Incremental Anomaly Detection Using Two-Layer Cluster-based Structure

by

Elnaz Bigdeli

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the degree of Doctor of Philosophy in Computer Science

Ottawa-Carleton Institute of Computer Science
School of Electrical Engineering and Computer Science
University of Ottawa

Abstract

Anomaly detection algorithms face several challenges, including processing speed and dealing with noise in data. In this thesis, a two-layer cluster-based anomaly detection structure is presented which is fast, noise-resilient and incremental. In this structure, each normal pattern is considered as a cluster, and each cluster is represented using a Gaussian Mixture Model (GMM). Then, new instances are presented to the GMM to be labeled as normal or abnormal.

The proposed structure comprises three main steps. In the first step, the data are clustered. The second step is to represent each cluster in a way that enables the model to classify new instances. The Summarization based on Gaussian Mixture Model (SGMM) proposed in this thesis represents each cluster as a GMM.

In the third step, a two-layer structure efficiently updates clusters using GMM representation while detecting and ignoring redundant instances. A new approach, called Collective Probabilistic Labeling (CPL) is presented to update clusters in a batch mode. This approach makes the updating phase noise-resistant and fast. The collective approach also introduces a new concept called 'rag bag' used to store new instances. The new instances collected in the rag bag are clustered and summarized by GMMs. This enables online systems to identify nearby clusters in the existing and new clusters, and merge them quickly, despite the presence of noise to update the model.

An important step in the updating is the merging of new clusters with existing ones. To this end, a new distance measure is proposed, which is a modified Kullback-Leibler distance between two GMMs. This modified distance allows accurate identification of nearby clusters. After finding neighboring clusters, they are merged, quickly and accurately. One of the reasons that GMM is chosen to represent clusters is to have a clear and valid mathematical representation for clusters, which eases further cluster analysis.

In most real-time anomaly detection applications, incoming instances are often similar to previous ones. In these cases, there is no need to update clusters based on duplicates, since they have already been modeled in the cluster distribution. The two-layer structure is responsible for identifying redundant instances. In this structure, redundant instance are ignored, and the remaining new instances are used to update clusters. Ignoring redundant instances, which are typically in the majority, makes the detection phase fast.

Each part of the general structure is validated in this thesis. The experiments include, detection rates, clustering goodness, time, memory usage and the complexity of the algorithms. The accuracy of the clustering and summarization of clusters using GMMs is evaluated, and compared to that of other methods. Using Davies-Bouldin (DB) and Dunn indexes, the distances for original and regenerated clusters using GMMs is almost zero with SGMM method while this value for ABACUS is around 0.01. Moreover, the results show that the SGMM algorithm is 3 times faster than ABACUS in running time, using one-third of the memory used by ABACUS.

The CPL method, used to label new instances, is found to collectively remove the effect of noise, while increasing the accuracy of labeling new instances. In a noisy environment, the detection rate of the CPL method is 5% higher than other algorithms such as one-class SVM. The false alarm rate is decreased by 10% on average. Memory use is 20 times lesser that that of the one-class SVM.

The proposed method is found to lower the false alarm rate, which is one of the basic problems for the one-class SVM. Experiments show the false alarm rate is decreased from 5% to 15% among different datasets, while the detection rate is increased from 5% to 10% in different datasets with two-layer structure. The memory usage for the two-layer structure is 20 to 50 times less than that of one-class SVM. One-class SVM uses support vectors in labeling new instances, while the labeling of the two-layer structure depends on the number of GMMs. The experiments show that the two-layer structure is 20 to 50 times faster than the one-class SVM in labeling new instances. Moreover, the updating time of two-layer structure is 2 to 3 times less than one-layer structure. This reduction is the direct result of ignoring redundant instances and using two-layer structure.

I dedicate this thesis to my beloved family

My mother **Soraya**, My father **Kazem**, My sister **Solmaz**, My brother **Mohammad**

For their endless love, support and encouragement

Acknowledgments

Though only my name appears on the cover of this dissertation, a great many people have contributed to its production. I owe my gratitude to all those people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever.

My deepest gratitude is to my supervisors, Prof. Stan Matwin and Prof. Bijan Raahemi. I have been amazingly fortunate to have supervisors who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. They taught me how to question thoughts and express ideas. Their patience and support helped me overcome many crisis situations and finish this dissertation.

My colleague and friend, Dr. Mahdi Mahommadi, has been always there to listen and give feedback. I am deeply grateful to him for the long discussions that helped me sort out the technical details of my work. I am also thankful to him for encouraging the use of correct grammar and consistent notation in my writings, and for carefully reading and commenting on countless revisions of this manuscript.

To my friend Zeinab, thank you for listening, offering me advice, and supporting me through this journey. Your support and care helped me overcome setbacks and stay focused on my graduate study. Thanks to Najmeh , Aali , Faezeh and Hossein, Mehrnoosh , Davood and Aida who are more than friends to me with their endless support and love; I would never forget your help and those beautiful moments I shared with you. Thanks to my lovely friend Mitra whose love and support far away from Iran, is always with me.

Thanks to all KDD lab members, Fatmeh, Arina, Wael and Fadwa, Parinaz for their friendship, and I deeply appreciate their support and belief in me.

Thanks to my brilliant friend Arman, for always being there as a friend and colleague, with constant motivation and support from whom I have learned many things through these years. Thanks to my friend Ebrahim, whose support and help even from oversees was with me all these years in bad and good moments.

I am very grateful to all the people I have met along the way, who have contributed to the development of my research. In particular, I am indebted to Dr. Mohsen Afsharchi who helped me pave the first steps in research, without the support and motivation from whom I would never have started my PhD journey.

Finally, I would like to thank the rest of my thesis committee: Prof. Sabine McConnell , Prof. Herna Viktor, Prof. Diana Inkpen and Prof. Babak Esfandiar, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

Most importantly, none of this would have been possible without the love and patience of my family. My family to whom this dissertation is dedicated to, has been a constant source of love, concern, support and strength all these years; I would like to express my heart-felt gratitude to them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Anomaly detection involves determining deviation from normal behaviour [Han et al., 2006] [Chandola et al., 2009]. Anomalies exhibit behaviors that do not follow the normal patterns. Other terms like outliers, exceptions, faults, and defects are used in different applications to refer to anomalies.

Anomaly detection methods are used in many applications, such as network security and fraud detection. Some methods are generic, and some are application-specific. In network security, anomaly detection includes the detection of possible network attacks [Teodoro et al., 2009]. With the increasing prevalence of such attacks in online applications, a reliable and fast approach is needed to cope with difficulties in these applications [Zhou et al., 2010] [Teodoro et al., 2009]. For credit card fraud detection, an anomaly detection system monitors all transactions and reports unusual transactions to the credit card company [Van and Faisal, 2011]. In medical applications, anomaly detection techniques are used to find abnormal activity in the human body, such as that caused by malignant tumors [Lin et al., 2005]. Anomalous patterns can cause serious and irreversible damage, and immediate action is needed to cope with them.

1.1 Motivation and Objectives

Anomaly detection systems must not only find previously known anomalies, but also new unknown anomaly patterns [Chandola et al., 2009]. Most anomaly detection approaches are signature-based, a method which is not able to detect new attacks. Signature-based methods work by matching incoming behavior with previously known signatures of attacks. As such, they are able to detect a number of known attacks. To be able to detect unknown attacks, a boundary can be created for each normal pattern. All instances outside the boundary of normal patterns are deemed to be anomalous and detection does not depend on a limited set of restricted rules.

The issue in anomaly detection is a lack of labeled data [Smith et al., 2002]. In all applications what is mainly known are normal behaviours along with a limited number of anomalous behaviors. In most anomaly detection applications, the data are imbalanced and unlabeled. Because of this limitation, supervised methods are not applicable; instead unsupervised and semi-supervised methods are better alternatives. Therefore, the attention draws toward cluster-based approaches.

In cluster-based approaches for anomaly detection, normal behaviors are modeled as a set of clusters, without requiring any previous knowledge of anomalous instances. The challenge is determining whether the new instances should be classified as normal or anomaly. In spherical-shape clustering methods, each cluster is represented by a radius and a center. For each new instance, the distance of the new instance to the center of a cluster is calculated. If the distance is less than the radius of the cluster, the instance is considered to be normal; otherwise, it is abnormal.

There are several problems with this approach. In a system with a set of normal patterns, detection of the number and shape of the patterns is challenging. Also, in many applications, finding a rigid boundary for a normal class is difficult, and current approaches do not preserve the exact shape or even a good approximation of the cluster. Moreover, in most applications such as network security, attacks appear similar to normal patterns, and it is difficult to distinguish them.

Unsupervised methods, such as those involving clustering are among the best options for detecting boundaries of normal patterns [Mohammadi et al., 2014]. However, normal patterns change over time, and a roster of recognized normal patterns may not be valid in the future. This introduces a requirement for an incremental structure to able to update normal patterns.

Another challenge in this area is the presence of noise. Although anomalous instances differ from normal ones, they do not show the random behavior of noise instances. Noise is random behavior in data that does not follow any pattern, while anomalous behaviors are not random and they follow specific patterns [Han et al., 2006]. Noisy instances are an inseparable part of real-world datasets. They affect the border of normal and abnormal instances to the point that it becomes difficult to distinguish noisy instances from abnormal instances. A major responsibility of anomaly detection methods is to mitigate the effect of noise on system performance.

1.2 Contributions

In this thesis, a two-layer cluster-based structure is presented to deal with the aforementioned problems. In the two-layer structure, normal behaviors are represented by a set of clusters, and all instances contained within the boundaries of the clusters are considered to be normal. These clusters are employed to label incoming instances as either normal or anomaly.

The structure has five main functional phases:

- Clustering data
- Representing clusters as Gaussian Mixture Models(GMMs)
- Collecting data in a 'rag bag' and labeling them collectively
- Updating clusters incrementally
- Creating coarse and fine-level clusters

Normal instances are clustered using a clustering algorithm, with each cluster represented by a GMM. Each new instance is fed into the GMMs of the clusters, and if the assessed membership value to one of the clusters exceeds a threshold, the instance is labeled as normal; otherwise, it is deemed anomaly.

In the third phase, new instances are collected in a storage called a 'rag bag' and clustered. These clusters are represented by GMMs consistent with the representation of current clusters, which makes the updating faster and easier.

To make this updating phase faster, the clusters are represented in two levels, coarse and fine. The coarse level is more general, with a less complex representation of clusters in terms of GMMs. The fine level is more detailed, with specific representation of clusters. Instances with high membership value for a cluster are ignored, since they are already represented in the current distribution of clusters. This decision is made in the coarse level, with less computation required. Further analysis is done in the fine level to specify the membership value for new instances.

Cluster-based representation brings many benefits for anomaly detection systems. Since there are no labeled data in anomaly detection, clustering of normal behaviors is effective. Nonetheless, clustering has its own challenges and problems. Many existing cluster-based methods use non-convex clustering approaches such as k-means, which do not preserve the original shape of each cluster. Furthermore, the number of clusters has to be determined, despite there being no information about it available beforehand. To solve these problems, this thesis proposes using a Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering approach, which is able to find arbitrary-shape clusters [Ester et al., 96]. DBSCAN fulfills two crucial requirements of anomaly detection methods:

- More precise boundaries for clusters are defined which improves anomaly detection accuracy.
- There is no need to know the number of clusters ahead of time. This is an essential feature, since there is limited knowledge about the number of normal profiles and their distributions.

Arbitrary-shape clustering methods are able to create an accurate boundary for clusters of any shape. However, there is a need to represent a cluster in a way that enables proper labeling of new instances. To label a new instance, the distance of the new instance to each cluster center is calculated. Using a threshold, the instance is labeled as being either inside or outside of the cluster. This approach is simplistic and inaccurate, since it models the cluster border with a simple circle, ignoring the original shape of the cluster. Finding an optimum way to represent clusters is another challenge for cluster-based methods. All cluster attributes need to be preserved, such as shape, instance distribution and instance scattering.

A solution for this problem is to preserve the whole set of instances that a cluster comprises, and find the distance of a new instance to all cluster members. In this way, all instances are considered in assessing the distance. This approach improves the accuracy by not limiting the shape of cluster to a circle. It is important to note that keeping all instances is a time and memory-consuming process, and it is not feasible in many applications.

To mitigate this performance problem, a summarization approach is presented in Chapter 4, which summarizes the data in each cluster. This approach preserves all data characteristics.

- The proposed approach, Summarization based on GMM (SGMM), represents each cluster as a GMM. SGMM is able to summarize clusters in such a way that all cluster features (such as the cluster shape, instance scattering and instance distribution) are preserved using a probabilistic model. Compared to other models, by SGMM the membership values of the instances to a cluster are calculated with a relatively low computational cost. However, finding a proper GMM for a cluster remains a challenging task. The SGMM algorithm finds the number of GMM components automatically, and determines all GMM parameters.

Typically normal behavior changes over time, and so an updating phase is an essential system feature. The two-layer structure presented in this thesis enables updating current clusters with newly extracted patterns. However, with rapid arrival rates of incoming instances in real-time applications, updating current clusters for every new instance is not efficient.

- A novel approach is proposed using the two-layer structure to perform the updating task. In two-layer structure instead of updating clusters with each new instance, new instances are collected in a rag bag. The new approach is described in Chapter 5 called Collective Probabilistic Labeling (CPL). New instances are collected and clustered. Each cluster is then represented by a GMM using the SGMM approach. The new clusters are merged with the closest current cluster. To find the

the closest cluster, a new distance measure is proposed in Chapter 6, to measure the distance of two GMMs.

After identifying nearby clusters, the next step is to update the clusters using the newly created clusters. Since everything is presented in the form of GMMs, updating clusters is quick. Each cluster is a GMM with a set of parameters. With every new GMM created in the rag bag the parameters of current clusters are updated, as it is described in Chapter 6.

- In the two-layer structure after finding nearby clusters, new GMMs are merged with current GMMs. GMM mean, covariance and weights are updated: this updating is a fast operation as it involves only simple calculations.

In online systems there tend to be many redundant instances: most of the data that arrive at the system every second are similar to previously seen data. There is no point in updating the system with these instances, since the data distribution already includes these data. With this in mind, the proposed system removes those instances that have a high membership value for one of the clusters; there is no need to update the cluster with that instances. This approach saves time and space for the updating step.

- The proposed two-layer structure is designed to reduce computation needs. The coarse-level, with less computation required, decreases the time of assigning the membership value. The redundant instances mentioned above are mainly labeled in the coarse level, thereby with less cost.

- The fine level is used to investigate instances that are new where the determination of their membership value needs more detailed calculation.

The two-layer approach presents an incremental structure that solves the mentioned issues. The proposed structure is faster and more accurate than other anomaly detection approaches. The closest performance to the proposed method was for one-class SVM, that has training time of between $O(n^2)$ and $O(n^3)$ based on optimization and kernel function, while the complexity of our algorithm is $O(n^2)$ in the worst-case, where n is the size of dataset.

1.2.1 Evaluation of the Method

Both synthetic and real datasets are used in the experiments. The real datasets include datasets from the UCI repository as well as a real network datasets collected in the lab. Experimental results are obtained with Matlab running on a machine with an Intel 3.4 GHz CPU and 4GB of memory. The algorithms are evaluated based on these criteria:

- The accuracy of the anomaly detection method is evaluated based on ROC of false alarm and detection rates. The ROC curve is created by changing the input parameters for each algorithm.
  - The Detection Rate (DR) shows percentages of anomalous instances that are detected correctly. The False Alarm rate (FA) is the proportion of normal instances incorrectly deemed abnormal .
- The clustering 'goodness' is another factor considered to verify that the cluster accuracy is not decreased over time by the incremental approach. Dunn [k. Dunn and Dunn, 1974] and Davies-Bouldin (DB) [Davies and Bouldin, 1979] are two indexes used to evaluate the goodness of clusters.

- Memory usage of the algorithms, based on the number of parameters that each method preserves for a trained model.

- Running time of algorithms for clustering and cluster summarization.

- The computational complexity of the algorithms for the training and testing phases is another measure used to compare algorithms.

In each dataset, some classes are chosen as normal and the rest as abnormal. The abnormal instances are then considered as the testing dataset. Then, normal classes are divided into two parts, one of which is the training set with the rest is added to the testing set. This means that the training dataset consists only of normal instances, but the testing dataset is a combination of normal and abnormal instances. The proposed algorithms are compared with prominent anomaly detection algorithms of five different categories:

- **Proximity based methods**: Local Outlier Factor (LOF) [Breunig et al., 2000]

- **Cluster-Based Methods**: Cluster Based Local Outlier Factor (CBLOF) [He et al., 2003]

- **Classifcation**: One-class SVM [Scholkopf et al., 2001]

- **Artificial immune system**: positive selection [Ebner et al., 2002], real value negative selection [Ji and Dasgupta, 2004], Simple Probabilistic Artificial Immune system approach (SPAI) [Mohammadi et al., 2014]

- **Statistical methods** : Gaussian Mixture Model (GMM) [Hajji, 2005]

1.2.2 Structure of the Thesis

The structure of thesis is as follows. In Chapter 2, anomaly detection methods and recent advances in this area are reviewed. The methods are grouped into six categories, and the most influential approaches in each category are described in detail. In Chapter 3, the general structure of two-layer cluster-based anomaly detection method is introduced; all components of the two-layer structure are discussed in this chapter as well.

In Chapter 4, the SGMM is explained in detail. Some experimental results are presented in this chapter, to clarify the effectiveness of using the SGMM in summarizing the data and labeling new incoming instances.

Chapter 5 presents the collective labeling step, done to label and update clusters, which is one of the main parts of the two-layer structure. Issues and problems associated with collective labeling are discussed in this chapter, and experimental results are presented to show the validity of collective labeling.

Chapter 6 describes the GMM-based incremental approach to update clusters. Details of the updating stage and the effect of collective updating are discussed in this chapter.

Chapter 7 synthesizes the components of the two-layer structure, and presents the general two-layer structure based on the proposed approaches from previous chapters. In Chapter 7 a description is given of how clusters can be represented in two levels; coarse and fine. The capability to ignore redundant instances, and remove noise effects are discussed in this chapter. The experimental results cover extensive testing of the two-layer structure. Finally, the conclusion and proposals for future work are presented in Chapter 8.

1.3 Published Papers

- A fast and noise resilient cluster-based anomaly detection [E.Bigdeli et al., 2015a]
- Incremental Cluster Updating Using Gaussian Mixture Model [E.Bigdeli et al., 2015c]
- A Fast Noise Resilient Anomaly Detection using GMM-based Collective Labelling [E.Bigdeli et al., 2015b]
- Summarizing Arbitrary shape clustering using Guassian mixture model [E.Bigdeli et al., 2014b]
- Cluster Summarization with Dense Region Detection [E.Bigdeli et al., 2014a]

Chapter 2

Literature Review

An anomaly is defined as a significant deviation from behaviors that are normal patterns. There are three different types of anomalies; individual, contextual, and collective.

Individual Anomalies

This type of anomaly can be detected by comparing an individual instance with the rest of data. As depicted in Figure 2.1, A_1, A_2, A_3 and A_4 represent individual anomalies, as they are far from the area of normal instances, N_1, N_2, N_3. Detecting individual anomalies is easier than a group of anomalous instances and most algorithms detect individual anomalies.

Contextual Anomalies

A contextual or conditional anomaly is defined as anomaly based on a specific condition or context. A simple example is seasonal outdoor temperatures. A temperature of $28C$ is normal during summer but it would be considered abnormal during winter. Another example can be seen in a person's annual credit card activity. Each person spends an average amount during most of the year, while expenses increase during vacation times like Christmas. A high volume of transactions is expected during holidays, since

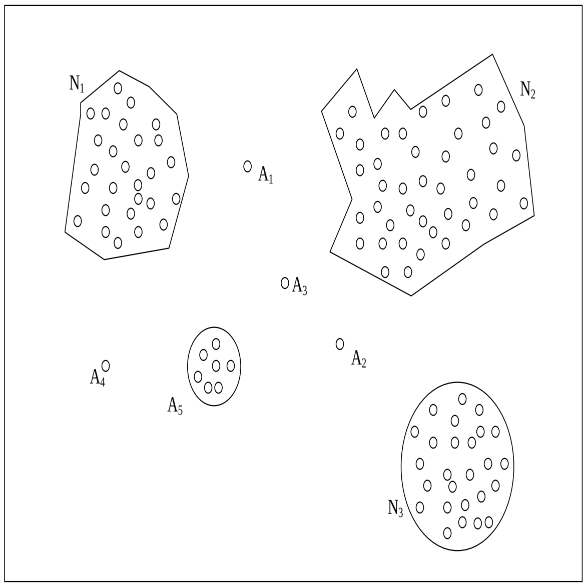


Figure 2.1: Individual anomalies in data space

people buy gifts or go on a long trips [Song et al., 2007]. Contextual anomalies are important considerations in time series [Basu and Meckesheimer, 2007] and spatial data [Kou et al., 2006].

Collective Anomalies

In some applications, the source of anomalies is a set of instances rather than a single one. Such a set is referred to as a collective anomaly. Figure 2.2 depicts an example of a collective anomaly in a time series. The area highlighted in red circle shows values that deviate from the normal pattern, although any single value in this region does not by itself abnormal behavior.

Normal behaviors may exhibit different patterns in different periods of time, because of variation in applications, users and functionality. Statistical methods may not work well in such a system. This is because a general normal model, based on a single behavior, is generated, with each single

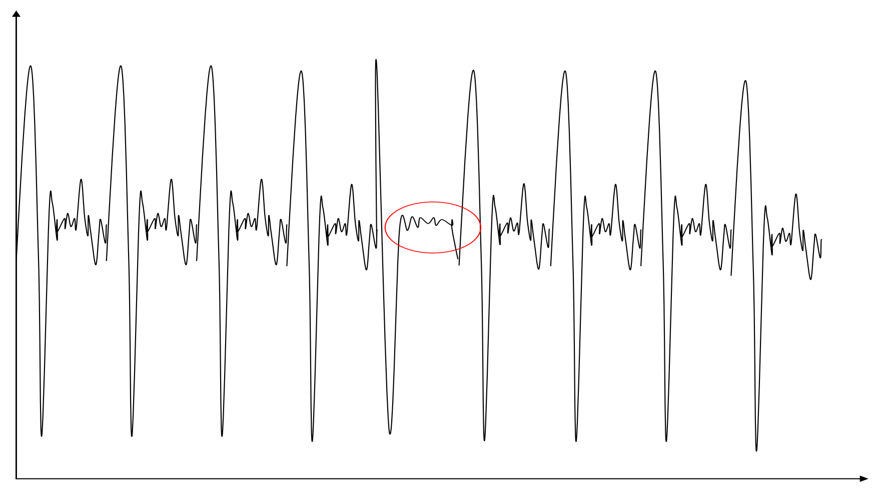


Figure 2.2: Collective anomalies in a time series

deviation considered to be anomalous. A single model is not representative of all potential normal behaviors. Research in the area of network intrusion detection shows that there is a set of normal behaviors for which using only a single border to distinguish normal from abnormal instances is not possible [Smith et al., 2002] [Mohammadi et al., 2014] [He et al., 2003]. Furthermore, a lack of labeled data makes the problem more complicated. A set of abnormal behaviors that is already known is limited to just the primary knowledge of the normal patterns, and cannot be used in conjunction with a supervised method for training a classifier. Most research efforts in this area focus on semi-supervised and unsupervised methods.

In this chapter, a literature review of anomaly detection methods is presented. In the first Section, well-known anomaly detection methods are presented including promising cluster-based methods. Since the cluster-based anomaly detection methods are the focus of this thesis, clustering approaches with their pros and cons are reviewed in Section 2.2. Cluster summarization and finding a proper representation of clusters, is the other focus of this thesis; the literature review of cluster summarization approaches is presented in Section 2.3. The proposed framework of this thesis is an incremental approach for anomaly detection and therefore, the incremental clustering approaches are reviewed in Section 2.4.

2.1 Anomaly Detection Approaches

In this section primary works in the area of anomaly detection are categorized, and for each category prominent works in the literature are reviewed. There are six main categories of anomaly detection algorithms: rule-based, statistical, proximity-based, Artificial Immune System(AIS), supervised and unsupervised methods. Each method is discussed including an analysis of their strengths and weaknesses.

2.1.1 Rule-based Methods

Rule-based or knowledge-based anomaly detection methods are among the first-developed and basic anomaly detection methods. In these methods, first, a set of rules for detecting anomalous behavior is extracted. If newly-captured behavior fits one of the rules, it is considered to be an anomaly [Ndousse and Okuda, 1996] [Sajja and Akerkar, 2010]. In the first step of rule-based methods, an algorithm such as decision tree, RIPPER and etc. is used to find the rules. Based on the quantity of training data, and the number of instances that are correctly classified by each rule, a confidence level is assigned to each extracted rule. For each new instances, the rules database is searched, and the best match is found. The inverse of the confidence level of the best-match rule is used as the anomaly score for new instance. [Fan et al., 2001] [Salvador and Chan, 2003].

A lack of labeled data in anomaly detection applications leads to unsupervised approaches for rule extraction. Some rule-extraction methods are user-dependent and rules are generated by an expert. The other approach is association rule mining which finds the rules in a dataset using a list of transactions from the current databases [Mahoney and Chan, 2003] [Qin and Hwang, 2004]. Anomaly detection methods based on association rule mining are applied in credit card fraud detection [Brause et al., 1999] and network security [Tandon and Chan, 2007].

Advantages and Disadvantages of Rule-based Methods

Rule-based methods include two steps. The first step is a training phase to define rules. The second step is a testing or detection phase to label new incoming instances. The time required for training varies according to the algorithm. The best training algorithm is association rule mining, but all algorithms for rule mining are time-consuming. The complexity of these algorithms is increased with the number of transactions, transactions width and the size of the item set [Shinichi and Jun, 2000] [Wijsen and .Meersman, 1998].

Since known anomalous behaviors are defined as a set of rules, this method is reliable in detecting the recurrence of previously known anomalies. This is why most of the current Intrusion Detection Systems (IDSs) use this approach. However, a major drawback is that these methods are not capable of detecting new and unseen anomalies. Moreover, the rule-based methods mainly rely on an expert's opinion, which may not be accurate, due to limited knowledge about any new anomalous behaviors. In the detection of new anomalous instances, a database of all normal rules has to be searched to find a matching case, which can take considerable time. The detection time is increased based on the size of database, which is inappropriate for online applications. This problem is called 'utility problem' [Minton et al., 1987] which indicates that the performance is decreased due to the time wasted trying to apply learned rules in situations that rules are incapable of solving them [Mooney, 1989].

Instead of just relying on expert knowledge and creating limited rules, another approach is to use some mathematical representation of normal behavior, such as statistical models based on various features of the dataset.

2.1.2 Statistical Methods

The assumption of statistical methods is that normal behaviors are generated based on a stochastic model. With statistical models, anomalies are in the low-probability regions. In statistical methods, some dataset features allow distinguishing normal and abnormal behaviors. For example, for network traffic, packet length and flow size are some features of the dataset. All statistic parameters, such as mean and variance are calculated for all extracted features, and a model is fitted to the data. For anomaly detection, a statistical model is fitted to normal data, and a statistical inference test detects any abnormal behavior. Thatte et al. use packet-size statistics and traffic rates to build a statistical model, and then employ the sequential probability ratio test (SPRT) in the detection phase [Thatte et al., 2011].

Another group of methods in this category works by considering normal behavior as a time series, combining it with signal processing techniques for anomaly detection purposes [Leng et al., 2009].

In another work by Hajji [Hajji, 2005], normal data are considered to be generated by a set of normal distributions, and all parameters of the model are generated based on the existing data. Any deviation from these models is considered to be an anomaly.

To find a statistical model for data, two sets of techniques are applied to the dataset.

- *Parametric techniques* assume an underlying distribution for data and then estimate the parameters of the distribution.
- *Non-parametric techniques* don't consider any underlying distribution for data.

Parametric Techniques

Parametric methods assume that data are generated based on a parametric distribution, with a probability density function $f(x, \theta)$, where x represents the data and θ is the parameter of the model. Parametric techniques are divided into two categories, Gaussian-based and regression-based models.

Gaussian-based Model

The basic assumption is that the data are generated based on a Gaussian distribution. Various statistical tests are applied to detect any deviation.

Box Plot Test: Using a Gaussian distribution for data, a simple statistical test for anomaly detection is the box plot test [Horn et al., 2001]. As shown in Figure 2.3, minimum, maximum, median, upper quartile Q_3 and lower quartile Q_1 are specified for a dataset. Inter-Quartile Range (IQR) is the region in $Q_3 - Q_1$. Each instance outside of upper quartile Q_3 and lower quartile Q_1 is considered as anomaly. All instances in the region of $Q - 1.5 \times IQR$ and $Q + 1.5 \times IQR$ are in the confidence interval of $\mu + 3 \times \sigma$ of a normal distribution with mean μ and standard derivation σ.

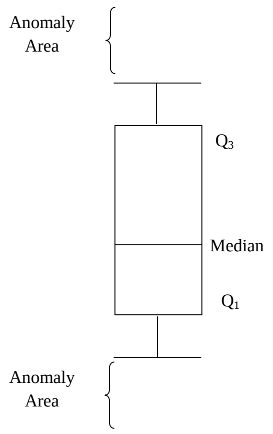


Figure 2.3: Box Plot

Grubb's Test: Grubb's test assumes the dataset has a normal distribution. A z score is calculated in a univariate dataset for each instance x, as follows: [Laurikkala et al., 2000]:

$$z = \frac{|x - \overline{x}|}{s} \tag{2.1}$$

$\overline{x}$ is mean and s is the standard deviation. A new instance is deemed to be anomalous if the z score of an instance is:

$$z > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}} \tag{2.2}$$

where N is the data size and $t^2_{\alpha/(2N),N-2}$ denotes the critical value of the $t-$distribution with $(N-2)$ degrees of freedom and a significance level of $\frac{\alpha}{2N}$. Grubb's test shows the absolute deviation from the mean. The level of deviation is defined based on the critical value, which can be calculated for any desired confidence level from the t-statistic (which is typically set to 95%).

For multivariate data, the z score is computed based on the Mahalanobis distance for instance x as follows:

$$z = (x - \overline{x})' s^{-1} (x - \overline{x}) \tag{2.3}$$

$\overline{x}$ is the mean and s is the covariance of the dataset. The same test is applied to z to determine anomalous score.

χ^2 *statistic*: Another test to detect anomalous samples is the chi-square test. χ^2 is calculated for each instance x_i as follows:

$$\chi^2 = \sum_{i=1}^{n} (x_i - E_i)^2 / E_i \tag{2.4}$$

E_i is the expected value for instance x_i, and n is the total number of variables. Each instance x_i with a large χ^2 is anomalous. χ^2 depicts deviation from the expected value. The χ^2 statistic is used, for example, to detect anomalous operating systems calls [Ye and Chen, 2001].

Regression-based Model

This approach is mainly used for time series anomaly detection. In this technique a regression model is fitted to data and deviation from the model considered as anomaly. The deviation is defined based on the difference between the predicted value and the observed value. There are various methods available to fit the regression model. The most well-known approaches are; Moving Average (MA) [Chatfield, 2004], Autoregression (AR) [Chatfield, 2004], Autoregressive Moving Average (ARMA) [Pincombe, 2005] and Autoregressive Integrated Moving Average (ARIMA) [Moayedi and Masnadi-Shirazi, 2008].

Autoregression (AR): This method calculates the value $y(t)$ at time t, based on previous outputs $y(t-r), 1 \leq r \leq m$:

$$y(t) = \sum_{i=1}^{m} a(i)y(t-i) + \epsilon(t) \tag{2.5}$$

$a(i)$ is the autoregressive coefficient and $\epsilon(t)$ is the noise at time t.

Moving Average (MA): This method predicts the value for output $y(t)$, based on the input values $x(t-r), 1 \leq r \leq m$:

$$y(t) = \sum_{i=1}^{m} b(i)x(t-i) + \epsilon(t) \tag{2.6}$$

$b(i)$ is the autoregressive coefficient and $\epsilon(t)$ is the noise at time t.

Autoregressive Integrated Moving Average (ARIMA): This method combines the AR and MA.

Non-parametric Techniques

Non-parametric techniques do not have any predefined assumptions on the dataset. These methods work with the original data with no predefined distribution model.

Histogram Method:

Histogram-based methods create and maintain a dataset histogram. New instances is compared to all the bins of the histogram, and if it does not match one of the bins, it is considered as anomalous. This method is used to detect network intrusion [Yamanishi et al., 2004] and fraud [Fawcett and Provost, 1999]. Variability in defining the size of the bins can lead to a general or specific model.

Kernel Function-based Method:

This method is based on Parzen window estimation. Probability Distribution Functions (PDF) for normal instances are estimated using kernel functions. Any new instance in the low probability area is deemed anomalous [Chow and Yeung, 2002].

Advantages and Disadvantages of Statistical Methods

The computational complexity of statistical methods strongly depends on the algorithm used in the model creation step. Most of the algorithms in this category are highly complex. For example, complexity of Expectation Maximization (EM) algorithm to find Gaussian mixtures is linear (in each iteration), but it needs a considerable number of iterations to converge. Moreover, these methods need to have entire dataset available and then set the model parameters based on the entire normal dataset. This shortcoming makes the method slow, and undesirable in online applications.

The advantage of these methods is that if the distribution assumption for statistical methods holds true, there is statistical justification and a confidence interval in anomaly detection. Moreover, in statistical methods there is no need to have labeled data.

There are some problems associated with these methods. First assuming a specific distribution for data may not be valid for many real datasets. Second, these models are usually fixed, while the nature of recent data is dynamic, and an adaptive model may be needed to represent their behavior over time.

2.1.3 Proximity-based Methods

Proximity-based methods assume that normal instances are in a dense area of data, while anomalous instances are in sparse regions, and far from dense regions. The key factor in this method is a reliable definition of the distance metric. In the case of continues numeric values for features in the dataset this distance will be the Euclidean distance. However, for categorical values, the distance is calculated in new forms, since there are no numeric values available to determine Euclidean distance [Boriah et al., 2008]. In most applications, there is a combination of both continuous and categorical values. There are approaches that can combine the similarity of continuous attributes with similarity of categorical ones [Otey et al., 2006] [Tan et al., 2005]. Proximity-based methods can be classified into two groups: distance-based and density-based methods.

Distance-based Methods

Distance-based methods assume that abnormal instances are remote from normal instances, based on a distance measure. Distance-based methods are of two main types: K-Nearest-Neighbors-based (KNN) and grid-based methods.

K-Nearest-Neighbors (KNN)

This is the most well-known distance-based method [Sricharan and III, 2011] [Zhou et al., 2010] [Orair et al., 2010]. KNN-based methods find the distance of each new instance to its K-nearest neighbors. If the distance of a new instance to its K-nearest neighbors exceeds a threshold, it is considered anomalous.

KNN-based approaches can themselves be divided into two groups. In the first, distances from each instance to all other instances are calculated, and a fixed radius r is considered. For each instance O_i, if the number of instances with distance less than r to O_i, is less than a threshold π, O_i is considered as anomalous; otherwise, it is normal [Zhang and Wang, 2006]. Based on threshold π and a radius r, instance O_i is anomaly if:

$$\frac{\|\{O_j|dist(O_i,O_j) < r\}\|}{\|D\|} \leq \pi \tag{2.7}$$

The numerator represents the size of the set of neighbors of O_i with distance less than r, and $\|D\|$ is the size of the dataset. If the number of neighbors is greater than a threshold π, the instance is normal; otherwise, it is abnormal.

For the second group of KNN-based approaches, distances from instance O_i to all other instances are calculated, and the K-nearest neighbors of O_i are determined. The average distance to all K-nearest neighbors of O_i is computed, and if the average distance is less than a threshold, O_i is considered as anomalous [Knorr et al., 2000]. Taking the average is not the only approach available; one alternative uses the sum of the distances from a new instance to its K-nearest neighbors [Zhang and Wang, 2006].

KNN is categorized as a 'lazy' classifier, since there is no training phase, and therefore no need to have any knowledge about the data. However, due to the large number of computations required to find the distances from a new instance to all other instances, KNN is time-consuming. Higher-level structures like hyper-grid have been tried to address this problem, and make it faster, but, it is still not fast enough for online applications [Xie et al., 2013].

Instead, the KNN method can be applied to subset of the data. This method reduces the computation time, with compromising accuracy [Wu and Jermaine, 2006]. Also, partitioning techniques can be applied to the dataset to prune non-anomalous areas. In this approach, the upper and lower bounds of the distance of each instance to its K-nearest neighbors in each partition are found, which are used to prune the non-anomalous area [Ramaswamy et al., 2000].

Grid-based Method

In the grid-based approach, the data space is divided up into grids with diagonals of length $\frac{r}{2}$. Two levels are defined for neighboring cells:

- For cell C, the C_{level_1} cell is the cell that, for any $O_i \in C$ and for any $O_j \in C_{level_1}$, $dist(O_i, O_i) < r$.
- For cell C, the C_{level_2} cell is the cell that for any $O_i \in C$ and for any $O_j \in C_{level_2}$, $dist(O_i, O_i) > r$.

Figure 2.4 depicts a data space with cells in two levels. All cells in the immediate neighborhood of cell C are considered to be in $level_1$ and the rest in $level_2$. The cell C is specified in red in Figure 2.4, with $level_1$ (L_1) the darker shade and $level_2$ (L_2) the lighter shade.

Based on the two levels defined for the neighbors of a cell C, anomalous instances are detected. Consider a to be the number of instances in cell C, b_1 the number of instances in $level_1$and b_2 the number of instances in $level_2$. n is the total number of instances. These two rules are used to determine the number of neighbors for each cell C and accordingly specify the normal and anomalous instances in the dataset. If the number of neighbors with distance less than r is more than a threshold π, then all instances in cell C are normal.

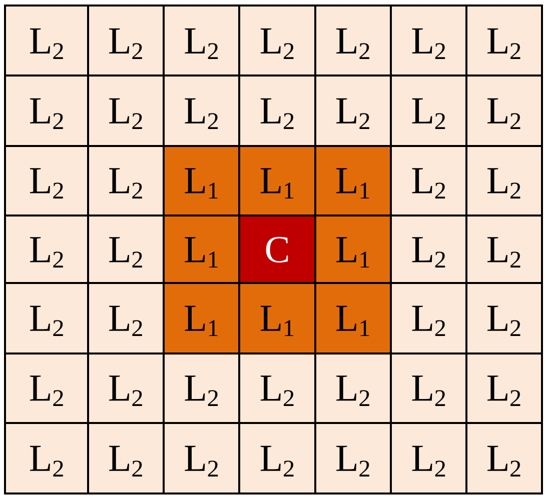


Figure 2.4: Grid of data space with two levels

If the number of neighbors with distance r is less than a threshold, then all instances in cell C are anomalous.

- All instances in cell C are considered as anomaly if $a+b_1+b_2 < \pi n+1$.
- All instances in cell C are not anomaly if $a + b_1 > \pi n$.

The first rule indicates that for every instance O_i in cell C, there are at least πn instances with distance less than r in the neighborhood, and therefore, instances in cell C are not outliers. The second rule specifies all instances in cell C as outliers, since the number of neighbors with distance less than r is less than the threshold πn.

The advantage of using a grid structure is that the whole cell can be labeled as normal or anomaly without checking each instance individually. Moreover, a limited number of cells can be checked instead of the entire dataset.

Density-based Methods

The second group of proximity-based anomaly detection methods is density-based. The basic assumption for these methods is that if the density around an instance is less than a threshold, then that instance is anomalous. While this method works well for some datasets, assumption about global density may not be appropriate for all applications. Consider clusters with different densities, as depicted in Figure 2.5. Based on distance-based methods, a_1 and a_2 would not be anomalies. Using a density-based method, if the same threshold is considered for the density of instances, not only a_1 and a_2 but also the entire cluster C_1 would be considered as anomaly. To solve this problem the concept of local density is introduced.

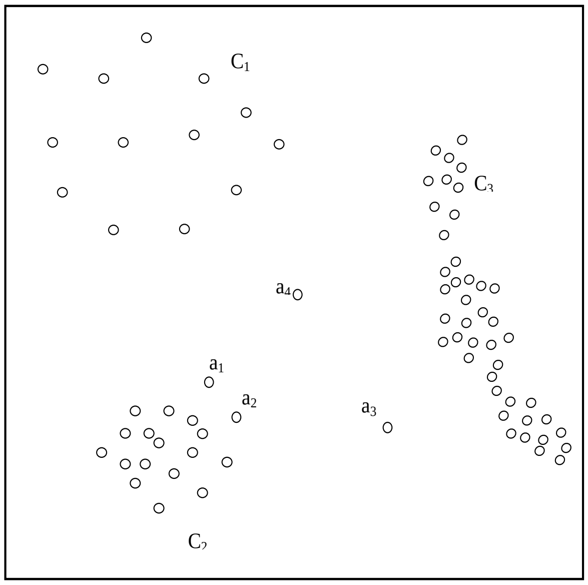


Figure 2.5: Cluster with different densities

Breunig introduced a Local Outlier Factor (LOF), which combines the density of an instance and its neighbors to find the local density factor [Breunig et al., 2000]. For each instance in the dataset, an outlier factor is computed, being a combination of the density of instance O_i and its relative density. In turn, the relative density of an instance is computed using the density of the neighbors of O_i. To find the relative density, LOF defines $k-$distance for O_i. $k-$distance of O_i, $dist_k(O_i)$ is the distance between O_i and its k-nearest neighbor. More, precisely in data space D, $dist_k(O_i)$ is the distance of O_i to instance $p \in D$ if:

- There are at least k instances $O_j \in D$ with $dist(O_i, O_j) \leq dist(O_i, p)$
- There are at most $k-1$ instances $O_j \in D$ with $dist(O_i, O_j) < dist(O_i, p)$

Based on $dist_k(O_i)$, the k-distance neighborhood of O_i is defined using:

$$N_k(O_i) = \{O_j | O_j \in D, dist(O_i, O_j) \leq dist_k(O_i)\} \tag{2.8}$$

The average distance to all members of $N_k(O_i)$ can be a local density estimation. However, since fluctuations can be high in all distances, the definition of reachability distance is introduced. For two instances O_i and O_j the reachability distance is defined as:

$$reachdist_k(O_i \leftarrow O_j) = \max\{dist_k(O_i), dist(O_i, O_j)\} \tag{2.9}$$

Based on this, the definition of local reachability density is defined:

$$lrd_k(O_i) = \frac{\|N_k(O_i)\|}{\sum_{O_j \in N_k(O_i)} reachdist_k(O_j \leftarrow O_i)} \tag{2.10}$$

The inverse of the distance is an indicator of density in a local reachability density. In this definition for density, there is no parameter for the number of neighbors, and density is defined based on the distances of instance O_i and its neighbors.

The (LOF) is defined as follows:

$$LOF_k(O_i) = \tag{2.11}$$

$$\frac{\sum_{O_j \in N_k(O_i)} \frac{lrd(O_j)}{lrd(O_i)}}{\|N_k(O_i)\|} = \sum_{O_j \in N_k(O_i)} lrd(O_j). \sum_{O_j \in N_k(O_i)} reachdist_k(O_j \leftarrow O_i)$$

In this formula, the local reachability densities of O_i and its neighbors are considered. Therefore, the density is compared to that of its neighbors, and the local density is considered rather than the global density. The average of the ratio of the local reachability density of O_i and its neighbors specifies the local outlier factor. With a low local reachability density for O_i and a high local reachability density for its k-nearest neighbors, the LOF value would be high. A high value for the LOF indicates that the local density of instance O_i is less than its neighbors.

To determine the outlier instances, density is calculated for each instance (which makes the algorithm slow). Density-based methods are categorized as static methods. To use them in online applications, a snapshot of the data is taken in different times, and anomalies are then classified. There has been limited research [Kim and Han, 2009] in this area toward making density-based methods dynamic and thus suitable for use in online applications.

Advantages and Disadvantages of Proximity-based Methods

All proximity-based methods have high computational complexity: their detection time is on the order of $O(n^2)$. These methods find the distance from each instance to all other instances in the dataset. The time complexity of this step is $O(n^2)$. Using methods like that of the $K-d$ tree is not sufficiently efficient since the computational complexity of these methods increases with the number of features. In some cases sampling techniques are applied, but they decrease accuracy. The relatively long detection time of proximity-based approaches are the main reason that they are considered unsuitable for online applications.

All proximity-based methods are considered to be unsupervised methods. Since there are no labeled data in the anomaly detection area, such a method is the best choice. However, since these methods compare each instance to all other instances (while the entire data is not available) and instances arrive over time, there would be no possibility of finding the outliers in an online context. Furthermore, there is neither enough time or space to collect the whole dataset and label new instances. The high computational complexity noted above makes this even worse. Another problem is related to the distance measures. In some application defining a proper distance measure can be difficult.

2.1.4 Artificial Immune Systems for Anomaly Detection

Approaches based on the concept of an Artificial Immune System (AIS) are inspired by immune systems and are used to solve problems in various domains, including anomaly detection applications. The two main algorithms used for anomaly detection involve negative and positive selection.

Negative Selection Algorithm

The use of a negative selection algorithm for anomaly detection was first introduced by Forrest et al. [Forrest et al., 1994]. The basic idea is to generate a number of detectors in the complementary space of normal instances. These detectors are employed to detect and classify new instances as normal or anomaly. Given a data space U and a set of normal instances N, the anomaly space A is defined as:

$$U = N \cup A \quad and \quad N \cap A = \varnothing \tag{2.12}$$

The number of detectors chosen, and how they are generated, are key attributes for negative selection. A simple detector for a negative selection algorithm uses a circle-based approach. Each generated detector is at the center of a circle of a given radius. All circles act as detectors, and each new instance is compared to these detectors. If a new instance is in the area of a detector, it is considered as anomaly [Hofmeyr and Forrest, 2000] [Balthrop et al., 2002] [Singh, 2002].

The initial negative selection algorithm employs the same radius around each detector, which is highly restrictive. To solve this problem, a new version of the algorithm called Real-Value Negative Selection (RVNS) was developed [Ji and Dasgupta, 2004]. This algorithm generates detectors with variable radii. For each normal instance in the data space, a circle with a specific radius is considered. Each generated detector is first checked against all other normal instances. If the detector is not in the area of normal circles, a radius is set for it. The radius is the distance of the new detector to its closest normal instance.

Figure 2.6 depicts the RVNS output. The dark-shaded circles are the normal instances. The other circles are detectors with variable radii. The radii are extended until they reach the boundaries of the closest circles. There are RVNS extensions that adapt the detector generation method and radii to find better representatives for anomalous instances [Aickelin et al., 2004] [Wu and Liang, 2005].

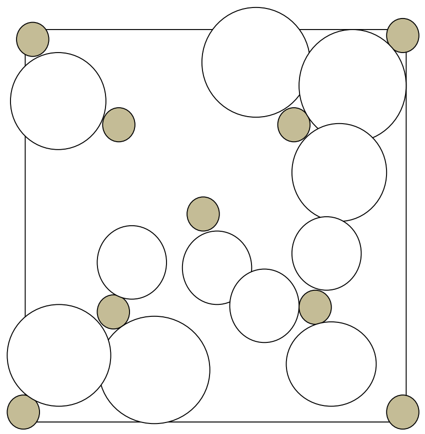


Figure 2.6: Real-Value Negative Selection with different radius for detectors

Positive Selection Algorithm

The positive selection algorithm is inspired from the biological immune system. In anomaly detection applications, this algorithm works only with normal instances [Ebner et al., 2002] [Stibor et al., 2005]. The positive selection steps are depicted in Figure 2.7. Some normal instances are selected as representative of normal behavior. Each selected instance, and the area in its neighborhood, is considered in the detection of normal instances. A circle of radius r around each normal instance specifies an area that is considered as normal. The area of all these circles is considered as the normal area. Each new instance is compared to all normal instances; if the new instance is not in the area of one of the normal instances, it is considered as anomaly, otherwise it is labeled as normal.

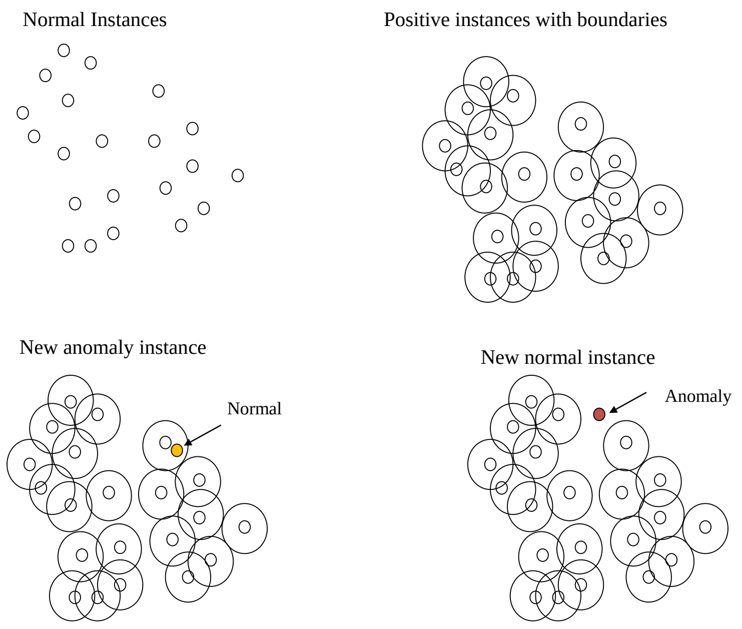


Figure 2.7: Positive selection with uniform detector radius

2.1.5 Supervised Approaches

Supervised approaches train a model using labeled data, and then employ the trained model to classify new instances. The assumption of these approaches is that the model is able to distinguish normal and abnormal instances in the given feature space. Supervised approaches can be divided into two groups: one-class and multi-class classification methods.

One-class classifiers are trained based on normal instances. In the training phase, the model is trained to separate the area of normal class from the rest of the data space. Figure 2.8 depicts a decision boundary for a normal class.

Multi-class classifiers are trained based on multiple normal classes and learn to classify each new instance against the other classes. The new instance is classified as anomaly if it does not belong any class [Stefano and Vento, 2000].

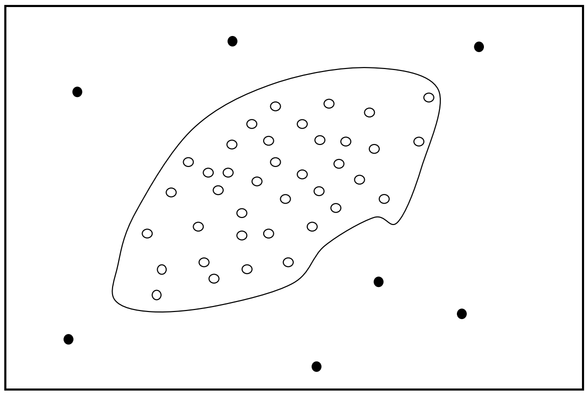


Figure 2.8: Normal class boundary

Neural Network Classifier

Neural network classifiers are used for anomaly detection in both multi-class and one-class modes. The multi-class types of neural networks are trained with multiple classes, with each new instance accepted or rejected based on a trained model [Odin and Addison, 2000].

Replicator Neural Networks (RNN) are trained with one-class labeled data [Hawkins et al., 2002]. The same input and output are considered for neural networks. Replicator neural networks label new instances x_i based on factor δ, defined as follows:

$$\delta = \frac{1}{n}\sum_{j=1}^{n}(x_{ij} - o_{ij})^2 \tag{2.13}$$

n is number of features, and o_{ij} the reconstructed output of RNN. If the distance between the reconstructed output and the original input exceeds a threshold, it is labeled as anomaly.

Supervised approaches like neural networks have been used for anomaly detection but a lack of labeled data renders some models insufficiently accurate [Han and Cho, 2006]. Shon and Moon employed a Self-Organized Feature Map (SOFM) to decrease the false alarm rate of neural networks [Shona and Moon, 2007].

Support Vector Machines

Since, the data on which anomaly detection is based are usually imbalanced and unlabeled, one-class classifiers are typically used in this area. The known behaviors that are derived from normal instances form the major part of each dataset. The number of known anomalous instances is typically small, and can make the data imbalanced. This led to the introduction of a new version of SVM called one-class SVM. This is a classifier trained only using the data of one class, separating the data of this class from other, unseen, classes [Scholkopf et al., 2001]. This approach is accurate because it characterizes boundaries over all normal instances. However, finding these boundaries has its own cost. In creating these boundaries, there is a need to preserve many support vectors which is impossible in many applications [Keerthi et al., 2006]. In particular, the number of support vectors increases with the dimensions of the data. Various methods can be used to decrease the time and memory complexity of a one-class SVM, but this is not efficient enough since it decreases the accuracy [Amer et al., 2013]. Another problem associated with SVM is a high false-alarm rate. To alleviate this problem, some approaches combine KNN with one-class SVM, but this undesirably increases the complexity [Chen et al., 2013].

2.1.6 Unsupervised Approaches

An unsupervised approach works with unlabeled data, which is a typical feature of datasets for which anomaly detection is applied. There are two sets of approaches in this category:

- Autoencoders
- Cluster-based approaches

In the following sections, the most well-known and recent approaches in each category are reviewed.

Autoencoders

The autoencoder (also known as auto-associative neural network) is a type of neural network used for novelty detection. The network is trained only with normal instances, and novelties are detected based on the trained model. By using the same input and output, the neural network is trained to be consistent in dealing with normal data. Since the network is trained with normal instances, the weights are based on normal patterns, and any new and abnormal pattern is inconsistent with the network. In other words, the input cannot be reconstructed for anomalous new instances [Japkowicz et al., 1995].

Autoencoders were first used for dimensionality reduction [Ranzato et al., 2006] [Bengio et al., 2007] [Bengio and LeCun, 2007]. Autoencoders learn the same features as Principle Component Analysis (PCA). The autoencoder structure is represented in Figure 2.9. An autoencoder has three layers: input, hidden and output layer. For a network with input with dimension D, $D = \{1, \dots, n\}$, the number of hidden units is defined based on the application and the type of transformation needed to be applied on data. The output $\hat{X}$ is the same as the input, or is a close estimation of it.

The output of the first layer is:

$$h_j = \sigma(\sum_i W^1_{ji} x_i + b^1_j) \tag{2.14}$$

and that of the second is:

$$\hat{x}_i = \sigma(\sum_j W^2_{ij} h_j + b^2_i) \tag{2.15}$$

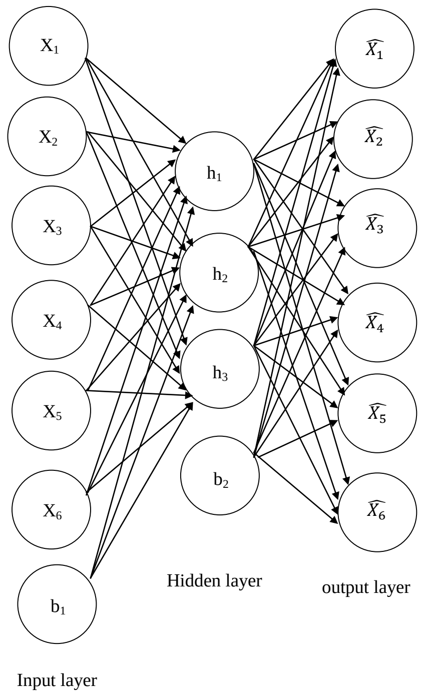


Figure 2.9: A simple autoencoder

where W_1 and W_2 are the weight matrices, b^1 and b^2 are the bias for level 1 and 2 in the structure. $\sigma(.)$ is activation function. In some approaches, W_1 and W_2 are optionally considered identical('tied weights').

The goal is to reconstruct the input while minimizing the error, $L(xz) = \|x - z\|^2$. To do so, a minimization function is used to find optimal weights and bias:

$$J(\theta) = \frac{1}{2N}\sum_{n}^{N}\sum_{i}(x_i^{(n)} - \hat{x}_i^{(n)})^2 + \frac{\lambda}{2}\sum_{l}\sum_{i}\sum_{j}(W_{ij}^{l})^2 \\ +\beta\sum_{l}\sum_{j}KL(\rho \| p_j^l) \tag{2.16}$$

where p_j^l is the mean activation for unit j in layer l, ρ is the desired mean activation, and N is the number of training instances. KL is Kullback-Leibler divergence [Kullback and Leibler, 1951]. The first term is an average sum-of-squares error term that minimizes the reconstruction error. The second term is a regularization term (also called a weight decay term) that tends to decrease the magnitude of the weights, and helps prevent overfitting. The weight decay parameter λ controls the relative importance of the two terms. ρ is a sparsity parameter and typically has a small value: the average activation of each hidden neuron j should be close to 0.05. The third term is the sparsity penalty term, toward having each unit being only partially activated. The sparsity constraint on the hidden units allows the autoencoder to discover structures of interest in data with a large number of hidden units.

The performance of autoencoders employed for dimensionality reduction can be compared to that of linear and kernel PCA. Autoencoders outperform the two PCA types, and increase the accuracy [Sakurada and Yairi, 2014].

Marais et. al. [Marais and Marwala, 2007] used autoencoders to detect internet worms. They applied autoencoders to detect the novelty on an specific dataset and it detects any novelty in data while it can be a normal behavior that has not been considered in the original data.

Wu and Guo [Wu et al., 2015] employed deep networks to design an adaptive framework for sensing and modeling a dynamic target through consideration of system resource constrains. They have used a model for novelty detection in analyzing traffic accidents and human brain function. Their approach involves creation of a framework and subsequent novelty detection.

A significant problem with autoencoders is overfitting. A Structured Denoising Autoencoder (StrDA), can use incomplete prior information to reduce the effect of overfitting. StrDA decreases overfitting while the false-alarm rate is still high [Tagawa et al., 2014].

Advantages and Disadvantages

Autoencoders are complex but accurate models, which have been recently applied to diverse anomaly detection problems. As noted autoencoders always overfit and generate a high false-alarm rate. Another issue arises from the chosen structure of the network, data representation and transformation methods. The number of hidden units and the activation function need to be chosen in a way such that the input are reconstructed in the output. Moreover, excessive training time is an issue for all types of neural networks.

Cluster-based Approaches

Cluster-based anomaly detection methods are divided into three categories. In the first category, if an instance does not belong to any cluster, it is considered anomalous. The DBSCAN algorithm introduced by Ester et al. finds clusters and anomalies simultaneously [Ester et al., 96]. In this algorithm, those instances which are not connected to any cluster are considered anomalous. DBSCAN algorithm is discussed in more detail in section 2.2.4. A disadvantage of this method is that it needs to have the entire dataset to find anomalous instances. In online applications, data arrives over time, and it is not practical to store all data. Rock [Guha et al., 2003] and SNN clustering [Ertoz et al., 2003] are two other types of clustering algorithms that do not force the entire data instances to be attached to a cluster. Those instances that do not belong to any cluster are labeled as anomaly.

In the second category for cluster-based methods, instances with distance less than a threshold to their closest cluster are considered as anomaly. It means that those instances that are far away from all clusters and they do not belong to any cluster are considered as anomaly. In the first cluster-based approach, detection of anomalous instances is done at the same time as cluster creation, while in the second approach clusters are generated first and then anomalies are detected.

Different clustering approaches, such as k-means, (EM) and Self Organizing map(SOM) [Keogh and Smyth, 1997] have been used to create clusters for anomaly detection [Smith et al., 2002]. By finding a center for each cluster, and setting up a threshold, any instance with a distance greater than that of the threshold to the center of clusters are deemed anomalous. However, representing the whole cluster using the center is inaccurate. Another problem with these methods is that the number of clusters is large, and to find the closest cluster to a new instance, all clusters must be checked. Mohammadi et al. [Mohammadi et al., 2014] presented the idea of using priority to rank the clusters in the testing phase, which makes the checking procedure faster.

In the third category for cluster-based anomaly detection, any instance that belongs to a small and sparse cluster is considered as anomaly. The well-known method in this category involves the Cluster-Based Local Outlier Factor (CBLOF) [He et al., 2003]. The CBLOF, is calculated based on the distance from each instance to the center of the cluster, and the distance of the cluster itself to other clusters. Variations of this approach have been proposed, using different distances and parameters to improve CBLOF [Pires and Santos-Pereira, 2005] [Eskin et al., 2002].

Advantages and Disadvantages of Cluster-based Approaches

Complexity of clustering depends on the chosen clustering algorithm. Most of the algorithms that find distances for all pairs of instances in a dataset have quadratic complexity.

Clustering-based approaches do not need labeled data to find anomalous instances. The result strongly depends on the clustering algorithm, and even more so on the cluster representation. As mentioned earlier, representing clusters by their centers or relying on their size and density can lead to many abnormal instances not being found.

Cluster-based approaches are interesting and applicable in anomaly detection system due to some reasons. First, there are unknown patterns that are encountered over time, and so unknown in the first training step. Therefore, using clusters that create boundaries around known patterns, separating them from unknown patterns, is the best solution. Moreover, in the absence of labeled data, unsupervised methods are the best choice. As a result, the focus of this thesis is on cluster-based methods. Since clustering is the main part of these methods, clustering algorithms will be reviewed as well.

2.2 Clustering Approaches

The algorithms available for clustering can be categorized into four groups: partition-based, hierarchical, spectral-based and density-based clustering [Han et al., 2006].

2.2.1 Partition-based Clustering Methods

K-means is a famous algorithm in the area of partition-based clustering. K-means receives the number of clusters as an input, iteratively determines the centers of clusters and assigns instances to each center based on the distance from the instance to the centers of clusters. The centers of k-means are artificial, and they are not part of input instances [MacQueen, 1967]. K-median is another version of k-means, finding centers from available instances [Jain and Dubes, 1988]. The K-medoid algorithm is similar to k-means, defining clustering as an optimization problem, and trying to find the best centers for clusters in an iterative mode that makes this clustering algorithm more complex and time-consuming [Kaufman and Rousseeuw, 1987].

Advantages and Disadvantages

Partition-based algorithms are among the oldest used in clustering. The complexity of these algorithms mainly depend on stopping criteria and the number of iterations. These algorithms are sensitive to noise, and the clusters are changed by the presence of noise. More importantly, these methods represent clusters with a center and a radius. This spherical representation loses interesting information, such as cluster shape and density distribution. Furthermore, in these algorithms there is a need to know the number of clusters before clustering can be done. In many applications, such information is not available.

2.2.2 Hierarchical Clustering Methods

Hierarchical clustering methods group data into a hierarchy, or tree, of clusters. Hierarchical clustering methods can be divisive or agglomerative, depending on whether a bottom-up or top-down strategy is chosen. The Agglomerative NeSting (AGNES) method starts by considering each instance as a cluster, then combining them into a hierarchy of clusters [Kaufman, 1990]. Divisive ANAlysis (DIANA) works in top-down mode, starting with all instances in one cluster before dividing them into clusters [Kaufman, 1990]. The chameleon algorithm creates a graph of an entire dataset using the KNN algorithm, then partitioning it into smaller sub-graphs. Sub-graphs are combined to create clusters and their related hierarchy [Karypis et al., 1999].

Advantages and Disadvantages

Considering clusters to be a hierarchy saves time in finding a proper cluster for each new instance. Also, it characterizes a logical hierarchy associated with a dataset. However, the basic problem in this algorithm is the challenge of finding a measure to use in dividing or combining clusters. For many applications, such a measure is difficult to find.

2.2.3 Spectral-based Clustering

All algorithms in this category are also called grid-based methods that divide the space of instances into cells, then algorithms use these cells to cluster data. STING [Wang et al., 1997] and CLIQUE [Agrawal et al., 1998] are two well-known spectral-clustering algorithms. Based on some parameters such as cell distribution, density and distance, cells are merged and clustered.

Advantages and Disadvantages

These algorithms are able to create arbitrary-shaped clusters, but a major drawback of these methods is the complexity involved in creating an efficient grid. The size of the grid varies with different dimensions, and setting grid size and merging the grids to find clusters is difficult. These problems make these algorithms inaccurate in many cases.

2.2.4 Density-based Clustering Methods

In the area of arbitrary-shape clustering, density-based methods are more interesting; DBSCAN [Ester et al., 96] and DENCLUE [Hinneburg and Gabriel,

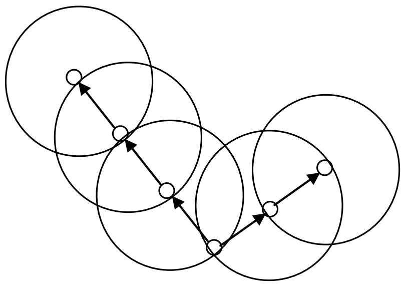


Figure 2.10: Connecting core points to create a cluster

2007] are the most famous ones. In density-based methods, clusters are created by connecting center of dense regions called core points. In DBSCAN algorithm, a point p is a core point if at least k points are within distance r of it, and those points are said to be directly reachable from p.

DBSCAN starts from a random point p_1 and if it is a core point, all of its neighbors are added to a new cluster C and marked as visited. If point p_2 in the neighborhood of core point p_1 is also a core point, all neighbors of p_2 are added to the cluster C. DBSCAN continues adding objects to C until C can no longer be expanded. The idea of connecting core point are depicted in Figure 2.10.

Advantages and Disadvantages

Density-based clustering algorithms are used to find arbitrary-shape clusters without having the number of clusters as an input parameter. However, a major concern for these algorithms is processing time. Based on the prevalence of real-time applications, there is interest in speeding up these algorithms, particularly for online applications [Guha et al., 2003] [Bifet et al., 2009] [Aggarwal et al., 2003].

2.2.5 Trajectory Clustering Methods

Most clustering approaches are used to cluster point data. However, there are some applications in which data associated with moving objects' trajectories are collected from satellites by tracking facilities, and need to be clustered.

The algorithm proposed by Gaffney represents a set of trajectories as a regression mixture model, with the EM algorithm employed to cluster data [Gaffney and Smyth, 1999] [Gaffney et al., 2006]. Another algorithm in this category is TRACLUS, which finds sub-trajectories in the trajectory data. The TRACLUS algorithm clusters data in two phases: partitioning and grouping. In the first phase, a formal trajectory is partitioned using the minimum description length (MDL). In the second phase, a density-based line-segment clustering algorithm is applied [Lee et al., 2007].

2.3 Cluster Summarization

In anomaly detection methods, the number of clusters is unknown and cluster shapes are arbitrary. Of the methods reviewed, density-based clustering method is the best choice. The main challenge with all clustering methods is how to represent a cluster with minimum space. Summarization eases the complexity of arbitrary-shape clustering methods. K-means uses a simple representation using a center and radius, to summarize a cluster. A naive way to represent an arbitrarily-shaped cluster would be to represent each cluster using all cluster members. This approach is not practical, and it does not properly represent cluster properties. Summarization does not capture how data is distributed in the cluster.

There are different ways to summarize arbitrary shape clusters [Yang et al., 2011] [Cao et al., 2006] [Chaoji et al., 2011]. These algorithms use the general idea behind the clustering methods for arbitrary shape clusters.

In summarization, the idea is to detect dense regions and to summarize the regions using core points. A set of proper features is then considered to summarize the dense regions and their connectivity. Chen and Tu [Yang et al., 2011] created a grid for each cluster; based on the idea of connecting dense regions, the core or dense cells are kept, along with their connections and related features. In all summarization approaches, these features play a crucial role. In the proposed approach by Song and Wang, the location, range of values and status connection vectors of cells are kept to find and connect dense cells [Yang et al., 2011]. The grid-based approach has some drawbacks. First, creating grids on each cluster is time consuming. Second, considering all grids means expending a lot of processing time and memory space, which is impractical in many cases.

Cao et al. [Cao et al., 2006] used core points to generate the cluster summary. The most significant drawback of this work is that the number of core points is large and in some cases is equal to the number of input instances. Moreover, a fixed radius specifies the neighborhood, that does not represent the distribution of instances in each cluster [Cao et al., 2006]. Chaoji et al. represent a density-based clustering algorithm named ABACUS for creating arbitrary-shape clusters [Chaoji et al., 2011]. The summarization part of their approach is based on finding core points and the relative variance around points. In most of arbitrary-shape clustering methods, two parameters are needed: the number of neighbors, and a radius. The most interesting and noticeable part of the work of Chaoji et al. is that the number of neighbors is the algorithm's only parameter, and that the radius is generated using the data distribution. The significant drawback of their method is that the algorithm may generate many core points.

In all these summarization approaches, the focus is on preserving cluster members; any usage of clustering for classification is not considered. The approach in this thesis is to summarize clusters using GMM. This approach covers both requirements: it is a good representation of a cluster, and it can be used for classification.

2.4 Incremental Cluster Updating

A well-known algorithm for clustering data in online applications is BIRCH, introduced by Zhang et al [Zhang et al., 1997]. The BIRCH algorithm creates a hierarchical structure of clusters in the form of a tree which makes this algorithm fast for searching. In spite of this, there are some disadvantages. Creating a hierarchy of data is usually a difficult task, which involves setting many parameter. As well, updating a tree structure is not a straightforward task. Moreover, BIRCH employs the idea of a center and a radius to define a cluster, which makes it inappropriate for non-convex clusters. In addition, center-based approaches are sensitive to noise, and have low accuracy in clustering new instances.

STREAM [O'Callaghan et al., 2002] is an incremental algorithm that develops weighted medians over time. In this algorithm, the LSEARCH is used to find cluster medians. Each new data instance is added to the set of current medians, and LSEARCH is applied to find new medians. For accurate clustering, a large number of medians need to be preserved, which is not appealing for an online environment.

Clustream [Aggarwal et al., 2003] is an incremental algorithm that combines the ideas of BIRCH and STREAM in a framework. Clusters are updated in pyramidal time steps, with existing clusters replaced by new ones. Clustream manifests the same problems as BIRCH and STREAM.

In the area of density-based clustering algorithms, DenStream [Cao et al., 2006] is used to make DBSCAN an incremental clustering algorithm. Using

DBSCAN as the main algorithm for initialization and updating stages, each new instance is assigned to a cluster, where the distance is less than a radius to a core micro-cluster. The core micro-clusters are connected through density, a reachable concept that involves using the entire dataset in the updating step. Dealing with the whole dataset makes the algorithm slow, and unsuitable for online applications.

Grid-based algorithms can also be adapted for use in incremental and online environment [Chen and Tu, 2007]. This is similar to DenStream, finding dense cells of grid and connecting them. In addition to the problems mentioned earlier for DenStream, constructing a grid on the whole data space has its own limitations.

Incremental GMM algorithms are another group of incremental clustering algorithms. These assume that all data is generated based on a GMM [Hajji, 2005]. In this group, a single GMM is found for the entire dataset. After generation of the first GMM, each new instance is fed to the current GMM, and the GMM is updated based on the new instance [Song and Wang, 2005] [Declercq and Piater, 2008] [Hennig, 2010].

The approach in this thesis is a combination of GMM-based incremental clustering and traditional incremental clustering algorithms. A new structure is introduced, which is fast and accurate for online data clustering, as explained in the next chapter.

2.5 Summary

In this chapter, anomaly detection methods are reviewed. The pros and cons of various methods are discussed, toward finding promising approaches. Rule-based methods are generally not good candidates, since they are restricted to a set of rules, and these rules are not able to detect new and previously unseen anomalies.

Statistical methods are time-intensive, and also require specific assumptions about data distribution that may not be valid for all applications. However, with accurate mathematical models, there can be statistical justification and a specified confidence interval in anomaly detection. Proximity-based methods are suitable for all types of datasets for detecting anomalies, without the need to create a model. These approaches do not require labeled data, but their need to work with the entire dataset makes them time-consuming for detection.

Artificial Immune Systems (AIS) approaches rely on generating many self and non-self detectors to represent normal and abnormal patterns, for use in labeling new instances. AIS-based approaches employ many instances for labeling new instances, making them slow for detection. Also, using a subset of instances to represent the entire normal and abnormal pattern is not an accurate approach. An advantage of AIS methods is their allowance for the possibility of having different normal patterns in a dataset.

Supervised approaches rely on labeled data to train a model for anomaly detection. In the presence of labeled data, these methods can be among the most accurate ones. On the other hand, unsupervised approaches that do not require any assumptions about a dataset, and do not require labeled data, and so are good candidate for anomaly detection applications. In unsupervised methods, clustering approaches consider each cluster as representative of a normal pattern in a dataset. Clusters are used to separate the border of the normal pattern from the rest of the data. This approach is similar to AIS-based methods, with the difference that not every instance is used to represents normal patterns. As a result, the clustering approach is fast and accurate in terms of creating a suitable boundary with clusters.

Among the clustering approaches, DBSCAN [Ester et al., 96] is a good candidate, since it generates arbitrary shape clusters that lead to more accurate boundaries for clusters. However, other clustering approaches may be more appropriate, based on the application and its requirements.

Clusters are used to represent normal patterns, but the goal is to also use them in labeling new instances. A limitation of cluster-based approaches is that they do not consider any proper representation for clusters in specifying the associated area of each cluster. In most approaches, the center of a cluster is considered as representative of the clusters, or the entire set of cluster members. Anomalies are specified based on their distance to the centers or to all cluster members. However, using a center can be inaccurate, and processing the entire set of cluster members is time-consuming. Various cluster summarization methods are available to mitigate this problem.

ABACUS as cluster summarization approach uses the idea of finding dense regions inside clusters to summarize clusters. This method is a time-consuming method, since it uses many iterations to find the best representation for clusters. Instead, a summarization based on GMMs can be introduced to decrease the computation time. Using a GMM representation combined with a clustering approach overcomes inherent issues of cluster-based and statistical approaches. A strength of GMM-based approach is the statistically-justified probabilistic model used in the detection phase, which makes later updating of the model fast and easy.

For use in an online application, any model needs to be updatable. Hierarchical clustering approaches such as BIRCH [Zhang et al., 1997] and STREAM [O'Callaghan et al., 2002] are fast in terms of labeling new instances. However, they suffer from the difficulty of the need to create a hierarchy in the first place, and also that updating the hierarchical structure is time-consuming. Processing by an incremental DBSCAN is time-consuming as well. The GMM-based approaches are fast in labeling new instances, with simple and accurate updating strategies to update GMMs. However, use of a single GMM for an entire dataset is typically inaccurate; using an incremental approach for GMMs in combination with clusters mitigates the problem.

The approach proposed in this thesis combines the strengths of different approaches to introduce an innovative anomaly detection system.

Chapter 3

Incremental Clustering Architecture

In this chapter, the general structure of the proposed model for anomaly detection is presented. Four steps are considered to design the online anomaly detection method. The fellow chart in Figure 3.1 shows these four steps. In the proposed structure, first the normal dataset is clustered. Each cluster is representative of a normal pattern. The employed clustering approach is arbitrary shape clustering method. To use clusters as classifiers, each cluster is represented by GMMs. Detection phase uses GMMs to find the similar clusters, and accordingly, label new instance as normal or abnormal. Last step is to update model with an incremental updating approach. Each step is discussed in more details in this chapter.

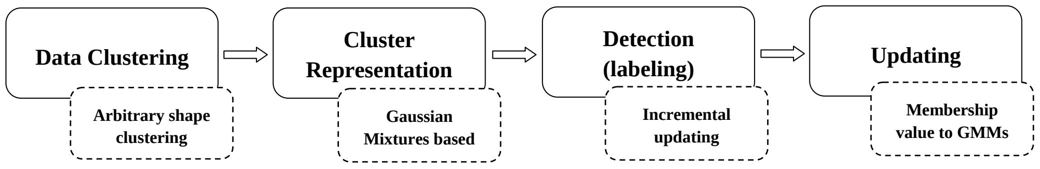


Figure 3.1: The four steps to design online anomaly detection system

The structure of the proposed method for clustering and cluster representation phases is shown in Figure 3.2. In the first step, data are clustered using a clustering method. Then each cluster is represented using a GMM to fulfill two purposes. First, each cluster is summarized in a way such that the main components of each cluster are preserved, but without keeping a direct record of all instances in each cluster. Second, the generated GMM for each cluster is considered an accurate model, that can be used to classify incoming instances either normal or anomaly. Clustering, allows the borders of normal classes to be defined without needing labels. Following clustering and GMM representation, a model is trained using normal instances.

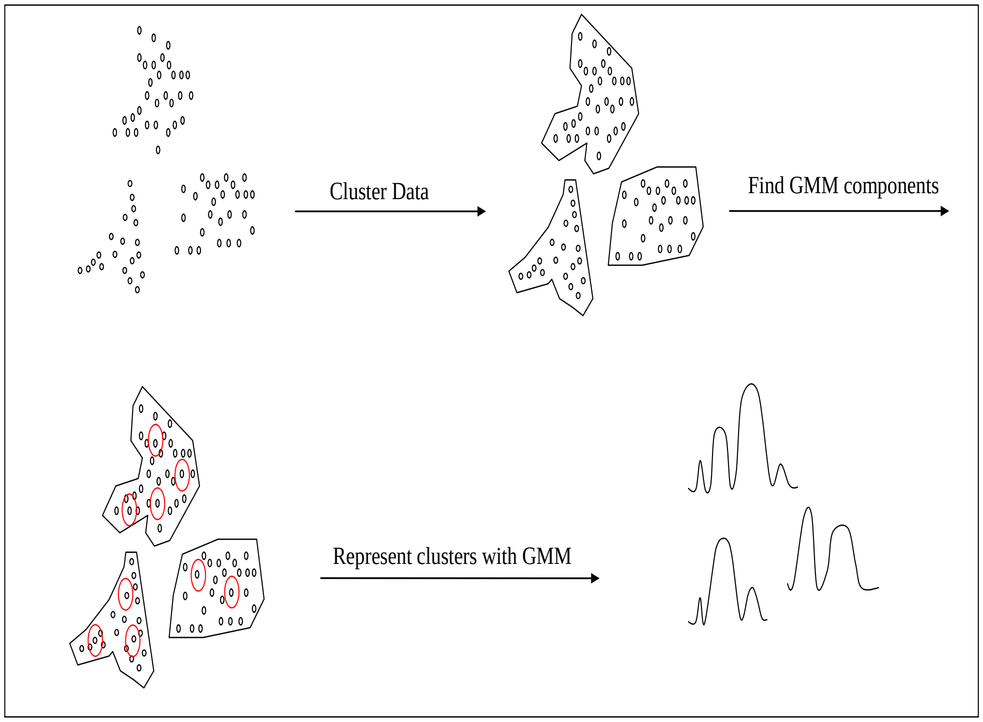


Figure 3.2: Clustering a normal dataset, and representing each cluster by a GMM

Clusters are created using an arbitrary shape clustering method. These methods are able to detect the actual shape of each cluster or at least precise estimation. One way of preserving the shape and the distribution for each cluster is to keep a record of all instances in each cluster, which requires a large volumes of memory and time. As mentioned, GMM can be used as an effective means to represent clusters. By employing GMM, the proposed algorithm is able to represent the true shape of each cluster, without keeping the entire set of instances in memory.

Each GMM consists of a number of Gaussian distributions, with the quantity determined as an input parameter. In Chapter 4, a new method is given for finding the number of components for each GMM automatically.

After representing each cluster as a GMM, the proposed structure will be able to detect anomalies. The GMM is a probabilistic model, which is used to assign a membership value and so specify whether a new instance belongs to the current clusters set. As shown in the Figure 3.3, if the membership value of an instance exceeds a threshold for a particular cluster, the instance is considered normal, and otherwise an anomaly.

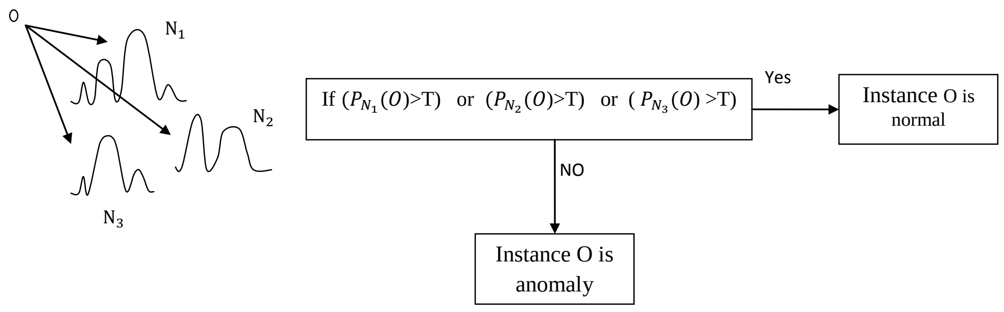


Figure 3.3: Labeling new instances as normal or anomaly

The model described thus far is the most simplistic cluster-based model. In the next section, some modifications for improvement are proposed.

3.1 Clustering Normal Behavior

Based on the requirements of an application and the nature of data, different kinds of clustering algorithms can be applied, each with advantages and disadvantages.

In the proposed model, arbitrary shape clustering methods are used to determine more accurate shapes for each cluster. Arbitrary shape clustering methods are useful here, because they generate clusters and the number of clusters simultaneously with no need to know the number of clusters. More importantly, in anomaly detection applications, more accurate boundaries of clusters need to be generated; arbitrary shape clustering methods are among the best choices for this task.

There are two categories of arbitrary shape clustering methods; spectral-based and density-based clustering [Han et al., 2006].

In spectral clustering, STING [Wang et al., 1997] and CLIQUE [Agrawal et al., 1998] are prominent ones. They generate arbitrary shape clusters by connecting dense regions using a grid structure. The presence of the grid structure increases system complexity; it is difficult to manage, especially in an online environment.

Because of this problem, density-based methods are often superior. The most frequently used method is DBSCAN [Ester et al., 96]. Its general approach is to connect dense data regions. This has its own particular limitations. To record the shape of a cluster, either a majority of instances are stored, or the boundary of each cluster is detected. Either way, the memory complexity and performance requirements for these methods are not negligible. Moreover, a technique is needed that not only preserves the shape of the cluster, but also retains valuable information about that cluster for use in further investigation.

The SGMM method that is presented in Chapter 4 overcomes this limitation, and enables arbitrary shape clustering methods to be used in online applications. SGMM represents each cluster as a GMM, consisting of a set of normal distributions that reasonably represents different kinds of datasets.

The SGMM method has similarities to some GMM-based probabilistic methods. In a method presented by Hajji, the EM algorithm is applied to the entire dataset and the entire data is represented by a single GMM [Hajji, 2005]. In our approach, instead of dealing with the entire dataset, clustering is applied first, with each cluster represented by a distinct GMM. A first advantage of this approach is decreased complexity, since there is no need to deal with the entire dataset concurrently. Moreover, a GMM for each cluster is found in parallel and independently.

Estimating a separate GMM for each cluster is easier and more accurate than estimating a single GMM for the entire set of instances. This is because different parts of the data come from different distributions, and separating them makes them independent and more accurate. In the EM algorithm, the number of GMM components should be determined as an input parameter, while the number of Gaussians is chosen automatically by the clustering algorithm. The SGMM algorithm is itself an automatic approach, which finds the number of GMM components in each cluster. Taken together, a combination of a clustering method (such as DBSCAN) and the SGMM method creates a more accurate representation of the range of normal behaviors in the dataset.

However, cluster updating is still required, and is described in the next section.

3.2 Incremental Cluster Updating

To update clusters represented by GMMs the Collective Probabilistic Labeling (CPL) approach is proposed. In this approach instead of updating clusters with each new instance, new instances are collected, and the clusters are updated based on a set of new instances. As shown in step 1 of Figure 3.4, initial data are clustered, and represented by GMMs. Thereafter, new instances are collected in a bag called rag bag and are clustered. In step 2, both current and new clusters are represented by GMMs. With an incremental approach presented in Chapter 6 close clusters are found and merged. The processes of finding distances and merging clusters requires many detailed steps, which are discussed in Chapters 5 and 6. The clusters are updated as shown in step 3. The shape of the updated clusters changes, representing a new shape and structure consistent with the most recent data.

3.3 Two-Layer Cluster-based Structure

The structure presented in Figure 3.4 is a general view of the whole updating structure we present in this thesis. A significant challenge for effective clustering in an online environment is to find a fast way to update existing clusters based on new instances. A two-layer structure is proposed, which is able to perform updates in an efficient way.

The two-layer structure is depicted in Figure 3.5. As shown in this figure, there are two layers in this structure; fine and coarse. In the coarse level, each cluster is represented by a GMM with m components and in the fine level by

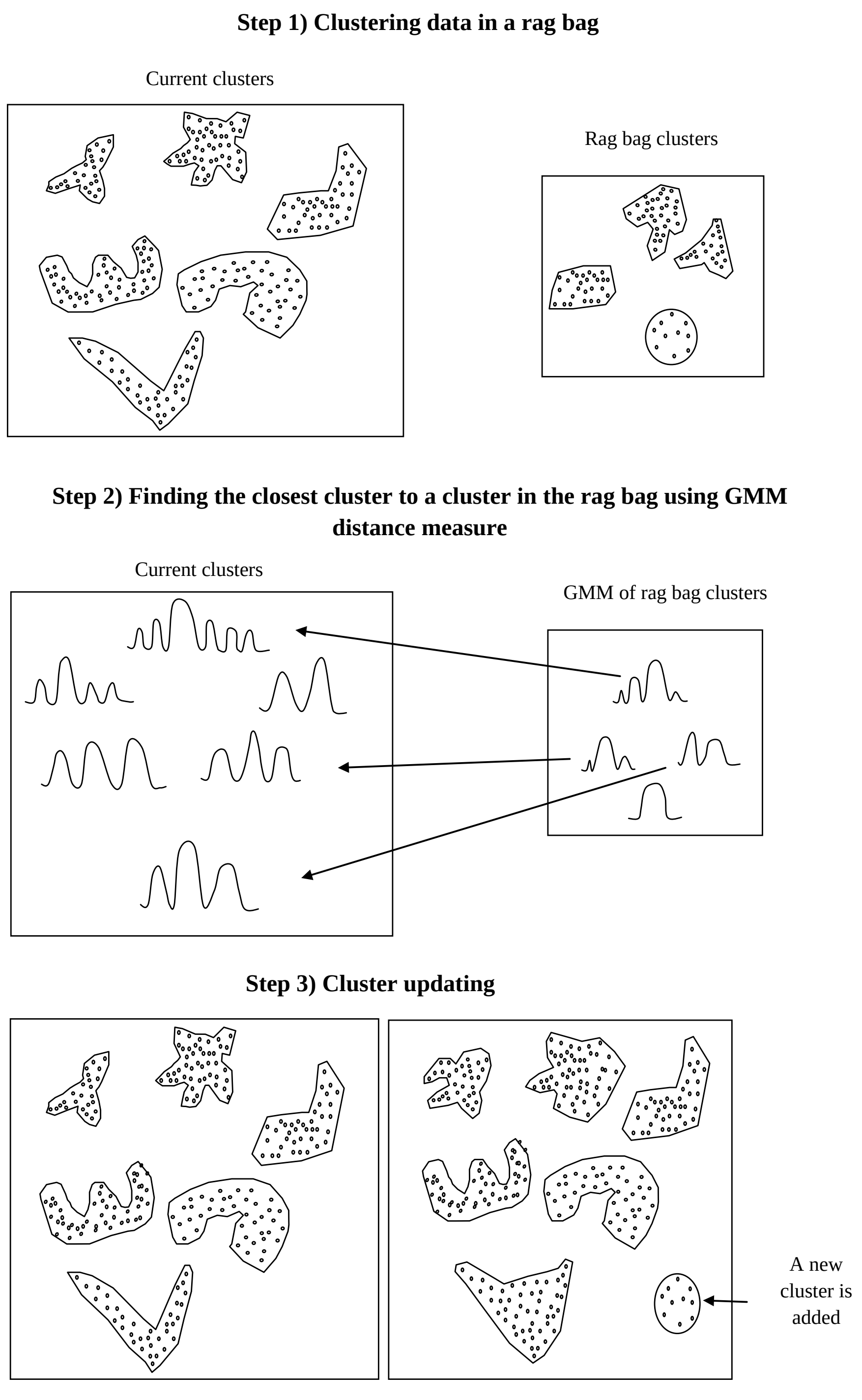

Step 1) Clustering data in a rag bag
Current clusters
Rag bag clusters
Step 2) Finding the closest cluster to a cluster in the rag bag using GMM distance measure
Current clusters
GMM of rag bag clusters
Step 3) Cluster updating
A new cluster is added

Figure 3.4: Cluster updating steps

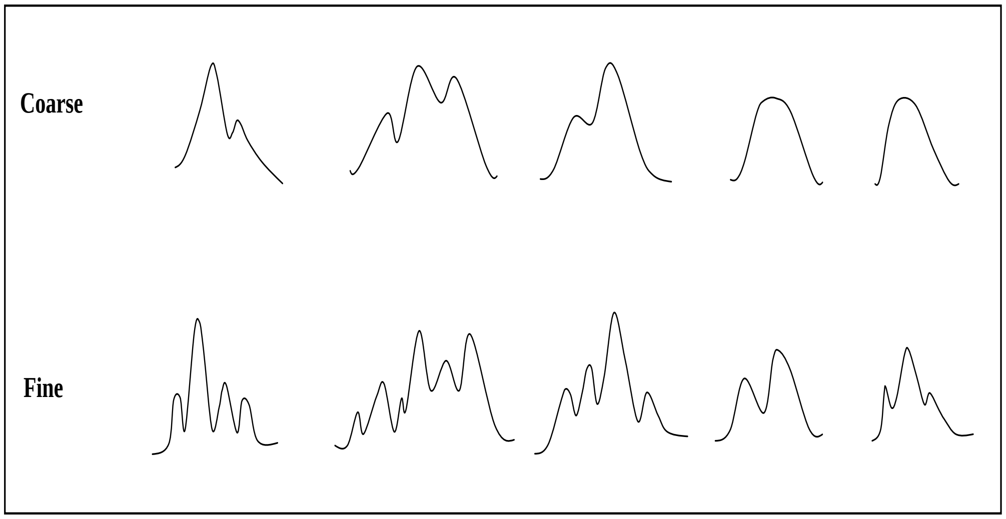


Figure 3.5: Two-layer clutter-based structure

n components, where $n > m$. The coarse level is generic and the fine level specific and thereby more accurate and computation-intensive in estimating the shape of each cluster, instance scattering and instance distribution. In the training phase, clusters are estimated in both coarse and fine levels with m and n components. This means that each cluster is presented in two different ways, one of which is more accurate than the other, but it involves more computation to classify new instances.

The role of the rag bag is to keep new instances which do not manifest any known behavior. The instances with a membership value less than a predetermined threshold of existing cluster GMMs are kept in rag bag. To distinguish these instances, the proposed method employs thresholds in both the coarse and fine levels.

The two-layer structure has two main responsibilities. First, it labels incoming instances and identifies them as normal or abnormal. Second, the structure is used to update the current clusters. This updating strategy is accurate, fast and noise resistant.

Each new instance is fed first to coarse level; if the instance can be assigned to a cluster with high probability, the instance does not need to be send to the fine level. If the membership value of a new instance to clusters is below a threshold, it is sent to the rag bag directly. If further exploration is needed, the new instance is sent to the fine level. If the instance belongs to one of the clusters and has a high membership value, the instance is labeled as normal and we ignore the instance in updating. However, if the instance does not belong to one of the clusters, it is sent to the rag bag; it may be used to update the clusters offline to save time. Most instances tend to be redundant instance (duplicates of known ones) and so are assigned to a cluster because of a high membership value.

In summary, the two-layer structure has five components:

- To cluster the first chunk of normal instances and each new batch of data:
  - As discussed in the Chapter 4, we used density-based clustering approaches to cluster data. However, any other clustering method can be used in this structure and it is not dependent on clustering approach.
- Representing clusters with GMMs:
  - The GMM representation of data eases the complexity of storing clusters in memory, presents a good summary of clusters, and allows clusters to be updated efficiently and quickly.
  - A new SGMM algorithm is presented in Chapter 4, representing each cluster as a GMM.

- Collecting data in a rag bag and labeling them collectively:
  - In two-layer structure instances are collected, and then used to update clusters; this needs a new approach to label a collection of instances.
  - In Chapter 5, a new CPL algorithm is presented to add new data to a cluster and label them based on their proximity to one of an existing cluster.
  - A new cluster distance measure is introduced in chapter 5.
- Updating clusters incrementally:
  - The incremental clustering approach based on GMM representation of clusters is presented Chapter 6.
- To create coarse and fine-level clusters:
  - The coarse and fine levels are created to speed up the updating procedure as discussed in Chapter 7.

The whole structure puts these components together and it creates an incremental approach for cluster updating for anomaly detection application. In the following Chapters, all steps mentioned above are discussed with more details.

Chapter 4

Cluster Representation using GMM

As mentioned in the previous chapter, clusters are the main components of the proposed anomaly detection method. In this method, clusters are employed to label new, incoming instances. Each new instance is compared to existing clusters, and the closest one is chosen as a cluster that the new instance belongs to.

There are various options for assigning a new instance to a cluster. One way is to create a boundary for each cluster. If the new instance is inside the boundary of a cluster, then the new instance belongs to that cluster. Finding the boundary of arbitrary shape clusters is a complex and time-consuming process, especially in high-dimensional problems. Moreover, it is necessary to consider many faces to keep the borders of clusters created by a convex in higher dimensions, and the number of faces grows exponentially with dimension [Kersting et al., 2010] [Hershberger et al., 2009].

Figure 4.1a shows that for even a simple cluster in a two-dimensional space, there is a need to find many vectors to distinguish the inside and outside of the cluster. Another approach to label a new instance is to use simple cluster representation. Clustering algorithms such as k-means use centers and radius to represent clusters. Using this representation, the distance

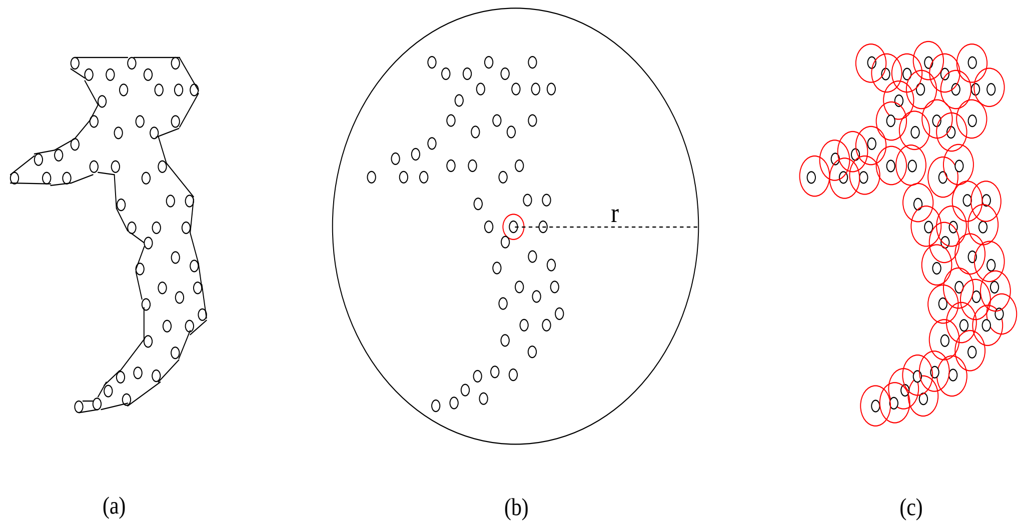


Figure 4.1: Approaches for cluster representation. a) The internal area of the cluster is separated from the outside using boundary vectors. b) K-means cluster, with center specified by the red circle, and having radius r. C) A cluster represented by sets of circles with equal radii.

of a new instance to the center of a cluster is calculated. If this distance is less than the radius of the cluster, the new instance is attached to that cluster [Mohammadi et al., 2014] [Gaddam et al., 2007].

K-means and partition-based clustering methods are sensitive to noise. Moreover, these methods require knowledge of the number of clusters, which is usually not available in anomaly detection applications. Figure 4.1b shows a cluster found by the k-means algorithm, with the entire area in the circle considered within the cluster area. Any instance that lies in this area is deemed as normal. Since the boundary is inaccurate, the detection of anomalies would be open to error.

To overcome these problems, arbitrary shape clustering methods are used. In particular, DBSCAN was chosen, since it is less complex and faster than other methods of this type [Ester et al., 96]. Arbitrary shape clustering methods are theoretically ideal but the representation and analysis of each cluster can still cause many problems. Representing each cluster in such a way that all cluster characteristics are correctly recorded is a challenging task. Preserving all instances of all clusters is not feasible for a large quantity of data, or in online applications with a large arrival rate for new instances.

A simple way to preserve the shape and specify the border of a cluster is to define a circle enclosing each instance in a cluster. First, instances are selected as the centers, and a radius r is chosen. Contained instances near the center (distance less than the given threshold, r) are removed from the dataset, and the centers are used to represent the set of removed instances. This approach keeps centers and radii in place of instances. To label a new instance, it is compared with all centers, and if the instance is inside the boundary of one of the centers, the new instance is assigned to that cluster. Otherwise, it is outside the area of all the clusters and consequently it is anomaly [Cao et al., 2006]. This structure is represented in Figure 4.1c, in which to preserve accuracy, the radius of each circle is kept small.

There are drawbacks using circle-based summarization. To keep a firm and rigid boundary, the radius should be minimized, which keeps many instances as depicted in Figure 4.1c. The boundary is well-preserved with all instances in the cluster but the clusters lose their generality. Since new instances are classified by comparison with all circles inside each cluster, the decision is binary, and there is no probabilistic value available to categorize a new instance as an anomaly or normal.

Considering all problems, we present our approach to represent each cluster. The crucial point to keep in mind for cluster representation is the need to preserve cluster features, while keeping the generated summary as small as possible. Our proposed approach is to represent each cluster as a GMM.

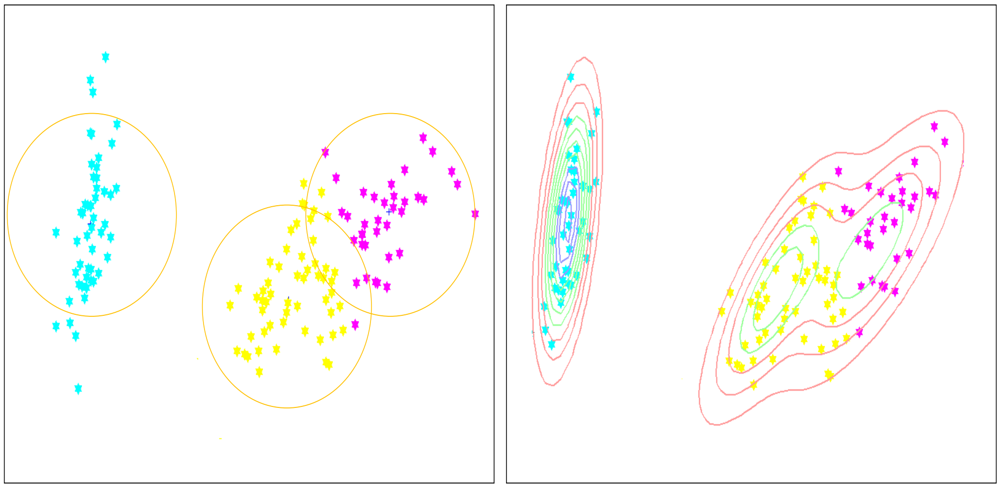


Figure 4.2: Circle and ellipse-based representation

From a geometrical point of view, a GMM-based representation with the elliptical shape allows more flexibility than circles. Figure 4.2 shows the boundaries created using both approaches. Unlike the circle-based model, in the GMM-based model, there is a probabilistic representation of clusters, and consequently the prediction used for attachment of an instance to a cluster is probabilistic. In such a probabilistic model, the probability of attachment of a point to a cluster in the boundary of clusters is less than the center of the cluster.

In the following sections of the chapter, fundamental concepts are first described and then the algorithm is explained that is used to find a suitable GMM for each cluster.

4.1 Background

In this section, a formal definition of GMM is presented.

Definition 1. A Gaussian Mixture Model is a combination of sets of normal distributions. Given a feature space $f \subseteq R^d$, a Gaussian Mixture Model $G : f \to R$ with n components is defined as:

$$G(x) = \sum_{i=1}^{n} w_i \mathcal{N}_{\mu_i,\sigma_i}(x)$$

$$\mathcal{N}_{\mu_i,\sigma_i}(x) = \frac{1}{\sqrt{(2\Pi)^d \|\sigma_i\|}} e^{\frac{1}{2}(x-\mu_i)\sigma_i^{-1}(x-\mu_i)^T} \tag{4.1}$$

A GMM with n components is defined with a set of centers μ_i, with covariance around these centers σ_i where $i = \{1, \cdots, n\}$. All centers and their related covariance are combined using the weights w_i, $i = \{1, ..., n\}$, generated based on the number of neighbors that belongs to each center point.

How can a proper GMM be found for each cluster? Each cluster consists of a set of instances. A collection of these instances generates different distributions in each cluster. In other words, each cluster consists of sets of regions that are connected, using some common instances among these regions. The goal of density-based clustering algorithms is to connect dense regions to from a cluster. This fact gives an idea of how to represent each cluster. Each cluster with a set of dense regions is represented by the centers of these regions and their related statistical information. In density-based clustering methods, centers of dense regions are called core points. The proposed algorithm in this chapter SGMM, finds the core points in each cluster considering each core point in a cluster as a center of GMM component. To clarify the definition of center of the dense region, we define the concept of core point.

Definition 2. In dataset D for a given k as number of neighbors and radius r, an instance O_i is a core point if:

$$\{O_i \in C | \forall O_j \in D, \|d(O_i, O_j) < r\| \geq k\} \tag{4.2}$$

where C is a set of core points and ($\|\|$) indicates the number of instances with distance less than r from the core point. The chosen magnitudes of k and r are critical in identifying valid clusters. Core points are used to generate the backbone and summary of a cluster. In recent work by Chaoji, using a given k and core points, the backbone of the cluster is detected [Chaoji et al., 2011]. The same idea is used here, but with concentrating on decreasing the number of core points, and the time required for finding the core points.

The idea for summarization of each cluster in our algorithm is based on Gaussian Mixture Model. Since the use of GMMs preserves the distribution of data using a set of normal distributions, it is a good candidate for summarizing any cluster. An EM algorithm could be applied to each cluster to find its GMM representation, but the EM algorithm needs to have knowledge of the number of components for each GMM. Since the number of GMM components is unknown, a new algorithm SGMM, is proposed in place of EM. In SGMM, the number of components is found with a recursive algorithm, based on which the GMM for each cluster is defined.

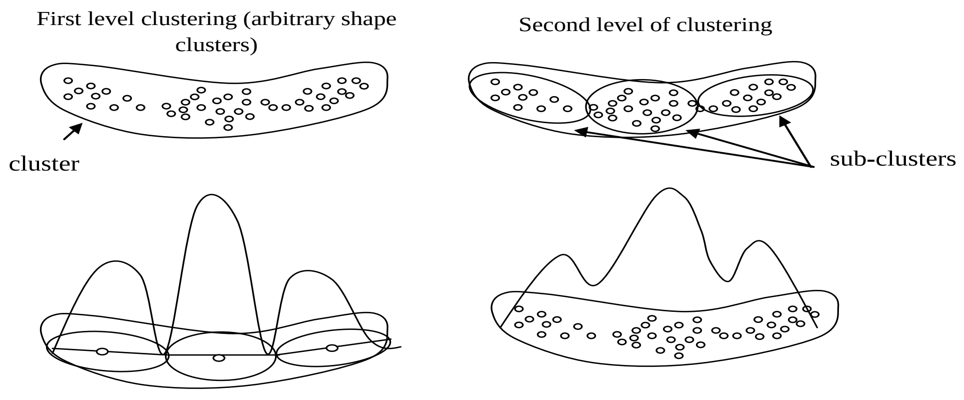

First level clustering (arbitrary shape clusters)
Second level of clustering
cluster
sub-clusters

Figure 4.3: The two-levels of clustering

4.2 Summarization Based on GMM (SGMM)

Instead of keeping all instances for each cluster, a GMM is used to represent each cluster. The main contribution in this part is using clustering inside clustering. In other words, after building clusters, each cluster is itself a target of another clustering algorithm. The second level of clustering is applied to summarize the clusters. Figure 4.3 depicts this idea. For each cluster, a normal distribution is built based on the instances that are placed in each sub-cluster. By combining the normal distributions of sub-clusters, a GMM is generated for the entire cluster. Using two -level clustering, enables the summarization to be more accurate and faster. The SGMM algorithm find the sub-clusters for each cluster.

The Summarization based on Gaussian Mixture Model (SGMM), has three main steps, depicted in Figure 4.4. First, a set of instances called core points are found. These instances are representative of a cluster, and they are

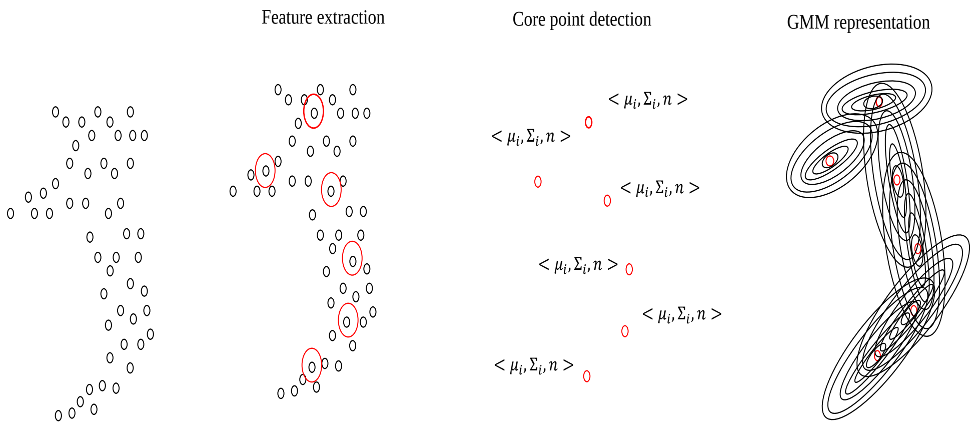


Figure 4.4: The three SGMM steps used to summarize each cluster

able to generate the original cluster as needed. After detecting the backbone instances, is absorption step, where the instances attached to the core points are absorbed and thus represented by the core points. By introducing a new feature set for each instance, the cluster is summarized while preserving its original distribution. Finally, each cluster is represented by a GMM. Each of these steps is described in more detail in the following paragraphs.

Finding Core Points

The algorithm starts with the core point detection phase, with a radius as an input parameter. The radius is employed to find dense regions. Based on the input radius, the number of neighbors is found for each instance. A temporary list is made for possible core points; at first, all instances are in the list. Another list contains final core points. Every instance with more neighbors than other instances is a good candidate to be a core point. Algorithm starts with an instance with a maximum number of neighbors and puts it on the list of core points. A chosen instance is representative of all its neighbors; so the neighbors can be removed from the temporary list of core points. After removing the first core point neighbors, the next step is to find in temporary list the next instance with the maximum number of neighbors. This instance is added to the list of core points, and then its neighbors are deleted from the temporary list. These steps are repeated recursively for the rest of the instances in the temporary list, so identifying all possible core points. Therefore, by setting a proper radius and after all iterations, the number of core points has been specified. The core points are representative of structure and distribution of clusters. the In the following, we present a new definition for core point based on our goal.

Definition 3. Consider r to be a radius for instance O_i, and $N(O_i)$ the set of neighbors of instance O_i.

$$N(O_i) = \{O_i \in D | dist(O_i, O_j) < r\} \tag{4.3}$$

where dist(O_i, O_j) is the Euclidean distance between instances O_i and O_j and D is the dataset.

Definition 4. A core point is a point that has more neighbors than all its neighbors:

$$\{O_i \in C | O_j \in N(O_i), \|N(O_i)\| > \|N(O_j)\|\} \tag{4.4}$$

where dist(O_i, O_j) is the Euclidean distance between instances O_i and O_j and D is the dataset.

Algorithm 4.1 shows the pseudocode for the process of finding the core points.

Absorption and Cluster Feature Extraction

The goal of summarization is to find a good representation of each cluster.

Algorithm 4.1: Pseudo code for finding the core points

```
Data: all instances in cluster $C$
Result: core points of cluster $C$
for $i = 1 : Size(C)$ do
    for $j = 1 : Size(C)$ do
        if $dist(O_i, O_j) < r$ then
            N[$O_i$]=N[$O_i$]+1;
        end
    end
end
SortedList =Sort(N)
while $SortedList \neq empty$ do
    add the next element $O_i$ in SortedList to core points list ;
    remove the neighbors of $O_i$;
    update SortedList ;
end
```

Core points are the only instances that are preserved in each cluster, while the rest of the instances in the cluster are removed. After finding all core points in each cluster, the next step is to define a cluster using its core points. Note that considering only the core points, the cluster distribution cannot be represented. Therefore, to represent cluster characteristics, the core points have to be accompanied by a set of related features. A set of features is necessarily defined for each core point that are a good representative of the cluster distribution.

Definition 5. (Core point Feature) (CF) Each core point is represented by a triple $CF_i =< c_i, \sigma_i, w_i >$.

In this definition c_i is the core point and σ_i is the covariance calculated using the core point and all instances in its neighborhood. $w_i = n/CS$ is the weight of the core point, n is the number of instances in the neighborhood of core point c_i, and CS is the cluster size. w_i shows the proportion of instances in the neighborhood of the core point c_i. Using the features of each core point, instances scattering around each core point are estimated, without keeping the entire set of instances in the neighborhood.

GMM Representation of Clusters

After finding the core points and the features, a GMM is generated for each cluster. Based on all $CF_i, i = \{1, \cdots, m\}$, a GMM is defined over a particular cluster. Each component for the GMM is created using a core point, a covariance and a weight. In the formula in Equation 4.1, μ_i is the centre of the i_{th} GMM component, which is set to the coordination of the core point and therefore, $\mu_i = c_i$. The covariance is set to the covariance of i_{th} core point covariance, that is σ_i. The weight assigned to each component is the weight of the core point and w_i in the Equation 4.1 is set to the weight of the core point c_i which is w_i.

To use a cluster as a class for classification, the closest cluster to an incoming instance needs to be identified. In an arbitrary shape cluster, to find the closest cluster to a new instance, it must be compared with all instances in the cluster. A GMM-based presentation is used to find the closest cluster by feeding the new instance to the GMM formula in Equation 4.1 to find the membership value.

Each summarization technique has to preserve the original shape and the distribution of the data. Data summarization using GMM preserves both. In SGMM, the centers are selected in such a way that they follow the general structure of the data. The algorithm begins with dense regions in the cluster, and then continues to find the most scattered part to cover all instances in the cluster. In the SGMM method, the core points are the ones that are in the center of dense regions, and they cover all data. Therefore, for each region a representative instance is chosen, and a collection of the representative

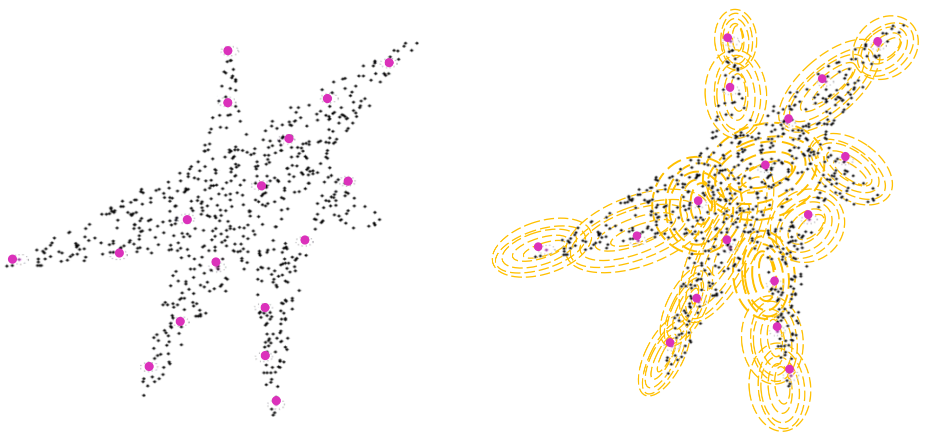


Figure 4.5: In the illustration on the left, points highlighted in the dataset represent the core points detected by the SGMM algorithm. These points successfully indicate the general structure of the cluster. On the illustration on the right, the contour-plots represent the GMMs of cluster.

instances represent the entire cluster.

Figure 4.5 shows the core points generated for a cluster. The cluster in the figure was generated using Matlab, and the SGMM algorithm applied to the cluster. The results show that the core points generated by SGMM follow the general structure of the cluster.

4.3 Experimental Results

The SGMM algorithm is an accurate model in comparison with other cluster summarization approaches. In this section, SGMM performance and efficiency are compared with that of similar approaches. First, the SGMM goals are reviewed and then different experiments are described to find the effectiveness of the method.

The first goal of the SGMM algorithm is to represent clusters using GMM to label incoming instances in two-layer structure. Therefore, the first empirical step is to verify the accuracy of the SGMM model in labeling new instances. The second goal is to function without undue memory needs or complexity. Since most anomaly detection algorithms are used in an online environment, it has to be confirmed that our approach consumes less time and memory in comparison with other algorithms. Furthermore, in some applications like network security, there is a need to regenerate data in some periodic fashion to find out the changes that happen during the time. The SGMM is able to regenerate original data using the relevant GMMs.

Based on these three goals, three sets of experiments were run on different datasets to assess the capabilities of the SGMM. Both synthetic datasets and the real datasets were used in the experiments. All experimental results were generated using Matlab running on a machine with an Intel 3.4 GHz CPU and 4GB of memory.

The first test was done with a synthetic dataset, the results of which are shown in Figure 4.6. This dataset is used to visualize the effectiveness of the proposed algorithm. The figure shows on the left, four clusters and set of instances used in the testing phase. In the right side of the figure, each cluster is represented by a GMM depicted with contour plots.

The UCI datasets were used in experiments to evaluate the accuracy of the algorithm on real datasets. These datasets and their features are presented in Table 4.1. First, some classes were chosen as normal, with the rest as abnormal. The abnormal instances were then considered as the testing dataset. The normal classes were divided into two parts, one of which formed the training set, and the rest added to the testing set. The training dataset consisted of only normal instances, with the testing dataset comprised of

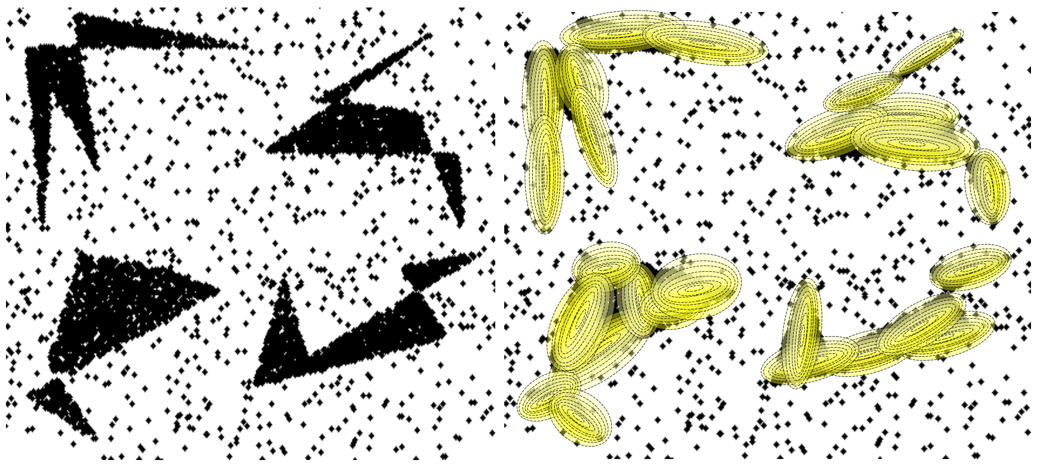


Figure 4.6: Synthetic dataset. Four clusters with arbitrary shape detected by the DBSCAN algorithm and on the right each cluster is represented by GMMs.

normal and abnormal instances.

Dataset	Dataset size	# of features	Normal classes	Anomalous classes
KDDCUP99	10000	41	1	2, 3, 4, 5
Segment	2310	19	1, 2, 3, 4	5, 6, 7
Synthetic	5000	2	1, 2, 3	4

Table 4.1: Summary of the UCI datasets

4.3.1 Clustering Accuracy

As mentioned, clustering is used as a pre-processing step toward anomaly detection. The normal data are clustered into clusters which were considered to represent normal behaviors. Then, a membership value is calculated for each new instance. If the new instance belongs to a cluster, the membership value exceeds the threshold and it is deemed normal, otherwise an anomaly. Three datasets shown in table 4.1 were used to test the accuracy of anomaly detection. As far as could be surmised, there has been no previous academic work published that uses GMM for classification as is done in the proposed method.

The core points were created using a simple method, that requires less processing time and memory space. Therefore, there can be a doubt in the accuracy of categorizing using the clusters created with our method? The next experiment shows that the approach does not decrease the accuracy of categorizing using clusters.

Figures 4.7 and 4.8 depict the comparison of the SGMM with other methods. The only approach that has been used for an arbitrary shape cluster to label new instance has been to consider all members of clusters, which is the naive approach. The naive approach simply finds the distance of a new instance to all cluster members; a cluster that has the minimum distance to the new instance is the one that the new coming instance should belong to. Another experiment was to use the GMM created by the ABACUS method [Chaoji et al., 2011], which is a well-known method for summarizing arbitrary shape clusters. Using ABACUS method the core points are specified and used as the centers of GMM components. The ROC curve of ABACUS depicts the accuracy of GMM created by ABACUS in labeling new instances. The membership value threshold is the main parameter for SGMM, ABACUS, and naive approaches based on which, the ROC curve is produced.

Based on the detection and false alarm rates for the KDDCUP99 dataset, the ROC curve is depicted in Figure 4.7. It shows that while ABACUS is

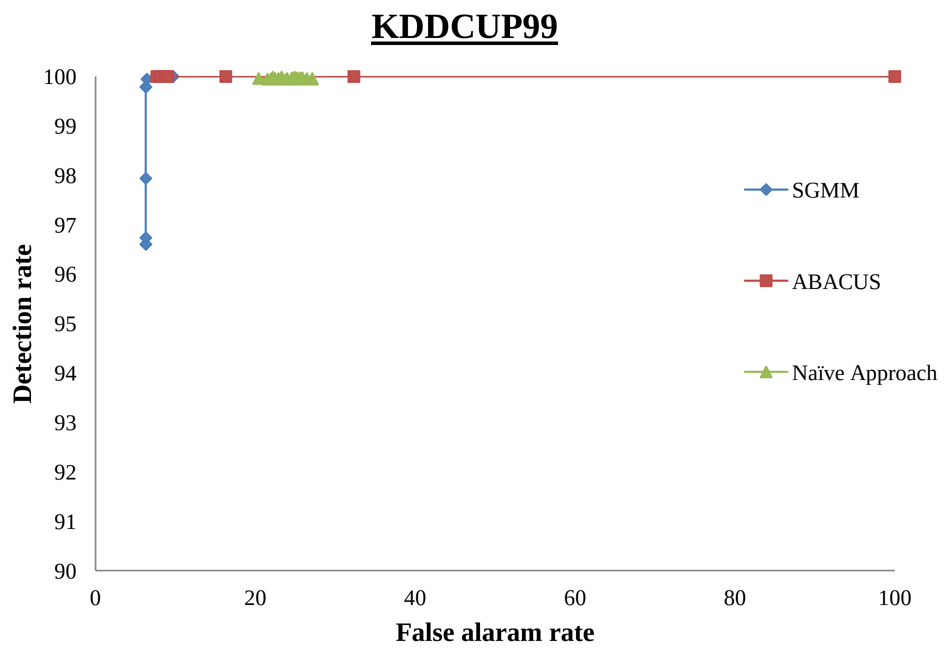


Figure 4.7: False alarm and detection rate for the KDDCUP99 dataset

more time-consuming and finds too many core points, the proposed method has the same and sometimes better accuracy. The result comes from 30 independent runs; the variance on the best result is less than 1.3 for the detection rate and 1.5 for the false alarm rate. The results in the figure are the best ones for each algorithm.

Figure 4.8 shows the results for a synthetic dataset. The accuracy of SGMM is better than that of both the naive and ABACUS methods. This result shows that in spite of using clustering in categorizing new instances, the accuracy is still adequate and comparable to the accuracy of the other classification methods.

Table 4.2 shows results for other datasets, confirming that summarizing

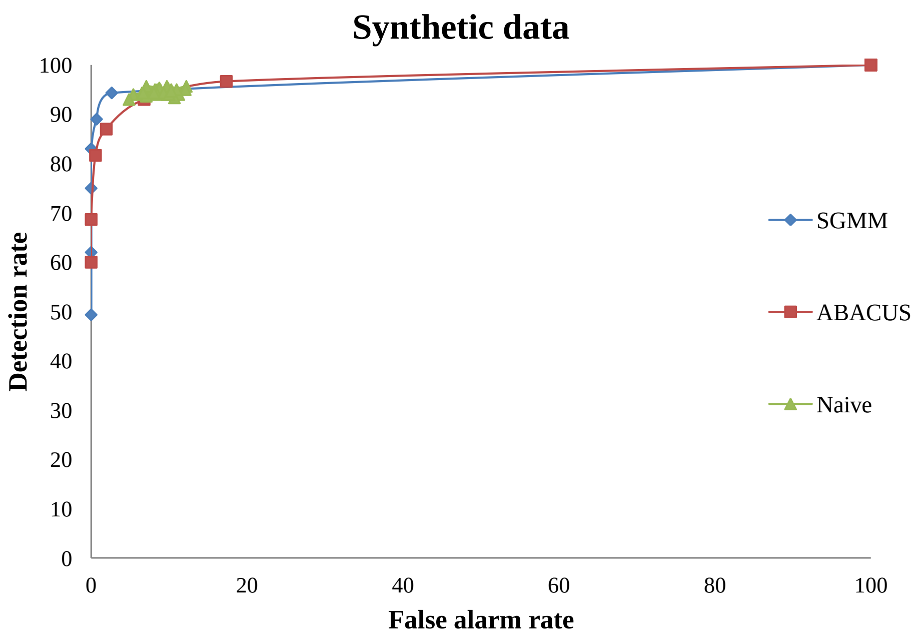


Figure 4.8: False alarm and detection rate for the synthetic dataset

and clustering data using the SGMM method outperforms the ABACUS method. The first value in each ordered pair shown in Table 4.2 is the false alarm rate, and the second the detection rate. The detection rate is 5% higher on average and the false alarm rate 3% lower. However, the main goal is not to increase the accuracy. As will be shown in the next section, the goal is instead to show that a smaller number of core points can be found in less time but still getting at least the same result as other methods. The result in Table 4.2 covers 30 independent runs. The variance of the best result is less than 1.5% for the detection rate and 1.7% for the false alarm rate. The results in the figure are the best ones for each algorithm.

Dataset/Algorithm	SGMM	ABACUS
Synthetic data	$(2\%, 94\%)$	$(1.9\%, 87\%)$
KDDCUP99	$(6.3\%, 100\%)$	$(9.0\%, 100\%)$
Segment	$(33.8\%, 90.3\%)$	$(32.8\%, 94.5\%)$

Table 4.2: Accuracy of the SGMM and ABACUS methods in anomaly detection. The first value in each ordered pair is the false alarm rate, and the second the detection rate.

4.3.2 Running Time and Memory Usage

Two main concerns in the area of summarization are minimizing the time spent to find the summary of a cluster, and the number of instances preserved for it. Table 4.3 shows the time spent to find the summary of a cluster for different datasets. The table shows that using the SGMM decreases the time required to summarize each cluster. The SGMM method used only one iteration to find core points, while the ABACUS method ran many iterations; therefore, SGMM is faster.

Table 4.4 shows that the SGMM summarized clusters with fewer core points. The reason lies in the efficiency of the method the SGMM algorithm uses to find core points. If an instance is a core point, all instances in its neighborhood are removed from set of possible core points, and are not considered further. That is why, both the processing time and space complexity are reduced.

Tables 4.3 and 4.4 show the time and space used by the SGMM and ABACUS. In terms of time, the SGMM is approximately three times faster

Dataset/Algorithm	SGMM	ABACUS
Synthetic	388_{sec}	1261_{sec}
KDDCUP99	222_{sec}	836_{sec}
Segment	7_{sec}	19_{sec}

Table 4.3: Time complexity for the ABACUS and SGMM methods

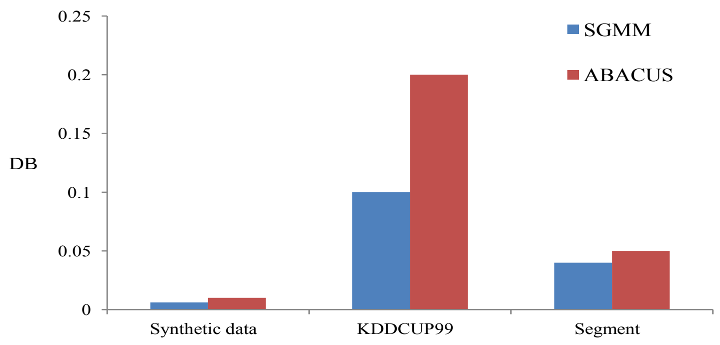


Figure 4.9: Running Time of ABACUS and SGMM methods

Dataset/Space	SGMM	ABACUS
Synthetic	51	120
KDDCUP99	46	272
Segment	52	55

Table 4.4: Number of core points for the ABACUS and SGMM methods

than the ABACUS algorithm. In terms of memory usage, on average the usage of the SGMM is at least half of the ABACUS method. To show the difference better the bar charts on Figure 4.9 depicts the time difference as well.

4.3.3 Clustering Goodness

To test the ability of the SGMM algorithm in data regeneration, a set of experiments was set up in which the dataset was summarized as a GMM, and then the GMM was used to re-generate the dataset. The performance of the SGMM was compared with that of the ABACUS method, which is a well-known method for summarizing arbitrary shape clusters [Chaoji et al., 2011]. The difference between the original and regenerated datasets based on core points shows the strength of the summarization algorithm. Experimental results show that the SGMM method summarized the dataset better than the ABACUS method. To visualize the results, synthetic data with four clusters were generated.

Figure 4.10 from left to right for each row shows the original dataset, the core points of each cluster and the dataset set regenerated using core points, using the SGMM (top row) and ABACUS (bottom row) methods. The figure demonstrates that the core points follow the original structure of the clusters, and that the regenerated clusters are similar to the original ones. The summary generated by the SGMM method regenerated the original dataset better than the ABACUS method. The difference between the accuracy of summarization of the two methods is most clearly distinguished in the cluster with a the shape of a five-pointed star. The SGMM summary regenerated the data with a star shape, but the ABACUS method could not regenerate the same shape as accurately. This figure is just for visualization of the result. To quantitatively assess the efficiency of the algorithms, cluster validity indexes are used.

There are two main ways to measure the validity of clusters: internal and external evaluation indexes.

Internal Evaluation

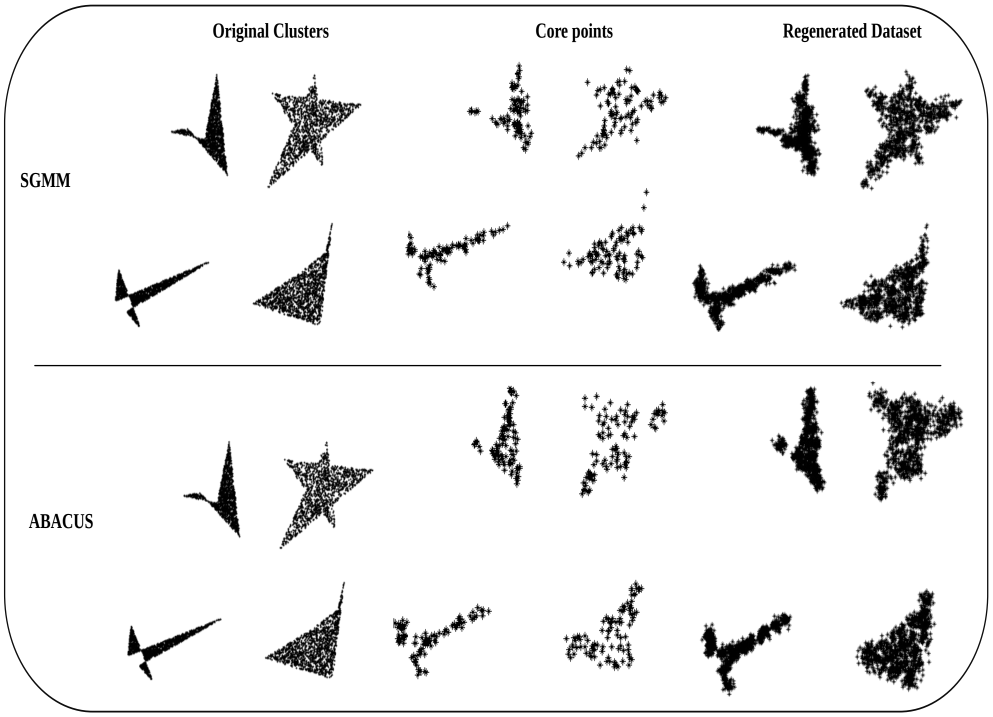


Figure 4.10: In each row, the first dataset pair is the original dataset, the second pair is the core points and the third pair is the regenerated dataset, using the core points generated with the ABACUS and SGMM methods

The internal evaluation is based on the data, with no ground knowledge. Internal evaluation is based on high similarity inside clusters and high dissimilarity among clusters. There are three indexes for internal evaluation of clusters:

- Davies-Bouldin (DB) index [Davies and Bouldin, 1979]
- Dunn index [k. Dunn and Dunn, 1974]
- Silhouette coefficient [Rousseeuw, 1987]

External Evaluation

External evaluation is based on some ground knowledge of data, such as the number of classes that exists in the data. External evaluation indexes measure how closely the clusters conform to the pre-determined benchmark classes. The external cluster evaluation indexes include:

- Rand measure [Rand, 1971]
- F-measure [van. Rijsbergen, 1979]
- Jaccard index [Tan et al., 2005]
- Fowlkes-Mallows index [Fowlkesa and Mallowsa, 1983]
- The Mutual Information [Teukolsky et al., 2007]
- Confusion matrix [Stehman, 1997]

Details of these indexes are in given in Appendix A.2. In applications for which there is ground knowledge available of the data, external evaluation is better than the other approaches. However, since in most clustering applications there is no background information, using an internal evaluation is often better option. Different studies have been set up to find the best metric for clustering, with the conclusion that the best option is application-dependent [Rendn et al., 2011] [Bruna et al., 2007].

In anomaly detection applications, there is no background knowledge of the number of classes. Therefore, internal indexes have been used. Among the three internal indexes, the Silhouette coefficient has high computational complexity when used to measure the distances between each two pairs inside and outside the clusters. To decrease the computation time and allow working with larger datasets two metrics were employed; the Dunn and DB indexes.

The Dunn index [k. Dunn and Dunn, 1974] is a validity index that identifies compact and well-separated classes, and is defined by Equation 4.5 for a specific number of classes:

$$D_{nc} = \min_{i=1,\cdots,nc}\{\min_{j=i+1,\cdots,nc}\{\frac{dist(c_i, c_j)}{\max_{k=1,\cdots,nc} diam(c_k)}\}\} \tag{4.5}$$

Here nc is the number of classes, and $dist(c_i, c_j)$ is the dissimilarity function between two classes c_i and c_j. A large value of this index indicates the presence of compact and well-separated classes. In the experiments, the Dunn index was first calculated for the original datasets. Then, the dataset was summarized and re-generated using the final core points of GMMs. Dunn indexes were calculated for the original and regenerated datasets. Finally, the difference is calculated between the Dunn indexes of the original dataset and the one that of the regenerated one. The test result shows that the SGMM method regenerates the data, following the shape and distribution of the original data. Table 4.5 shows the results of this experiment. Each value in this table is the average of the results from 30 independent runs.

Dataset/Algorithm	SGMM	ABACUS
Synthetic data	0.02	0.07
KDDCUP99	0.02	0.03
Segment	0.0	0.01

Table 4.5: Differences of the Dunn indexes for the original and regenerated datasets

The closer the value to zero, the better the result obtained. The SGMM

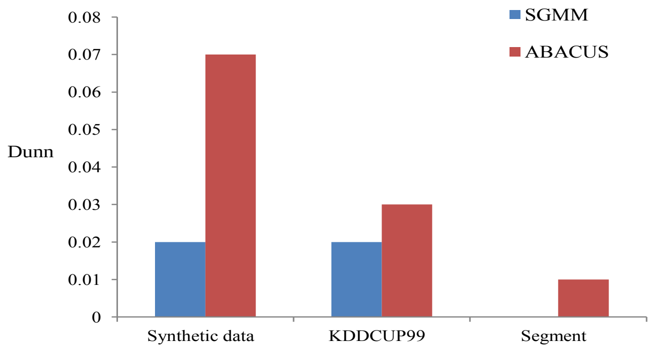


Figure 4.11: Differences of the Dunn indexes for the original and regenerated datasets

consistently outperforms the ABACUS method by far. For example in the case of synthetic dataset, the difference between the indexes for the dataset generated by the SGMM method and the original dataset is almost 0.02 while for the ABACUS method the difference is around 0.07. The SGMM method was superior in recreating the data distribution of the original dataset. To show the difference better the bar charts on Figure 4.11 depicts the difference as well.

The experiment was repeated using the DB cluster index, which is a function of relating the ratio of the sum of within-cluster scatteredness to the between-cluster separation. The DB index is defined in Equation 4.6:

$$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left[\frac{S_n(Q_i)+S_n(Q_j)}{S_n(Q_i,Q_j)}\right] \tag{4.6}$$

n is the number of clusters, S_n is the average distance of all objects of the cluster to their cluster centre, and $S_n(Q_i, Q_j)$ is the distance between clusters centers. The ratio is small if the clusters are compact and far from each other. Consequently, the DB index will have a small value for good clustering. Similar to the test using the Dunn index, distances were found for the original and regenerated data. Table 4.6 shows the experimental results in assessing the DB index for different datasets. To show the difference better the bar charts on Figure 4.11 depicts the difference as well.

Dataset/Algorithm	SGMM	ABACUS
Synthetic data	0.006	0.01
KDDCUP99	0.1	0.2
Segment	0.04	0.05

Table 4.6: Difference of DB Index for original and regenerated dataset

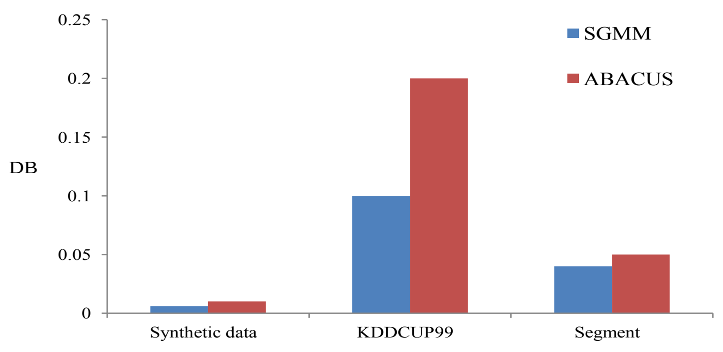


Figure 4.12: Difference of DB Index for original and regenerated dataset

Results based on assessing the Dunn and DB indexes show that the SGMM method generates a more accurate summary of datasets, and in case of regenerating the original dataset, the SGMM is better at creating the original pattern. More extensive results are shown the in Appendix A.3.

4.4 Conclusion

In this chapter, a new approach was presented for summarizing the arbitrary shape clusters, which can be used to generate accurate clusters and requiring less expenditure of time and memory. The proposed SGMM algorithm, represents each cluster with a GMM using sets of core points. Each GMM is representative of the distribution of the set of members in a cluster.

Moreover, the SGMM is able to successfully identify the closest cluster for each incoming instance. Experimental results based on evaluating Dunn and DB indexes confirmed that the distribution of clusters is preserved after summarization. The regeneration ability of the SGMM method to regenerate clusters was empirically found to be better than that of the ABACUS. The ROC curves presented in the experimental results show that while the time and memory requirements were three times less than those of the ABACUS method, the accuracy of labeling new instances was the same as that of ABACUS. In some cases, the detection rate was increased 5% and the false alarm rate was decreased 3%.

Chapter 5

Collective Labeling

The central part of the two-layer structure introduced in Chapter 3 is the updating phase. From Figure 3.5, all clusters are represented as GMMs, which makes everything in the two-layer structure based on GMMs. All clusters are updated on detection of any concept drift. With GMMs used to represent clusters the updating phase must be GMM-based.

A simple method for updating each cluster is to update it for each new coming instance. The main problem with this approach is that it is time-consuming, as well as being sensitive to noise. To mitigate this problem instead of updating clusters with each new instance, these new instances are collected and then are used to update clusters. Based on the two-layer structure, all new instances are sent to the rag bag to be used in updating. This structure saves time by collective updating. To update clusters with new instances, all data in the rag bag are themselves clustered. Based on these clusters, a GMM is estimated in the same way in the training phase. Then the GMMs in the rag bag are merged to current GMMs.

There are three possible actions in updating phase:

- Merge a cluster with a cluster in the rag bag

- Add a new cluster
- Remove a cluster

All new instances must first find their closest cluster. In this case, instead of labeling instances separately, the whole new cluster gets a label. Therefore, there is a need to measure the distances for clusters in the rag bag and all current clusters. A small distance between two GMMs shows that they have been generated based on the same distribution, and can be considered as similar behavior. With this knowledge, the clusters in the rag bag can be labeled by finding the distances between the new GMMs and the current GMMs. If the similarity between a GMM in the rag bag and one of the current GMMs is more than a threshold, the cluster in the rag bag have been generated by the same distribution as that of current cluster. Consequently, the cluster in the rag bag and its instances are labeled as normal and are attached to that cluster. In this chapter the approach for labeling a group of instances collectively is discussed and the details of updating clusters are presented in the chapter 6.

We propose an approach called Collective Probabilistic Labeling (CPL), to label clusters. Figure 5.1 depicts the the main idea of the of CPL method to label clusters, by finding close clusters.

CPL has two main steps:

- Cluster data in the rag bag
- Measure the distance between two clusters, and label the new clusters

With clustering of the data in the rag bag, all new instances in a cluster are labeled, based on their collective characteristics represented by a GMM. Therefore, since all instances are labeled based on group behavior, the effects

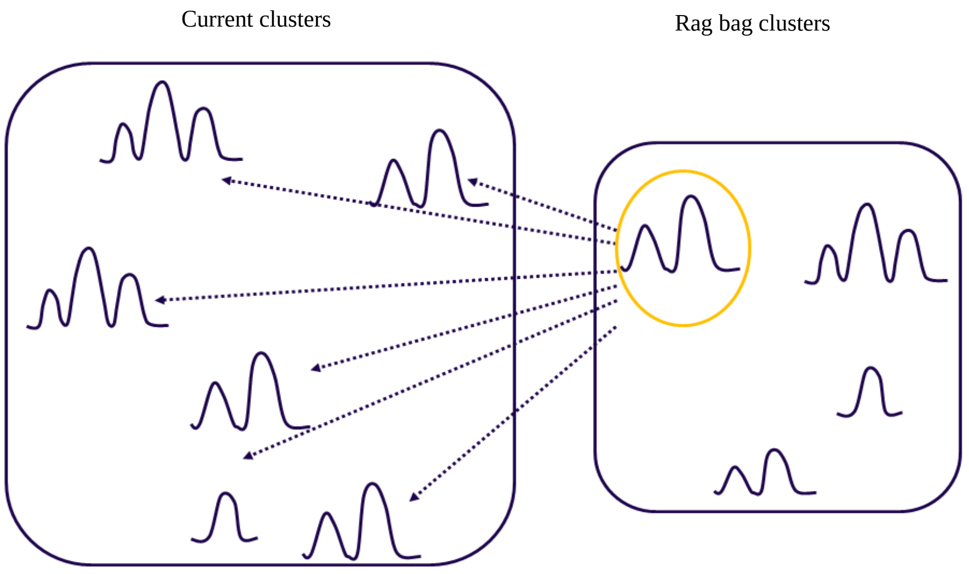


Figure 5.1: Finding the right cluster for new instances collectively

of noise can be reduced using this method. The only computation required in the CPL method is finding a cluster with similar characteristics to those of the rag bag clusters, and then assigning a label to the clusters in the rag bag and, consequently, to its entire instances.

In the clustering phase of new instances, DBSCAN approach is employed, likewise the training phase. DBSCAN finds the number of clusters automatically while there is no direct knowledge available of the shape and the number of new clusters. Next, the distance between new and current clusters must be measured. All clusters are represented using GMMs, and the distance measure has to quantify the distance between two GMMs. In the next section, a new approach is described to measure the distance between two GMMs.

5.1 An Improved GMM Distance Measure Based on Kullback-Leibler Distance

In the proposed method, new instances are labeled collectively using their respective GMMs. Finding a proper measurement affects the whole updating process and can improve the accuracy of the labeling phase. In this section, we describe the proposed measure which is based on the Kullback-Leibler distance.

Based on the definition in chapter 4, a GMM is defined as follow:

$$G(x) = \sum_{i=1}^{n} w_i \mathcal{N}_{\mu_i,\sigma_i}(x)$$

$$\mathcal{N}_{\mu_i,\sigma_i}(x) = \frac{1}{\sqrt{(2\Pi)^d \|\sigma_i\|}} e^{\frac{1}{2}(x-\mu_i)\sigma_i^{-1}(x-\mu_i)^T} \tag{5.1}$$

Using the Equation 5.1, the Kullback-Leibler distance between two normal distributions is [Kullback and Leibler, 1951] .

$$KL\left(\mathcal{N}_0 | \mathcal{N}_1\right) = \frac{1}{2}\left(tr\left(\sigma_1^{-1}\sigma_0\right) + \left(\mu_1 - \mu_0\right)^T \sigma_1^{-1}\left(\mu_1 - \mu_0\right) - k - ln\left(\frac{det\sigma_0}{det\sigma_1}\right)\right) \tag{5.2}$$

where $\mathcal{N}_0$ and $\mathcal{N}_1$ are normal distributions based on definition in Equation 5.1. μ and σ are mean and covariance of a normal distribution. $det(A)$ is determinant of matrix A. $tr(A)$ is the trace of an $n-by-n$ square matrix A, which is defined to be the sum of the elements on the main diagonal.

The Kullback-Leibler distance is frequently used to measure the distance between two normal distributions, not specifically two GMMs. Different measures were introduced to measure the distance between two GMMs. The available measures are designed based on a specific application's requirements. For example, variational distance introduced in by Hershey, et.al. works well in speech recognition [Hershey et al., 2007]. In this method, a combination of distances of two normal distributions is considered.

A Monte-Carlo method approximates the Kullback-Leibler divergence between two GMMs through the use of efficiently large sampling techniques. This method is efficiently applied in image processing applications, in which the entire set of instances in images need to be considered. Due to the high computational cost of this approximation, Goldberger et al. [Goldberger et al., 2003] proposed a matching-based approach for measuring distance between two GMMs. In this approach, instead of computing the distance between each two normal components in two GMMs, only the distances of the closest components are considered, and they are combined using their relative weights:

$$\begin{gathered} D_{Goldberger}\left(G_a|G_b\right) = \\ \sum_{i=1}^{n} w_i^a \left(KL\left(\mathcal{N}_i^a|\mathcal{N}_{\pi(i)}^a\right) + \log\left(\frac{w_i^a}{w_{\pi(i)}^b}\right)\right) \\ Where \\ \pi\left(i\right) = argmin_j\left(KL\left(\mathcal{N}_j^a|\mathcal{N}_j^b - log\left(w_j^b\right)\right)\right) \end{gathered} \tag{5.3}$$

where G_a and G_b are GMMs, $\mathcal{N}_i$ is the i^{th} normal distribution with w as GMM weights .

Since in our structure GMMs are used to estimate the distribution and instance scattering in each cluster, a distance function is needed that considers the entire shape of the cluster (that is all components in each GMM). In contrast with Goldberger's method, our proposed method considers all pairwise distances of all normal component from both GMMs. A main feature of the proposed metric is that it is symmetric, whereas the Kullback-Leibler distance is not. To find a symmetric distance, the distances of the all components of the first GMM to the second one, and all components of the second GMM to the first one are calculated, and an average of two is calculated.

In the proposed distance measurement method, the distance from the first GMM's components to the second GMM's components are found using the weights of the first GMM, as follows:

$$D(G_a|G_b) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_i^a KL\left(\mathcal{N}_i^b|\mathcal{N}_j^a\right) \tag{5.4}$$

The distance of the second GMM's components to the first GMM's components are found using the weights of the second GMM:

$$D(G_b|G_a) = \sum_{i=1}^{m}\sum_{j=1}^{n} w_i^b KL\left(\mathcal{N}_i^b|\mathcal{N}_j^a\right) \tag{5.5}$$

The final distance is the average of Equations 5.4 and 5.5.

$$D_{clusters} = \frac{1}{2}(D\left(G_a|G_b\right) + D\left(G_b|G_a\right)) \tag{5.6}$$

In the next section, the proposed collective labeling approach and the new distance function are evaluated based on different criteria.

5.2 Evaluation of the CPL Method

In this section, experimental results are presented and the CLB method is compared with the Basic Probabilistic Anomaly Detection method (BPAD) [Mohammadi et al., 2014] and two other well-known methods, one-class SVM [Scholkopf et al., 2001] and LOF [Breunig et al., 2000]. All experimental results were done in Matlab running on a machine with an Intel 3.4 GHz CPU and 4GB of memory.

The algorithms were evaluated based on three criteria. The first criterion is Detection Rate (DR) and the second is the False Alarm Rate (FA). The third is the memory requirement for each method to find the right cluster for the new instances. FA and DR are presented with their respective ROC curves. There is an input parameter for each algorithm, based on which the ROC curve can be drawn. For LOF, the number of instances in the neighborhood of the core point is the main parameter. The membership value threshold is the main parameter for CPL, BPAD, and SVM, based on which, the ROC curve is produced.

As mentioned in experiments of Chapter 4, the training dataset just consisted only of normal instances, with the testing dataset a combination of normal and abnormal instances. 5% noise was added to the test dataset to build noisy datasets. Different datasets (in size and number of features) were selected from the UCI repository. Table 5.1 shows a summary of the datasets and their normal classes.

In the first set of experiments, the datasets were used are 'clean' with no added noise. Each test result is for running an algorithm 30 times, and getting the best result. The variance is less than 2% in detection and false alarm rates. The results in the figure are the best ones for each algorithm.

Dataset	Dataset size	Number of features	Normal classes	Anomalous classes
MagicGamma	19020	10	$1, 4$	$2, 3, 5, 6$
Shuttle	43500	9	$1, 2, 3$	$4, 5, 6$
Waves	5000	40	1	$2, 3$

Table 5.1: Summary of datasets used for CPL testing

Figure 5.2 shows the experimental results for the Waves dataset. The ROC curves in this figure illustrate that the LOF algorithm was not successful in either detecting the anomalies or recognizing normal behavior. For the Waves dataset, and even without any noisy instances, the LOF method cannot compete with the three other methods. The performance of BAPD, with DR and FA of 60% and 18% respectively, was better than the one-class SVM and LOF. The performance of the CPL method is similar to that of one-class SVM and BAPD. The Waves dataset has 40 features, and since the one-class SVM does not work well in higher dimensions, its performance was not good for this dataset.

Figure 5.3 shows the results of the same experiment for the Shuttle dataset which are similar to those for the Waves dataset. CPL, BPAD, and one-class SVM are almost the same based on the DR and FA and they all significantly outperform the LOF. The best result was for the one-class SVM and BPAD, each with 6% FA and 80% DR followed by CPL with 8% FA and 76% DR.

In the next experiment, the anomaly detection methods were applied on MagicGamma dataset, with results shown in Figure 5.4. Here, without adding noise to the datasets, the BPAD method performed better than the other methods. CPL and the one-class SVM showed the same performance, and the both methods outperformed the LOF in terms of DR and FA. For

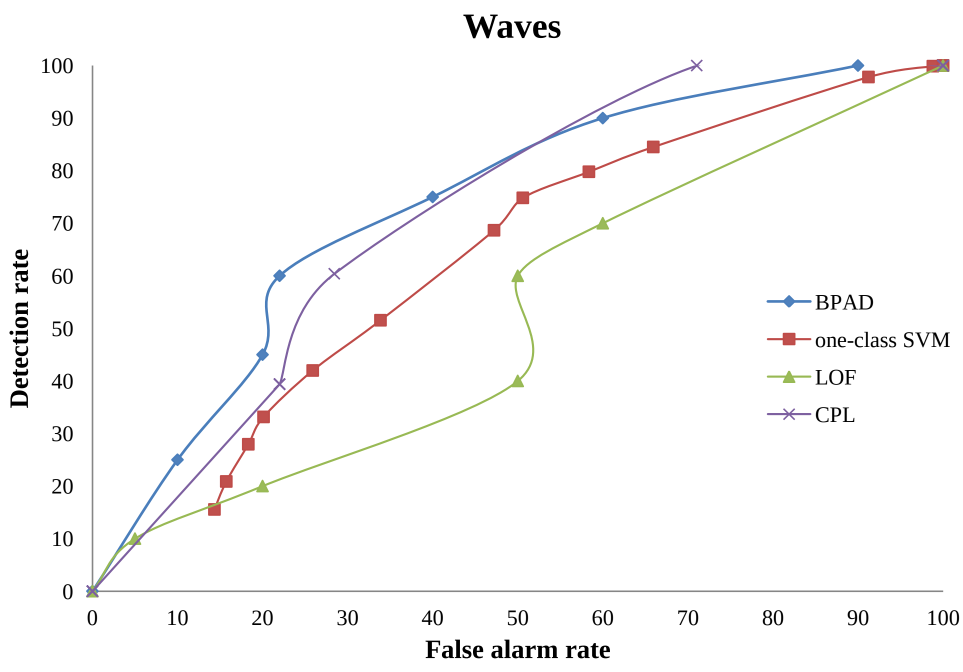


Figure 5.2: The ROC curves of the BPAD, one-class SVM, LOF and CPL methods applied to the Waves dataset

further experiments, the LOF was not included because of its poor performance in comparison with that of BPAD, CPL and the one-class SVM.

In the following, the performance of the anomaly detection methods was measured for use on noisy datasets. A level of noise was added to the datasets, and then DR and FA assessed for these noisy instances. For building the noisy dataset, some instances were first selected randomly, from which some features were selected, and a level of noise added to them. The value added to each feature depended on the maximum and minimum value for the feature. It is 5% of the maximum-minimum values. Figure 5.5 shows the outcome of the experiment for the Waves dataset.

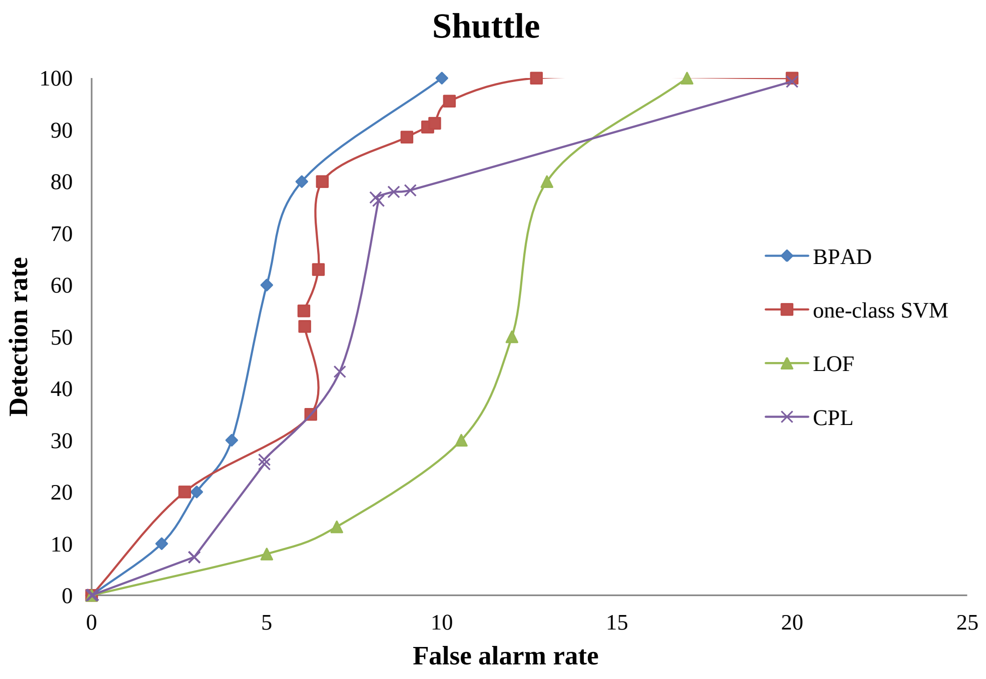


Figure 5.3: The ROC curves of the BPAD, one-class SVM, LOF and CPL methods applied to the Shuttle dataset

A comparison between Figure 5.2 and Figure 5.5 (tests with and without noise on Waves dataset) shows that after adding noise, the detection rate for all three methods is lower. For CPL, the DR before adding noise (in the best case on the ROC curve) is 60%, and the FA is 22%. After adding noise, the DR goes down to 57% and the FA rises to 25%. The same results were observed for BPAD and the one-class SVM. However, the accuracy reduction for CPL in a noisy environment is lesser than that of the one-class SVM and BPAD. This means that with the added noise in the dataset, CPL is more efficient than one-class SVM and BPAD and they are both more noise-sensitive than CPL. For CPL method, the instances were labeled collectively,

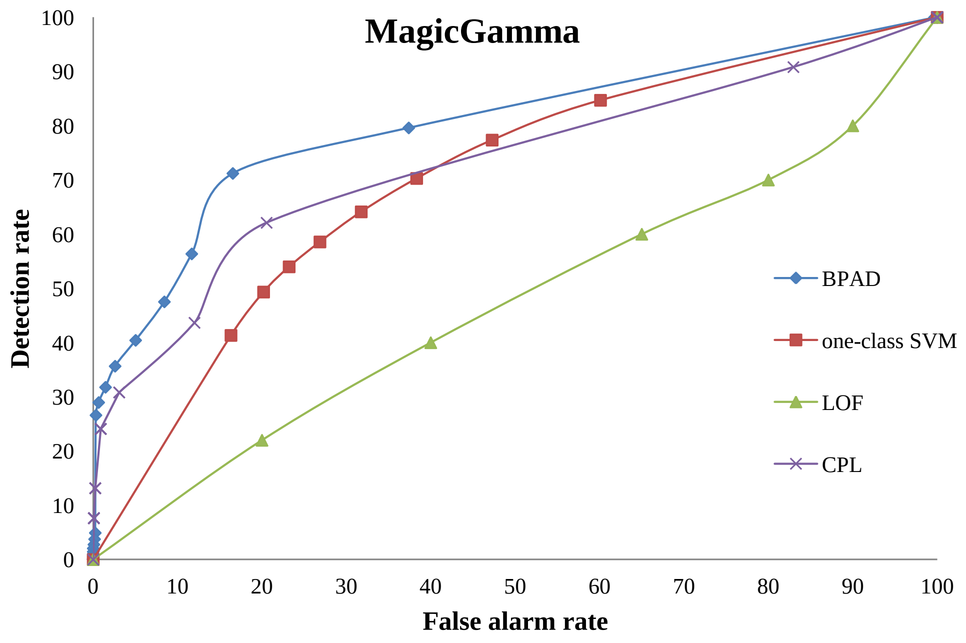


Figure 5.4: The ROC curves of the CPL, one-class SVM, LOF and CPL methods applied to the MagicGamma dataset

and the algorithm labeled the noisy instances correctly as well.

Figure 5.6 shows the experimental results for the noisy Shuttle dataset. Compared with the clean Shuttle dataset (Figure 5.3), CPL was the best method followed by one-class SVM and BPAD algorithms, respectively. In its best point on the ROC curve, CPL shows an 75% detection rate and 10% false alarm rate. For noisy Shuttle dataset CPL was the best-performing method.

In the next experiment, the performance of the three anomaly detection algorithms applied on the noisy MagicGamma dataset was evaluated, with the results presented in Figure 5.7. With the added noise, the accuracy of three methods was reduced although the detection and false alarm rate of

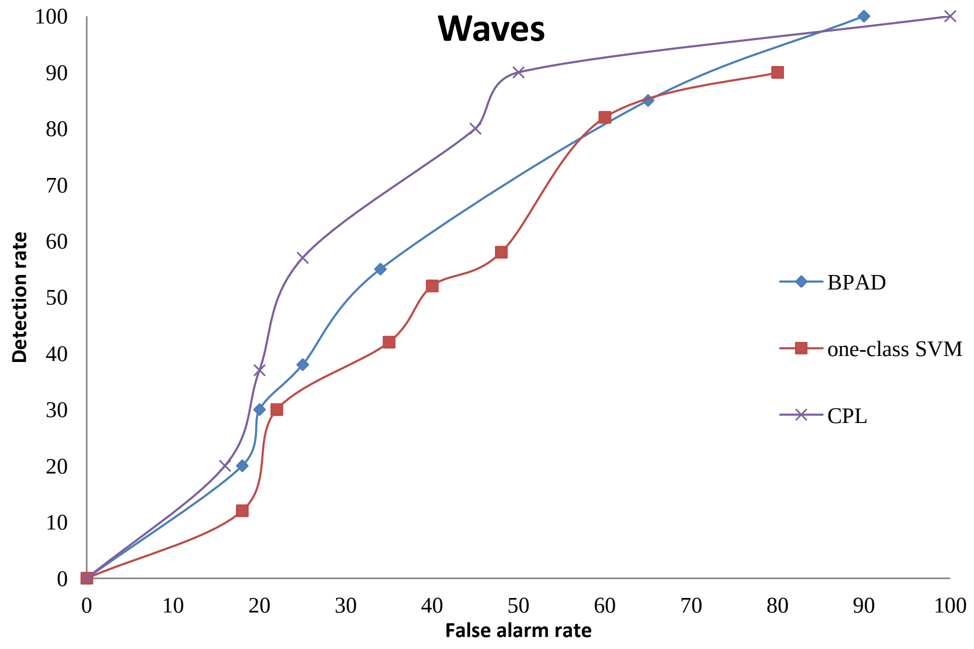


Figure 5.5: The ROC curves of the BPAD, one-class SVM and CPL methods applied to the noisy Waves dataset

the CPL and one-class SVM were almost the same, and both were better than those of BPAD.

Based on the results using the three noisy datasets, in the presence of noise, collective labeling (CPL) is better than the one-class SVM and BPAD. The computational complexity of the one-class SVM is its major drawback. In the next section, we compare the memory complexity of CPL and one-class SVM on different datasets.

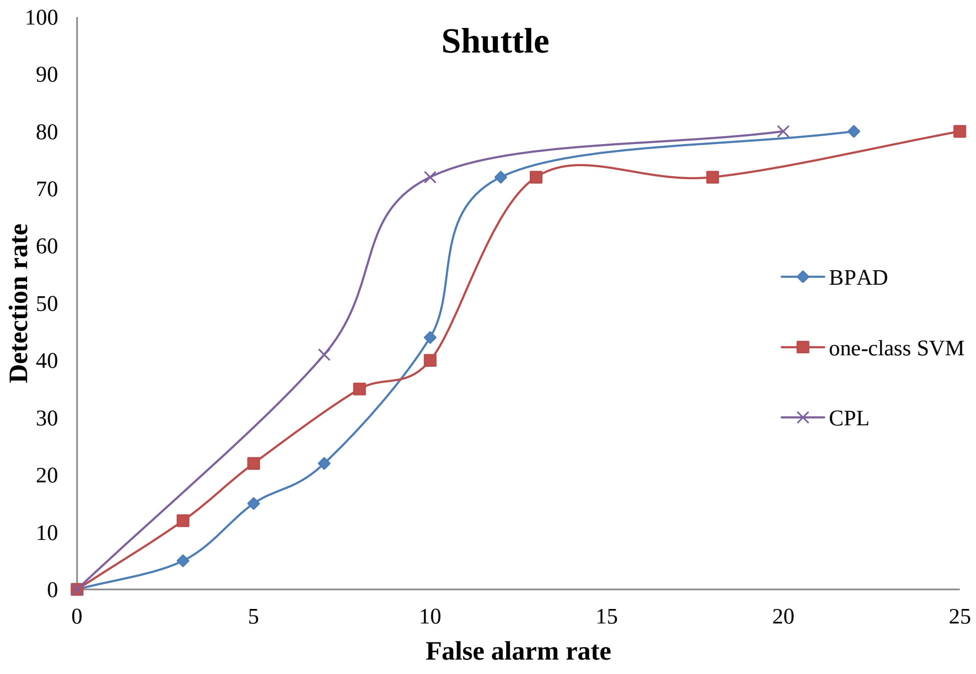


Figure 5.6: The ROC curves of the BPAD, one-class SVM and CPL methods applied to the noisy Shuttle dataset

5.3 Memory Complexity of CPL and one-class SVM

The one-class SVM surrounds the area of each class (in case of anomaly detection the normal instances) with some surfaces. By increasing the number of features, the number of surfaces will be increased exponentially, and this is a challenging drawback for the one-class SVM. In the previous experiments, the one-class SVM and CPL methods performed almost the same in terms of detection and false alarm rate. As such, if the CPL performs better than the one-class SVM based on memory complexity, it could be concluded that, in general, CPL is better than one-class SVM.

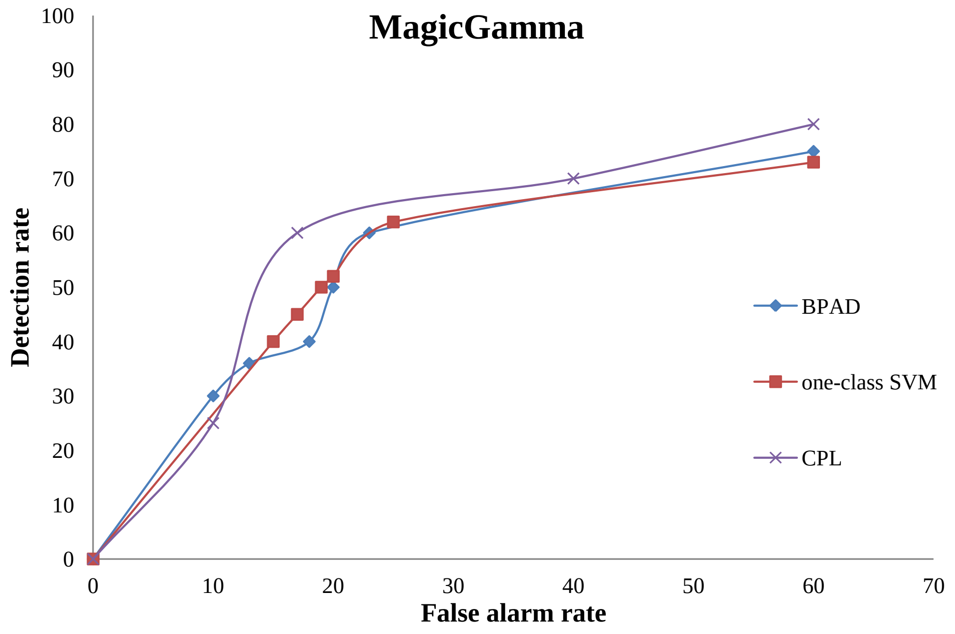


Figure 5.7: The ROC curves of the BPAD, SVM and CPL methods applied to the noisy MagicGamma dataset

Each surface in the one-class SVM is a hyperplane consisting of f points and each point is a set of f features (f is the dimension of the input dataset). For determining the self area (normal insatnces area), the one-class SVM needs to keep a number of instances (support vectors) in memory, with each instance represented by f features. In the CPL method, each cluster is estimated by a GMM, where each GMM itself is a set of GMMs. Each Gaussian consists of a mean and a variance vector, each of which has f features. As such each Gaussian needs to keep $2f$ features in memory.

In the next experiment, the memory complexity of CPL is compared with that of the one-class SVM. Table 5.2 shows the results on Shuttle, Waves,

DataSet/Algorithm	one-class SVM	CPL
Waves	$1431 \times 40 = 57240$	$[(2 \times 6 \times 4) + (2 \times 3 \times 7)] \times 40 = 90 \times 40 = 3600$
Shuttle	$17387 \times 9 = 156483$	$[(2 \times 8 \times 6) + (2 \times 17 \times 3)] \times 9 = 198 \times 9 = 1782$
MagicGamma	$5249 \times 10 = 52490$	$[(2 \times 9 \times 6) + (2 \times 11 \times 3)] \times 10 = 174 \times 10 = 1740$

Table 5.2: Memory requirement comparison between CPL and one-class SVM

and MagicGamma datasets.

For one-class SVM on Waves, the number of instances which should be kept in memory is 1431, and as the Waves dataset has 40 features, the total memory usage for one-class SVM is 1431×40. For CPL, there are two sets of GMMs, one of which is for training and the other for testing. For the training set, the clustering algorithm extracts six clusters on each of which four Gaussian distributions are employed to build the GMM. For the testing set, there are three clusters, and seven Gaussian distributions are used to estimate a GMM in each cluster. As such, the used memory for CPL on the Waves dataset is 90×40. CPL memory usage is 15 times lesser than the one-class SVM on the Waves dataset.

On the Shuttle dataset, the difference in memory usage is more significant: 17389×9 for the one-class SVM and 198×9 for CPL. For MagicGamma, the memory usage for the one-class SVM is almost 30 times higher than that of CPL.

In the previous experiment, we illustrated that CPL and the one-class SVM are almost the same in terms of false alarm and detection rate, however, in terms of memory complexity; the two algorithms are not comparable. This illustrates that the CPL performs better than the one-class SVM, BPAD and LOF methods.

5.4 Conclusion

In this chapter, a noise-resistant collective labeling approach was presented to label new incoming collectively. The GMMs were used to summarize data in arbitrary shape clusters. The proposed CPL method introduce a new distance metric to measure the distance of new groups of instances with the current clusters. The new instances are attached to their closest cluster.

A primary advantage of a collective labeling method is that it is noise-resistant. The proposed collective labeling method was compared with an individual labeling approaches in terms of detection DR, FA and memory complexity. The experimental results demonstrate that the CPL approach outperforms the BPAD, one-class SVM and LOF methods (based on individual labeling) in terms of detection and false alarm rate on noisy dataset. The detection rate was 5% higher and the false alarm rate was 10% lower over different datasets.

Since the number of support vectors generated by the one-class SVM is $10 - 15$ times greater than the number of Gaussian mixtures, needed for the comparisons, the detection time is increased accordingly.

Chapter 6

Incremental Cluster Updating Using GMM

In an online environment, the nature of incoming data changes over time. In this context, there is a need for algorithms that are able to update clusters in response to new trends in data behavior.

A simplistic way to update clusters based on incoming data is to add the new data to the current data, and then apply the clustering algorithm on the entire data. This approach is impractical in many online applications. As an alternative, incremental clustering algorithms are proposed in which clustering algorithms keep a history of data, and adding new data to the current cluster in an incremental way.

Incremental clustering algorithms need specific characteristics. First, the incoming instance has to be processed quickly to find the closest cluster. Second, the clusters have to be updated quickly to be able to adapt to the changes in the data. Third, the clusters have to be compact and well-separated over time, and not grow too fast by adding every incoming instance. Finally, the incrementally-generated clusters need to be as close as possible to the clusters generated using the simplistic approach mentioned above.

Incremental clustering algorithms such as STREAM [O'Callaghan et al., 2002] and Clustream [Aggarwal et al., 2003] update clusters efficiently in terms of memory use and time complexity. In these algorithms, finding the closest cluster to a new incoming instance is based on finding the distance of the new instance to the clusters centers. If the distance to a center is less than a threshold then, the new instance is assigned to that cluster.

Another design trend for incremental clustering estimates the entire dataset using a single GMM, in which each GMM component is a representative of a cluster. Each new instance is fed to the GMM, and used to update the GMM. However, use of a single GMM for the entire set of data has its problems. The model is complex, and the updating step involves accessing the all instances in the dataset. To overcome this problem, in this chapter, the advantages of both trends in incremental clustering algorithms are exploited. In the proposed approach, instead of finding a single GMM for the entire dataset, using the proposed approach in Chapter 4 each cluster is represented by a GMM. According to the result on Chapter 4, the GMM generates the original data with good accuracy.

Moreover, the GMM formula finds the closest cluster to the new instance with a simple calculation as explained in Chapter 5. From an updating perspective; the CPL approach instead of updating clusters based on each new incoming instance, the instances are collected, and a group of instances is used to update the GMMs. In this way, the updating phase is less sensitive to noise. Moreover, since updating is done offline, large quantities can be accommodated, even for online applications.

In the following, we explain the algorithm in more detail. The updating algorithm presented in this chapter employs the SGMM and CPL as two main components to update clusters.

6.1 Cluster Updating Based on GMM

In this section, the proposed method CUGMM (Cluster Updating based on GMM) is presented in detail.

There are three main steps as shown in Figure 6.1. In the first step, a clustering algorithm is applied to the input data to partition the dataset into clusters as shown on the left in Figure 6.1. In the second step, SGMM algorithm presented in Chapter 4 is employed to estimate the distribution and shape of each cluster with a GMM. The same procedure is then applied to the new clusters, and each cluster is represented by a GMM as shown on the right in Figure 6.1.

In the updating step, the collective labeling approach (CPL) presented in Chapter 5 is used. In this step, the newly generated clusters are compared with the existing ones. If the new GMM is close enough to one of the existing cluster GMMs, they are merged, creating an updated cluster. Otherwise, it is either added as a new cluster or it is deleted

A GMM representation summarizes a cluster into a set of means and covariances. That is to say, instead of keeping the entire instances of each cluster in memory, the GMM is used to estimates the statistical features of the cluster which requires less memory. As such, there are three advantages of using GMM to represent a cluster. First, the GMM can be used to regenerate the original cluster with an acceptable approximation. Second, clusters are represented in a way that new instances are assigned to the right cluster with acceptably low processing time. Third, the updating phase is fast because finding the closest cluster to a new one involves a simple distribution distance measurement.

In comparison to existing online clustering algorithms that update clusters based on each new instance, the proposed approach updates clusters in a

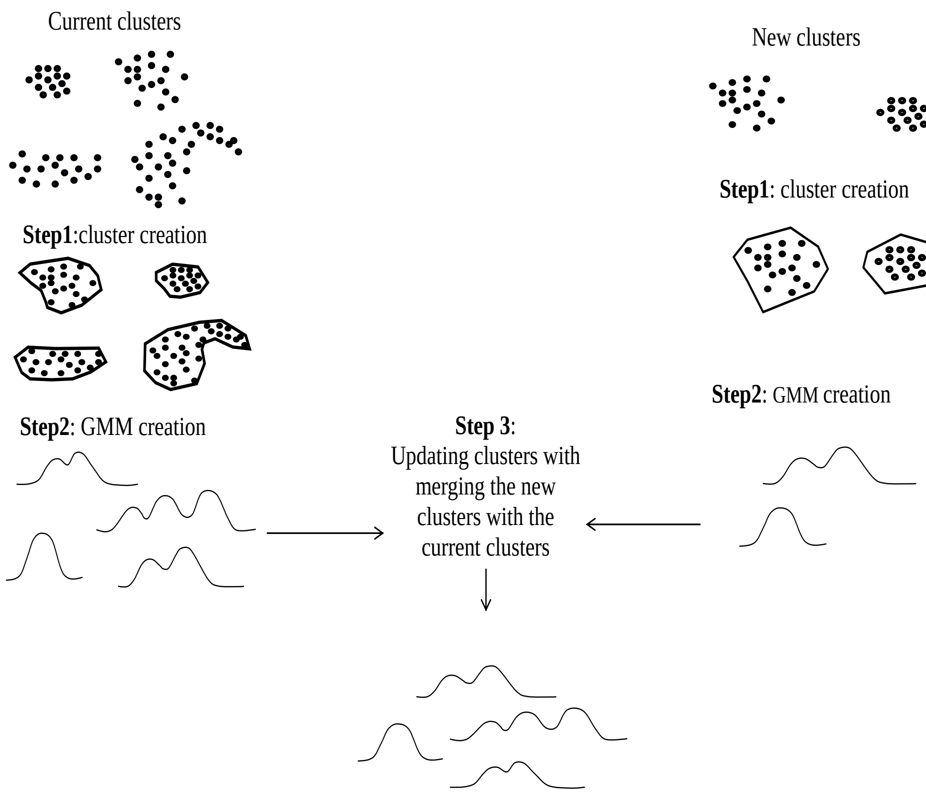


Figure 6.1: General structure of incremental cluster updating

batch using the CPL method. The updating procedure consists of two main steps: finding the two closest clusters, and then merging these close clusters.

6.1.1 Finding Two Closest GMMs

To update clusters, the closest clusters are first identified. The proposed distance measure introduced in Chapter 5 is employed.

$$D(G_b|G_a) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_i^b kl(\mathcal{N}_i^b|\mathcal{N}_j^a)$$
$$D(G_a|G_b) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_i^b kl(\mathcal{N}_i^b|\mathcal{N}_j^a)$$
$$D_{cluster} = (D(G_a|G_b) + D(G_b|G_a))/2 \tag{6.1}$$

6.1.2 Merging the Two GMMs

After finding the two closest GMMs, the next step is to merge them. Assume that the two left most GMMs in Figure 6.2 are the two close GMMs. To combine the two GMMs, the two close components in these GMMs are found and then combined. Based on the Kullback-Leibler measure, the first component in GMM_2 is close to the second component in GMM_1, and the second component of GMM_2 is close to the third component in GMM_1. Therefore, in the updating phase, the second and third components of GMM_1 are updated into a new GMM which becomes the updated version of GMM_1.

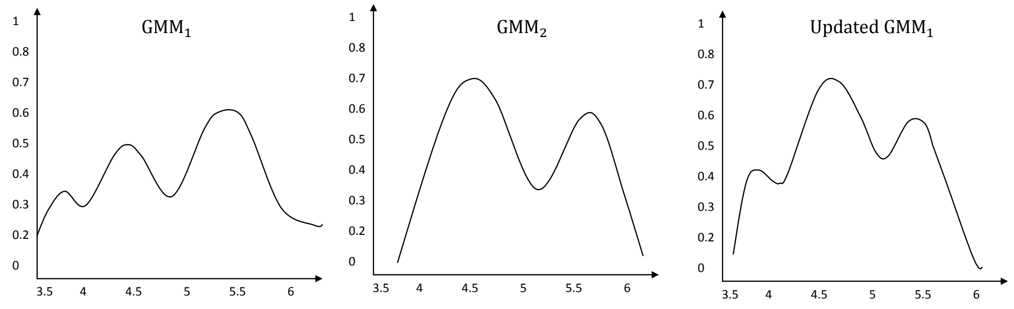


Figure 6.2: Merging two GMMs

In the proposed approach, new data instances $\{x_1, \cdots, x_m\}$ are clustered to clusters $\{C_1, \cdots, C_k\}$. The SGMM algorithm finds all equivalent GMMs; $\{G_1, \cdots, G_k\}$, for clusters $\{C_1, \cdots, C_k\}$. Each GMM consists of a set of components $G_i = \{g_i^1, \cdots, g_i^s\}$ where s is the number of GMM components. Each GMM component is represented by $g_i^j = \{\mu_i^j, \sigma_i^j, w_i^j\}$. Considering that G_a and G_b are two close GMMs. In G_a the i_{th} component that is g_a^i is close to j_{th} component of G_b that is g_b^j. Therefore, the new GMM component is created based on merging $g_a^i = \{\mu_a^i, \sigma_a^i, w_a^i\}$ and $g_b^j = \{\mu_b^j, \sigma_b^j, w_b^j\}$. To update the current component, three different parameters have to be updated.

The mean of the normal distribution μ_{new} is updated based the μ_a^i and μ_b^j.

$$\mu_{new} = \frac{w_a^i \mu_a^i + w_b^j \mu_b^j}{w_a^i + w_b^j} \tag{6.2}$$

The covariance is updated based on the covariance, mean and weights of the two components.

$$\sigma_{new} = \frac{w_a^i \sigma_a^i + w_b^j \sigma_b^j}{w_a^i + w_b^j} + \frac{w_a^i \mu_a i {\mu_a^i}^T + w_b^j \mu_b j {\mu_b^j}^T}{w_a^i + w_b^j} + \mu_{new} {\mu_{new}}^T \tag{6.3}$$

The weight is updated based on the current the number of instances of the two normal distributions that are merged with s and l the total numbers of components of G_a and G_b.

$$w_{new} = \frac{w_a^i + w_b^j}{\sum_{i=1}^{s} w_a^i + \sum_{j=1}^{l} w_b^j} \tag{6.4}$$

The updating is based on updating method for Gaussian Mixtures that is introduced in [Song and Wang, 2005].

6.2 Discussion on the efficiency of the Proposed Method

Summarization is a primary feature of the proposed method. Each Gaussian represents some instances (a part of the cluster) with a mean and variance that indicates the scattering for the cluster. The summarization feature of the proposed method, enables finding the proper clusters for new instances with only few calculations.

In a noisy environment, there is a high chance of encountering noisy instances during the updating phase. If the updating process is triggered whenever a new instance is introduced to the model, noisy instances would be likely to update model. Noisy instances may then bias or change the attributes of the clusters. This may culminate in inaccurate clusters that do not successfully model the real behavior of the normal data.

Summarizing the clusters with a set of core points and estimating the characteristics of the clusters through GMMs benefits the proposed method in a noisy environment. The proposed method does not update the existing model when a single new instance is presented to the model. Instead, the incoming instances are collected and clustered into some clusters. The extracted clusters are used to represent the new instances.

In the proposed method, after clustering new instances, the outliers and any clusters that have a number of the instances less than a given threshold are removed from the dataset. This prevents clusters from being changed by noisy input data. In the experimental results section, the performance of the proposed method in the presence of noise is characterized.

As mentioned in section 6.1.2, the weight of the final combined component is calculated based on Equation 6.4 in which the component that has a bigger weight, will have a larger share in the updated component. In some cases, the number of noisy instances in the same neighborhood, may be enough to shape a component of a GMM. In this case, the noisy component should be close to other clean components in other clusters. If the algorithm decides to combine two GMMs (one of the GMMs consists of the noisy component), there will be two approaches available for the algorithm.

First, when the noisy component is not close enough to any components in the other GMM, and the number of instances in the component is less than a particular value, the proposed method eliminates the component from its GMM. In the second approach, if the noisy component is not close to any component in the other GMM, but the number of instances is not small enough to be eliminated, the noisy component is removed from the GMM and form a new GMM (cluster). This new GMM has one component, representing the entire set of instances in the new cluster. If the newly formed cluster is not a noisy cluster, by adding new instances it may be able to find a cluster close enough to be merged, and thereby shape a new cluster. However, if the newly formed cluster is a noisy cluster, the cluster will remain isolated and not have a chance to be merged with other clusters.

6.3 Experimental Results

The proposed method was empirically evaluated, based on different criteria consisting of five metrics: Sum of Squares distance (SSQ), Dunn, DB, SD and Purity.

First, the data were divided into n parts with the first part is used to generate the initial clusters, and the rest added to the existing clusters incrementally in n rounds (iterations). The number of rounds chosen was varied, based on dataset size. That is to say, in each round a new part of data were added to the previously seen data, to examine the learning capability of the proposed method. Some UCI repository datasets were used in the experiments (KDDCup99, Shuttle, MagicGamma), as well as a two-dimensional synthetic dataset, to visualize the clustering results in each iteration.

6.4 Performance Metrics

The approach presented in this chapter updates clusters incrementally using the collective labeling approach and GMM representation. The naive approach to update cluster is to keep all cluster members and then add new coming instances to the current clusters (this approach is called CWR in this chapter). The other approach for cluster updating is CluStream that updates clusters feature set using every new coming instance. Since our approach does not keep the whole clusters and it updates clusters using GMMs, the goodness of clusters after each updating should be measured and compared to other algorithms. To validate the quality of updated clusters four clustering goodness SSQ, Purity, Dunn and DB are employed.

The Sum of Squares distance (SSQ) measures the scattering of instances based on center nodes [Aggarwal et al., 2003]. SSQ is calculated based on Equation 6.5:

$$SSQ = \sum_{j=1}^{k} \sum_{i=1}^{m} (s_i - C_j)^2 \tag{6.5}$$

where s_i is an instance in the dataset and C_j is the closest cluster to s_i.

The purity metric measures clusters homogeneity and finds the instances in each cluster which come from other clusters, based on Equation 6.6.

$$purity = \sum_{i=1}^{k} \frac{\|T_i\|}{\|C_i\|}; T_i = \{s_i \in C_j | label(s_i) = label(C_i)\} \qquad (6.6)$$

In this Equation, $label(s_i)$ is the actual label of s_i in the dataset and the label for cluster $label(C_i)$ is generated based on majority of labels of instances in clusters C_i.

Dunn [k. Dunn and Dunn, 1974] and DB [Davies and Bouldin, 1979] indexes are two metrics to measure clustering goodness. These two metrics are explained in Section 4.3.3 in Chapter 4.

6.4.1 Evaluation based On SSQ

In this section, the results of SSQ for the KDDCup99, Shuttle and MagicGamma dataset using CUGMM and Clustream [Aggarwal et al., 2003] and Clustering the Whole data in each Round (CWR) are presented.

In CWR, the first part of data is clustered. The second part of the data is added into the existing one and the clustering algorithm applied to the whole dataset. That is to say, in each round, the previous clusters are ignored and the new ones are built based on the entire dataset received by the current round. This means that there is not an incremental learning process in CWR which is why it is expected to outperform other algorithms in most cases. However, the main point is that the CWR is more accurate than other methods and can be a base for comparisons. The number of rounds was changed based on the size of the dataset, and five rounds were selected to depict the results.

Figures 6.3-6.6 show the SSQ result for the CUGMM, Clustream and

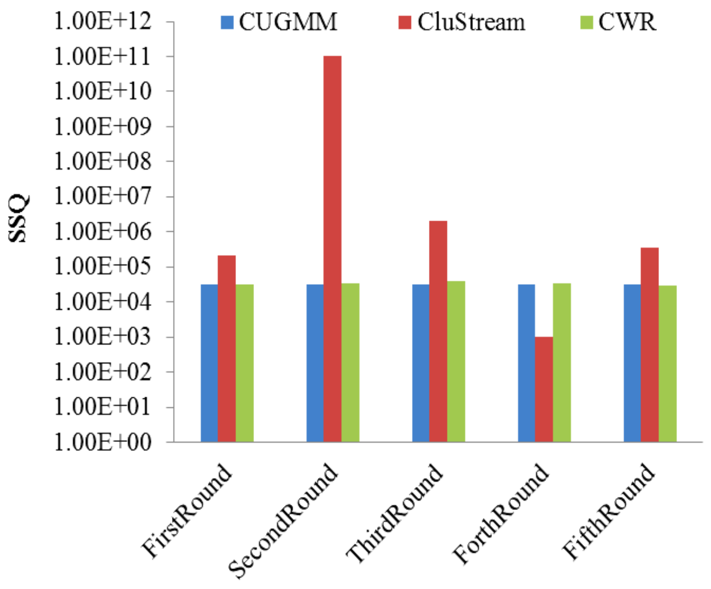


Figure 6.3: SSQ for the KDDCUP99 dataset

CWR in different rounds. The higher the value of SSQ, the lower the clustering accuracy is. Since CWR uses all the data added in each round, it should have the lowest SSQ and as a result better clustering. In Figures 6.3-6.6, the best approach is the one with lower SSQ and a value close to that of CWR.

These results characterize the accuracy of clusters by considering each cluster as a GMM. In Clustream, the clustering is based on considering core clusters and the distance of a new instance to the center of the cluster. As shown in the Figure 6.3, the CUGMM shows better performance in clustering the new incoming instances incrementally, since the results are close to that of CWR, note that CWR considers the entire set of instances in each round of clustering, while the CUGMM updates the clusters incrementally.

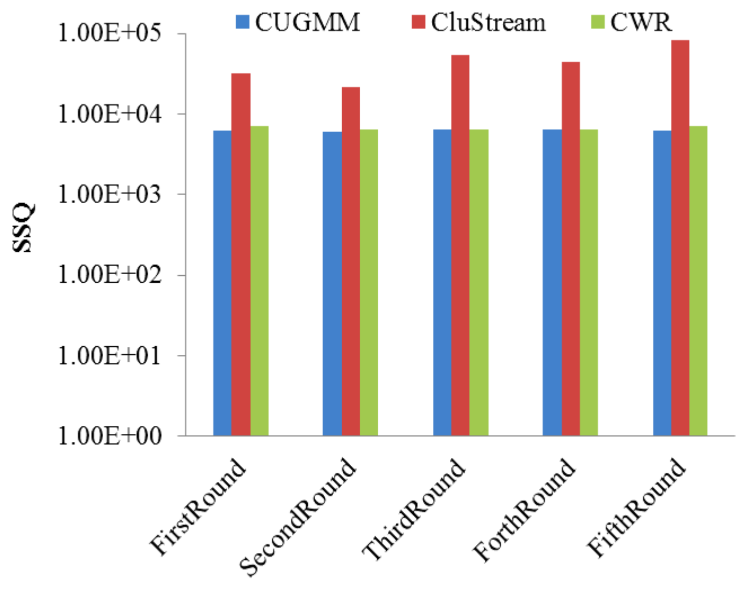


Figure 6.4: SSQ for the MagicGamma dataset

Figure 6.4 shows MagicGamma dataset results which confirm that the CUGMM works better than Clustream algorithm, and it almost equals the performance of the CWR method. For example, in the first round the CUGMM SSQ is 6221, 32210 for Clustream and 7100 for the CWR model. The lower error rate shows that the proposed approach does not lose accuracy in terms of clustering over time.

Figure 6.5 shows a similar result for the Shuttle dataset from the UCI repository. In the last round, in which the whole dataset is fed to the system, the SSQ for the CUGMM algorithm is 677 which is less than the 3275 for the Clustream. The results in Figure 6.6 for the synthetic dataset introduced in section 4.3 are almost the same as those of the previous experiments, in which the CUGMM shows better performance in comparison with CluStream and

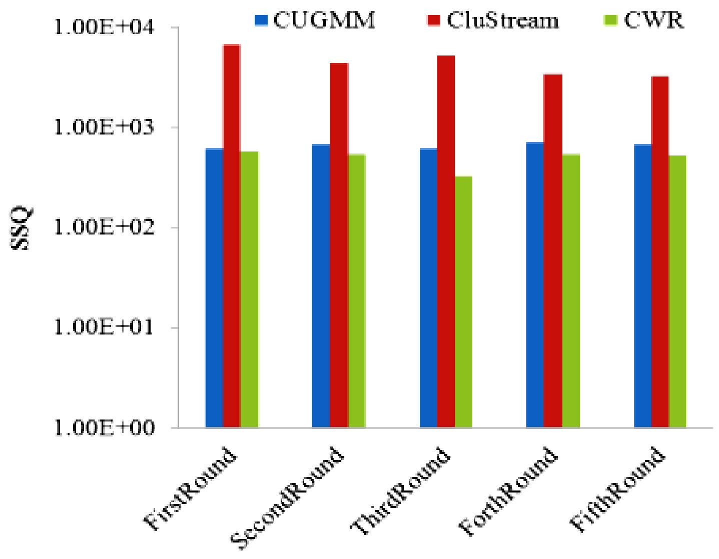


Figure 6.5: SSQ for the Shuttle dataset

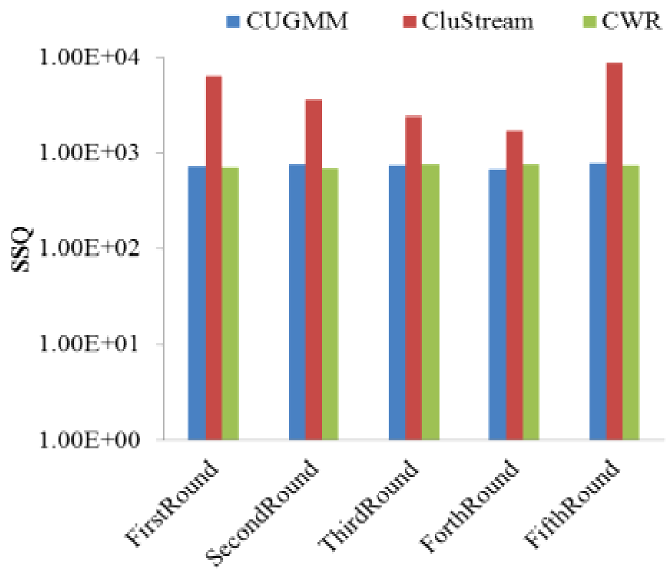
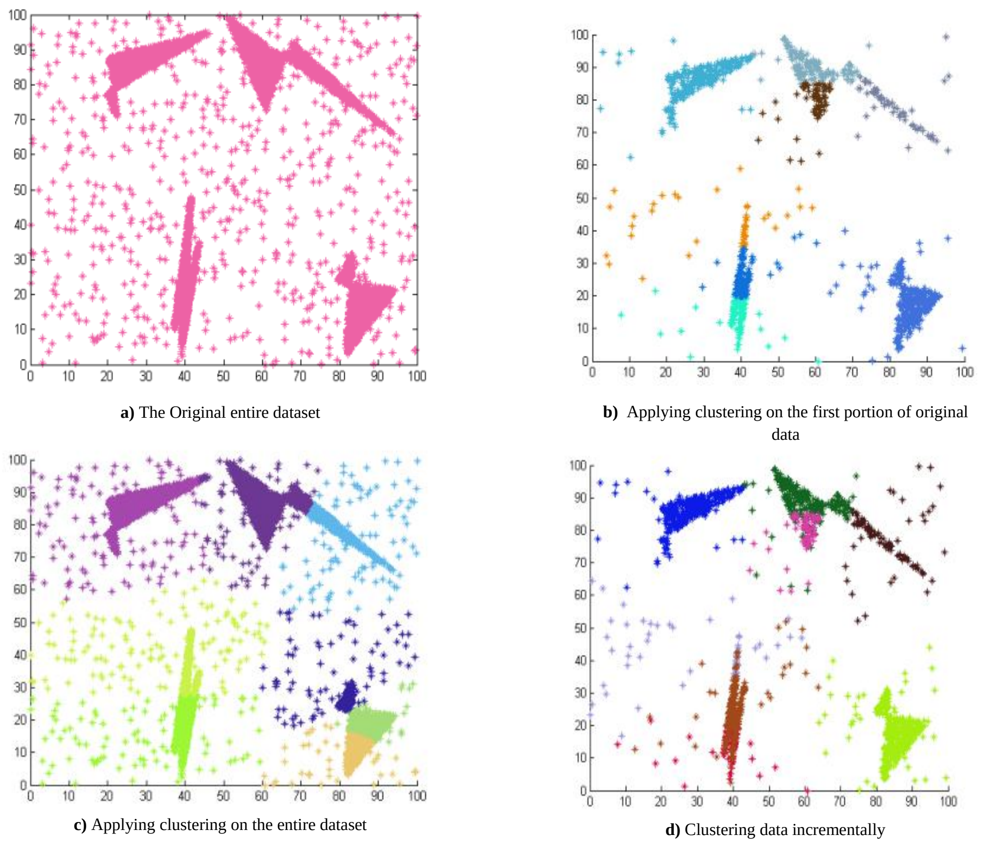


Figure 6.6: SSQ for synthetic dataset

negligibly different from CWR.

Figure 6.7 displays the results of applying the proposed algorithm on the synthetic $2D$ dataset. In the first round, 20% of the data was used to create the initial clusters. The rest of the dataset was added to the clusters over 12 rounds. Figure 6.7a shows the entire original dataset, Figure 6.7b the initial clusters generated with 20% of the data, Figure 6.7c the clusters generated using the CWR method and Figure 6.7d the final clusters using the CUGMM. To depict the noise resiliency performance of the proposed method, results are shown over 12 rounds.


a) The Original entire dataset

b) Applying clustering on the first portion of original data

c) Applying clustering on the entire dataset

d) Clustering data incrementally

Figure 6.7: Updating initial clusters over 12 rounds with CUSGMM

As depicted in Figure 6.7, most of the noisy instances were removed from the dataset using the CUGMM while there were many noisy instances in the result of CWR method. The reason is that in the CUGMM, the number of noisy instances in a particular area is not big enough to shape a new cluster.

In the next experiment, the proposed method was evaluated using the Dunn, DB, SD, and Purity indexes. For DB and SD the lowest value shows the best performance, while for Dunn and Purity the highest is the best (Table 6.1). As mentioned CWR is more accurate than other methods and can be a base for comparisons. In Table 6.1, the best method which is the closest to CWR is specified for each index.

For the KDDCup99 dataset, in each round the CUGMM algorithm had a better result than Clustream, and it is close to the results for the CWR. For the DB index, the CUGMM shows a stable performance over all rounds, while Clustream and CWR fluctuated between $[0.4 - 0.8]$ and $[0.2 - 0.7]$, respectively. For SD, most of the time the CUGMM showed better results than that of Clustream, and close to CWR results. The proposed method (CUGMM) outperformed Clustream, and generated the same clusters as CWR. The results for the DB index showed that the generated clusters were well-created in all rounds. The result for the SD confirmed the previous results.

The results for Purity showed that with the CUGMM algorithm, the purity value inside clusters (based on the label of the data) was higher than that of Clustream, and close to that of CWR.

6.5 Conclusion

The incremental updating approach represents all clusters using a GMM, keeping a summary of each cluster, it is updated quickly and more accurately.

Instead of processing each new incoming instance individually, the instances are collected and grouped into clusters. For each cluster, a GMM is defined to estimate the instances' scattering and distribution. In the updating phase, a modified Kullback-Leibler distance is used to find the closest cluster to the new clusters. After finding a pair of close clusters from current and new clusters, their corresponding GMMs are merged. A merging strategy is introduced that enables the algorithm to detect noisy instances and remove them from the rest of the process.

The proposed method was empirically compared with two other clustering algorithms, CWR and Clustream using various cluster validity indexes. The results showed that the updating approach tends to follow the original shape of clusters over time. The results for different datasets and based on various factors showed that updating clusters using GMMs is more accurate than Clustream.

The Sum of SQuared distance (SSQ) is decreased 2 to 5 times on average for different datasets, in comparison to Clustream and the basic CWR. Based on different clustering indexes for the clusters, accuracy of the Clustream and the basic model fluctuated between $[0.4, 0.8]$ and $[0.2, 0.7]$, respectively while the CUGMM method had more stable clustering in different rounds.

DataSet/Clustering index	Dunn	DB	SD	Purity
KDD	CUGMM	CUGMM	CUGMM	CUGMM
	CUGMM	CUGMM	CUGMM	CUGMM
	CUGMM	Clustream	CUGMM	CUGMM
	CUGMM	Clustream	CUGMM	CUGMM
Shuttle	Clustream	CUGMM	CUGMM	CUGMM
	CUGMM	CUGMM	CUGMM	CUGMM
	CUGMM	Clustream	CUGMM	CUGMM
	CUGMM	CUGMM	CUGMM	Clustream
MagiccGamma	CUGMM	CUGMM	CUGMM	CUGMM
	CUGMM	Clustream	CUGMM	CUGMM
	CUGMM	CUGMM	CUGMM	Clustream
	CUGMM	Clustream	CUGMM	CUGMM
Synthetic	CUGMM	CUGMM	CUGMM	CUGMM
	CUGMM	CUGMM	Clustream	CUGMM
	CUGMM	Clustream	CUGMM	CUGMM
	Clustream	CUGMM	CUGMM	CUGMM

Table 6.1: Performance of clustering based on Dunn, DB, SD, Purity indexes

Chapter 7

Two-Layer Structure

The two-layer structure for updating clusters incrementally was presented in Chapter 3 with five main components. The whole structure works if the idea and approach for each component works efficiently. Each part of the structure is introduced and examined in different chapters. The two-layer structure puts all these pieces together, and encapsulating an incremental approach for cluster updating. The goal of creating such a structure is to update clusters efficiently by removing the noisy instances, while ignoring redundant instances that are already consistent with the model.

In this chapter the two-layer structure is presented in two layers and the idea of ignoring redundant instances is discussed.

7.1 Coarse and Fine Levels in Two-Layer Structure

Figure 7.1 presents the two-layer structure first shown in Figure 3.4 with added details. Each new instance based on its similarity to the existing clusters is labeled. If the new instance shows a similarity less than a threshold to all clusters, it is sent to a rag bag to update the model in a batch mode.

Using the coarse level with simple and lightweight computation, the structure quickly labels whichever instances are not similar to any known existing patterns. The coarse level is also able to recognize instances that are quite similar to previously-seen patterns.

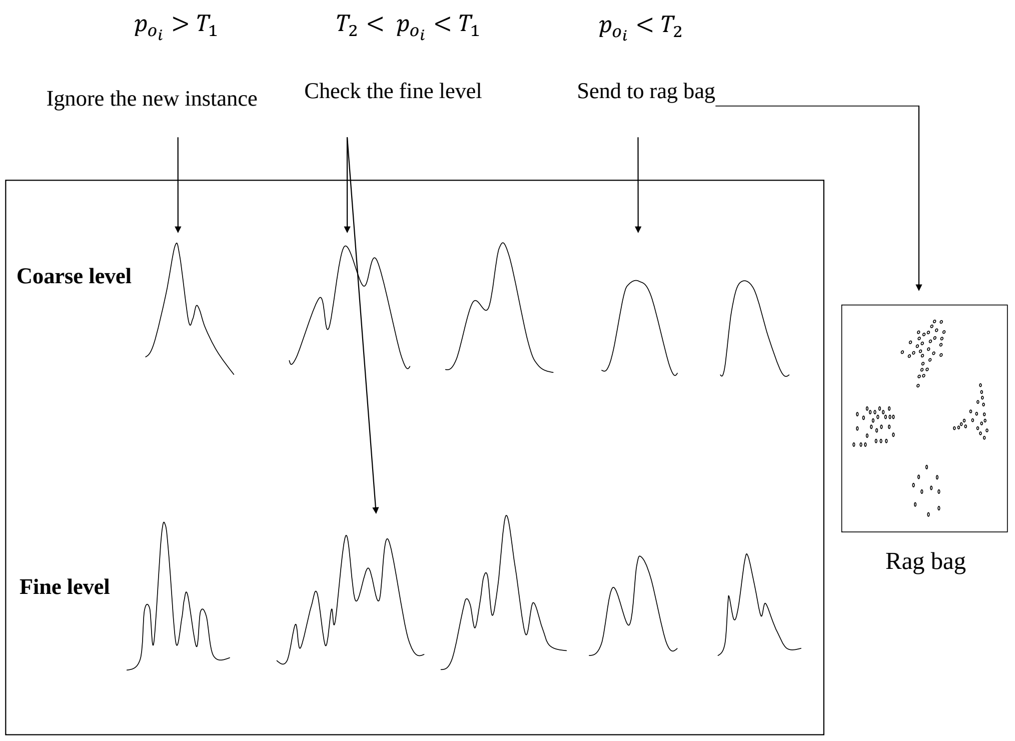


Figure 7.1: Selecting new distinct objects and put them in rag bag

The fine level is employed for those instances that are 'in the middle'-not very similar not very different. To make such a model work, some threshold is required. To do so, all possible cases are considered for new incoming instances.

All possible cases are divided between the coarse and fine levels as they arrive. In Figure 7.1, T_1, T_2 and T_3 are the thresholds and p_{o_i} is the membership value for a incoming instance, calculated based on GMM formulas in Chapter 4.

- Possible incoming instances and outcomes for the coarse level:
    - New instance belongs to a cluster with membership value more than a specified threshold (T_1 or upper bound threshold).
        - In this case, this instance belongs to a specific cluster with high probability, and there is no need to investigate other clusters. Also the main point here is that it is not necessary to use this instance for further updating, because cluster distribution would not be changed with this instance. The new instance is labeled with a coarse level, without invoking the fine level, and so with less computational cost.
    - If the membership value for the new instance is less than a specified threshold (T_2 or lower bound threshold)
        - The instance is considered as a new and unseen instance, and it is sent to the rag bag directly. In this case, the algorithm assumes that the instances' similarity to the region of known and existing patterns is low and as such, it is either noise or a new anomalous instance. The detection process is finished at the coarse level, with no need to go to the fine level.
    - If the membership value for an incoming instance is between T_1 and T_2.
        - The instance is sent to fine level for more investigations.
- Possible incoming instances and outcomes for the fine level:

- In the fine level, the new instance is introduced to the clusters, and based on the generated membership values a decision is made. In this level, there is a threshold (T_3), based on which an instance is categorized as an unseen pattern or a known instance. If the generated membership value is higher than T_3, the instance belongs to one of the clusters in the fine level, otherwise it is considered as new and is sent to the rag bag.

The overall algorithm for two-layer cluster-based structure is presented in Algorithm 7.1.

7.2 Removing Redundant Instances

A new instance is deemed redundant if it is quite similar to the existing model. In the two-layer structure, the whole structure is based on clusters, and each cluster is represented by a GMM. In such a representation, a redundant instance is an instance that has a high membership value for one of the GMMs. That is to say, this instance is close to the center of one of the GMMs. This instance has been included in the distribution of data in the first stage, and there is no point in updating the clusters with this instance.This assumption is simple, and speeds up the updating procedure.

However, there are some concerns about it. By ignoring such instances, some information may be missed in the updating stage. These instances are mainly similar to the initial instances that were used to generate the clusters. Therefore, these instances are already represented in the model, with a proper distribution. Updating each cluster with redundant instances sets a new mean and covariance for each component, which are not very different from the original mean and covariance.

Algorithm 7.1: Pseudo-code of the two-layer architecture

```
Data: cluster GMMs, instance O
Result: maximum probability of attachment of instance O to clusters
$p_O$=probability of attachment instance O to $GMM_{C_1}$;
for $i = 2$ to number of clusters do
    if $p_O > T_1$ then
        Ignore instance O;
    else
        $p_O = max\{probality\ of\ attachment\ to\ GMM_{C_i}, p_O\}$;
    end
end
if $p_O < T_2$ then
    Send it to rag bag;
end
if $T_2 < p_O < T_1$ then
    $p_O$=probability of attachment instance O to $GMM_{C_1}$ in fine level;
    for $i = 2$: number of clusters do
        if  $p_O > T_3$ then
            Ignore the instance O;
        else
            $p_O =$
            $\max\{probability\,of\,attachment\,to\,GMM_{C_i}\,in\,fine\,level, P_O\}$;

        end
    end
else

end
if $p_O < T_3$ then
    Send it to rag bag;
end
```

Another concern is that other instances may be considered as new and anomaly that were never merged with current clusters. This hypothesis can be true when instances are considered separately in updating. In the two-layer approach, new instances are first clustered, and then merged with current clusters. The new clusters are representative of new distributions which can be similar to previous ones, and can be merged with them. Based on experiments these instances are mainly near the border, and with the effect of gradually changing the cluster boundaries.

7.3 Setting Thresholds

As shown in Figure 7.1, thresholds need to be set. To set these thresholds, an approach is needed that specifies the threshold without just using simple trial and error. To set up thresholds T_1 and T_2 in the structure, the concept of the confidence interval in a normal distribution is used as shown in Figure 7.2.

A confidence interval gives an estimated range of values, which is likely to include an unknown population parameters. The estimated range is calculated from a given set of instance data.

For a population with unknown mean μ and known standard deviation σ, a confidence interval for the population mean, based on a simple random instance (SRS) of size n, is:

$$\overline{x} \pm z^* \frac{\sigma}{\sqrt{n}} \tag{7.1}$$

where z^* is the upper $(1 - C)/2$ critical value for the standard normal distribution and C is the confidence level. This interval is exact only when the population distribution is normal. For large instances from other population

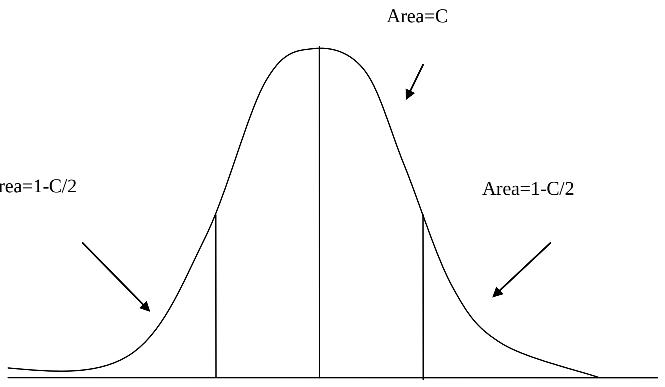


Figure 7.2: A confidence interval for normal distribution

distributions, the interval is deemed approximately correct by the Central Limit Theorem. The confidence intervals for means and covariance include the actual value of these parameters.

This idea can be used to set up confidence intervals for membership values that lead to finding the thresholds. The confidence interval covers most of distribution and can be used to identify instances that are in the confidence interval. These members are fed to the GMM formula to determine the membership values for these instances. The average membership value is calculated for these instances, and considered to be the membership value for the redundant instances:

$$M = \{O_i | \overline{x} - z^* \frac{\sigma}{\sqrt{n}} < O_i < \overline{x} + z^* \frac{\sigma}{\sqrt{n}}\} \tag{7.2}$$

M is a set of members O_i that are in the confidence interval of the normal distribution.

$$\overline{m} = \frac{\sum_{O_i \in M} \mathcal{N}(O_i)}{\|M\|} \tag{7.3}$$

$\mathcal{N}(O_i)$ is the membership value of instance O_i in normal distribution $\mathcal{N}$, $\|M\|$ is the number of members of set M and $\overline{m}$ is the average of membership values for all members of M.

Thus far, this pertains to a single Gaussian distribution. However, we wish to deal with a set of normal distributions. The thresholds of each normal distribution are combined to find a threshold for the set of normal distributions. The confidence interval for each component in the GMM is shown in Figure 7.3.

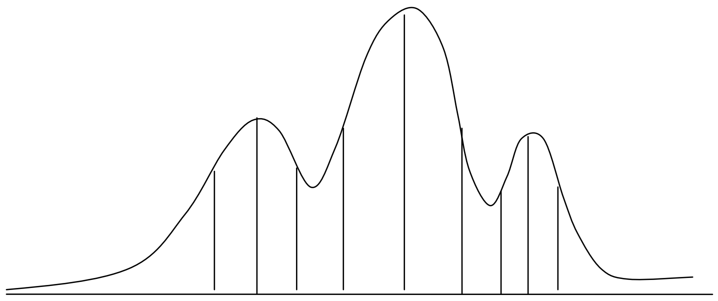


Figure 7.3: A confidence interval for Gaussian Mixtures

For a GMM with n components, there is a set of members M_i in the confidence interval of each component in the GMM:

$$L = \{M_1, \cdots, M_n\} \tag{7.4}$$

The membership value for each component $M_1, \cdots, M_n$ is calculated according to Equation 7.3.

$$\overline{L} = \{\overline{m_1}, \cdots, \overline{m_n}\} \tag{7.5}$$

Each $\overline{m_i}$ in $\overline{L}$ should be multiplied by the weight w_i of the component to determine the overall membership value for the whole Gaussian.

$$T_1 = \sum_{i=1}^{n} \overline{m_i} \times w_i \tag{7.6}$$

By finding instances in the confidence interval, their membership values can be determined, and they can be combined based on their weights. Each instance in the confidence interval should have a higher membership value. Therefore, if the membership value for an instance is greater than this value, it is considered to be a redundant instance.

There is another threshold that needs to be set: T_2 as shown in Figure 7.1. To configure this threshold, a similar approach is used by focusing on less probable instances. To find these instances, all instances outside of the confidence interval are considered and their membership value calculated:

$$M = \{O_i | \overline{x} - z^* \frac{\sigma}{\sqrt{n}} < O_i \ OR \ O_i > \overline{x} + z^* \frac{\sigma}{\sqrt{n}}\} \tag{7.7}$$

For each component i the set of instances M_i is created, and for each the average membership value $\overline{m}_i$ is calculated:

$$\overline{m}_i = \frac{\sum_{O_i \in M_i} \mathcal{N}(O_i)}{\|M_i\|} \tag{7.8}$$

Using the weight w_i of each component i , the new threshold is found:

$$T_2 = \sum_{i=1}^{n} \overline{m_i} \times w_i \tag{7.9}$$

This value is an indicator for instances that are far away from the distribution, and should be sent to the rag bag for the updating stage.

The third threshold T_3 is set in the same way as threshold T_1.

7.4 Experimental Results

Experiments described in this section, were used verify that removing redundant instances does not have an effect on the accuracy of the algorithm, and it is faster than simply updating clusters with every new instance.

Metrics for Comparisons

Accuracy was measured through assessing false alarm and detection rate in anomaly detection applications. The incremental method was applied to data for r rounds, and in each round the results were collected. The ROC curves reported, shows the average accuracy of algorithm for these r rounds. The results for each algorithm also resulted from running the algorithms 30 times.

Other experimental measurements included the time and space complexity. Space and running time were compared for labeling new instances, and also for updating the current model.

Finally, we show the visual results of the synthetic data set to depict how the noise is removed by employing two-layer structure.

Dataset Description

Table 7.1 displays the details of the datasets used in our experiments. As shown in this table, KDDCUP99, Darpa98, NSLKDD, DataSetMe, IUSTSip and INRIASip datasets are seven datasets for our experiments.

For each dataset mentioned, the training segment contains only self-instances, and the test segment includes a combination of self and non-self instances. In the following, we will describe the details of the datasets.

Dataset	# of features	# of classes	Normal class	Anomaly class	Training set size	Test set size
$KDDCUP99$	41	5	1	2, 3, 4, 5	17020	82980
$Darpa98$	5	2	1	2	175359	759334
$NSLKDD\ P$	41	2	1	2	13449	22544
$NSLKDD\ M$	41	2	1	2	13449	11850
$DataSetMe$	10	2	1	2	8274	9063
$IUSTSip$	7	2	1	2	130760	412375
$INRIASip$	9	3	1	2, 3	74483	10756

Table 7.1: Datasets Used for Experiments

An important feature of DatasetMe dataset is that it is specifically generated for anomaly detection purposes. DatasetMe uses D-ITG as the background traffic generator. Two separate computers are assigned as the D-ITG transmitter and receiver, respectively. A computer is assigned as a victim (i.e. as a destination for all kinds of attacks). The dataset includes different types of attacks through the Internet on the victim computer. Each instance of the DatasetMe is the result of analyzing the behavior of the network over $2s$. Table 7.2 shows the feature description of the DataSetMe. The features extracted from the header of the packets. The test dataset held 18 different attacks, some of which are well-known attacks from (PacketStrom) and Nmap (NMap). Other attacks were generated in the lab.

Table 7.3 shows the summary of the NSL-KDD dataset. NSL-KDD is a dataset suggested to solve the problems of the KDDCUP99 dataset. However, the new version of the KDDCUP99 dataset still suffers from the same problems, and may not be a good representative of real networks [McHugh, 2000].

IUSTSip dataset was generated in the lab, and included SIP DoS attacks. The implementation details of these datasets are discussed in [Asgharian et al., 2011] and [Asgharian et al., 2012]. In this dataset, different types of SIP flooding attacks are generated, dumped and labeled. These attacks are based on INVITE, REGISTER and RINGING messages in SIP.

INRIASip dataset is generated in INRIA France. This dataset includes INVITE flooding attacks in VoIP networks and also considers the telephony spam threat (SPIT), as well as normal traffic. Two SIP proxy servers were used for their test-bed. Anomaly DoS traffic and SPIT traffic are included in the dataset, and both were used in the experiments [Mohamed. Nassar, 2008] [Mohamed. Nassar, 2009].

Number	Name of features	Description
1	Number of packets	Number of input and output packets in the recent $2s$
2	$\%TCP$	Rate of the TCP packets in the recent $2s$
3	$\%UDP$	Rate of the UDP packets in the recent $2s$
4	$\%IGMP$	Rate of the IGMP packets in the recent $2s$
5	$\%Other$	Rate of the packets that are not TCP, UDP and IGMP in the recent $2s$
6	AvgLenPackets	Average length of packets in the recent $2s$
7	RndSrcPort	Number of different source ports that are used during the recent $2s$
8	RndDesPort	Number of different destination ports that are used during the recent $2s$
9	$\%AckPackets$	Rate of the Ack Packets in the recent $2s$
10	$\%SynPackets$	Rate of the Syn Packets in the recent $2s$

Table 7.2: The summary of the DatasetMe dataset

DataSet Name	Type	# of features	Size
Train +	Train	41	125973
Train20%+	Train	41	25192
Test+	Test	41	22544
Test-	Test	41	11825

Table 7.3: Summary of the NSL-KDD dataset

The best method was chosen in each category of anomaly detection methods, and these algorithms were used in the experiments. The list of these methods is presented in Table 7.4.

Category	Method
Cluster-based methods	CBLOF
Proximity-based methods	LOF
Classification	One-class SVM
Artificial immune system	RVNS, positive selection, SPAI
Statistical methods	Single GMM

Table 7.4: Anomaly detection methods

CBLOF [He et al., 2003] method is a cluster-based anomaly detection method. This method first clusters the data, it creates a metric based on the size of the cluster and distance of an instance to the center of its cluster. Based on the value and a threshold the instances are labeled as normal or abnormal. LOF [Breunig et al., 2000] is another method to find the anomaly score for an instance. In this approach, the local densities of an instance is measured using the local density of the instances in its neighborhood. Real Value Negative Selection (RVNS) [Ji and Dasgupta, 2004] is one of the most famous algorithms in Artificial Immune systems. In this algorithm, a set of detectors is generated that does not have any overlap with the normal instances, which are used in the detection step. Positive selection algorithm considers all normal instances as representatives of normal patterns. By considering a circle around each instance it creates a boundary for each normal pattern [Seiden and Celada, 1992] [Sim and Lee, 2003]. SPAI presented by Mohammadi et al. combines the positive selection with Parazen window to create a new probabilistic approach for anomaly detection [Mohammadi et al., 2014]. Among one-class classification methods, one-class SVM is among the best and most accurate one [Yuting et al., 2013]. The last algorithm considered for comparisons is GMM-based approach which is in the category of statistical approaches Haji [Hajji, 2005]. This considers the whole normal dataset as a single GMM. Every incoming instance is compared to this GMM, and if it deviates from this distribution, it is considered abnormal. All mentioned methods were discussed in detail in Chapter 2.

In the following two sections, the results are presented for experiments of using the two-layer structure for anomaly detection. First, the accuracies of all algorithms were compared based on false alarm and detection rate. The results using different datasets confirm that the two-layer structure is better than the other algorithms, in most cases. However, the main capability of the two-layer structure is to enable the whole model to be updated quickly requiring minimum space. This structure was as accurate as the other models, while being updated over time in batch mode. Removing the burden of redundant instances in the updating phase improved the updating and labeling phase.

False Alarm and Detection Rate

In this section, the results of comparison of the accuracy of the presented approach with all other algorithms are presented. Tests involved the methods listed in Table 7.4 on all datasets presented in the previous section.

As with the previous comparisons for anomaly detection, ROC curves are depicted based on the false alarm and detection rate. Higher detection rate with low false alarm rate shows better performance.

The first dataset used was KDDCUP99. As shown in Figure 7.4 the performance of the two-layer approach and one-class SVM were close, and superior to the other algorithms. The performance of positive selection and SPAI were almost the same, and both were lower than that of the two-layer approach. RVNS has less attractive results, since representing the whole structure of anomalies with a set of artificially generated instances cannot represent the whole space of anomalous behavior.

The GMM algorithm had less accuracy than other algorithms in terms of false alarm and detection rate. The main problem with the GMM approach is that it considers only a single GMM of all the data, which is an overly general representation of the data and may generate a low detection rate.

CLBLOF and LOF had poor results, in comparison to the other algorithms. The CBLOF algorithm finds outliers by clustering them and calculating a factor based on the distance of an instance to the center and to other members of the cluster. The first problem is to consider the center of the cluster as a representative to measuring the distance. Those factors that use distance to the center of the cluster are among the poor approaches, since the center of the cluster cannot represent the clusters' distribution. LOF assesses the outlier-ness factor of an instance, using the local density of each instance. This factor is not applicable to all problems, since it just removes noise and not a collection of outliers.

An important aspect of these experiments is that in using the two-layer structure, clusters are updated over the time while ignoring redundant instances in batch mode. There may be doubt about the potential accuracy of the model in terms of false alarm and detection rate. However, the results on the KDDCUP99 and other datasets showed that accuracy does not deteriorate with incrementally updating clusters while ignoring redundant instances and employing the batch updating approach. On the other hand, incremental and batch updating saved time, and the proposed approach was faster than the others, while still accurate. The following results were from running the algorithm over 30 independent runs.

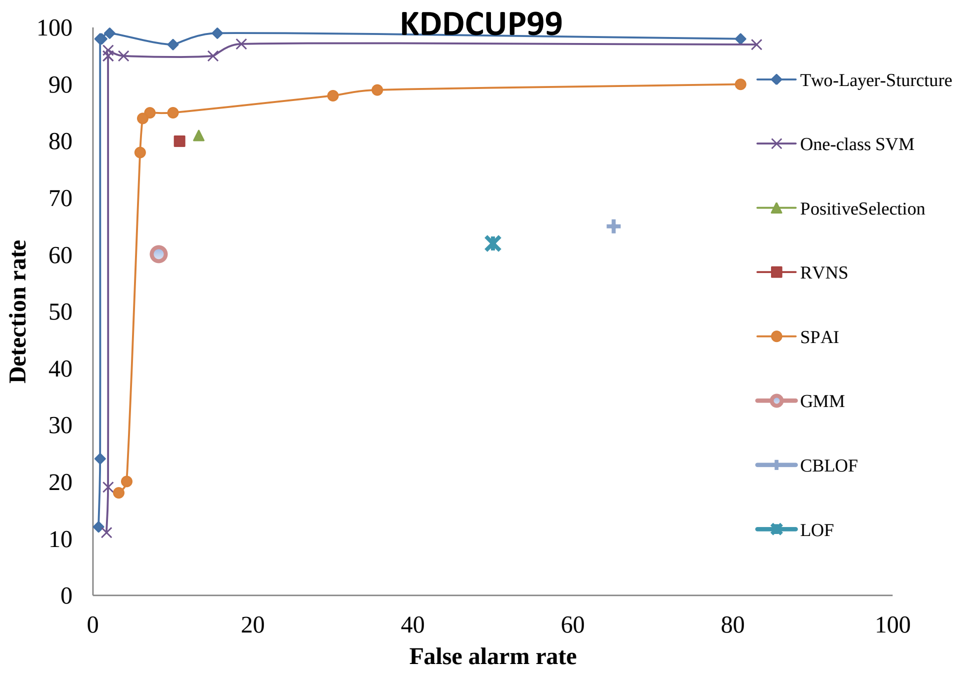


Figure 7.4: ROC curve for KDDCUP99 dataset for different algorithms

In the next experiment, the proposed method was evaluated on the Darpa dataset, which includes information about sessions. Similar to the DataSetMe, the Darpa dataset was collected from a real network. All the BSM files in the dataset were used to assess the proposed method in a practical scenario, with large amount of data. The training dataset comprised the first two weeks, with the four remaining weeks the test dataset. As with the other experiments, the normal instances of the first two weeks were considered for training.

Figure 7.5 shows the experimental results for the Darpa dataset. The two-layer structure and one-class SVM performed almost the same, and both were better than the other algorithms. The proposed method not only outperformed the other methods, it also reduced the effect of noise and considerably decreased detection and updating time. The detail of time and space complexity will be discussed in Section 7.4.

In Figure 7.6, the results of the comparison using the NSL-KDD+ dataset are presented. The accuracy of the two-layer structure and one-class SVM were better than the other methods. However, the false alarm rate of the two-layer structure was better than that of one-class SVM. While the one-class SVM creates a more accurate boundary, it overfits to the data, and that is why it generated more false alarms. The two-layer structure employs the strength of clustering and a distribution representation of data to create a more accurate and less restricted model. The GMMs generated for each cluster were a good representative of each cluster, with straightforward updating and labeling of new incoming instances. SPAI and positive selection were the second-best performers.

SPAI and positive selection performance were less than the two-layer structure, since their approach highly depends on the number of detectors

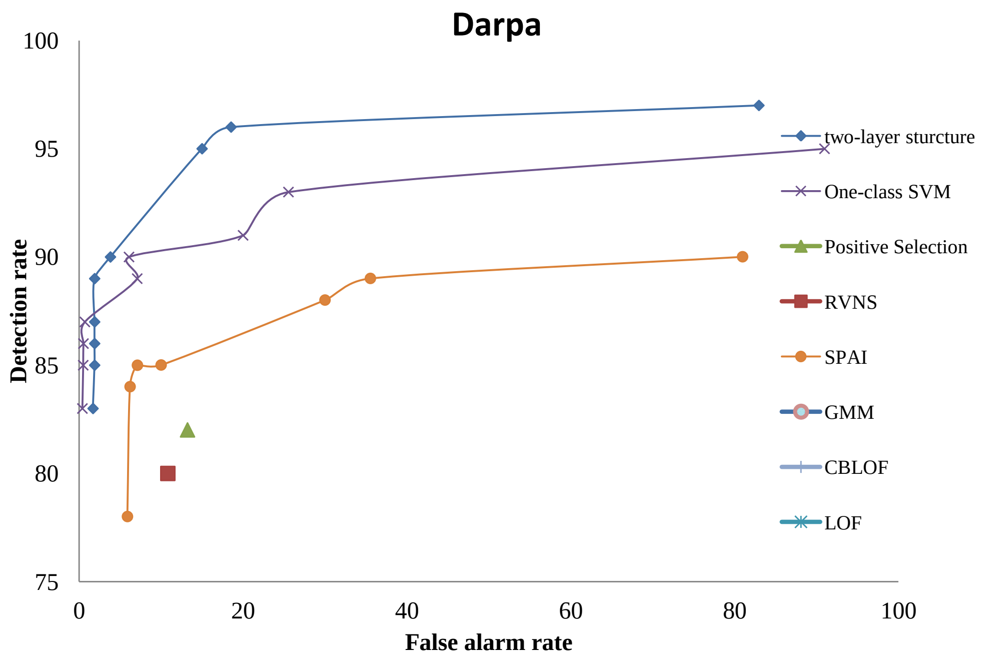


Figure 7.5: ROC curve for Darpa dataset for different algorithms

and the radius, which is difficult to determine. Restricting the results to the detectors that are not representative of the entirety of normal behavior decreases the accuracy of the associated predictions. The GMM approach did not have an interesting result. LOF and CBLOF did not exhibit good accuracy; since they are dependent on the clustering and density of the dataset in different regions of data space.

The fourth dataset used for our experiments is NSL-KDD- dataset. Again, the two-layer structure and the one-class SVM performed better than the others (Figure 7.7). This shows that even in applying clustering, GMM

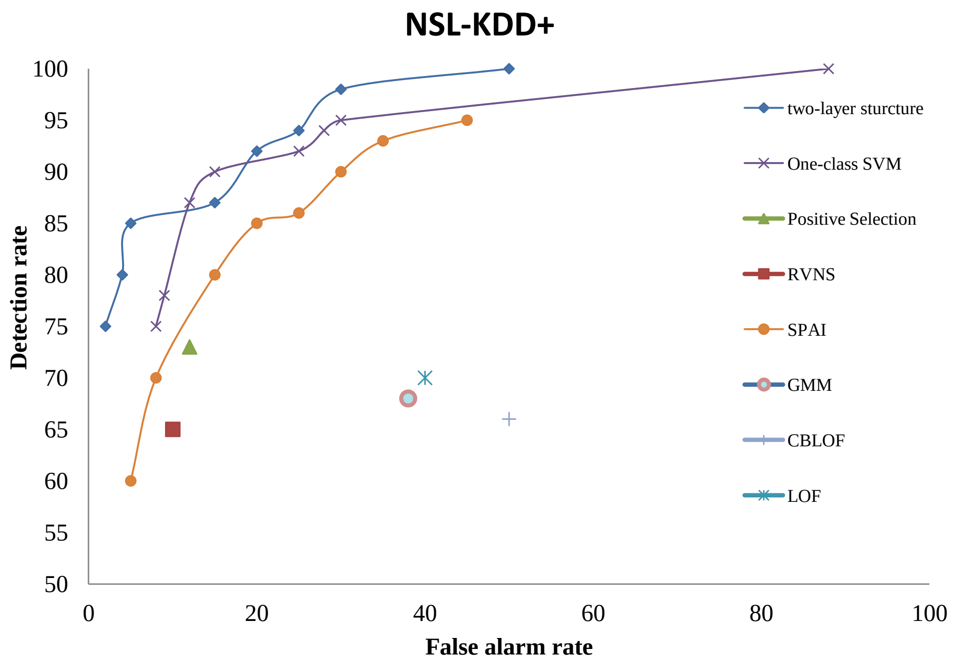


Figure 7.6: ROC curve for NSL-KDD+ dataset for different algorithms

representation and batch updating, there was not a decrease in the performance of the proposed method.

One of the real network datasets available is DatasetMe, which was generated in the lab for experimental purposes. This dataset represents two main properties of network data: the redundancy of instances in a short period, and the appearance of similar applications within a short time. Figure 7.8 shows the result of applying different methods on DatasetMe. The result shows that the two-layer structure was best among the other methods. It confirms that on such a real dataset, the clustering and GMM representation were an excellent representative for normal data. The second best one was one-class SVM, and its performance was almost the same as that of the

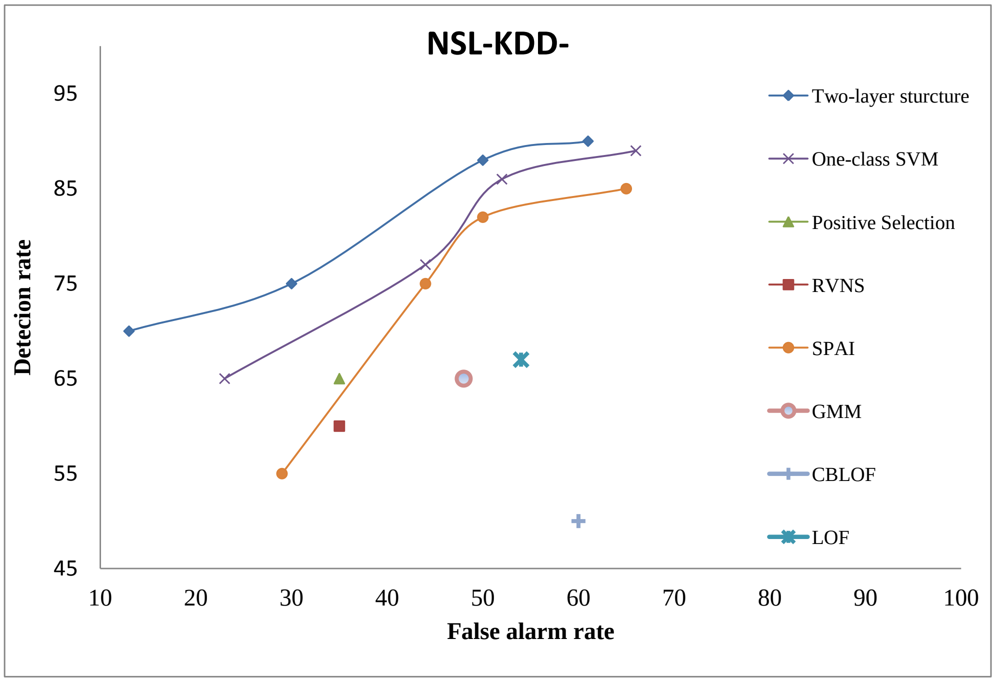


Figure 7.7: ROC curve for NSL-KDD- dataset for differet algorithms

two-layer with the difference being false alarm rate. The high accuracy of one-class SVM is based on a large number of support vectors that it generates. However, it creates an over-fitted model with high detection rate but also high false alarm rate.

On the other hand, preserving and updating a large number of support vectors are difficult tasks, especially in incremental and online applications. The performance of the other methods was less than the two-layer structure and one-class SVM while suffering from the complexity of applying the models to online environments. The space and time complexity of the proposed approach will be discussed in the next section, showing that the two-layer structure outperformed the other methods.

There are many repeated instances in this dataset. This redundancy is a normal behavior of computer networks as explained before. The result verified that the two-layer approach was successful in reducing the repeated instances. It also confirmed that reduction does not have any negative effect on the false alarm and detection rates.

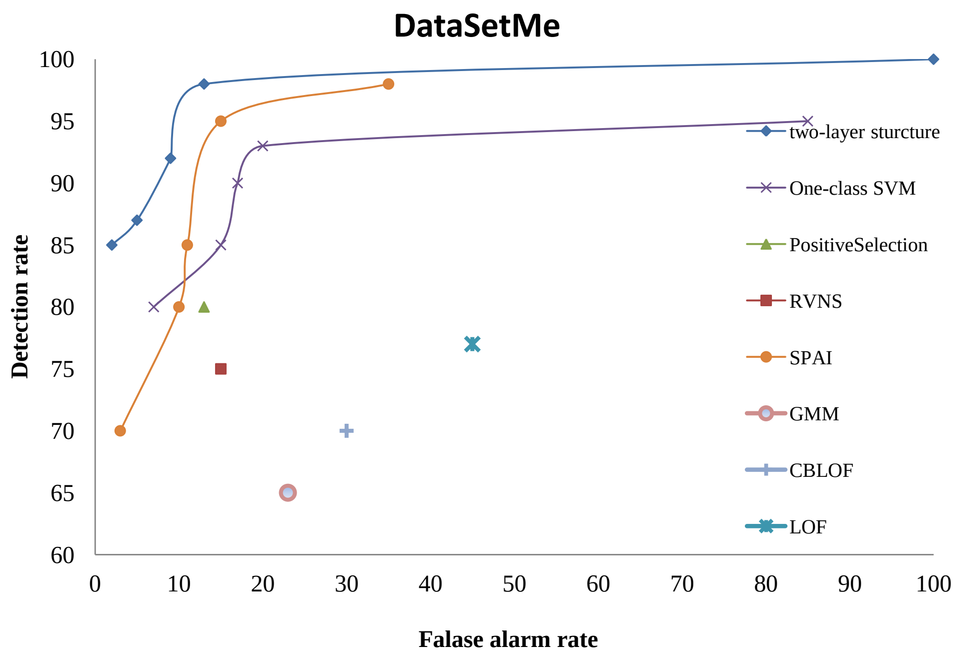


Figure 7.8: ROC curve for DatasetMe dataset for different algorithms

Another real-world dataset generated was the IUSTSip dataset, with results shown in Figure 7.9. Similar to the other experiments, the performance of the proposed method on this dataset was better than the other methods. The two-layer structure had 94% detection rate and 4% false alarm rate while the one-class SVM have 90% detection rate and 10% false alarm rate. The false alarm rate of the approach was better than one-class SVM, since the proposed method presents better generalization in comparison to the other methods. SPAI was the third best method. The other methods were below average, with high false alarm and less detection rate.

As will be discussed in section 7.4, the computational cost of the proposed method is lower than SPAI and one-class SVM algorithms. The results of these experiments confirmed that in real situations, with all types of the protocol running on the networks, the proposed method worked better than the other algorithms.

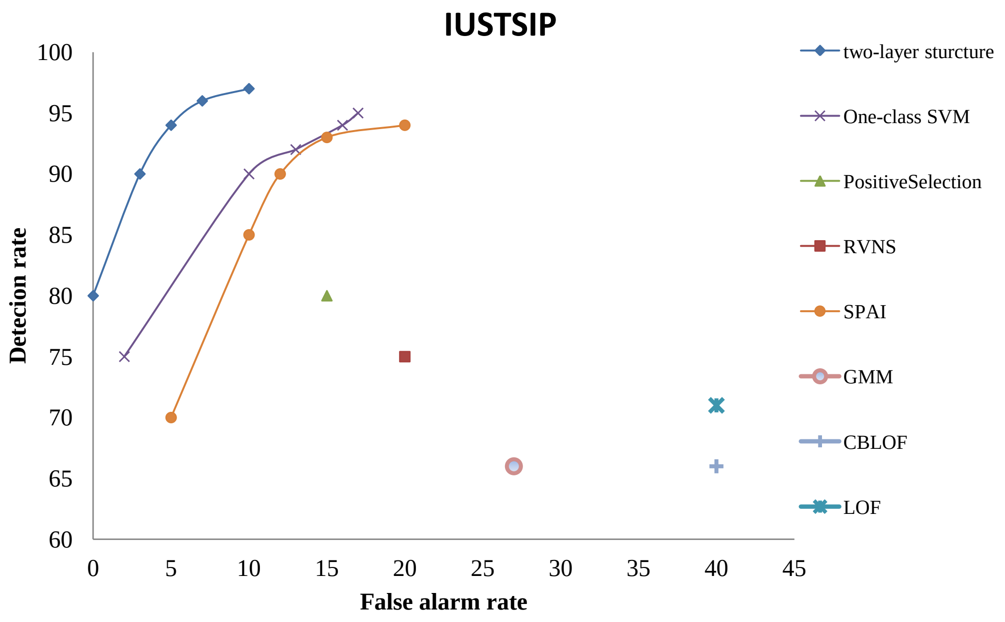


Figure 7.9: ROC curve for IUTSIP dataset for different algorithms

INRIASip dataset was also used to evaluate the proposed method on another real SIP-based dataset. The result on this dataset (Figure 7.10) confirmed that the two-layer structure performed better than the other algorithms in terms of false alarm and detection rate.

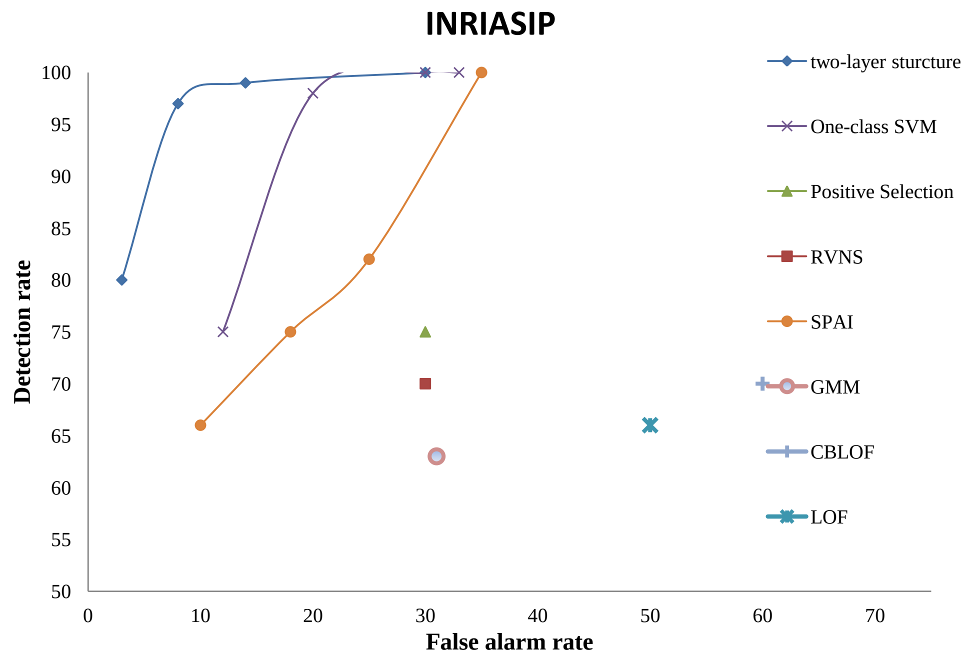


Figure 7.10: ROC curve for INRIASip dataset for differet algorithms

Computational Complexity

The complexity of algorithms consists of space and time complexity. In the following, the time and space complexity of all mentioned approaches are discussed and it shown that the space and time complexity of the proposed approach in terms of labeling new instances and updating stage are better than other approaches.

Instead of assessing only the running time of algorithms, the approaches were compared based on their algorithmic complexity.

The best algorithms in terms of accuracy were selected and compared. LOF and CBLOF algorithms do not generate sufficiently accurate results. Moreover, LOF and CBLOF algorithms use the whole data to find the outlierness factor for each instance. As a result there is no point in adding them in our comparisons.

Positive selection and RVNS algorithms use many instances to create an acceptable result, which sometime goes up to the size of the training data. Therefore, they are excluded from comparisons.

The GMM algorithm is not accurate enough to consider for further experiments. The SPAI algorithm also uses a large number of instances to generate accurate results, and it is not comparable to the proposed method and is excluded from comparisons.

For time and space complexity comparisons, this leaves the one-class SVM and the two-layer structure. The one-class SVM is a powerful algorithm for classification. However, its high computational cost in the test phase makes it difficult to use in large scale applications and online environments [Kang and Cho, 2014]. Both the training and testing phases of SVM are time-consuming. Standard SVM training has $O(n^3)$ time and $O(m)$ space complexities, where n is the training set size and m is the number of support vectors [Tsang et al., 2005]. Test time prediction is linear in the number of features and support vectors.

There are different approaches toward making SVM fast and applicable to an online environment. Kang and Cho employed hybrid neural network (HNN), to speed up the SVM algorithm, although this improvement in speed decreases the accuracy [Kang and Cho, 2014]. Geebelen et al. presented a new approach to decrease the number of support vectors by dividing up the feature space, and obtaining the support vectors with a hard margin [Geebelen et al., 2012].

To make it clear, the decision function for labeling new instances, in both SVM and two-layer structure are discussed in the following.

the decision function for SVM is 7.10:

$$f(x) = sgn(\sum_{i=1}^{n} \alpha_i y_i K(x, xi) + b) \tag{7.10}$$

where n is the number of support vectors. The labeling is linear in the number of support vectors.

The GMM formula is shown in Equation 7.11.

$$g(x) = \sum_{i=1}^{n} w_i N_{\mu_i, \Sigma i}(x) \tag{7.11}$$

where n is the number of GMMs. The labeling is linear in the number of GMMs components.

As can be seen in Table 7.5, the one-class SVM used a lot of support vectors to preserve the boundary of a class. There are two problems with a large number of support vectors. First, labeling new instances would be time-consuming. Second, the updating stage would be difficult with a large number of support vectors. However, as shown in Table 7.5 the relative space complexity of the two-layer structure was small. It represents all datasets as a set of GMMs and all calculations to label new instances are simple for finding a membership value for each GMM. Other than fast labeling of new instances, the updating step would be also fast. According to Equation 7.11, for KDDCUP99, there were 180 Gaussian components, and the calculation was repeated 180 times, while using Equation 7.10 one class SVM runs similar calculation 5106 times.

DataSet	two-layer structure	One-class SVM
KDDCUP99	180*3	5106
Darpa98	300*3	47346
NSL_KDDP	98*3	3765
NSL_KDDM	110*3	4034
DataSetMe	56*3	2068
IUSTSip	287*3	45766
INRIASip	109*3	22344

Table 7.5: Memory usage of the one-class SVM and the two-layer structure

To show the efficiency of two-layer structure in removing redundant instances, the two-layer structure is compared to CUGMM algorithm presented in Chapter 7. The CUGMM method does not use the idea of two layers and removing redundant instances. Table 7.6 shows the results of the comparisons of two-layer with CUGMM method on different datasets. To show the difference better the bar charts on Figure 7.11 depicts the difference as well.

As shown in the result, the updating time of two-layer structure is 2 to 3 times less than CUGMM method. This reduction is the direct result of ignoring redundant instances and using two-layer structure. To show the difference better the bar charts on Figure 4.11 depicts the difference as well.

To make sure that the accuracy is not decreased the one-layer structure (CUGMM) is compared to two-layers structure and the results on table 7.7 show that the result is almost the same.

Dataset/Algorithm	two-layer structure	CUGMM (one-layer)
KDDCUP99	225_{sec}	845_{sec}
Darpa98	641_{sec}	734_{sec}
NSL_KDDM	245_{sec}	634_{sec}
NSL_KDDM	245_{sec}	634_{sec}
DataSetMe	125_{sec}	301_{sec}
IUSTSip	388_{sec}	1261_{sec}
INRIASip	325_{sec}	811_{sec}

Table 7.6: Updating time for the two-layer structure and CUGMM methods

Removing Noise

As mentioned, the proposed algorithm is noise-resistant. The first effect of noise on a dataset is that it decreases the accuracy of the detection algorithm. In the following, results are shown for the output of the algorithm with incremental clustering over 10 rounds. The first dataset on the top left of Figure 7.12 is the original dataset created syntactically. This dataset included four main clusters, with the rest of the data represented in the data space being just random noise added to the data. The clusters were generated with random shape to show the ability of the algorithm to work with different shapes of clusters. Each cluster was representative of a normal behavior in the data space.

The top right figure in the Figure 7.12 depicts the result of running of the algorithm for the first round, which created the initial clusters to be updated in the next rounds. These clusters were representative of normal clusters and they followed the shape of the initial normal clusters represented original

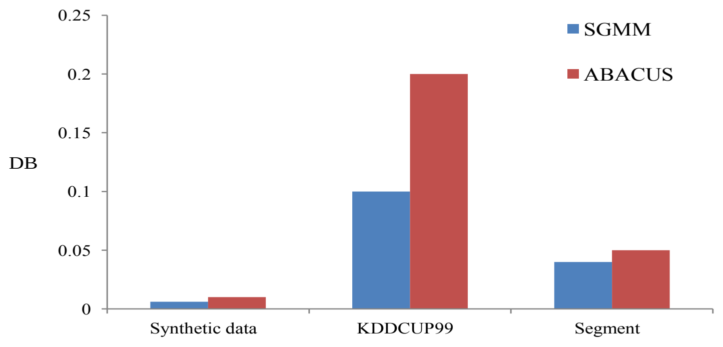


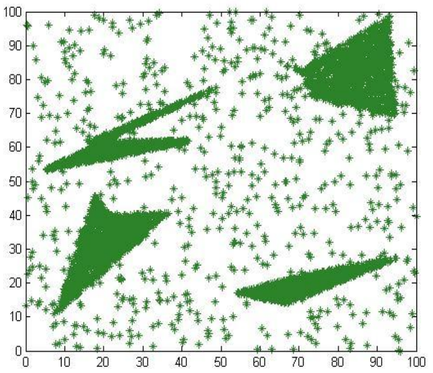
Figure 7.11: Bar chart of updating time for the two-layer structure and CUGMM methods

dataset. This means that each cluster was representative of the distribution for each normal behavior.

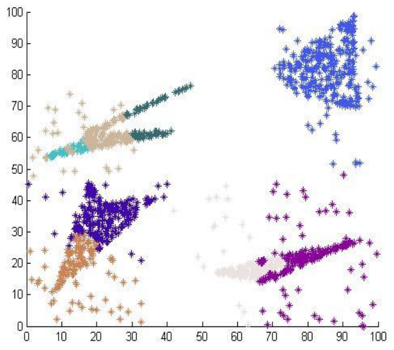
In the second round more data were added, and these data included the noise as well. In the third picture in Figure 7.12 in the second round some data is added to the top left clusters. After updating and merging the clusters, it is evident that the top left cluster was updated, and that noise instances added to data were removed. The top left cluster has a more solid outline, with less noise around its boundary.

During round five of updating, the bottom-left cluster was updated, and the noise was in the previous figure was removed, with more solid boundaries generated for that cluster. After five rounds, the algorithm still preserved the shape of the clusters and made the boundaries of clusters even more concrete with new incoming instances.

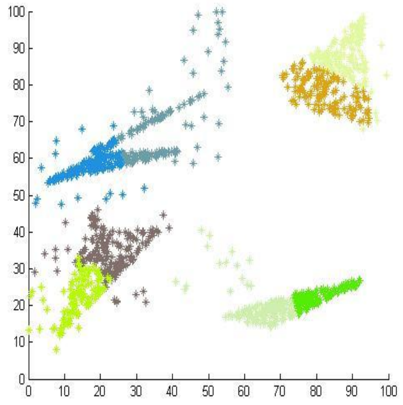
During the seventh round, the top right cluster was getting more instances and also the algorithm removed more noise around this cluster. Most of the



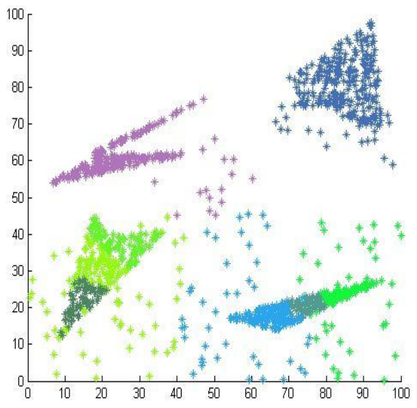
Original data



Cluster after round 1

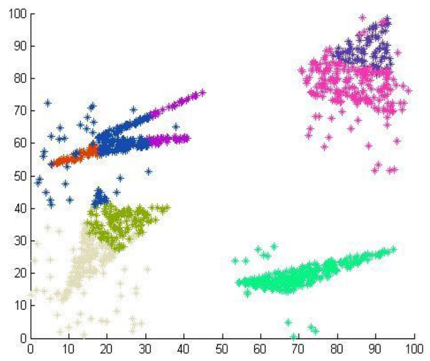


Cluster after round 2

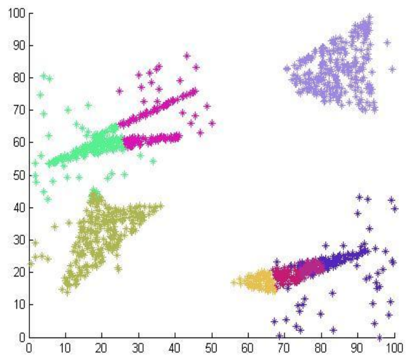


Cluster after round 3

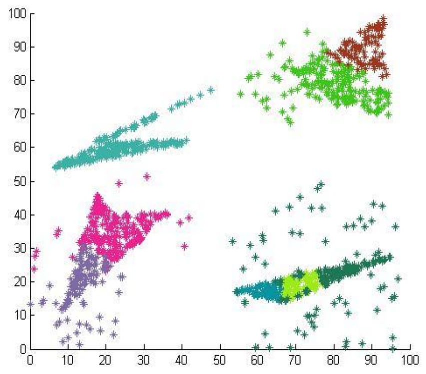
Figure 7.12: Incremental clustering in the first three rounds



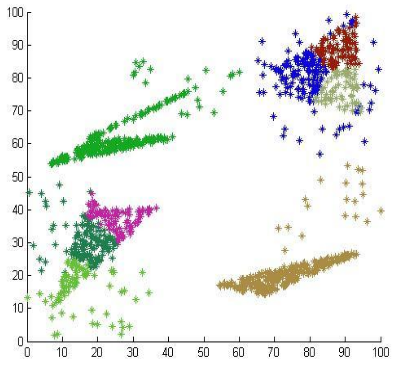
Cluster after round 4



Cluster after round 5

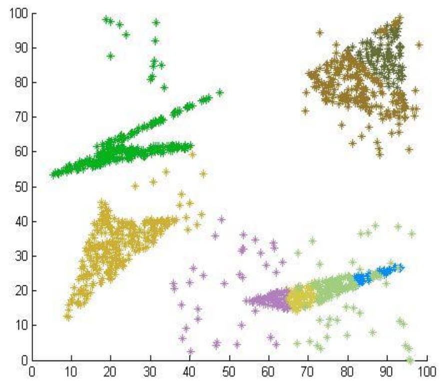


Cluster after round 6

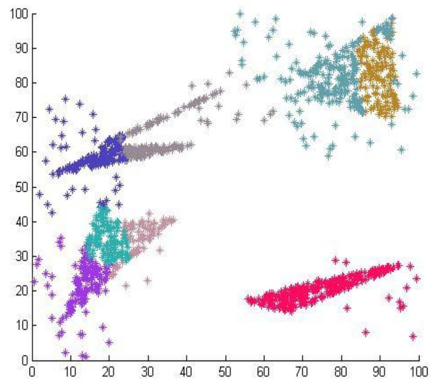


Cluster after round 7

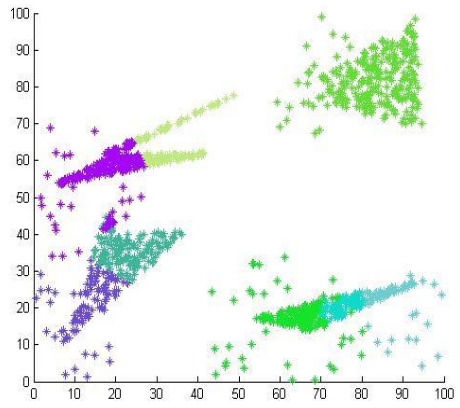
Figure 7.13: Incremental clustering in the fourth through seventh rounds



Cluster after round 8



Cluster after round 9



Cluster after round 10

Figure 7.14: Incremental clustering in the eighth through tenth rounds

Dataset/Algorithm	two-layer structure	CUGMM (one-layer)
	(DR,FA)	(DR,FA)
KDDCUP99	$(98, 2)$	$(98, 1)$
Darpa98	$(96, 3)$	$(94, 4)$
NSL_KDDM	$(85, 7)$	$(84, 8)$
NSL_KDDM	$(67, 12)$	$(65, 15)$
DataSetMe	$(99, 4)$	$(95, 8)$
IUSTSip	$(95, 2)$	$(95, 1)$
INRIASip	$(96, 2)$	$(95, 4)$

Table 7.7: Detection and false alarm rate for two-layer structure and CUGMM methods

noise instances that are depicted in the first figure were removed, although with some noise still evident around the clusters.

The result of the incremental algorithm on the ninth round shows that the bottom right clusters was the same as the original cluster,with little or no noise around it. Some noise was added to this cluster however, although there was less noise than in the original data. Moreover, the remaining noise was mainly visible in the boundaries of clusters.

The noise instance cannot create a distribution close to one of the already existing clusters. Therefore, in the merging step they are excluded for merging and are removed. This approach helps to remove any noise that may make clusters divergent. Moreover, the redundant instances were removed in each step based on the two-layer architecture. The result on the Figure 7.14 shows that this removal did affect the clusters.

7.5 Conclusion

This chapter presented the entire two-layer structure for anomaly detection. In this chapter the idea of removing redundant instances was introduced with defining the coarse and fine levels. Since the main focus of this thesis is on presenting a new incremental structure to work in an online environment, the time a central focus. The addition of the coarse and fine levels helps the structure to remove redundant instances with simple computations in the fine level. Most of new instances are removed since, they have already been accounted in the current model.

To test the accuracy of the structure, the false alarm and detection rate of the proposed method were compared to those of other methods. The results show that the proposed method increased the detection rate from 5% to 10%, and the false alarm rate is decreased 5% to 15% over different data sets. The closest performance to the proposed method was for one-class SVM, that has training time of between $O(n^2)$ and $O(n^3)$ based on optimization and kernel function, while the complexity of our algorithm is $O(n^2)$ in the worst-case, where n is the size of dataset.

Also as it is mentioned the testing time of supprt vector machine depends on the number of support vectors it generates. The result shows that to reach the high accuracy the number of support vectors for one-class SVM is very high. The experiments show that the two-layer structure is 20 to 50 times faster than the one-class SVM in labeling new instances.

Chapter 8

Conclusion and Future work

In this thesis, a two-layer structure was presented for anomaly detection.

This two-layer structure first partitions data into some clusters. These clusters are representative of normal behavior in a data space, with the assumption that everything outside of these boundaries is abnormal. Therefore, cluster representation is important.

To create boundaries for clusters, arbitrary shape clustering methods are employed. Other clustering methods were considered as well, and the two-layer structure does not depend on clustering approach. Labeling incoming instances with a naive approaches is time-consuming. To solve this problem, SGMM approach is proposed that represents each cluster as a GMM. GMM representation of each cluster involves characterizing each cluster using a set of means, covariances and weights to retain cluster distribution. The SGMM algorithm presented in Chapter 4 uses a recursive function to find the number of GMM components of a cluster automatically. With the GMM representation of a cluster, new incoming instance can be labeled quickly since the instance is simply fed to a GMM formula and the resulting calculation of membership value for each GMM is rapid. The GMM is just a summary, but it is capable of generating the original data accurately. From summarization perspective, the SGMM is three times faster than the ABACUS method, and memory usage is half that of the ABACUS method. However, the accuracy of the SGMM was not decreased despite the reduced time and memory requirements.

The two-layer structure is intended for use in an online environment, for which the main concerns are space and time complexity. To further decrease the space and time complexity, a collective labeling approach was proposed in Chapter 5 labels new incoming instances collectively. This enables updating of clusters using clusters of new instances rather than updates based on every new incoming instance. Based on collective labeling, new instances are collected and then added to the appropriate clusters. The major step in CPL algorithm was how to measure the distance between existing and new clusters. As mentioned, the SGMM algorithm represents existing and new clusters as a GMMs. As a result, distribution distance can be used to measure the distance of new and current clusters, and thereby to find the closest clusters.

There are various approaches available to measure the distance of two GMMs which are typically used in the areas of signal and image processing. A distance measure was presented in Chapter 5 as a means of measuring distance between two clusters. Experiments on these methods show that this approach was able to find the closest cluster with high accuracy. The proposed CPL method increased the detection rate by 5% and false alarm rate is decreased 10% over different datasets.

In Chapter 6, an incremental updating step was described. This incremental structure uses an SGMM and CPL as its two primary components. Based on the GMM representation of both current and new clusters, the updating phase involves merging the two GMMs. This merging is based on some well-studied formulas, presented in Chapter 6. In the experiments in

Chapter 6, the whole structure using a GMM-based approach was shown to outperform other approaches in terms of clustering goodness and accuracy in anomaly detection applications. The SSQ was decreased by 15% on average over different datasets in comparison to Clustream and the basic model. Based on different clustering indexes the the clusters accuracy of the Clustream and basic models fluctuated between $[0.4 - 0.8]$ and $[0.2 - 0.7]$ respectively, while the CUGMM had more stable clustering in different rounds.

The last step was to assemble the pieces of the this structure into two-layer structure. As mentioned in Chapter 3, the important feature added to this structure is its ability to remove redundant instances. In many applications, most of new incoming instances are redundant, and their presence does not change the distribution. The first layer in two-layer structure labels redundant instances with a simple calculation, and the second layer sends new instances to the rag bag. The results show that removing redundant instances did not negatively affect the accuracy of the algorithm, although it had a beneficial effect on the speed of the algorithm. Experiments show the false alarm rate is decreased from 5% to 15% among different datasets, while the detection rate is increased from 5% to 10% in different datasets with two-layer structure.

The memory usage for the two-layer structure is 20 to 50 times less than that of one-class SVM. One-class SVM uses support vectors in labeling new instances, while the labeling of the two-layer structure depends on the number of GMMs. The experiments show that the two-layer structure is 20 to 50 times faster than the one-class SVM in labeling new instances. Moreover, the updating time of two-layer structure is 2 to 3 times less than CUGMM method. This reduction is the direct result of ignoring redundant instances and using two-layer structure.

To summarize all design decisions and innovations proposed in the thesis:

- Presenting a two-layer structure for anomaly detection
- Summarization based on GMM
- Finding number of GMM components automatically
- Collective labeling approach
- Introducing a new distance measure
- Cluster updating using GMM
- Removing redundant instances
- Decrease updating and labeling time

8.1 Limitations and Future Work

Despite these advances, there remain aspects which could be improved. The overall structure created for anomaly detection is based on a GMM representation, which may not be suitable in some applications. The amount of time needed for clustering and GMM creation in the updating phase is another area for improvement. The updating strategy could be improved by considering different approaches for merging and splitting clusters. Each of these is addressed in the following subsections.

8.1.1 Alternate Distribution Models

A Gaussian representation treats each internal cluster component as if generated from a normal distribution. This kind of representation is suitable for many datasets, since a mixture of normal distributions is employed for each cluster. However, other kinds of distribution could be considered. Aside from its Gaussian particulars, the entire two-layer structure could be suitable for use with other distributions, although a new distance measures and representations would need to be considered.

8.1.2 Decreasing the Complexity of the Algorithm

Another limitation open to improvement is the running time of the algorithm. The overall complexity of the algorithm is $O(n^2)$, based on finding distances for all pairs of points. This type of calculation could be improved with using a KD-tree approach [Moore, 1991]. Another complexity concern is clustering and GMM creation time for new clusters. To solve this problem, a new distributed approach for clustering and GMM creation could be used. The clustering algorithm used here, DBSCAN, has already been implemented in a distributed version [He et al., 2011]. A new version of SGMM is being implemented using the Hadoop in our research lab (KDD lab) at the University of Ottawa, which could make the proposed algorithm scalable and suitable for a large dataset. Note that the new version of SGMM implemented using Hadoop generated comparable results to the non-distributed version.

8.1.3 Concept Drift Detection

Another avenue for improvement is the structure of the rag bag. In the updating step, the algorithm reviews the data in the rag bag prior to updating the current clusters. The proposed approach collects new instances in a rag bag, using a fixed memory allocation. An alternative could focus on collecting data in rag bag till detecting concept drift, rather than using a fixed memory size. Concept drift detection can follow a variety of approaches, which can be used to find an efficient time to update existing clusters with new clusters in rag bag. A concept drift measure can be defined based on the distance between GMMs in the rag bag and existing GMMs. A simple approach would be to find the average distance of new GMMs from their closest clusters. If the distance is found to be above a threshold, the clusters are updated. A method for comparing a covariance matrix and a mean matrix of GMMs introduced in [Song and Wang, 2005] could be used to determine the similarity of current and existing GMMs. This measure could be used as a basis for comparison and detection of concept drift, and allow identification of a best time for updating.

8.1.4 Improvements in Updating

The updating phase introduced in this thesis could be improved with better updating strategies. The strategy in the model described in the thesis uses a simple merging approach to update clusters. During updating, clusters can change, and new clusters can be added to each current cluster. A way to improve the model would be to develop a new strategy for merging and splitting clusters. In the proposed method, new GMMs are added to current GMMs, which increases the number of components. Two new approaches could be used for removing or splitting GMM components.

For the first, those with small weights could be removed, so as to avoid having GMM component quantities grow unreasonably. Alternatively, current GMMs could be split into two or more GMMs, based on the number of components and their weights. This approach would allow the GMMs to create new normal patterns in data that are created by time [Arandjelovic and Cipolla, 2005].

Another approach would deal with merging clusters. As GMMs grow, they would need to be merged if their boundaries were close; they could to be merged in such a way as to decrease the complexity associated with different clusters [Hennig, 2010].

List of References

[Aggarwal et al., 2003] Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceeding VLDB '03 Proceedings of the 29th international conference on Very large data bases*, pages 81–92.

[Agrawal et al., 1998] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *Proceeding SIGMOD '98 Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105.

[Aickelin et al., 2004] Aickelin, U., Greensmith, J., and Twycross, J. (2004). Immune system approaches to intrusion detection a review. In *In Proceedings of the 3nd International Conference on Artificial Immune Systems (ICARIS)*, page 316329.

[Amer et al., 2013] Amer, M., Goldstein, M., and Abdennadher, S. (2013). Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceeding ODD '13 Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15.

[Arandjelovic and Cipolla, 2005] Arandjelovic, O. and Cipolla, R. (2005). Incremental learning of temporally-coherent gaussian mixture models. In *In British Machine Vision Conference*, page 759768.

[Asgharian et al., 2011] Asgharian, Z., Asgharian, H., Akbari, A., and Raahemi, B. (2011). A framework for sip intrusion detection and response systems. In *Computer Networks and Distributed Systems (CNDS), 2011 International Symposium on*, pages 100–105.

[Asgharian et al., 2012] Asgharian, Z., Asgharian, H., Akbari, A., and Raahemi, B. (2012). *Detecting Denial of Service Attacks on SIP Based Services and Proposing Solutions*. Privacy, Intrusion Detection and Response: Technologies for Protecting Networks.

[Balthrop et al., 2002] Balthrop, J., BForrest, S., and BGlickman, M. (2002). Revisiting lisys: Parameters and normal behavior. In *In Congress On Evolutionary Computation*, page 10451050.

[Basu and Meckesheimer, 2007] Basu, S. and Meckesheimer, M. (2007). Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11:137–154.

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer wise training of deep networks. *Advances in neural information processing*, 19:153.

[Bengio and LeCun, 2007] Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards ai. *Large-Scale Kernel Machines.*

[Bifet et al., 2009] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavald, R. (2009). New ensemble methods for evolving data streams. In *Proceeding KDD '09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148.

[Boriah et al., 2008] Boriah, S., Chandola, V., and Kumar, V. (2008). Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the SIAM International Conference on Data Mining*, pages 243–254.

[Brause et al., 1999] Brause, R., Langsdorf, T., and Hepp, M. (1999). Neural data mining for credit card fraud detection. In *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*, pages 103 – 106.

[Breunig et al., 2000] Breunig, M. M., Kriegel, H. P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. In *Proceeding SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104.

[Bruna et al., 2007] Bruna, M., Simab, C., Huaa, J., Loweya, J., Carrollc, B., Suha, E., and Dougherty, E. (2007). Model-based evaluation of clustering validation measures. *Pattern Recognition*, 40:807 824.

[Cao et al., 2006] Cao, F., Ester, M., Qian, W., and Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *Proceedings of the SIAM International Conference on Data Mining*, page 328339.

[Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection : A survey. *ACM Computing Surveys*, 17:172.

[Chaoji et al., 2011] Chaoji, V., Li, G., Yildirim, H., and Zaki, M. J. (2011). Abacus: Mining arbitrary shaped clusters from large datasets based on backbone identification. In *Proceedings of the SDM 2011*, pages 295–306.

[Chatfield, 2004] Chatfield, C. (2004). *The analysis of time series : an introduction*. Chapman and Hall CRC, 6th edition.

[Chen et al., 2013] Chen, Y., Qian, J., and V, S. (2013). A new one-class svm for anomaly detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3567–3571.

[Chen and Tu, 2007] Chen, Y. and Tu, L. (2007). Density-based clustering for real-time stream data. In *Proceeding KDD '07 Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.

[Chow and Yeung, 2002] Chow, C. and Yeung, D. Y. (2002). Parzen-window network intrusion detectors. In *In Proceedings of the 16th International Conference on Pattern Recognition*, pages 10–18.

[Davies and Bouldin, 1979] Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1:224 – 227.

[Declercq and Piater, 2008] Declercq, A. and Piater, J. . (2008). Online learning of gaussian mixture models - a two-level approach. In *VISAPP 2008: Proceedings of the Third International Conference on Computer Vision Theory and Applications*, pages 605–611.

[E.Bigdeli et al., 2014a] E.Bigdeli, M.Mohammadi, B.Raahemi, and S.Matwin (2014a). Cluster summarization with dense region detection. In *ommunications in Computer and Information Science.*

[E.Bigdeli et al., 2014b] E.Bigdeli, M.Mohammadi, B.Raahemi, and S.Matwin (2014b). Summarizing arbitrary shape clustering using guassian mixture mode. In *Proc. the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*.

[E.Bigdeli et al., 2015a] E.Bigdeli, M.Mohammadi, B.Raahemi, and S.Matwin (2015a). A fast and noise resilient cluster-based anomaly detection. In *Pattern Analysis and Applications*.

[E.Bigdeli et al., 2015b] E.Bigdeli, M.Mohammadi, B.Raahemi, and S.Matwin (2015b). A fast noise resilient anomaly detection using gmm-based collective labelling. In *IEEE Technically Sponsored Science and Information Conference*.

[E.Bigdeli et al., 2015c] E.Bigdeli, M.Mohammadi, B.Raahemi, and S.Matwin (2015c). Incremental cluster updating using gaussian mixture model. In *Canadian Conference on AI*.

[Ebner et al., 2002] Ebner, M., Breunig, H. G., and Albert, J. (2002). On the use of negative selection in an artificial immune system. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 957964.

[Ertoz et al., 2003] Ertoz, L., Steinbach, M., and Kumar, V. (2003). Finding topics in collections of documents: A shared nearest neighbor approach. *In Clustering and Information Retrieval*, 10:83104.

[Eskin et al., 2002] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. *In Proceedings of Applications of Data Mining in Computer Security*, pages 78–100.

[Ester et al., 96] Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (96). A density based algorithm for discovering clusters in large spatial databases with noise. In *Published in Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, page 226231.

[Fan et al., 2001] Fan, W., Miller, M., Stolfo, S. J., Lee, W., and Chan, P. K. (2001). Using artificial anomalies to detect unknown and known network intrusions. In *In Proceedings of the 2001 IEEE International Conference on Data Mining. IEEE Computer Society*, pages 123–130.

[Fawcett and Provost, 1999] Fawcett, T. and Provost, F. (1999). Activity monitoring: noticing interesting changes in behavior. In *In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press*, pages 53–62.

[Forrest et al., 1994] Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R. (1994). Selfnonself discrimination in a computer. In *In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy. IEEE Computer Society Press.*

[Fowlkesa and Mallowsa, 1983] Fowlkesa, E. B. and Mallowsa, C. L. (1983). A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78:553–569.

[Gaddam et al., 2007] Gaddam, S., Phoha, V., and Balagani, K. (2007). K-means+id3: A novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods. *Knowledge and Data Engineering, IEEE Transactions on*, 19:345 – 354.

[Gaffney et al., 2006] Gaffney, S., Robertson, A., Smyth, P., Camargo, S., and Ghil, M. (2006). Probabilistic clustering of extratropical cyclones using regression mixture models. In *Technical Report UCI-ICS 06-02*.

[Gaffney and Smyth, 1999] Gaffney, S. and Smyth, P. (1999). Trajectory clustering with mixtures of regression models. In *In Proc. 5th ACM SIGKDD Intl Conf. on Knowledge Discovery and Data Mining*, page 6372.

[Geebelen et al., 2012] Geebelen, D., Suykens, J., and Vandewalle, J. (2012). Reducing the number of support vectors of svm classifiers using the smoothed separable case approximation. *Neural Networks and Learning Systems, IEEE Transactions on*, 23:682–688.

[Goldberger et al., 2003] Goldberger, J., Gordon, S., and Greenspan, H. (2003). An efficient image similarity measure based on approximations of kl divergence between two gaussian mixtures. In *Computer Vision. Proceedings. Ninth IEEE International Conference on*, pages 487 – 493.

[Guha et al., 2003] Guha, S., Meyerson, A., Mishra, N., Motwani, R., and andLiadan. O'Callaghan (2003). Clustering data streams: Theory and practice. *Journal IEEE Transactions on Knowledge and Data Engineering*, 15:515–528.

[Hajji, 2005] Hajji, H. (2005). Statistical analysis of network traffic for adaptive faults detection. *TRANSACTIONS ON NEURAL NETWORKS*, 16:1053– 1063.

[Han et al., 2006] Han, J., Kamber, M., and Pei, J. (2006). *Data Mining: Concepts and Techniques, Third Edition.* The Morgan Kaufmann Series in Data Management Systems, 500 Sansome Street, Suite 400, San Francisco, CA 94111.

[Han and Cho, 2006] Han, S. J. and Cho, S. (2006). Evolutionary neural networks for anomaly detection based on the behavior of a program. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36:559–570.

[Hawkins et al., 2002] Hawkins, S., He, H., Williams, G. J., and Baxter, R. A. (2002). Outlier detection using replicator neural networks. In *In Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180.

[He et al., 2011] He, Y., Tan, H., Luo, W., Mao, H., Ma, D., Feng, S., and Fan, J. (2011). Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 473–480.

[He et al., 2003] He, Z., Xu, X., and Deng, S. (2003). Discovering cluster-based local outliers. *Journal Pattern Recognition Letters*, 24:1641–1650.

[Hennig, 2010] Hennig, C. (2010). Methods for merging gaussian mixture components. *Advances in Data Analysis and Classification*, 4:3–34.

[Hershberger et al., 2009] Hershberger, J., Shrivastava, N., and Suri, S. (2009). Summarizing spatial data streams using clusterhulls. *Journal of Experimental Algorithmics (JEA)*, 13.

[Hershey et al., 2007] Hershey, J., Thomas, J., Watson, R., and Olsen, P. (2007). Approximating the kullback leibler divergence between gaussian mixture models. In *Acoustics Speech and Signal Processing 2007. ICASSP 2007. IEEE International Conference on*, pages 317–320.

[Hinneburg and Gabriel, 2007] Hinneburg, A. and Gabriel, H.-H. (2007). Denclue 2.0: fast clustering based on kernel density estimation. In *Proceeding IDA'07 Proceedings of the 7th international conference on Intelligent data analysis*, pages 70–80.

[Hofmeyr and Forrest, 2000] Hofmeyr, S. and Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary Computation*, 8:443473.

[Horn et al., 2001] Horn, P. S., Feng, L., Li, Y., and Pesce, A. J. (2001). Effect of outliers and nonhealthy individuals on reference interval estimation. *Clinical Chemistry*, 47:2137–2145.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Japkowicz et al., 1995] Japkowicz, N., Myers, C., and Gluck, M. (1995). A novelty detection approach to classification. In *IJCAI'95 Proceedings of the 14th international joint conference on Artificial intelligence*, pages 518–523.

[Ji and Dasgupta, 2004] Ji, Z. and Dasgupta, D. (2004). Real-valued negative selection algorithm with variable-sized detectors. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 287–298.

[k. Dunn and Dunn, 1974] k. Dunn and Dunn, J. (1974). Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95–104.

[Kang and Cho, 2014] Kang, S. and Cho, S. (2014). Approximating support vector machine with artificial neural network for fast prediction. *Expert Systems with Applications*, 41:49894995.

[Karypis et al., 1999] Karypis, G., Han, E., and Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *Journal Computer*, 32:68–75.

[Kaufman, 1990] Kaufman, L. (1990). *Finding Groups in Data. An Introduction to Cluster Analysis.* John Wiley, New York.

[Kaufman and Rousseeuw, 1987] Kaufman, L. and Rousseeuw, P. (1987). *Clustering by means of Medoids. In Statistical Data Analysis Based on the L1Norm and Related Methods.* Y. Dodge, North-Holland.

[Keerthi et al., 2006] Keerthi, S. S., Chapelle, O., and DeCoste, D. (2006). Building support vector machines with reduced classifier complexity. *Journal The Journal of Machine Learning Research*, 7:1493–1515.

[Keogh and Smyth, 1997] Keogh, E. and Smyth, P. (1997). A probabilistic approach to fast pattern matching in time series databases. *In Proceedings of Third International Conference on Knowledge Discovery and Data Mining, AAAI Press*, pages 24–20.

[Kersting et al., 2010] Kersting, K., Wahabzada, M., Thurau, C., and Bauckhage, C. (2010). Hierarchical convex nmf for clustering massive data. In *ACML*, pages 253–268.

[Kim and Han, 2009] Kim, M. S. and Han, J. (2009). A particle-and-density based evolutionary clustering method for dynamic networks. *Journal Proceedings of the VLDB Endowment*, 2:622–633.

[Knorr et al., 2000] Knorr, E. M., Ng, R. T., and Tucakov, V. (2000). Distance-based outliers: algorithms and applications. *The VLDB Journal The International Journal on Very Large Data Bases*, 8:237–253.

[Kou et al., 2006] Kou, Y., Lu, C. T., and Chen, D. (2006). Spatial weighted outlier detection. In *In Proceedings of SIAM Conference on Data Mining*, pages 110–118.

[Kullback and Leibler, 1951] Kullback, R. and Leibler, A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

[Laurikkala et al., 2000] Laurikkala, J., Juhola1, M., and Kentala, E. (2000). Informal identification of outliers in medical data. In *In Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pages 20–24.

[Lee et al., 2007] Lee, J.-G., Han, J., and Whang, K.-Y. (2007). Trajectory clustering: a partition-and-group framework. In *SIGMOD '07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604.

[Leng et al., 2009] Leng, M., Xinsheng, L., Guolv, T., and Xiaohui, X. (2009). Time series representation for anomaly detection. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pages 628–632.

[Lin et al., 2005] Lin, J., Keogh, E., and Herle, H. V. (2005). Approximations to magic: Finding unusual medical time series. In *Computer-Based*

Medical Systems, 2005. Proceedings. 18th IEEE Symposium on, pages 329 – 334.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, pages 281–297.

[Mahoney and Chan, 2003] Mahoney, M. V. and Chan, P. K. (2003). Learning rules for anomaly detection of hostile network traffic. In *In Proceedings of the 3rd IEEE International Conference on Data Mining. IEEE Computer Society*, page 610.

[Marais and Marwala, 2007] Marais, E. and Marwala, T. (2007). Predicting the presence of internet worms using novelty detection. In *arXiv:0705.1288*.

[McHugh, 2000] McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *Journal ACM Transactions on Information and System Security (TISSEC)*, 3:262–294.

[Minton et al., 1987] Minton, S., Carbonell, J., Etzioni, O., Knoblock, C., and Kuokka, D. (1987). Acquiring effective search control rules: Explanation-based learning in the prodigy system. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 120–140.

[Moayedi and Masnadi-Shirazi, 2008] Moayedi, H. Z. and Masnadi-Shirazi, M. A. (2008). Arima model for network traffic prediction and anomaly detection. In *International Symposium on Information Technology*, page 16.

[Mohamed. Nassar, 2008] Mohamed. Nassar, Radu. State, O. F. (2008). Monitoring sip traffic using support vector machines. In *Recent Advances in Intrusion Detection*, pages 311–330.

[Mohamed. Nassar, 2009] Mohamed. Nassar, Radu. State, O. F. (2009). Voip malware: Attack tool and attack scenarios. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6.

[Mohammadi et al., 2014] Mohammadi, M., Akbari, A., Raahemi, B., Nassersharif, B., and Asgharian, H. (2014). A fast anomaly detection system using probabilistic artificial immune algorithm capable of learning new attacks. *Evolutionary Intelligence*, 6:135–156.

[Mooney, 1989] Mooney, R. J. (1989). The effect of rule use on the utility of explanation-based learning. In *In Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 725–730.

[Moore, 1991] Moore, A. (1991). A tutorial on kd-trees. In *University of Cambridge Computer Laboratory Technical Report No. 209.*

[Ndousse and Okuda, 1996] Ndousse, T. and Okuda, T. (1996). Computational intelligence for distributed fault management in networks using fuzzy cognitive maps. In *Communications, 1996. ICC '96, Conference Record, Converging Technologies for Tomorrow's Applications. 1996 IEEE International Conference on*, pages 1558 – 1562.

[O'Callaghan et al., 2002] O'Callaghan, L., Mishra, N., Meyerson, A., and Guha, S. (2002). Streaming-data algorithms for high-quality clustering. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 685–694.

[Odin and Addison, 2000] Odin, T. and Addison, D. (2000). Novelty detection using neural network technology. In *In Proceedings of the COMADEN Conference.*

[Orair et al., 2010] Orair, G. H., Teixeira, C. H. C., Jr., W. M., and Wang, Y. (2010). Distance-based outlier detection: consolidation and renewed bearing. *Journal Proceedings of the VLDB Endowment*, 3:1469–1480.

[Otey et al., 2006] Otey, M. E., Ghoting, A., and Parthasarathy, S. (2006). Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery*, 12:203–228.

[Pincombe, 2005] Pincombe, B. (2005). Anomaly detection in time series of graphs using arma processes. In *ASOR BULLETIN*, page 210.

[Pires and Santos-Pereira, 2005] Pires, A. and Santos-Pereira, C. (2005). Using clustering and robust estimators to detect outliers in multivariate data. *In Proceedings of International Conference on Robust Statistics*, pages 35–44.

[Qin and Hwang, 2004] Qin, M. and Hwang, K. (2004). Frequent episode rules for internet anomaly detection. In *In Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications. IEEE Computer Society*, pages 810–818.

[Ramaswamy et al., 2000] Ramaswamy, S., Rastogi, R., and Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In *In Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438.

[Rand, 1971] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association (American Statistical Association)*, 66:846850.

[Ranzato et al., 2006] Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. *Advances in Neural Information Proccessing Systems.*

[Rendn et al., 2011] Rendn, E., Abundez, I., Arizmendi, A., and Quiroz, E. (2011). Internal versus external cluster validation indexes. *INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS*, 5:27–34.

[Rousseeuw, 1987] Rousseeuw, P. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics*, 20:5365.

[Sajja and Akerkar, 2010] Sajja, P. S. and Akerkar, R. (2010). Knowledge-based systems for development. *Advanced Knowledge Based Systems:Model, Applications and Research*, 1:1–11.

[Sakurada and Yairi, 2014] Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *MLSDA'14 Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4.

[Salvador and Chan, 2003] Salvador, S. and Chan, P. (2003). Learning states and rules for time-series anomaly detection. In *In Proceedings of the 2003 IEEE International Conference on Data Mining. IEEE Computer Society*, pages 180–188.

[Scholkopf et al., 2001] Scholkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Journal Neural Computation*, 13:1443–1471.

[Seiden and Celada, 1992] Seiden, P. E. and Celada, F. (1992). A model for simulating cognate recognition and response in the immune system. *Journal of Theoretical Biology*, 158:329357.

[Shinichi and Jun, 2000] Shinichi, M. and Jun, S. (2000). Traversing itemset lattice with statistical metric pruning. In *In Proc. of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems*, pages 15–28.

[Shona and Moon, 2007] Shona, T. and Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177:37993821.

[Sim and Lee, 2003] Sim, K. and Lee, D. (2003). Modeling of positive selection for the development of a computer immune system and a self-recognition algorithm. *International Journal of Control, Automation, and Systems*, 1:453–458.

[Singh, 2002] Singh, S. (2002). Anomaly detection using negative selection based on the r-contiguous matching rule. In *In Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, page 99106.

[Smith et al., 2002] Smith, R., Bivens, A., Embrechits, M., Palagiri, C., and Szymanski, B. (2002). Clustering approaches for anomaly-based intrusion detection. In *In Proceedings of the Intelligent Engineering Systems through Articial Neural Networks. ASME Press*, pages 579–584.

[Song and Wang, 2005] Song, M. and Wang, H. (2005). Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In *Proceedings of the SPIE 5803, Intelligent Computing: Theory and Applications III.*

[Song et al., 2007] Song, X., Wu, M., Jermaine, C., and Ranka, S. (2007). Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 19:631–645.

[Sricharan and III, 2011] Sricharan, K. and III, A. O. H. (2011). Efficient anomaly detection using bipartite k-nn graphs. In *In Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 478–486.

[Stefano and Vento, 2000] Stefano, C. and Vento, M. (2000). To reject or not to reject: that is the question an answer in case of neural classifiers. *IEEE Transactions on Systems, Management and Cybernetics*, 30:84–94.

[Stehman, 1997] Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62:7789.

[Stibor et al., 2005] Stibor, T., Mohr, P., Timmis, J., and Eckert, C. (2005). Is negative selection appropriate for anomaly detection ? *In: Genetic and Evolutionary Computation GECCO*, 8:843873.

[Tagawa et al., 2014] Tagawa, T., Tadokoro, Y., and Yairi, T. (2014). Structured denoising autoencoder for fault detection and analysis. In *JMLR: Workshop and Conference Proceedings (ACML)*, page 96111.

[Tan et al., 2005] Tan, P. N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley, -.

[Tandon and Chan, 2007] Tandon, G. and Chan, P. K. (2007). Weighting versus pruning in rule validation for detecting network and host anomalies. In *KDD '07 Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 697–706.

[Teodoro et al., 2009] Teodoro, P. G., Verdejo, J. E. D., Fernandez, G. M., and Vazquez, E. (2009). Anomaly-based network intrusion detection: techniques, systems and challenges. *Computer Security*, 28:18–28.

[Teukolsky et al., 2007] Teukolsky, S., Vetterling, W., and Flannery, B. (2007). *Conditional Entropy and Mutual Information*. The Art of Scientific Computing, New York: Cambridge University Press.

[Thatte et al., 2011] Thatte, G., Mitra, U., and Heidemann, J. (2011). Parametric methods for anomaly detection in aggregate traffic. *Networking, IEEE/ACM Transactions on*, 19:512–525.

[Tsang et al., 2005] Tsang, I. W., Kwok, J. T., and Cheung, P. (2005). Core vector machines:fast svm training on very large data sets. *Journal of Machine Learning Research*, 41:363392.

[Van and Faisal, 2011] Van, B. J. and Faisal, S. (2011). Distortion measurement for automatic document verification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 289 – 293.

[van. Rijsbergen, 1979] van. Rijsbergen, C. J. (1979). *Information Retrieval.* Butterworth.

[Wang et al., 1997] Wang, W., Yang, J., and Muntz, R. R. (1997). Sting: A statistical information grid approach to spatial data mining. In *Proceeding VLDB '97 Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195.

[Wijsen and .Meersman, 1998] Wijsen, J. and .Meersman, R. (1998). On the complexity of mining quantitative association rules. *Data Mining and Knowledge Discovery*, 3:263281.

[Wu et al., 2015] Wu, C., Guo, Y., and Ma, Y. (2015). Adaptive anomalies detection with deep network. In *Proceeding of The Seventh International Conference on Adaptive and Self-Adaptive Systems and Applications.*

[Wu and Jermaine, 2006] Wu, M. and Jermaine, C. (2006). Outlier detection by sampling with accuracy guarantees. In *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 767–772.

[Wu and Liang, 2005] Wu, Z. and Liang, Y. (2005). Self-regulating method for model library based artificial immune systems. In *In Proceedings of 4th International Conference on Artificial Immune Systems*, page 353365.

[Xie et al., 2013] Xie, M., Hu, J., Han, S., and Chen, H. (2013). Scalable hypergrid k-nn-based online anomaly detection in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 24:1661–1670.

[Yamanishi et al., 2004] Yamanishi, K., Takeuchi, J. I., Williams, G., and Milne, P. (2004). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Data Mining and Knowledge Discovery*, pages 275–300.

[Yang et al., 2011] Yang, D., Rundensteiner, E. A., and Ward, M. O. (2011). Summarization and matching of density-based clusters in streaming environments. *Journal Proceedings of the VLDB Endowment*, 5:121–132.

[Ye and Chen, 2001] Ye, N. and Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. In *International conference on Quality and Reliability Engineering*, pages 105–112.

[Yuting et al., 2013] Yuting, C., Jing, Q., and Saligrama, V. (2013). A new one-class svm for anomaly detection. In *Proceedings of Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3567–3571.

[Zhang and Wang, 2006] Zhang, J. and Wang, H. (2006). Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and Information Systems*, 10:333–355.

[Zhang et al., 1997] Zhang, T., Ramakrishnan, R., and Livny, M. (1997). Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1:141–182.

[Zhou et al., 2010] Zhou, C. V., Leckie, C., and Karunasekera, S. (2010). A survey of coordinated attacks and collaborative intrusion detection. *Computers and Security*, 29:124–140.

Appendix A

A.1 Acronyms

Acronym	Definition
AIS	Artificial Immune System
AR	Autoregression
ARIMA	Autoregressive Integrated Moving Average
CPL	Collective Probabilistic Labeling
CUGMM	Cluster Updating based on GMM
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
EM	Expectation Maximization
GMM	Gaussian Mixture Model
IDS	Intrusion Detection Systems
LOF	Local Outlier Factor
MA	Moving Average
PCA	Principle Component Analysis
RVNS	Real-Value Negative Selection
SGMM	Summarization based on Gaussian Mixture Model
StrDA	Structured Denoising Autoencoder

A.2 Cluster Validity Indexes

The most well-known clustering indexes include:

Silhouette

Silhouette is an internal cluster measure that defines good clustering as being cohesive and well-separated. The formula for Silhouette is as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{A.1}$$

For instance i, $a(i)$ is the average dissimilarity of i with all other data within the same cluster, and $b(i)$ is the lowest average dissimilarity of i to any other cluster, of which i is not a member.

Rand measure

Rand index measures how similar clustering is to the original classes:

$$RI = \frac{TP + TN}{TP + TN + FN + TN} \tag{A.2}$$

where TP is the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives.

F-measure

F-measure includes the contribution of false negatives by weighting recall:

$$\begin{aligned} F_\beta &= \frac{(\beta^2+1).P.R}{\beta^2.P+R} \\ R &= \frac{TP}{TP+FN} \\ P &= \frac{TP}{TP+FP} \end{aligned} \tag{A.3}$$

where P is the precision rate and R is the recall rate. Increasing β allocates an increasing amount of weight to recall.

Jaccard index Jaccard index finds similarity between the clusters returned by the clustering algorithm and the benchmark classifications.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{A.4}$$

where A is the cluster, and B is the original class. A higher value indicates better clustering.

Fowlkes-Mallows index Fowlkes-Mallows index calculates the similarity of clusters and a benchmark classification:

$$FM = \sqrt[2]{\frac{TP}{TP+FP}.\frac{TP}{TP+FN}} \tag{A.5}$$

where TP is the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives. The higher the value, the better the clustering.

Normalized Mutual Information (NMI)

Normalized Mutual Information employs information theory to measure how much information is shared between a cluster and a ground-truth classification:

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{A.6}$$

where, $I(X,Y)$ denotes the mutual information between two random variables X and Y and $H(X)$ denotes the entropy of X, X will be consensus clustering while Y will be the true labels.

A.3 Clustering Goodness

The difference in clustering goodness between the original and regenerated datasets based on different factors is represented in the Table A.1.

A.4 Clustering Performance of Different Algorithms

The results of table A.2 are presented here, with detailed numerical values.

Dataset-Index	Dunn	DB	SD	SSQ
	(CSGMM, BACUS)	(CSGMM, BACUS)	(CSGMM, BACUS)	(CSGMM, BACUS)
MagicGamma	(0.0010, 0.0032)	(1.04, 1.09)	(0.48, 0.64)	(169, 1235)
Wave	(0.020, 0.024)	(0.11, 0.29)	(0.62, 0.83)	(125, 2243)
Shuttle	$(4.6e-05, 2.74e-05)$	(0.49, 0.49)	(0.32, 0.45)	(179, 7999)
Segment	(0.0045, 0.014)	(0.33, 0.44)	(0.12, 0.41)	(7790, 9314)
P2P	$(6.5e-06, 0.00014)$	(0.46, 0.67)	(0.55, 0.63)	(207, 1021)
KDD	$(2.04-05, 0.0038)$	(1.04, 1.05)	(0.69, 1.11)	(1463, 7788)
Diabet	(0.001, 0.008)	(1.65, 1.98)	(0.29, 0.31)	(135, 192)
CMC	(0.09, 0.17)	(2.93, 3.11)	(0.09, 0.17)	(116, 1497)
Synthetic	(0.22, 0.33)	(1.82, 2.15)	(0.05, 0.19)	(275, 1823)

Table A.1: Difference of DB indexes for original and regenerated datasets

Clustering index	Dunn			DB			SD			Purity		
DataSet/Algorithm	CUGM	Clustream	CWR	CUGM	Clustream	CWR	CUGM	Clustream	CWR	CUGM	Clustream	CWR
KDD	1×10^{-6}	2×10^{-8}	0.0003	0.613	0.81	0.35	2.01	3.2	2.22	0.76	0.63	0.79
	0.000 25	1×10^{-6}	0.0001	0.634	0.88	0.46	2.32	2.24	2.26	0.72	0.61	0.73
	0.000 25	1×10^{-6}	0.0014	0.634	0.78	0.35	1.93	3.24	2.28	0.77	0.65	0.79
	0.000 25	1×10^{-6}	0.0006	0.634	0.71	0.69	2.3	2.27	2.27	0.74	0.66	0.73
Shuttle	7×10^{-6}	2×10^{-8}	3×10^{-5}	0.332	0.49	0.2	6.35	7.15	6.42	0.96	0.81	0.95
	5×10^{-6}	1×10^{-8}	3×10^{-5}	0.355	0.53	0.3	3.98	4.35	4.08	0.95	0.84	0.98
	7×10^{-5}	2×10^{-7}	3×10^{-5}	0.332	0.47	0.28	4.75	5.80	3.85	0.96	0.83	0.90
	9×10^{-7}	2×10^{-8}	2×10^{-5}	0.363	0.59	0.29	3.88	4.98	3.98	0.96	0.85	0.95
MagiccGamma	0.004 89	0.0001	0.0222	0.777	0.87	0.66	1.89	2.89	1.9	0.76	0.64	0.73
	0.073 85	0.0233	0.2372	0.628	0.88	0.5	1.87	2.75	1.89	0.75	0.63	0.73
	0.017 56	0.0012	0.1052	0.702	0.901	0.66	1.88	2.84	1.9	0.75	0.64	0.75
	0.019 60	0.0039	0.0339	0.727	0.94	0.7	1.89	2.71	1.9	0.75	0.64	0.75
Synthetic	2×10^{-9}	1×10^{-11}	1×10^{-5}	0.40	0.45	0.35	2.25	3.25	2.08	0.89	0.79	0.91
	2×10^{-9}	1×10^{-11}	1×10^{-5}	0.39	0.43	0.29	2.04	4	1.91	0.89	0.75	0.9
	1×10^{-8}	5×10^{-10}	8×10^{-6}	0.38	0.74	0.38	2.6	3.1	1.88	0.9	0.82	0.88
	2×10^{-7}	4×10^{-9}	6.0×10^{-6}	0.38	0.68	0.28	2.33	4.47	1.77	0.9	0.81	0.9

Table A.2: Performance of clustering based on Dunn, DB, SD and Purity indexes