Media Engineering and Technology
German University in Cairo
and
Multimedia Analysis and Data Mining Competence Center
German Research Center for Artificial Intelligence (DFKI GmbH)
Kaiserslautern, Germany

Comparison of Unsupervised Anomaly Detection Techniques

Bachelor Thesis

Author:	Mennatallah Amer
Supervisor:	Markus Goldstein
Reviewer:	Prof. Dr. Andreas Dengel Prof. Dr. Slim Abdennadher
Submission Date:	20 September, 2011

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

Mennatallah Amer
20 September, 2011

Acknowledgment

First of all, I would like to thank my parents for encouraging me to pursue my dreams and their support in the various aspects of my life. I would like to thank my brother and sister for always being there for me. I am grateful for *Prof. Dr. Slim Abdennadher* and *Prof. Dr. Andreas Dengel* for giving me the opportunity to perform my bachelor project at the German Research Center for Artificial Intelligence (DFKI GmbH). I would like to thank *Dr. Thomas Kieninger* and Jane Bensch for organizing every aspect of our stay in Germany. I am very thankful to my supervisor *Markus Goldstein* for his support and guidance through out the project. I would also like to thank my friends who were my family during this time in Germany. Finally I would like to thank everyone in the DFKI for the wonderful work environment and for making this an unforgettable experience.

Abstract

Anomaly Detection is the process of finding outlying record from a given data set. This problem has been of increasing importance due to the increase in the size of data and the need to efficiently extract those outlying records which could indicate unauthorized access of the system, credit card theft or the diagnosis of a disease. The aim of this bachelor thesis is to implement a RapidMiner extension that contains the most applicable unsupervised anomaly detection algorithms to enable non-experts to easily apply them. Second an evaluation of the implemented algorithms was carried out in an attempt to show the relative strength and weakness of the algorithms. Two new algorithms were introduced. The first one is a global variant of cluster-based local outlier factor (CBLOF) which tries to overcome its shortcomings, the second one is local density based algorithm called local density cluster-based outlier factor. The performance of the implemented and the proposed algorithms were evaluated on real world data sets from the UCI machine learning repository. The proposed algorithms showed promising results where they outperformed CBLOF.

Contents

Chapter 1

Introduction

A record is said to be anomalous or outlying if it's behavior does not conform with the behavior of the majority of the dataset. These records have been of increasing interest because their presence could indicate unauthorized usage of the system, a failure in part of the system or a diagnosis of a disease.

There are many domains which require anomaly detection systems. One of which is intrusion detection where a user can try to gain extra privileges on the system or an unauthorized user can try gain access on the system. One of the main challenges in this domain is that the data size is enormous thus even small false alarm rates would overwhelm the analyst. Moreover the nature of both normal and anomalous packets is constantly changing. Another domain is fraud detection. Organizations that needs such systems include banks, credit card companies, insurance companies and phone companies. Such systems monitor individual user profiles and report any uncommon behavior. Anomaly detection is also applied in the medical sectors. Anomalies in this domain can indicate a disease, a fault in one of the instruments or incorrect usage of the instruments, all of which require immediate action to be taken. Other domains include but are not limited to industrial damage control, image processing and sensor networks.

There are many approaches to solve the anomaly detection problem. The approaches that are more widely applicable are unsupervised approaches as they do not require labeled data. Even though many techniques exist, the strength and weaknesses of each of them are still unfolding. This bachelor thesis attempts to provide more insights into the behavior of some unsupervised anomaly detection algorithms. In order to do that a RapidMiner [10] Extension *Anomaly Detection* was developed that contains several unsupervised anomaly detection techniques. In contrast to machine learning, there is no freely available toolkit such as the extension implemented for non-experts in the anomaly detection domain. The target of the extension was to get those unsupervised algorithms efficiently implemented in order to be able to perform the necessary comparisons as well as release that extension so that non-experts, analysts and researchers could freely use the algorithms and experiment with them.

The contributions of this thesis can be summarized in the following points:

1. Releasing a RapidMiner Extension *Anomaly Detection*[1] that contains the following unsupervised anomaly detection techniques:
    (a) k-NN Global Anomaly Score.
    (b) Local Outlier Factor (LOF).
    (c) Connectivity-Based Outlier Factor (COF).
    (d) Local Outlier Probability (LoOP).
    (e) Influenced Outlierness (INFLO).
    (f) Local Correlation Integral (LOCI).
    (g) Cluster-Based Local Outlier Factor (CBLOF).
    (h) Local Density Cluster-Based Outlier Factor (LDCOF).
2. Introducing two clustering based algorithms, unweighted-CBLOF and LDCOF. The first is a global variant of CBLOF, the latter applies the local density based approach to the output of clustering algorithms.
3. Evaluation and comparison of the implemented algorithms on real world data sets.

[1] Project url http://madm.dfki.de/rapidminer/anomalydetection

Chapter 2

Background

This chapter contains an overview of anomaly detection describing the terminologies used and the different faces of the anomaly detection problem. It also gives a brief introduction on RapidMiner why it was the data mining tool of choice and the different terminologies used in the software.

2.1 Anomaly Detection

2.1.1 The Anomaly Detection Problem

In this section we will describe the various aspects of the anomaly detection problem. Information in this section was obtained from the surveys [4, 7].

There are several topics related to anomaly detection. Noise detection, which is processing the data in order to remove unwanted noise so that the patterns in the data could be better studied. However this differs from anomaly detection in the sense that in anomaly detection we are interested in finding those records rather than filtering them. Novelty Detection, which refers to the detection of new patterns that were previously absent or overlooked. The normal model is usually modified by those patterns which is the major discrepancy between it and anomaly detection. The approaches used to solve those related topics are often similar to outlier detection.

There are a number of challenges that makes the anomaly detection problem increasingly obscure. To begin with the border line between normal and anomalous behavior is often imprecise. Also in certain domain such as in intrusion detection the normal behavior is constantly evolving such that those changes might be mistakenly identified as outliers. Moreover the anomaly detection techniques needs to be adapted to the different application domains. Also the scarcity of labeled data for training and validation imposes limitations on the results and conclusions reached.

Anomalies can be classified into either *point* anomalies, *contextual* anomalies or *collective* anomalies. The earlier is when single data records deviate from the remainder of the data sets. This is the simplest kind and the one which is most addressed by the existing algorithms. Contextual anomalies is when the record has behavioral as well as contextual attributes. The same behavioral attributes could be considered normal in a giving context and anomalous in another. Whilst the collective anomalies is when a group of similar data are deviating from the remainder of the data set. This can only occur in data sets where the records are related to each other. Contextual anomalies can be converted into point anomalies by aggregating over the context. The algorithms implemented in the extension all explicitly handle point anomalies.

Anomaly Detection techniques can be applied in either *on-line* or *off-line* modes. On-line mode refers to when new data are continually introduced and anomalies should be detected in those data. Off-line mode refers to when anomalies are detected in already existing databases.

Anomaly detection techniques can be divided according to the type of training data expected. The first type is the supervised technique. This technique operate on two phases first the training phase and then the testing phase. The training set for supervised algorithms contains labels for both normal and abnormal records, according to this set a model is learned in the training phase that is latter used to label the unclassified record. The second type is the semi-supervised. Similar to the earlier technique it also requires a training phase. The training set however contains only normal records. A record would be labeled outlying if it deviates from the learned normal model. The third technique is the unsupervised technique. The unsupervised technique expects no labels in the training set. It usually does not have a training phase as all the data is expected to be present before the initialization of the algorithm. It assumes that anomalies are much less common than the normal data in the data set.

Even though the unsupervised anomaly detection might not be the most robust technique, it is the most applicable. The other techniques both require labeling data to produce the appropriate training set which is an expensive, time consuming and burdensome task. Supervised techniques produces reliable outcomes with low false alarm rates when the abnormal behavior encountered was present in the training set. However since the range of abnormal behaviors are impossible to cover as they are always changing, the technique can not detect new types of attacks. Semi-supervised approaches are more intuitive as normal behavior can be appropriately modeled especially in fields such as aviation control and medical diagnosis. Difficulties that face this model is that the training set should cover the broad spectrum of normal behavior. An inappropriate training set would produce misleading results. Also in fields where the normal behavior is constantly evolving such as in intrusion detection the model should be frequently modified. The unsupervised technique does not require any labels which makes it more convenient. On the other hand if the main assumption that this technique employs is violated the results would be useless. Another advantage of the other two techniques is that they are appropriate for on-line anomaly detection as incoming data can be classified using the learned model. Unsupervised techniques can be adapted for on-line detection upon the availability of a sufficiently large data base which would enable comparison with new data.

There are two possible outputs for anomaly detection algorithms, *scores* and *labels*. Scoring assigns a degree of outlierness to each record, while labeling outputs whether the record is anomalous or not. Scores are more informative to analysts and they can be readily converted to labels by choosing a particular threshold. Therefore the algorithms that were implemented in this extension were chosen to output scores.

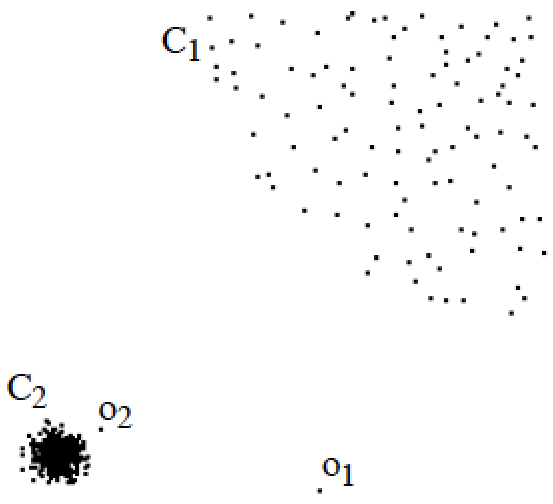


Figure 2.1: 2D data set [3] demonstrating the difference between local and global approaches.

2.1.2 Approaches to Anomaly Detection

There are many approaches proposed in order to solve the anomaly detection problem. In this section we will highlight the properties of each approach.

The approaches could either be *global* or *local.* Global approaches refer to the techniques in which the anomaly score assigned to each instance is with respect to the entire data set. On the other hand local approaches is when the anomaly score represents the outlierness of the data point with respect to its neighborhood. The local approaches detect outliers that can not be detected with the global methods. Figure 2.1 shows an example of such a case. As shown there are two clusters C_1 and C_2 where C_2 is more dense that C_1. o_1 is an example of an outlier that can be equally detected by global and local methods. o_2 however is an outlier that can only be detected by the local methods. o_2 is an outlier relative to cluster C_2, however it will not be labeled as an outlier by the global methods as the relative distance between it and the elements in C_2 is comparable to the distance between the elements, in cluster C_1, thus if the global threshold was chosen to be this distance o_2 will be unseen by the global methods.

According to the anomaly detection survey [4] the techniques can be grouped into one of the following main categories *classification* based, *nearest-neighbor* based , *clustering* based and *statistical* based.*Classification* based algorithms are mainly supervised algorithms that assumes that the distinction between anomalous and normal instances can be modeled for a particular feature space. *Nearest-neighbor* based algorithms assume that anomalies lie in sparse neighborhoods and that they are distant from their nearest neighbors. They are mainly unsupervised algorithms. *Clustering* based algorithms work by grouping similar objects into clusters and assume that anomalies either do not belong to any cluster, or that they are distant from their cluster centers or that they belong to small and sparse clusters. The majority are unsupervised algorithms but some semi-supervised algorithms were introduced. *Statistical* approaches label objects as anomalies if they deviate from the assumed stochastic model.

In the implemented extension nearest-neighbor based and clustering based algorithms were implemented. This is mainly because they are intuitively unsupervised which suits the aim of the extension.

2.2 RapidMiner

RapidMiner [10] is an international open-source data mining framework. It enables users to model complex knowledge discovery processes as it supports nested operator chains. RapidMiner can function off-line on a given data set. Also it can be used to model complex processes by the aid of its graphical user interface. Moreover RapidMiner can be used as a library in other programs.

There are several reasons which made RapidMiner the data mining tool of choice. First it has many data loading, modeling, preprocessing and visualization methods. This avoids the trouble of preprocessing the data sets. It also helps to visualize the results. It has an easy to use yet robust graphical user interface that facilitates the modeling of different complex processes. It is also modular. This allows to use some functionalities for the extension, for example, the distance measures were used for the anomaly detection operators. Finally it is easily extensible.

2.2.1 Terminology

These are some of the common terminologies used in RapidMiner.

Example Table Is the basic data storage object in RapidMiner. The data is organized in a row-wise fashion and the columns are just rationally present. The programmers have little interaction with the example table itself.

Example Set It is based on the example table where the examples and the attributes represent the rows and the columns respectively.

Special Attribute Special attributes are those that are not used in the normal processing as they have predefined roles. For example the label, cluster, id and outlier attributes.

Operator Operators are the building blocks for RapidMiner. They can be simple operators or they can be more complex where other operators can be nested in them. The input and the output of the operators are defined as well as the conditions that the input should satisfy.

2.2.2 Including RapidMiner Extensions

There are two ways to include the RapidMiner extension into RapidMiner. First you can use the update manager which is available in the help menu to install *Anomaly Detection* extension [1]. Also the jar file can be added to the lib/plugins folder of your RapidMiner installation directory.

[1] Currently *Anomaly Detection* extension is available on Rapid-i market place which is currently running in the beta phase. In order to include the extension using the update manager you need to change the update url to *http://rapidupdate.de:8180/UpdateServer* from the preferences accessible from the tools menu.

Chapter 3

Algorithms

3.1 Nearest-Neighbor Based

Algorithms that are based on nearest-neighbor based methods assume that the outliers lie in sparse neighborhoods and that they are distant from their nearest neighbors.

Through out the remainder of the thesis let k denote a positive integer, r a real number, D the data set and partitions $\{o, p, q\} \subseteq D$.

3.1.1 Neighborhoods

The neighborhood is defined as the set of points lying near the object and thus affecting its anomaly score. There are two types of neighborhoods shown in figure 3.1; the k-neighborhood and the r-neighborhood. These neighborhoods are explained below.

k-distance(p) is equal to $d(p, q)$ where $q \in D$ and q satisfies the following conditions. The 5-distance(p) is shown in figure 3.1(a).

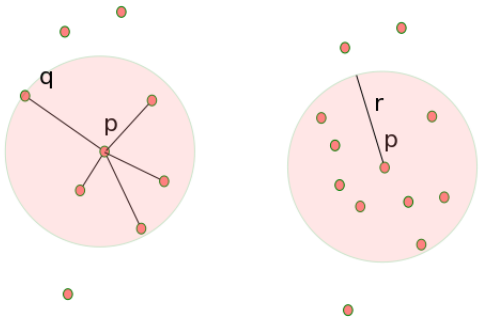
1. For at least k objects q' ∈ D it holds that $d(p, q') \leq d(p, q)$
2. For at most k-1 objects q' ∈ D it holds that $d(p, q') < d(p, q)$

k-neighborhood(p) is the set of objects that lie within k-distance(p). The shaded region in figure 3.1(a) shows the neighborhood.

r-neighborhood(p) is the set of objects lying within r distance from p. The shaded region in figure 3.1(b) shows the neighborhood.

The k-neighborhood(p) would be denoted by $N_k(p)$ and the r-neighborhood by $N(p, r)$ for the rest of the thesis.

Density based approaches that use k-neighborhood can face some problems in case there are duplicates in the data set. This arises as the density is inversely proportional to the distance and in case we have at least k+1 duplicates of some point then the k-distance would be equal to 0 and thus the estimated density would be infinite. The solution that



(a) k-neighborhood for k=5.

(b) r-neighborhood

Figure 3.1: Neighborhoods examples.

was proposed in [3] was utilized for these cases. The solution states that the conditions of the k-distance defined above would only apply to objects with distinct spatial coordinates. Meaning that if we have $D = \{p_1, p_2, p_3, p_4\}$ where the coordinates of p_2 is the same as p_3 and $d(p_1, p_2) = d(p_1, p_3) \leq d(p_1, p_4)$, then 2-distance(p_1) would correspond to $d(p_1, p_4)$ and not $d(p_1, p_3)$.

It should be noted that the k-distance(p) is always unique, while the cardinality of the k-neighborhood set could be greater that k.

3.1.2 KNN: K Nearest Neighbors

K nearest neighbors is a global distance based algorithm. The neighborhood used for this algorithm is the k-neighborhood. The anomaly score is either set to the average distance of the nearest k neighbors similar to the algorithm proposed in [2] or to the k-distance like the algorithm proposed in [12]. The earlier approach has equation 3.1.

$$knn(p) = \frac{\sum_{o \in N_k(p)} d(p, o)}{|N_k(p)|} \tag{3.1}$$

3.1.3 LOF: Local Outlier Factor

Local outlier factor was originally proposed in [3]. This is the first local density based algorithm. LOF uses the k-neighborhood.

Local density based methods compare the local density of the object to that of its neighbors. For the LOF to accomplish that the following definitions were used.

reach-dist(p,o) The reachability distance is the maximum of d(p,o) and k-distance(o). It is mainly introduced for smoothing local density.

Local reachability density (lrd) The local reachability density of object p relative to $N_k(p)$ is the inverse of the mean reachability distance over the neighborhood set.

$$lrd_{N_k(p)}(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} \text{reach-dist}(p, o)}$$

Local Outlier Factor (LOF) The local outlier factor is the ratio between the average local reachability density of the neighborhood to that of the object.

$$LOF_{N_k(p)}(p) = \frac{\sum_{o \in N_k(p)} lrd_{N_k(o)}(o)}{|N_k(p)| \cdot lrd_{N_k(p)}(p)}$$

The values of the LOF oscillates with the change in the size of the neighborhood. Therefore to improve the results a range is defined for the size of the neighborhood and the maximum LOF score over that range is taken as the final score. The authors of [3] provided some guidelines for choosing the bounds of the neighborhood size range. The lower bound should be greater than 9 in order to smooth statistical fluctuations and it should represent the size of the smallest non-outlying cluster that can be present in the data set. The upper bound should represent the maximum number of objects that can possibly be local outliers which is typically around 20.

Normal data would have a LOF score of approximately equal to 1, while outliers will have scores greater than 1. This is explained by the fact that if the data lies within a cluster then local density would be similar to that of its neighbors getting a score equal to 1. For a sufficiently large data set a LOF score of up to 2 would indicate that the point is normal.

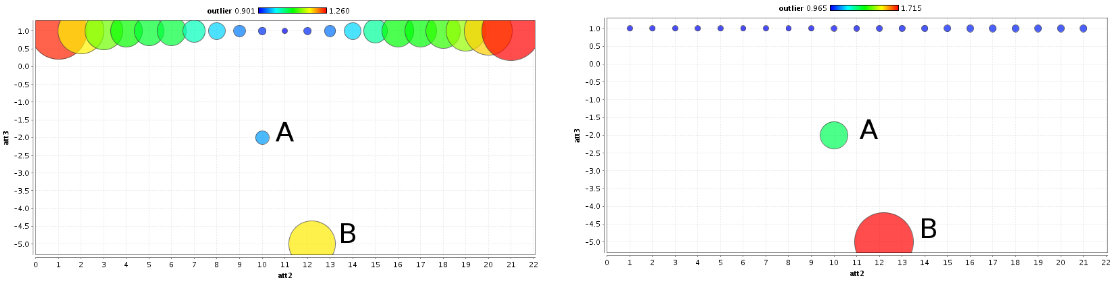
As the local density based methods are able to detect outliers that were unseen by the global methods and because of the easy interpretability of its score several variants of LOF were developed. Some of which are explained in the following sections.

3.1.4 COF: Connectivity-Based Outlier Factor

The connectivity-based outlier factor is a local density based approach proposed in [13] in order to handle outliers deviating from spherical density patterns. It is a variant of LOF that also uses the k-neighborhood.

In contrast to LOF, COF is able to handle outliers deviating from low density patterns. An example of such patterns are straight lines. The data set shown in figure 3.2 shows an example of the superiority of COF in handling outliers deviating from a straight line. The outlying points are points A and B, where A is less outlying than B. LOF fails to label A as outlying having a score of 0.957, while COF labels A as outlying having a score of 1.274. The points at the end of the straight line are given a high outlier score in case of LOF which is inaccurate as they occur within the straight line pattern. Point B is identified as outlying with both algorithms as it is sufficiently far from the pattern.

COF accomplishes its aim by representing the local density of the points differently. The local density in the COF is the inverse of the average chaining distance. The average



(a) LOF results

(b) COF results

Figure 3.2: 2D data set [13] that shows the ability of COF to handle outliers deviating from low density patterns. The color and the size are proportional to the outlier score. k=10.

chaining distance in contrast to the local reachability distance of the LOF does not use the distance between the point to the points in its neighborhood. Figure 3.3 shows the difference in calculation of the local density for LOF and COF. The dotted lines show the distances used for the local density calculations of LOF, while the solid lines show that of COF. The exact method of calculation of the average chaining distance is explained below.

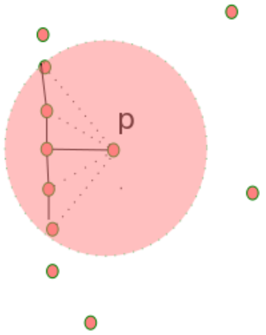


Figure 3.3: Example showing the distances used in the calculation of the local density of LOF and COF. The distances of LOF are represented by the dotted lines, while that of COF are represented by the solid lines.

During the calculation of the average chaining distance for a point p we distinguish between two sets, the set of objects connected to p and the remainder of the neighborhood set, let P denote the earlier and Q denote the latter.

The distance between the two sets and the nearest neighbor of set P in Q are defined as follows. Let $o \in P$ and $q \in Q$.

d(P,Q) The distance between the set P and Q is the minimum distance between their elements, denoted be d(P,Q)

nearest-neighbor(P,Q) q is the nearest neighbor of P in Q if ∃o such that d(o,q)= d(P,Q).

The average chaining distance is calculated using the cost description sequence(CDS). The easiest way to explain the CDS is by the use of the following pseudo code.

```
declare CDS[ $|N_k(p)|$ ]
set P to $\{p\}$
set Q to $N_k(p)$
for $i = 1 \to |N_k(p)|$ do
  set q to nearest-neighbor(P,Q)
  set CDS[i] to d(P,Q)
  set P to $P \cup \{q\}$
  set Q to $Q - \{q\}$
end for
```

The average chaining distance and the COF are computed using the following formula.

Average chaining distance

$$\text{ac-Dist}_{N_k(p_1)}(p_1) = \sum_{i=1}^{r} \frac{2(r-i+1)}{r(r+1)} \cdot CDS_i \text{ where } r = |N_k(p_1)|$$

Connectivity based outlier Factor (COF) The connectivity based outlier factor is the ratio of the average chaining distance of p to the mean of the average chaining distance over the k-neighborhood of p.

$$COF_k(p) = \frac{|N_k(p)| \cdot \text{ac-dist}_{N_k(p)}(p)}{\sum_{o \in N_k(p)} \text{ac-dist}_{N_k(o)}(o)}$$

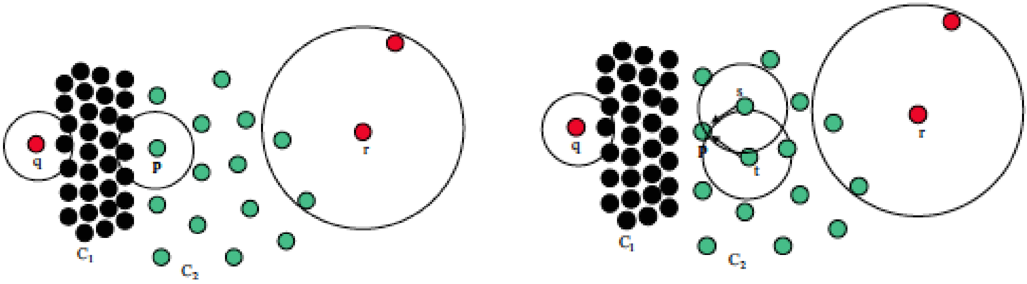
As observed by the formulas the average chaining distance is the weighted sum of the cost description sequence. The earlier edges have a greater contribution to the sum than the latter edges.

Like LOF a score near to 1 indicates that the point is not an outlier. A score much greater than 1 indicates outlierness.

3.1.5 INFLO : Influenced Outlierness

INFLO was introduced in [8]. It is also based on LOF, however it expands the neighborhood of the object to the influence space (IS) of the object.

INFLO was introduced in order to handle the case where clusters with varying densities are in close proximity. Figure 3.4(a) shows an example of such a case. The data sets has two clusters C_1 and C_2, where C_1 is more dense than C_2. Point p for instance would have the same or an even higher LOF score when k is equal to 3 as point q. This is because the nearest neighbors of p all lie within cluster C_1 as shown in the figure. This is counter intuitive as point p actually lies within cluster C_2. The influence space overcomes that problem by taking more neighbors into account, namely the reverse k nearest neighbors set (RNN_k). $RNN_k(p)$ is the set of objects that has p in its k-neighborhood set. This is shown in figure 3.4(b) where s and t are the reverse neighbors of p. The definitions of RNN_k and IS are given below.



(a) K-neighborhood of points, k=3.

(b) Reverse k nearest neighbors, k=3.

Figure 3.4: 2D data set [8] that shows the motivation behind the introduction of INFLO.

Reverse k Nearest Neighbor set (RNN)

$$RNN_k(p) = \{q | q \in D \wedge p \in N_k(q)\}$$

Influence Space (IS)

$$IS_k(p) = RNN_k(p) \cup N_k(p)$$

The calculation of the INFLO score is performed as local density approaches by comparing the local density of tbe object to the average of its influence space. The formulas used are given below.

Local Density $den_k(p)$ is the inverse of its k-distance.

$$den_k(p) = \frac{1}{\text{k-distance}(p)}$$

Average density $den_{avg,k}(S)$ where $S \subseteq D$ is equal to the average local density over set S.

$$den_{avg,k}(S) = \frac{\sum_{i \in S} den_k(i)}{|S|}$$

Influenced Oultierness (INFLO)

$$INFLO_k(p) = \frac{den_{avg,k}(IS_k(p))}{den_k(p)}$$

Similar to the previous methods an INFLO score of 1 represents objects lying inside clusters, while an INFLO score greater than 1 identifies local density outliers.

3.1.6 LoOP: Local Oultier Probability

The algorithm was proposed in [9]. It works on the k-neighborhood of the object. LoOP is a local density based method that uses some statistical concepts to output the final score. This combines the advantages of both approaches. The local density based methods which do not assume the data is following any distribution and the sound mathematical reasoning of the statistical model.

The LoOP score represents the probability that a particular point is a local density outlier. This probability enables easy comparison of data points with the same data set as well as across different data sets.

In order to achieve that the LoOP is calculated as follows. The normalization factor is denoted by λ .

The standard distance $\sigma(o, N_k(o))$ is defined as the standard deviation of the distance around o.

$$\sigma(o, N_k(o)) = \sqrt{\frac{\sum_{s \in N_k(o)} d(o,s)^2}{|S|}}$$

Probabilistic set distance (pdist) $pdist(\lambda, o, N_k(o))$ defined as follows.

$$pdist(\lambda, o, N_k(o)) = \lambda \cdot \sigma(o, N_k(o))$$

Probabilistic Local Outlier Factor (PLOF) $PLOF_{\lambda, S(o)}(o)$ represents the ratio of the density estimation.

$$PLOF_{\lambda, N_k(o)}(o) = \frac{pdist(\lambda, o, N_k(o)) \cdot |N_k(o)|}{\sum_{s \in N_k(o)} pdist(\lambda, s, N_k(s))} - 1$$

The aggregate Probabilistic Local Outlier Factor (nPLOF) This is the scaling factor that makes the score independent from any distribution.

$$nPLOF(\lambda) = \lambda \cdot \sqrt{\frac{\sum_{o \in D} PLOF_{\lambda, N_k(o)}(o)^2}{|D|}}$$

Local Outlier Probability (LoOP)

$$LoOP_{N_k(o)}(o) = \max(0, erf(\frac{PLOF_{\lambda, S}(o)}{nPLOF \cdot \sqrt{2}}))$$

The approach is based on the following two assumptions:

1. That the point is the center of its neighborhood set.
2. The behavior of the distances simulates the behavior of the positive leg of the normal distribution.

The first assumption is violated particularly when the point is outlying. This violation will result in the overestimation of the PLOF as a result of the increase in the standard distance of the outlying point. This effect is in fact desirable because it will emphasize that the point is outlying.

The second assumption in contrast to the statistical methods makes the assumption only about the distribution of the distances and not the distribution of the points. This assumption holds for both Manhattan and euclidean distances according to the central limit theorem.

The addition of statistical concepts to the local density methods makes the LoOP scores independent from any distribution. This makes it capable of handling non-uniform clusters such as clusters generated by a Gaussian model which is handled poorly by LOF for instance.

3.1.7 LOCI: Local Correlation Integral

This algorithm was proposed in [11].The local correlation integral algorithm does not have any critical parameters such as the k in the algorithms discussed in the previous sections. This stems from the conviction of the original paper that the determination of such critical parameters should not be left for the end users to decide as they are probably domain experts and have limited knowledge about the materialization of the algorithms. This saves the users the troublesome task of trying different parameter setting to obtain the optimal output. This is done by considering all the different r values and taking the maximum score.

The local correlation integral algorithm flags the object as an outlier if the difference in the local density between it and its neighbors is considerable. To detect that the relative deviation of the local density from the average local neighborhood density is obtained; this deviation is called multi-granularity deviation factor ($MDEF$). $MDEF$ is calculated as follows.

Let α denote a real number between 0 and 1 and $n(p,r)$ denote the cardinality of the r-neighborhood.

Average number of neighbors

$$\hat{n}(p_i,r,\alpha) = \frac{\sum_{p \in N(p_i,r)} n(p,\alpha r)}{n(p_i,r)}$$

Standard deviation of the r-neighborhood

$$\sigma_{\hat{n}}(p_i,r,\alpha) = \sqrt{\frac{\sum_{p \in N(p_i,r)} (n(p,\alpha r) - \hat{n}(p_i,r,alpha))^2}{n(p_i,r)}}$$

Multi-granularity factor (MDEF)

$$MDEF(p_i,r,\alpha) = 1 - \frac{n(p_i,\alpha r)}{\hat{n}(p_i,r,\alpha)}$$

Standard deviation of MDEF

$$\sigma_{MDEF}(p_i,r,\alpha) = \frac{\sigma_{\hat{n}}(p_i,r,\alpha)}{\hat{n}(p_i,r,\alpha)}$$

Outlying points would have an MDEF value closer to 1. The object is flagged as an outlier if the ratio between $MDEF$ and σ_{MDEF} exceeds a constant proposed in the original publication to be 3. The score assigned to the points in our implementation corresponds to that ratio.

The algorithm uses two different neighborhoods the r-neighborhood and the αr-neighborhood. The earlier is called the *sampling* neighborhood over which the $n(p, \alpha r)$ is averaged and the latter is called the *counting* neighborhood. This is done in order to decouple the neighborhood size so that the MDEF of outlying points would be greater than zero as desired.

Even though the idea of having no critical parameters seems appealing it comes at the expense of the time and space complexity. The time complexity of the algorithm is $\mathrm{O}(n^3)$. This means that a 2D data set of 3000 items would take about 1 hour to run. The increase in the time complexity is due to the need to loop over each possible r and calculate the corresponding MDEF and σ_{MDEF}. The time complexity is not the main problem as usually unsupervised anomaly detection techniques operate in an off-line mode. The major problem is with the space complexity which is $\mathrm{O}(n^2)$ as we need to store the distances between each pair of points in the data set. A data set of size 6000 items would probably cause a memory exception which restricts the use of the algorithm.

3.2 Clustering Based

The process of bundling similar objects into clusters is referred to as clustering. Clustering based anomaly detection techniques operate on the output of clustering algorithms. They assume that anomalous points lie in sparse and small clusters, or that they are not assigned to any cluster, or that they lie far from their cluster centroid. The algorithms that were implemented use the output of any good clustering algorithm. The initial step followed by these algorithms is the division of the clusters into large and small clusters in a manner similar to what was proposed in [6].

The following sections have the following organization. First the method of division into small and large clusters is explained. Then the different approaches to the calculations of the anomaly score are described in the following sections.

3.2.1 Small and Large Clusters

Two different approaches were applied in order to make the division into small and large clusters. The first one is similar to what was applied in [6], while the second one is similar to what was applied in [1].

Method 1

In CBLOF [6] there are two parameters that distinguish small clusters from large clusters α and β. α is a real number between 0 and 1 and represents the ratio of the data set that is expected to be normal. β is a real number that represents the minimum ratio of the size of the large cluster to the small clusters.

Let $C = \{C_1, C_2, \dots, C_l\}$ be the set of clusters sorted by their size in decreasing order. Let LC be the set of large clusters and SC be the set of small clusters. We want to find an integer b such that $1 \leq b \leq l$ and $LC = \{C_1, \dots, C_b\}$ and $SC = C - LC$. According to [6] b should satisfy either of these conditions.

Condition 1

$$\sum_{i=1}^{b} |C_i| \geq |D| \cdot \alpha$$

Condition 2

$$b \neq l \land \frac{|C_b|}{|C_{b+1}|} \geq \beta$$

The intuition behind the first condition is that majority of the data set should be normal and thus considered as large clusters, while the second condition can be explained by that the fact that the if the smallest large cluster is β times larger than the largest small cluster, then there is a considerable difference in the size of the two sets.

Method 2

The second method proposed in [1] uses only 1 parameter γ. γ specifies the percentage of the average number of elements in the clusters that would represent the minimum number of elements in the large cluster.

Condition for small clusters

$$cluster\ size < \gamma \cdot \frac{number\ of\ records}{number\ of\ clusters}$$

3.2.2 CBLOF: Cluster-Based Local Outlier Factor

$$CBLOF(p) = \begin{cases} |C_i| * min(d(p, C_j))\ where\ p \in C_i, C_i \in SC\ and\ C_j \in LC \\ |C_i| * d(p, C_i)\ where\ p \in C_i\ and\ C_i \in LC \end{cases} \tag{3.2}$$

The formula simply states that anomaly score is equal to the distance to the nearest large cluster multiplied by the size of the cluster the object belong to. Figure 3.5 illustrates this concept. Point P lies in the small cluster C_2 and thus the score would be equal to the distance to C_1 which is the nearest large cluster multiplied by 5 which is the size of C_2.

The authors of [6] claim that weighting by the size of the cluster makes this method local. However, it is our believe that this does not make the method local as the score is not normalized relative to the neighborhood.

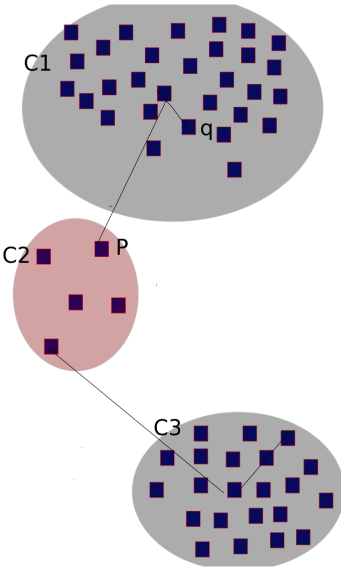


Figure 3.5: An illustration of the distance to the nearest large cluster. In this example C_1 and C_3 were identified as large clusters, while C_2 was identified as a small cluster.

3.2.3 Unweighted-CBLOF: Unweighted Cluster-Based Local Outlier Factor

Upon the conduction of the initial experiments, it has been observed that equation 3.2 could lead to misleading results. Figure 3.6 shows an example of such a case. The blue cluster is the only small cluster in this synthetic data set. We have two points A and B, while it is obvious that A is more outlying than B, the score assignment shows otherwise. This can be explained by the fact that point B is multiplied by the size of the green cluster which is much larger than the size of the blue cluster. The example shows that outlying points belonging to large clusters are discriminated against. Hence I propose a modified algorithm as shown in equation 3.3.

$$unweighted - CBLOF(p) = \begin{cases} min(d(p, C_j)) \, where \, p \in C_i, C_i \in SC \, and \, C_j \in LC \\ d(p, C_i) \, where \, p \in C_i \, and \, C_i \in LC \end{cases} \tag{3.3}$$

This is similar to equation 3.2 except that the weighting factor which was the size of the cluster is removed.

Figure 3.7 shows the results of the proposed method 3.3. As expected point A has a higher outlier score than point B. What should be also noted is that even the points deep inside the blue cluster have a high outlier score as we consider the small clusters outlying.

3.2.4 LDCOF: Local Density Cluster-Based Outlier Factor

Local density based methods are popular as the anomaly score is normalized relative to the neighborhood. Moreover the anomaly score has a natural threshold that would

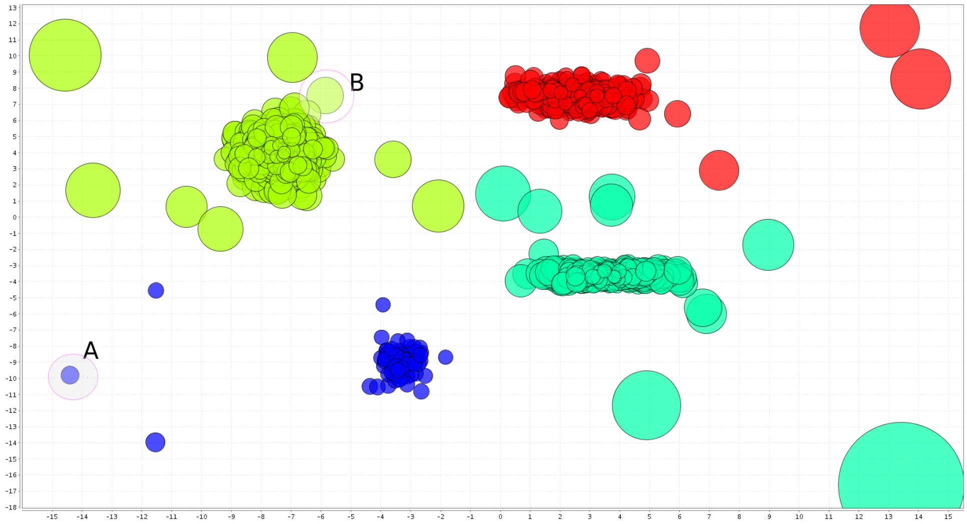


Figure 3.6: Example of the results of CBLOF algorithm on synthetic data set. The size of the point indicates the outlier score. The color indicates the cluster to which the point belongs to.

indicate whether the points are outlying or not as outlying points are expected to have an anomaly score which is greater than 1. In attempt to apply the local density based method to the output of clustering algorithms, the following was proposed.

Average cluster distance

$$distance_{avg}(C) = \frac{\sum_{i \in C} d(i, C)}{|C|}$$

Local Density Clustering Outlier Factor (LDCOF)

$$LDCOF(p) = \begin{cases} \frac{min(d(p,C_j))}{distance_{avg}(C_j)} \; where \; p \in C_i, C_i \in SC \; and \; C_j \in LC \\ \frac{d(p,C_i)}{distance_{avg}(C_i)} \; where \; p \in C_i \; and \; C_i \in LC \end{cases}$$

LDCOF score is defined as the distance to the nearest large cluster as illustrated in figure 3.5 divided by the average cluster of the large cluster.The intuition behind this is that since small clusters are considered outlying, the elements inside the small clusters are assigned to the nearest large cluster which becomes its local neighborhood. Thus the anomaly score is computed relative to that neighborhood.

3.2.5 Recommended clustering algorithms

The algorithms discussed above work on the output of any good clustering algorithm, however some algorithms worked better than others. Algorithms that take as a parameter the number of clusters such as k-means and k-medoids worked better than algorithms

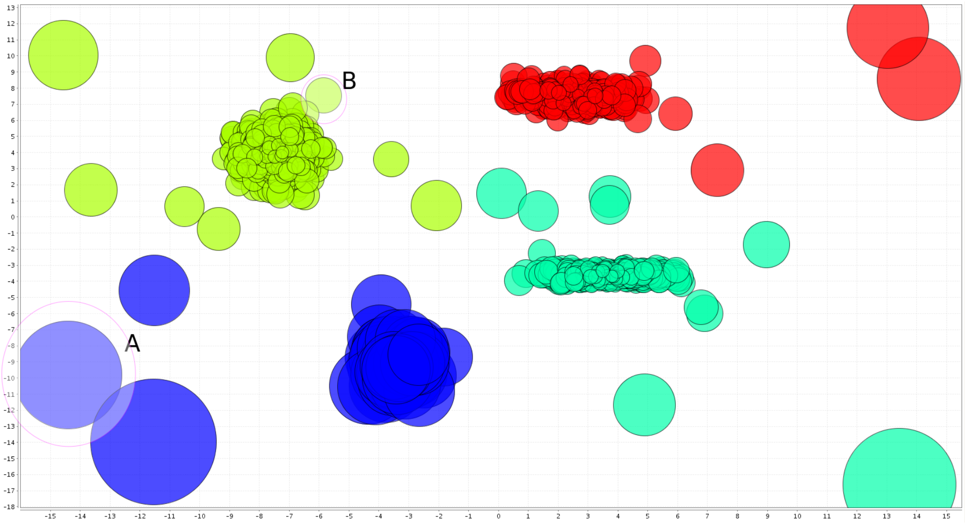


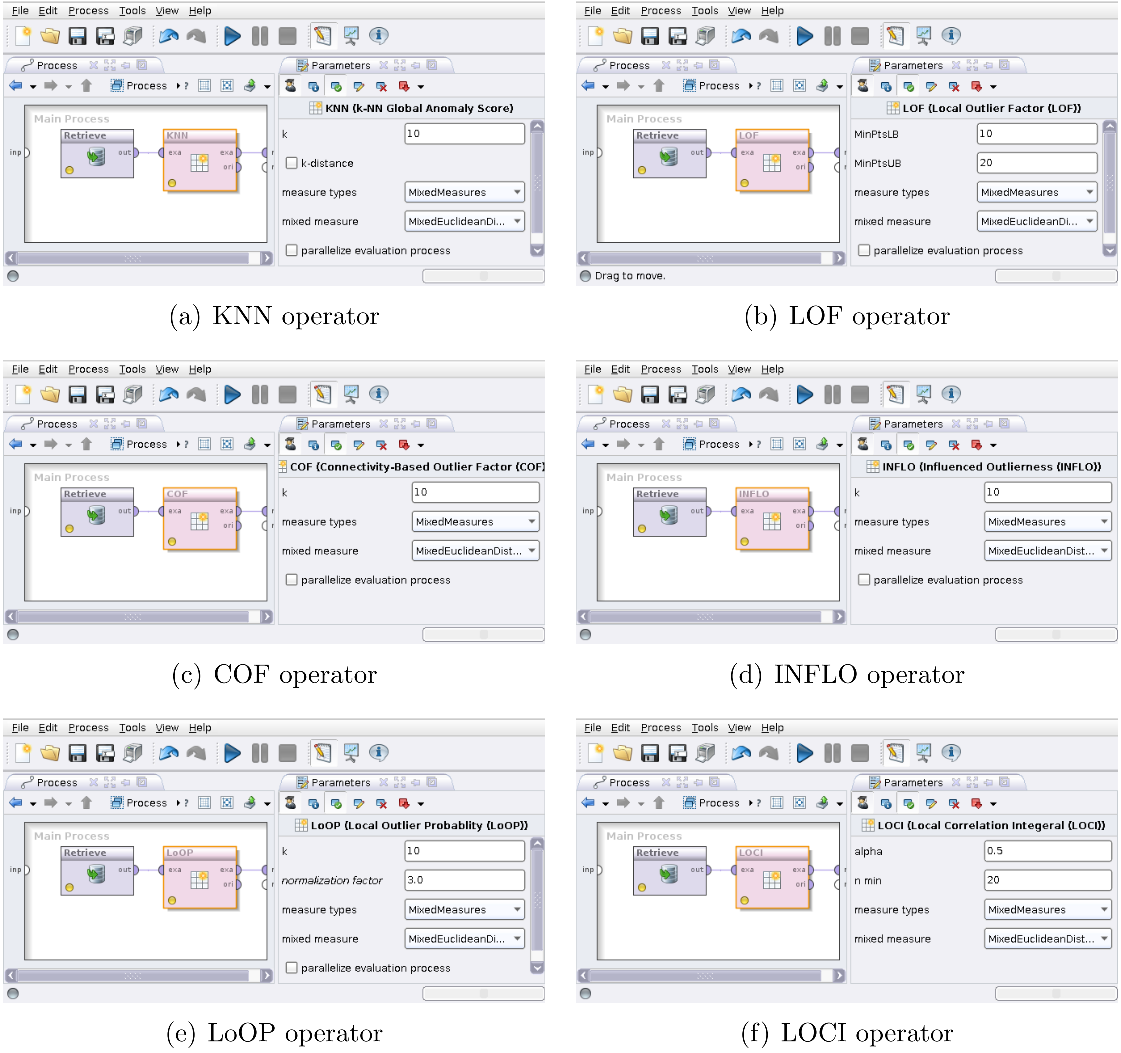
Figure 3.7: Example of the results of unweighted CBLOF algorithm on synthetic data set. The size of the point indicates the outlier score. The color indicates the cluster to which the point belongs to.

that determine the number of clusters such as X-means particularly when the number of clusters is overestimated. This is mainly because of two reasons; that the algorithms use the distance to the cluster centroid and that they operate on the assumption that small clusters are outlying clusters. Overestimating the number of clusters causes the division of the large clusters into smaller clusters each with its own centroid. This is particularly useful in case of Gaussian distribution for instance where the distribution of the points is not spherical and without that division points at the peripherals could be falsely identified as outliers. The second reason is that the small clusters are considered as outlying, this is ensured by overestimating the number of clusters.

3.3 Operators in RapidMiner

Figure 3.8 shows the nearest-neighbor based operators, while figure 3.9 shows the clustering based operators. The nearest-neighbor based operators takes as an input an example set and outputs the example set with the computed outlier attribute as well as the original example set. The clustering based algorithms have two inputs the cluster model and the clustered set. The cluster model contains the record to cluster mapping of the clustered set. It is important to feed the operator the clustered set with its corresponding cluster model for the correct computation. The operators output the cluster model, the clustered set with the computed outlier attribute as well as the original clustered set. All the input of the operators were passed through to enable them to be fed to other operators for more complex processes.

As shown in the parameter tabs in both figures 3.8 and 3.9 each algorithm has two parameter *measure types* and the *measure*. These parameters allow the user to control which distance function is used in order to compute distances between instances. We



KNN (k-NN Global Anomaly Score)

k: 10

☐ k-distance

measure types: MixedMeasures

mixed measure: MixedEuclideanDi...

☐ parallelize evaluation process

(a) KNN operator

LOF (Local Outlier Factor (LOF))

MinPtsLB: 10

MinPtsUB: 20

measure types: MixedMeasures

mixed measure: MixedEuclideanDi...

☐ parallelize evaluation process

Drag to move.

(b) LOF operator

COF (Connectivity-Based Outlier Factor (COF)

k: 10

measure types: MixedMeasures

mixed measure: MixedEuclideanDist...

☐ parallelize evaluation process

(c) COF operator

INFLO (Influenced Outlierness (INFLO))

k: 10

measure types: MixedMeasures

mixed measure: MixedEuclideanDist...

☐ parallelize evaluation process

(d) INFLO operator

LoOP (Local Outlier Probablity (LoOP))

k: 10

normalization factor: 3.0

measure types: MixedMeasures

mixed measure: MixedEuclideanDi...

☐ parallelize evaluation process

(e) LoOP operator

LOCI (Local Correlation Integral (LOCI))

alpha: 0.5

n min: 20

measure types: MixedMeasures

mixed measure: MixedEuclideanDist...

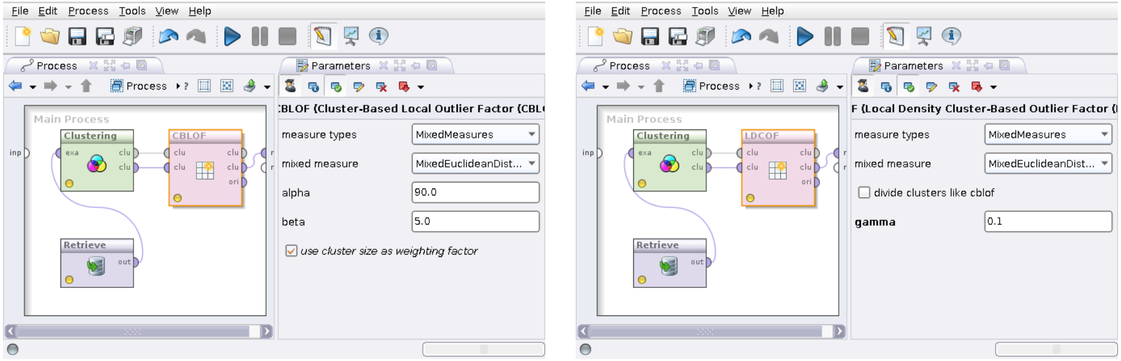
(f) LOCI operator

Figure 3.8: The nearest-neighbor based operators in RapidMiner. On the right hand side of each figure, the parameter tab is shown with the default parameter settings.

used the already defined distance measures in RapidMiner.

The operators in figures 3.8 from (a) to (e) all use the k-neighborhood and thus they have some similar parameters. First, there is the parameter k which is the size of the neighborhood. The LOF operator in figure 3.8(b) has two parameters instead of parameter k, *MinPtsLB* and *MinPtsUB* which correspond to lower and upper bound of the size of the neighborhood as explained in section 3.1.3. They also have the parameter *parallelize evaluation process* which allows the user to have control on whether the algorithm should run in parallel or not. In case the user chooses to run the evaluation process in parallel he will be allowed to specify the number of threads as shown in figure 3.10.

The nearest-neighbor based operators also have individual algorithm related parameters. KNN has the parameter *k-distance* that specifies whether the anomaly score should be set to the k-distance instead of the default average distance over the neighborhood set. LoOP 3.8(e) has the parameter *normalization factor* which corresponds to λ in the equations described in section 3.1.6. LOCI has the parameter *alpha* which determines the size of



BLOF (Cluster-Based Local Outlier Factor (CBL

measure types	MixedMeasures
mixed measure	MixedEuclideanDist...
alpha	90.0
beta	5.0

☑ use cluster size as weighting factor

F (Local Density Cluster-Based Outlier Factor (

measure types	MixedMeasures
mixed measure	MixedEuclideanDist...

□ divide clusters like cblof

gamma	0.1

(a) CBLOF operator

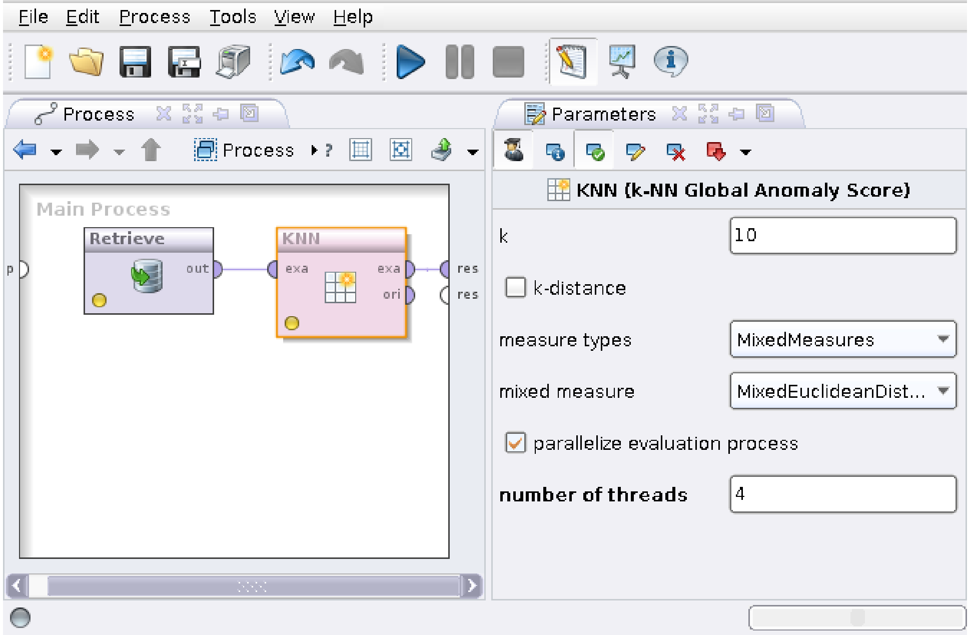
(b) LDCOF operator

Figure 3.9: The clustering based operators in RapidMiner.On the right side of each figure, the parameter tab is shown with the default parameter settings.

the counting neighborhood and *n min* which specifies the minimum number of neighbors in the sampling neighborhood used in order to smooth the statistical fluctuations that exist for small neighborhood sizes.

The CBLOF operator shown in figure 3.9(a) has the two parameters *alpha* and *beta* that controls the division into small and large clusters as described in section 3.2.1. It also has the parameter *use cluster size as weighting factor* that determines whether the anomaly score is calculated similar to CBLOF or similar to unweighted-CBLOF.

The LDCOF operator shown in figure 3.9(b) has two different parameters. Parameter *divide clusters like cblof* which controls whether the division into small and large clusters should be performed like method 1 or method 2 described in section 3.2.1. Parameter *gamma* which correspond to the parameter described in method 2. If the first parameter is set to true, the user will be allowed to set the values of *alpha* and *beta* the parameters of method 1.



KNN (k-NN Global Anomaly Score)

k 10

☐ k-distance

measure types MixedMeasures

mixed measure MixedEuclideanDist...

☑ parallelize evaluation process

number of threads 4

Figure 3.10: KNN operator with the process set to run in parallel. The user is allowed to specify the number of threads

Chapter 4

Implementation

In this chapter we will take a look at the implementation of the algorithms discussed in the previous section, as well as the optimizations performed.

4.1 Architecture

The extension has mainly two modules, the operator and the evaluator module. The operator module contains the RapidMiner operator of each algorithm. Each operator defines the parameters of the algorithm, then at run time it retrieves those parameters and perform any necessary processing on the example set. The operator module is the one that deals with the RapidMiner logic . The second module is the evaluator module which contains the logic of the algorithms.

The architecture of the operator module is shown in figure 4.1. All operators are subclasses *AbstractAnomalyDetectionOperator*. The method *doWork()* is over loaded by the method *doWork(exampleSet: ExampleSet , attributes: Attributes , points: double [][])*. The earlier initializes the outlier attribute and does any necessary preprocessing, the latter is the one that does the actual work and it should be overridden by the subclasses. *AbstractNearestNeighborAnomalyDetectionOperator* is the super class of all nearest-neighbor based algorithms. Nearest-neighbor based operators should override *doWork(exampleSet: ExampleSet, attributes: Attributes , points : double[][], weight: int[])* in order to handle duplicates as explained in section 4.2.1. *AbstractClusteringBasedAnomalyDetectionOperator* is the super class of the clustering based algorithms.

The architechture of the evaluator module is shown in figure 4.1. All evaluators implement the interface *Evaluator* which has the method *evaluate()* which is where the algorithms are executed. The algorithms that uses the k-neighborhood extends *KNNEvaluator*. *KNNThread* and *KNNThreadSynchronized* are the threads that parallelize the code as described in section 4.2.2. *LDCOFEvaluator* uses the method *CBLOFEvaluator.assignLargeClusters* in case the division into large and small clusters was implemented similar to CBLOF.

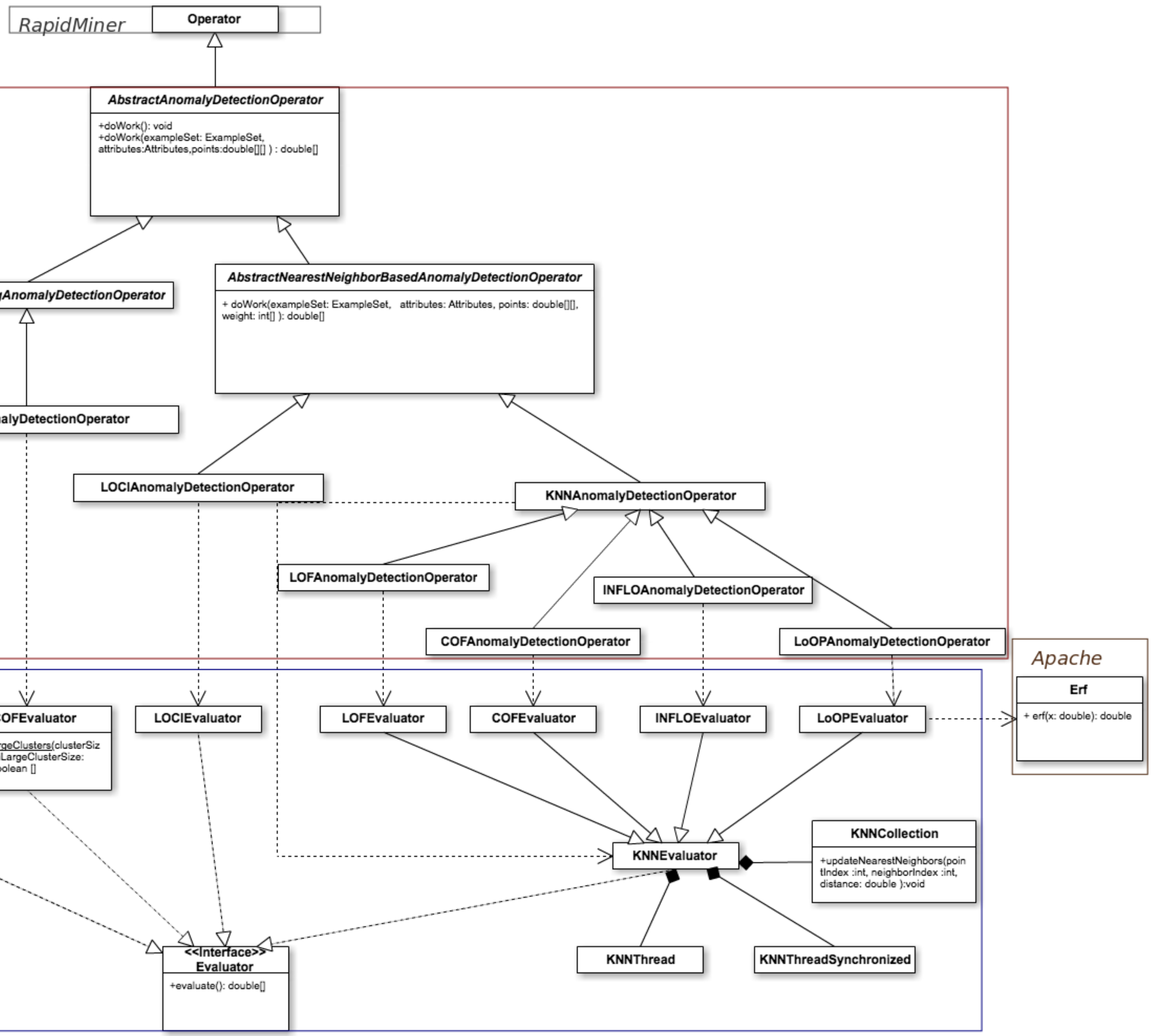

RapidMiner
Operator
Operator
AbstractAnomalyDetectionOperator
+doWork(): void
+doWork(exampleSet: ExampleSet, attributes:Attributes,points:double[][]) : double[]
AbstractClusteringAnomalyDetectionOperator
AbstractNearestNeighborBasedAnomalyDetectionOperator
+ doWork(exampleSet: ExampleSet, attributes: Attributes, points: double[][], weight: int[]): double[]
LDCOFAnomalyDetectionOperator
CBLOFAnomalyDetectionOperator
LOCIAnomalyDetectionOperator
KNNAnomalyDetectionOperator
LOFAnomalyDetectionOperator
INFLOAnomalyDetectionOperator
COFAnomalyDetectionOperator
LoOPAnomalyDetectionOperator
Apache
Erf
+ erf(x: double): double
Evaluator
CBLOFEvaluator
+assignLargeClusters(clusterSize: int[], alpha: double, beta :double): boolean []
LDCOFEvaluator
+assignLargeClusters(clusterSize: int[], minLargeClusterSize: double): boolean []
LOCIEvaluator
LOFEvaluator
COFEvaluator
INFLOEvaluator
LoOPEvaluator
KNNEvaluator
KNNCollection
+updateNearestNeighbors(pointIndex :int, neighborIndex :int, distance: double):void
KNNThread
KNNThreadSynchronized
<<Interface>>
Evaluator
+evaluate(): double[]

Figure 4.1: Class Diagram for *Anomaly Detection* extension

4.2 Optimizations

4.2.1 Handling Duplicates

In application domains such as in network intrusion detection there exists many duplicates in the data set. Figure 4.2 shows the pipeline that handles duplicates. The original data set is processed removing all the duplicates and assigning to each record in the new data set a corresponding weighting factor which is equal to the number of records with the same coordinates in the original data set. The algorithms operate on the new data set and finally the scores are mapped producing the result set.

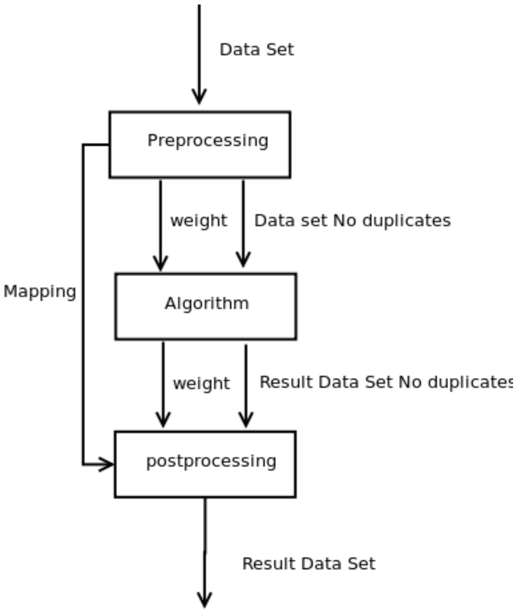


Figure 4.2: Illustration of the processing performed to handle duplicates

This procedure has two advantages. First it can significantly reduce the number of records which would positively reflect on the performance. Second it facilitates the handling of duplicates for local density based methods as explained in section 3.1.1.

4.2.2 Parallelization

The main bottle neck for the nearest-neighbor based algorithms is the need to compute the distance between each pair of points in the data set. This is typically done in n^2 operations, however it can be done in $\frac{(n-1)\cdot n}{2}$ if we take into account that the distance functions are symmetric. This means that the distance of p_1 to p_2 is the same as the distance of p_2 to p_1.

In order to speed up the algorithms the code was parallelized for computing the k nearest neighbors. This can be done by one of the following methods. First we can compute the n^2 distances and avoid the need for any synchronization. The second option would be to compute only $n^2/2$ distances, this would need some kind of synchronization because two threads could attempt to update the nearest neighbors of the same point at the same time. For synchronization, both, Java ReentrantLock and synchronized blocks were evaluated. Using synchronized blocks was faster that using ReentrantLock, thus ReentrantLock was dismissed from further consideration.

The comparison between the two proposed methods is not straight forward. This is due to the trade off is between the time it takes to compute the distance function and the waiting time of the threads. The waiting time is affected by the update of the nearest neighbors, which has a complexity of O(k) in the worst case and O(1) in the best case. The complexity of the update method is determined by the order of computation where if we initially found the right k nearest neighbors then the further calls of the update method would be in constant time.

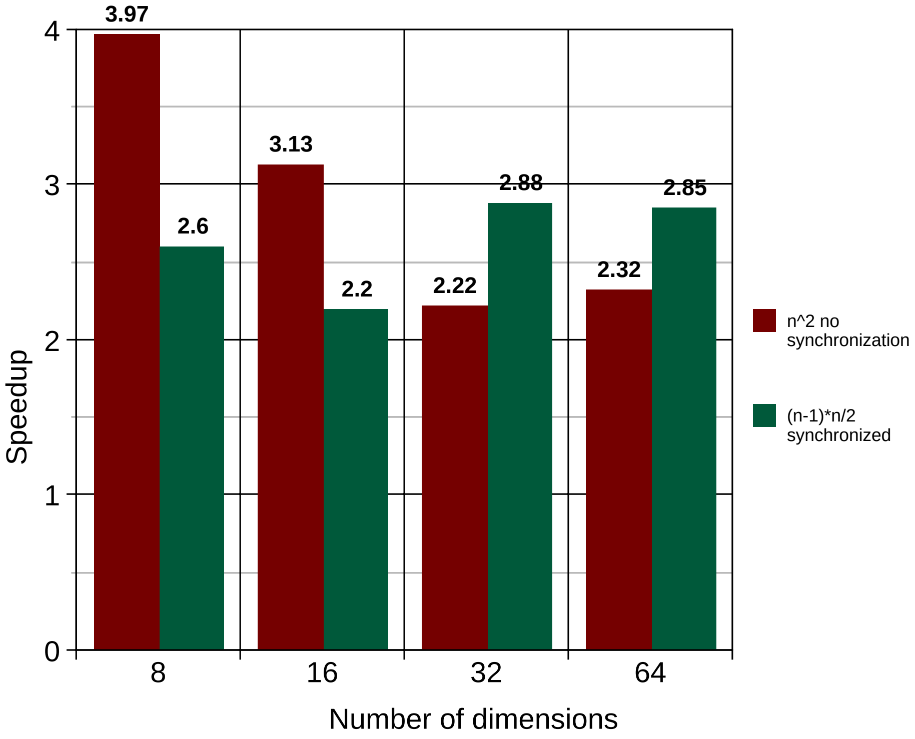


Figure 4.3: The speed up of the two proposed methods using 4 cores. The first one is avoiding synchronization and the second is using synchronized blocks for synchronization.

Figure 4.3 shows the results of running the two approaches on 100000 randomly generated data instances with varying the dimensionality using 4 threads. Since the order of the data set affects the execution time we ran the algorithms three times on the shuffled data set and the average time was taken.

As observed in figure 4.3 for smaller dimensions avoiding synchronization is better while for larger dimensions computing less distances is better. This is because for higher dimensions the cost of computing the distances becomes more expensive than the waiting time of the threads. We defined a threshold for the dimensionality of the data set. This threshold was set to 32. Data sets having lower dimensionality will be parallelized without synchronization, while synchronization would be used for data set with higher dimensionality.

Chapter 5

Experiments

Experiments were performed on real world data sets to compare the performance of different algorithms. The second objective was to verify that the algorithms are correct by running the algorithm on data sets that were previously used in the literature.

The real world data sets used are all from the UCI machine learning repository[5]. Due to the limited availability of data sets suitable for unsupervised learning, the data sets used are all intended for classification tasks. Since these data sets usually have a comparable number of records in each class, preprocessing is required in order to make them compatible with the unsupervised anomaly detection assumption.

5.1 Evaluation of Anomaly Detection Algorithms

The standard way of comparing the performance of anomaly detection techniques is using the receiver operating characteristic (ROC) curve and the area under that curve (AUC ROC).

The ROC is a graphical plot of the true positive rate against the false positive rate(see definitions below). It clarifies the trade off between having a higher number of correctly classified records and incurring the cost of having more records falsely predicted. It is applicable when the data can be categorized into two classes a positive class and a negative class. In our case the earlier corresponds to the outlier class and the later to the normal class. The boundary between the classes is determined by a continuous discrimination factor that is varied in order to produce the curve. The discrimination factor is equivalent to the outlier score in the output of the implemented anomaly detection algorithms.

True Positive Rate The ratio between the number of items that are correctly assigned to the positive class and the total number of items in the positive class.

False Positive Rate The ratio between the number of items that are falsely assigned to the positive class and the total number of items in the negative class.

There are several characteristics of the ROC curve. An example ROC curve is shown in figure 5.1. The ideal algorithm would produce a line passing through the perfect classification point. The perfect classification point indicates that there exists a value for the outlier score which distinctly separates the outlier class from the normal class. A completely random algorithm would produce the random guessing line. If the curve lies below that line, then the algorithm performs opposite to what is expected.

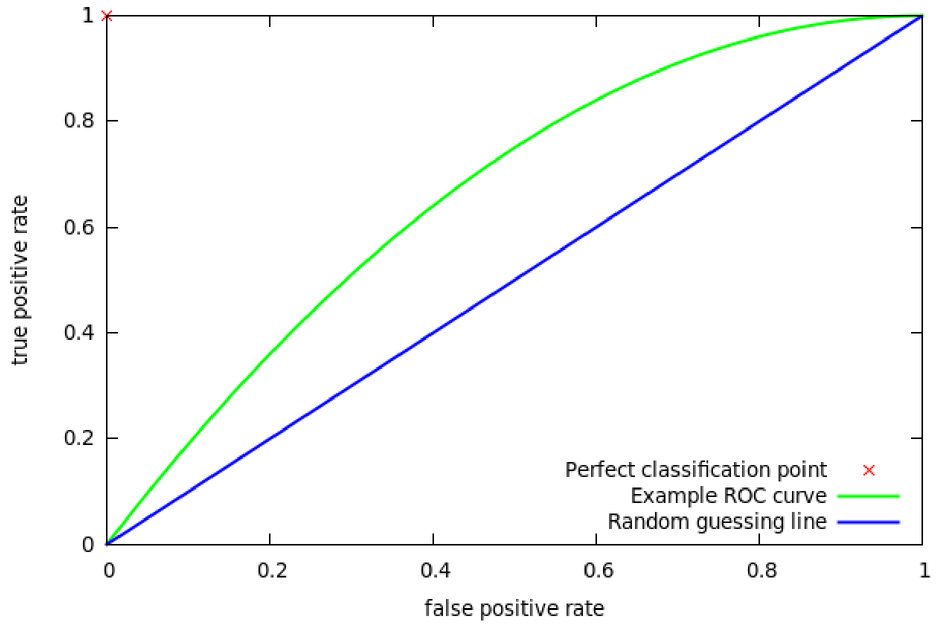


Figure 5.1: Roc curve example

The Area under the ROC curve approximates the probability that an anomalous point would have higher outlier score than a normal point. This is used in the anomaly detection problem in order to be able to find the optimal parameter choice as well as compare the performance of different algorithms whilst varying similar parameters.

The ROC curve for the anomaly detection problem is obtained as follows. The points are sorted according to the outlier score in non-increasing order. Then as the outlier score is varied the corresponding rates are calculated.

5.2 Data Sets

5.2.1 Breast Cancer Wisconsin (Diagnostic)

The data contains 569 records. Of which 357 are benign and 212 are malignant. Each record has 30 real-valued attributes. The preprocessing applied is similar to that performed in [9] where the first 10 malignant records are kept.

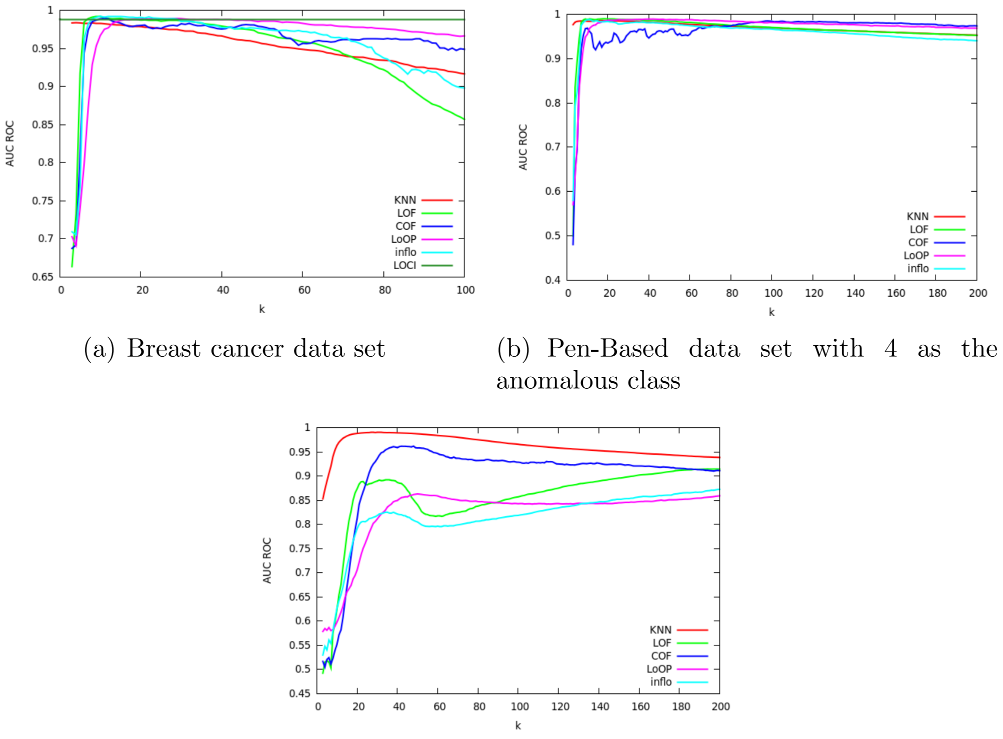
5.2.2 Pen-Based Recognition of Handwritten Text

The data set consists of digits written by 45 different writers. The feature vector of each digit has 16 attributes and there is a total of 7494 entries[1].

Two different preprocessing steps were applied on this data set. The first method is similar to that performed in [9] where digit 4 was chosen to be the anomalous class and only the first 10 records were kept in the data set, resulting in a data set of size 6724. In the second method digit 8 was chosen to be the normal class and thus all the remaining classes were sampled keeping only the first 10 digits of each class.The resulting data set was of size 809.

5.3 Results and Discussion

5.3.1 Nearest-Neighbor Based



(a) Breast cancer data set

(b) Pen-Based data set with 4 as the anomalous class

(c) Pen-Based data set with 8 as the normal class

Figure 5.2: AUC ROC for nearest-neighbor based algorithms

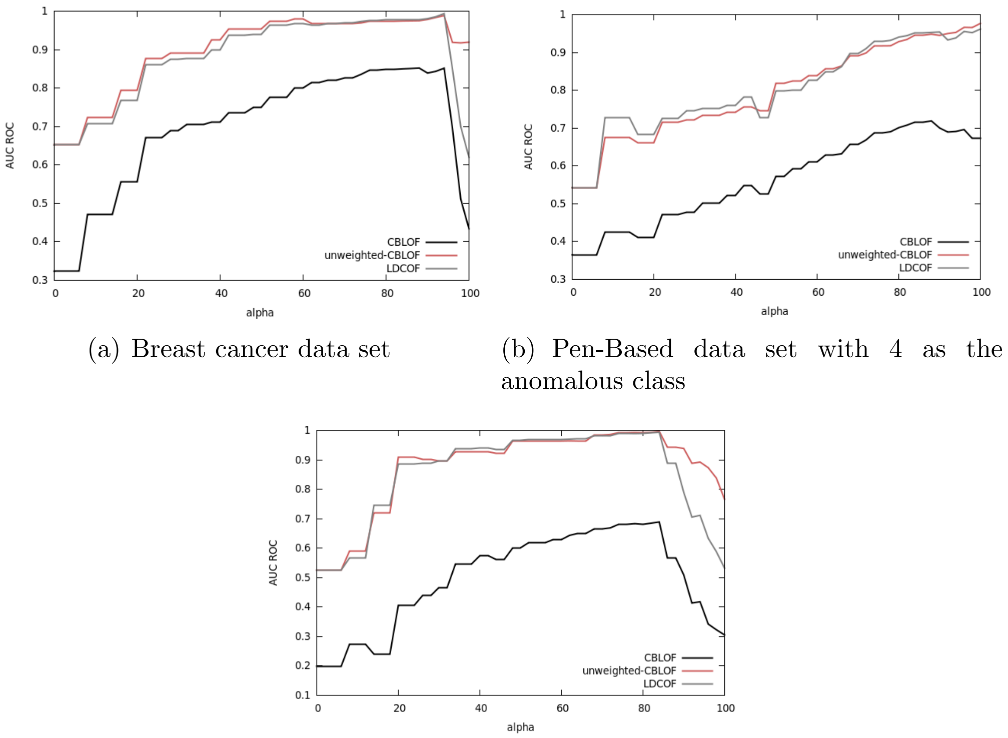
Figure 5.2 shows the AUC ROC of the algorithms on the different data sets. In figures 5.2(a) and 5.2(b) the local density based methods are superior to the KNN which is a global method. the performance of LOF and INFLO are very similar performing better at lower values of k ,while for higher values of k LoOP performs best. In figure 5.2(a)

[1] Only the training data set was used

LOCI was also plotted even though it does not vary with k. This was done in order to show the power of LOCI as given the inappropriate choice of k such as at k equal 40, LOCI performs best with it's default parameter setting. In figure 5.2(c) KNN which gives a global score performs best, followed by COF.

The results shown in figure 5.2(c) for second preprocessing of the pen-based data set can be explained by that the normal class probably form a sparse cluster which makes the distinction between the outlying normal records and the actual outliers increasingly difficult. Thus the global method performs best. This is further supported by the fact that COF out performs the other local density based algorithms. However in the remaining data sets there is probably more variation in the densities and thus local methods are superior.

5.3.2 Clustering Based



(a) Breast cancer data set

(b) Pen-Based data set with 4 as the anomalous class

(c) Pen-Based data set with 8 as the normal class

Figure 5.3: AUC ROC for clustering based algorithms

There are many factors that would affect the output of the clustering based algorithms. One of which is the division into large and small clusters. Thus in figure 5.3 we plotted the effect of varying α on the AUC ROC which in term affect the division into large and small clusters. The algorithms operated on the output of K-Means clustering algorithm with k set to 30. β was set to infinity such that the division would only be affected by the choice of α.

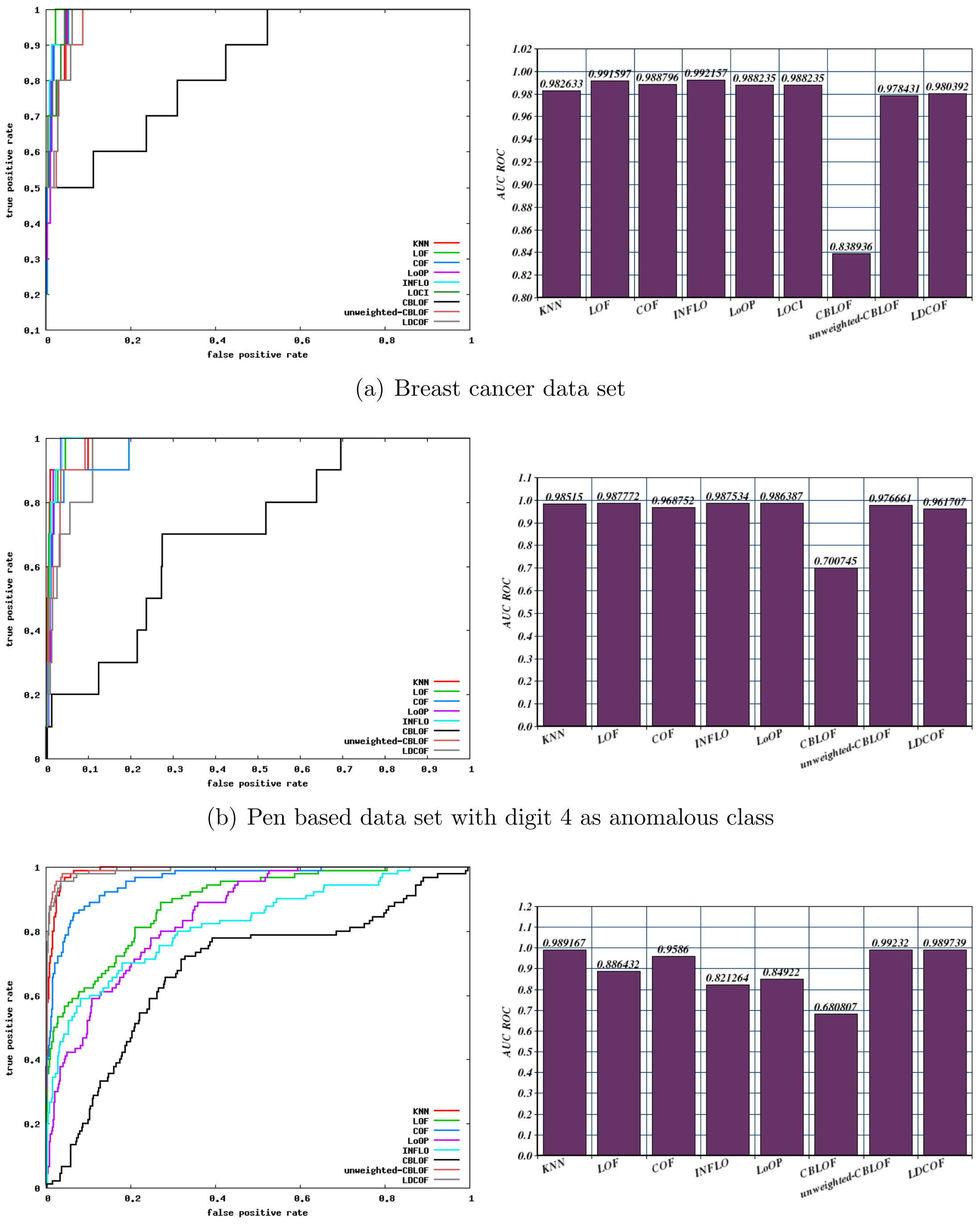
Figures 5.3(a), 5.3(b) and 5.3(c) all show that both unweighted-CBLOF and LDCOF are superior to CBLOF. Figures 5.3(a) and 5.3(c) show that the division into large and small clusters are better than having no division which is at $\alpha = 100$. This might be explained by the size of those data sets which is much smaller than the data set of the pen digits having 4 as the anomalous class. It is believed that all the implemented algorithms produce better results when the number of clusters(k for the K-means algorithm) is over estimated and thus 30 for a data set of size 6724 is not the same as 30 for smaller data sets of size 569 and 809.

5.3.3 Nearest-Neighbor based vs. Clustering based

The comparison between nearest-neighbor and clustering based algorithms can be performed by examining the ROC curve of the algorithms which is shown in figure 5.4.

The optimal parameter values were set for each algorithm in order to compare the best case performance of the algorithms. This was obtained from the results of the AUC ROC curves in the previous sections. Figure 5.4(a) shows the ROC curves for the breast cancer data set. All the nearest-neighbor based algorithms used k equal to 10 except LoOP where k was set to 20. The default parameter settings were used for LOCI. The parameters α and β of the clustering based algorithms were set to 90 and 5 respectively. The results of the pen based data set using 4 as the anomalous are shown in figure 5.4(b). The parameter setting of the nearest-neighbor based algorithms was similar to the one used in breast cancer data set, where parameter k in all but LoOP was set to 10, whilst in LoOP it was set to 20. The parameters α and β were set to 90 and 5 respectively for CBLOF and 100 and ∞ for unweighted-CBLOF and LDCOF. This is mainly because as observed for this data set the results were best for the latter algorithms when the clusters were not divided into small and large clusters as shown in figure 5.3(b). The last data set shown in figure 5.4(c) which is the pen based data set with 8 as the normal class used k equal to 40 for nearest-neighbor based algorithms and α and β equal to 80 and 5 for clustering based algorithms.

The performance of the nearest-neighbor algorithms and the clustering algorithms are comparable. Namely for the unweighted-CBLOF and LDCOF which performed slightly worse than the best nearest-neighbor based algorithms on the first two data sets; the breast cancer data set and the pen based data set with digit 4 as the anomalous class. The same algorithms slightly outperformed the best nearest-neighbor based algorithms on the last data set the pen based data set with digit 8 as the normal class. CBLOF performed worst on all data sets.



(a) Breast cancer data set

(b) Pen based data set with digit 4 as anomalous class

(c) Pen based data set with digit 8 as the normal class.

Figure 5.4: ROC curves for the data sets and their corresponding AUC ROC values.

Chapter 6

Conclusion and Outlook

A RapidMiner extension *Anomaly Detection* was implemented that contains the most well-known unsupervised anomaly detection algorithms. This extension will enable analysts to easily use those algorithms and integrate the operators into more complex processes. Also, the extension will enable researchers to perform further comparisons on the algorithms which would be beneficial as there is a general scarcity of those comparisons in the literature.

Unweighted-CBLOF and LDCOF were introduced in this work. The initial results look promising as they both out performed CBLOF in the experiments. The observed behavior of both algorithms is very similar. However the scores of LDCOF are more readily interpretable. This is because like LOF the normal records would have an anomaly score of approximately 1 whilst records having a score greater than 1 would be possible outliers. Further experiments needs to be carried in order to verify the results.

The experiments implemented for the evaluation and comparison of the algorithms are not sufficient. This was mainly due to the lack of time as well the limited number of data sets suited for anomaly detection. The second objective of the thesis turned out to be an interesting research area, which requires further research.

6.1 Recommendations

The experience of working with the algorithms has enabled me to derive a few pointers that might be useful especially for beginners. For small data sets LOCI algorithm would be the best choice as it can be ran with its default setting without having to worry about the appropriate parameter values. Also for small data sets usually global methods perform better than local methods as the data set does not contain enough clusters with varying densities. The parameter k should be given a larger value in case of LoOP than in case of LOF and INFLO. For large data sets clustering based algorithms would work best as they are more efficient only computing distances between points to the cluster centroids. Also the results of the experiments showed that the optimal parameter setting of α is usually around 90 which is the default setting, making the parameter choice easier than nearest-neighbor based methods. The choice between whether to use local or global approaches are data set dependent, both should be tried to determine the best option. This can be determined by examining the top most outliers.

6.2 Outlook

Comparison of the algorithms on synthetic data sets is necessary. This is because real data sets usually have high dimensions which makes it harder to visualize the data set as to fully understand the characteristics of the data set in order to better explain the behavior of the algorithms. This is especially true for local density based methods as the motivation of each one is very specific.

There are a lot of factors that would affect the clustering based methods implemented in the Anomaly Detection extension. There is the clustering algorithm used, the parameter settings of this algorithm in addition to the method employed in order to divide the clusters into small and large clusters. Due to the limited amount of time we were not able to investigate the effects of these factors. It is possible that CBLOF would perform better with a clustering algorithm other than k-means. More effort should be put in the investigation of those factors to verify the initial results obtained.

LOCI algorithm has a nearly linear approximation which is called *approximate local correlation integral (aLOCI)*. We tried to implement this algorithm however the choice of some of the parameters described in [11] were vague. aLOCI is especially appealing as it combines the advantages of LOCI avoiding its limitations. Adding the algorithm to the extension would be of great value.

Currently the extension works on static data sets. Nearest-neighbor based algorithms that uses the k-neighborhood could be modified to handle data in an on-line fashion. This can be achieved by saving the data structure used to store the nearest neighbors. An incoming record would only need to be inserted in this data structure and the corresponding anomaly score calculated which is linear in time.

Bibliography

[1] Moh'd Belal Al-Zoubi. An Effective Clustering-Based Approach for Outlier Detection. *European J. Scientific Research*, 28(2):310–316, 2009.

[2] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Principles of Data Mining and Knowledge Discovery*, volume 2431 of *Lecture Notes in Computer Science*, pages 43–78. Springer Berlin / Heidelberg, 2002.

[3] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jrg Sander. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104, Dallas, Texas, USA, May 2000. ACM.

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. Technical report, University of Minnesota, 2007.

[5] A. Frank and A. Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2010.

[6] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641 – 1650, 2003.

[7] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85–126, 2004. 10.1023/B:AIRE.0000045502.10941.a9.

[8] Wen Jin, Anthony Tung, Jiawei Han, and Wei Wang. Ranking outliers using symmetric neighborhood relationship. In Wee-Keong Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang, editors, *Advances in Knowledge Discovery and Data Mining*, volume 3918 of *Lecture Notes in Computer Science*, pages 577–593. Springer Berlin / Heidelberg, 2006.

[9] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Loop: local outlier probabilities. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1649–1652, New York, NY, USA, 2009. ACM.

[10] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale (now: Rapidminer): Rapid prototyping for complex data mining tasks. In *Proceeding of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 2006, 2006.

[11] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. *Data Engineering, International Conference on*, 0:315, 2003.

[12] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages "427–438", New York, NY, USA, 2000. ACM.

[13] Jian Tang, Zhixiang Chen, Ada Fu, and David Cheung. Enhancing effectiveness of outlier detections for low density patterns. In Ming-Syan Chen, Philip Yu, and Bing Liu, editors, *Advances in Knowledge Discovery and Data Mining*, volume 2336 of *Lecture Notes in Computer Science*, pages 535–548. Springer Berlin / Heidelberg, 2002.