TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

BERKAY KICANAOGLU

UNSUPERVISED ANOMALY DETECTION IN UNSTRUCTURED LOG-DATA FOR ROOT-CAUSE-ANALYSIS

Master's Thesis

Examiner: Prof. Moncef Gabbouj
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 4 March 2015

ABSTRACT

BERKAY KICANAOGLU: Unsupervised Anomaly Detection
in unstructured log-data for root-cause-analysis
Tampere University of Technology
Master's Thesis, 64 pages, 0 Appendix pages
April 2015
Master's Degree Programme in Information Technology
Major: Signal Processing
Examiner: Prof. Moncef Gabbouj
Keywords: anomaly detection, density-based clustering, neural networks, support vector data description, pattern recognition, machine learning,data mining

Anomaly detection has attracted the attention of researchers from a variety of backgrounds as it finds numerous applications in the industry. As a subfield, fault detection plays a crucial role in growing telecommunications networks since failures lead to dissatisfaction and hence financial drawbacks. It aims at identifying unusual events in the system log files. System logs are messages from the elements of the network to highlight their status. The main challenge is to cope with the rate the data volume grows. Traditional methods such as expert systems are no longer practical making machine learning approaches more valuable.

In this thesis work, unsupervised anomaly (fault) detection in unstructured system logs is investigated. The effect of various feature extraction methods are investigated in terms of the gain they provide. Also, the baseline dimensionality reduction method Principal Component Analysis (PCA) and its effects are given. Additionaly, autoencoders are studied as an alternative dimensionality reduction technique. Four different methods based on statistics and clustering as well as a framework to clean datasets from anomalies are discussed. A high detection (classification) rate with 99.69% precision and 0.07% false alarm rate are achieved in one of the datasets while similar results have been achieved with variations in the recall in the other dataset. The studies show that the dimensionality reduction can greatly improve the performance of the classifiers used and reduce the computational complexity in anomaly detection.

PREFACE

This work has been conducted at the Department of Signal Processing of Tampere University of Technology in collaboration with TIETO within ICT SHOK D2I project.

I would like to express my gratitudes to my supervisor Prof. Moncef Gabbouj for engaging me in such an exciting subject and for the support throughout the thesis. Furthermore, I believe I owe Honglei Zhang and Stefan Uhlmann big thanks for they contributed in this work with their valuable opinions, comments and guidance. Also, I would like to thank Alexandros Iosifidis for bringing a different perspective to the thesis and being very supportive.

There are many other to thank for being there when things got out of hand. However, my friends with whom I shared a lot deserves a huge warm thanks for their friendship and support. In addition to those, special thanks go to my team mates and coaches with whom we shared pain, fun and, of course, glorious victories.

Finally, I would like to thank my family for being there no matter what. Their belief and support always make me go one step ahead from where I am.

Berkay Kicanaoglu

Tampere, June 2015

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

ANN	Artificial Neural Network
AUC	Area Under Curve
BoL1	Bag-of-lines (Legal Word Selection Rule Set-1)
BoL1	Bag-of-lines (Legal Word Selection Rule Set-2)
BoW	Bag-of-words
CD	Core Distance
DB	Density-based
DOS	Density Outlier Score
FAR	False Alarm Rate
FN	False Negative
FP	False Positive
KNN	K-nearest Neighbors
KL	Kullback-Leibler
KSVDD	Kernel Support Vector Data Description
LD	Local Density
LOF	Local Outlier Factor
ML	Maximum Likelihood
NB	Naive Bayesian
$\mathcal{NP}$-hard	Nondeterministic Polynomial Time-hard
OPTICS	Ordering Points to Identify the Clustering Structure
OS	Outlier Score
PCA	Principal Component Analysis
RD	Reachability Distance
ROC	Receiver Operating Characteristic
SVDD	Support Vector Data Description
SVM	Support Vector Machines
TN	True Positive
TP	True Negative
w.r.t.	with respect to

$\boldsymbol{a}$	center vector of the data description, page 36
b	bias value, page 17
β	parameter controlling the sparsity constraint, page 18
c	class label, page 25
$\mathcal{C}$	set of all possible classes, page 25
$\boldsymbol{g}$	approximating function, page 5

g	number of clusters, page 6
$\boldsymbol{h}$	mapping function, page 5
$h_{W,b}(.)$	hypothesis function learnt by the autoencoder given W and b, page 17
$\mathcal{D}$	data set, page 5
$J(.)$	squared error cost function, page 18
$J_{sparse}(.)$	squared error cost function with sparsity constraint, page 18
$p(.)$	probability, page 6
$p(.;,)$	joint probability, page 6
$\Pr(. \mid .)$	conditional probability, page 25
π_i	mixing proportion associated with an indexed component, page 6
θ_i	parameter vector associated with an indexed component, page 6
θ_i^c	conditional probability $\Pr(x_i \mid c)$, page 25
ϵ	parameter to determine the radius of neighborhood, page 32
λ	weight decay parameter, page 18
k	number of nearest neighbors, page 30
$MinPts$	minimum number of points to form a cluster, page 32
$N_\epsilon(.)$	ϵ-neighborhood of a sample, page 32
ρ	sparsity parameter, page 18
α_i	lagrange multiplier, page 36
γ_i	lagrange multiplier, page 36
$\hat{\rho}_j$	average activation in an indexed neuron in the hidden layer, page 18
ξ_i	slack variable to introduce flexibility in data description boundary, page 36
$\mathcal{L}$	log-likelihood, page 5
L	error function, page 36
$\mathbf{P}$	linear transformation matrix, page 13
R_i	the radius of the sphere w.r.t. an indexed sample, page 30
R	the radius of the data description, page 36
μ_i	expected value of an indexed feature, page 14
$\sigma_{i,j}^2$	covariance of two indexed features, page 14
$\boldsymbol{S}_{\mathbf{X}}$	covariance matrix associated with the data matrix, page 14
s_l	number of neurons in lth layer, page 18
σ	smoothing parameter for the RBF kernel, page 39
$\boldsymbol{x}$	multi-dimensional vector, page 4
x_i	ith (scalar) feature of a multi-dimensional vector $\boldsymbol{x}$, page 4
$x^{(i)}$	multi-dimensional vector associated with an index, page 17
$\boldsymbol{x}_i$	multi-dimensional vector associated with an index, page 5

$\boldsymbol{x}^*$	novel multi-dimensional vector, page 25
$\mathcal{X}$	input space, page 5
$\mathbf{X}$	matrix containing data (row-major), page 13
y_i	label associated with an index, page 5
$\mathcal{Y}$	category space, page 5
W^T	weight matrix associated with a neuron, page 16

1. INTRODUCTION

The advances in electronics and multimedia technologies in past few decades coined a new term referred to as 'Big Data'. As use of smart devices and social media becomes a cornerstone in our lives, the amount of data generated and stored every day rises up to unprecedented figures. Big Data refers to this type of very high-volume data sets which cause the traditional software tools to fail in management and interpretation of the content. As a consequence of the emerging 'Big Data' notion, the role of effective multimedia information retrieval and recognition of patterns becomes more vital. Anomaly detection is a field of interest in pattern recognition dealing with all possible forms of data such as text, image and audio.

Anomaly detection aims to identify certain events which do not conform with the general patterns in the data sets. It is a popular topic in the academia since it finds extensive use in many engineering disciplines and industry. A small subset of applications can be listed as fraud detection, intrusion detection and fault detection. There exist numerous methods proposed in pattern recognition and machine learning fields addressing these problems. The methods that fall into unsupervised learning category should be investigated and tailored for textual data. An up-to-date survey on the various proposals in literature can be found in [43].

1.1 Anomaly Detection

Anomaly (outlier) detection is a significant problem which has been studied in the domains of pattern recognition and machine learning. Various anomaly detection techniques have been proposed for certain tasks such as fraud detection [23], cyber-intrusion detection [29], fault detection [35, 57], industrial damage detection [32] and image processing [7]. Since the problems vary in the nature of data as well as the domain of application, different methods from machine learning and pattern recognition have been studied along with different signal processing approaches.

Analysis of textual data can be associated with many interesting industrial and daily life problems. For instance, anomaly detection can be used to bring users the novel news headlines or documents in a search engine or it can be exploited to identify possibly spam emails by a service provider. More specifically, anomaly detection can also be used for fault detection in large-scale systems such as telecommunication networks. In this work, unstructured system logs have been delved into, in order to reveal anomalous behavior exhibited by the elements of a telecommunications network.

System logs are in fact time-series signals although they may not necessarily be sampled uniformly in terms of time. However, they contain large amount of information to be extracted, especially when the data size grows to an extent which is burdensome to handle manually. Hence, it is obvious that research for efficient methods is necessary.

1.2 Objectives of the Thesis

The objectives of this thesis consist of understanding the nature of the data (unstructured system logs), searching for useful feature representations and understanding the existing methods and concepts in unsupervised domain in order to apply on the given datasets. Additionally, the objectives include studying dimensionality reduction methods for low-dimensional manifolds determination for enhancing the performance of anomaly detection methods in general and comparing the performance of various methods.

1.3 Results of the Thesis

The main results for this thesis are that:

- conventional feature extraction method for textual data, bag-of-words, can be replaced by an analogous method bag-of-lines to improve the results for the given problem.
- scoring-based techniques can lead to very good results by allowing more flexibility for the analyst to focus on the most relevant anomalies in this problem.
- dimensionality reduction is crucial for both time efficiency and performance.

1.4 Structure of the Thesis

The organization of the thesis is as follows. Chapter 2 presents a literature overview on pattern recognition, anomaly detection, dimensionality reduction methods and evaluation techniques. Chapter 3 gives more detailed description of the used methods including other building blocks such as preprocessing and feature extraction. Chapter 4 provides the evaluation of methods in terms of parameters and performance. Finally, the thesis is concluded in Chapter 5 by discussions and future research plans.

2. THEORETICAL BACKGROUND

This chapter provides the basic knowledge for pattern recognition and machine learning concepts that will be used throughout the thesis.

2.1 Pattern Recognition

The term *Pattern Recognition* refers to the analysis and interpretation of data through discrimination and classification encapsulating all the stages required such as problem formulation and data collection [55](see Fig. 2.1). In simplest words, it aims at finding regularities and patterns in data. Most often, it is used interchangeably with the term *Machine Learning* although they differ in certain aspects. However, there exist a sheer amount of overlap in what they cover, making it hard to distinguish the boundaries. They both origin from the artificial intelligence with pattern recognition mainly having a greater tendency to formalize and explain the findings (patterns). In general, a pattern refers to d-dimensional feature vector $\boldsymbol{x} = [x_1, \ldots, x_d]^T$ which corresponds to the measurements/observations for an object/sample. Hence, the set of measurements (observations) depend on the problem and the investigator. However, it is clear that the choice of features to use affect the performance of any pattern recognition system and must be conducted with care. The handcrafted feature engineering has been lately taken over by automatic feature learners in certain application areas such as object recognition [31, 37].

2.1.1 Learning Paradigms

In pattern recognition, approaches can be mainly categorized into two based on the availability of labels. These categories are named as *supervised learning* and

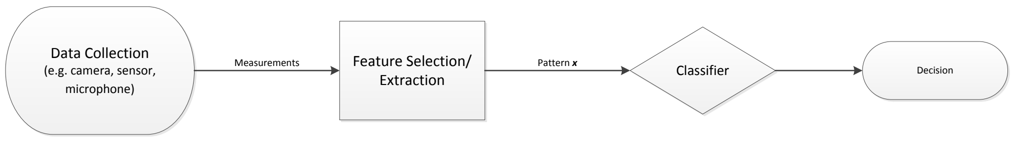

Data Collection
(e.g. camera, sensor, microphone)
Measurements
Feature Selection/ Extraction
Pattern x
Classifier
Decision

Figure 2.1 *Basic building blocks of a pattern classifier*

unsupervised learning. The labels are the correct class tag associated with each pattern $\boldsymbol{x}$ in the dataset.

Supervised learning algorithms are provided with a set of labeled data instances, $\mathcal{D}$,

$$\mathcal{D} = \big\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\big\}$$

where $(\boldsymbol{x}_i, y_i)$ is the ith pattern-label pair and N is the total number of patterns (samples). In general, it is referred to as training set. This mapping actually defines a function $\boldsymbol{h} : \mathcal{X} \to \mathcal{Y}$. The task for the learner is to determine an approximating function $\boldsymbol{g} : \mathcal{X} \to \mathcal{Y}$ which can produce a mapping from input space to category space as accurate as possible w.r.t. $\boldsymbol{h}$. The algorithms usually model the data sets based on the class characterization provided by labels and uses this model information to classify novel patterns by testing them against it. Supervised learning has found various applications via various algorithms such as k-nearest neighbors (KNN), support vector machines (SVM), decision trees and artificial neural networks (ANN). Although these algorithms have been proven to be very powerful tools for certain applications, they have particular disadvantages. First, collecting labels may be difficult. Especially, when the data volume is exceptionally large as in the 'Big Data', this process becomes extremely expensive. The second is that real world problems do not always contain discriminative labels. On the contrary, many uncertainties and ambiguities exist in them. These disadvantages should be taken into account as well as the advantages when designing a learning machine.

Unlike supervised learning, unsupervised learning aims at finding the underlying structure, i.e. the relationships between points in the dataset in the absence of labels. Hence, these type of algorithms do not have any guidance to measure the quality of their solutions (during modeling the relationships). In comparison with supervised learning schemes, unsupervised learning can be useful when labeling is not possible or difficult. These methods can be used as a preprocessing step for supervised algorithms as they can provide a *priori* information or prototypes to the learner. However, they may suffer to learn in complex cases since there is no supervision. Therefore, once again, the choice of the right approach is data-specific and it must take the nature of the problem under consideration. There exist many methods that fall into this category and clustering is one of the most commonly used techniques. The examples of common clustering methods can be listed as k-means, mean-shift, expectation-maximization etc.

The goal of any clustering algorithm is to discover sets of objects such that the objects in the same set exhibit similarities whereas objects from different sets (clusters, groups) show dissimilarities. In other words, the goal of clustering is to maximize intra-class similarities while keeping inter-class similarities as low as possible. There is a vast amount of literature on clustering [56]. Most of the early research was conducted in the fields of biology and zoology, although clustering methods have been utilized in many fields of science including signal processing and psychology. The clustering approaches can be categorized w.r.t. their characteristics as follows:

Hierarchical methods: These clustering procedures are generally used for data summarization based on a given dissimilarity matrix. The hierarchy is most often represented by a dendrogram which is a tree diagram. Breaking the dendrogram at a particular level generates a partition of the data set into disjoint groups. This approach is especially useful if one searches for an interactive analysis tool which allows to see the relationships in a datasets at various degrees of linkage. It has been used in many applications such as microarray gene data analysis [24].

Mixture Models: Each different cluster in the dataset is assumed to be generated from a distinct probability distribution. The distributions may be of the same distribution family (e.g. Gaussian) with different parameter set θ or they may be of different distributions families. The dataset is then assumed to be described by a finite mixture distribution of the form

$$p(\boldsymbol{x}) = \sum_{i=1}^{g} \pi_i p(x; \theta_i)$$

where π_i are the mixing proportions ($\sum_{i=1}^{g} \pi_i = 1$), g is the number of clusters and $p(x; \theta_i)$ is d-dimensional probability function with a parameter vector θ_i [55]. The main task is to estimate π_i, θ_i and g. Once they are estimated the clustering is achieved.

Sum-of-squares methods: These are often called as *partitioning* or *centroid-based clustering* methods. Their main task is to partition the space into g so that each subspace contains objects from one particular cluster. The methods vary depending on the choice of clustering criterion optimized. In practice, most of the methods find a local optima since the optimization problem is itself $\mathcal{NP}$-hard. The popular examples are k-means and fuzzy k-means [39].

Spectral clustering: These methods exploit the eigenvectors of a graph Laplacian to map the data into another space where the underlying structure is easier to capture [38].

2.1.2 Structure of a Pattern Recognition System

An unsupervised pattern classification system consists of the following functional blocks:

Preprocessing: This block deals with the raw data retrieved from the source. Basically, preprocessing might incorporate numerous signal processing techniques such as smoothing, trimming and filtering depending on the data type (e.g. text, audio, image etc.). It is a crucial step since the raw data may contain many inconsistencies and errors. Additionally, for most of the algorithms, the raw data is not suitable as input and hence it should be formatted into interpretable form by preprocessing.

Feature Extraction/Selection: Features are the relevant observations/measurements obtained from the phenomena which are under investigation. Extraction methods vary according to the problem (e.g. object-detection, shape recognition, document classification etc.). It is highly problem-oriented and usually requires sheer amount of efforts to design best-matching features for the given task. Feature selection refers to determining the most relevant subset of features which brings about the best representation, for instance, in terms of discrimination. For clustering and classification problems, one desires to have clusters/classes with the highest intra-class and the lowest inter-class similarity to be able to obtain better performance.

Training: In supervised learning, training is commonly used to refer to the initial phase where the training data set which contains the labels is utilized to build class models. These models are then used to classify novel inputs by testing them against the model or models (e.g. used in one-against-all classifiers).

Classification: This block realizes the purpose of the system such as clustering, regression or classification depending on the problem. A system can either be designed to assign categories for each sample or to gather together points that behave similarly as in clustering.

A classifier in general can produce two types of output, namely, labels and scores. The former is used generally to strictly classify the given novel samples into finite number of categories whereas the former can assign probabilities or scores which can show the likelihood of a given sample to belong to a certain category.

2.1.3 Evaluation Methods and Metrics

The evaluation of the performance of pattern recognition systems is as important as the system itself. It is essential to understand the capabilities of all the building blocks including preprocessing and feature extraction. In order to ensure that the performance of the classifier is stable, one should take into account approach-specific (supervised, unsupervised) and problem-specific properties.

In the field of machine learning, there exist certain standard evaluation metrics which are commonly used to describe different aspects of the systems. The most common forms are accuracy, precision and recall. However, it is hard to select according to which of all the existing measures one should rely on to determine the performance or make a comparison of classification methods. Fig. 2.2 illustrates the major measures in the context of anomaly detection.

- *Accuracy* describes the ratio of correct classification. It measures how many of samples are assigned to the correct classes and how many are assigned incorrectly. It is the most common form of performance metrics especially in machine learning. Although it tells about each class, it may fail in class-imbalanced situations as in anomaly detection. Formally, it is defined as follows.

  $$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

  where TP, TN, FP, FN stand for true positive, true negative, false positive and false negative, respectively.

- *Precision* indicates the rate with which the classifier returns the true positives. It is useful, however there is usually a significant trade-off between precision and recall. Therefore, it is generally not so descriptive of the performance alone. In Fig. 2.2, high precision corresponds to small number of samples from the normal class inside the sphere.

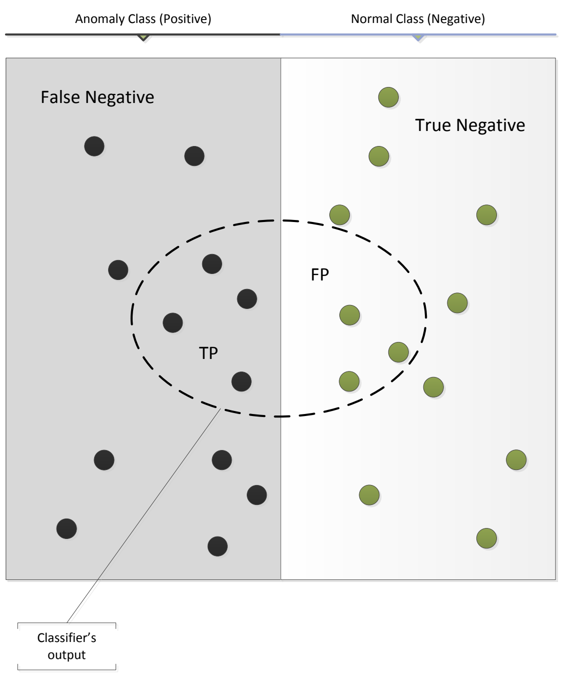
  $$Precision = \frac{TP}{TP + FP}$$

- *Recall* measures what percent of the relevant class can be retrieved by the classifier. Often it is considered together with precision since there exists a visible trade-off.

  $$Recall = \frac{TP}{TP + FN}$$

- *False Alarm Rate* measures how many of the irrelevant class samples are labeled as relevant. In anomaly detection, false alarm rate refers to the ratio between the number of incorrectly detected normals and total number of normals.

  $$False\ Alarm\ Rate = \frac{FP}{FP + TN}$$



	(True) Class-p	(True) Class-n
(Predicted) Class-p	True Positive	False Positive
(Predicted) Class-n	False Negative	True Negative

***Figure* 2.2** *Common evaluation metrics in the context of anomaly detection.*

- *Geometric Mean* is used to represent two different metrics simultaneously (e.g. accuracy and recall). In the scope of this thesis work, it is defined as follows.

$$Gmean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}}$$

Although these evaluation metrics can provide reasonable amount of information, one can opt for other methods which can be composite of two or more metrics such as Receiver Operating Characteristic (ROC) curves. ROCs can especially be useful in binary classification case when the classifier output is score-type. The graph is generated based on the threshold swept through various values. They can be useful when there exists a set of models, methods or parameters involved and a comparison is required. It proves to be more robust against class-imbalancies. Furthermore, the area under the curve (AUC) can also be a good indicator of the performance. However, it may suffer since these curves are not unique and multiple methods or parameters can result in the same AUC. This may make drawing a conclusion from them harder.

In supervised learning, there is a problem referred to as *overfitting* which results in low test accuracy (or high test error vice versa) while the training error is low. The problem is due to system's over-learning (memorizing) the training data such that making generalizations on the unseen data becomes harder and less accurate. In order to be able to measure system's performance in generalization, k-fold cross-validation is used. It simply divides the dataset into k subsets with equal size and uses only one subset at a time for testing while the remaining $k - 1$ sets are used for training. The results are calculated by averaging over all folds. In addition to fighting with overfitting, it can also give better conclusions for stochastic methods by considering the variations due to the random operations within. Hence, it is a common practice to cross-validate in machine learning.

2.2 Anomaly Detection

Anomaly detection refers to finding non-conforming patterns to normal behavior which is defined by the problem under consideration. These patterns which do not resemble the target (normal) behavior have found many terms such as anomalies, outliers, exceptions, surprises, pecularities and contaminants in many disciplines. However, the terms anomaly and outlier are most commonly used interchangeably. Anomaly detection has been a popular research topic and has great importance in many application domains such as fraud detection, intrusion detection, health care, fault detection and etc.

One of the main reasons why anomaly detection lies at the heart of many critical tasks is that these patterns can be translated into an actionable information immediately [13]. In a computer network, an anomaly may correspond to a hacked computer which leaks confidential information to unauthorized computers [34]. In credit card transaction data, anomalies can indicate possible fraud attempts [2]. Finally, in the context of this work, anomalies may correspond network elements such as an antenna which is malfunctioning or has broken down.

Anomaly detection faces many difficulties arising from the variations in definition of anomaly in the application domain or its abstract notion. In very simple words, it deals with finding the patterns that do not conform to the expected normal behavior. Following this notion, a very straight-forward method would attempt to determine the boundary around the normal points excluding the outliers. However, things might get complicated due to the following reasons listed in [13].

- Defining a very discriminative region for the normal behavior might be impractical since the boundary between normal and anomaly classes can be vague. A normal point very close to this boundary can be anomaly or vice versa.
- In case of malicious anomalies such as software viruses, defining the boundary becomes challenging since these malicious activities can disguise themselves as normal.

- The normal behavior might not be static, i.e. it can evolve in time, making the normal representation useless over time.
- The definition of anomaly varies from domain to domain. For instance, small variations in medical data can imply anomaly while similar fluctuations fall into normal behavior in weather forecasting. Hence, it becomes less straightforward to adapt an algorithm from another domain into the problem domain.
- The lack of labeled data for training is a major problem.

As a consequence of the given problems, anomaly detection becomes hard to tackle. Almost every problem brings their own formulation leading to hardships for methods to produce a generalized solution.

On top of the conceptual difficulties in definition of what is normal or not, anomalies show diversity in type making everything more complicated. There are three major types of anomalies to be considered when designing an anomaly detection system since the type (characteristic) of anomalies being targeted determines the approach to be developed.

- *Point anomalies* corresponds to data samples which qualify as anomaly with respect to the rest of the data set. It is the simplest type on which the most of the research has been focused. To illustrate, one can imagine a transaction of \$1000 as anomaly for a bank account which typically deals with relatively small amounts such as \$50-100.
- *Contextual (conditional) anomaly* refers to an anomalous behavior which is considered as an anomaly only in certain contexts and not in others. For example, a temperature recording, say 45°C can be considered as an anomaly during winter whereas it might be ordinary (normal) for a day during summer.
- *Collective anomalies* refers to collections of data samples which are anomalous altogether. For instance, in an electrocardiogram output, individual samples with low signal amplitudes may not represent anomalous behavior whereas a collection of similar samples in a consecutive manner can correspond to a collective anomaly.

2.2.1 Anomaly Detection in Unstructured System Log Files

The anomaly detection task in unstructured log files is to identify network components which signal unexpected signs such as malfunctioning and deactivation. The root-cause of a problem can be assumed to stem from the logical or physical neighborhood. Hence, in a real setting, the logs reported by physically or logically connected elements should reflect the situation ongoing in the subgraph (subnetwork). If one can localize the problem by the help of the symptoms broadcast via the logs, then one can get closer to understanding the root-cause.

Unstructured system logs constitutes a time-series signal in which individual lines (logs) do not imply anomalous behavior. The anomalies show themselves in a collective manner but not necessarily sequentially. Unlike host-based intrusion detection systems it is not possible to track sequences of logs to lead to a conclusion, however comparison of the sequences via histograms can bring about solutions.

The key challange in this domain is that the data is in streaming fashion leading to extremely large volumes as in intrusion detection. Therefore it requires efficient on-line analysis methods. Another problem encountered is the false alarm rate due to the large volume of the data. Even very small false alarm rates can cause trouble for the analyst. Hence, the system must be developed such that false alarms are reduced to acceptable levels.

2.2.2 Methods Used in Anomaly Detection

As explained earlier, the problem definition differs in different applications, so do the techniques used. In the literature, many have developed various methods according to the data type and the task. Table 2.1 lists some of the approaches and their applications in a compact way. Some of these techniques require labels (supervised) while some do not such as clustering-based techniques.

Method	References
Statistical Profiling using Histograms	[23, 19, 1, 4, 33]
Mixture of Models	[21, 28, 27]
Neural Networks	[30, 25, 26, 41, 44, 2, 51, 36]
Support Vector Machines	[40, 17, 48]
Clustering Based	[3, 50, 49, 9]
Nearest Neighbor Based	[20]
Bayesian Networks	[18]

Table 2.1 *Various works used in numerous anomaly detection applications*

2.3 Transformations

Sometimes, the available features may not serve well in terms of the resulting performances. This may happen when the true dimensionality of the data is lower than the dimensionality of the data representations, i.e. when the samples lie on a manifold. Dimensionality reduction may become extremely helpful in certain cases in which the algorithms can not model the normal data class against the outlier (anomaly) class well enough. These methods provide more compact and robust representations based on the original feature vectors. In this section, Principal Component Analysis (PCA), which is a baseline method for dimensionality reduction, and autoencoders, a special subcategory of neural networks, will be covered.

2.3.1 Principal Component Analysis (PCA)

Principal component analysis is one of the most widely used unsupervised dimensionality reduction methods. Although PCA, in the very basic terms, attempts to describe the data based on a new linear combination of its basis vectors, the ordering of the new basis vectors according to how *principal* they are makes it a viable method to reduce the dimensionality.

To formally express the idea, define $\mathbf{X}$, $\mathbf{Y}$ to be $m \times n$ matrices and $\mathbf{P}$ to be an $m \times m$ matrix. One can see matrix $\mathbf{X}$ as the original feature matrix while $\mathbf{Y}$ is the reexpression of $\mathbf{X}$ related by the linear transformation $\mathbf{P}$. Letting columns of $\mathbf{X}$ represent samples and rows represent features (attributes), one can express this transformation as

$$\mathbf{PX} = \mathbf{Y} \tag{2.1}$$

Let $\boldsymbol{p_i}$ be the ith row of $\mathbf{P}$ and let $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ be the ith column of $\mathbf{X}$ and $\mathbf{Y}$, respectively.

$$\mathbf{PX} = \begin{bmatrix} \boldsymbol{p_1} \\ \vdots \\ \boldsymbol{p_m} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \cdots \boldsymbol{x_n} \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} \boldsymbol{p_1} \cdot \boldsymbol{x_1} & \cdots & \boldsymbol{p_1} \cdot \boldsymbol{x_n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{p_m} \cdot \boldsymbol{x_1} & \cdots & \boldsymbol{p_m} \cdot \boldsymbol{x_n} \end{bmatrix}$$

Hence each element of $\boldsymbol{y_i}$ is the dot product of corresponding row of $\mathbf{P}$ and $\boldsymbol{x_i}$. If the Eq. 2.1 is considered as change of basis, then $\boldsymbol{y_i}$ is the projection of $\boldsymbol{x_i}$ onto the basis $\{\boldsymbol{p_1}, \cdots, \boldsymbol{p_m}\}$.

In order to determine the best re-expression for **X**, generally the covariance matrix is used. Covariance matrix is adequately descriptive to provide all the required information about the features and their interrelations. For better representation of the data, the noise and redundancies should be minimized or removed if possible. Redundancy here may imply that there exist features that are highly correlated and hence some of them can be expressed in terms of the others.

One way to decorrelate the data is to diagonalize the covariance matrix by eigenvector decomposition. However, to be able to use it properly, one should manipulate the dataset by extracting the mean from each feature so that it is in *mean deviation form*, i.e. zero mean. For the sake of clarity in the rest of the section, let $\boldsymbol{x_i}$ denote all the observations related to the ith feature (e.g. ith row of $\mathbf{X}$). Then,

$$Var[\boldsymbol{x_i}] = \boldsymbol{E}[(\boldsymbol{x_i} - \mu_i)^2] \tag{2.2}$$

where μ_i is the expectation of $\boldsymbol{x_i}$ and equal to zero. Therefore,

$$Var[\boldsymbol{x_i}] = \sigma_i^2 = \boldsymbol{E}[\boldsymbol{x_i}\boldsymbol{x_i}] \tag{2.3}$$

In the same spirit, one can show that covariance is equal to the following.

$$Cov[\boldsymbol{x_i},\ \boldsymbol{x_j}] = \sigma_{i,j}^2 = \boldsymbol{E}[\boldsymbol{x_i}\boldsymbol{x_j}] \tag{2.4}$$

There are two important properties of covariance which will be useful.

1. $\sigma_{i,j}^2 = 0$, when features i and j are completely uncorrelated.
2. $\sigma_{i,j}^2 = \sigma_i^2$, if $i = j$.

Since $\boldsymbol{x_i}$ is defined as row vector, one can calculate covariance with dot product where expectation is substituted by normalization.

$$Cov[\boldsymbol{x_i},\ \boldsymbol{x_j}] = \sigma_{i,j}^2 = \frac{1}{n-1}\boldsymbol{x_i}\boldsymbol{x_j^T} \tag{2.5}$$

Using Eq. 2.5, one can show that covariance matrix of $\mathbf{X}$ is

$$\boldsymbol{S}_{\mathbf{X}} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^{\boldsymbol{T}} \tag{2.6}$$

such that

$$\boldsymbol{S}_{\mathbf{X}} = \begin{bmatrix} \sigma^2_{1,1} & \cdots & \cdots & \cdots & \sigma^2_{1,n} \\ \sigma^2_{2,1} & \sigma^2_{2,2} & \cdots & \cdots & \sigma^2_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \sigma^2_{n-1,n-1} & \sigma^2_{n-1,n} \\ \sigma^2_{n,1} & \cdots & \cdots & \cdots & \sigma^2_{n,n} \end{bmatrix}$$

As $\sigma^2_{i,j} = \sigma^2_{j,i}$ by definition of covariance, $\boldsymbol{S}_{\mathbf{X}}$ is symmetric. The diagonal entries of $\boldsymbol{S}_{\mathbf{X}}$ are the variances while off-diagonals are the covariances. After this point, solution of PCA requires the eigenvector decomposition as mentioned earlier.

The task is to determine $\mathbf{P}$ so that $\boldsymbol{S}_{\mathbf{Y}}$ is diagonalized given $\mathbf{Y} = \mathbf{PX}$.

$$\begin{aligned} \boldsymbol{S}_{\mathbf{Y}} &= \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^{\boldsymbol{T}} && (2.7) \\ &= \frac{1}{n-1}(\mathbf{PX})(\mathbf{PX})^T && (2.8) \\ &= \frac{1}{n-1}(\mathbf{P}\mathbf{X}\mathbf{X}^{\boldsymbol{T}}\mathbf{P}^{\boldsymbol{T}}) && (2.9) \\ &= \frac{1}{n-1}\mathbf{P}(\mathbf{X}\mathbf{X}^{\boldsymbol{T}})\mathbf{P}^{\boldsymbol{T}} && (2.10) \\ &= \frac{1}{n-1}\mathbf{P}\mathbf{A}\mathbf{P}^{\boldsymbol{T}} && (2.11) \end{aligned}$$

where $\mathbf{A} = \mathbf{X}\mathbf{X}^{\boldsymbol{T}}$. With the help of the following theorems from linear algebra, one can proceed to the solution.

Theorem 1. *A matrix is symmetric if and only if it is orthogonally diagonalizable.*

Theorem 2. *A symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors.*

Theorem 3. *The inverse of an orthogonal matrix is its transpose.*

By Thm.2, symmetric matrix $\mathbf{A}$ can be diagonalized as

$$\mathbf{A} = \mathbf{E}\mathbf{D}\mathbf{E}^{\mathbf{T}} \qquad (2.12)$$

where $\mathbf{D}$ is the diagonal matrix and $\mathbf{E}$ is the matrix of eigenvectors. Choosing $\mathbf{E} =$

$\mathbf{P}^{\boldsymbol{T}}$ and substituting 2.12 into 2.11,

$$\boldsymbol{S}_{\mathbf{Y}} = \frac{1}{n-1}\mathbf{PAP}^{\boldsymbol{T}} \tag{2.13}$$
$$= \frac{1}{n-1}\mathbf{P}(\mathbf{P}^{\boldsymbol{T}}\mathbf{DP})\mathbf{P}^{\boldsymbol{T}} \tag{2.14}$$
$$= \frac{1}{n-1}(\mathbf{PP}^{\boldsymbol{T}})\mathbf{D}(\mathbf{PP}^{\boldsymbol{T}}) \tag{2.15}$$
$$= \frac{1}{n-1}(\mathbf{PP}^{\mathbf{-1}})\mathbf{D}(\mathbf{PP}^{\mathbf{-1}}) \tag{2.16}$$
$$= \frac{1}{n-1}\mathbf{D} \tag{2.17}$$

$\mathbf{P}^{\mathbf{-1}} = \mathbf{P}^{\boldsymbol{T}}$ in 2.16 is the result of Thm.3. From 2.17 it is clear that PCA diagonalizes $\boldsymbol{S}_{\mathbf{Y}}$ given $\mathbf{P}$'s rows are eigenvectors of $\mathbf{XX}^{\boldsymbol{T}}$. Since the eigenvalues of $\mathbf{D}$ are sorted w.r.t. their magnitudes, one can use this information to reduce dimensionality. The eigenvectors corresponding to smaller eigenvalues can be omitted without much loss (loss will be proportional to the variance they account for). This will lead to $\mathbf{Y}$ with $\tilde{m}$ dimensions such that $\tilde{m} < m$.

In this thesis work, PCA is generally applied before the decision-making algorithms in order to obtain more robust results. Using the principal components that correspond to the largest variances brings about more insight about the data since dynamics of the phenomenon reveals itself in those directions.

2.3.2 Autoencoders

Autoencoders are a subcategory of (feed-forward) artificial neural networks (ANN) which possesses auto-association property. It is an unsupervised learning algorithm trained by using backpropagation method [45]. Autoencoders usually have a hidden layer which has less number of neurons compared to visible layers. The main goal of this particular type of networks is to learn how to reconstruct the data from a lower dimensional space representation. Before explaining in further detail, it is useful to define fundamental building blocks of an ANN.

Neuron

A neuron is the elementary computing unit in neural networks and hence in autoencoders. It is inspired from the biological structures in the brain which are responsible for learning. The computational counterpart of the biological neurons consists of synapse-like components and a body. The synapses have synaptic weights or weights which determine the output of a neuron. The neuron computes the product $W^T\boldsymbol{x}$ where $\boldsymbol{x} \in \Re^n$. However, the output of a neuron is usually determined by a

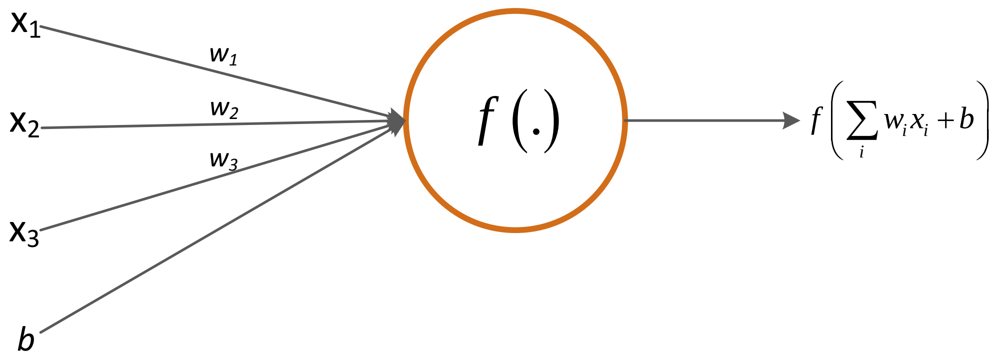


***Figure** 2.3 Neuron: Elementary unit of Artificial Neural Networks (ANN)*

$$f(z) = \begin{cases} \dfrac{1}{1+e^{-z}} & \cdots \text{ Sigmoid} \\ \\ \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}} & \cdots \text{ Tanh} \end{cases}$$

***Figure** 2.4 Commonly used activation functions*

nonlinear *activation function* $f(.)$ such as sigmoid or tangent-hyperbolic (tanh) as depicted in Fig. 2.3 and Fig. 2.4.

Autoencoder

Autoencoders are unsupervised in the sense that labels are not required because the target is the input itself. Due to the shape of configuration as in Fig. 2.5, autoencoder's task becomes to learn a good representation in a lower dimension imposed by the number of neurons in the hidden layer (bottleneck) so that the reconstruction error at the output is small. Therefore, if the training is successful the autoencoder discovers a new set of features in the bottleneck. In other words, autoencoder tries to learn the identity function which seems trivial unless there are constraints on the size of hidden layer (e.g., $<$ visible layer size).

As a member of feed-forward neural network family, autoencoder has two modes of operation, namely, forward and back propagation. In the former, the input is fed from the input layer and the activations of each neuron are calculated layer-by-layer. The error is calculated at the output and backpropagated. During backpropagation the weights are modified by amounts they account for in the error. In this thesis work, a special type of error function which sets constraints on the activation levels in the hidden layer is used (sparsity condition). In order to explain the difference between the squared error objective function and the one with the sparsity constraint, denote the dataset as $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\right\}$ where $(x^{(i)}, y^{(i)})$ is the ith sample

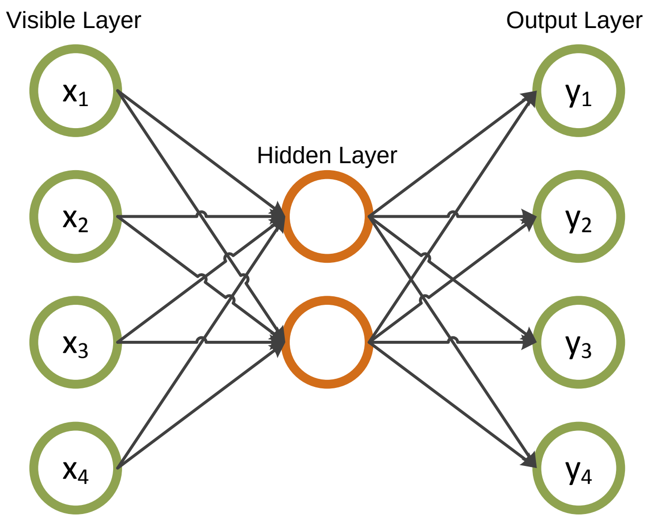


Figure 2.5 *Example autoencoder configuration*

and its target. Also, denote the hypothesis function the autoencoder learns given the weights W and biases b as $h_{W,b}(\boldsymbol{x})$. Then, the squared error cost function becomes [42]:

$$J(W,b) = \frac{1}{n}\sum_{i=1}^{n} J(W,b;x^{(i)},y^{(i)}) + \frac{\lambda}{2}\sum_{l=1}^{l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2 \tag{2.18}$$

$$J(W,b) = \frac{1}{n}\sum_{i=1}^{n} \mid\mid h_{W,b}(x^{(i)}) - y^{(i)} \mid\mid^2 + \frac{\lambda}{2}\sum_{l=1}^{l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2 \tag{2.19}$$

where the second term is the weight regularization which makes the network favor smaller weights. The weight decay helps prevent from overfitting. The variable s_l is the size of lth layer. The parameter λ controls the trade-off between the squared error and the weight decay.

In order to impose the sparsity condition, one should denote the activation of the jth neuron due to input pattern $x^{(i)}$ in the hidden layer h with $a_j^{(h)}(x^{(i)})$. Then, it is desired to limit the average activation by the *sparsity parameter*, ρ. The average activation for the jth neuron in the hidden layer in Fig. 2.5 can be calculated as:

$$\hat{\rho}_j = \frac{1}{n}\sum_{i=1}^{n} a_j^{(2)}(x^{(i)}) \tag{2.20}$$

Then, this can be inserted into the cost function using the Kullback-Leibler (KL) divergence, KL($\rho \mid\mid \hat{\rho}_j$) which is given by:

$$KL(\rho \mid\mid \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{2.21}$$

Finally, the cost (objective) function to be minimized becomes:

$$J_{sparse}(W,b) = J(W,b) + \beta \sum_{j=1}^{s_2} KL(\rho \mid\mid \hat{\rho}_j) \tag{2.22}$$

where β controls the influence of the sparsity constraint.

The autoencoder using this type of $J_{sparse}(W,b)$ are referred to as sparse autoencoders.

Training of Autoencoders

As in the same fashion with the feed-forward type ANNs, sparse autoencoders can be trained with backpropagation [45]. In the beginning of the training network weights are initialized to small random values. However, the initialization determines the results as gradient-descent or its variants used in backpropagation can get stuck in undesired local minima depending on the initial values. Therefore, pre-training is highly recommended. The pretraining strageties usually involve unsupervised learning approaches. In this way, the network weights are adapted to the properties of the training data. Supervised fine-tuning on top of such an unsupervised learning approach can be used to increase discrimination.

Also, in order to increase the training speed, the inputs should be normalized. This fact results from the partial derivatives which drive the optimization task. When the inputs are large then the derivative of the non-linear function (e.g, sigmoid) becomes too small and it takes more time to converge to the minima.

3. METHODOLOGY

In this chapter, the algorithms which have been used during the experiments and the details of their implementation are given in an elaborate manner. Throughout this chapter, the following steps will be covered respectively: data pre-processing, feature extraction, transformations and classification.

3.1 Data Preprocessing

Most of the pattern recognition/machine learning algorithms usually employ vectors of numerical values, symbols and indicator variables in order to produce a solution for the problem. Therefore, whenever the raw data are not suitable to be immediately fed into the classifier/clustering algorithm, the preprocessing need arises.

In problems related to textual data, the general approach is to form dictionary for the most common or relevant words. Hence the preprocessing mainly deals with shaping of the log data to generate a good dictionary. In order for the feature extraction to output powerful representations, the rows (lines) of the system log file should be processed so that certain classes of words or other elements (e.g. ID numbers of network components) are discarded. The point is to extract words that bear useful information.

Based on different word selection rules, one can obtain various representations for the data. The purpose of different preprocessing methods is to determine the most robust representation leading to a better performance in the decision stages. In the following subsections legal word selection rules utilized are described in algorithmic fashion.

3.1.1 Legal Word Selection Rule Set-1

This procedure ignores any word containing a character which is not an element of the English alphabet. This process is described in Algorithm 1. The features generated using this selection rule will be referred to as *BoL* in the following chapters.

```
Data: Log file
Result: Set of legal words
initialization;
$legalWords \longleftarrow \emptyset$;
while not at the end of file do
    read current line;
    $S \longleftarrow \emptyset$;
    if not empty then
        discard the time stamp;
        discard all the words containing a character which is not in the English
        alphabet ;
        $S \longleftarrow$ eligible words;
        while $S \neq \emptyset$ do
            if $word \in legalWords$ then
                do nothing;
            else
                $legalWords \longleftarrow word$;
            end
            $S \longleftarrow S - word$;
        end
    else
        go to next line;
    end
end
```

Algorithm 1: Word Selection Rule Set-1

3.1.2 Legal Word Selection Rule Set-2

Unlike Legal Word Selection Rule Set-1, this set of rules eliminates only very specific type of words that occurs frequently. These words usually indicate the IDs of network elements which do not necessarily help but increase the size of dictionary.The steps followed for this process is described in Algorithm 2. The features generated using this selection rule will be referred to as *BoL2* in the following chapters.

```
Data: Log file
Result: Set of legal words
initialization;
$legalWords \longleftarrow \emptyset$;
while not at the end of file do
    read current line;
    $S \longleftarrow \emptyset$;
    if not empty then
        discard the time stamp;
        discard the words containing '#EID' and/or 'eNodeB';
        $S \longleftarrow$ eligible words;
        while $S \neq \emptyset$ do
            if $word \in legalWords$ then
                do nothing;
            else
                $legalWords \longleftarrow word$;
            end
            $S \longleftarrow S - word$;
        end
    else
        go to next line;
    end
end
```

Algorithm 2: Word Selection Rule Set-2

3.2 Feature Extraction

In this section, the details of the feature types and the way they are generated from the data are described.

3.2.1 Windowing

In time-series data analysis, one of the common practices is to partition the data into smaller chunks which are usually called *windows*. Since the system logs are generated with respect to time, it will be considered as a multi-dimensional time-series signal. Therefore, after the preprocessing step a windowing operation is applied on the logs. At this point, it is reasonable to present two different types of windows as follows:

1. *Time windows:* These windows span certain time frame and are likely to include distinct number of lines in each window.
2. *Fixed-length windows:* These windows do not take time frame into account, however contain equal number of lines in every window.

Windowing is done such that the consecutive windows overlap by a rate which is referred to as *overlap ratio* in general. However, since the experiments did not yield any improvement based on the overlap ratio, we will use an overlap ratio of 33% in the rest of the thesis.

Throughout the work, features are extracted using the second type (i.e. fixed-length) windows. This is due to the fact that we observed no significant benefit is obtained from time windows in the absence of labels.

3.2.2 Feature Descriptions

In tasks such as document classification, the most common practice is to form the histogram of the data given a dictionary. Feature extraction basically means to count the occurrences of each dictionary element in a given window. Hence, the feature vector is the histogram of the window and the size of the feature vector is equal to the size of the dictionary.

In this thesis, two types of features are investigated as listed below.

1. *Bag-of-words (BoW):* This approach uses the dictionary of words which qualify as legal. In the dictionary, each entry is unique. In the datasets provided by Tieto, there exist 285 and 301 words in the BoWs, respectively.
2. *Bag-of-lines (BoL):* This approach uses the dictionary of unique lines which constitute of the elements of the legal word set. In the aforementioned datasets, the BoL are of sizes 90 and 98, respectively (if generated using Algorithm 1).

The bag-of-lines are generated by concatenating the legal words to form a string which is unique. It differs from the traditional bag-of-words approach in the sense that it takes into account the order of the words. We should note that this type of representation may not be useful for other applications on textual data. The reason is that it is based on the assumption that the log information broadcast by different machines in the network belong to a certain (shared) set of messages. In other words, the log generation can be imagined as a random process which has a finite sample space. This assumption makes this type of features not easily generalizable.

3.3 Algorithms

In this section, the methods which have been investigated for anomaly detection are discussed in detail.

3.3.1 A Naive Bayesian Approach (NB)

Naive Bayesian classification scheme is one of the most popular approaches used in text classification applications such as spam filtering. The advantage of this method is that it is relatively simple to implement and its efficiency (training and test require one pass over the data). It has been shown that it works effectively in many applications including anti-spam filtering and text categorization [5, 46].

In Naive Bayes classifiers, the data is classified using conditional probabilities. It exploits the Bayes' Theory to compute the class conditional probabilities linked to each instance. The *naivety* in the name comes from the assumption that the attributes of the instances are independent.

Let $\boldsymbol{x} = [x_1, x_2, x_3, \ldots, x_N]^T$ represent the sample (the evidence in Bayesian terms) where x_i is the attribute (feature) i and let $c \in \mathcal{C}$, where $\mathcal{C}$ is the set of all possible classes, represent the classes. Also, let H denote a hypothesis, e.g. that $\boldsymbol{x}$ belongs to *normal* class. Then, Bayes Theorem states the following.

$$\Pr(H \mid \boldsymbol{x}) = \frac{\Pr(\boldsymbol{x} \mid H)\Pr(H)}{\Pr(\boldsymbol{x})} \tag{3.1}$$

In the Bayesian context, the terms in 3.1 are interpreted as:

- $\Pr(H)$ is the *prior* probability which indicates the initial degree of belief in the hypothesis.
- $\Pr(\boldsymbol{x} \mid H)$ is the *conditional* probability which denotes the likelihood of observing sample $\boldsymbol{x}$ given the hypothesis H.

In traditional Naive Bayesian settings, a joint model for the N-dimensional attribute (feature) vector and the class c is built as follows.

$$\Pr(\boldsymbol{x}, c) = \Pr(c)\prod_{i=1}^{N}\Pr(x_i \mid c) \tag{3.2}$$

For classification problems, the task is reduced to calculate $\Pr(H \mid \boldsymbol{x}^*)$ which denotes the probability of satisfying the hypothesis H given the observation $\boldsymbol{x}^*$, i.e., the probability of $\boldsymbol{x}^*$ belonging to class c. Hence the conditional probability can be rewritten as $\Pr(c \mid \boldsymbol{x}^*)$ for the sake of formula convention. Now, by using Theorem (3.1) combined with the Total Probability Theorem which is given in (3.3):

$$\Pr(\boldsymbol{x}) = \Pr(\boldsymbol{x} \mid c_1)\Pr(c_1) + \ldots + \Pr(\boldsymbol{x} \mid c_k)\Pr(c_k) \tag{3.3}$$

will lead to

$$\Pr(c \mid \boldsymbol{x}^*) = \frac{\Pr(\boldsymbol{x}^* \mid c)\Pr(c)}{\Pr(\boldsymbol{x}^*)} = \frac{\Pr(\boldsymbol{x}^* \mid c)\Pr(c)}{\sum_c \Pr(\boldsymbol{x}^* \mid c)\Pr(c)} \tag{3.4}$$

where c takes the values in the set of classes $\mathcal{C}$. The anomaly detection problem in this work basically involves two classes (i.e. normal and anomaly), two class scheme is to be assumed throughout this method. The conditional probability in (3.4) will assign a score of how likely it is for a novel test sample to belong to the class c when classification is realized.

Parameter Estimation using Maximum Likelihood (ML)

Let $\mathcal{D}$ denote the dataset such that $\mathcal{D} = \left\{(\boldsymbol{x}^k, c^k), k = 1, \ldots, M\right\}$ with $\boldsymbol{x}$ having binary features $x_i^k \in \{0, 1\}, i = 1, \ldots, n$. c^k is the class label for the kth instance. For the sake of simplicity of the remaining steps in the Maximum Likelihood procedure, θ_i^c will be used to represent $\Pr(x_i = 1 \mid c)$ as in [8, pp. 234] . After this point, it is sufficient to use $1 - \theta_i^c$ to denote $\Pr(x_i = 0 \mid c)$ as axioms of probability assert normalization.

Using the independence assumption, the conditional probability $\Pr(\boldsymbol{x} \mid c)$ can be rewritten as:

$$\begin{aligned}\Pr(\boldsymbol{x} \mid c) = \Pr(x_1, x_2, \ldots, x_N \mid c) &= \prod_{i=1}^{N} \Pr(x_i \mid c) && (3.5)\\ &= \prod_{i=1}^{N} (\theta_i^c)^{x_i}(1 - \theta_i^c)^{1-x_i} && (3.6)\end{aligned}$$

Since the attributes are assumed to be binary for simplicity, x_i can only take values 1 or 0 leading to either θ_i^c or $1 - \theta_i^c$ in the product in 3.6. Let's also denote the number of samples in class-0 and class-1 by k_0 and k_1, respectively. Following these, the log-likelihood of the attributes and class labels becomes:

$$\mathcal{L} = \sum_k \log \Pr(\boldsymbol{x}^k, c^k) = \sum_k \log \Pr(c^k) \prod_n \Pr(x_n^k \mid c^k) \tag{3.7}$$

$$= \left\{ \sum_k \sum_n x_n^k \log \theta_i^{c^k} + (1 - x_n^k) \log(1 - \theta_i^{c^k}) \right\} + k_0 \log \Pr(c = 1) + k_1 \log \Pr(c = 2) \tag{3.8}$$

To extract the parameters from the likelihood expression, one should set the derivative with respect to θ_i^c equal to zero. The resulting parameters are as follows.

$$\theta_i^c = \Pr(x_i = 1 \mid c) = \frac{\text{number of appereance of } x_i = 1 \text{ in class c}}{\text{number of samples in class c}} \tag{3.9}$$

By setting the derivative of $\mathcal{L}$ w.r.t. class probability $\Pr(c)$ to zero, we obtain

$$\Pr(c) = \frac{\text{number of samples in class c}}{\text{number of samples in whole data set}} \tag{3.10}$$

Now, one has all the required probabilities for the classification task which will be explained in the next chapter.

Classification of Novel Samples

Classification of novel samples by using the model learnt in training phase is straightforward. An input $\boldsymbol{x}^*$ is classified based on the conditional probabilities.

$$\text{c=} \begin{cases} 1, & \text{if } \Pr(c = 1 \mid \boldsymbol{x}^*) > \Pr(c = 0 \mid \boldsymbol{x}^*) \\ 0, & \text{otherwise} \end{cases}$$

The above expression $\Pr(c = 1 \mid \boldsymbol{x}^*) > \Pr(c = 0 \mid \boldsymbol{x}^*)$ can be rewritten using Bayes' Rule. It can be shown that it becomes

$$\frac{\Pr(\boldsymbol{x}^* \mid c = 1) \Pr(c = 1)}{\Pr(\boldsymbol{x}^*)} > \frac{\Pr(\boldsymbol{x}^* \mid c = 0) \Pr(c = 0)}{\Pr(\boldsymbol{x}^*)} \tag{3.11}$$

One can omit $\Pr(\boldsymbol{x}^*)$ in the denominators easily. Then, taking the logarithm of the both sides of the inequality, one obtains

$$\log \Pr(\boldsymbol{x}^* \mid c = 1) + \log \Pr(c = 1) > \log \Pr(\boldsymbol{x}^* \mid c = 0) + \log \Pr(c = 0) \tag{3.12}$$

Up to this point, the method has been developed under the assumption that there exist labels, c. However, in order to be able to use Naive Bayes in unsupervised learning problems where labels are not available there are a few more steps to take.

The approach here is to adopt Naive Bayesian idea into a scoring function. This scoring function will then be utilized to assign an anomaly score to each point in the data set. The higher the score is, the higher the probability that sample belongs to the normal class is. Let us define this function asserting the following assumptions on the problem.

1. The number of samples in anomaly (outlier) class is more likely to be much less than the number of samples in the normal class.
2. The attributes are generated from a set of Gaussian distributions, i.e. $x_n \sim \mathcal{N}(\mu_n, \sigma_n)$.

The first assumption implies that we can model the normal class using the entire dataset. Modeling can be done by ML as stated earlier. Here, the trick is to treat every element in the data set as if they show normal behavior so that we can omit the *prior*s (i.e. $\Pr(c)$), whose computation is not possible in the absence of labels.

Following the second assumption that each attribute is generated from an unknown normal distribution, we can model θ_i^cs as univariate distributions given the ML parameters. Furthermore, this assumption helps us to break the binary attribute condition.

Finally, the score function is ready to be formulated. Let $\boldsymbol{x}^*$ be a n-dimensional sample from the data set. The following function will assign this sample a *normality* score based on 3.12 the first assumption.

$$\mathcal{F}(\boldsymbol{x}^*) = \log \Pr(\boldsymbol{x}^* \mid c = \text{Normal}) = \sum_{i=1}^{n} \log \Pr(x_i \mid c = \text{Normal}) \tag{3.13}$$

Clustering based on Normality Scores

The algorithm terminates after the model is learnt and samples are associated with a normality score which indicate the degree they belong to the normal class. The instances which are likely to be anomalies are determined by setting a threshold $\mathcal{T}_{normal}$ such that

$$c = \begin{cases} \mathit{normal}, & \text{if } \mathcal{F}(\boldsymbol{x}^*) > \mathcal{T}_{normal} \\ \mathit{anomaly}, & \text{otherwise} \end{cases}$$

The selection of the optimal threshold is determined through experimentation. It will be given in detail in the next chapter.

3.3.2 Density-based Approach

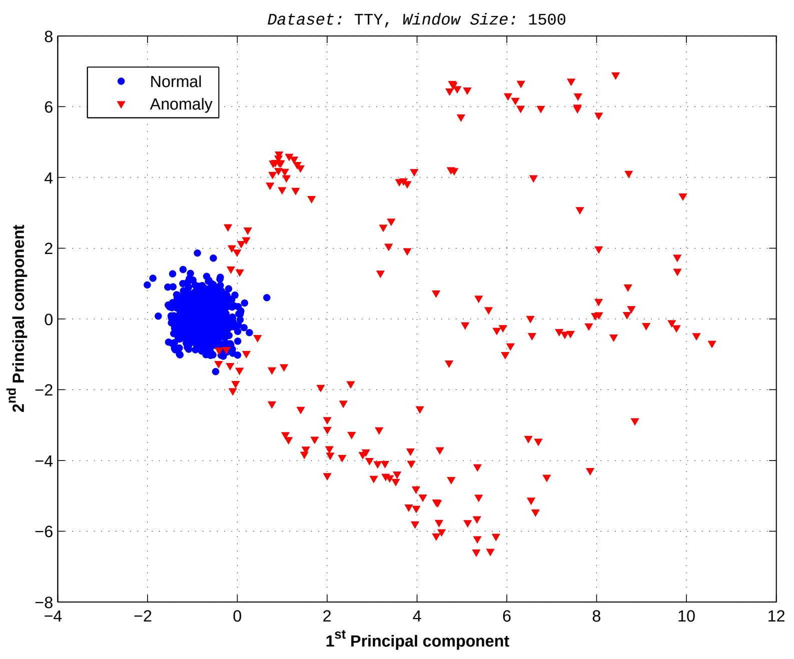
Applying clustering algorithms on a data which has a varying distribution in terms of density of points in the multidimensional space is not as straightforward as it seems. Due to the nature of the problem, traditional algorithms such as *K-means* which partition the multidimensional space into subregions may not give desired output (result). The reason is that they, in certain ways, force every data instance to belong to a certain cluster. However, the fact that the anomalies usually are not many in the population eventually leads them to be assigned to the closest cluster centroid. This is not desirable in the case of anomaly detection. Hence, one needs to approach the problem from different perspectives. The approach used in this work stems from the following assumptions on outliers (anomalies).

Assumption 1. *Normal data instances are expected to lie closer to the cluster centroid which is the closest.*

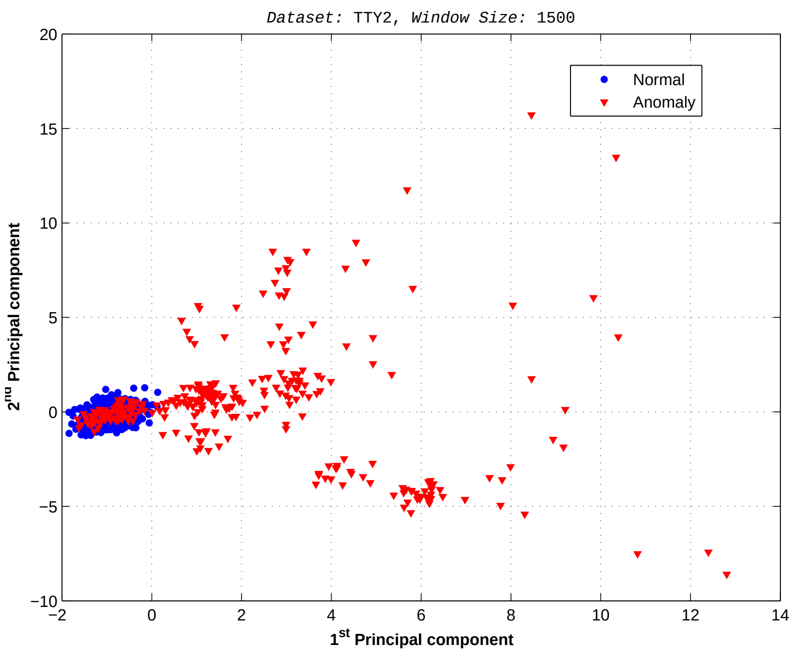
Assumption 2. *It is assumed that normal data points form a larger, more compact (denser), cluster whereas anomalies either group in smaller clusters or form sparser clusters.*

Studying the real data in telecommunicaiton networks has shown that the anomalies constitute a considerable amount of the population and they exhibit a group behavior as can be seen in Fig. 3.1. Therefore, the assumptions given above are valid for this problem.

Since being anomaly is not a binary property in this context, a *score* must be associated with each sample. This score should measure the likelihood that any given sample belongs to the anomaly class. Many researchers have proposed density measures to account for the *outlierness* such as LOF and OPTICS-OF [10, 11]. However, these methods generally focus on local outliers and may not achieve results as good as desired since the datasets in this thesis contain a large number of anomaly points which tend to group together. Hence, a density measure which is more robust and simpler must be devised and used.



(a) PCA on TTY dataset for visual inspection



(b) PCA on TTY2 dataset for visual inspection

Figure 3.1 *Visualisation of the datasets in 2 dimensional space determined by applying PCA helps to understand the problem better. Figures 3.1(a) and 3.1(b) shows that the normal samples lie in a higher density region while anomalies reside in a sparser region.*

Before going into the details of the algorithm, let us denote the dataset with $\mathcal{D} = \{\boldsymbol{x}_k, k = 1, \ldots, M\}$ slightly different than the previous subsection 3.3.1 for convenience. Then, let us define the local density of a given point in the dataset $\mathcal{D}$.

Definition 3.3.1. Let d_{ij} be the Euclidean distance between sample x_i and sample x_j, where $i, j \in \{1, 2, \ldots, M\}$. From the symmetry of Euclidean distance function, $d_{ij} = d_{ji}$. Also, let $\mathcal{P}_i$ denote the set whose elements are the k-nearest neighbors of sample x_i. Then, the radius of the smallest hypersphere surrounding elements of $\mathcal{P}_i$ is given by $R_i = \max\{d_{ij} \mid j \in \mathcal{P}_i\}$. Finally, the local density for the sample x_i is defined by:

$$\begin{aligned}\textbf{Local Density} = \mathrm{LD}(\boldsymbol{x_i}) &= \frac{k}{\text{Volume of the hypersphere}} \\ &\approx \frac{k}{R_i^2}\end{aligned}$$

In the above Def. 3.3.1, the volume of the hypersphere is approximated by R_i^2 assuming a regular sphere as this normalization factor is to be used in the same order for all samples in the dataset. Finally, the *outlierness* (normality) score can be defined as follows.

Definition 3.3.2. Given the local density of a sample, the density outlier score is given as

$$\begin{aligned}\textbf{Density Outlier Score} &= \mathrm{DOS}(\boldsymbol{x_i}) \\ &= \frac{1}{\mathrm{LD}(\boldsymbol{x_i})}\end{aligned}$$

Since outliers lie in a larger sphere as their pairwise distances are relatively larger, their local densities are smaller. As a consequence, their density outlier score (DOS) gets larger compared to normal samples. However, density outlier score may suffer in identifying certain anomalous objects even if they are clearly behaving oddly. This situation may arise when anomalies tend to cumulate around a certain point in space, i.e. when they form clusters themselves. Although Assumption 2 takes into account these phenomena, if the density of these clusters gets similar to those of normal samples, it breaks. Therefore, one must include another term in the final outlier score in order to identify those ones correctly. Using Assumption 1, those objects located far away (relatively) w.r.t. the mean can be penalized. Hence, the final outlier score becomes:

$$\begin{aligned}\textbf{Outlier Score} &= \mathrm{OS}(\boldsymbol{x_i}) \\ &= \text{Density Outlier Score}(\boldsymbol{x_i}) + \beta \; \textit{Distance to Mean} \\ &= \frac{1}{\mathrm{LD}(\boldsymbol{x_i})} + \beta \times d_{i\mu}\end{aligned}$$

where $d_{i\mu}$ is the euclidean distance of object (sample) x_i to the mean ($\boldsymbol{\mu}$) of the dataset. The variable β is controlling the penalization. Setting $\beta = 0$ would imply that the anomality is solely based on the local density whereas any nonzero β implies that both assumptions are taken into account simultaneously.

Clustering based on Outlier Scores
Similar to the Naive Bayesian approach, clustering is conducted by simple thresholding after every data instance is associated with an *outlier score*. Let us denote the threshold according to which the instances are categorized as $\mathcal{T}_{outlier}$ such that

$$c = \begin{cases} \textit{anomaly}, & \text{if } OS(\boldsymbol{x_i}) > \mathcal{T}_{outlier} \\ \textit{normal}, & \text{otherwise} \end{cases}$$

The selection of the optimal threshold is again determined through experimentation. It will be given in detail in the next chapter.

3.3.3 Ordering Points to Identify the Clustering Structure (OPTICS)-based Approach

In this section, another density-based clustering method OPTICS [6] will be investigated as well as its anomaly detection variant OPTICS-OF [10]. OPTICS simply uses the definitions of anomalies to generate an ordering of the samples in a dataset (based on their density characteristics) rather than clustering it explicitly. In the context of this thesis, OPTICS is to be used with its variant OPTICS-OF to produce an anomaly score as have been done in the previous sections. In the way to construct a metric to evaluate samples, one should consider the definitions to be given step-by-step. These definitions are usually used for density-based clustering approaches to identify samples' characteristics relative to their neighborhoods in high-dimensional feature space.

Definition 3.3.3. (ϵ-neighborhood)
The ϵ-neighborhood defines a set of samples which lie within a (hyper)sphere of radius ϵ.

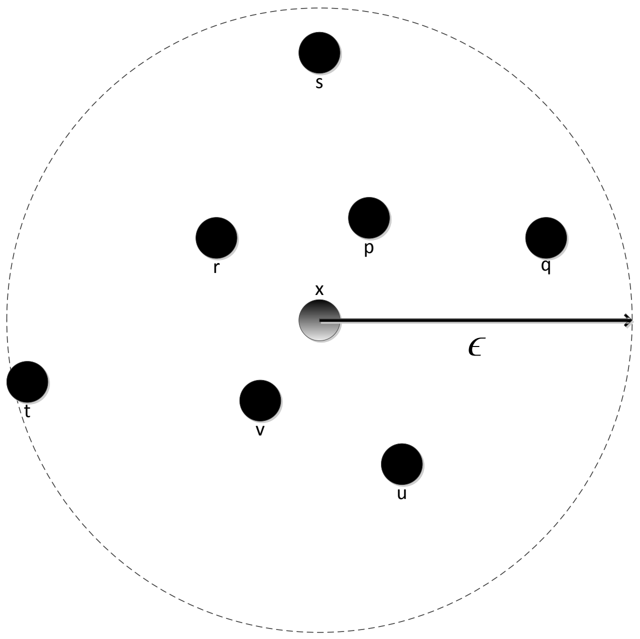


Figure 3.2 *The ϵ-neighborhood ($N_\epsilon(\boldsymbol{x})$) constitutes of the neighbors, $\boldsymbol{x}_i$, of $\boldsymbol{x}$ which satisfy $\|\boldsymbol{x} - \boldsymbol{x}_i\| \leq \epsilon$. In this figure, $\boldsymbol{x}_i \in N_\epsilon(\boldsymbol{x}) = \{p, q, r, s, t, u, v, x\}$. The sample $\boldsymbol{x}$ is by definition in this set.*

Definition 3.3.4. (directly-density-reachable)
A sample $\boldsymbol{x}$ is said to be density-reachable from another sample $\boldsymbol{q}$ in dataset $\mathcal{D}$ given an ϵ and *MinPts* if

- $\boldsymbol{x} \in N_\epsilon(\boldsymbol{q})$, where $N_\epsilon(\boldsymbol{q})$ is the set of samples in ϵ-neighborhood of q.

  and

- $\mid N_\epsilon(\boldsymbol{q}) \mid \geq$ *MinPts*. This is referred to as *core object condition.*

Definition 3.3.5. (Density-reachable)
A sample $\boldsymbol{x}$ is density-reachable from another sample $\boldsymbol{q}$ in dataset $\mathcal{D}$ given an ϵ and *MinPts* if there exists a chain of samples $\boldsymbol{x}_1 = \boldsymbol{x}, \ldots, \boldsymbol{x}_n = q$ such that every x_{i+1} is directly-density-reachable from x_i. To note, density reachability is not necessarily symmetric and only those core samples can have this symmetry.

Definition 3.3.6. (Density-connected)
A sample $\boldsymbol{x}$ is density-connected from another sample $\boldsymbol{q}$ in dataset $\mathcal{D}$ given an ϵ and *MinPts* if there exists a sample $\boldsymbol{o}$ such that $\boldsymbol{x}$ and $\boldsymbol{q}$ are density-reachable from it. Unlike density-reachability, densitiy connectedness is symmetric.

The above definitions can be used to conduct density-based clustering as DBSCAN does [22]. A density*cluster* consists of density connected samples. Any sample in

$\mathcal{D}$ that does not belong to any of the existing clusters is considered as *noise* (not necessarily anomaly).

Although the idea in DBSCAN sorts out the clustering problem when there is noise in the dataset, it relies on the choice of parameters ϵ, *MinPts* which in return affect the performance of the operation. OPTICS aims at going around this pitfall by considering multiple density clusters w.r.t. multiple parameters simultaneously. In order to be able to achieve this, it follows an *order* by which the clusters are expanded. This order imposes to choose the density-reachable object w.r.t. the smallest ϵ value. Consequently, the higher density clusters are handled first. While forming this ordering, *core-distance* and *reachability-distance* are stored and they will be utilized in the anomaly detection.

Definition 3.3.7. (Core-distance)
A sample $\boldsymbol{x}$'s core distance is defined as follows given ϵ, *MinPts*.

$$CD(\boldsymbol{x}) = \begin{cases} \text{UNDEFINED}, & if \mid N_\epsilon(\boldsymbol{x}) \mid < MinPts \\ MinPts - distance(\boldsymbol{x}), & otherwise \end{cases}$$

where *MinPts-distance*$(\boldsymbol{x})$ is defined as the distance between $\boldsymbol{x}$ and its *MinPts*' ϵ-neighbor. The core-distance can be interpreted as the minimum distance ($\bar{\epsilon}$) between sample $\boldsymbol{x}$ and $\boldsymbol{q} \in N_\epsilon(\boldsymbol{x})$ such that $\boldsymbol{x}$ becomes a core object w.r.t. $\bar{\epsilon}$ when $\boldsymbol{x}$ satisfies *core object condition* (see Def.3.3.4). In Fig. 3.3, core-distance notion is illustrated.

Definition 3.3.8. (Reachability-distance)
The reachability distance of sample $\boldsymbol{q}$ w.r.t. another sample $\boldsymbol{x}$ is defined to be:

$$RD(\boldsymbol{q}, \boldsymbol{x}) = \begin{cases} \text{UNDEFINED}, & if \mid N_\epsilon(\boldsymbol{x}) \mid < MinPts \\ \max(CD(\boldsymbol{x}),\ distance(\boldsymbol{q}, \boldsymbol{x})), & otherwise \end{cases}$$

This definition is very useful in the sense that anomalies are lying relatively remotely to such core objects. To exploit it, further definitions are necessary to be made. Additionally, to simplify the calculations all the samples will be considered as core objects [16]. Thus, definition of core-distance and reachability-distance will demand slight changes.

- *Core-distance* is modified to be the distance between sample $\boldsymbol{x}$ and its *MinPts*th nearest neighbor.
- *Reachability-distance* is modified such that:

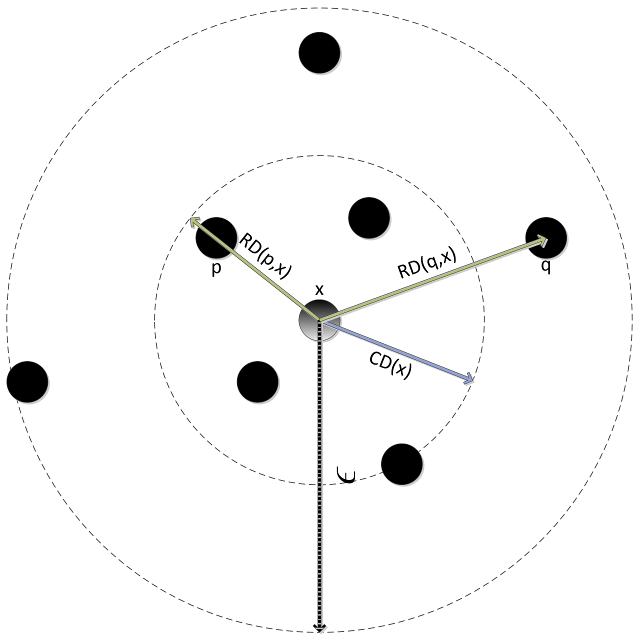


Figure 3.3 *Illustration of core and reachability distances when MinPts is set to 5. The ϵ-neighborhood ($N_\epsilon(\boldsymbol{x})$) contains more than MinPts samples implying that $\boldsymbol{x}$ is satisfying the core object condition.*

$RD(\boldsymbol{x}) = \max(CD(\boldsymbol{q}),\ distance(\boldsymbol{x}, \boldsymbol{q}))$, where $\boldsymbol{q}$ is the nearest neighbor of sample $\boldsymbol{x}$.

Based on [10], the following will be used to compute outlier scores.

Definition 3.3.9. (Local reachability density)
This basically measures the average local reachability distance of sample($\boldsymbol{x}$) given its *MinPts* nearest neighbors. Then, a score for each sample is generated inversely proportional to this value.

$$lrd(\boldsymbol{x}) = \frac{1}{\displaystyle\sum_{q \in N_{MinPts}(\boldsymbol{x})} \frac{RD(\boldsymbol{q})}{MinPts}} \tag{3.14}$$

Finally the outlier score can be calculated as

$$OS(\boldsymbol{x}) = \frac{\displaystyle\sum_{q \in N_{MinPts}(\boldsymbol{x})} \frac{lrd(\boldsymbol{q})}{lrd(\boldsymbol{x})}}{MinPts} \tag{3.15}$$

The outlier score now does not only summarize the density information of a given object but also explains the density of the neighboring samples. In a uniform cluster where each sample's local reachability density is equal this score will be equal to 1. However, if a suspicious object lies closer in the neighborhood of uniformly distributed instances, then it will have a higher score which may indicate it is more likely to belong to anomaly class. Thus, it is easier to detect anomalies which exhibit very similar behavior to the normal class.

To sum up, the reachability distance provides better description of the topology of samples because samples located closely will have very similar reachability distances since the core distances of those are likely to be similar. It is shown in [10] that this distance function reduces the fluctuations of the inter-object (i.e. inter-samples) distances proving to be good asset.

Clustering based on Outlier Scores

Let us denote the threshold according to which the instances are categorized as $\mathcal{T}_{outlier}$. Then,

$$\text{c}=\begin{cases} anomaly, & \text{if } OS(\boldsymbol{x_i}) > \mathcal{T}_{outlier} \\ normal, & \text{otherwise} \end{cases}$$

where c represents the class assignment of the ith object.

3.3.4 Support Vector Data Description

Besides the probabilistic and clustering approaches, outlier detection methods that learn decision functions based on normal training data can also be employed. In this section, the attention will be laid on Support Vector Data Description (SVDD) and its non-linear variant Kernel Support Vector Data Description (KSVDD) proposed in [52].

Support Vector Data Description (SVDD)

The aim in SVDD is to determine a spherical boundary describing the normal class as well as possible while rejecting the anomalies (outliers) so that any sample not conforming to this description (anomaly) is identified. This approach allows to adjust the outlier sensitivity by introducing flexibility in the spherical boundary and hence can be enhanced to represent the normal (target) data samples better. It is inspired from the Support Vector Machine (SVM) classifier which has been widely studied and used since 1960s [12, 14].

Let $\mathcal{D} = \{\boldsymbol{x}_k, k = 1, \ldots, M\}$ denote the dataset where $\boldsymbol{x}_k$ is the kth sample and a column vector satisfying $<\boldsymbol{x},\boldsymbol{x}>=\boldsymbol{x}^T\cdot\boldsymbol{x}$. The data description is then to be represented by a hypersphere with radius $R>0$ and center $\boldsymbol{a}$. The fundamental idea is to minimize the volume of the hypersphere by minimizing R^2 under the restriction that all the samples in $\mathcal{D}$ are included in the sphere. However, this restriction leads to being unable to differentiate between normal and anomaly samples in $\mathcal{D}$. Therefore, the method loosens this restriction by introducing slack variables in the error function which are responsible for penalization of objects lying outside of the boundary.

The error function is defined as

$$F(R,\boldsymbol{a}) = R^2 \tag{3.16}$$

with the constraints:

$$\|\boldsymbol{x}_i-\boldsymbol{a}\|^2 \leq R^2, \forall i \tag{3.17}$$

In order to account for the anomalies, the slack variables are incorporated into Eq. 3.16 as follows.

$$F(R,\boldsymbol{a}) = R^2 + C\sum_i \xi_i \tag{3.18}$$

with the constraints that most of the samples are surrounded by the sphere:

$$\|\boldsymbol{x}_i-\boldsymbol{a}\|^2 \leq R^2+\xi_i,\ \xi\geq 0\ \forall i \tag{3.19}$$

Eq. 3.18 attempts to minimize the (hyper)sphere while tolerating the outliers. The parameter C is introduced to control the influence of each objective (smaller volume, smaller errors) in the error function. In order to impose the constraints in 3.19 into the error function in 3.18, one can rewrite the error function using Lagrange multipliers as:

$$\begin{aligned}L(R,\boldsymbol{a},\alpha_i,\gamma_i,\xi_i) = R^2 + C\sum_i \xi_i - \sum_i \alpha_i[R^2+\xi_i-(\|\boldsymbol{x}_i\|^2-2\boldsymbol{a}\cdot\boldsymbol{x_i}+\|\boldsymbol{a}\|^2)]\\ -\sum_i \gamma_i\xi_i\end{aligned} \tag{3.20}$$

where the Lagrange multipliers $\alpha_i \geq 0$ and $\gamma_i \geq 0$. This error function, L, must be minimized with respect to $R,\boldsymbol{a},\xi_i$ and maximized with respect to α_i and γ_i. By setting the coresponding derivatives to zero, one obtains the following constraints:

$$\frac{\partial L}{\partial R} = 0, \text{ leading to, } \sum_i \alpha_i = 1 \tag{3.21}$$

$$\frac{\partial L}{\partial \boldsymbol{a}} = 0, \text{ leading to, } \boldsymbol{a} = \frac{\sum_i \alpha_i \boldsymbol{x_i}}{\sum_i \alpha_i} = \sum_i \alpha_i \boldsymbol{x_i} \tag{3.22}$$

$$\frac{\partial L}{\partial \xi_i} = 0, \text{ leading to, } 0 \leq \alpha_i \leq C \tag{3.23}$$

Inserting these constraints into L results in the following optimization problem.

$$L = \sum_i \alpha_i(\boldsymbol{x_i} \cdot \boldsymbol{x_i}) - \sum_{i,j} \alpha_i \alpha_j (\boldsymbol{x_i} \cdot \boldsymbol{x_j}) \tag{3.24}$$

Maximization of 3.24 produces an α_i for each object $\boldsymbol{x_i}$. Depending on a given object $\boldsymbol{x_i}$, there are three possible cases for the Lagrange multipliers.

Case-1: If object $\boldsymbol{x_i}$ lies inside the sphere, i.e. $\|\boldsymbol{x}_i - \boldsymbol{a}\|^2 < R^2$, the constraint is satisfied. Hence, $\alpha_i = 0$ and $\gamma_i = 0$.

Case-2: If object $\boldsymbol{x_i}$ lies on the boundary of the sphere, i.e. $\|\boldsymbol{x}_i - \boldsymbol{a}\|^2 = R^2$, the constraint should be imposed. Hence, $0 < \alpha_i < C$ and $\gamma_i = 0$.

Case-3: If object is beyond the boundary, i.e. $\|\boldsymbol{x}_i - \boldsymbol{a}\|^2 > R^2$, then α_i becomes maximized. Hence, $\alpha_i = C$ and $\gamma_i > 0$.

Since constraint in 3.22 states the center of the sphere is determined by the weighted some of all samples in $\mathcal{D}$, the objects falling into the first case are not required to describe the data. Hence, the remaining objects whose α_is are non-zero are referred to as the *support vectors* of the description.

In order to test an object against the model, its distance to the center $\boldsymbol{a}$ is calculated. An object is considered to be *normal* if it lies within the hypersphere. The radius R can be computed using only those support vectors residing on the boundary $(0 < \alpha_i < C)$ as follows:

$$R^2 = \|\boldsymbol{x}_j - \boldsymbol{a}\|^2 = \boldsymbol{x}_j^2 - 2\boldsymbol{a} \cdot \boldsymbol{x}_j + \boldsymbol{a}^2 \tag{3.25}$$

$$= (\boldsymbol{x}_j \cdot \boldsymbol{x}_j) - 2\sum_i \alpha_i (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) + \sum_{i,j} \alpha_i \alpha_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \tag{3.26}$$

for any support vector $\boldsymbol{x}_j$ with $\alpha_j \neq C$. In other words, the support vectors that are positioned outside the sphere are not included implying that only those support vectors with $0 < \alpha_j < C$ are sufficient to represent the description.

SVDD WITH NEGATIVE (ANOMALY) EXAMPLES

If the dataset contains both normal and anomaly samples, then a richer description can be obtained. However, the SVDD equations shown so far have to be modified in order to incorporate the information provided by the negative examples. To formulate, the normal objects are denoted with indices i and j whereas the anomalies are denoted with indices l and m.

$$F(R, \boldsymbol{a}, \xi_i, \xi_l) = R^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l \tag{3.27}$$

with the following conditions:

$$\|\boldsymbol{x}_i - \boldsymbol{a}\|^2 \leq R^2 + \xi_i \text{ and } \|\boldsymbol{x}_l - \boldsymbol{a}\|^2 \leq R^2 - \xi_l,\ \xi \geq 0\ \forall i, l \tag{3.28}$$

In a similar fashion with the previously explained SVDD, the following constraints can be shown after imposing 3.28 and setting the derivatives of L w.r.t. R, $\boldsymbol{a}$, ξ_i, ξ_l to zero:

$$\frac{\partial L}{\partial R} = 0,\ \text{leading to},\ \sum_i \alpha_i - \sum_l \alpha_l = 1 \tag{3.29}$$

$$\frac{\partial L}{\partial \boldsymbol{a}} = 0,\ \text{leading to},\ \boldsymbol{a} = \sum_i \alpha_i \boldsymbol{x}_i - \sum_l \alpha_l \boldsymbol{x}_l \tag{3.30}$$

$$\frac{\partial L}{\partial \xi_i} = 0,\ \text{leading to},\ 0 \leq \alpha_i \leq C_1,\ \forall i \tag{3.31}$$

$$\frac{\partial L}{\partial \xi_l} = 0,\ \text{leading to},\ 0 \leq \alpha_l \leq C_2,\ \forall l \tag{3.32}$$

In [52], substitution of(3.29) -(3.32) is shown to result in:

$$\begin{aligned} L = \sum_i \alpha_i (\boldsymbol{x}_i \cdot \boldsymbol{x}_i) - \sum_l \alpha_l (\boldsymbol{x}_l \cdot \boldsymbol{x}_l) \\ - \sum_{i,j} \alpha_i \alpha_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) + 2 \sum_{l,j} \alpha_l \alpha_j (\boldsymbol{x}_l \cdot \boldsymbol{x}_j) - \sum_{l,m} \alpha_l \alpha_m (\boldsymbol{x}_l \cdot \boldsymbol{x}_m) \end{aligned} \tag{3.33}$$

Although this method provides a more flexible representation (at the expense of reduced tightness of the description), it requires labels. Hence, it is not to be used in this work.

KERNEL SVDD (KSVDD)

In [52] Tax and Duin explains another method for more flexible description of the data. This method exploits a kernel trick by which the inner product $\boldsymbol{x}_i^T \boldsymbol{x}_j$ is replaced by a kernel function so that $K(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$. As a result, the rigid spherical boundary is transformed into a less rigid one. The kernel function is used as a mapping from the original feature space to a possibly higher dimensional feature space. However, the kernel function should be selected so that it maps the normal samples into the sphere while leaving the anomalies outside. In the literature, many kernel functions have been proposed for Support Vector Machines [47, 54]. In the rest of the chapter and the thesis work, the kernel function is going to be taken as the Radial Basis function (RBF):

$$K(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \tag{3.34}$$

In order to test an object $\boldsymbol{z}$ with respect to the center, the following inequality is utilized.

$$\begin{gathered} \|\boldsymbol{z} - \boldsymbol{a}\|^2 = \boldsymbol{z}^2 - 2\boldsymbol{a} \cdot \boldsymbol{z} + \boldsymbol{a}^2 < R^2 \\ (\boldsymbol{z} \cdot \boldsymbol{z}) - 2\sum_i \alpha_i (\boldsymbol{x}_i \cdot \boldsymbol{z}) + \sum_{i,j} \alpha_i \alpha_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) < R^2 \end{gathered} \tag{3.35}$$

Since $K(\boldsymbol{z}, \boldsymbol{z})$ will be equal to 1 for the RBF kernel and the last term is independent of the test object $\boldsymbol{z}$, it can be rewritten as

$$\sum_i \alpha_i \exp\left(\frac{-\|\boldsymbol{z} - \boldsymbol{x}_i\|^2}{2\sigma^2}\right) > \frac{R^2}{2} + C_R \tag{3.36}$$

where $C_R = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$. Hence, whenever the object $\boldsymbol{z}$ satisfies 3.36, it is considered to be of the normal class.

Possible Drawbacks and solutions

Support Vector Data Description is a very powerful tool to describe the normal behaviour. However, it suffers if the number of outliers in the training dataset are large and they deviate from the remaining points in the multidimensional space. The problem stems from the constraint that all the data points must be surrounded by the description and that this constraint leads to a loose description of the normal class. Loose descriptions are not desirable since it makes it harder to track the outliers that lie very close to the normal class boundaries. Given our dataset, it is not hard to conclude that SVDD will suffer from the outlier population unless one provide a clean (few or no outliers) enough subset of the data which represents the normal behavior. Experiments have shown that SVDD on a dataset such as TTY2-500k performs extremely poorly.

In the absence of labels (unsupervised learning), one can exploit any of the previous methods to reach at a clean representation of the normal dataset. Then, this cleaned dataset might be used to describe the normal class with possibly a small group of normal-like anomalies. Finally, KSVDD (or SVDD) can be used to classify unseen data quite efficiently compared to previous methods since the number of support vectors used to describe whole dataset is much less than the size of the dataset. For comparison purposes, one can think the density based methods to have computational complexity of $\mathcal{O}(n^2)$ where n is the number of samples in a given dataset. SVDD will produce k support vectors such that $k \ll n$ resulting in less computation need.

A Two-tier Anomaly Removal Framework

In this framework, the Naive Bayesian method's output is piped to the Density-based method in order to discard as many anomalies as possible with precision and recall as high as possible. The main outline of the procedure is depicted in Fig. 3.4. The resulting dataset is then used to model the target class with two parameters, R and $\boldsymbol{a}$.

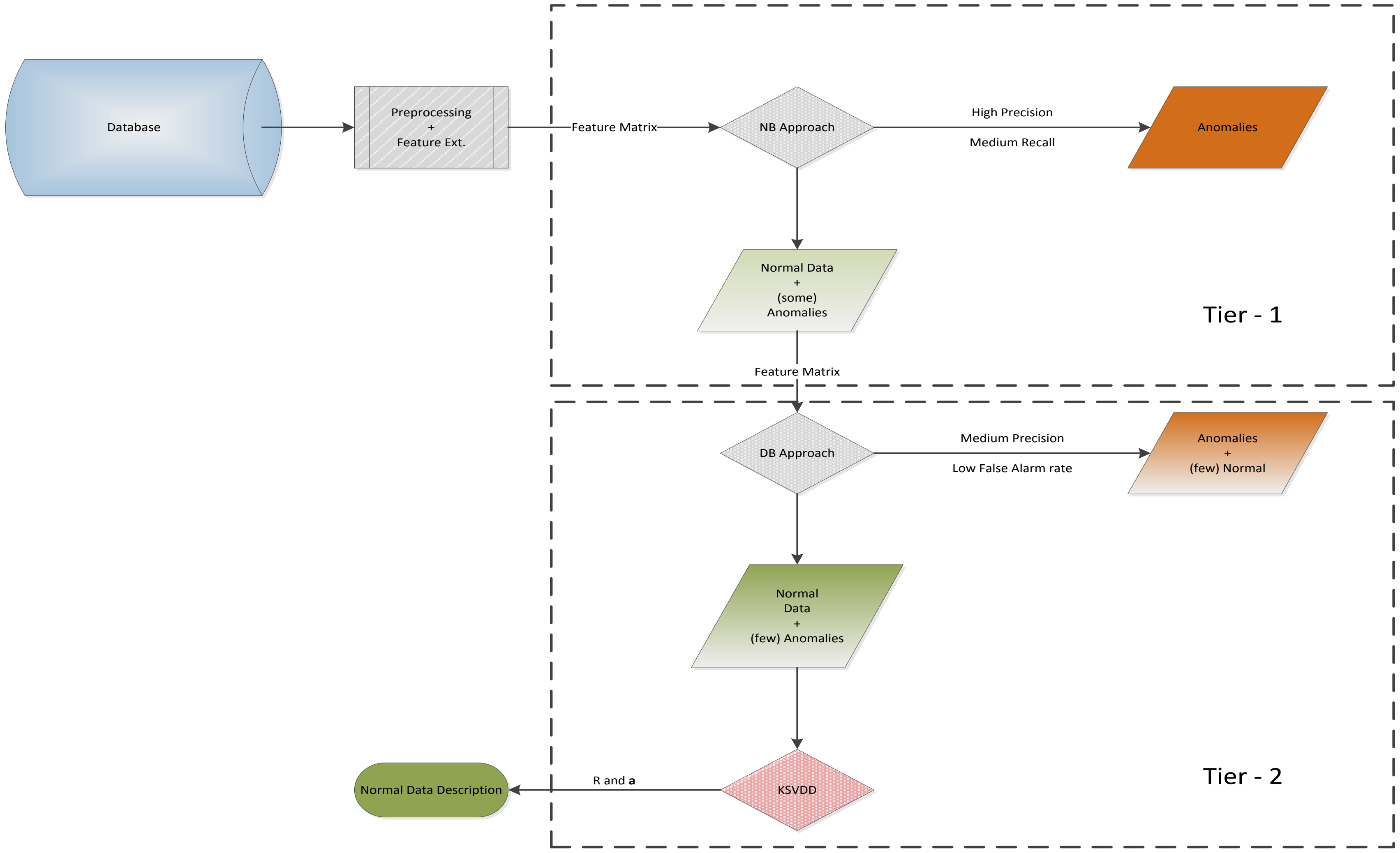


Figure *3.4 Two-tier anomaly removal framework*

4. EVALUATION

In this chapter, the performances of the given methods are given in elaborate detail. Additionally the tunable parameters which affect the results significantly are explained.

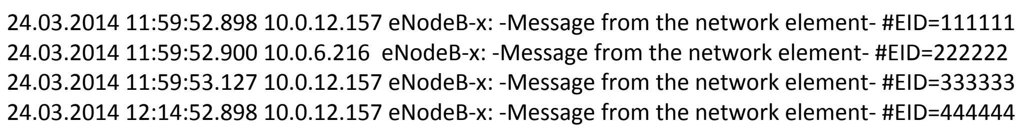
4.1 Datasets and Platform

The datasets used in this thesis are provided by Tieto as a part of project **D2I**. There are exactly two different datasets which are generated artificially to simulate a real telecommunication network which is located in Poland. Due to confidentiality, it is not allowed to present the datasets here (A course representation of its structure can be seen in Fig. 4.1). However, the first dataset, namely, TTY-500k consists of 500330 lines of system logs which are thrown by the network elements while the second one, TTY2-500k, contains 516692 of those. The main characteristic of TTY-500k is that it includes only one type of fault (anomaly) hidden in the dataset (as well as the normal). On the other hand, TTY2-500k houses 5 different types of faults (anomalies) of varying lengths besides the normal. However, to note, the so-called fault, numbered as 2, is not considered during evaluation as it simply occurs from the beginning to the end of records. In Fig. 4.2, the layout of anomalies are shown. Each fault type has a different characteristic (in practical terms) and different average length (in time). Table 4.1 shows the average length of each fault type.

Fault Type No.	Length
1	20 minutes
2	2.5 days
3	5 minutes
4	9 seconds
5	2 minutes

Table 4.1 *The average time interval in which each fault takes place in TTY2 dataset.*

Unlike supervised learning paradigm, unsupervised learning (methods) usually do not require data separation for training and testing purposes. The methods used in this work generally take the whole dataset at once, then either learn a model and test



```
24.03.2014 11:59:52.898 10.0.12.157 eNodeB-x: -Message from the network element- #EID=111111
24.03.2014 11:59:52.900 10.0.6.216  eNodeB-x: -Message from the network element- #EID=222222
24.03.2014 11:59:53.127 10.0.12.157 eNodeB-x: -Message from the network element- #EID=333333
24.03.2014 12:14:52.898 10.0.12.157 eNodeB-x: -Message from the network element- #EID=444444
```

Figure 4.1 *The structure of the log lines is illustrated in the above figure. A system log basically includes the time of reporting, the type of the system element reporting, IP, the message and a system related ID.*

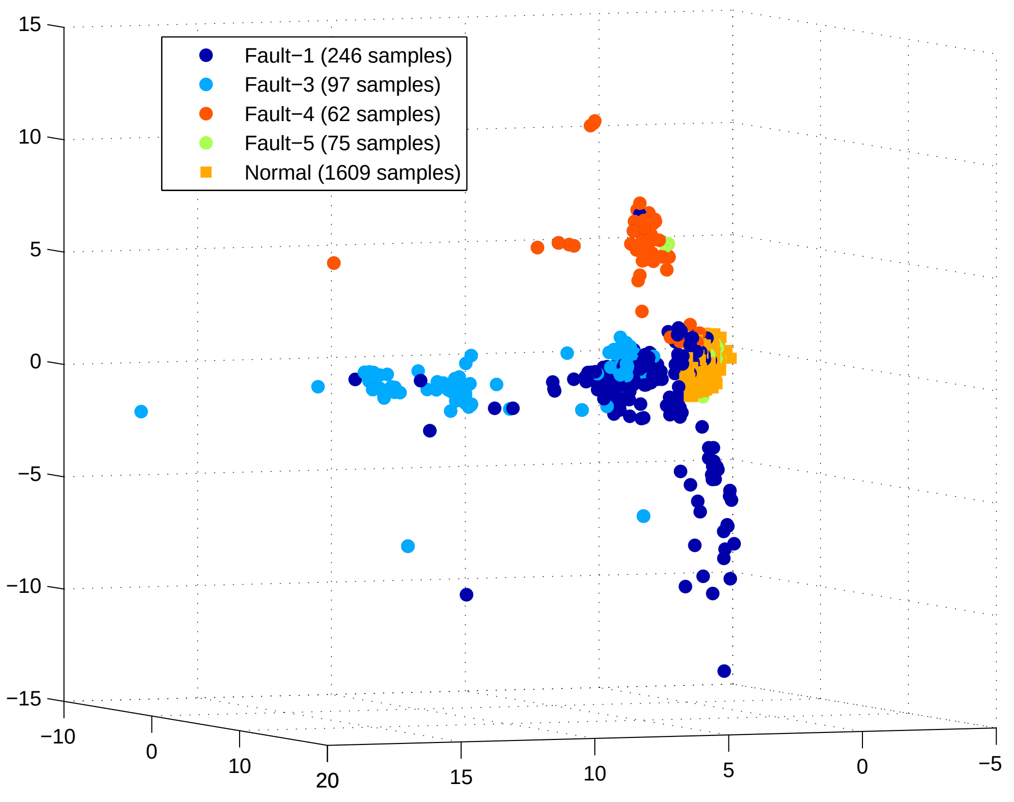


Figure 4.2 *TTY2-500k dataset is projected onto the first three principal components for visualization purposes. The four fault types that are taken into consideration are represented by different colors. One can easily see that the normal class forms a very dense sphere.*

each sample against it or cluster into two categories. Furthermore, these methods (algorithms) are deterministic implying that the results remains the same as long as the experiments are conducted under the same conditions. However, the results may show variance depending on the groundtruth generation technique from the provided fault-lists in which the start and end times of each fault are indicated. In general, groundtruth is strict. An instance is either anomaly or not in the anomaly detection context. However, since a time-series data is of concern and instances are basically fixed-length windows, it becomes unclear to determine whether an instance is anomalous or not. There are 3 different cases:

- The given window does not contain any lines which fall into one of the given fault time intervals in fault list.
- The given window contains a certain number of lines which fall into (at least) one of the given fault time intervals in fault list.
- The given window consists only of lines which fall into (at least) one of the given fault time intervals in fault list.

The first and the last cases are trivial. However, the second case is hard to figure out. In most of the experiments, the second case is handled by marking those windows containing at least one line from the fault intervals as anomaly. On the other hand, depending on how labeling is done, the results are likely to change. If it is considered as a binary problem, those windows (samples) with very few lines from any of the fault intervals are considered as anomaly. Increasing the threshold in selection may assign these hard-to-classify samples to the normal class and consequently the performance is likely to enhance.

Although occasionally C++ and Python was used for research purposes, the main implementation platform was MATLAB. For preprocessing and feature extraction, functions implemented by us was used. For Kernel Support Vector Data Description, the open source toolbox (Data description toolbox dd tools 2.0.0 [53]) developed in Delft University of Technology was used. The MATLAB implementation of OPTICS was obtained from [15]. The author is responsible for the implementation of the remaining methods.

4.2 Evaluation setup

In Section 3, it has been shown that each method has their own internal or external parameters which can be tuned to achieve better performance. Therefore, one should consider each method as a function of those and evaluate accordingly. However, optimization of multiple of those at the same time might hamper the analysis. Thus, certain parameters should be fixed (not necessarily at a random value) while the effect of other parameters are observed during the experiments. We will present the major parameters that are subject to tuning in an ordered manner.

4.3 Results for the Naive Bayesian Approach

In this section, the performance of Naive Bayesian approach is presented. The effects of certain parameters, feature extraction methods and dimensionality reduction are shown in order to fully understand the capabilities and limitations of this approach.

4.3.1 Effect of Dimensionality Reduction

Since dimensionality reduction has been observed to play a crucial role in performance, it is presented initially so that the remaining details can be studied more clearly in the rest of the chapter.

Dimensionality refers to the number of observation types (features) in the feature vector. It is usually not known which of these are useful for good results prior to analysis. Hence, redundancy of features may bring about complex and noisy representation of the dynamics of the studied phenomenon. It is sometimes more desirable to capture the underlying manifold in a lower dimensional space where the behavior of the target classes are less vague. As a consequence, it is a good practice to study how much the data dimensionality can be reduced without losing too much information.

To demonstrate the effect of dimensionality reduction in a clear way, we are going to fix some parameters (see Table 4.2) and compare the results under the same conditions except for the number of dimensions.

Parameter	**Value**
Feature extraction method	Rule set-1
Window size	1000
Dataset	TTY2

Table 4.2 *The parameters that are kept constant.*

Without Dimensionality Reduction

As in Fig. 4.5, it is observed that there is a narrow threshold interval where the recall and g-mean values indicate relatively acceptable performance. It is clear that depending on the threshold parameter $\mathcal{T}_{normal}$ there is a huge trade-off between g-mean(or precision) and false alarm rate. Setting $\mathcal{T}_{normal}$ to mean value results in 16.98% false alarm rate while achieving 84.58% recall. This implies almost 85% of the anomalous samples are returned with the precision 62.55% while approximately 17% of the normal samples are misclassified as anomaly.

With Dimensionality Reduction

When dimensionality reduction methods are applied, it is expected that the results

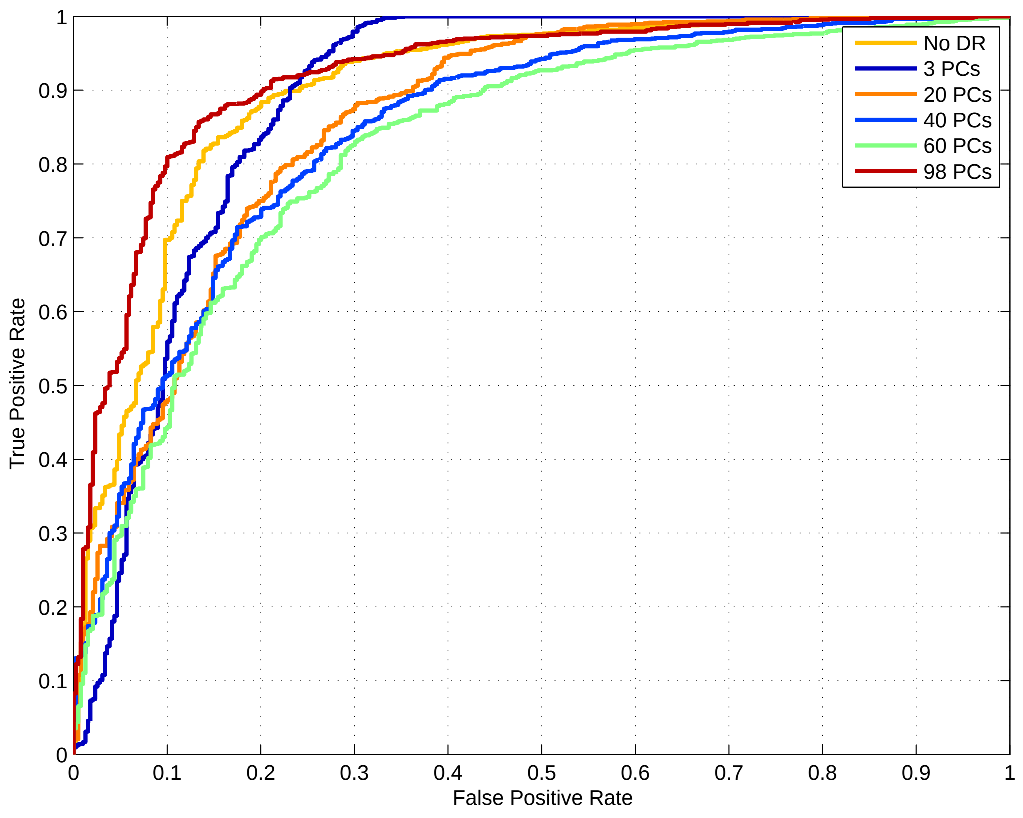


Figure 4.3 *Unsupervised classification results for the configuration given in Table 4.2 when dimensionality reduction is applied.*

should improve in certain ways since a more compact and noise-reduced representation is present. As can be seen from Fig. 4.3, using Principal Component Analysis on the *raw* features to reduce dimensionality (or transform feature space) provides better classification in general. It has been observed that reducing the dimensionality of the data to three brings about a drastic decrease at the same threshold value while providing 100% precision (g-mean of 94.03%). Although the recall rate is decreased to 60.93% from 84.58%, achieving zero false alarm rate can be desirable from the system perspective. In practice, this might correspond to less maintenance cost since there is no need to hire experts to go and check a machine which is properly operating. One should also investigate the number of principal components with which the performance is boosted. From Fig. 4.4, an interesting observation can be deduced. The figure underlines the fact that the principal components which correspond to the largest variances (i.e. 1-5) and the principal components which correspond to the smallest variances (i.e. 90-98) are the most significant. The reason for the former group to be useful is clear. However, why the latter group boosts the performance might have an interesting explanation. If one thinks in terms of variances, the last few principal components describe the data in a very tight way. The normal instances are distributed around a point with small variance while the

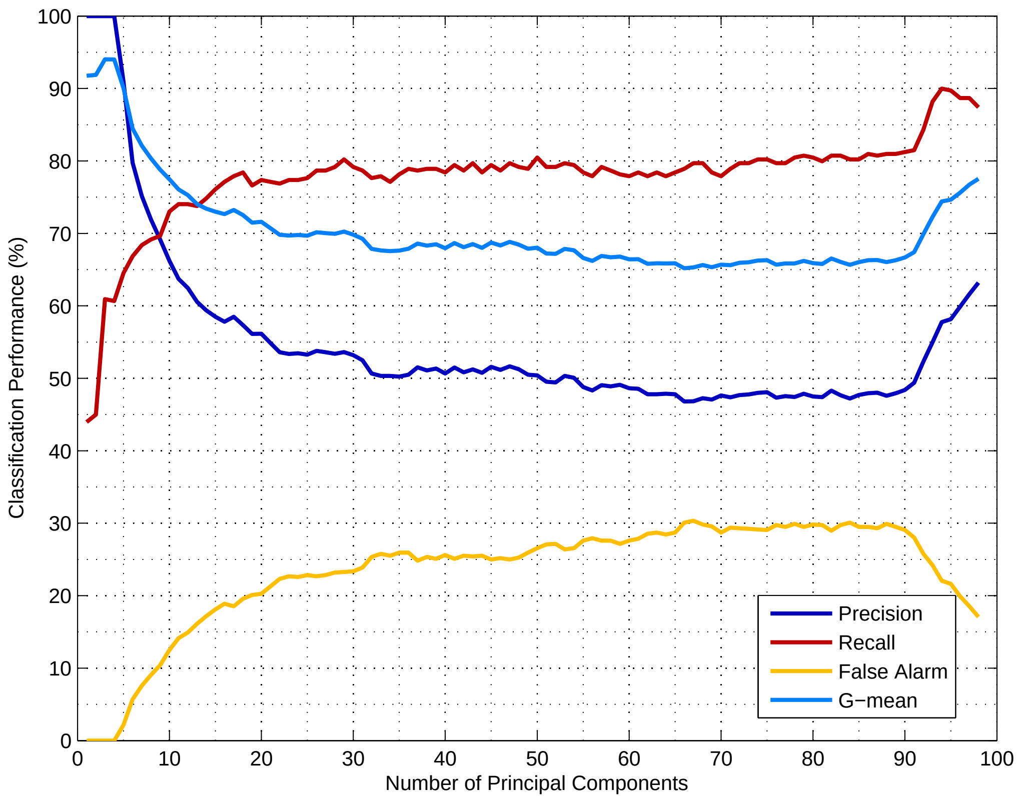


Figure 4.4 *The performance of Naive Bayesian method used in this thesis work against the number of principal components used ($\mathcal{T}_{normal}$ = Mean).*

anomalies are likely to account for most of the variance. Therefore, Bayesian approach assigns lower normality scores to the possibly anomalous objects. When only the first five and last nine principal components are used for identifying anomalies, the false alarm rises up to 0.5172%. On the other hand, recall is increased by 12% compared to three-component case (assuming that threshold is set at mean value.).

Another advantage of dimensionality reduction in general sense is that it offers drastic decrease in the computational complexity in terms of memory and time, especially when the size of the processed batch is large. For instance, when on-line decision making is crucial, it becomes very efficient.

4.3.2 Effect of the Feature Extraction Methods

It is clear that feature extraction affects the performance since every feature set has its own strengths and weaknesses in describing the in multidimensional space. To investigate the advantages and disadvantages of the methods used to generate feature vectors, we can fix some parameters and vary the feature types. In Fig. 4.5, two different types explained in Section 3.1 as well as BoW are compared in terms

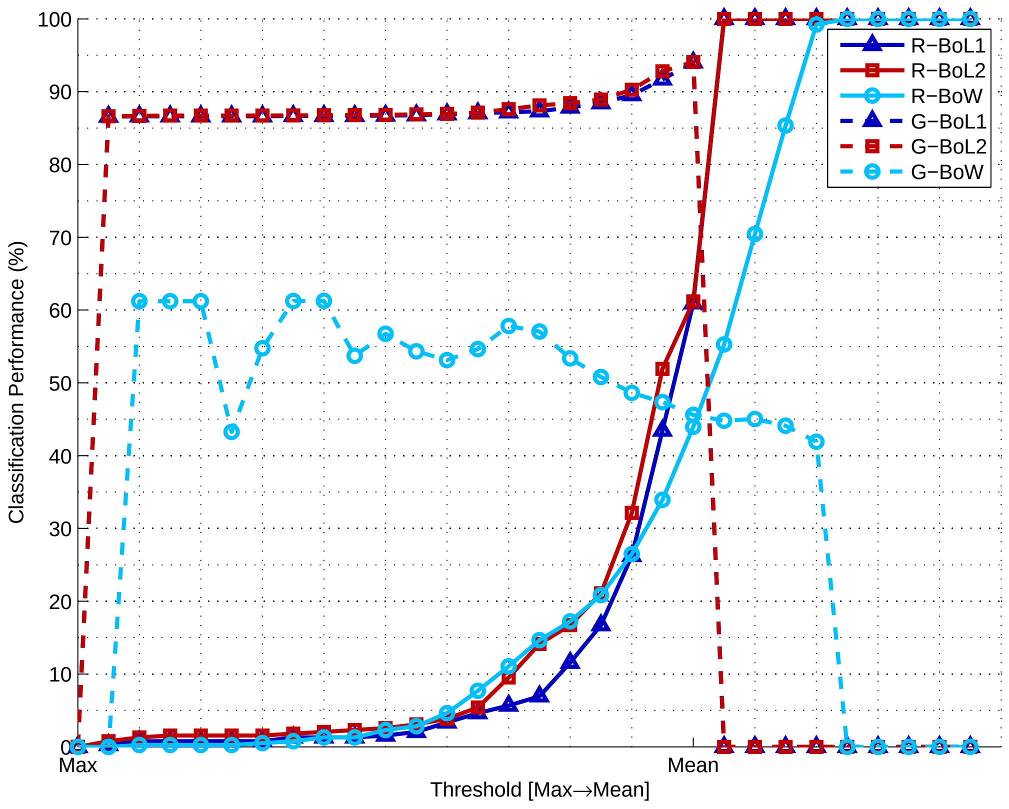


Figure 4.5 *The comparison of different features. In the legend, R and G stand for recall and g-mean, respectively. Also, BoL1 corresponds to the bag-of-lines features generated based on Legal Word Selection Rule-1 whereas BoL2 corresponds to those generated based on Legal Word Selection Rule-2.*

of recall and g-mean. Bag-of-words yields a weaker representation power compared to Bag-of-Lines features as its g-mean score is less than half of its counterparts'. However, it is hard to differentiate between BoL features generated based on different set of selection rules.

4.3.3 Effect of Window Size

Investigating the effect of window size is not simple when the time interval that the window spans is not known as in this case (fixed-windows). However, experiments have shown that a right choice of window length can lead to better recall values. The experiments also indicated that false alarm and g-mean values are not as window size dependent as the recall value as shown in Fig. 4.6.

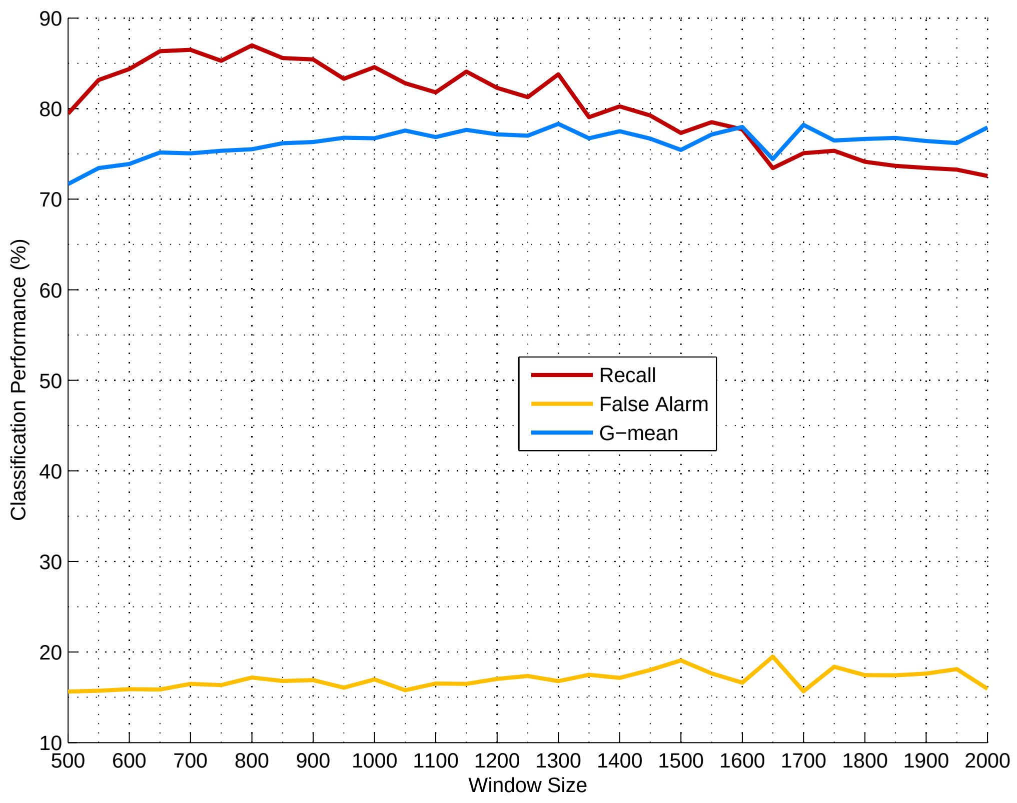


Figure 4.6 *The variation of performance values over various window sizes without dimensionality reduction ($\mathcal{T}_{normal}$ = Mean)*

4.4 Results for the Density-Based Approach

In similar fashion to Naive Bayesian Approach, we show the characteristics and performance of the density-based approach.

4.4.1 Effect of the Window Sizes

Although the windows size directly affects the results, the efforts to explain in what ways it relates to improvements did not yield any strong correlation to certain phenomenon such as average time spanned by windows. One may expect that windows of length (in terms of time) close to duration of the first fault type (major/common fault type, $\approx$ 23 minutes) should lead to better results. However, this is shown not to be the case by the analysis. Actually fixed-windows of length 1800 are found to correspond to 24 minute time intervals although nor this value or its multiples show no trait for improvement.

4.4.2 Effect of Dimensionality Reduction

The density-based approach heavily relies on the euclidean distance between each sample pairs. In similar sense to k-nearest neighbor (KNN), the distance metric should be selected so that the multidimensional nature of the data do not affect the results. In case it does not improve significantly by changing the metric, one can consider dimensionality reduction methods.

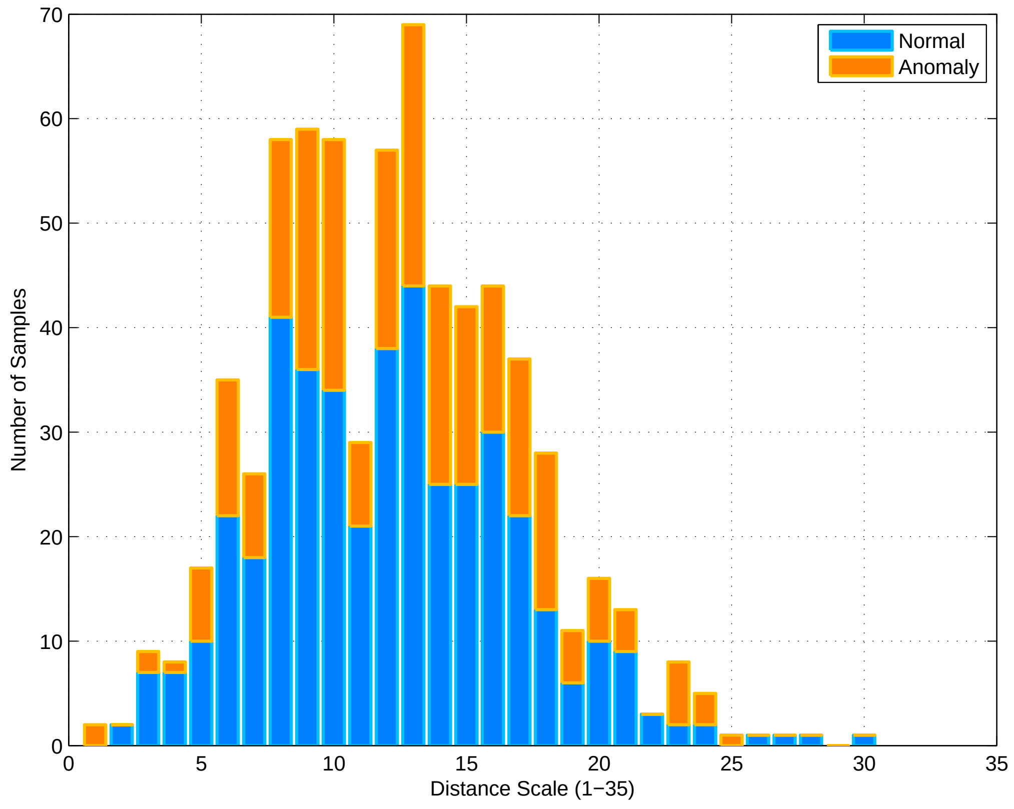
Density-based methods assume the average pairwise distance in the normal class is small and anomalies are well separated from the normal samples. In Fig. 4.7, it is shown that PCA helps to solve this issue (To generate the results, the configuration in Table 4.2 is used.). In fact, PCA, by making linear transform of the basis vectors, synthesizes a new set of basis which maintain the topological relationships (distances etc.) of samples. Compared to the original space, the noise and redundancies between features are eliminated in this new space spanned by the new basis vectors. Hence, anomalies mostly constitute the samples whose kth neighbors are the remotest. Fig. 4.8 shows up to what extent the number of principal components can bring the performance and after which point adding new dimensions is not necessary. The results indicate that it is possible to outperform the Naive Bayes approach simply using the local densities of points.

4.4.3 Effect of the Feature Extraction Methods

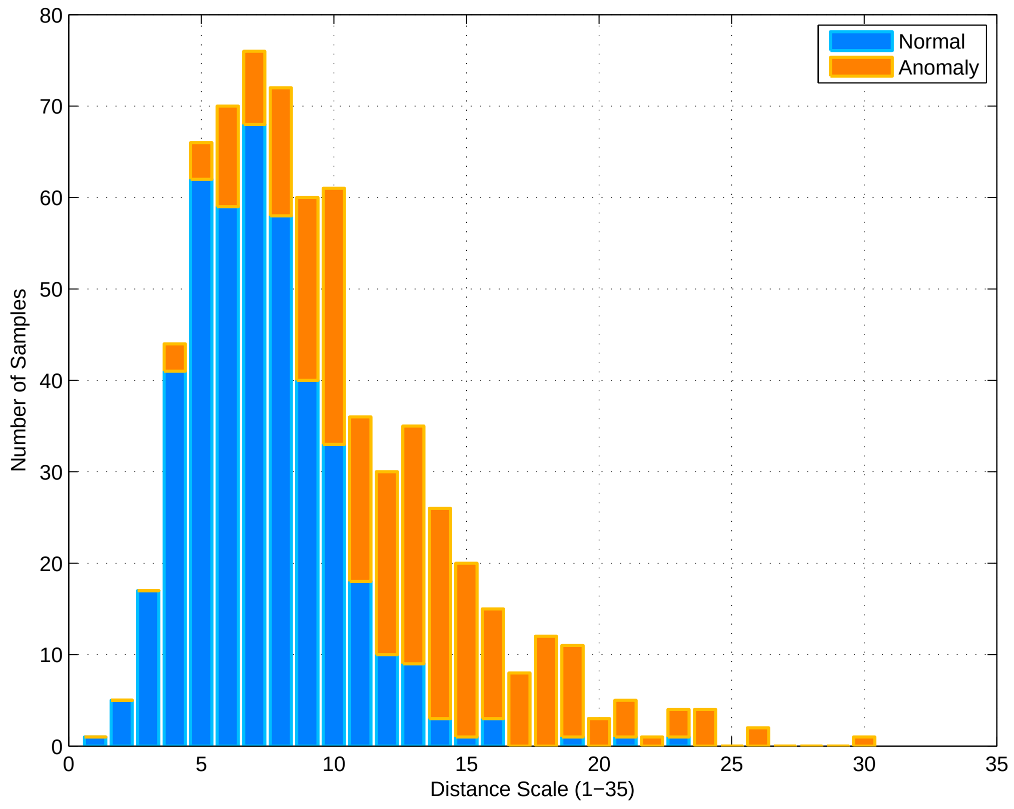
In order to compare how the feature extraction methods help the algorithms, we will use Receiver Operator Operating Characteristic (ROC) curves. The data used in this comparison is TTY2-500k and window size is 1000. Additionally, in all configurations the basis of the datasets are replaced by equal number of principal components (98, 107 and 301 respective to the order on Fig. 4.9) to make sound comparison. Fig. 4.9 shows that BoL features possess superiority compared to BoW features. The areas under the curves differ significantly.

4.4.4 Effect of k

The parameter, k, which controls the neighborhood size affects the performance as shown in Fig. 4.11. Its influence is not so critical or significant as to classification performance, however it must be taken into account in order to fine-tune the overall system. Small values up to 5 are sufficient. Actually, the areas under the ROC curves indicate that higher neighborhoods do not help but deteriorate the performance.



(a) Original dataset



(b) PCA transformed dataset with the same number of dimensions

Figure 4.7 *Above histograms show the distribution of kth neighbor distances. Fig.(a) tells that original features are not very discriminative for separating two classes. On the other hand, PCA transformed features are more convenient since clearly anomalies tend to have relatively remote kth neighbors (see Fig.(b)).*

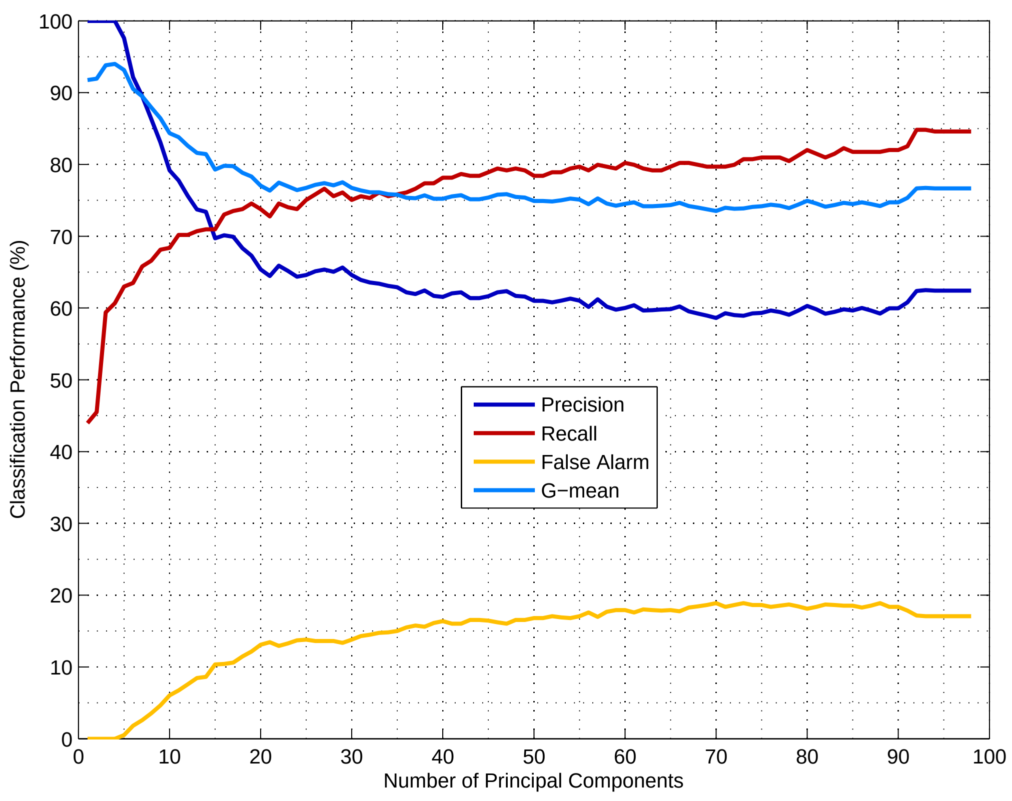


Figure 4.8 *The variation of performance values against the number of principal components used ($\mathcal{T}_{normal}$ = Mean, $\beta = 0$)*

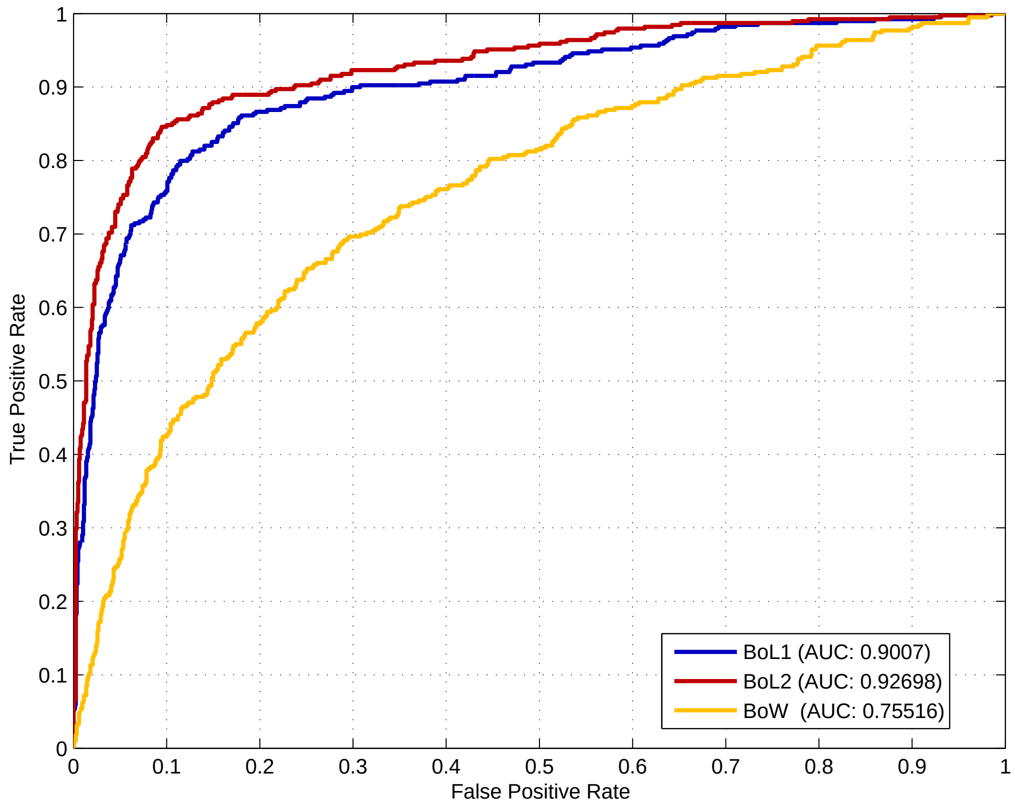


Figure 4.9 *The effect of different feature types in classification performance ($\beta = 0$).BoL1 corresponds to the bag-of-lines features generated based on Legal Word Selection Rule-1 whereas BoL2 corresponds to those generated based on Legal Word Selection Rule-2.*

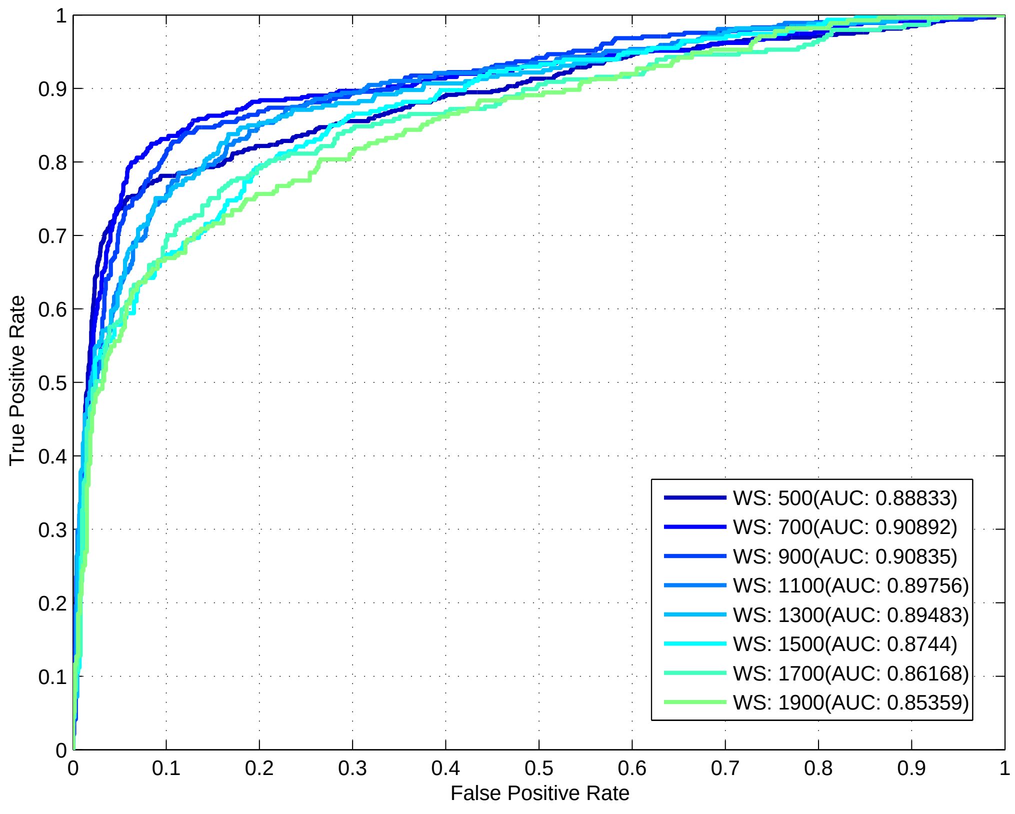


Figure 4.10 *ROC curves for different window sizes*

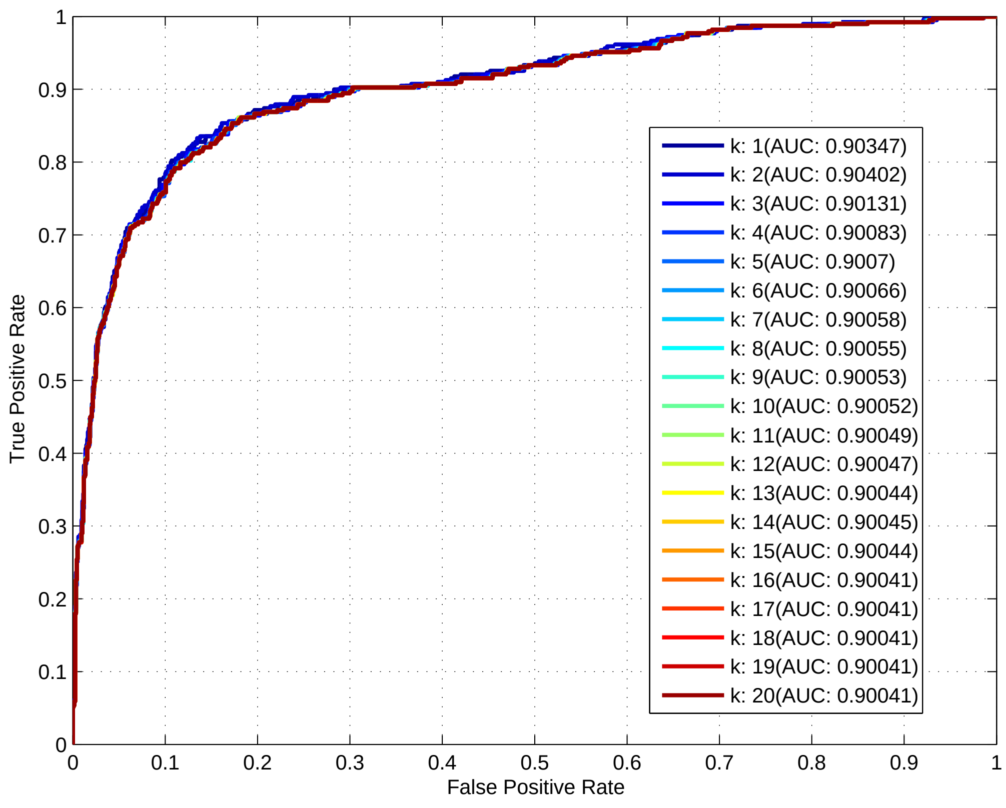


Figure 4.11 *The effect of the parameter k in density-based approach.*

4.5 Results for the OPTICS-based Approach

In the experiments it was observed that this approach performed comparedly worse than the previous methods. Specifically, this method resulted in higher false alarm rate in general and lower precision in comparison.

4.5.1 Effect of Dimensionality Reduction

OPTICS-based method exhibits a fluctuant behavior as the number principal components is increased. When PCA is not applied (threshold set at the mean, $k = 40$), the precision and recall are around 40% while false alarm rate is considerably high, 42%. Using 20 of the most principal components reduce the false alarm rate up to 5% while increasing the precision and recall to 80 and 60%, respectively. It is still very crucial to transform (not necessarily reduce) this dataset from the original feature coordinates to new ones. Although stated previously, it is useful to remind that OPTICS-based approach relies on pairwise distances. The problem is again that the histogram features are creating very entangled clouds making it harder to identify the anomalies based on pairwise distances.

4.5.2 Effect of $MinPts$

Although the neighborhood size is not a significant factor for the previous method, parameter $MinPts$ plays a significant role in OPTICS-based approach. Letting small numbers of elements to form clusters does not result in good classification as Fig. 4.12 indicates. The experiments have shown that the parameter $MinPts$ enhances the discriminative power of outlier scores up to 40 neighbors after which the performance values saturate.

4.5.3 Effect of Feature Extraction

As shown with the the previosly shown methods, BoW exhibits inferior performance compared to BoL method. One can note the great difference between the ROC curves of the BoW and BoL features in Fig. 4.14. Although BoW features results in better TPR compared to previous methods (see Fig. 4.9), it still performs worse than any of the others. BoL2 (Legal Word Selection Algorithm-2) outperforms BoL1 by a small margin in AUC.

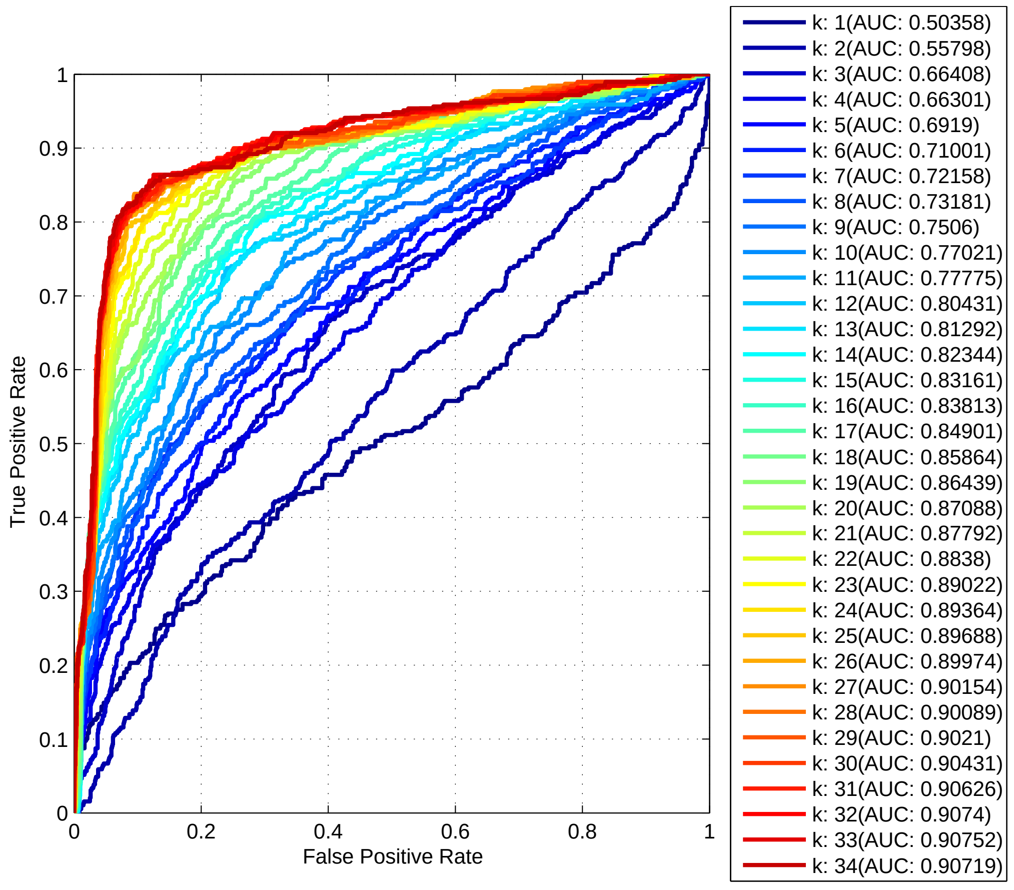


Figure 4.12 *The effect of the parameter $MinPts$ in OPTICS-based approach.*

4.5.4 Effect of the Window Sizes

Unlike the previous methods OPTICS-based approach has been observed to operate better with comparedly larger window sizes than 2000. In the range of 500-4000, best results are obtained with window size of 3200 when the threshold, $\mathcal{T}_{outlier}$ (see Fig. 4.13), is set to the mean value of the scores obtained from the dataset.

4.6 Results for Sparse Autoencoders

Since the effects observed due to autoencoders are different than PCA, it is better to investigate the results separately. In the thesis, it has been observed that using the sparse autoencoders in a cascaded system where the bottleneck representation of the first autoencoder is fed to the second one as input leads to interesting results (see Fig. 4.15). The study showed that if the training can be done effectively, i.e. the random initialization is good enough, the normal and anomaly classes can both be represented as clusters as in Fig. 4.16. Consequently, it enables one to use simpler

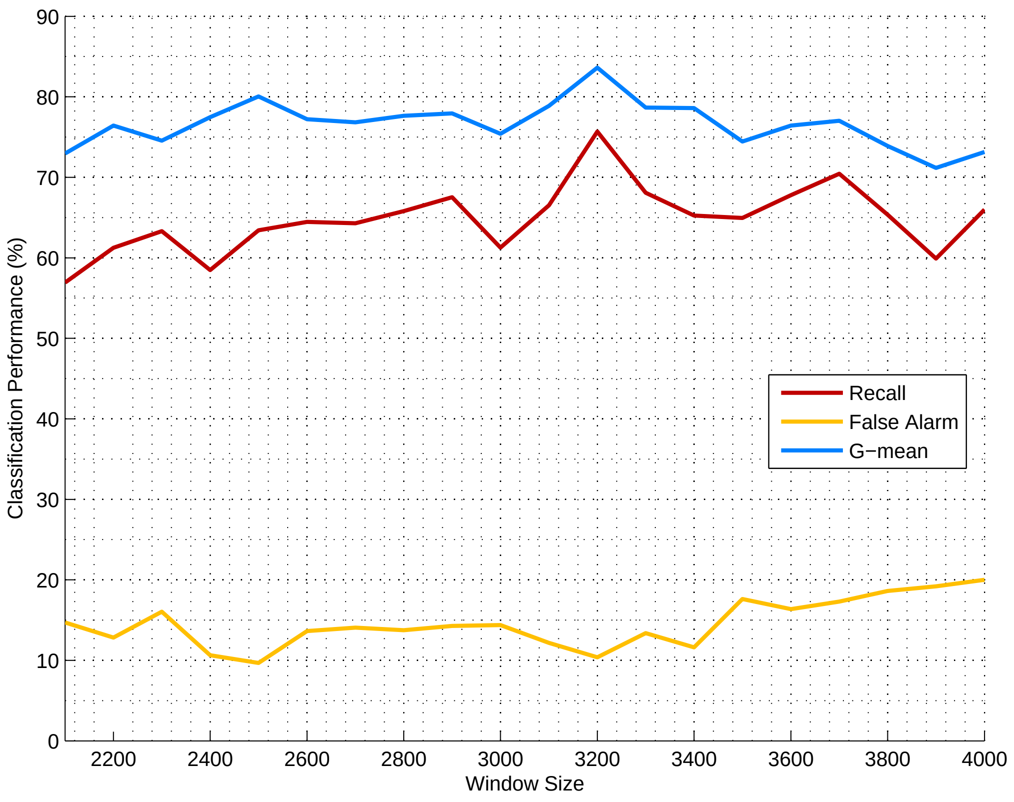


Figure 4.13 *The effect of the window size for OPTICS-based approach. (Configuration in Table 4.2 with 10 principal components and MinPts = 40.)*

methods such as k-means. By 'simpler', it is meant that the number of parameters to be controlled is reduced since for instance, k-means only requires the number of clusters and in the case of anomaly detection it is equal to two. Hence, autoencoder offers practical advantage. In figures 4.17 and 4.18, the clustering results are given by histograms. These histograms represent the distribution of precision and recall in 100 runs of the autoencoder scheme explained earlier. They show that if the network's parameters are randomly initialized to good starting points, then gradient descent algorithm can find a good local minimum by which the classes can be well-separated. The experiments have shown that the performance of the OPTICS-based approach can be boosted by this representation. It has been observed that 100% precision, 90.45% recall and 0% false alarm rate can be achieved simultaneously when $MinPts$ is set to 300. Also, naive Bayesian approach provides very similar results whereas density-based approach fails to identify the anomalies as the basic assumption on outliers is lost with this representation.

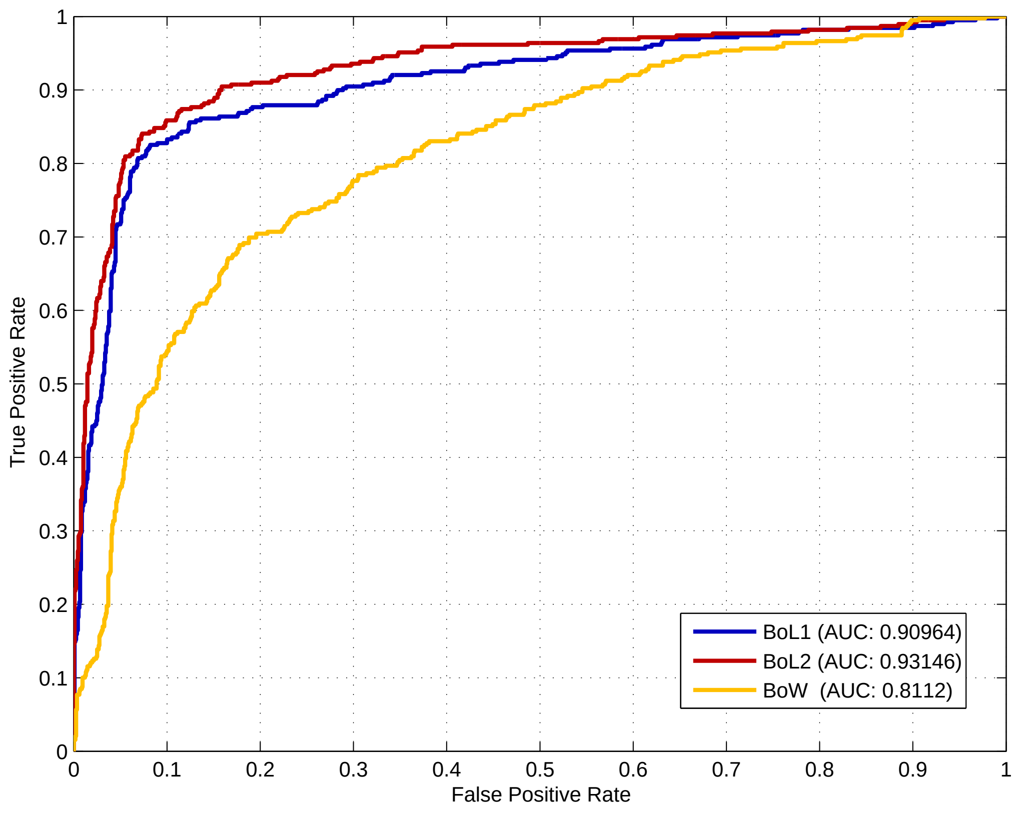


Figure 4.14 *Comparison of feature types*

Parameter	Value
Visible Layer Size (1st Network)	90
Hidden Layer Size (1st Network)	45
Weight Decay (λ)	0.01
Sparsity (ρ)	0.4
Sparsity Penalty (β)	3
Visible Layer Size (2nd Network)	45
Hidden Layer Size (2nd Network)	3
Weight Decay (λ)	0.001
Sparsity (ρ)	0.1
Sparsity Penalty (β)	3

Table 4.3 *The network parameters used in autoencoder scheme(Maximum number of iterations is 1000 for both.)*

4.7 Results for SVDD

As explained in Section3.3.4, SVDD method has been observed to fail with the dataset TTY2-500k as the proportion of the anomalies are not neglible compared to general anomaly problems. However, considering its benefits in reducing time and space complexities in large databases, it is desirable to devise a method exploiting

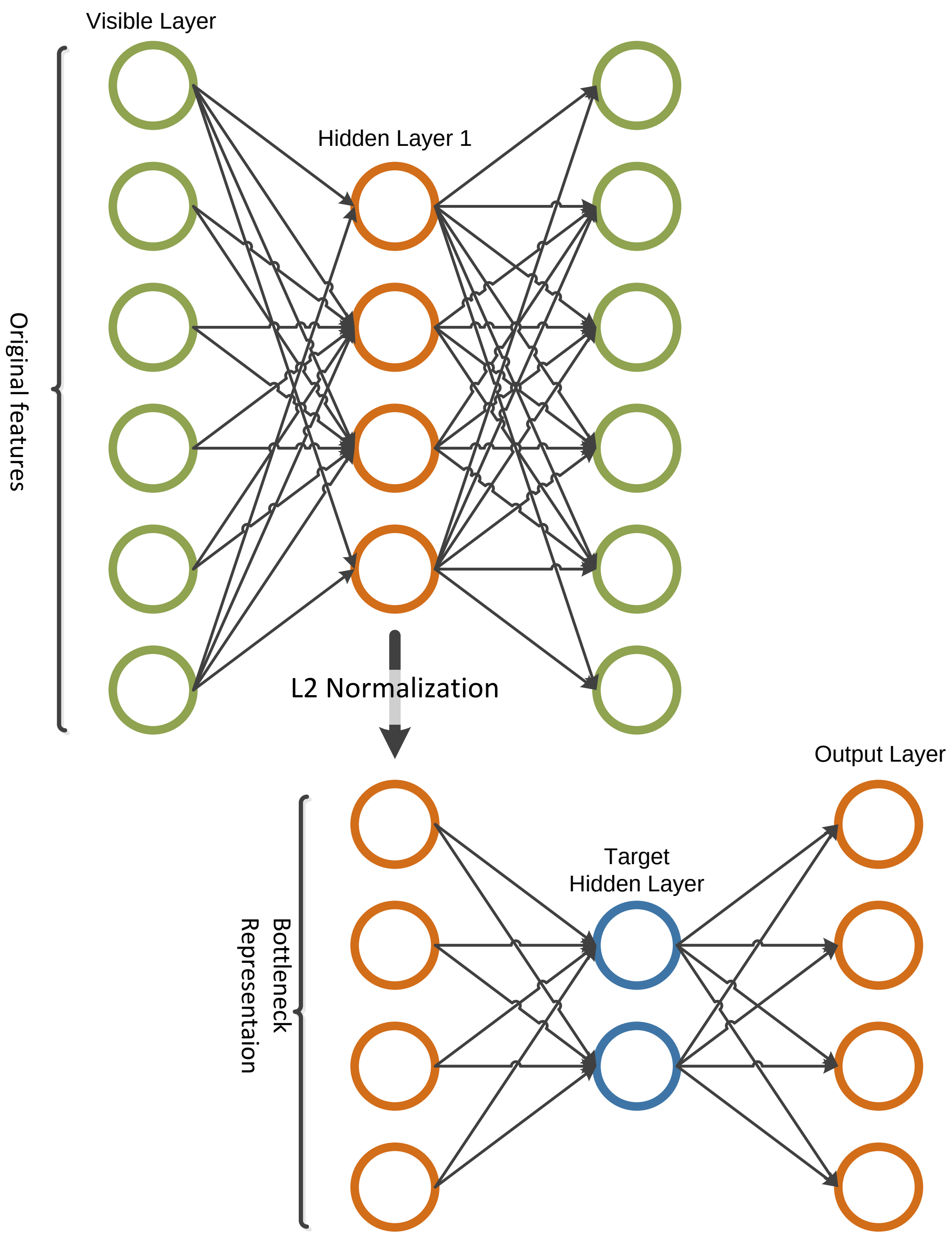


Figure 4.15 *The dimensionality reduction scheme using autoencoders*

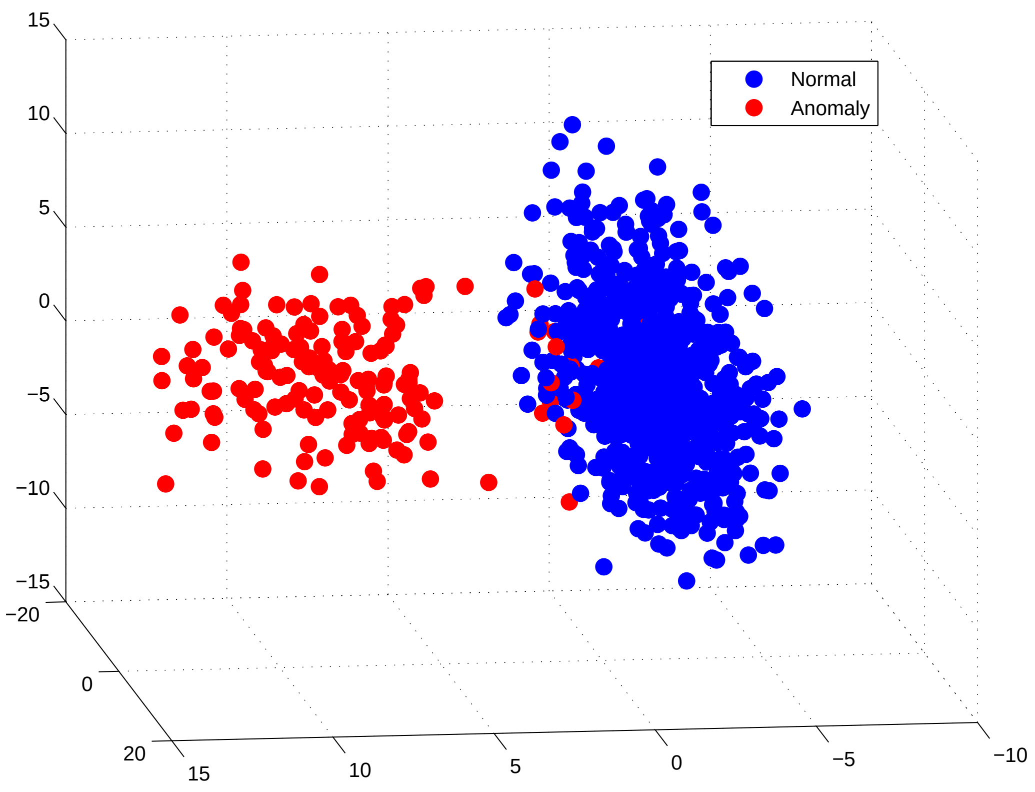


Figure 4.16 *The representation of the TTY dataset by the autoencoder network when the window size is 1500.*

its properties. The given two-tier anomaly removal framework can be used to see its performance. Table 4.4 shows the distribution of classes in terms of numbers in the input and output of the framework. Table shows that the anomalies composes one quarter of the whole dataset. This is clearly excessive for a good description of the target (normal) class. After the removal, the cleaned dataset mostly constitutes of the normal instances where only 7.9% of it belongs to anomaly class.

To obtain this result,

- Naive Bayesian classifier was given the whole dataset with 3 principal components and $\mathcal{T}_{normal}$ equal to mean score in the first tier.
- Density-based approach was given the output of first tier with 2 principal components. ($\mathcal{T}_{anomaly}$ was set to the mean value of all obtained scores.)

KSVDD with $\sigma = 2$ resulted in very high precision and recall values (on average) for the unseen data in 10-fold cross-validation test. Fig. 4.19 shows the average performance values over several values of parameter C. The best average (95% of g-mean) occurs when the parameter C is 0.3 and $\sigma = 2$ and larger values for C, the G-mean decreases implying that normal instances are confused with the anomalies

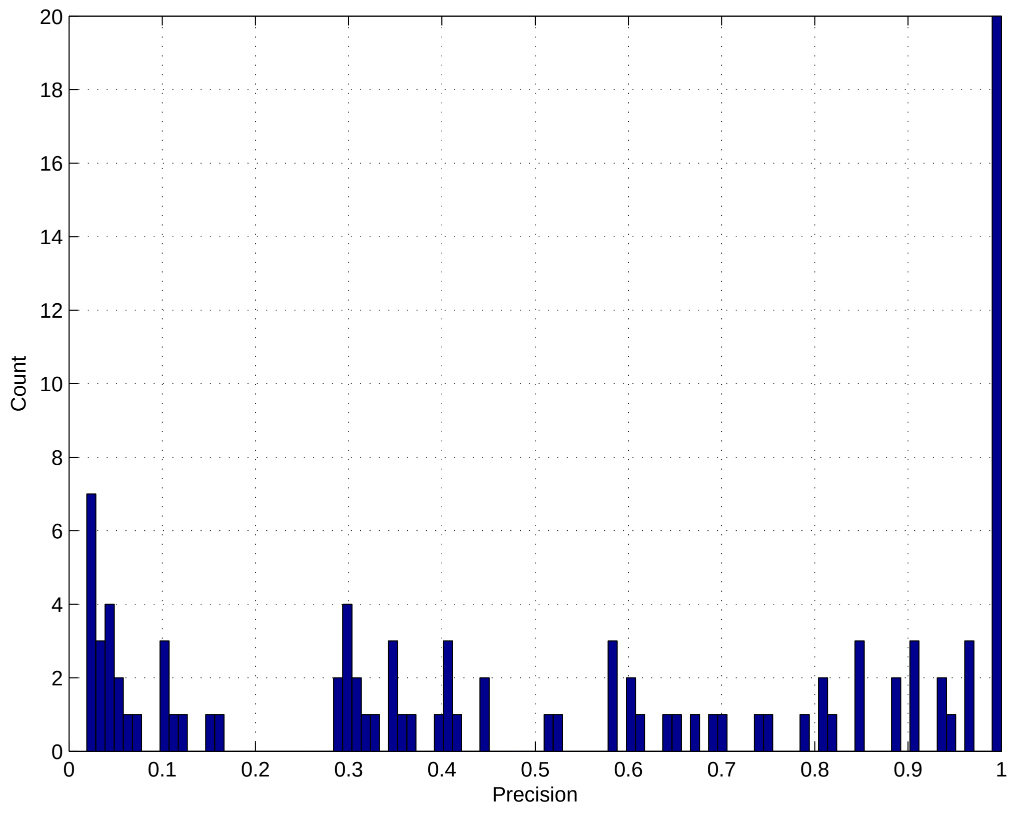


Figure 4.17 *The distribution of k-means clustering precision in 100 runs of autoencoder system (k=2)*

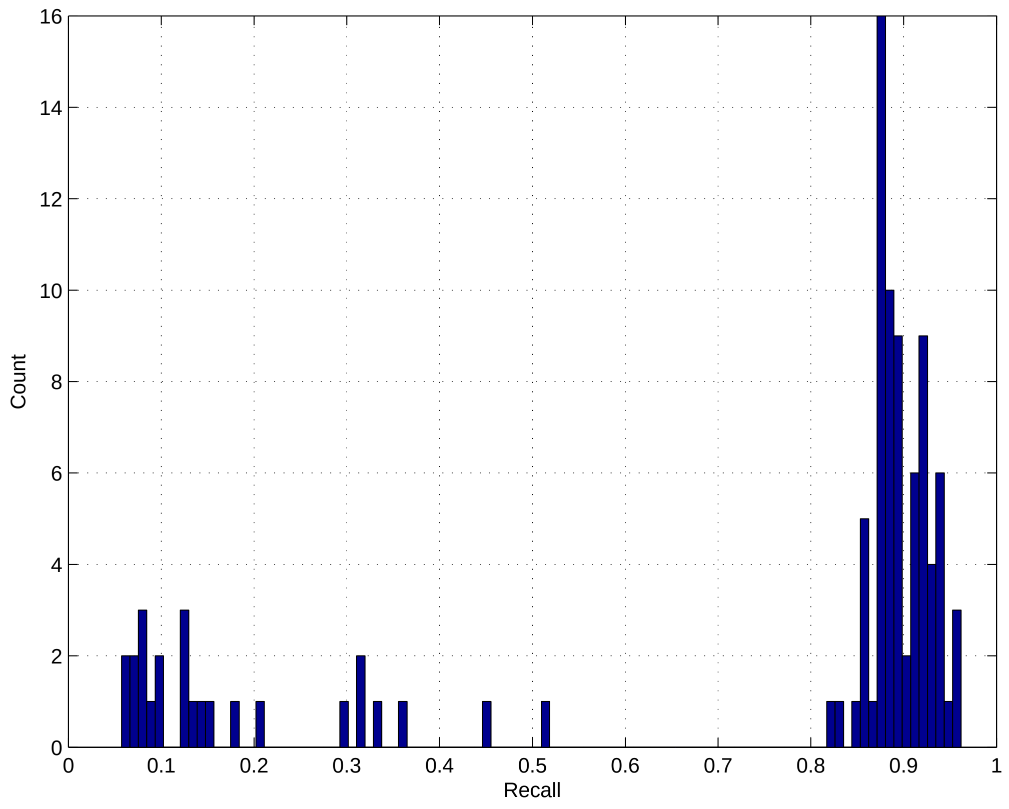


Figure 4.18 *The distribution of k-means clustering recall in 100 runs of autoencoder system (k=2)*

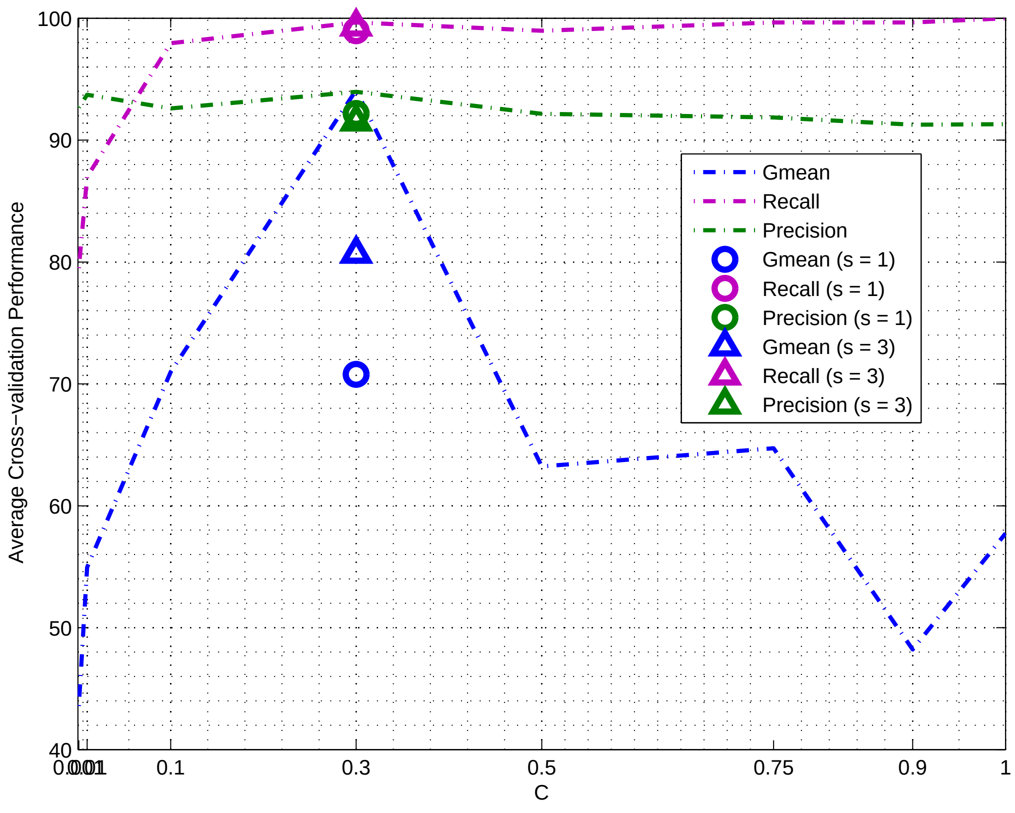


Figure 4.19 *Average classification results for KSVDD in 10-fold cross-validation ($s = 2$). The circles and triangles represent the classification results with $s = 1$ and $s = 3$, respectively (when C is constant at 0.3).*

(same is valid for smaller values as well). Additionally, the RBF parameter σ affects the g-mean value negatively as it is increased further than 2. It is also possible to obtain similar results with linear SVDD. One advantage of linear SVDD would be the reduced testing time since it only requires to compare the distance of object to center point $\boldsymbol{a}$. However, since KSVDD provides more flexible boundaries it was opted for in this section.

Data	Total	No. of Normal	No. of Anomaly	Anomaly Rate
Input	1529	1160	389	**25.1**%
Output	1221	1124	97	**7.9**%

Table 4.4 *The number of samples from each class before and after the anomaly removal.*

4.8 Performance Comparison

In this section, a general comparison of the mentioned methods in terms of performance will be given. However, the fact that it is not fair to compare methods under the same configurations leads to compare the 'best' of each. For defining 'best', one should think in the anomaly detection context. What is in general desired in such systems is low false alarm rate (FAR), high precision and recall in the order of priority. The configurations which minimize FAR and correspond to the best trade-off between precision-recall are considered to be the best. Table 4.5 and Table 4.6 include some of the best results recorded. Although there exist other configurations which can possibly deliver similar results, the Tables 4.5 and 4.6 are used to show the limits. On TTY dataset which includes only one type of fault, the Naive Bayesian approach and density-based method performed very well. OPTICS-based approach (OPTICS-OF) failed to provide as high precision as its counterparts with small values of $MinPts$ because of the distribution of the outliers. Even though increasing the minimum size of cluster ($MinPts = 200$) resulted in very satisfactory results similar to its counterparts, it was outperformed by the other two approaches in terms of recall. As for TTY2 dataset, the results showed that there is a significant decrease in the recall compared to TTY dataset. Still, the methods (NB and DB) could provide zero false alarm. OPTICS-based approach shows improvement in terms of recall but it is outperformed by the other methods by 20% in precision.

(TTY)	**Precision**	**Recall**	**F. Alarm**	**Gmean**	**Configuration**
NB	99.53	97.69	0.0778	99.57	BoL2, 1000, 4PC
DB	99.69	96.30	0	99.69	BoL2, 1250, 3PC
OPTICS	100	90.45	0	99.20	BoL, 1500, 3 (Autoencoder)

Table 4.5 *Comparison of classification performances for TTY dataset*

(TTY2)	**Precision**	**Recall**	**F. Alarm**	**Gmean**	**Configuration**
NB	100	75.40	0	95.03	BoL2, 1500, 8PC
DB	100	74.32	0	94.1	BoL2, 1750, 5PC
OPTICS	80.63	79.07	11.04	84.22	BoL2, 2200, 19PC

Table 4.6 *Comparison of classification performances for TTY2 dataset*

5. DISCUSSIONS AND CONCLUSIONS

The work of this thesis is related to anomaly detection in unstructured system logs while the methods used vary in terms of the approach. Of numerous classifiers proposed in the literature, the emphasis is set on naive Bayesian approach and density-based approaches as well as well-established support vector machines. First, a set of feature extraction methods (i.e., BoW, BoL) are discussed in terms of representation power. Secondly, dimensionality reduction methods such as PCA and a type of auto-associative neural networks -autoencoder- are investigated. Finally, a pattern recognition system where unsupervised methods are used to clean the anomalies in this type of datasets are shown so that some of the powerful supervised methods such as Support Vector Data Description (SVDD) can be used without the need for labels.

Two different data representation methods based on dictionaries are investigated namely, bag-of-words and bag-of-lines (bag-of-strings). The former is a commonly used representation form for textual data while the latter one is designed specifically for this type of data. It has been shown that bag-of-lines provide a better representation of the distributions compared to bag-of-words as the performance is observed to be affected significantly by the choice.

Another observations was that dimensionality reduction is crucial in most of the cases as the original feature set may contain redundancies and highly-correlated subsets of features which deteriorate the representation and classification performance. PCA proved to be a powerful tool for this purpose. Alternatively, autoencoders have produced comparable results to baseline PCA. However, as with all neural network models, the training of the autoencoder needs special care as the initialization of the network parameters affects the results significantly. There exists possible methods to conduct pre-training which could guarantee of reaching at better local minima points however it was out of the scope of this thesis work.

As anomaly detection can take many forms due to the application and the type of anomalies residing in the data, the methods designed to work well against certain types of anomalies do not necessarily bring about very good results. Therefore, most of the methods are tailored to work well against the anomalies in our data sets. This required well understanding of the methods and the data sets themselves. The density-based methods have shown good results especially when the general density values were considered rather than local densities (data set specific conclusion). Also, it is shown that if the feature distributions are well-studied, one can benefit from what probabilistic density estimation approaches can provide in an unsupervised way. Another result is that traditional methods such as k-means, mean-shift and spectral clustering do not perform well as anomalies do not exhibit either a very compact cluster or a completely sparse distribution. Therefore, density-based detection methods such as LOF and OPTICS-OF are adapted.

Although the autoencoders were not under the spotlight as much as PCA was, the neural networks offer great possibilities. The representations learned by autoencoders can lead to different results. However, the behavior of neural networks in relation with the parameters should be studied and understood well. The initialization of especially deep architectures is extremely important since the optimization algorithms such as gradient-descent can lead to different local minimas (not necessarily to the global minima) in multi-modal objective function landscapes. Further research must be conducted on automatic feature learning from textual data, as in convolutional networks for images, since representations such as the histogram-based features are very limiting. Also, in order to deal with anomalies in very large volumed datasets, further research has to be conducted on modeling normal class in a compact way (e.g. using few support vectors) when labels are not available (unsupervised).

REFERENCES

[1] C. C. Aggarwal, "On abnormality detection in spuriously populated data streams," in *In Proceedings of ACM SIAM Conference on Data Mining*, 2005.

[2] E. Aleskerov, B. Freisleben, and B. Rao, "Cardwatch: a neural network based database mining system for credit card fraud detection," in *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*, Mar 1997, pp. 220–226.

[3] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic detection and tracking pilot study final report," in *In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 194–218.

[4] Anderson, Lunt, Javitz, Tamaru, and Valdes, "Detecting unusual program behavior using the statistical components of NIDES," may 1995. [Online]. Available: http://www.csl.sri.com/papers/5sri/

[5] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," *arXiv preprint cs/0006013*, 2000.

[6] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, June 1999. [Online]. Available: http://doi.acm.org/10.1145/304181.304187

[7] M. F. Augusteijn and B. A. Folkert, "Neural network classification and novelty detection," *International Journal of Remote Sensing*, vol. 23, no. 14, pp. 2891–2902, 2002.

[8] D. Barber, *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

[9] R. J. Bolton, D. J. Hand, *et al.*, "Unsupervised profiling methods for fraud detection," *Credit Scoring and Credit Control VII*, pp. 235–255, 2001.

[10] M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Optics-of: Identifying local outliers," 1999.

[11] ——, "Lof: Identifying density-based local outliers," in *PROCEEDINGS OF THE 2000 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*. ACM, 2000, pp. 93–104.

[12] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, June 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1009715923555

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, July 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[14] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273–297.

[15] M. Daszykowski, "MATLAB implementation of OPTICS," http://chemometria.us.edu.pl/download/OPTICS.M.

[16] M. Daszykowski, B. Walczak, and D. L. Massart, "Looking for natural patterns in analytical data, 2. tracing local density with optics." *Journal of Chemical Information and Computer Sciences*, vol. 42, no. 3, pp. 500–507, 2002. [Online]. Available: http://dblp.uni-trier.de/db/journals/jcisd/jcisd42.html#DaszykowskiWM02

[17] M. Davy and S. Godsill, "Detection of abrupt spectral changes using support vector machines an application to audio signal segmentation," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 2, May 2002, pp. II–1313–II–1316.

[18] C. Diehl and I. Hampshire, J.B., "Real-time object classification and novelty detection for collaborative video surveillance," in *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, vol. 3, 2002, pp. 2620–2625.

[19] S. Donoho, "Early detection of insider trading in option markets," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 420–429. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014100

[20] L. Ertöz, M. Steinbach, and V. Kumar, "Finding topics in collections of documents: A shared nearest neighbor approach," in *Clustering and Information Retrieval*, 2003, pp. 83–104.

[21] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 255–262. [Online]. Available: http://dl.acm.org/citation.cfm?id=645529.658128

[22] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[23] T. Fawcett and F. Provost, “Activity monitoring: Noticing interesting changes in behavior,” in *In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 53–62.

[24] B. Gautam, P. Katara, S. Singh, and R. Farmer, “Drug target identification using gene expression microarray data of toxoplasma gondii,” *International Journal of Biometrics and Bioinformatics (IJBB)*, vol. 4, no. 3, p. 113, 2010.

[25] A. Ghosh, J. Wanken, and F. Charron, “Detecting anomalous and unknown intrusions against programs,” in *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*, Dec 1998, pp. 259–267.

[26] S. Hawkins, H. He, G. Williams, and R. Baxter, “Outlier detection using replicator neural networks,” in *Data warehousing and knowledge discovery.* Springer, 2002, pp. 170–180.

[27] D. B. T. Hofmann, L. D. Baker, T. Hofmann, A. K. Mccallum, and Y. Yang, “A hierarchical probabilistic model for novelty detection in text,” 1999.

[28] G. Hollier and J. Austin, “Novelty detection for strain-gauge degradation using maximally correlated components,” 2002, pp. 257–262.

[29] W. Hu, Y. Liao, and V. R. Vemuri, “Robust anomaly detection using support vector machines,” in *In Proceedings of the International Conference on Machine Learning.* Morgan Kaufmann Publishers Inc.

[30] A. Jagota, “Novelty detection on a very large number of memories stored in a hopfield-style network,” in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. ii, Jul 1991, pp. 905 vol.2–.

[31] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, “Learning convolutional feature hierarchies for visual recognition.”

[32] E. Keogh, S. Lonardi, and B. Y.-c. Chiu, “Finding surprising patterns in a time series database in linear time and space,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’02. New York, NY, USA: ACM, 2002, pp. 550–556. [Online]. Available: http://doi.acm.org/10.1145/775047.775128

[33] C. Krügel, T. Toth, and E. Kirda, "Service specific anomaly detection for network intrusion detection," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, ser. SAC '02. New York, NY, USA: ACM, 2002, pp. 201–208. [Online]. Available: http://doi.acm.org/10.1145/508791.508835

[34] V. Kumar, "Parallel and distributed computing for cybersecurity," *Distributed Systems Online, IEEE*, vol. 6, no. 10, pp. –, 2005.

[35] P. Kumpulainen and K. Hätönen, "Local anomaly detection for network system log monitoring," in *Proceedings of the 10th International Conference on Engineering Applications of Neural Networks*, 2007, pp. 33–44.

[36] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems 2, NIPS 1989.* Morgan Kaufmann Publishers, 1990, pp. 396–404.

[37] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning.* ACM, 2009, pp. 609–616.

[38] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11222-007-9033-z

[39] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[40] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *the Journal of machine Learning research*, vol. 2, pp. 139–154, 2002.

[41] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: a statistical anomaly approach," *Communications Magazine, IEEE*, vol. 40, no. 10, pp. 76–82, Oct 2002.

[42] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, and C. Suen, "UFLDL Tutorial," *http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial*, 2010.

[43] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[44] M. Ramadas, S. Ostermann, and B. Tjaden, "Detecting anomalous network traffic with self-organizing maps," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, G. Vigna, C. Kruegel, and E. Jonsson, Eds. Springer Berlin Heidelberg, 2003, vol. 2820, pp. 36–54. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45248-5_3

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Learning Representations by Back-propagating Errors, pp. 696–699. [Online]. Available: http://dl.acm.org/citation.cfm?id=65669.104451

[46] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A bayesian approach to filtering junk E-mail," in *Learning for Text Categorization: Papers from the 1998 Workshop.* Madison, Wisconsin: AAAI Technical Report WS-98-05, 1998. [Online]. Available: citeseer.ist.psu.edu/sahami98bayesian.html

[47] A. J. Smola, B. Scholkopf, and K.-R. Muller, "The connection between regularization operators and support vector kernels." *Neural Networks*, vol. 11, no. 4, pp. 637–649, 1998. [Online]. Available: http://dblp.uni-trier.de/db/journals/nn/nn11.html#SmolaSM98

[48] Q. Song, W. Hu, and W. Xie, "Robust support vector machine with bullet hole image classification," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 32, no. 4, pp. 440–448, Nov 2002.

[49] A. Srivastava, "Enabling the discovery of recurring anomalies in aerospace problem reports using high-dimensional clustering techniques," in *Aerospace Conference, 2006 IEEE*, 2006, pp. 17 pp.–.

[50] A. Srivastava and B. Zane-Ulman, "Discovering recurring anomalies in text reports regarding complex space systems," in *Aerospace Conference, 2005 IEEE*, March 2005, pp. 3853–3862.

[51] M. Taniguchi, M. Haft, J. Hollmen, and V. Tresp, "Fraud detection in communication networks using neural and probabilistic methods," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 2, May 1998, pp. 1241–1244 vol.2.

[52] D. Tax and R. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004. [Online]. Available: http://dx.doi.org/10.1023/B%3AMACH.0000008084.60811.49

[53] D. Tax, "Data description toolbox dd tools 2.0.0," 2013.

[54] V. N. Vapnik, *Statistical Learning Theory.* Wiley-Interscience, 1998.

[55] A. Webb and K. Copsey, *Statistical Pattern Recognition.* Wiley, 2011.

[56] R. Xu and D. Wunsch, II, "Survey of clustering algorithms," *Trans. Neur. Netw.*, vol. 16, no. 3, pp. 645–678, May 2005. [Online]. Available: http://dx.doi.org/10.1109/TNN.2005.845141

[57] A. Ypma, E. Ypma, and R. P. Duin, "Support objects for domain approximation," in *In ICANN'98, Skovde (Sweden).* Springer, 1998, pp. 2–4.