distribution	Outlier Distribution	Only Fitted by DML
1		Normal	
2		Log-Normal	
3		Uniform	
4	Normal	Exponential	
5		Pareto	
6		Gamma	✓
7		Beta	✓
8		Normal	
9		Log-Normal	
10		Uniform	
11	Half-Normal	Exponential	
12		Pareto	
13		Gamma	✓
14		Beta	✓
15		Normal	
16		Log-Normal	
17		Uniform	
18	Log-Normal	Exponential	
19		Pareto	
20		Gamma	✓
21		Beta	✓
22		Normal	
23		Log-Normal	
24		Uniform	
25	Exponential	Exponential	
26		Pareto	
27		Gamma	✓
28		Beta	✓
29		Normal	✓
30		Log-Normal	✓
31		Uniform	✓
32	Gamma	Exponential	✓
33		Pareto	✓
34		Gamma	✓
35		Beta	✓
36		Normal	✓
37		Log-Normal	✓
38		Uniform	✓
39	Beta	Exponential	✓
40		Pareto	✓
41		Gamma	✓
42		Beta	✓