INDERDEPARNTMENTAL PROGRAMME OF POSTGRADUATE STUDIES IN INFORMATION SYSTEMS

Master thesis

DATA MINING IN COMPUTER NETWORK DATA: INTRUSION DETECTION SYSTEMS

KATSAVELIS ZISIS

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του μεταπτυχιακού διπλώματος ειδίκευσης στα Πληροφοριακά Συστήματα

September 2014

Abstract

In this paper we outline the basic points of the fast growing field of data mining (or as it also known knowledge discovery from data) emphasizing on one critical subdomain known as outlier detection or anomaly detection. We examine this particular area under the prospective of the computer networks security. Anomaly detection is the identification of rare, abnormal observations which contain valuable information, and for that reason they are the main objects of interest in the specific problem domain.

At the first parts we analyze the different outlier definitions as described in bibliography together with the major algorithms and techniques that exist until today. We analyze the characteristics and the basic architect of a NIDS focusing on the state of the art system for each category SNORT and MINDS. At the end we contact an experiment with a small subset of the KDD'99 data set with the help of WEKA and the LOF, in order to highlight the importance of the measure, attribute and data object selection and their role to the final quality of the results.

At the end, we presented the challenges, that the ideal solution will be a combination of the two major NIDS categories, the old data sets that are used in the experiments which are unable to describe the contemporary characteristics of the network traffic, the issues that are occurring due to the constant increase of the networks, and we forecast that future research will needed in specific directions of the network intrusion detection, for example towards cloud data, wireless data etc.

Contents

1. Introduction

Usually, the majority of the books related to data mining, contain a section for preprocessing the data by cleaning or eliminating the outliers and the extreme values. This step actually is crucial for the majority of the domains, since the outliers and the extreme values do not have any significance at all for the analyst. In contrast with the above problem domains, in network intrusion detection, the outliers are the data object with the most important information amongst the normal data objects.

Network intrusion detection as a domain problem, utilizes the anomaly detection in order to identify attacks or misbehavior in general in the network data.

Anomaly detection is the identification of rare, abnormal observations which contain valuable information, and for that reason they are the main objects of interest in the specific problem domain.

We begin with the first part were the general background information regarding to the outliers is presented.

At the second part we take a look at the different outlier definitions as described in bibliography and we cover briefly the major algorithms and techniques that exist today regarding the outlier detection subject. At the end of each part the challenges and benefits of each algorithm are presented.

We continue with the third part where the architecture and the characteristics of the "Intrusion detection system" are presented. The different modules of the system and all the possible categories of an intrusion detection system are analyzed. We overview the two "state of the art" network intrusion detection systems SNORT and MINDS, and at the end the benefits and challenges section concludes the third part.

Before we conclude, at the fifth part, overviewing the open issues and challenges that intrusion detection systems are facing which are indicating the need for future research, at the forth part we describe an experimental process. In that section we use the LOF technique over the KDD'99 data set, and with the help of WEKA we add a "hands on" perspective in this paper.

1.1. Outliers

The first definition of the outlier was given by Hawkins in 1980. Since then, numerous definitions have been proposed but the main meaning of the outlier is still captured at the Hawkins definition [Hawkins 1980]:"An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". Almost thirty years after, all the proposed definitions do not differ significantly: Every data object, observation that differs/deviates from expectation (Jiawei, Kamber, & Pei, 2012). The process of finding the outliers in a data collection is called outlier detection, or anomaly detection.

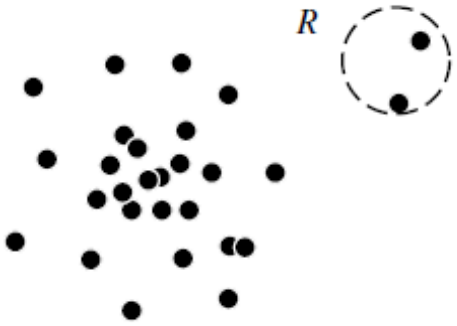


Figure 1: The R set is consider is an outlier set. (Jiawei, Kamber, & Pei, 2012)

Outliers are different from noisy data. Noisy data are also data object that are different from the expectation but they carry no information and have no significance at all. Eliminating noisy data can be a difficult and challenging task due to the fact that noisy data appear as outliers or "random error" . The outlier detection can be divided in two major tasks:

- Detect the outlier data object in the whole data set. By detect here, we mean to identify all the outliers that exist in the given data set, that fall in the outlier definition that was chosen or implied by the domain problem. Choosing the definition will characterize the outliers is a decision that have to been taken by a domain expert and will have a significant effect on the outcome of the analysis.
- Justify why a particular data object is an outlier. This task although it does not seem so important, it can have an impressive impact on the results especially if the analysis is using real world's data. With this task real outliers will be separated from noisy data objects. Furthermore by accomplishing this step, outliers can be divided in categories, and also every outlier will be in pair with a logical explanation of its outlierness. In general there is no automated process to accomplish this task, and usually a domain expert will analyze further the particular data object in order to give the final verdict.

1.2. Type categories of outliers

1.2.1. Global outliers: In a given data set a global outlier deviates significantly from the rest data object in the set. This type of outlier is the most easy to detect amongst the other types. The majority of the outlier detection techniques and methods are aiming to this category. Usually when random error data objects or noisy data in general are identified incorrectly as outliers, are falling in this category. Unfortunately in real situations, global outliers are not so common and when they occur the have a little significance or no significance at all.

1.2.2. Contextual outliers: In a given data set contextual outliers are characterized as outliers (according to the global outlier definition) only under the consideration of a particular condition. For example let's assume that we have a data set which contains

temperature measurements of a specific place. An observation of 30C is not considered as an extreme observation (outlier), but if we also suppose that this observation occurred in January, at a place somewhere around Athens, then this assumption makes that specific data object, outlier. We quickly realize that the attributes that will be selected for the outlier analysis have a massive impact on the results. Also the number of the attributes that will be selected is a crucial decision. Too many attributes will lead to a high dimensional problem difficult to be handled, but on the other hand selecting only a few attributes will lead to a superficial analysis, failing to identify contextual or collective outliers. The attributes can be distinguished in two categories:

- Contextual attributes. Those are the attributes that apply to the data object's context: For example if we collect temperature data then the date, time, location, technique of measurement can be some examples of contextual attributes.
- Behavioral attributes. Those are the attributes that apply to the data object's character: For example, regarding to our previous temperature data humidity, pressure, wind power and direction can be some examples of behavioral attributes.

In order to describe the outlier, not only the data object should be given, but also a set of context attributes, that identified this particular data object as an outlier. Global outliers can be recorded as contextual outliers but with an empty set of context attributes. With that in mind we can say that contextual outliers are a generalization of global outliers. As it was mentioned before, the selected attributes drive the quality of the results. A domain expert with a good understanding of the data mining process is required to select the set of the attributes in any case.

1.2.3. Collective outliers: In a given data set, a data object is called a "collective outlier" if it belongs to a subset that deviates significantly, as a whole, from the given data set. The data objects that belong to such a data subset can or cannot be outliers if they are examined individually, one by one, and not as a whole data set. Once more the outlier detection analysis will take place only on a subset of the attributes, so a domain expert is required to take the appropriate decision. Collective outliers are the most interesting outliers and have an important role in the applied outlier analysis. Identifying collective outliers is the primary task for domains that are related to "trend discovery". Trend discovery is examining collective outliers and their evolution in time, in order to conclude if the outlierness was an occasional observation, or is something that has a momentum and may progress in time.

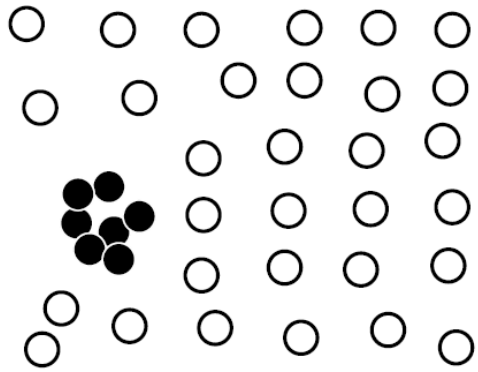


Figure 2: The bolt data objects are consider as "collective outliers". (Jiawei, Kamber, & Pei, 2012)

2. Anomaly detection techniques

2.1. Statistical approaches

2.1.1. General idea

In statistical approach methods the general idea is to identify the mechanism (model) that produces the normal data. After acquiring this information, the next step is to examine every particular data object and the possibility that it is produced by that model. In other words systems that fall in this category are trying to understand the nature of the normal data and the model that generates them. Then every new data object is examined against the known model and an outlierness score (possibility or deviation) is assigned to it. Outliers are considered the data object that have the highest score (in some cases the lowest score depending on the technique). A threshold is required, which indicates the percentage of the data objects that will be identified as outliers, or a number which any score under this number will automatically lead to the conclusion that this particular data object is an outlier.

2.1.2. Outlier definition

The majority of the techniques are considering the outliers as they were defined in 1980 by Hawkins [Hawkins 1980]. In other words we can say that as an outlier is considered every data object that has the lower probability of being generated by the normal data object model that was constructed from the normal observations.

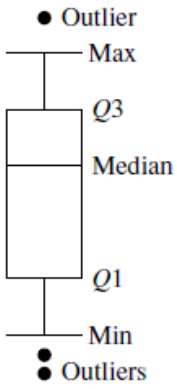
2.1.3. Description

We can assume for example that the normal data objects in that data have been generated by the normal distribution. For example let's assume that we have i data objects x_i $i \in \mathbb{N}$. In order to identify the distribution we have to calculate the mean μ and the standard deviation σ:

$$\hat{\mu} = \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2.$$

It is know that the 99.7% of the objects should fall in the (μ±3σ) fragment. So if we set the threshold for outlier detection at 0.3% then we can identify as outlier every data object that is outside of this area, or every data object that is further than 3σ from the median. It is also possible to draw a boxplot of the data set and identify as outliers all the data object that are not in the quartiles for the normal distribution. Every object that is outside of the boundaries of the box plot can be considered as outlier



Every data object that is outside of the boundaries of the box plot can be considered as an outlier

The above technique works perfectly in every domain problem as far as the data that have to be analyzed are not too complicated. But in real life, there are cases that the data are more complicated and it may happen that the data set is a union of data objects that were produced by two or more different distributions. This means that this time we have to assume that there are two (or more) normal distributions $\Theta_1(\mu_1, \sigma_1)$ and $\Theta_2(\mu_2, \sigma_2)$ and the probability of any data object O to be generated by them will be $Pr(O| \Theta_1,\Theta_2) = F_{\theta 1}(o) + F_{\theta 2}(o)$ where F_θ is the probability density function of the distribution. As before, we define that as outlier, we will consider every data object with the lowest probability sum of F_θ in general. A variation of this approach will be to assume that the data are following an unknown distribution in general, and try to learn the distribution and it's characteristics from the data. In this case the analysis is not tight to the normal distribution and the outcome model may describe the data better than the normal distribution, but the user will have to provide parameters that will drive the model building process. The user specified parameter usually is very crucial and the outcome of the analysis is relied on this selection.

Usually the user does not have any information in order to provide the required parameters so in most cases a "seek and find" approach with multiple attempts is followed. To our best knowledge the statistical approaches seem to be abandoned by the researchers, so we will not examine those approaches further in this paper. Related work about statistics-based can be found in (Manikopoulos & Papavassiliou, 2002)

2.1.4. Challenges and benefits

To begin with, the assumption that the data objects are following the normal distribution in most cases is too strict, leading in a model that is unable to fit in the data and describe them. As a result, normal data objects are incorrectly marked as outliers (false positive ratio) or outliers are not identified at all (false negative), affecting the reliability and the precision of the results. On the other hand if the user decides to go with the more flexible technique and try to identify the distribution of the data he will have to provide some parameters. If the user fail to provide correctly the above mentioned parameters then the model may fail again to describe the data, or it may over fit the normal data and increase dramatically the false positive ratio. Furthermore a learning data set is required with normal data for learning the model, and another data set with normal and outlier data (with labels) for controlling the precision of the model. In particular domain problems is really difficult or even impossible to build a normal only data set or a labeled data set.

The statistical approaches always assume that normal data are the most common data objects in the data set and outliers are always rare cases. Although that this assumption in general is true, there are some cases in real life scenarios that the outliers are the same in numbers as the normal objects or even more. For example in network intrusion when the attacker executes a DOS attack then the outliers (attacker's packets) can outnumber the normal packet traffic in the network.

In high dimensional spaces the data may be really complicated which means that oversimplified models that are trying to describe the data with one or two distributions are contaminated to fail. Trying to tackle the problem by introducing a higher number of mixed distribution will increase the computational costs without having the equivalent results.

On the other hand statistical approaches are expensive when they try to learn the data model. After building the data modal, the outlier detection for every data object is a low cost procedure which can be calculated on the fly giving a huge advantage for "on line" systems.

2.2. Proximity based approaches

2.2.1 General idea

Proximity based approaches, try to identity the outliers based on how similar is the data object O in particular with the rest of the data objects in the data base. For this purpose, a *similarity measure* or function is needed which will convert the "similarity" into a score. Data objects with low or high score (depends on the method) are marked as outliers. Based on this idea there are a lot of different approaches. For example there are approaches that they compare the similarity measure only to a neighborhood of the current data object or

they compare it only to a sample of the data. Also there are approaches that are using similarity measures which are based on the distance of the data objects or are based on the density of the data. One can apply classification on these approaches as local methods, distance methods, or density methods.

2.2.2. Outlier definition

For better understanding we provide the definition of the outliers as they were provided by the authors of the corresponding algorithms that we will present below. Every definition in this area has in common the dependency on the similarity measure (function). Also in every definition we can see that the authors try to involve the area of the similarity measure that was applied. Although all of them are defining what an outlier is, the user defined parameters that exist in the definition can really lead to different results:

1. "An object O in a dataset T is a UO (p, D)-outlier if at least fraction p of the objects in T are ≥ distance D from O." (Knorr & Raymond, A unified notion of outliers: Properties and computation, 1997), (Knorr, Raymond, & Tucakov, Distance-based outliers: algorithms and applications, 2000)
2. "For a positive integer k and a data point p ∈ DS, let $D^k(p)$ denote the distance between the k^{th} nearest neighbor of p and p. Then given two positive integers k and n, a point p is in the top n outliers of DS if no more than n - 1 other points in DS having a higher value of D^k than p." (Li & Guo, 2007)

2.2.3. Distance based outliers.

Distance based outliers are defined by the first definition as described above. In other words a distance based outlier is a data object which is the furthest from the rest data objects. The similarity measure that is used to measure the distance can vary but usually Euclidian, Mahalanobi's or Manhattan distance is used. First we need to introduce the k-neighborhood (kN(o)) of a data object o which belongs to a data set T (as defined in the first definition):

$$kN(o) = \{x \in T, x \neq o : D(x,o) \leq k\}$$

Where D(x,o) is the distance (similarity) function for the data objects o and x. The parameter k is called the radius of the neighborhood and usually is a parameter provided by the user. The user is requested also to provide another parameter called threshold which is notated by π. Usually the parameter π is given as a fraction of the size of the data set T. With the above said, the problem is translated into the following:

Find all the data objects o that the k-neighborhood has less data objects than π|T| or find all the data objects o that the π nearest neighbor is further than r. It is obvious that for the same data set T and different parameters k, π or different distance function D the results may differ.

2.2.3.1. Nested loop algorithm

In (Knorr, Raymond, & Tucakov, Distance-based outliers: algorithms and applications, 2000) the nested loop algorithm was presented. The nested loop algorithm is a straight

forward approach for the outlier definition that we discussed earlier (outlier definition number 1) which, as expected, has not the best performance results.

Input:

- a set of objects $D = \{o_1, \ldots, o_n\}$, threshold r $(r > 0)$ and π $(0 < \pi \leq 1)$;

Output: $DB(r, \pi)$ outliers in D.

Method:

```
for i = 1 to n do
    count ← 0
    for j = 1 to n do
        if i ≠ j and dist(o_i, o_j) ≤ r then
            count ← count + 1
            if count ≥ π · n then
                exit {o_i cannot be a DB(r, π) outlier}
            endif
        endif
    endfor
    print o_i {o_i is a DB(r, π) outlier
endfor;
```

Figure 3: Nested looped pseudo code for distance based outliers (Jiawei, Kamber, & Pei, 2012)

As (Aggarwal, 2013) mentions the nested loop approach fall in the complexity group of the $O(n^2)$ algorithms. Although the approach tries to compare all the data objects with all the rest members of the set, in a data set where the number of the distance based outliers is not high the inner loop will end before an exhaust comparison will take place as the object will be marked as not outlier. This fact does not change the complexity of the worst case scenario but the average complexity can be improved significantly. Another challenge that this approach has to face is the number of I/O. If the data set is big enough and cannot fit in the memory then the number of I/O increases and the same happens for the execution time. In (Knorr, Raymond, & Tucakov, Distance-based outliers: algorithms and applications, 2000) this fact is taken by consideration and an approach is presented where the data set is divided to subsets big enough to fit two of them in memory at once. This approach reduces the number of I/O and improves the execution time.

2.2.3.2. Grid based algorithm

In (Knorr, Raymond, & Tucakov, Distance-based outliers: algorithms and applications, 2000) another algorithm is presented which is more efficient than the nested loop algorithm in the case of less than 5 dimensions. The grid based algorithm is more efficient because it process cell by cell the data set and not object by object. In order to proceed we introduce the concept of the cells and the level i or L(i) neighbor cells. By $C_{x,y}$ we notate the cell C

that is at the intersection of row X and column Y. As level 1 neighbor cells we refer to all cells that:

$$L_1(C_{x,y}) = \{C_{u,v} : u = x \pm 1, v = y \pm 1, C_{u,v} \neq C_{x,y}\}$$

In other words as level 1 neighbors of a cell $C_{x,y}$ we define all the surrounding cells that share one side with the cell $C_{x,y}$ or they share one of the four corners. Every cell that is not boundary cell has 8 L_1 cells in a data set with 2 dimensions. (See figure 4).

Now as level 2 neighbor cells we define the following cells:

$$L_2(C_{x,y}) = \{C_{u,v} : u = x \pm 3, v = y \pm 3, C_{u,v} \notin L_1(C_{x,y}), C_{u,v} \neq C_{x,y}\}$$

In other words level 2 cells of the cell C are the level 1 neighbors of the $L_1(C)$ and their level 1 neighbors also. Note that the layer 1 neighbors is 1-cell thick but the layer 2 neighbors of a cell is 2-cell thick. Every cell that is not at the boundaries of a two dimension data set has forty level 2 neighbor cells. (See figure 9)

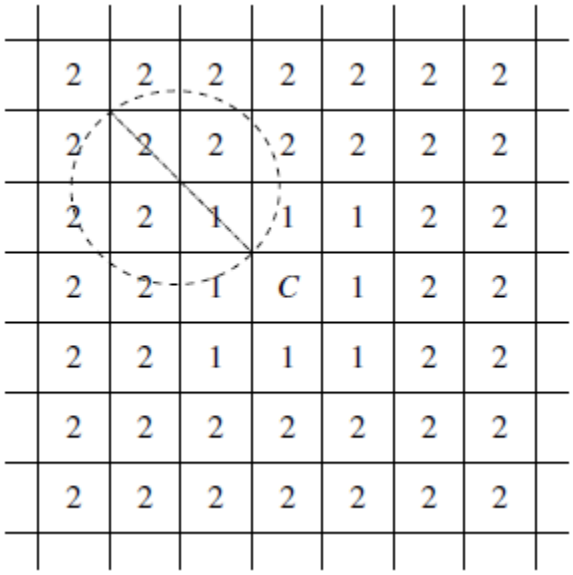


Figure 4 : A cell C and its level 1 and level 2 neighbor cells (Jiawei, Kamber, & Pei, 2012)

Mr. Knorr and the rest authors proved that if you select the length of the square grids to be $l = \frac{k}{2\sqrt{D}}$ where k is a user defined parameter and D is the number of the dimensions that the data set has then the following statements are true:

1. If there are more than M objects in $C_{x,y}$, then none of the objects in $C_{x,y}$ is an outlier.
2. If there are more than M objects in $C_{x,y} \cup L_1(C_{x,y})$, then none of the objects in $C_{x,y}$ is an outlier.
3. If there are less than M objects in $C_{x,y} \cup L_1(C_{x,y}) \cup L_2(C_{x,y})$ then every object in $C_{x,y}$ is an outlier.

Based on the above statements the grid based algorithm (or FindAllOutsM) starts by quantizing the data set in cells and matching every data object in its appropriate cell. After

that it marks all the cells that fulfill the first statement as red. It continues with the level 1 neighbors of all the red cells which fulfill the statement 2 and are marked as pink. Also as pink is marked every cell that fulfills statement 2 but it is not a level 1 neighbor of a red cell. When the algorithm has finished marking all the cells that are possible to be marked, then it starts performing a nested loop algorithm for the unmarked cells (white cells). Grid based algorithm has a reduced execution time ,comparing to the nested loop algorithm, because it quickly identifies large areas of the data set that are not outliers while in the same time, it provides similar results as the nested loop algorithm. The complexity of the algorithm is O (m + N) where m is the number of the cells $m \ll N$ which is a huge improvement comparing to the O (N^2) that the nested loop algorithm is offering.

Algorithm FindAllOutsM

1. For $q \leftarrow 1, 2, \ldots m, \quad Count_q \leftarrow 0$
2. For each object P, map P to an appropriate cell C_q, store P, and increment $Count_q$ by 1.
3. For $q \leftarrow 1, 2, \ldots, m, \quad$ if $Count_q > M$, label C_q *red*.
4. For each *red* cell C_r, label each of the L_1 neighbours of C_r *pink*, provided the neighbour has not already been labelled *red*.
5. For each non-empty *white* (i.e., uncoloured) cell C_w, do:
   a. $Count_{w2} \leftarrow Count_w + \sum_{i \in L_1(C_w)} Count_i$
   b. If $Count_{w2} > M$, label C_w *pink*.
   c. else
      1. $Count_{w3} \leftarrow Count_{w2} + \sum_{i \in L_2(C_w)} Count_i$
      2. If $Count_{w3} \leq M$, mark all objects in C_w as outliers.
      3. else for each object $P \in C_w$, do:
         i. $Count_P \leftarrow Count_{w2}$
         ii. For each object $Q \in L_2(C_w)$, if $dist(P, Q) \leq D$:
             Increment $Count_P$ by 1. If $Count_P > M$, P cannot be an outlier, so goto 5(c)(3).
         iii. Mark P as an outlier.

Figure 5: Pseudo code for the grid based algorithm (Knorr, Raymond, & Tucakov, Distance-based outliers: algorithms and applications, 2000)

Attention is required when someone tries to generalize the grid based algorithm in data sets with more than two dimensions. In order for the statements 1-3 to be true the thickness of the level 2 cells, must be increased depending on the dimensions of the data set. Although the length of the cell l, was defined as a function of the number of the dimensions D the definition of the level 2 neighbors is not including the number of dimensions. If we want

to generalize the definition of the level 2 neighbors, we can say the following (Knorr, Raymond, & Tucakov, Distance-based outliers: algorithms and applications, 2000):

$$L_2(C_{x1,\ldots,xD}) = \{C_{u1,\ldots,ud}\} : u_i = x_i \pm [2\sqrt{D}], \forall\, i\ 1 \leq i \leq D, C_{u1,\ldots,ud} \notin L_1(C_{x1,\ldots,xk}), Cu1,\ldots,uk \neq C_{x1,\ldots,xk}\}$$

Also based on the same resource, experimental results show that the performance advantage of the grid based algorithm is vanished if the data set is big enough or at a minimum it has more than 5 dimensions. In these cases is more preferable to use the simple nested loop algorithm since the calculation overhead for the cells is skipped.

2.2.4. Summary for the distance based approaches

In the previous section we presented two algorithms for distance based outlier detection. The algorithms that were presented are considered as the state of the art in the distance based outlier detection field. Although they are simple and easy to implement they face two critical challenges.

First of all both approaches fail to handle the curse of the dimensionality. Although the grid based algorithm is performing better than the simple nested loop in data sets with less than 5 dimensions, in real life data sets with huge amount of data objects and more than 5 dimensions both algorithms have a drawback in the aspect of performance.

In both algorithms the user has to provide two parameters. The parameter r for the neighborhood and the parameter π for the minimum number of the data objects that have to be in the neighborhood before any data object is marked as an outlier. Both parameters affect the result of the analysis, and usually more than one iterations are required with different r, π as input in order for the user to be able to identify the optimum parameter values that provide an acceptable true negative, false positive ratio.

Furthermore in complex real life data sets where the data objects are forming clusters with different distances between them the distance based algorithms fail to identify the true outliers. There are situations where local outliers may exist and may be apart from a cluster in a lower distance of a less dense cluster. In these situations, it is difficult and even unfeasible to identify optimum parameters r,π for identifying with any error all the outliers in the data set. For example if we consider the following data set (figure 6) we can understand that all the distance based algorithms will identify O_3 as an outlier and also they will identify incorrectly O_4 as well.

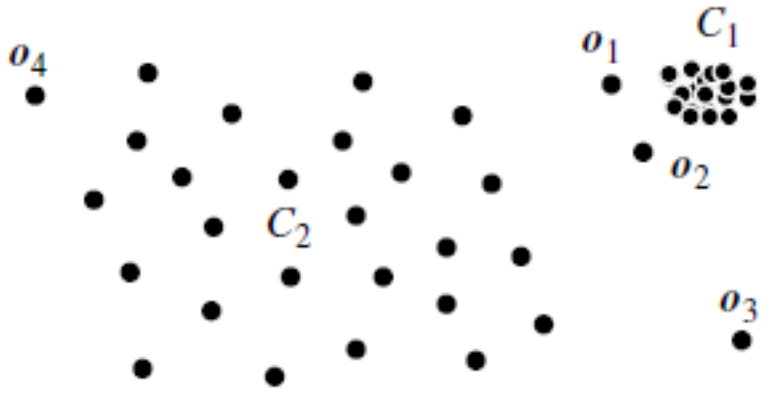


Figure 6: Local and global outliers (Jiawei, Kamber, & Pei, 2012)

The outliers O1 O2 will be ignored and not flagged. If the user tries different input parameters in order not to identify the O4 incorrectly as an outlier, then he will miss also O3. On the other hand, if he tries to change the input parameters in order to identify the O1 O2 as outliers, then every member in the cluster C2 will be identified incorrectly as an outlier. The above figure underlines the issues that the any technique will have to face, as long as it does not take under consideration the variation of any applied measure in different subsets of the initial set.

2.2.5. Density based outliers

2.2.5.1. General idea

In general density based approaches assign to every data object in the data set, a number, something like a score, which describes the relative density of that object's neighborhood. After that, like the distance based approaches, the algorithm will define a neighborhood of the data object and it will compare the score of the object with the score of the objects that belong to this neighborhood trying to identify if it is an outlier or not. The key here is that the range of the neighborhood is not static or user defined for all the objects in the set, but the algorithm itself calculates the best range depending on the current data object. This fact helps to overcome the drawback that distance based approaches are facing: the local outliers. Also density approaches are powerful in set with high dimensionality and that's why they are in the center of the academic research.

2.2.5.2. Outlier definition

Although the authors of the "state of the art" algorithms in the density based approach that are presented below, have not provide a clear definition of what an outlier is, we can say that the following state is the definition of density based outlier:

> "An object is an outlier if, in some way, is significantly different from its neighbors" (Papadimitriou, Kitagawa, & Gibbons, 2003)

Some can argue that the definition provided above is not so different form the distance based definitions, but the fact that the neighborhood and the way to determine the difference is not strictly defined, is the turning point to allow a different approach.

2.2.5.3. Local outlier factor (LOF algorithm)

To begin with we need to introduce the k-distance of an object p, the k-distance neighborhood of an object p and the reachability distance of an object p. All the definitions bellow are presented as in (Breunig, Kriegel, Raymond, & Sander, 2000):

For a k > 0 the k-distance of p notated as $dist_k(p)$ is the distance d (p,o) such that

1. at least k objects o' $\in D$ (o' $\neq$ p) where d (p,o') $\leq$ d (p,o).
2. at most k-1 objects o' $\in D$ (o' $\neq$ p) where d (p,o') < d (p,o).

In other word the $dist_k(p)$ is the distance of the k^{st} closest neighbor to the data object p.

Similar the k-distance neighborhood of the object p is:

$$N_{distk(p)}(p) = \{q \in D - \{P\} : d(p,q) \leq dist_k(p)\}$$

In other words the k-distance neighborhood is the subset of all the objects that are less or equally close with the k-nearest neighbor of the object p. Keep in mind that the cardinality of the k-distance neighborhood may be more than k due to the fact that some objects may have the same distance from the object p.

Now for the reachability distance p for an object o:

$$reach\text{-}dist_k(p,o) = \max\{dist_k(o), d(p,o)\}$$

Which in other words means that the reachability distance is the k-distance if the object is closer than the k-nearest neighbor or the normal distance if it is further.

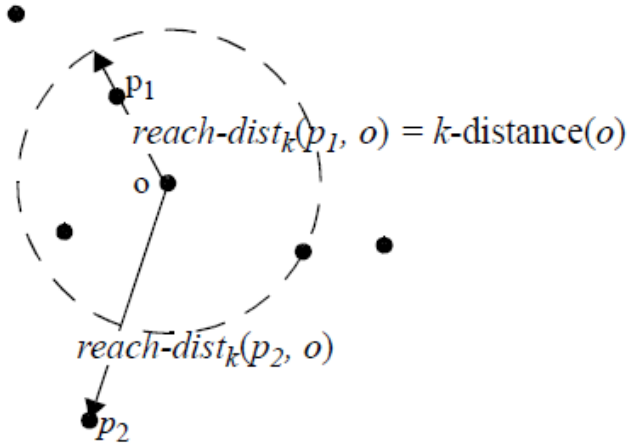


Figure 7: Reachability distance for k =4, (Jiawei, Kamber, & Pei, 2012)

The reachability distance is not symmetric because in most times the following statement is true:

$$\text{reach-dist}_k(p,o) \neq \text{reach-dist}_k(o,p)$$

Reachability distance is used as a smoothing effect because if an object has really close neighbors to it, it can manipulate the local outlier factor. The parameter k is user defined and it is his responsibility to control how powerful the smoothing effect will be.

Before we define the local outlier factor of an object p, we need to introduce one more definition, the definition of the local reachability density of an object p:

$$\text{Lrd}_{\text{minPts}}(p) = \left(\frac{\sum_{o \in NminPts(p)} reach-distMinPtsN(p,o)}{|NminPts(p)|}\right)^{-1}$$

In other words the "local reachability density of an object p is the inverse of the average reachability distance based on the minPts-nearest neighbors of p" (Breunig, Kriegel, Raymond, & Sander, 2000)

With all the above said we are ready to define the local outlier factor of an object p:

$$\text{LOF}_{\text{minPts}}(p) = \frac{\sum_{o \in NminPts(p)} \frac{lrdminPts(o)}{lrdminPts(p)}}{|NminPts(p)|}$$

The LOF of object p can capture the degree to which we can call p an outlier. In other words LOF is the average of the ratio of the local reachability density of p and of the local reachability density of the objects that are closer than the k closest neighbor. The min points parameters is user defined as mentioned before and can affect the outcome of the analysis. Now once we have calculated all the LOF for every data object in the data set it is easy to identify the outliers because the LOF is self-describable from its nature. A LOF value of around one means that the data object in considered as a normal data object. A LOF value lower than one indicates that the data object belongs to a higher dense area, where on the other hand a LOF value greater than one shows that the data object belongs to a less dense area therefore is considered as an outlier. We have to notice here that for a different data set and for a different domain problem the threshold of which all the data objects with higher LOF are considered as outliers is different. For example in a specific data set a LOF value around 1.25 may be considered as a value for a normal data object, where in a different data set a LOF value of 1.05 may already be considered as an outlier.

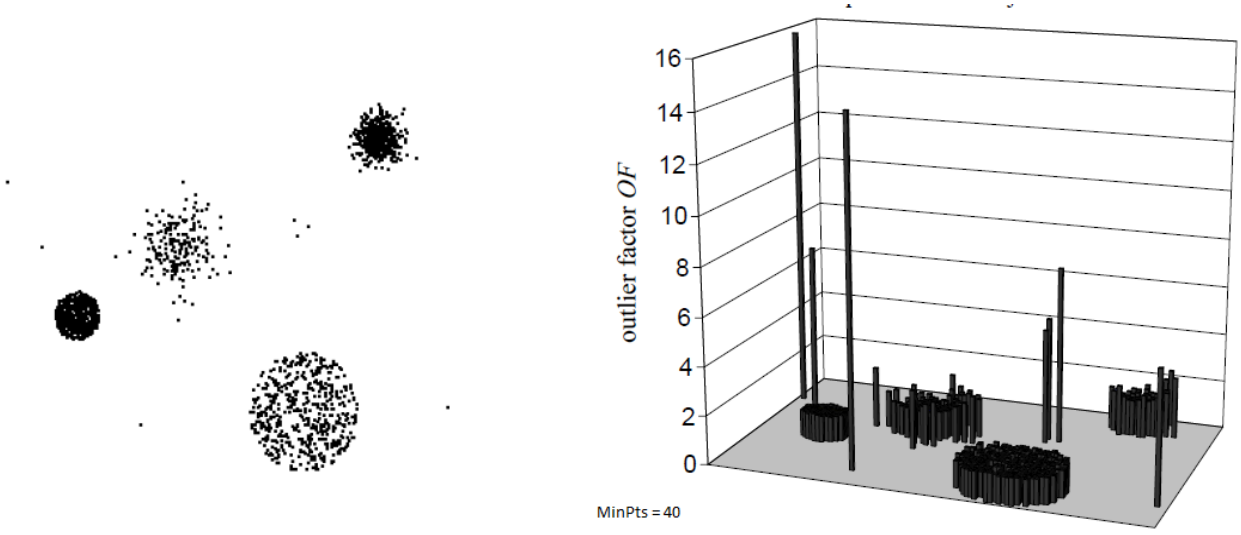


Figure 8: Outlier factors for point in a sample (Breunig, Kriegel, Raymond, & Sander, 2000)

For example, in the figure 8, one is easy to see that any data object with a LOF value greater than 1 or 1.10 can be considered as an outlier. Note that the data set is 2-dimension containing a low density Gaussian cluster of 200 objects, three 500 object clusters with different densities and some outliers.

In the same paper (Breunig, Kriegel, Raymond, & Sander, 2000) in which LOF was presented, experiments were also contacted with 64-dimension data sets demonstrating the real power of the density based approaches. Before we analyze the complexity of the LOF approaches we should outline that the algorithm is a combination of two steps. The first step is to find all the MinPts nearest neighbors for every object p in the data set and the second step is to calculate the LOF for the data object p and for the MinPts as well. For the first step the complexity of the algorithm varies between the linear O (n) and the O(nlogn), depending on the implementation, but in the worst case it can reach up to O (n^2) depending on the dimensionality of the data set and the approach that will be used in order to form all the neighborhoods. For the second step the initial data set is not needed, since the output of the first step contains all the necessary information for the computation of the LOFs. In the worst case scenario the complexity will be O (n) for the second step but keep in mind that the set that first step produces is smaller but it requires double scan. To sum up a total of O (n^2 +2n) is needed in the worst case scenario.

2.2.5.4. LOCI and aLOCI: (approximate) Local correlation integral

In (Papadimitriou, Kitagawa, & Gibbons, 2003), based on the LOF algorithm, the authors presented a faster and novel algorithm for identifying local outliers. Local correlation integral or LOCI at it was named is following the density approach and it seems more appealing than LOF because it doesn't require any input parameters from the user.

To begin with we need to define the following (P here is the data set and p_i a data object that belongs to the P):

$$n(p_i,ar) = |p \in P : d(p,p_i)<ar|,$$

Or in plain words, by $n(p_i,ar)$ we symbolize the number of the data objects that exist in the ar-neighborhood of the object p_i.

And with

$$\hat{n}\ (p_i,r,a) = \frac{\sum_{p \in n(pi,r)} n(p,ar)}{n(pi,r)}$$

the average of n(p,ar) over the subset of the r neighbors of p_i. One can mention the two different neighborhoods that are being used. We quote the definitions of the counting neighborhood and of the sampling neighborhood: "Counting neighborhood (or ar-neighborhood) is the neighborhood of radius ar over which each n(p,ar) is estimated. The sampling neighborhood (or r-neighborhood) is the neighborhood of radius r over which we collect samples of n(p,ar) in order to estimate the $\hat{n}$ (p_i,r,a) " (Papadimitriou, Kitagawa, & Gibbons, 2003).

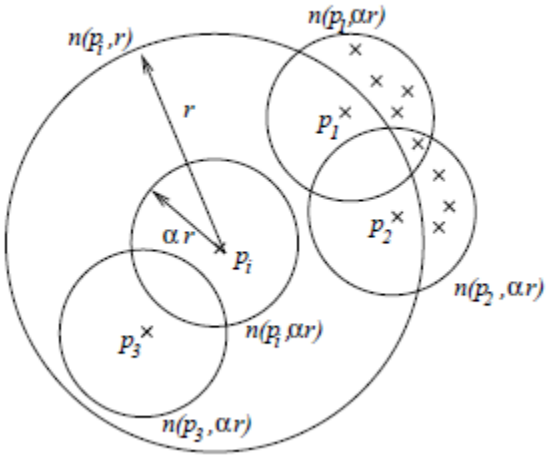


Figure 9: Graphical representation of the sampling and counting neighborhoods for pi (Papadimitriou, Kitagawa, & Gibbons, 2003)

With the above defined we can introduce the multi granularity deviation factor or MDEF:

$$\text{MDEF}\ (p_i,r,a) = \ 1 - \frac{n(pi,ar)}{\hat{n}(pi,a,r)}$$

The LOCI method relies on the standard deviation of the counting neighborhood over the sampling one:

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

The LOCI algorithm starts by calculating the MDEF and the standard deviation of the MDEF, for every p_i data object in the data set, flagging as outlier the data objects their MDEF value is greater than the standard deviation of it. As one can notice there are some parameters to be defined by the user. The r_{min} and the r_{max} for the standard deviation calculation the a for the sampling/counting neighborhoods. But as (Papadimitriou, Kitagawa, & Gibbons, 2003) are mentioning the r_{max} can be the highest r in the data set P and as r_{min} someone can choose an r that is small enough not to introduce statistical errors in the MDEF and σ_{MDEF} values (an r that gives $\hat{n}_{min} = 20$ neighbours is suggested) With the above said it is obvious that the user has only to decide the value k_σ in order to identify the outliers:

$$MDEF(p_i, r, \alpha) > k_\sigma \sigma_{MDEF}(p_i, r, \alpha)$$

In the same paper LOCI plotters were introduced, which are a plot of the sampling and counting neighbourhoods in the range of 3 standard deviations versus r. The LOCI plot can be drawn without any extra calculation (perquisite is that the LOCI algorithm has been run and finished) and it is based on the result of the LOCI algorithm. The LOCI plots can give a visualization of the outliers and their neighbourhoods helping the user to justify and understand the reason of the outlierness of every candidate.

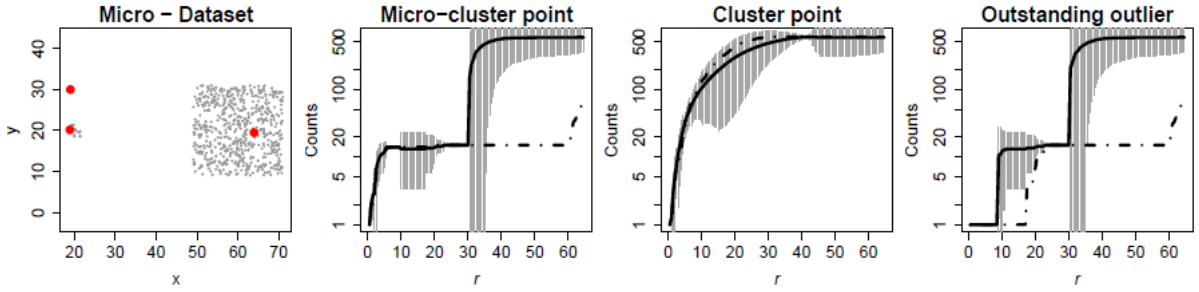


Figure 10: LOCI plots from an actual data set (The solid line shows $\hat{n}$ and the dashed line is n *is) (Papadimitriou, Kitagawa, & Gibbons, 2003)*

The aLOCI algorithm is an approximate version of the LOCI algorithm that we briefly described above. The aLOCI instead of using neighborhoods which may lead in numerous calculations, it splits the dataset in 2r-cells and replaces the neighborhoods with the cells. Based on the cells and the sum of the neighbors that exist in every cell a math formula can be defined for the approximate value of the MDEF and its standard deviation. For the approximate version of the algorithm the complexity is linear with the data size and the dimensionality.

2.2.6. Summary of the density based approaches.

Density based approaches give an answer to the problem with the local outliers that distance based approaches fail to handle. On the other side density based approaches add

an extra overhead in the calculation related with the formation of the neighborhoods. In practice density based approaches will require more runtime memory size due to the fact that they have to store a lot of runtime calculated information. Of course the above mentioned drawback are well known in the people that are involved in this area, and approaches that are taking extra care of these aspects have been presented like (Kriegel, Kroger, Schudert, & Zimek, 2009), (Lazarevic & Kumar, 2005). aLOCI is also one approach to this direction. In general density based approaches are considerate as the state of the art in the outlier detection due to the fact that are able to handle the overall and local outliers, in reasonable computation costs, are able to handle multiple dimensions and no special user parameters, except certain cases, or training of the algorithm is required.

2.3. Clustering /Classification based outlier detection

2.3.1. General idea

In clustering based approaches the general idea is based on the fundamentals of the data mining techniques. These approaches are taking benefit from all the well-known unsupervised algorithms and techniques that have been developed for clustering the data. Usually the first step is to apply a technique in order to identify the clusters that exist amongst the data objects. Despite the technique that will be used the outcome should be the clusters of the data objects and knowledge for every data object in which cluster belongs to. After the first step is completed successfully the second step is to identify which clusters are rare clusters and which ones are common clusters. The data objects that consist the rare clusters are considered as outliers. Every new data object is then compared with every cluster in order to identify the cluster that it belongs to.

The same idea is applied to the classification approaches. In classification based approaches the data objects are classified in two major classes the good or normal data and the bad or anomalous data. All the well know classification techniques can be used like the Bayesian classifier, support vector machine techniques, decision trees etc. After the training phase, its simple to identify in which class a new data object belongs to.

2.3.2. Outlier definition

Approaches in this area are using the definition as is was introduced by Hawkins in 1980, (Zengyou, Xiaofei, & Shengchun, 2003):

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

2.3.3. Cluster based local outlier factor (CBLOF)

In (Zengyou, Xiaofei, & Shengchun, 2003) the authors proposed the cluster based local outlier factor. The CBLOF is a factor applied to every data object in the data set showing the "score" of outlierness of the data object. CBLOF is taking under consideration the outlier definition as it was proposed by Hawkins but also tries to capture the spirit of the local outliers as density based approaches also do.

To begin with we need to state that the result of the first step (the clustering step) is a set of k clusters:

$$\{C_1, C_2, \dots, C_k\}: C_1 \cup C_2 \cup .. \cup C_k = D \text{ and } Ci \cap Cj = \emptyset \ \forall\ i, j \leq k, i \neq j$$

In other words the result of the first step should be k clusters or classes that contain all the data objects in the data set D, with no overlapping clusters or classes. The next step is to provide parameters α, β in order to split the clusters in two major categories, the large clusters (LC) and the small clusters (SC). (Suppose that $|C_1| \geq |C_2| \geq \dots \geq |Ck|$)

$$(|C_1|+|C_2|+\dots+|C_b|) \geq |D|\ \alpha \quad (1)$$

$$|C_b|/|C_{b+1}| \geq \beta \quad (2)$$

With parameters α, β provided the user is able to identify the cluster C_b which is named "boundary cluster". Every C_i with $i \leq b$ is considered as a large cluster (LC) and all the rest are considered as small clusters (SC). Based on that we can say that $\alpha \in (0, 1)$ and $\beta > 1$.

$$CBLOF(t) = |C_i| * \min(distance(t, C_j)) \text{ where } t \in C_i,\ C_i \in SC \text{ and } C_j \in LC, \text{ for } j=1 \text{ to } b.$$

OR

$$CBLOF(t) = |C_i| * distance(t, C_i) \text{ where } t \in C_i \text{ and } C_i \in LC.$$

In other words the CBLOF is determined by the size of the cluster that the data object belongs to multiplied by the distance of the closest large cluster, or the same cluster that the data object belongs to if it's a large cluster. In (Zengyou, Xiaofei, & Shengchun, 2003), the authors proposed that as a distance measure is sufficient to adopt the same similarity measure that was used in the clustering step.

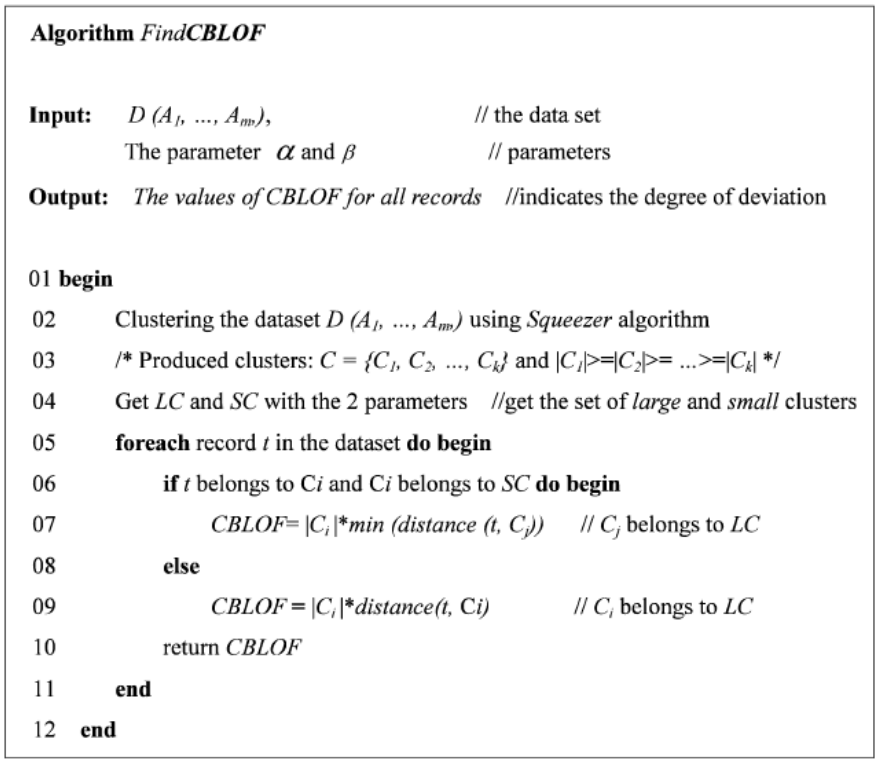
2.3.4. Find CBLOF algorithm

In (Zengyou, Xiaofei, & Shengchun, 2003) the Find CBLOF algorithm was introduced. In the first part of Find CBLOF the squeezer clustering algorithm is employed in order to identify all the clusters in the data set. After that the clusters are order by their size in order to identify the boundary cluster. Having found the boundary cluster, with one more additional scan for the whole date set, the CBLOF values are calculated for every data object in the data set. Once more, the user is required to set a threshold point for the CBLOF values, in order to identify the outliers that have greater CBLOF value than the threshold.

For the first step, the clustering step, squeezer algorithm requires O(n). FindCBLOF requires also O(n) to calculate all the values since it can be done in only one data set scan. Calculation costs for ordering the clusters are not taken under consideration since the

number of the clusters k is much more smaller than the data objects, k << |D|. Overall Find CBLOF requires O(2n) calculations.

FindCBLOF computational costs are low and the fact that it requires only two full data base scans makes it ideal for databases that are large enough to not to fit in the memory. Big databases that cannot fit in the memory require always extra I/Os which are time consuming. Also FindCBLOF does not require label data or any training at all making it flexible for a variety of domain problems. On the other side user is required to provide the α, β parameters which can be a tricky decision with an impact on the result of the analysis.



Algorithm *FindCBLOF*

Input: $D(A_1, \ldots, A_m)$, // the data set
The parameter α and β // parameters

Output: *The values of CBLOF for all records* //indicates the degree of deviation

```
01 begin
02     Clustering the dataset D (A1, ..., Am) using Squeezer algorithm
03     /* Produced clusters: C = {C1, C2, ..., Ck} and |C1|>=|C2|>= ...>=|Ck| */
04     Get LC and SC with the 2 parameters   //get the set of large and small clusters
05     foreach record t in the dataset do begin
06         if t belongs to Ci and Ci belongs to SC do begin
07             CBLOF= |Ci|*min (distance (t, Cj))     // Cj belongs to LC
08         else
09             CBLOF = |Ci|*distance(t, Ci)          // Ci belongs to LC
10         return CBLOF
11     end
12 end
```

Figure 11: FindCBLOF algorithm (Zengyou, Xiaofei, & Shengchun, 2003)

2.3.5. Summary of the clustering and classification based approaches

Unsupervised techniques (clustering approaches) are fast and are appropriate for outlier detection "on the fly". The computational costs of identifying the clusters that exist among the data may be high, but once the clusters are constructed, the identification of a new data object is with almost no computational cost because usually the clusters are low in numbers comparing with the numbers of the data objects. Also clustering approaches are able to identify collective outliers that may form a small cluster themselves.

On the other hand clustering approaches require big data sets during the training phase and if possible the data set should contains all the possible variations of the data. In large data sets computational costs during the training phase are really high. The outcome of the analysis heavily depends on the clustering algorithm, which algorithm usually, if not always, is not developed and optimized for outlier detection.

Supervised or semi-supervised techniques (classification approaches) share the same virtues as the clustering approaches, but they have an additional need of a labeled data set which in many domains is difficult or even impossible to be constructed. Also outlier data objects, in most of the cases, are rare objects amongst the data set and this can affect and bias the classifier during the learning process. Numerous different hybrid techniques have been proposed in order to tackle the class imbalance problem.

2.4. Approaches with respect to high dimensionality

2.4.1. General idea:

The previously mentioned approaches, in general, suffer from the curse of dimensionality. Some of the above mentioned approaches are trying to cope with this problem, successfully or not. In bibliography there are a lot of approaches that have been introduced and they are trying to give an answer to the dimension problem. This kind of approaches do not employee proximity measures (ex. distance measures) because proximity measures have the tendency to deteriorate as the number of dimensions increases. Instead they employee methods that include heuristics which can cope better with the curse of dimensionality

2.4.2. Angle based outlier factor (ABOF)

In (Kriegel, Schubert, & Zimek, Angle-Based Outlier Detection in High-dimensional Data, 2008) the authors introduced the Angle Based Outlier Factor, or ABOF, a technique that's based on the variation of the angle between a particular data object and the rest of the data objects. As one can notice a data object that is not an outlier and it is close to the center of the cluster, surrounded by other data objects (for example the data object α in the figure 12) the variation of the angles are smaller comparing to the data objects that are on the boundary of the cluster (data object β) and even smaller comparing to the outliers that are far from the cluster and are not surrounded by any other data object (data object γ).

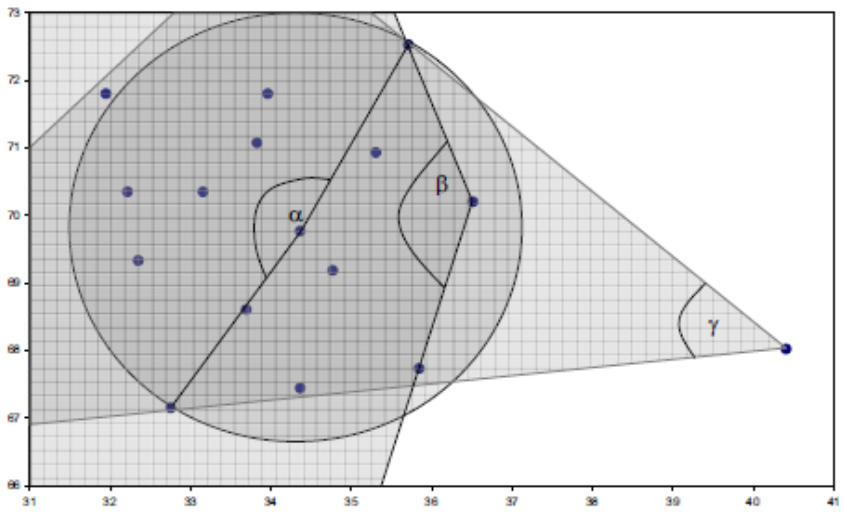


Figure 12: Angle based outlier factor (Kriegel, Schubert, & Zimek, Angle-Based Outlier Detection in High-dimensional Data, 2008)

Mathematically for each point α the ABOF is defined as it follows:

$$\text{ABOF}(\alpha) = \text{VAR}_{\beta,\gamma} \frac{(\overrightarrow{\alpha\beta}, \overrightarrow{\alpha\gamma})}{|\overrightarrow{\alpha\beta}|^2 * |\overrightarrow{\alpha\beta}|^2}, \beta,\gamma \neq \alpha$$

The angle based outlier factor also uses the distance measure but only as a secondary measure to normalize the outlier factor. The algorithm that is proposed with the ABOF is to calculate the ABOF for every data object in the data base and order them in descending order. User has to set only a threshold to set the boundary between the outliers and the normal data. In order to calculate the ABOF for every data object we need all the possible pairs of data objects that can be formed the particular data object α. After the formation of the pairs the calculation will take place in all the possible pairs of the above mentioned pairs.

One will notice that the ABOF with the above brute force approach needs $O(n^3)$ which is not so appealing especially for large data sets. In (Kriegel, Schubert, & Zimek, Angle-Based Outlier Detection in High-dimensional Data, 2008) an approximation method was also proposed called FastABOF. FastABOF offers $O(n^2+nk^2)$ where k is the approximation factor. As a tradeoff the user in FastABOF loses the parameter free approach and he has to provide the number of the k nearest neighbors that the calculation of the ABOF will take place. Also another tradeoff has to do with the definition of the neighbors. Since the definition of the neighborhood is heavily based on distance measures the FastABOF loses its ability to cope with high dimensions and the results deteriorate with the number of the dimensions. The lower bound ABOD or LB-ABOD was also proposed. For this method the user has to provide also the lowest number of the outliers that wants to be detected r. LB-ABOD also requires $O(n^2+nk^2)$ as the FastABOF, and an extra $O(rn^2)$ for the filter - refinement process, but it can cope with higher dimensions.

2.4.3. Isolation forest - iForest

Isolation forest is a technique, introduced by (Liu, Ting, & Zhou, 2008) based on the simple observation that outliers are less in numbers comparing to the normal data objects and that the outliers have attributes that are completely different from the normal ones. Exploiting the above observation the authors proposed a method that is based on the isolation of the outliers and does not try to model to the normal behavior in order to identify the outliers.

At the iForest technique a forest of classification trees (iTrees) is created by randomly selected attributes and split values for the selected attribute. The trees are of the same format as the binary search trees and not longer than Log (n) which is the average length of a binary tree for n attributes. The reason behind that is that the outliers are close to the root since are the most bizarre and uncommon data objects in the database. After the creation of the iForest every data object needs to be checked and assigned with the anomaly score S based on the following:

$$S(x,n) = 2^{-\frac{E(h(x))}{C(n)}}$$

Where c(n) is the average path length of the unsuccessful tree binary search:

$$c(n) = 2H(n-1) - (2(n-1)/n)$$

And H(x) is the harmonic number and is given by:

$$H(x) = \ln(x) + e$$

In (Liu, Ting, & Zhou, 2008) the following assessment was made, in order to identify the outliers:

1. If the s is close to 1, then the object is an outlier.
2. If the s is much smaller than 0.5, then it is safe to regard the data object as a normal data object.
3. If all objects return an s around 0.5 then the whole data set does not have any distinct anomaly.

User has to provide two parameters in order to use the iForest technique. As it is well known, decision trees have the tendency to over fit to the training data loosing efficiency when it comes to the real data. The first parameter that the user has to provide, ψ, is trying to solve the over fit problem, and it's the number of the objects that will be used for training purposes. The second parameter, t, controls the size of the forest to be build, and in fact, it is the actual number of the trees. During the training phase the complexity is $O(t\psi\log\psi)$. In the evaluation phase the algorithm calculates all the anomaly score for every data object in the data set, which means that it is required one data base read for every tree, $O(nt\log\psi)$

3. Intrusion Detection Systems (IDS)

An IDS is a system (software and hardware) that automates the process of the of the intrusion detection. Intrusion detection is one of the main tasks that every security administrator has to accomplish successfully every day. By intrusion detection we mean the discovery of every "attack" that occurs in a computer network. But what is considered as an attack? First of all an attack has not to be always successful. As an attack we can define any attempt, with or without success, of a security policy violation, in any aspect no matter if its goal is to harm the confidentiality or the integrity or even the availability of the network and its resources.

Attacks can be divided in 4 major categories:

- Probing. The attacker tries to gather as many information as he can of the network topology, the network services, the activities of the users in the network, the infrastructure etc. The information that will be collected during this attack is precious for the attacker in order to plan and execute any future attacks. This type of attack is not considered as the main activity of the attacker but it is a part of his preparation. The main attack may happen immediately after the end of this attack, but in most cases it will occur after a few days or weeks. During this attack the

attacker will use tools to help him in network packet sniffing, address scanning, port scanning.

- Denial Of Service (DOS). The attacker tries to diminish or completely interrupt the availability of one or more services in the network. It may be the primary attack or it can be considered as the first wave of the attacks that will follow. Attackers usually execute DOS attacks, prior to their main attack in order to deactivate the security systems of a network like firewalls, NIDS, allowing them to execute their primary attack more easily and with a bigger possibility to succeed.
- User to root (U2R). In this type of attacks the attacker is usually an "insider". He may be an employee of the company, a student of the university, a user that has some access to the network and its services. The attacker tries to gain a higher access level to a machine or a service, in which he has normal user access, by exploiting a software bug or a misconfiguration. As soon as the attacker will gain the required access he will try to steal information that it was unreachable to him before, or he will try to install software, reconfigure machines or services, delete logs that will help him to hide his identity, or place a "hole" in the systems allowing him to re attack more easily in the future.
- Remote to local (R2L).It is the same category as the U2R but in this case the user is usually an "outsider", a company's client, a member of the website, a typical consumer of the web service. The attacker is outside the security perimeter of the network and he has limited access to some services. By this attack he will try to gain higher level access to that services or he will try to gain access to services that he didn't have access at all before.

Usually in the real world an "attack" is something that lasts in time , it has a life time and the above mentioned categories are events of the attack that are happening in the particular time frame that the attack takes place. As is mentioned in (Chen, Chen, & Lin, 2009) the order of the events is as it follows: 1. Probing → 2. DOS → 3. U2R or R2L. At the end of that sequence the attacker will perform his main action which is to steal data, modify or delete data or implant a backdoor which will be available for his future attacks or for other attackers.

It is obvious now that attacks last in time and the sooner the system administrator notices the above mentioned events the more time will have to avoid the final action of the attacker which will is the most harmful. of all the actions, for the organization or the network. The IDS as a tool for attack discovery, automates the process by monitoring (scanning, sniffing) the network packets and by analyzing them. The monitoring is accomplished by sensors that are carefully placed by the network administrators, trying to gather packets that are incoming/outgoing traffic for the most important infrastructure in the network. Sensors are also placed to the entry points of the users in the network, or to points that the local private network interacts with outer networks which are not managed by the organization's administrators. A sensor can save all the packets that detect or it can ignore some packets according to the configuration provided by the security administrators. The sensors can operate in all the layers of the OSI layer model, but usually they capture packets in the

TCP/IP layer or on the application layer. The monitoring phase is really important and affects dramatically the outcome of the analysis phase, but it is not in the scope of this paper. In this paper we will examine the second phase, the phase of the analysis. In this phase data mining techniques are used in order to extract vital information, and patterns that exist in the data sets that were collected by the sensors in the monitoring phase.

3.1. IDS Architecture and characteristics

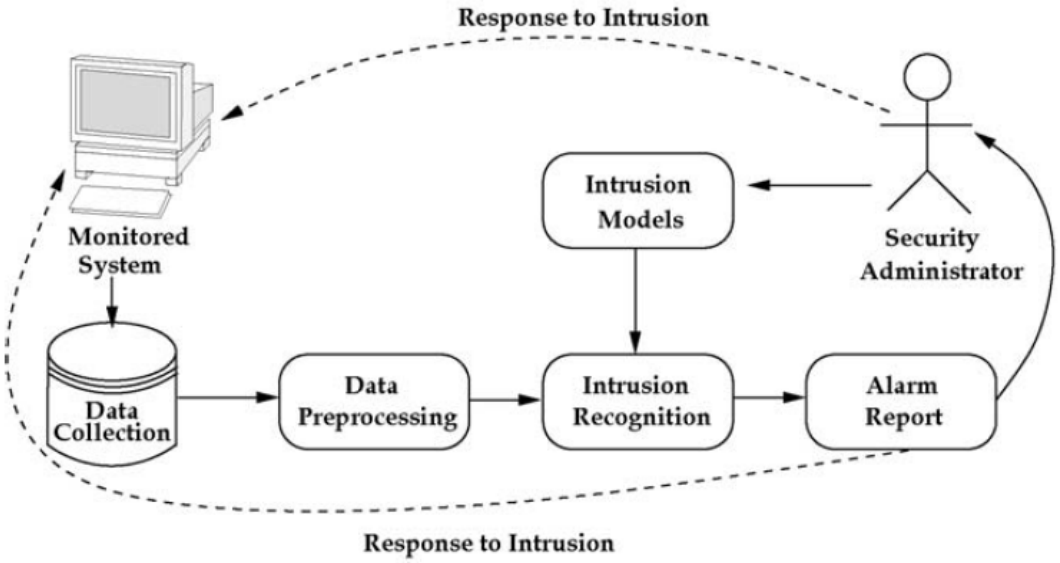


Figure 13: IDS Architecture (Shelly & Banzhaf, 2009)

In the simplest form of an IDS the modules that are mentioned in the Figure 13 are present. The **Monitored System** can be a host, meaning a computer station or a server, can be only one service which is running on a server, or it can be a part of the network or the whole network itself.

3.1.1. Data collection: In this part of the IDS the sensors are collecting the data sets that will be analyzed later. The sensors can collect any type of relevant data for example network dump files, network traffic, command sequences that users send to machines, or even low level system information such as CPU calls, system utilization and temperature. Here we have to make clear that by sensors we do not necessary mean a hardware device, or an extra software that has to be installed. The already installed software on a server may have the ability to provide to the security administrator log files containing the desired data. So in this case a piece of software that was develop for other purposes can play the role of a sensor.

Depending on the data that the IDS collects we can distinguish the following categories:

1. Host Intrusion Detection System (HIDS): Systems belonging to this category usually gather information that is related to one particular host, which is the host that is under protection. The data can be the login time of the users, the utilization of the system, CPU calls, number of I/O operations etc. The primary goal is to discover threats that are targeting the protected system.

2, Network Intrusion Detection System (NIDS): Network intrusion detections system need network data usually. The data can be packets of all the layers of the network stack but usually, are coming from the TCP/IP layer. Therefore their goal is to protect the whole network trying to detect threats that are happening in the network against any host or service that is part of the protected network.

3.1.2. Data Preprocessing: For a better utilization of the system the raw data should be optimized and prepared for the data analysis phase. In most cases this part of the IDS is considered as a sub part of the Data Collection subsystem.

3.1.3. Intrusion Recognition and intrusion models**:** This is the main part of the IDS. The data have been collected and preprocessed and are ready for the analysis which will take place in this part of the IDS. The selected algorithm (or sets of algorithms) will take place now and perform the analysis on the data. The algorithm selection is not an easy decision. The nature of the algorithm will have an impact on the final result quality. The analyst has to decide from a large variety of algorithms that the data mining field can offer, as we overviewed in the previous section of this paper. Depending of the type of algorithm and the type of the approach that will be selected, we can divide all the intrusion detections systems in two major categories. In this part we give a brief outline but later we will discuss them more thoroughly:

1. Rule based – Misuse (Signature based). The security administrator tries to define a set of rules that can be used to identify if a given behavior is consider as an intrusion or not. Systems like that are difficult to build and maintain. The sets of rules can reach numbers with a lot of digits leading to a system that is very difficult to be maintained ,altered, updated or even debugged in case of malfunctioning. For every incoming data object the system will try to match the header of all the rules that exists in the database. If the system is able to identify a rule that matches, then this rule is executed leading to the body of the rule which is an action, a report or an alarm. Signature based systems are really simple in the way that they work, a fact that means that they are simple to run but on the other hard one can anticipate a lot of drawbacks. They are unable to detect variation of a known attacks, unable to cope with unknown attacks, have a need to update their rule definitions as regularly as possible, etc. On the other side due to their simplicity they can demonstrate a really low false detection rate which makes them ideal for real life use.

2. Statistic method based – Anomaly detection. The system is trying to identify the exceptional cases, intrusion data, with the help of statistic approaches and models that have been proposed in the data mining field, in the particular area of the outlier detection. Usually systems that fall in this category employee one or more outlier identification

techniques that were described in the previous section of this paper. Depending on the algorithm that will be used, training data may be required, or human interaction may be required in order for the system to be trained. In complex scenarios, like computer network data, those systems are really difficult to be trained, due to the number of dimensions and parameters, or sometimes the trainer is unable to provide a label or a class for every data object that exists in the training set. In contradiction with the previous category, anomaly detection systems do not have the need for a regular update and they have the ability to identify previously unseen attacks, but on the other hand, anomaly detections systems have a high "false negative" rate, and are heavily based on the assumption that the majority of the network data is "normal" data and the non-legitimate data are statistically different from the normal data, which in real life is not always the case.
In bibliography one can find all kinds of methods parametric or non-parametric, proximity or density based methods, cluster or classifications based. Every algorithm that is well known in the data mining science field supervised or non-supervised can be tuned and used in outlier detection analysis. Algorithms that are based in support vector machine, genetics, rough sets, fuzzy logic are also used especially when the dimension of the space is really high which is the case of the IDS.

3.1.4. Alarm Report**:** The alarm report subsystem of an IDS is responsible for generating reports or taking proper actions. Usually the security administrator can configure this part of the IDS with respect to his needs and the needs of the network users. The alarm report subsystem can generate reports or alarms that can be sent directly to an email address or it can take actions like auto blocking IP addresses, block certain packets in the network, or even bring systems down in order to avoid data loss due to data penetration. Of course an automated respond action in order to block the recently detected attack is the desirable goal here, but the administrator needs to take under consideration the false negative rate in order not to block legitimate traffic in the network which may lead to user's frustration. (Estevez-Tapiador, Teodoro, & Vardejo, 2004). Usually systems that produce only reports or notifications are called "passive" or intrusion detection systems, in contradiction with the systems that are taking actions against the threat that are called "active or intrusion prevention systems IPS.

3.2. Rule based – Misuse (Signature based)

Misuse systems use a set of rules to identify known attacks and apply appropriate actions. It is obvious that for unknown attacks or for attacks that are known but there is no matching rule in the database no actions are taken.

The concept behind those systems is to provide a rule which is generic enough to identify as many variations of the attack as possible, but not so generic to wrongly identify legitimate traffic as malicious. For example for the "guessing password attack" the rule can look like the following (Sobh, 2005):

If <Num_Of_Failed_Logins> in 2minutes is greater than 4 → raise alarm.

Or

If <Num_Of_Failed_Logins> in 1 minute is greater than 100 → Block IP.

The first of the two rules assumes that there is a person behind the log in attempts who may have forgotten his password, or he is trying manually to guess a password that he doesn't have access, therefore only an alarm it's the appropriate action since the security administrator will have time to respond to this attack, which may not be an attack at the if the user just have forgotten or mistyped his password. Regarding the second rule, someone expects an automated system behind the numerous failed logins attempts. It may be a rogue computer, or a computer with a virus executing brute force guessing password attacks. In this case an automated action like blocking this particular IP is recommended. Notice that automated actions like the previous one are usually taken if the administrator is sure that is not a normal user behavior, 100 failed logins in less than a minutes is not a typical user's behavior.

3.2.1. SNORT

We will deepen more in the area of the rule based systems with the help of SNORT. SNORT (https://www.snort.org) is an open source NIDS which combines the benefits of signature (rule) based protocol and anomaly based inspection. SNORT is the most widely spread NIDS and has become the "de facto" standard for IPS/IDS real life solutions. It was released in 1998 by Sourcefire and until today it remains free. As every open source software, it has its own community which evolves, supports and develops the software itself but also numerous add-ons and plugins in order to modify the standard functionality to user's needs.

Figure 14: SNORT logo (www.snort.org)

SNORT can perform protocol analysis and content snatching making it capable of detecting a huge variety of attacks like buffer overflows, stealth port scans, CGI attacks, SMB probes, OS Fingerprinting etc. It also supports real time reporting including syslog, file logs, UNIX socket or Win Popup messages etc. Due to its large community, experienced and really keen on security, developers and system administrators have developed an endless list of add-ons and plugins for SNORT that extend the capability of the standard software. These extensions can be installed as preprocessors or postprocessors/alert generation (figure 15)

3.2.1.1 Snort architecture

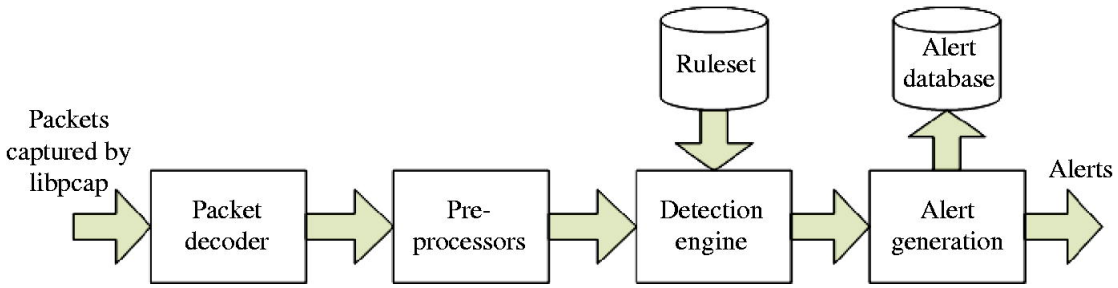


Figure 15: SNORT Architecture (Rehman, 2003)

Packet capture module: Used to capture network traffic using different libraries depending on which layer of the TCP/IP stack the analysis is performed. Usually the libcacp library (http://www.tcpdump.org/) is used. This module can be part of SNORT or it can be a third party software.

Packet decoder: It ensures to form the data structures of the packages captured and identify the network protocol. Packet decoder can work on different types of packets (like Ethernet, SLIP, PPP, etc.) decode them and prepare them for further analysis.

Preprocessor: Preprocessors are plugins developed generally in C which process the packets provided by the decoder and ensemble the packets received. Preprocessors are in charge of providing the data in the format that the detection engine can process. For example if plan text triggered rules are used then the preprocessor must return plain text and not encoded strings. Their main purpose is either to identify a kind of attack on their own, or re-organize and modify the input for the detection engine, helping the detection engine module to identify an attack successfully and in a more efficient way, meaning quicker response and more understandable rules for the security administrators. The preprocessors are configured in snort.conf file configuration. In this part some preprocessors are described in order for the reader to acquire a better understanding of how SNORT works:

- *SfPortscan*: This preprocessor detects directly the port scan attacks without the help of the detection engine. This type of attack belongs to the first phase of the attack where the attacker tries to identify the services that are running on a machine and the active ports. SfPortscan also generate the alerts directly, without the help of the alert mechanism, and it is able to create only on report per attacker host, reducing the number of the alerts significantly while helping the administrator to have a clearer view of what exactly happened. SfPortscan focuses only in this type of attack and its variations (decoy, distributed filtered, portsweep) making it a perfect tool for identifying port scan attacks, while reducing the number of the rules, that have to be stored in the database, and the number of the alerts that are produced by the alert generation module. More information can be found at http://manual.snort.org/node78.html.

- *Frag3*: Frag3 is a target-based IP defragmentation preprocessor. In a highly complex network, like the internet today, IP packets can be fragment increasing in this way the difficulty level for the detection engine to trigger the appropriate rules. Also attackers tend

to fragment their packets on purpose, and in complex ways trying to hide their attack in the fragmentation of the IP packets. Frag3 has the ability to defrag the packets putting them in the correct order. Frag3 comes also with mappings from the most well-known hardware and software vendors who implement IP fragmentation on their products based on the IP RFC's or even based on their proprietary protocols. (http://manual.snort.org/node60.html)

- *HTTP inspect*: This is a preprocessor that handles all HTTP traffic to help speed it through to the detection engine. This preprocessor purpose is HTTP traffic profiling and normalization, trying to detect http evasion and http proxy attacks. To demonstrate, imagine that a malicious user is trying to access the `/etc/passwd` file which in UNIX environments is a file that contains user names, groups, and security policies in general. The appropriate SNORT rule which every security administrator will have is the following:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC
/etc/passwd";flags: A+; content:"/etc/passwd"; nocase;
classtype:attempted-recon; sid:1122; rev:1;)
```

This rule says that if any external user under any port tries to access the http servers on the port 80 and his message contains the string "etc/passwd" then create an alert. But what will happen if the attacker sends and http request which is like "`GET /etc//\//passwd`" or like "GET %65%74%63/%70%61%73%73%77%64" which is the same as before but in hex encoding (imagine that the attacker is also capable of hiding his request in two different IP packets a case that refers to the previous preprocessor). The answer is that the detection engine will not trigger the above rule since the string matching will fail, but unfortunately the HTTP server will respond back since it is a proper http request. HTTP_inspect preprocessor is there to help the security administrators with this kind of problems and more. More information can be found at: http://www.symantec.com/connect/articles/ids-evasion-techniques-and-tactics or http://www.symantec.com/connect/articles/ids-evasion-unicode

- *ARP spoof*: The ARP spoof decoder, decodes ARP packets detecting ARP attacks and inconsistent Ethernet to IP mapping. An attacker can send spoofed ARP packets onto the LAN redirecting the traffic to come through a certain host which is under his control, becoming the "man in the middle". After that, he can inspect the traffic, insert malicious packets in the normal traffic or even stop all the communications. If a security administrator tries to identify the above mentioned attack with handmade rules in the detection engine, he will find himself lost in numerous rules very soon. ARP spoof preprocessor does not create any output for the detection engine module, but it creates its own alerts directly speeding up the detection process. More information about this preprocessor can be found at http://manual.snort.org/node151.html

The above mentioned preprocessor is a part of endless list. The selection that was presented in this paper is considered as "must have" tools for every SNORT implementation. For a complete list, further details and configuration readers can find more at this link: http://manual.snort.org/node17.html

Detection engine: It is the most important part of the SNORT software. Detection engine analyzes the packets based on the rules that are configured in the rule set. When a rule, or more rules, are matched then the detection engine executes the appropriate actions as they are defined in body of the rule. The detection engine analyses the packets on the fly, which means that performance is a critical aspect for the whole process. If the detection engine is overload with more packets that it can handle then it will start dropping packets leading to an incomplete analysis. The administrator must take under consideration the following performance parameters in order to avoid the above mentioned situation (Rehman, 2003):

1. Number of rules. The numbers of the rules that are define in the rule set and the complexity of them is the first parameter that must be taken under consideration.

2. Power of the machine. The power of the machine that the snort is running. CPU power and memory size are always critical factors for those processes. Of course the speed of the internal buses that these components communicate with each other is also a critical factor.

3. Load on the network. Highly load network means more packets to analyze, which means more workload for the detection engine. This is the first parameter that the administrator must analyze before he starts configuring the system and the rule set. Everything is in a relation with the nature of the network that the analysis will take place.

The help of the preprocessors must not be ignored here. Preprocessors can offer "out of the box" solutions for certain attacks, reducing the number of the rules in the rule set and the load of the detections engine. Also preprocessors can preprocess data in order to reduce the load of the detection engine is certain situations. In the rule matching process the detection engine tries to match as many rules as possible with the given data object. In case where more than one rule is matched then the rule with the highest priority will be selected to generate the alarm or to enforce any other actions defined in the rule. The task or organizing a rule set is the most challenging for the rule based systems. On the internet there are web sites which are focusing only in the construction of the optimal rule set for different type of networks. The type of the network, (wireless , Ethernet, etc.) the speed of the network, the utilization of the network, the number of the users, the exposure in attacks are some of the factors that will defined the rule set that will be used.

Detection plugins (Or postprocessors or output plugins or alert generators): By default SNORT creates a simple text log file with all the alerts (which is the result of the rules that were matched and applied by the detection engine). With the help of the output plugins the administrator can modify the standard behavior, defining and configuring where and how the alerts will be stored. Administrators have the ability to save the file in XML format (eXtensible Markup Language) or in CSV format in order to input the data in another software for a better visualization or for further analysis concepts (XML output module, CSV output module). There are plugins that are able to build SNMP traps (SNMPv3 output module) or SMB (for Microsoft windows based machines, alert_smb output module) or to send messages to Syslog servers in order to register the events (alert_syslog output module). But the most common output plugin which is also the most used by system administrators is the MySQL output module which stores all the alerts in a

database server like MySQL or Oracle. With the help of the databases, administrators can use graphical interface software like Snorby (https://snorby.org/), ACID (**A**nalysis **C**onsole for **I**ntrusion **D**atabases, (http://acidlab.sourceforge.net/), or Squil (http://sguil.sourceforge.net/) which will help me them understand more about the events, and help run queries against the database like the time period where the 10 most important attacks against the web server occurred etc. Some of them also support mobile clients for the data representation and for quick alerting purposes making the life of the administrator much easier. It is obvious that the representation of the data and the query ability that these tools provide, can make a huge difference in the quality of the overall process. Discovering the attack is for sure the primary goal of the NDIS in general but giving the ability to the security administrator to see the overall picture of the incidence is also crucial.

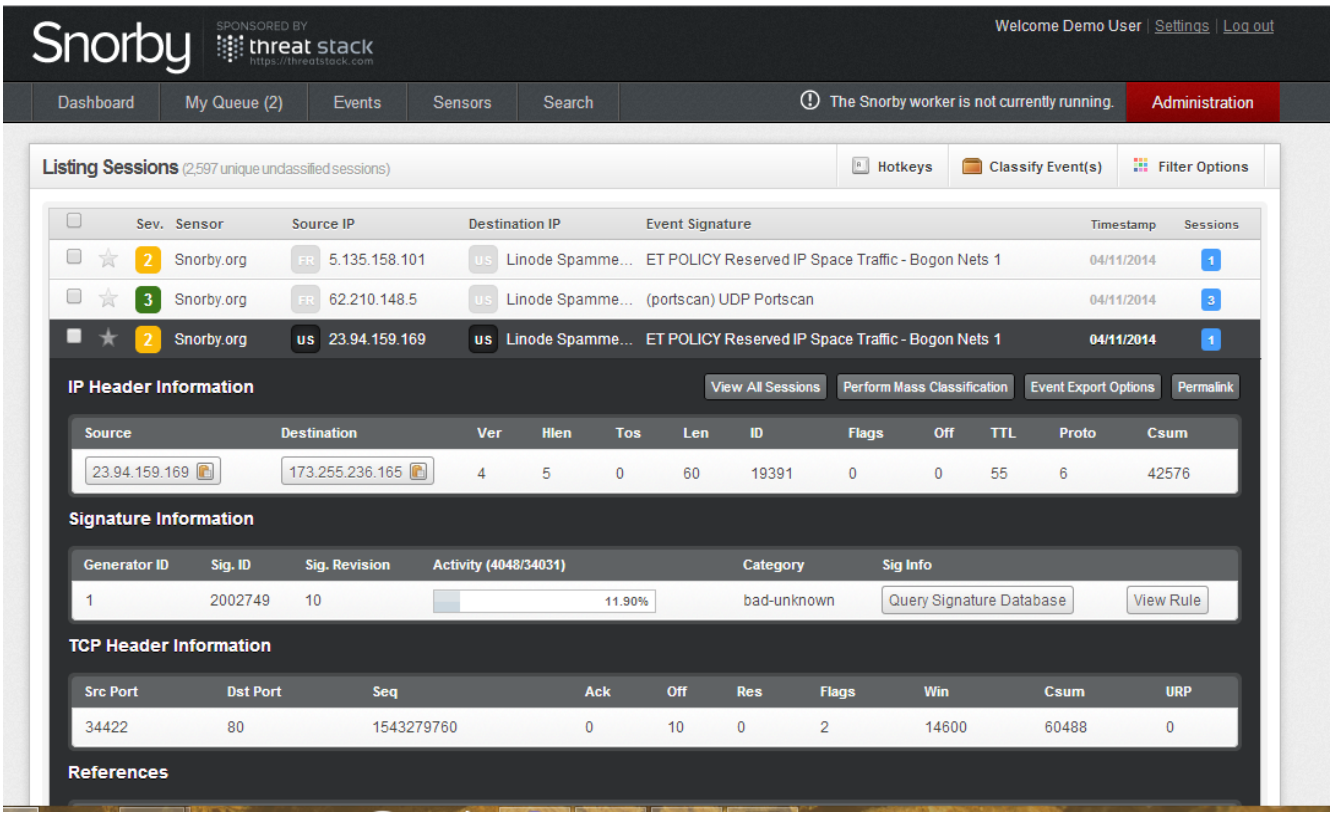


Figure 16: Snorby session listing (http://demo.snorby.org/)

3.2.1.2. SNORT rules

The rule in general has two parts. The rule header which is followed by the rule body (or rule options.) The rule header contains the information that is used by the detection engine in the matching process and the options of the rules are used in the alert engine after the rule is activated.

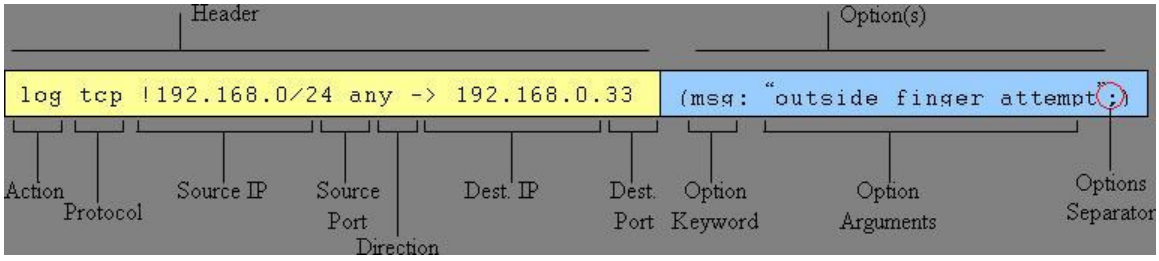


Figure17: SNORT rule format (http://www.linuxjournal.com/article/4668)

Action: The action part of the rules defines the action that must be taken in case of the rule is applied. Detection engine uses the action part to prioritize the rules, in the case where more than one rule is matched for a certain data object. By default SNORT has five categories of rules.

Pass: Detection engine will ignore this packet. Although this rule seems not important it plays one important role: to speed up the process ignoring packets that are trusted by the administrators.

Log: This action will just log the packet. It is used when administrators want to log something that will be consider important later.

Alert: The alert action will log the file and it will send an alert to the alert engine.

Activate and *Dynamic:* Activate types will execute an alert version of the rule but after that they will call all the related dynamic rule for further investigation of the packet. Dynamics rules are never matched by default except if are called by an activate type of rule.

SNORT gives the ability to the administrators to create more types of alerts according to their needs.

Protocol: The type of the protocol that the rule applies to. For example TCP, UDP, ICMP, IP are coming by default. There are custom detection engines that support other types of protocols like PPP or IGRP, RIP etc.

Source/Destination IP and port: In those parts the IP and the port number of the source and the destination is defined. SNORT supports CIDR block masks making the development of rules easier and less complex. Also in case of complex networks, SNORT support variables which means that the administrator can use for example for the destination address the variable $LOCAL. In that case he has to provide in the configuration file all the IP address (or ranges) that are consider as $LOCAL.

Direction: The direction operator (it is "->" for one way or "<>" for both ways) defines if only the packets form the <source> to the <destination> will be matched or also the reply of the <destination> to the <source> will be matched. This comes handy when SNORT tries to identify attack that may occur in sessions (Telnet, POP3, etc.)

The **option part** of the rules (or the body of the rule) can me modified depending on the alert plugins that the administrator is using. The standard format is that the option part is

a set of <option keyword> : <option arguments> pairs. For example “Priority : 1” is used by the ACID browser in order to set the priorities and colors of the alerts when the alert list page is displayed. Snorby is using the “Classify : DOS” in order to classify the events in certain classes. The **options separator** is used to separate the pairs.

If we want to translate the rule that we have as an example in figure 17, in natural language we can write “Find all the packets that are coming from every IP address in any port, except the 192,168.0.0 subnet, and are heading to the machine with the IP 192.168.0.33 in any port and log them with a message saying ‘outside finger attempt.’ ”

3.2.2. Challenges and benefits for rule based systems

Rule based systems are easy to use. Anyone with some technical background can install the software, download a rule set from the internet that is the closest to his needs and he is ready to go. The issues come when the administrator tries to configure the system to fit exactly to his needs. Rule based systems are highly customizable, a truth that is consider as a benefit but on the other side someone can easily find himself lost in the configurations.

Another drawback of the rule based systems is the number of the rules that someone has to have in his rule set in order to have a system which will provide safety for a decent number of attacks. For example SNORT version 2.9 (25 April 2014) the standard community rule set have more than 3 thousand rules. It is easy to understand that if the administrator needs to configure a service that means that he has to modify a rule or something is not performing as expected and needs debug of the rule set, then this process if not impossible it will be characterized are a really demanding one. The high number of rules also implies that any rule based system will need a power full machine for rule matching purposes. Of course by the time the rule set is getting bigger and bigger in numbers making difficult for the administrators to maintain it in a defragged format, which will be more time efficient for the detection engine to use it.

Rule based systems are really powerful with the known attacks. As soon as there is a signature for an attack and a rule to match with the signature the rule base system will always identify the attack. There are cases that the attackers are using variations of attacks and they try to hide the signature of the attack with several techniques but as we saw in the case of SNORT administrators have tools and ways to cope with that. Furthermore novel attacks or complex attacks some times are based in simple or known attacks which can be detected by misuse systems (Holm, 2014)

Rule based systems have a false negative rate close to zero, which makes them ideal for real life scenarios. On the other side rule based signature are unable and almost blind to detect new and previously unseen attacks. Rule based system are incapable to detect attacks that lack oh the attack’s signature, due to their nature. This fact forces administrators to keep updating constantly the signature rule set, and further more to keep using other security systems because they cannot rely only on them

3.3. Minnesota Intrusion Detection System (Anomaly detection)

In (Ertoz, et al.), the University of Minnesota and the Army High Performance Computing Research Center (AHPCRC) designed and developed together an intrusion detection system with the name MINDS (http://minds.cs.umn.edu/). The system belongs to the "anomaly detection" category, in which systems are able to identify a possible attack in the network with the help of numerous outlier detections techniques.

In contrast with SNORT, MINDS goal is not to identify the attack on the fly and execute the appropriate action immediately, but to give the ability to the network analyst to examine all the anomalous behavior further and generate valuable information about all the security incidents. MINDS is unable to operate on its own as an automated system, since the knowledge of the domain expert is a requirement.

As an input MINDS requires the net flows that will be examined. The information that is used for the analysis phase is only the base information that someone can extract from the TCP/IP layer. For example the source IP/Port, destination IP/Port, the payload of the packets etc. MINDS is not processing the content of the packet as SNORT does. Before the net flows are imported into MINDS the analyst is preprocessing the data and clear all the communications between trusted hosts or communication that are detected as malicious with other techniques.

After the analysis, the outcome is an order list with all the malicious connections order by the anomaly score. The anomaly score is the actual result of the analysis phase. In order for the anomaly score to be calculated, one or more outlier detection techniques are used as they were presented in the previous section "Outlier detection techniques". After the outcome has been produced the analyst needs to examine further the possible malicious connections and make the final decision, if the connection will be identified as an intrusion or not. If the connection is marked as malicious, then the knowledge database with the known attacks that was used in the first step to clear the unprocessed data is updated with the new information.

3.3.1. MINDS Architecture

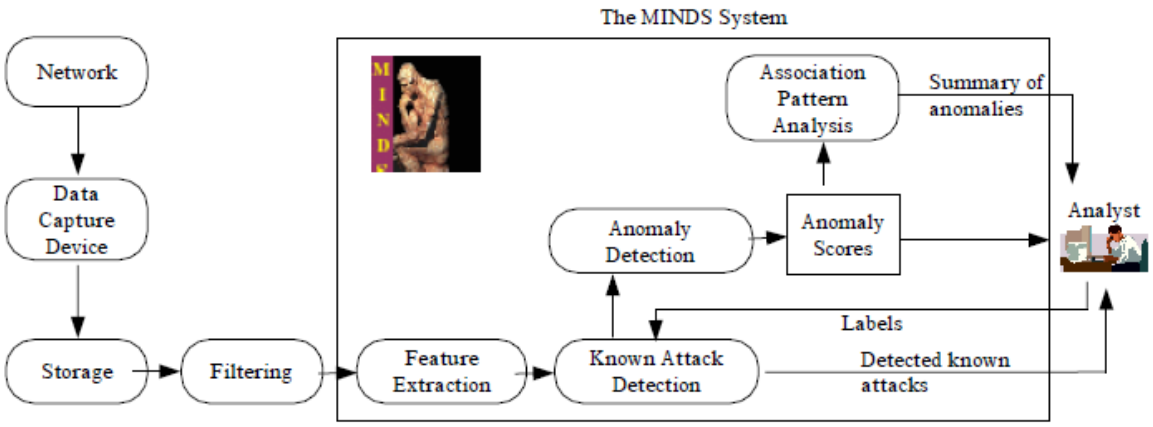


Figure 18: MINDS Architecture (Ertoz, et al.)

3.3.1.1. Feature Extraction**:** This module of MINDS is responsible to extract all the required information for further analysis. After the required attributes are extracted this module is also responsible to group and make the appropriate aggregations on the results. The basic features that are required are the number of flow to a unique destination IP address, the number of flows from a unique source IP address, the number of flows from a source IP to the same destination port with any destination IP address and the number of flows to the destination IP address from the same source port. The above features can be calculated in a time window frame, like in the last T seconds, but also in a numerical window frame, like in the last N flows. Calculating the features in these two different window frames gives the ability to identify variation of the attacks, when the attacker is using a sophisticating way to blend his attack with the normal flow of legitimate data. For example in a guessing password attack, or in the case of a port scanning attack, the attacker may slow down the rate of the requests that he sends out, with the hope that other users may interact with the same server, making the identification of his attack tougher for any IDS that might scan the network for malicious activities.

3.3.1.2. Known Attack Detection**:** The known attack detection module is pretty similar with any signature based intrusion detection system like SNORT. Actually in the case of MINDS the SNORT IDS has been employed. The detection module will identify the known attacks on the spot and eliminate them from the data before the analysis project. There are two main benefits for this action. First of all, by eliminating the known attacks, the number of the data objects to be examined is reduced, speeding up the overall process and reducing the final workload for the analyst. Excluding the known attacks from the process will not affect the quality of the results of the analysis because for the already known attacks there is no need for further investigation. Furthermore it will help the outlier technique to identify outliers properly because, by removing certain outliers from the data set, it magnifies the outlierness of the remaining outliers. The rule database is always

updated at the end of the process with the new results, increasing the number of the "known attacks" that can be identified of the fly.

3.3.1.3. Anomaly detection**:** This is the main module that performs the analysis. The input is the remaining data object of the known attack detection module combined with some configuration parameters provided by the analyst. The output is an ordered list with the flows that are considered as malicious, ordered by the malicious score. The algorithm that is used in MINDS is the LOF (local outlier factor) as it was described earlier. But in general any outlier technique can be used in this module.

3.3.1.4. Association pattern analysis**:** In this module MINDS with the help of the analyst will mind frequent association rules in order to update the rule database for the known attack detection module. Association rules will be mined from the anomalous class (for example as anomalous class we can define the first 10% of the data objects with the highest anomalous score) but also form the normal class (for example as a normal class we can define the last 30% of the data objects with the lowest anomalous score). Rules that will be mined from the normal class will describe the "trusted connections" and rules mined from the anomalous class will describe the "known attacks". Both of them will define the data objects that the known attacks detection engine will eliminate for the next round.

Sometimes mining association rules is a tough task, especially in the case of network data, due to fact of the class imbalance. But with the previous mentioned classes, the class imbalanced problem can be easily handled. MINDS is also grouping all the same association rules which have similar support and confidence percentages over the same attributes. By grouping the association rules, it reduces the number of the rules keeping almost the same accuracy. MINDS also offers a way to order the minded rules based on certain measures, giving the ability to the analyst to select the rules that will be imported in the known attack detection rule database.

3.3.2. Challenges and benefits for anomaly detection systems

Although most of the anomaly detection systems are configurable regarding the sensitivity of the outlier detection technique that is used, it is common to demonstrate a high false negative ratio especially when this ratio is compared to the relevant ratio of the signature based systems. As a result of the high negative false ratio, the workload for the analyst is multiplied. In many situations the analyst will have to spend time trying to clarify if the flagged data object is an intrusion attack of a system's fault.

Furthermore the results are not including an indication or the reasons behind the system's decision that a particular data object is considered as malicious. It is common for the analyst to investigate the outlierness of the malicious data object without any prior knowledge or any directions from the system making his task in many cases a lot tougher. Of course the need of a domain expert to value the results at the end, can be considered on its own as another drawback.

The nature of the network data can be described as high dimensional and heterogeneous, meaning that the technique that will be used should respect these unique characteristics and

cope with them in an efficient way. The majority of the proposed techniques are not optimized to handle in a proper manner the high dimensionality and the heterogenic data.

The proposed outlier detection techniques are focusing only in the outlier detection without taking under consideration the nature of the data and the field that will be applied. In this particular area of the IDS, it is critical to use a technique that will take under consideration the fact that the data are coming from a network stream. Techniques and algorithms that are exploding the characteristics that are describing network stream data will be ideal for intrusion detection systems.

On the other hand, the primary advantage of the outlier detection systems, is that they have the ability to detect novel attacks. As a novel attack can be considered a previously unseen attack that was developed recently (zero day attack) or a violation of a new security policy on the network, given the fact that the majority of the users respect the network security policies.

Another virtue for the outlier detection systems, is that detection of attack variants is also possible. If a malicious connection is flagged as an outlier, usually any variant of this attack will be flagged also. Although it is possible that in order to be detect all the possible variants that may exist in the data set, different attributes of the data or different aggregation functions should be used, meaning that more than one analysis iterations are required.

4. An experimental apply of the LOF

In this part of the paper we will try to identify the outliers that exists in a subset of the original KDD'99 data set. With the help of the WEKA open source software we will run an implementation of the LOF algorithm trying to identify the optimum threshold value for the specific dataset. The WEKA software, (http://www.cs.waikato.ac.nz/~ml/weka/) is a framework developed in Java by the university of Waikato, New Zealand. The version of WEKA that we have used in this paper is 3.7.11 (x64 developer's edition) and the version of the LOF plugin for WEKA was 1.0.4. The server configuration, that the experiment was contacted, consists of 16GB RAM and an Intel Xeon CPU E5-2670. The operating system was Microsoft Windows Server 2012 Standard edition.

4.1. The data set

The KDD'99 data set (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html) consists of laboratory data while simulating real life scenarios. The researchers captured all the network connections that were created in the lab in a time frame of 7 days. In this time interval, legitimate or normal connections were record and also malicious or bad connections. In the provided set there is an attribute called "label" which indicates the type of the connection. For the legitimate connections the label "normal was used" while for the malicious connections the name of the attack was used as a label.

The dataset contains 42 attributes including the basic features of the connection like duration of the connection, the protocol type, the total bytes, etc., but also features that

were recommended by an expert like the number of the previous unsuccessful connections in the last 2 seconds or the number of the previous ports that a request was sent in the last 2 seconds etc. The exact attribute description of the data set is provided in the comment section of the actual dataset file. In our case, due to the limitation of our hardware, we used 1% of the actual data (random selection amongst the records). We also removed all the data records with the label "Smurf" and "Neptune" because this kind of attacks produced a lot of data records overwhelming the normal label. All the attributes were used in the analysis except the "label" which as we mentioned before indicates the type of the attack.

No.	Label	Count
1	back	2203
2	teardrop	979
3	loadmodule	9
4	rootkit	10
5	phf	4
6	satan	1589
7	buffer_overflow	30
8	ftp_write	8
9	land	21
10	spy	2
11	ipsweep	1247
12	multihop	7
13	pod	264
14	perl	3
15	warezclient	1020
16	nmap	231
17	imap	12
18	warezmaster	20
19	portsweep	1040
20	normal	97277
21	guess_passwd	53

Figure19: Our dataset label attribute. The dataset consists of 106029 records in total

4.2. Results

We execute the LOF algorithm, while using the Euclidean distance as a measure to identify the neighborhoods, and using 10 as the minimum data objects for the lower bound and 40 for the upper bound. The implemented version of the LOF that we used provided the option to work in parallel process for some parts of the process so we set it up for maximum 6 parallel process. The total wall time of the process was approx. 20 minutes.

As expected the result was a file, the same as the input file, but with an extra attribute called LOF where the value of this attribute is the LOF score for every record. The minimum value of the LOF in our data set was 0 while the maximum was 22912449.74. The mean was 5965.289 but the standard deviation, which was equal to 209318.736, outlines the big deviation of the LOF amongst the data.

For an easier way to identify the optimum threshold of the LOF score in order to identify the outliers we separate the LOF score in 5 groups as they follow in table 1:

Table 1: LOF score groups		
Group name	Minimum LOF	Maximum LOF

0	0	1
1	>1	1.25
2	>1.25	1.50
3	>1.50	1.75
4	>1.75	2
5	>2	∞

For each group we provide the TP, TN, FP, FN values:

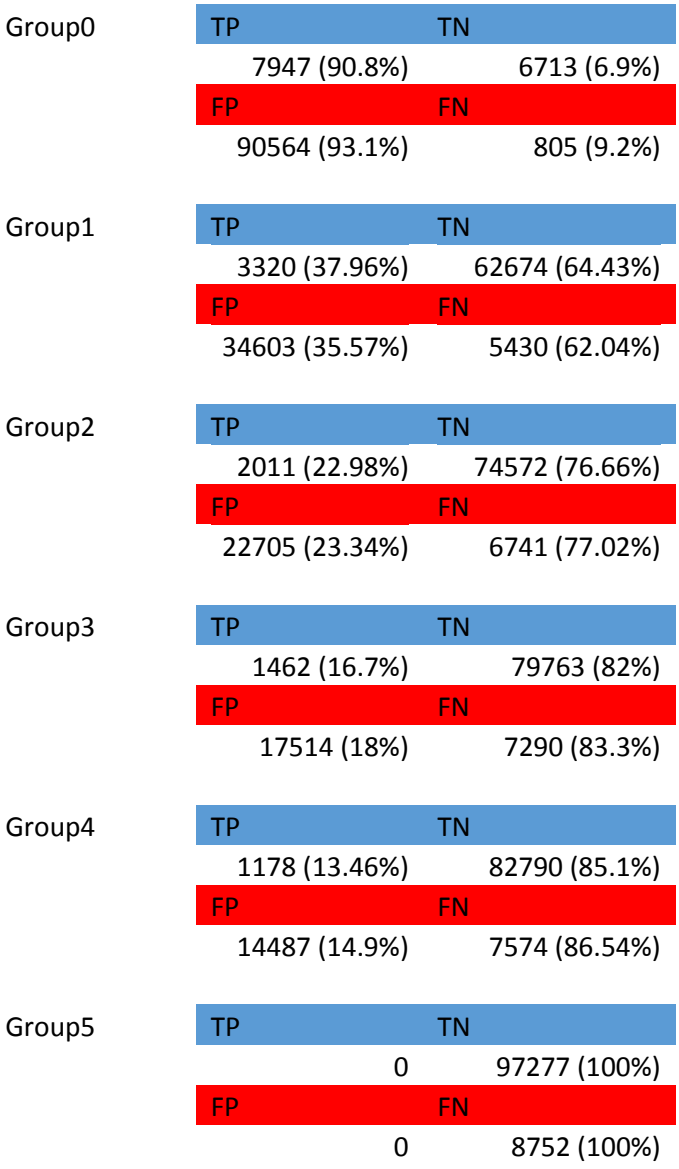


Figure 20: TP,TN,FP,FN values per group

By TP (true positive) we symbolize the number of the malicious connections that were identified successfully as non-legitimate in contrast with the FP (false positive) which is the number of the normal connections that were identified as malicious traffic. By TN (true negative) we symbolize the number of the legitimate connections correctly identified as

normal traffic in contrast with the FN (false negative) which is the number of malicious connections incorrectly identified as normal traffic.

Base on the above result the accuracy for every group will be as it follows:

Group name	Accuracy
0	0.138
1	0.622
2	0.722
3	0.765
4	0.792
5	0.917

The selection of a group in which the threshold will belong to is a tough decision. The optimum selection will be a group that will identify as many malicious connections (TP) as possible but at the same time keeping low the number of misclassified legitimate connections (FP). In our case the LOF algorithm performs poorly so the best selection will be group 1 or group 2 depending on the user's preferences.

4.3 Conclusion

In our case LOF performed really poorly. Although we removed two attacks (Smurf and Neptune) that were outnumbering the normal data objects, in favor of LOF result quality, still LOF was not able to distinguish the malicious from the normal traffic. In the appendix B, we can see that LOF was able to detect some particular attacks (for example PingOfDeath, Perl Attack, Phf/CGI request attack ,Imap Spy and more) with even more than 80% success rate using group 1 or even group 2. The real drawback in our experiment is the fact that LOF is identifying a lot of legitimate traffic as malicious.

The fact that as a distance measure for the calculation of the LOF we used the Euclidean distance, can affect dramatically the quality of the results since the quality of the Euclidean measure degrades as the number of the dimensionality increases (Aydin, Zaim, & Ceylan, 2009) (Lazarevic, Ozgur, Ertoz, Srivastava, & Kumar, 2005). Distance measures that respect the dimensionality should be used in case of an increased number of dimensions. The WEKA's LOF implementation currently supports only the Euclidean distance.

Furthermore different type of attacks need different attribute set to be examined. In our case we used the complete attribute data set, but the results will be different if we divide the whole process in smaller sub process which every sub process will have a subset of the original attribute set.

Finally the random selection of the 1% of the complete data set can play a significant role in the quality of the results. Our selection affected the "label" attribute balance in the dataset forcing us to remover the two attacks from the remaining dataset (Smurf, Neptune).

A better way for reducing the dataset is prosed in (Yang & Li, 2007).

5. Open issues and challenges

The first network intrusion detection systems have been proposed more than a decade ago, but in general they need time and evolution in order for network administrations to heavily rely on them. The majority of the IDS today are used by organizations with large scale networks, like universities or companies that the safety of their network is a critical business aspect. Smaller organizations will start using them when IDS reach the point where a systems administrator with no expertise on IDS can operate them flawlessly.

As we discussed earlier signature based systems have the ability to identify attacks in real time and actually this is the main reason that SNORT as an open source solution, is used in real scenarios today. In order for anomaly detection NIDS to be used widely in real life some enchantments are mandatory. First of all anomaly detection NIDS need to eliminate the high false negative percentage that usually demonstrate. Furthermore a reduction of human interaction is needed in the analysis phase. Organizations with no particular interest in the network security can find it difficult to have one or more employees committed in this task.

The majority of the proposed systems (Catania & Garino, 2012), (Daniel, Couto, & Lin, 2003), (Garcia-Teodoro, Diaz-Verdejo, Macia-Fernadez, & Vazquez, 2009), (Gogoi, Bhattacharyya, Borah, & Kalita, 2011) (Teodoro, Verdejo, Fernadez, & Vazquez, 2008) are using as a test or development data set the KDD'99 data set (more information can be found at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html) which contains lab data. The data set despite the fact that lacks from real life data, it also lacks from data that exist in our contemporary networks like data that are coming from wireless devices, or form new protocols like TCP/IP v6, or even new trends and technologies that users are using like torrent file exchange protocol or the "Tor project" (https://www.torproject.org/about/overview.html.en).

Although the majority of the proposed systems are working in batch mode meaning that they are not dealing with the live ongoing traffic of the network, the increase of the network speeds is an issue for the NIDS. Higher network speed means more information exchanged over the network which leads to more data objects to be analyzed by the network intrusion systems. The battle here is uneven since the industry, as expected, is using more resources to increase the network speeds than the NIDS speed. Of course the constant increase of the network speeds is an issue that affects primarily the signature based intrusion detection systems. Hardware string pattern recognition techniques can help to that direction as (Jaic, Smith, & Sarma, 2014) suggest.

With new technologies arriving in our lives every day we can forecast that intrusion detections systems will evolve and contribute to the safety in every one of them. New categories for intrusion detection systems are constantly formed like Cloud Based Intrusion Detection Systems (CBIDS) (Patel, Taghavi, Bakhtiyari, & Junior, 2013), (Modi, et al., 2012), or wireless intrusion detection systems (WIDS) or detection systems that are focusing on virtual machines (Liao, Lin, Lin, & Tung, 2013).

6. Appendix

6.1 Appendix A

The comments from the KDD'99 actual file:

%Data set for KDD Cup 1999

%Modified by TunedIT (converted to ARFF format)

%http://www.sigkdd.org/kddcup/index.php?section=1999&method=info

%This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between ``bad" connections, called intrusions or attacks, and "good" normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

%The training and test datasets are also available in the UC Irvine KDD archive.

%KDD Cup 1999: Tasks

%This document is adapted from the paper Cost-based Modeling and Evaluation for Data Mining With Application to Fraud and Intrusion Detection: Results from the JAM Project by Salvatore J. Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K. Chan.

%Intrusion Detector Learning

%Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between ``bad" connections, called intrusions or attacks, and ``good" normal connections.

%The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The 1999 KDD intrusion detection contest uses a version of this dataset.

%Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

%The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

%A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

%Attacks fall into four main categories:

% * DOS: denial-of-service, e.g. syn flood;

% * R2L: unauthorized access from a remote machine, e.g. guessing password;

% * U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow" attacks;

% * probing: surveillance and other probing, e.g., port scanning.

%It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the "signature" of known attacks can be sufficient to catch novel variants. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.

%Derived Features

%Stolfo et al. defined higher-level features that help in distinguishing normal connections from attacks. There are several categories of derived features.

%The ``same host'' features examine only the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc.

%The similar ``same service'' features examine only the connections in the past two seconds that have the same service as the current connection.

%"Same host" and "same service" features are together called time-based traffic features of the connection records.

%Some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Therefore, connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. This yields a set of so-called host-based traffic features.

%Unlike most of the DOS and probing attacks, there appear to be no sequential patterns that are frequent in records of R2L and U2R attacks. This is because the DOS and probing attacks involve many connections to some host(s) in a very short period of time, but the R2L and U2R attacks are embedded in the data portions of packets, and normally involve only a single connection.

%Useful algorithms for mining the unstructured data portions of packets automatically are an open research question. Stolfo et al. used domain knowledge to add features that look for suspicious behavior in the data portions, such as the number of failed login attempts. These features are called ``content'' features.

%A complete listing of the set of features defined for the connection records is given in the three tables below. The data schema of the contest dataset is available in machine-readable form.

%feature name description type

%duration length (number of seconds) of the connection continuous

%protocol_type type of the protocol, e.g. tcp, udp, etc. discrete

%service network service on the destination, e.g., http, telnet, etc. discrete

%src_bytes number of data bytes from source to destination continuous

%dst_bytes number of data bytes from destination to source continuous

%flag normal or error status of the connection discrete

%land 1 if connection is from/to the same host/port; 0 otherwise discrete

%wrong_fragment number of ``wrong'' fragments continuous

%urgent number of urgent packets continuous

%Table 1: Basic features of individual TCP connections.

%feature name description type

%hot number of ``hot'' indicators continuous

%num_failed_logins number of failed login attempts continuous

%logged_in 1 if successfully logged in; 0 otherwise discrete

%num_compromised number of ``compromised'' conditions continuous

%root_shell 1 if root shell is obtained; 0 otherwise discrete

%su_attempted 1 if ``su root'' command attempted; 0 otherwise discrete

%num_root number of ``root'' accesses continuous

%num_file_creations number of file creation operations continuous

%num_shells number of shell prompts continuous

%num_access_files number of operations on access control files continuous

%num_outbound_cmds number of outbound commands in an ftp session continuous

%is_hot_login 1 if the login belongs to the ``hot'' list; 0 otherwise discrete

%is_guest_login 1 if the login is a ``guest''login; 0 otherwise discrete

%Table 2: Content features within a connection suggested by domain knowledge.

%feature name description type

%count number of connections to the same host as the current connection in the past two seconds continuous

%Note: The following features refer to these same-host connections.

%serror_rate % of connections that have ``SYN'' errors continuous

%rerror_rate % of connections that have ``REJ'' errors continuous

%same_srv_rate % of connections to the same service continuous

%diff_srv_rate % of connections to different services continuous

%srv_count number of connections to the same service as the current connection in the past two seconds continuous

%Note: The following features refer to these same-service connections.

%srv_serror_rate % of connections that have ``SYN'' errors continuous

%srv_rerror_rate % of connections that have ``REJ'' errors continuous

%srv_diff_host_rate % of connections to different hosts continuous

%Table 3: Traffic features computed using a two-second time window.

%http://www.sigkdd.org/kddcup

6.2. Appendix B

Pivot table show the the count of every “labe” attribute and the participation percentage per group:

Row Labels	Count of Anomalous_Group	Percentage

0	**7518**	
back	344	0.156150704
ipsweep	108	0.086607859
nmap	83	0.359307359
normal	6713	0.069009118
portsweep	144	0.138461538
satan	63	0.039647577
teardrop	37	0.037793667
warezclient	26	0.025490196
1	**60586**	
back	1033	0.468906037
buffer_overflow	3	0.1
guess_passwd	35	0.660377358
imap	2	0.166666667
ipsweep	737	0.591018444
land	17	0.80952381
nmap	87	0.376623377
normal	55961	0.575274731
pod	146	0.553030303
portsweep	202	0.194230769
rootkit	2	0.2
satan	875	0.550660793
teardrop	741	0.756894791
warezclient	745	0.730392157
2	**13209**	
back	337	0.152973218
buffer_overflow	1	0.033333333
guess_passwd	3	0.056603774
ipsweep	161	0.129109864
land	1	0.047619048
multihop	2	0.285714286
nmap	4	0.017316017
normal	11898	0.122310515
pod	27	0.102272727
portsweep	275	0.264423077
satan	250	0.157331655
teardrop	119	0.121552605
warezclient	128	0.125490196
warezmaster	3	0.15
3	**5740**	
back	129	0.058556514
buffer_overflow	4	0.133333333
ftp_write	2	0.25

	guess_passwd	2	0.037735849
	imap	2	0.166666667
	ipsweep	73	0.058540497
	land	2	0.095238095
	loadmodule	1	0.111111111
	multihop	1	0.142857143
	nmap	4	0.017316017
	normal	5191	0.053363077
	pod	7	0.026515152
	portsweep	77	0.074038462
	rootkit	3	0.3
	satan	158	0.099433606
	teardrop	50	0.051072523
	warezclient	34	0.033333333
4		**3311**	
	back	95	0.043123014
	buffer_overflow	2	0.066666667
	guess_passwd	1	0.018867925
	imap	2	0.166666667
	ipsweep	46	0.036888532
	land	1	0.047619048
	nmap	1	0.004329004
	normal	3027	0.031117325
	pod	2	0.007575758
	portsweep	48	0.046153846
	satan	46	0.028949025
	teardrop	20	0.020429009
	warezclient	20	0.019607843
5		**15665**	
	back	265	0.120290513
	buffer_overflow	20	0.666666667
	ftp_write	6	0.75
	guess_passwd	12	0.226415094
	imap	6	0.5
	ipsweep	122	0.097834804
	loadmodule	8	0.888888889
	multihop	4	0.571428571
	nmap	52	0.225108225
	normal	14487	0.148925234
	perl	3	1
	phf	4	1
	pod	82	0.310606061
	portsweep	294	0.282692308

rootkit	5	0.5
satan	197	0.123977344
spy	2	1
teardrop	12	0.012257406
warezclient	67	0.065686275
warezmaster	17	0.85
Grand Total	**106029**	

7. References

Aggarwal, C. C. (2013). *Outlier Analysis.* New York USA: Springer New York Heidelberg Dordrecht London.

Aydin, A. M., Zaim, H. A., & Ceylan, G. K. (2009). A hydrid intrusion detection system design for computer network security. *Elsevier, Computers and electrical engineering*, 10.

Breunig, M. M., Kriegel, H.-P., Raymond, N. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. *International conference on Managment of Data.* Dallas TX: ACM SIGMOD.

Catania, C. A., & Garino, C. G. (2012). Automatic network intrusion detection: Current techniques and open issues. *Elsevier,Computers and Electrical Engineering*, 11.

Chen, C.-M., Chen, Y.-L., & Lin, H.-C. (2009). An efficient network intrusion detection. *Elsevier,Computer communications*, 8.

Daniel, B., Couto, J., & Lin, J.-L. (2003). *Bootstrapping a Data Mining Intrusion Detection System.* Melbourne, Florida, USA: ACM.

Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.-N., Kumar, V., Srivastava, J., & Dokas, P. (n.d.). *Minnesota Intrusion Detection System.* AHPCRC, University Of Minesota.

Estevez-Tapiador, J. M., Teodoro, P. G., & Vardejo, J. E. (2004). Anomaly detection methods in wired networks: A survey and taxonomy. *Elsevier, Computer communications*, 16.

Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernadez, G., & Vazquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 18-28.

Gogoi, P., Bhattacharyya, D. K., Borah, B., & Kalita, J. K. (2011). A survey of outlier detection methods in network anomaly identification. *IEEE,The computer journal*, 19.

Holm, H. (2014). Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter? *Internation conference on System Science.* Hawaii: IEEE.

Jaic, K., Smith, M. C., & Sarma, S. N. (2014). *A Practical Network Intrusion Detection System for Inline FPGAs on 10GbE Netwrok Adapters.* South Carolina, USA: IEEE.

Jiawei, H., Kamber, M., & Pei, J. (2012). *DATA MINING Conepts and techniques.* Waltham, USA: Elsevier.

Knorr, E. M., & Raymond, N. (1997). *A unified notion of outliers: Properties and computation.* American Association for Artificial Intelligence.

Knorr, E. M., Raymond, N. T., & Tucakov, V. (2000). Distance-based outliers: algorithms and applications. *The VLDB journal*, 273-253.

Kriegel, H.-P., Kroger, P., Schudert, E., & Zimek, A. (2009). *LoOP: Local Outlier Probabilities.* Honk Kong, China: CIMK.

Kriegel, H.-P., Schubert, M., & Zimek, A. (2008). *Angle-Based Outlier Detection in High-dimensional Data.* Las Vegas, Nevada, USA: ACM.

Lazarevic, A., & Kumar, V. (2005). *Feature Bagging for Outlier Detection.* Chigago Illinois USA: ACM.

Lazarevic, A., Ozgur, A., Ertoz, L., Srivastava, J., & Kumar, V. (2005). *A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection.* Minneapolis: Department of Computer Science Department, University of Minnesota.

Lee, W., & Stolfo, S. J. (2000). *Data mining approaches for intrusion detection.* New York: Columbia university, department of Computer Science.

Li, Y., & Guo, L. (2007). An efficient network anomaly detection scheme based on TCM-KNN algorithm and data reduction mechanism. *Workshop on information assurance* (pp. 221-227). New York: IEEE.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., & Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Network and Computer Applications*, 16-24.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *Internation conference on Data Mining.* Omaha, Nebraska, USA: IEEE.

Manikopoulos, C., & Papavassiliou, S. (2002, October). Network intrusion and fault detection: A statistical anomaly approach. *Telecommunications network security*, pp. 76-82.

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. (2012). A survey of intrusion detection techniques in cloud. *Elsevier, Journal of network and computer applications*, 16.

Modi, C., Patel, D., Borsaniya, B., Patel, A., Patel, H., & Rajarajan, M. (2013). A survey of intrusion detection techniques in Cloud. *Network and Computer Applications*, 42-57.

Papadimitriou, S., Kitagawa, H., & Gibbons, P. B. (2003). LOCI: Fast Outlier Detection using the Local Correlation Integral. *International conference on Data engineering.* Bangalore, India: IEEE.

Patcha, A., & Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *ScienceDirect, Computer networks,* 23.

Patel, A., Taghavi, M., Bakhtiyari, K., & Junior, J. C. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. *Network and Computer Applications*, 25-41.

Qiu, H., Eklund, N., Hu, X., Yan, W., & Lyer, N. (2008). Anomaly detection using data clustering and neural networks. *InternationalJoint Conference on Neural Networks (IJCNN 2008)* (pp. 3727-3632). IEEE.

Rehman, R. U. (2003). *Intrusion Detection Systems with Snort.* New Jersey: Prentice Hall.

Shelly, X. W., & Banzhaf, W. (2009). The use of computational intelligence in intrusion detection systems: A review. *Elsevier,Applied soft computing,* 35.

Sobh, T. S. (2005). Wired and wireless intrusion detection system: Classifications, good characteristics and state-of-the-art. *Elsevier, Computer standards & interfaces,* 25.

Teodoro, G. P., Verdejo, D. J., Fernadez, M. G., & Vazquez, E. (2008). Anomaly-based network intrusion detection: Techniques systems and challenges. *Elsevier, Computers & security,* 11.

Yang, L., & Li, G. (2007). An Efficient Network Anomaly Detection Sheme Based on TCM-KNN Algorithm and Data Reduction Mechanism. *Workshop on Information Assurance* (pp. 221-227). West Point, NY: IEEE.

Zengyou, H., Xiaofei, X., & Shengchun, D. (2003). Discovering cluster-based local outliers. *Pattern recognition letters*, 1641-1650.