CERIAS Tech Report 2019-4
Adversarial Anomaly Detectio
by Radhika Bhargava
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

ADVERSARIAL ANOMALY DETECTION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Radhika Bhargava

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Chris Clifton, Co-Chair

School of Computer Science

Dr. Shimon Y. Nof, Co-Chair

School of Industrial Engineering

Dr. Jean Honorio

School of Computer Science

Dr. Ninghui Li

School of Computer Science

Approved by:

Dr. Voicu Popescu

Head of the School Graduate Program

Dedicated to my parents.

ACKNOWLEDGMENTS

As I look back at the time spent at Purdue University, I am reminded of the many people who helped me in making this journey a success. First and foremost, I would like to thank my advisor Prof. Cris Clifton for the Invaluable guidance, encouragement, and support throughout my graduate studies at Purdue University. He has given me the freedom to pursue various projects without objection and helped me build analytical skills and critical thinking that are essential in research.

I would also like to express gratitude to my advisor Prof. Nof for his mentorship, encouragement and advice at many different times. His endless guidance is hard to forget throughout my life.

I sincerely thank my thesis committee members Prof. Jean Honorio and Prof. Ninghu Li for their insightful feedback. Their valuable comments and suggestions helped in making this dissertation much stronger. I would like to thank Prof. Elias Barenboim and Prof. Pedro Fonseca for their time, interest, and helpful comments.

I gratefully acknowledge the funding received towards my PhD from the Northrop Grumman Corporation. I am also pleased to thank Jason Kobes for his constructive feedback.

I would like to express my gratitude to all past and present PRISMers. It has been, and will continue to be, a pleasure to be part of such a diverse team of highly talented individuals at PRISM. A special mention to Dr. Rodrigo E. Reyes Levalle and Dr. Hao Zhong for their support at the beginning of my career as a researcher.

Thanks to all my friends at Purdue who have made my time here more enjoyable. A special thanks to Parag for your friendship and for being a constant source of encouragement.

My deepest gratitude to my parents for their unwavering support, love and encouragement throughout the years. Many thanks to my aunt, sister, brother-in-law

and nephews for their love and support. This thesis would not have been possible without the strength and love I received from my husband Mridul. Thank you for being with me through the tough times during this journey.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Bhargava, Radhika PhD, Purdue University, August 2019. Adversarial Anomaly Detection. Major Professor: Chris Clifton.

Considerable attention has been given to the vulnerability of machine learning to adversarial samples. This is particularly critical in anomaly detection; uses such as detecting fraud, intrusion, and malware must assume a malicious adversary. We specifically address *poisoning attacks*, where the adversary injects carefully crafted benign samples into the data, leading to concept drift that causes the anomaly detection to misclassify the actual attack as benign. Our goal is to estimate the vulnerability of an anomaly detection method to an unknown attack, in particular the expected minimum number of poison samples the adversary would need to succeed. Such an estimate is a necessary step in risk analysis: do we expect the anomaly detection to be sufficiently robust to be useful in the face of attacks? We analyze DBSCAN, LOF, one-class SVM as an anomaly detection method, and derive estimates for robustness to poisoning attacks. The analytical estimates are validated against the number of poison samples needed for the actual anomalies in standard anomaly detection test datasets. We then develop defense mechanism, based on the concept drift caused by the poisonous points, to identify that an attack is underway. We show that while it is possible to detect the attacks, it leads to a degradation in the performance of the anomaly detection method. Finally, we investigate whether the generated adversarial samples for one anomaly detection method transfer to another anomaly detection method.

1 INTRODUCTION

Anomalies are data patterns that deviate from the normal behavior. Anomaly detection is important because the anomalous items translate into significant and actionable information in a wide variety of application domains [1]. E.g., in credit card data, anomalies signify that there has been a fraudulent transaction or identity theft has taken place. Anomalies in medical sensor data can be monitored to provide preemptive health warnings [2], or vibrations in a machine could be evidence that the machine is faulting [3]. Anomaly detection techniques are also used to monitor network traffic and identify intrusions [4]. Anomalies in diagnostic medical images or health monitoring sensors could indicate the presence of diseases [5]. In industrial anomaly detection, sensor data can be used for early identification of faults in manufacturing systems thereby preventing disruption of the systems [6].

A number of anomaly detection systems have been developed based on unsupervised and supervised machine learning techniques such as support vector machines, clustering, neural networks etc. The difference between unsupervised and supervised techniques is that unsupervised do not require labelled data but supervised does. Unsupervised techniques are especially beneficial in the anomaly detection domain; as there is a class imbalance between normal instances and anomalies in the training data. Anomalies are significantly less as compared to the normal training data instances. These techniques are often based on traditional machine learning techniques, such as classifiers that classify the event as an anomaly or normal data [7], or clustering methods where an item not falling in a cluster is considered an anomaly. A broad classification of anomaly detection approaches includes nearest-neighbor, clustering, classification, statistical, information-theoretic and spectral [1].

Classification techniques learn a discriminating boundary between the normal class and anomalous class, in the given feature space. Nearest neighbor techniques assume

that normal data lies in dense neighborhood whereas anomalous data occurs far away from their closest neighbor. Clustering techniques assume that normal belongs to a cluster whereas anomalies are not part of any cluster. In statistical techniques, normal data belongs to a high probability region of a statistical model whereas anomalies belong to a low probability region. Information theoretic approaches assume that the data irregularities are anomalies. The approach adopted by spectral anomaly detection technique is to embed data in a lower dimension space (e.g., using eigenvalues and eigenvectors) where anomalies and normal data points are different.

While anomalies can occur for many benign or malicious reasons, in security applications of anomaly detection; the "interesting" anomalies are the consequence of malicious actions (fraudulent credit transactions, intrusion attempts, malware, etc.). As a result, we should be concerned that an adversary may actively try to defeat the anomaly detection system. Security applications (e.g., credit card fraud detection, intrusion detection, etc.), of machine learning assume the presence of an adversary who is actively trying to circumvent the machine learning algorithms. This has led to the growth of studying the security of machine learning algorithms, i.e., adversarial machine learning - the study of the efficacy of machine learning algorithms in the presence of an adversary. Adversaries are no stranger to the fact that machine learning tools are being applied to security domains and are actively looking for opportunities to subvert these tools by exploiting the assumptions that the practitioners make (e.g., data points in the dataset are independent, dynamic training of machine learning algorithms). Whenever anomalous behavior is a result of malicious adversaries; the malicious adversary adapts himself so as to look more benign.

1.1 Motivation

Machine learning algorithms have been shown to be vulnerable to adversarial examples, i.e., carefully crafted malicious samples that are imperceptible to humans, but can alter the classifications of the machine learning algorithm. For classification,

the goal of the crafted adversarial samples is bypass the target model; for instance, spam emails that are classified as legitimate by a spam filter, anomalies that are detected as normal. These input modifications or perturbations are introduced by an adversary to yield specific - targeted misclassification. Adversarial samples have been shown to circumvent intrusion detection [8], fraud detection, or to mislead autonomous navigation systems [9].

Anomaly detection systems are used for security purposes like intrusion detection, fraud detection, virus detection. We can assume that whenever machine learning approaches are used to provide security against illicit activities, an adversary will try to circumvent these measures. These systems are used to adapting the security tools to the dynamic properties of the data by being trained and retrained on new data. This adaptability property of the anomaly detection approaches makes them vulnerable to the manipulation of the learning environment by an adversary. E.g., an adversary may inject fake data to modify the decision boundary of the algorithm [10–14]. The main challenge addressed in this paper is that an adversary tries to actively manipulate the data to escape detection [1].

Security (encompassing the security of machine learning techniques) is an arms race [14, 17]. Spam filtering is an example of an "arms race" that is characterized by increasing sophistication at the end of both the spammer and the defender. The defender and the attacker aim to accomplish their objectives by modifying their behavior in response to the strategy of their adversary . Anomaly detection is no different - attackers evade anomaly detection systems (intrusion detection, credit card fraud detection) [18] by developing new attack vectors to circumvent known defense mechanisms.

To stay ahead in this arms race we need to follow these three principles [19]-

1. Know your adversary - "If you know the enemy and know yourself, you need not fear the result of a hundred battles" - Sun Tzu. If you know your adversary

[1]Note that we are using adversarial machine learning in the sense of [15]. There has been recent use of the term adversarial in the sense of using two machine learning models to train each other, e.g., [16]; this is significantly different from the issue addressed in this paper

which encompasses his knowledge, capabilities and goals you will be able to formulate a better defense mechanism. This involves building a threat model which characterizes the adversary.

2. Be proactive - This entails testing your defense mechanism against all possible attacks that you can develop given the threat model

3. Defense mechanism - Robust techniques can be built if you know your adversary and his capabilities. Designing a threat model and being proactive is a precursor to building more robust mechanisms.

In this thesis, we assume that the adversary can poison the input training data and analyze the vulnerability of anomaly detection approaches to this attack. The goal of these samples is that given a malicious sample 'a', the adversary wants to add poison samples so that the classification of the malicious sample 'a' changes to benign by altering the decision boundary of the machine learning algorithm. This kind of attack is relevant in anomaly detection where the adversary is actively trying to escape detection but wants his sample to be unaltered. Consider a scenario - a person has stolen a credit card and wants to make a large purchase. Input perturbations would require the adversary to alter his spending amount so as to be misclassified which defeats the purpose of the adversary. The adversary would rather induce fake data sets so that he can disguise himself among benign-looking data points.

1.2 Thesis Statement & Research Questions

We hypothesize, that given an anomaly detection approach, and an adversary who poisons the training data to try to cause a target attack point to be misclassified as benign, we can:

1. estimate the expected minimum effort (number of poison samples) the adversary would need to inject, and

2. adapt anomaly detection techniques to increase resistance to such a poisoning attack. These can be done by the defender knowing only the general distribution of adversarial attacks, with no prior knowledge of a specific attack.

Based on this hypothesis our thesis answers the following questions:

1. Given that an adversary knows that the underlying algorithm is, what is the optimal attack strategy for a targeted attack and can we bound the adversary's effort?

2. Can we predict the vulnerability of the anomaly detection methods under this attack?

3. Can we detect these attacks?

1.3 Contributions

We first present a model to analyze the security of different anomaly detection approaches. We look at classification, clustering and nearest-neighborhood techniques to derive bounds on the attacker's effort required to subvert the anomaly detection approach. We derive defense mechanism to improve the robustness of the techniques. We then investigate the transferability of adersarial samples.

1.3.1 Security of DBSCAN

DBSCAN [20] is a popular density based clustering algorithm for anomaly detection and has been effectively applied in a variety of domains [21–23]. The method assumes that the normal data lies in high density regions whereas anomalies or outliers are the points that lie in low density regions. One of the advantage of the density based method is that it is good at discovering clusters of arbitrary shapes making it one of the dominant and ubiquitous technique in anomaly detection.

DBSCAN starts with an arbitrary object in the dataset and checks the number of objects within a given radius. If the number of objects with in that radius are

more than the threshold required for a cluster, it is marked as a normal point and if the number of objects in within the given radius are less than the minimum number of objects required, then the data point is marked as anomaly. DBSCAN is a global density based algorithm, i.e., the results depend on the global density threshold settings.

Our contributions include proposing a framework to address situations where the adversary is unable to change their own data, but *can* create fake entities to attempt to make their actual data look less like an anomaly. We then evaluate how vulnerable DBSCAN-based anomaly detection is to such an attack and have answered the following questions:

1. Given, that we know our anomaly detection approach is DBSCAN what is the adversary's lowest-cost attack strategy?

2. Can we bound the adversary's effort w.r.t to the attack strategy?

3. In the face of an unknown attack, can we build a model which can estimate the adversary's effort and quantify the degradation in the vulnerability?

4. Can we develop a detection technique which can be used as a counter measure for this attack?

1.3.2 Security of LOF

The nearest-neighborhood techniques like Local Outlier Factor [24](LOF), is a technique developed specifically for anomaly detection, rather than an application of a different machine learning approach. It classifies anomalies based on its local neighborhood and uses either the distance to its β^{th} nearest neighbor or the relative density of each data point to compute the anomaly score. The assumption of a nearest neighborhood technique is that normal data lies in dense neighborhoods whereas anomalous data occurs far from their neighbors (they have a lower density as compared to their neighbors). LOF has been applied to anomaly detection in a variety

of domains [25, 26]. The main advantage of LOF is that it is capable of detecting outliers in a dataset even if it contains clusters of different densities.

DBSCAN is a global density algorithm whereas LOF is a nearest neighbor algorithm. The difference between LOF and DBSCAN is that DBSCAN decides anomalies based on global parameters whereas LOF decides based on the relative density of its neighborhood.

Our contributions in analyzing the security of LOF includes developing an optimal attack strategy by exploiting the assumptions that a practitioners make (e.g., independence, weak stochastic properties of the data) and modeling vulnerability to this attack. We have answered the following questions -

1. What is the optimal attack strategy for a targeted attack on LOF and can we bound the adversary's effort?
2. Can we quantify the degradation in the performance of LOF to this attack?
3. Can we make LOF robust to this attack?
4. Given that DBSCAN and LOF are both density-based techniques, can the attack samples/strategy be transferred?

1.3.3 Security of One-Class SVM

Classification based anomaly detection techniques learn a classifier from the given observations and then classifies a given data point as normal or anomalous. These techniques operate under the assumption that they can train the classifier to distinguish between anomalous and normal data from the given feature space. Examples include neural networks, support vector machines (SVM) etc. A number of variants of support vector machines have been developed for anomaly detection in temporal sequences [27], system call intrusion detection [28, 29]. One-class SVM [30] was developed for anomaly detection and it assumes that the training data belongs to only

one-class, i.e., the normal class. Outliers are any data points that do not belong to this class.

Our contributions include developing an optimal attack strategy by exploiting typical assumptions (e.g., independence, weak stochastic properties of the data) and modeling vulnerability to this attack. We have answered the following questions:

1. Given that an adversary knows that the underlying algorithm is one-class SVM, what is the optimal attack strategy for a targeted attack and can we bound the adversary's effort?

2. Can we quantify the degradation in performance and vulnerability of one-class SVM under this attack?

1.4 Organization of the Thesis

The thesis is organized as follows: in Chapter 2 we give an overview of the related work on anomaly detection techniques and approaches for security evaluation of different machine learning algorithms. Chapter 3 presents an attack model that has been used to develop optimal attack strategies for DBSCAN, LOF and one-class SVM. The security analysis of DBSCAN has been conducted in Chapter 4. This includes developing an optimal attack strategy, bounding the efforts of the adversary, assessing the vulnerability of DBSCAN to this attack and presenting a detection strategy for the given attack strategy. Similar analysis has been conducted for LOF (Chapter 5) and one-class SVM (Chapter 6). We have then analyzed the transferability of the adversarial samples generated for the poisoning attack among the three different techniques in Chapter 7. Finally, our conclusions and future work is presented in Chapter 8.

2 RELATED WORK

This chapter includes an overview of the related work on anomaly detection techniques and approaches for security evaluation of different machine learning algorithms.

2.1 Anomaly Detection

Anomaly detection techniques consist of two phases, training and testing [31]. In the training phase, models of normal behavior are derived from unlabelled or labelled training data. In the testing phase, the models learned are queried to identify whether the new data is anomalous or not. Anomaly detection is similar to noise detection [32] - the main difference is that noise is irrelevant to an analyst but anomalies are of interest. The main challenges faced during anomaly detection are -

1. Identification of a normal region is difficult as one needs to encompass the entire normal regions. Anomalies that are very close to the normal region are often misclassified as normal data points.

2. The anomalies are very similar to noise, hence is it difficult to distinguish between anomalies and noise and generally requires a domain expert.

3. Anomalies are constantly evolving over time. In many applications, due to the dynamic and evolving nature of the data the current concept of normal behavior might not be adequately representative in the future. This is especially relevant in security application like intrusion detection, fraud detection etc. E.g., in intrusion detection, adversaries are constantly adapting themselves to the changing environment so that they can circumvent the techniques [33].

The challenges in anomaly detection give the adversary an opportunity to exploit their weaknesses to their advantage. As we explore in the following chapters, it is

these limitations which make it easy for an adversary to bypass the machine learning techniques. We now review clustering, nearest neighbor and classification based anomaly detection techniques in detail.

2.1.1 Clustering Based Anomaly Detection

Clustering based anomaly detection is primarily an unsupervised technique that groups similar patterns together. Clustering techniques can be further classified according the following assumptions that the techniques rely on [1]

1. Normal data belongs to a cluster whereas anomalous data does not belong to any cluster.

2. Normal data belongs to more dense clusters whereas anomalous data belongs to sparse clusters.

3. Normal data lies closer to the cluster centroid whereas anomalous data lies further away from the cluster centroid.

Domains where clustering based techniques are frequently applied include detecting fraudulent credit card transaction [34, 35] and intrusion detection systems. In credit card fraud, the data typically comprises of records which include information about the user, credit card transaction amount, time between consecutive credit card usage etc. The anomalous activities are generally point anomalies e.g., high payments, purchasing expensive items and items that have never been purchased before. These anomalies can be easily identified using clustering as these point anomalies are different compared to the rest of the training data.

Another instance where clustering is extensively used is intrusion detection [23, 36, 37] because a majority of the data is unlabelled. Even if we were able to obtain data by simulating interactions it is difficult to get a good representative of a data set which comprises of all the normal activities. Hence, clustering being an unsupervised technique gives an advantage over other approaches.

DBSCAN [20] is a density based clustering algorithm that has been adopted for various anomaly detection applications [21, 38, 39]. The general idea behind this technique is to expand a cluster as long as the density of the neighborhood is above the user defined threshold. The points that do not belong to any cluster are treated as anomalies and are further investigated.

Several variants of DBSCAN have been proposed. Chen et al. [40] have used the R^* tree in DBSCAN for determining the neighborhood of an object based on the radius. A hybrid structure using DBSCAN and KMeans [41] has been proposed to improve the accuracy of the intrusion detection systems. DBSCAN has also been successfully applied for clustering in evolving data streams [42].

2.1.2 Nearest Neighbor Based Anomaly Detection

Another approach for anomaly detection is nearest neighbor which is based on the assumption that normal data is similar to its neighborhood whereas anomalies are different as compared to its neighborhood. These techniques classify data based on a distance or similarity measure, which can be computed in different ways. One of the standard metric for continuous attributes is euclidean distance but other metrics have also been applied [43]. Computing the similarity for multi variate data instances is done by computing it for a single attribute and then aggregating it [43]. Categorical attributes are more complex, but can be measured using chi-square statistic or more complex measures [44].

These approaches are similar to clustering based techniques - they both compute a function of distance to determine the similarity between data points. The key difference is that clustering approaches take a global approach whereas these take a local approach. Clustering techniques determine if a data instance falls in a cluster or not; nearest neighbor determines how different a data point is in comparison to its local neighborhood.

Nearest neighbor based anomaly detection techniques broadly use two different approaches to compute the anomaly score of a data instance: [1]:

1. The first approach calculate the anomaly score as the distance between a data point and its β^{th} nearest neighbor; the higher the distance the more anomalous a data point is.

2. The second approach is to compare the density of a data point to the average density of its neighborhood.

In regions of varying densities, density based techniques perform poorly because they consider global parameters of the data set rather than local parameters. Consider a region where a sparse and a dense cluster exist. Let there be a data point which is relatively at a shorter distance to the dense cluster as compared to the distance between the data points in a sparse cluster but at a greater distance as compared to the distance between the data points in a dense cluster. This data point will be identified incorrectly as normal because even though it is a local outlier; it is still a normal data point.

To address the problem of varying densities in a data set, a variety of techniques have been proposed to compare the density of a data point to its local neighborhood. These techniques calculate an anomaly score based on the relative density of each data instance or the distance of its data instance to it's β^{th} neighbor. For a given data point, to calculate the distance to its β^{th} nearest neighbor we construct the smallest hyper-sphere whose mass is at least β other instances and that is centered at the given data point. The radius of this hyper-sphere is equivalent to the distance to its β^{th} nearest neighbor. Thus, the density of a data point is approximately the inverse of the distance to the β^{th} nearest neighbor for that data point. A data point that is in a low density neighborhood is an outlier whereas a data point that lies in a high density is considered to be a normal point.

Local Outlier Factor [24] is a nearest neighbor technique that assigns an anomaly score based on the ratio of the average densities of a data instance's β neighbors and

the density of the data instance itself. Researchers have proposed variants of the LOF techniques. Multi-Granularity Deviation Factor (MDEF) [45], a variant of LOF was proposed that calculates the anomaly score based on the standard deviation of the local densities.

Tang et al. [46] proposed a connectivity based LOF which separates the notion between density and isolation, thereby detecting outliers independent of the densities. A simpler version of LOF was proposed by Hautamaki et al. [47]. They compute the outlier factor using the in-degree Number for each data instance. Multi-Granularity Deviation Factor (MDEF) [45], a variant of LOF was proposed that calculates the anomaly score based on the standard deviation of the local densities. LOF has been applied to finding anomalies in intrusion detection systems [25].

2.1.3 Classification Based Anomaly Detection

Classification techniques for anomaly detection are broadly grouped into two categories - multi class and single class. In multi-class, there are different normal-classes to which the training data can belong [48] and an anomaly is any instance which does not belong to any of the normal class. These techniques learn to distinguish between different normal classes. A test instance is considered anomalous if it does not belong to any of the normal classes. Another variant is to assign a confidence score to the classification made by the model. If none of the classes has a confidence score above the threshold, then the data instance is treated as anomalous.

In one-class classification the normal data belongs to a single class and anything that does not belong to this class is considered anomalous. These techniques learn by constructing a boundary around the majority of the probability of the data mass by using a one class algorithm, e.g., one-class SVMs [49], one-class Kernel Fisher Discriminants [50]. Any test instance that does not fall within the learned boundary is classified as anomalous.

Support Vector Machines (SVMs) [30] have been applied to anomaly detection in the one-class setting. Developed by Vapnik [51], SVMs view the classification problem as a quadratic optimization problem to learn a region that contains the training data. Complex regions are learned using kernels, such as radial basis function (RBF) kernel. The majority of the normal data mass lies within this region and anomalies are anything that lie outside this region. Variations of the above technique have been used for anomaly detection in audio signals [52]; in host-based Intrusion Detection System (IDS) for identifying anomalous access in the Microsoft Windows Registry [29]; in hybrid models for high-dimensional and large-scale anomaly detection [53].

The SVM technique has applications for various anomaly detection systems like system intrusion detection [54], road traffic [29, 55], audio signal data [52], wireless sensor networks [56] and temporal sequences [27].

Another variant finds the smallest hypersphere in the kernel space that contains all the training instances [57, 58]. A test instance is classified as anomalous if it lies outside the hypersphere otherwise it is classified as normal. Robust Support Vector Machines (RSVM) that are robust to anomalies have been proposed in Song et al. [2002]. They achieve robustness in their algorithm by incorporating the distance between the data point and the centeroid of the class to which it belongs in the margin of the hyperplane. Their technique has been applied to system call intrusion detection [Hu et al. 2003].

A hybrid of SVM and Linear regression has also been proposed by authors in [59]. They use a sliding approach and the algorithm uses a Reproducing Kernel Hilbert Space (RKHS) with Radial Basis Function (RBF) kernel. However, this kernel requires the system to be in a steady state otherwise the accuracy of the system decreases.

One of the advantage of the classification based methods over clustering and nearest neighbor based methods is that they are faster because the test instance is classified using a pre-computed model.

2.2 Adversarial Machine Learning

Machine learning has provided solutions for several different security applications - filtering spam email, identifying malware, detecting attacks against servers by identifying malicious payload and intruders, identifying illegitimate credit card transactions. All of these application employ models which are built using massive amounts of data. These models are deployed in dynamic environments which necessitate s the need for constant training and testing of these models. The existence of adversaries in these environments, complicates the application of the machine learning technique as it leads to an arms race between the defender building a state-of-the-art model and an adversary trying to circumvent the technique.

Adversarial machine learning is the branch intersecting between machine learning and security that aims to study the effectiveness of a machine learning algorithm against an adversary and to learn the capabilities and limitations of the attacker. Huang et al. [15] gave a taxonomy of attacks against a machine learning system. They categorized the attacks based on influence or the capability of an attacker (causative and exploratory), security violation (integrity, availability and privacy) and specificity defined as attackers intention (targeted and indiscriminate). They have presented a boiling frog attack for anomalous traffic detection in which they inject the training data with chaff every week so that the detector gets acclimated to chaff which eventually results in compromising the integrity of the system by increasing the false negatives.

The attacks fall broadly in two categories -

- Evasion Attacks: In evasion attacks the adversary aims to bypass the machine learning technique at test time by crafting adversarial samples.
- Poisoning Attacks: In poisoning attacks the adversary aims at fooling the machine learning technique by manipulating the training data so as to cause a concept drift.

We now review poisoning and evasions attacks and various defense techniques.

2.2.1 Poisoning Attacks

The aim of the poisoning attacks is to increase the misclassification rate by injecting poisonous samples at training time. They are categorized into [19]:

- Indiscriminate attacks: In these attacks, the aim of the attacker is to cause a denial of service by increasing the misclassification rate so as to render the machine learning technique ineffective.
- Targeted attacks: These attacks aim at causing specific misclassifications.

Recently [60, 61], poisoning attacks have been studied against neural networks. A popular example of the poisoning attack has been Microsoft Tay, which was an artificial intelligence chat bot designed to talk to people on Twitter. However, it was shut down after 16 hours because it started posting inflammatory and offensive tweets learned from users who were tweeting offensive, politically incorrect and inflammatory tweets about issues such as "redpilling" [62].

Another example of real world poisoning attack is Kaspersky Lab [63], a leading antivirus company. They were accused of manipulating the antivirus software of competing companies by injecting fake samples; though they said that they were false accusations and denied any wrong doing.

Another application of poisoning attacks is the data sets which are publicly available for researchers and industry practitioners to develop state of the art technique. E.g., if the self driving car datasets are poisoned then the damage that they can cause in the worst case is the loss of human life.

We give an overview of poisoning attacks w.r.t anomaly detection methods in general, and then clustering and SVM specifically.

Poisoning Attacks Against Anomaly Detection

Kloft et al. [64] have analyzed the online centroid based anomaly detection technique in the presence of an adversary. They have formalized the learning and the

attack process, derived an optimal attack policy and theoretical bounds on the efficacy of a poisoning attack in perfect knowledge and limited knowledge scenario. Nelson et al. [65] have developed a model to analyze the efficacy of poisoning attacks and demonstrate the feasibility of the attacks.

Fogla et al. [66], in order to avoid anomaly detection, have created polymorphic instances of network packets. This ensures that the attack packet(s) are derived from the same statistical distribution as normal traffic packets, thereby subverting the anomaly detection approach. Rubinstein et al. [67] have applied the boiling frog strategy to poison a PCA subspace anomaly detector and have shown that only a moderate amount of poisoned data can substantially decrease the efficacy of the detector. They have also proposed a robust anomaly detector based on robust statistics to combat this attack.

Newsome et al. [68] have developed a red herring attack that involves poisoning the dataset with spurious features and can mislead the signature generation for malware detection. They [69] have also implemented attacks against Polygraph and have shown that an adversary even with correctly labelled samples can degrade the performance of the learner.

Poisoning Attacks Against Clustering

Biggio et al. [70] have used single linkage hierarchical clustering to demonstrate obfuscation and poisoning attacks thereby, evaluating the security of clustering algorithms in adversarial settings.

Poisoning Attacks Against SVM

A number of attacks have been developed against support vector machines. Biggio et al. [71,72] have proposed an attack strategy by contaminating the training data set through label flipping. They have proposed two attack strategies - random label flips and adversarial label flips. Two attacks models - free-range attack model that allows

for unrestricted data corruption by the adversary and a restrained attack model that permits practical attacks by associating a cost for the adversary's effort has been developed by Zhou et al. in [73]. They have also developed optimal SVM learning strategies against the two attack models. While this shares some similarity with our work, our goal is different: estimate vulnerability to such attacks (including unknown attacks). As a result, we concentrate on the minimal number of poison samples for a successful attack, rather than developing attacks that are designed to evade detection. Evasion attacks have also been developed using gradient based approach in [11]. Biggio et al. have shown in [10] that an intelligent adversary has the ability to predict the change of the SVM's decision function and use it to construct malicious data.

2.2.2 Evasion Attacks

Evasion attacks consists of perturbing data points to generate adversarial samples which can be misclassified at test time. Examples include manipulating malware code so that it bypass the detector, modifying an email to circumvent a spam filter etc. They are classified into two categories [19].

- Indiscriminate Evasion Attacks: The goal of the attacker is to misclassify the adversarial sample irrespective of which the class.
- Targeted Evasion Attacks: In this attack, the goal is to get an adversarial sample misclassified to a given class.

The first evasion attacks was demonstrated on spam filtering [74–76]. The underlying idea was to modify the bad words with the good words i.e., append a spam email with a legitimate email. Further heuristic measures (like words that appear in legitimate email but not in spam have a greater efficacy) were also developed based on the knowledge that spam email is easier to collect than legitimate email. Barreno et al. have illustrated different classes of attacks on SpamBayes [77]. Dalvi et al. [78]

have classified the classification as a game between the adversary and the learner. The classifier produced by the game is optimal if the adversary's strategy is optimal.

The next effort was to study the robustness of non linear classifiers. Srndic et al. [79], demonstrated the effectiveness of an evasion attack against SVM based on malicious and benign pdf files. They also showed that the gaussian kernel is immune to these attacks. However, in the recent years this has been proven to not be true [11,80]. These authors have shown that non-linear kernels are as susceptible to attacks as linear kernels. The underlying methodology was derived from finding a way to invert complex non-linear classifiers, to identify the most relevant features and perturb them to cause the maximum damage. The approach used was to calculate the gradient of the loss function as it specifies the direction of the maximum change in the function w.r.t the output.

The next attempt was to fool deep learning models. Szegedy et al. [81] showed that it is possible to generate minimally - perturbed images called adversarial samples which can be misclassified by neural networks at test time. Papernot et al. [82] have shown that adversarial examples can be transferred from one model to another as long as both were trained to perform the same task. Carlini et al. [83] have demonstrated an attack that can be carried on audio samples in which the machine hears intelligent voices but the human hears unintelligible voice. Evasion attacks have also been applied to face recognition systems [84] in which images are accepted as inputs rather than the actual face of a human being.

Having given an overview of the attacks, we now give an overview of the defenses:

2.2.3 Defenses against Adversarial Samples

1. Countering Evasion Attacks: Two approaches have been used to develop defensive measures against evasion attacks [19]:

(a) Designing Secure Systems: This approach aims at designing secure systems from the ground up, hence most of the approaches include adversarial samples in their training data. These defenses are mostly focussed on white box attacks where the adversary has the knowledge fo the entire system. The first adversary-aware classifier was proposed in [78], where they viewed the classification problem as a game and then iteratively trained the classifier. This produced a classifier, which is optimal given that the adversary's strategy is optimal. Various other game theoretic approaches have been applied to develop robust classifiers. Globerson et al. [85], have developed a min-max game for adversarial feature selection in which the adversary has the capability to delete features with an upper bound on the number of features that can be deleted. Robust optimization is also modeled as a min-max game in which the inner optimization problem solves at maximizing the classifiers inaccuracy whereas outer aims at maximizing the classifiers accuracy [72]. Bruckner and Scheffer [86] have shown the importance of the existence of a unique Nash equilibrium for a single shot adversarial prediction games and investigate whether it exists. Zhou and Kantarcioglu [87] have extended the machine learning game to include different type of adversaries (some may modify only malicious data instance whereas other may modify benign data instances) and different forms of corruption (mild or aggressive). They have introduced a nested Stackelberg game to simultaneously deal with malicious data corruption and unknown adversary strategy. Robust machine learning games has also been modeled as Stackelberg games [88, 89] where the defender can control the cost that the adversary has to pay to evade detection by being the leader of the game. The effect of robust optimization is to smooth the decision boundary so as to make it less sensitive to adversarial perturbation.

(b) Detecting Samples: Another approach for building secure systems is to detect samples based on their distance from the training data in the fea-

ture space. If they are far away from the training data then the samples are rejected [90, 91]. These techniques are employed when the adversarial samples appear in a region which is scarcely populated by the training data and hence are referred to as blind-spots.

2. Countering Poisoning Attacks: While most of the recent work has been concentrated on detecting evasion attacks, a few techniques have been developed for poisoning attacks. Following are the techniques [19]:

   (a) Adversarial samples as outliers. This is based on the assumption that adversarial samples are derived from different data distributions as opposed to the normal training data point to cause an impact on the decision boundary and therefore can be treated as outliers. This is equivalent to solving the problem of outlier detection and can be detected by using standard anomaly detection approaches [92, 93].

   (b) Security by Obscurity: These techniques follow the paradigm of protecting the model by hiding the information from the attacker. They target black box techniques in which the attacker queries the model to improve their evasion attack or a substitute model built by them. Examples include (i) randomly collecting training data points for building the model, (ii) randomizing the output to give incorrect feedback to the attacker (iiii) using classifier ensembles which are difficult to reverse engineer and (iv) hiding the model or the training data.

3 ATTACK MODEL

The goal of this thesis is to provide a quantitative framework to do a "what-if" security analysis of the different anomaly detection technique. To do so, we first provide a framework to develop an attack strategy and then we determine how vulnerable the anomaly detection technique is when subjected to such an attack. To develop an attack strategy and to analyze the security of an anomaly detection technique we have specified an adversary's knowledge, capabilities and goals in accordance with the taxonomy specified in [15].

1. Adversary's Goal: We define the attacker's goal in terms of security violation and specificity as defined in [15].

   (a) Security violations include the following:

   - Integrity violations: Attack points are classified as normal.
   - Availability: Render the system unusable by increasing false positives and false negative.
   - Privacy: Adversary violates the privacy of the user by using the information observed from the learner.

   (b) Specificity defines the intention of the attacker and includes:

   - Indiscriminate: The attacker's goal is to misclassify a general class of points so as to increase the false positives and false negatives.
   - Targeted: The goal of the attacker is focussed on a single point

   We assume that the adversary want to perform a targeted attack which results in an integrity violation. Specifically, the adversary has a particular (predetermined, but unknown to the defender) point that they wish to have misclassified as normal.

2. Adversary's Knowledge: This is the extent to which an adversary knows about the anomaly detection model and encompasses the following -

   (a) Anomaly Detection Algorithm: The machine learning technique that an adversary wants to target. In order to build defensive systems, system designers should assume that the algorithm is known by the adversary.

   (b) Data: It is the data used to train and evaluate the model.

   (c) Feature Space: This includes the features used in the training of the model and includes relevant features learned by the model or specialized features used by the algorithm.

   We assume that while designing a secure system the "the enemy knows the system" i.e., an adversary will have knowledge about the system. This is in accordance with the current practices in the security community which embraces the fact that a system should be secure even if everything about the system except the key, is known to the adversary. We therefore consider two scenarios:

   (a) Limited Knowledge: In limited knowledge, we assume that an adversary knows only the distributions of the training data and the parameters of the learned model.

   (b) Perfect Knowledge: In perfect knowledge, the adversary has complete knowledge about the system, including the training data and the parameters of the learned model. For example, an attacker may have the ability to eavesdrop on the network traffic that is being used to train the model.

   We develop an attack strategy and compare the adversary's effort for both scenarios.

3. Adversary's Capabilities: Adversary's capability defines the extent to which the adversary can control the learning of the model by manipulating the training data set. We define it in terms of the influence (causative or exploratory) of the attack as defined in [15].

(a) Causative: The attacker can alter the learning process by manipulating the training data. The influence of the attacker over the learning process can range from having control over a fraction of the data set to influencing the data production (injecting fake data points).

(b) Exploratory: The attacker can probe the learned model to gain information about it but cannot alter the training process.

We assume that the adversary can poison the dataset by creating fake data points. This is applicable in various practical scenarios, e.g., in the case of spam filtering [15], where the adversary may easily send malicious samples. Malware code can also be manipulated in a constrained or an unconstrained manner to poison the malware detector [94]. We assume that the capability of an adversary is causative, i.e., generate fake data points to disguise the anomaly point.

4. Attack Strategy: Once the adversary's goals, knowledge and capabilities are defined, we can develop an attack strategy by moving the decision boundary so that the anomalous point lies within the boundary. Formally, the strategy can be defined as minimizing the number of attack points that have to be added so that the concept drift caused will result in the instance being classified as a normal point. The attack strategy is illustrated in Figure 3.1.

A variant of the attack is when an adversary can adapt the data point once the adversary knows the data. In this scenario, the data distribution remains the same but given the distance between the anomaly point and the decision boundary; the attacker can choose where to place the attack point based on the training data. Our goal is to minimize the effort of the adversary to get the anomalous point misclassified. Therefore, this attack variant allows the adversary to adapt the data point by modifying the features of the anomalous point so that it is closer to the cluster while still retaining the capability to cause the same damage as before. This reduces the effort of the adversary as it will be in a slightly more dense region.

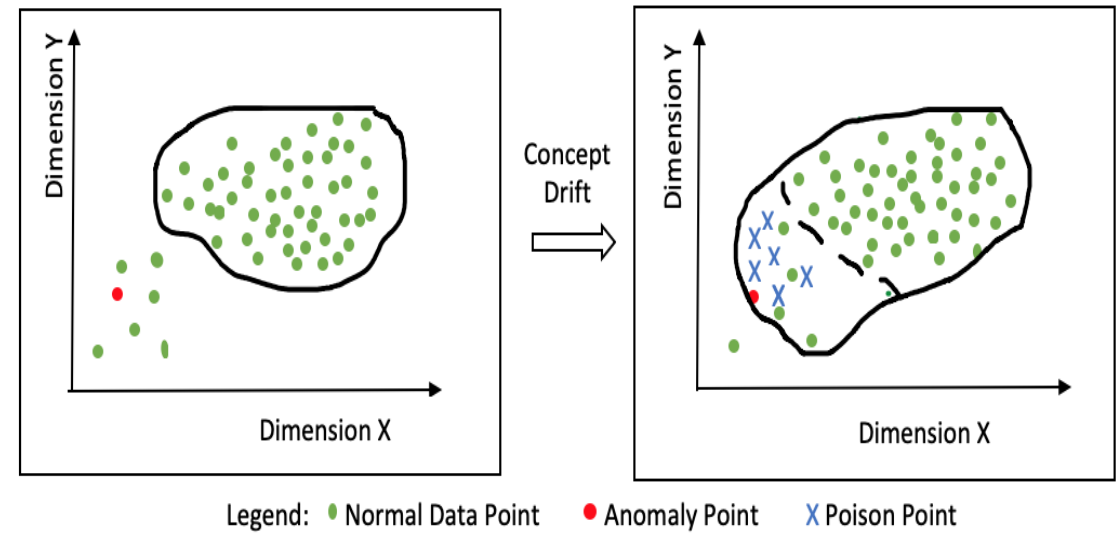

Figure 3.1. Poisoning Attack resulting in Concept Drift

Analyzing the vulnerability of this attack fits into our attack model as we are changing the location of the attack point; however the relative location of the attack point w.r.t to the distance to the closest point on the decision boundary remains the same. Therefore, this attack variant can be easily captured by the perfect knowledge scenario of the attack point.

4 POISONING ATTACKS AGAINST CLUSTERING BASED ANOMALY DETECTION (DBSCAN) AND DEFENSE

4.1 Introduction

Clustering, as an anomaly detection technique assumes that normal data belongs to a large or dense cluster and anomalies do not belong to any cluster or belong to a sparse cluster. DBSCAN [20] is a popular density based clustering algorithm for anomaly detection and has been effectively applied in a variety of domains [21–23]. The method assumes that the normal data lies in high density regions whereas anomalies or outliers lie in low density regions. One of the advantage of density based method is that they are good at discovering clusters of arbitrary shapes.

In this chapter, we evaluate how vulnerable DBSCAN-based anomaly detection is to an attack where the adversary is unable to change their own data, but *can* create fake entities to attempt to make their actual data look less like an anomaly. The goal is to help the defender estimate the risk posed by such attacks by estimating the effort required by the attacker to disguise the anomalous point under the constraint that it cannot change the anomalous point.

4.1.1 Background

For completeness, we give a few definitions of the DBSCAN algorithm. DBSCAN groups together points that are closely packed together marking low density region points as outliers. For the purpose of DBSCAN the points are classified as core object, (density-)reachable objects, and borders, as follows:

Definition 4.1.1 *Core Object - A core object o w.r.t ϵ and α is a data point such that $|N_\epsilon(o)| \geq \alpha$, where $N_\epsilon(o)$ is the ϵ neighborhood of the object o, ϵ is a user-defined*

parameter specifying the radius of a neighborhood, α is the minimum no. of points needed to form a dense region.

Definition 4.1.2 *Directly Density Reachable: An object p is directly density reachable from o if o is a core object and $p \in N_\epsilon(o)$*

Definition 4.1.3 *Density Reachable: An object p is density reachable from o if there exists a path $p_1, p_2, ..., p_n$, where $p_1 = p$ and $p_n = o$ such that p_{i+1} is directly density reachable from p_i and $p_2...p_n$ are core objects.*

Definition 4.1.4 *Border Object: An object p is a border object if it is not a core object but it is density reachable from another core object.*

Definition 4.1.5 *Density-Connected: An object p is density connected to an object q if there exists an object o such that both objects are density reachable from o.*

Definition 4.1.6 *Density Based Custer: A cluster C is a non empty subset of $\mathcal{D}$ such that it satisfies the following properties -*

1. *$\forall$ (p,q) if $q \in C$ and p is density reachable from C then $p \in C$.*
2. *$\forall (p, q) \in C$, p and q are density connected.*

4.1.2 Framework

To analyze the vulnerability or the security of a system we need to identify the adversary's goals and its capabilities. To this end, we rely on the taxonomy specified in [95] and adapt the framework described in Chapter 3.

Adversary's goal

The attacker wants to perform an integrity violation by including the anomaly (attack point) in the target cluster $\mathcal{C}$.

Adversary's knowledge

We assume that the adversary has knowledge of the training algorithm and in many cases partial or complete information about the training set, such as its distribution.

Adversary's Capability

In general we assume that the attacker can generate arbitrary data points for poisoning the training set.

Attack Strategy

Once the adversary's goals and capabilities are defined, we can develop an optimal strategy that specifies how to manipulate the dataset so as to meet the adversary's goal of obfuscating the anomaly. This can be achieved by making the neighborhood of the anomaly point denser so that it "looks-like" a normal data point. This is justified under the assumption that a normal datapoint belongs to a larger and a denser cluster. Formally, the strategy can be defined as minimizing the number of attack points that have to be added so as to include the anomaly in a given existing cluster. In formal terms, let $\mathcal{A}$ be the set of attack points, a be the anomaly point and $\mathcal{C}$ be the target cluster. Then, the adversary strategy can be formulated as

$$min|A|, \text{ s.t. } a \in \mathcal{C}$$

Figure 4.1 depicts the expansion of the cluster as a consequence of an attack on the anomaly detection technique.

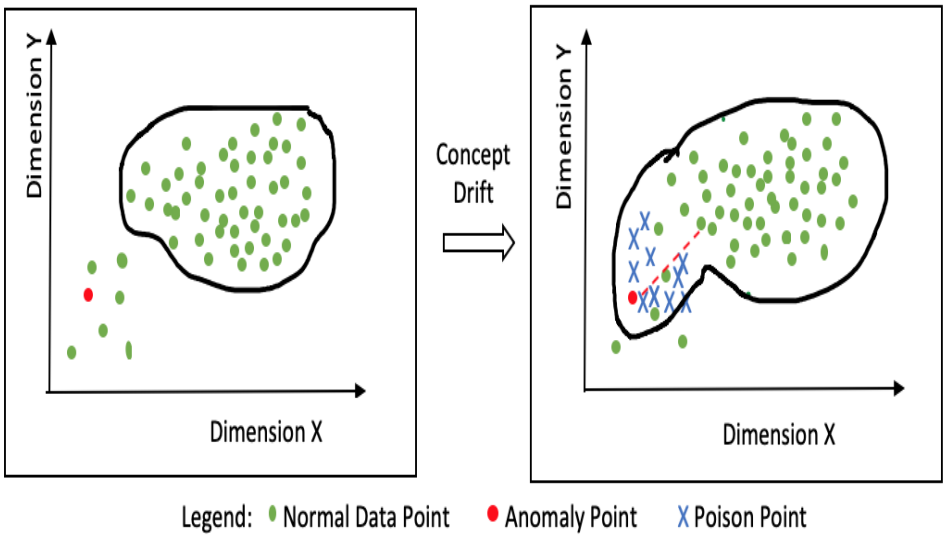

Figure 4.1. Expansion of the cluster as attack points are added

4.2 Attack Strategy & Assessing Risk

4.2.1 Adversary's Attack Strategy in Limited Knowledge Scenario

We now present the near-optimal attack strategy (or optimal if the adversary knows only the data distributions). To do so, we exploit the property inherent in DBSCAN that if a point is found to be in a dense part of a cluster, its ϵ-neighborhood is also part of that cluster. Let us assume that $\mathcal{C}$ containing the set of points is the target cluster that the adversary wants to be included in and let a be the anomaly point. The attack strategy is shown in 4.1. A reader can argue that it is easier to add α points in the ϵ neighborhood of the anomalous point a. However, that will be a trivial attack and easily detectable as it will increase the number of clusters by 1. We now present an algorithm for this attack.

Algorithm 1: Adversary's attack strategy in Limited Knowledge Scenario

1. Find $p \in \mathcal{C}$ s.t. $dist(a, p) = \min(dist(a, \mathcal{C}))$.
2. Let $d_{min} = \min(dist(a, \mathcal{C}))$.
3. Let i $= \lceil d_{min}/\epsilon \rceil$.
4. In a straight line between a and y place core points at $a_1^1 = \epsilon$, $a_1^2 = 2 * \epsilon$ ..... $a_1^i = i * \epsilon$.

We now give a proof of correctness for the above algorithm and then quantify the bound on the adversary's effort required to include the attack point in the cluster. Note, that the attack can be reduced from a multi-dimensional space to a single dimensional space because we are using the L_K metric to measure the distances.

Proof of Correctness: DBSCAN in the ExpandCluster function [20], for a given set of points in cluster $\mathcal{C}$, includes all the points which are at a distance ϵ from any core point $p \in \mathcal{C}$. At the first iteration, a_1^1 will be included in $\mathcal{C}$ as it is at ϵ distance from p. In the second iteration, a_1^2 will be included in $\mathcal{C}$ because it is at ϵ distance

from core point a_1^1 and so forth. At i+1 iterations, a will be included in $\mathcal{C}$ because of a_1^i.

Lemma 4.2.1 *To add i core points in a straight line, the minimum number of points that need to be added i.e., $E[|\mathcal{A}|] = \alpha * \ i/2$, where A is the set of attack points.*

Proof If $i = 1$, then the number of points needed to make a core object by definition 1 is α. If i is even, then every consecutive core point's e-neighborhood will intersect, hence $E[|\mathcal{A}|]$ = (i/2)*α. If i is odd, then the $E[|\mathcal{A}|]$ = (i/2 + 1)*α. Hence, $E[|\mathcal{A}|] = \alpha * \ i/2$ ■

Theorem 4.2.1 *For any $|\mathcal{D}|$ and n, let $d_{min} = \ min(dist(a, \mathcal{C}))$, where*

- *a is the anomaly point*
- *min(dist(a, $\mathcal{C}$)) is the minimum distance from a to the cluster $\mathcal{C}$.*

Then, $|\mathcal{A}| = (\alpha) * \ d_{min}/(2 * \epsilon)$

Proof $\forall \boldsymbol{n}$:dist is a function (L_k metric) that takes a and a point in $\mathcal{C}$ and returns a non-negative number, thereby reducing the n-dimensional attack to 1-dimensional.

From the above argument and Lemma 4.2.1, it immediately follows that $\forall\ n$, $|\mathcal{A}| = (\alpha) * \ d_{min}/(2 * \epsilon)$.

$\forall |\boldsymbol{\mathcal{D}}|$: We will use induction to prove that irrespective of the size of the set of data points the size of the set of the attack points for DBSCAN is $(\alpha) * \ d_{min}/(2 * \epsilon)$.

For $|\mathcal{D}| = 1$: Let $d_a = dist(a, p)$. As p is the only point in the cluster $\mathcal{C}$, from the above argument, we have to add only one core point o so as to include y in the 2^{nd} iteration. From Definition 1, a core object o is a data point such that $|N_\epsilon(o)| \geq \alpha$. In order to make o a core object, we have to add $\alpha - 1$ data points in its ϵ neighborhood. It follows that $|\mathcal{A}| = (\alpha) * \ d_{min}/(2 * \epsilon)$, which includes one core object and $\alpha - 1$ directly density reachable objects.

For $|\mathcal{D}| = t$: Let us assume it to be true for $\mathcal{D}$ = t and let $d^t_{min} = min(dist(a, \mathcal{C}))$.

For $|\mathcal{D}| = t + 1$: We assume that a point is added to the Cluster $\mathcal{C}$ as our target is to include a in $\mathcal{C}$ i.e., $a \in \mathcal{C}$. Let a_{t+1} be the point that has been added and let $d^{t+1}_{min} = dist(a, \mathcal{C} \bigcup a_{t+1})$.
Case 1: If $d^{t+1}_{min} \geq d^t_{min}$ then adding a_{t+1} point to $\mathcal{D}$ does not change the attack points or the attack set size as it is not the minimum distance from the cluster $\mathcal{C}$ to a.
Case 2: If $d^{t+1}_{min} < d^t_{min}$ then the attack set size changes to $|\mathcal{A}| = \alpha * \ d^{t+1}_{min}/(2 * \epsilon)$ from $|\mathcal{A}| = \alpha * \ d^t_{min}/(2 * \epsilon)$. This argument follows immediately from Lemma 1 and how we choose i in Algorithm 1.

Hence, $\forall |\mathcal{D}|$, $|\mathcal{A}| = (\alpha) * \ d_{min}/(2 * \epsilon)$. ■

4.2.2 Attacker's Attack Strategy in Perfect Knowledge Scenario

In the previous subsection, we presented the attackers optimal attack strategy in a limited knowledge scenario and bounded the effort required by the attacker as a function of the size of the attack dataset. In this section, we aim to analyze whether an adversary can do better when the adversary has knowledge of the training data set. The intuition behind reducing the number of attack points needed by an adversary, is that there is a low density region between the border of the cluster $\mathcal{C}$ (that an anomaly point wants to be included in) and the anomaly point. A small note - by low density region we refer to a region of radius ϵ that does not have enough data points to form a cluster. A combination of these data points and the attack points added by the adversary can transform a border object to a core object, thereby reducing the effort required by the adversary in terms of the size of the set of the attack points i.e., $|A|$. Let us define this low density region by $\mathcal{R}$ satisfying the following properties:

1. $\mathcal{R} \subset \mathcal{D}$
2. $\mathcal{R} \notin \mathcal{C}$

3. $\forall r \in \mathcal{R}, (dist(r,a) \leq dist(y,a)) \wedge (dist(r,y) \leq dist(y,a))$, where y is the border point of $\mathcal{C}$

To quantify the reduction in the effort of the adversary, we have to calculate the expected value of the size of this region. To do so, we propose a model and make a few assumptions about the distributions of the data. We now formally state the assumptions and proceed to calculate the size of $\mathcal{R}$.

Assumptions:

1. The Euclidean distances between the data points and the centroid of the cluster is drawn from a Gaussian distribution $Y \sim N(\mu, \sigma)$ with distances within the range $(\mu \pm 6\sigma^2)$.

2. We assume the distance of the anomaly point from the border of the cluster is $d_{min} = min(dist(a,y))$ as defined in Section 3.2.2. Let d_{min} be represented as $b * \epsilon$, where b is some constant.

3. Since ϵ is a parameter which represents distance we can rewrite ϵ as $c * \sigma$ where c is some constant.

4. $\forall r \in \mathcal{R}, dist(r,y) \geq 2\sigma$, where y is the centroid of $\mathcal{C}$. It is a reasonable assumption because of the two sigma effect of Gaussian distributions.

Based on these assumptions we calculate the expected size of $\mathcal{R}$ as follows:

$\mathbb{E}|\mathcal{R}| = \mathbb{E}_{-2\sigma-\epsilon}^{-2\sigma-2*\epsilon}(Y) + \mathbb{E}_{-2\sigma-2*\epsilon}^{-2\sigma-3*\epsilon}(Y) + ... + \mathbb{E}_{-2\sigma-(b-1)\epsilon}^{-2\sigma-b*\epsilon}(Y)$,

where $a * \epsilon = d_{min}$

$= \mathbb{E}_{-2\sigma-c\sigma}^{-2\sigma-2c\sigma}(Y) + \mathbb{E}_{-2\sigma-2c\sigma}^{-2\sigma-3c\sigma}(Y) + ... + \mathbb{E}_{-2\sigma-(b-1)c\sigma}^{-2\sigma-bc\sigma}(Y)$

$= \mathbb{E}_{c_1}^{c_2}(Y) + \mathbb{E}_{c_2}^{c_3}(Y) + ... + \mathbb{E}_{c_{(b-1)}}^{c_b}(Y)$,

where $c_1 = (-2-c) * \sigma$, $c_2 = (-2-2c) * \sigma, ...,$ $c_b = (-2-bc) * \sigma$

$= |\mathcal{D}| * Pr((c_1) \leq Y \leq (c_b))$

$$\mathbb{E}|\mathcal{R}| = |\mathcal{D}| * \frac{1}{2}[erf(\frac{c_b - \mu)}{\sigma\sqrt{(2)}}) - erf(\frac{c_1 - \mu)}{\sigma\sqrt{(2)}})], \tag{4.1}$$

where erf is the error function.

Theorem 4.2.2 *For any $|\mathcal{D}|$ and n, if an adversary has perfect knowledge about the training data, then* $|\mathcal{A}| = (\alpha) * \ d_{min}/(2 * \epsilon) \ - \mathbb{E}|\mathcal{R}|$

Proof This immediately follows from how we have defined $\mathcal{R}$ and Theorem 4.2.1. ■

Extension to Multiple Disjoint Clusters:

We have so far assumed that there is a single cluster for the adversary to be included in. We now extend our method to include disjoint clusters. Let there be k disjoint clusters and we assume that each cluster is drawn from a Gaussian Distribution $Y \sim N(\mu_1, \sigma_1), ..., Y \sim N(\mu_k, \sigma_k)$. The size of each cluster is $|\mathcal{C}_k|$, s.t. $\sum_{i=1}^{i=k} |\mathcal{C}_k| = |\mathcal{D}|$ and $\{\mathcal{C}_i\} \cap \{\mathcal{C}_j\} = \emptyset$. To calculate the $\mathbb{E}|\mathcal{R}|$, we find the cluster $C = \{\mathcal{C}_j\}$ which is closest to the anomalous point a. The value of the $\mathbb{E}|\mathcal{R}|$ is calculated similarly as when there is one cluster with two differences:

1. The value of the size of the cluster changes from $|\mathcal{D}| = |\mathcal{D}_j|$.

2. For any other cluster $\mathcal{C}_i$, where $i \neq j$, we calculate the number of points in the ϵ neighborhood of the anomalous point a that is contributed by $\mathcal{C}_i$. Since the clusters are disjoint and we are adding attack points between the cluster C_j (closest to a) and a, the addition of attack points is not in the neighborhood of any cluster $\mathcal{C}_i$, where $i \neq j$. Therefore, we only need to evaluate whether a cluster C_i has any normal points in the ϵ neighborhood of a. To do so, we assume that the anomalous point a is at a distance d_i from the mean μ_i of the $\mathcal{C}_i$. The expected value of ϵ neighborhood ($N_\epsilon^i(a)$) of the anomalous point a that is generated by $\mathcal{C}_i$ is given by:

$$\mathbb{E}|N_\epsilon^i(a)| = \mathbb{E}_{d_i-\epsilon}^{d_i+\epsilon}(Y_i)$$
$$= |\mathcal{D}_i| \ * \ Pr((d_i - \epsilon) \leq Y_i \leq (d_i + \epsilon))$$
$$\mathbb{E}|N_\epsilon^i(a)| = |\mathcal{D}_i| * \frac{1}{2}[erf(\frac{d_i - \epsilon - \mu_i)}{\sigma_i \quad \overline{(}2)}) - erf(\frac{d_i + \epsilon - \mu_i)}{\sigma_i \quad \overline{(}2)})], \tag{4.2}$$

Therefore, the expecated value of the size of the low density region $\mathcal{R}$, is given by

$$\mathbb{E}|\mathcal{R}| = |\mathcal{D}_j| * \frac{1}{2}[erf(\frac{\delta_{c_1} - \mu_j)}{\sigma_j \quad \overline{(2)}}) - erf(\frac{\delta_1 - \mu_j)}{\sigma_j \quad \overline{(2)}})] + \sum_{i \neq j} \mathbb{E}|N_\epsilon^i(a)| \tag{4.3}$$

4.2.3 Vulnerability to the attack

So far, we have developed an attack strategy and have bounded the efforts required by the adversary. We now assess the vulnerability to this attack under this attack strategy by analyzing how the false positives and false negatives change with the increase in the number of attack points. A false positive is any error when an anomaly detector (incorrectly) rejects a benign input; we measure the change in this rate with the increase in the number of attack points, whereas false negative is the term used to describe a network intrusion device's inability to detect true malicious events. To do so, we model the distances of the data points and the attack points from the centroid of $\mathcal{C}$ being derived from Gaussian distributions. The goal of this model is to predict the likelihood that an attacker will succeed given the ability to generate some number of fake points. We now state the assumptions formally for our model and then calculate the expected false positives and the false negatives.

Assumptions:

1. The Euclidean distance's between the data points and the centroid of the cluster is drawn from a Gaussian distribution $Y \sim N(\mu, \sigma)$ with distances within the range $(\mu \pm 6\sigma)$, where $\mu = 0$.

2. We assume that there is only one adversary i.e., there is only one anomaly point.

3. A false positive is any point p s.t. $p \in \mathcal{D}$ and $p \notin \mathcal{C}$.

4. To model false negatives, we assume that a false negative is any point a s.t. $a \in \mathcal{A} \cap \mathcal{C}$.

Based on our assumptions, we can calculate the expected false positives as follows:

$$\mathbb{E}(FP) = \sum_{p \in \mathcal{D}} p * Pr(p_{fp}), \text{where} \tag{4.4}$$

$$p(p_{fp}) = Pr(p \in \mathcal{D} | p \notin \mathcal{C}) \tag{4.5}$$

To simplify the calculation of $Pr(p_{fp})$, we assume that there exists a point $\lambda = c'\sigma$ s.t. all the data points sampled from the range $(\mu - \lambda, \mu + \lambda)$, are a part of cluster $\mathcal{C}$. Since we are assuming that the distance between the data points and the centroid of the cluster are drawn from a Gaussian distribution, we can safely say that the density decreases as we move away from the mean of the Gaussian distribution. I.e., the centroid of the cluster has a high density region whereas the border of the cluster is in a low density region. Let us assume that the border of the cluster is at the point λ. The expected density of this region $\mathcal{S}$ is $\mathbb{E}|\mathcal{S}_\lambda| = |\mathcal{D}| * Pr(-\lambda \leq Y \leq \lambda)$, as $\mu = 0$. A false positive will be any point which is not in this region. Therefore,

$$\mathbb{E}(FP) = |\mathcal{D}| - (|\mathcal{D}| * Pr(-\lambda \leq Y \leq \lambda))$$

$$\mathbb{E}(FP) = |\mathcal{D}| - (|\mathcal{D}| * (\frac{1}{2}[erf(\frac{\lambda - \mu)}{\sigma \quad \bar{(2)}}) - erf(\frac{-\lambda - \mu)}{\sigma \quad \bar{(2)}})]) \tag{4.6}$$

For completeness, Algorithm 2 describes the process to calculate the value of γ.

Algorithm 2: Calculating the value of λ

1. Let $\lambda = \epsilon$. Therefore, the range is $(-\epsilon, \epsilon)$
2. $\mathbb{E}(S_\lambda) = |\mathcal{D}| * Pr(-\lambda \leq Y \leq \lambda)$
3. ctr = 1
4. while $\mathbb{E}(S_\gamma) \geq \frac{\alpha}{\mathcal{D}}$
   - $\lambda = (ctr + 1) * \epsilon$
   - range r $= (-\lambda, -\lambda + 2 * \epsilon)$
   - $\mathbb{E}(S_\lambda) = |\mathcal{D}| * Pr(Y \in r)$
   - ctr++

Proof of Correctness : We are assuming that the distances are drawn from a Gaussian distribution with mean $\mu = 0$, therefore the density of any area is maximum

around μ and decreases as we move further away from the mean μ. This ensures that the expected value of the density of a cluster will be maximum around the mean of the Gaussian distribution with its density monotonically decreasing with the increase in the distance from the mean. Also, for every point DBSCAN scans a region of radius ϵ which justifies setting the range as $2^*\epsilon$. Based on these two facts we can say that the algorithm correctly computes the value of λ.

We now assess the vulnerability of the attack in terms of false negatives. According to our model, the expected false negatives will be $|\mathcal{A}| + 1$ as we are assuming only one adversary and if the adversary adds an attack point, then according to the attack strategy specified in Section 4.2.1 and Section 4.2.2 it will be a part of $\mathcal{C}$.

4.3 Experiments & Discussions

In the previous sections, we have bounded the efforts of the adversary and assessed the vulnerability for an obfuscation attack using poisoning and have modeled the impact on DBSCAN. We now apply our theoretical results to the real world domain of intrusion detection and validate our theoretical model. We use a greedy approach to determine the adversary's optimal strategy with perfect knowledge. While this is worst-case exponential, we were able to calculate this in our test datasets, allowing us to validate the defender's estimates.

Experimental Setup

We now describe the experimental setup and the parameters for each of the dataset.

1. KDD Cup '99 Dataset [96]: From this dataset, we have randomly sampled 700 normal data points from the training data set i.e., $|\mathcal{D}| = 700$. We also assume that there are 5 different types of adversaries corresponding to the 5 different attack types (back, neptune, teardrop, nmap and ipsweep). For each attack

type we average out our results over 10 different instances. For the dos attack we have set $\alpha = 30$ and $\epsilon = 0.4$. These parameters ensures that every instance of every adversary type is detected i.e., True Negative Rate = 0. For the user to root attacks we have set the parameters as $\alpha = 30$ and $\epsilon = 0.12$. We also assume a constant false positive rate of 0.02 and the evasion rate to be 0 when the initial training data set is trained.

2. Yahoo S5 [97] : We have randomly sampled 700 normal data points from the A4 benchmark. We assume that there is one type of adversary marked as change-points and have averaged out the results over 10 randomly sampled instances of the adversary. For this anomaly detection dataset, we have set $\alpha = 30$ and $\epsilon = 0.4$ ensuring the True Negative Rate is 0. The False Positive Rate is set to be 0.01.

3. Thyroid dataset [98]: For this dataset we have set $\epsilon = 0.4$, $\alpha = 30$ and the False Positive Rate as 0.02.

4. Credit Card fraud detection dataset [99] from the Kaggle repository. We have set $\epsilon = 0.8$, $\alpha = 30$ and the initial FPR is 0.016.

5. CICIDS2017 Intrusion Detection Dataset [100] - This dataset is published by the Canadian Cybersecurity Institute. From this dataset, we have chosen the DoS Hulk attack type and selected the features as proposed in [100]. We have chosen $\epsilon = 0.5$, $\alpha = 30$ and the False Positive Rate (FPR) to be 0.004.

4.3.1 Estimation of Adversary's Effort

In this experiment we aim to estimate the effort required by the adversary in terms of the attack size. Our results are presented in Figures 4.2 & 4.3 and demonstrate the number of attack points required to carry out the obfuscation attack. The results are also compared to the attack size predicted by the model developed in Section 4.2.2 in a Perfect Knowledge and Limited Knowledge Scenario.

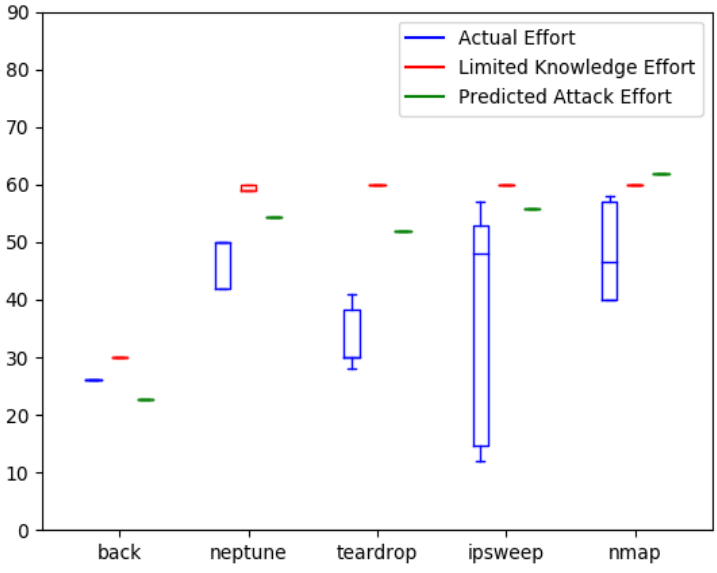

Figure 4.2. Adversary's Effort - KDD'99 Dataset

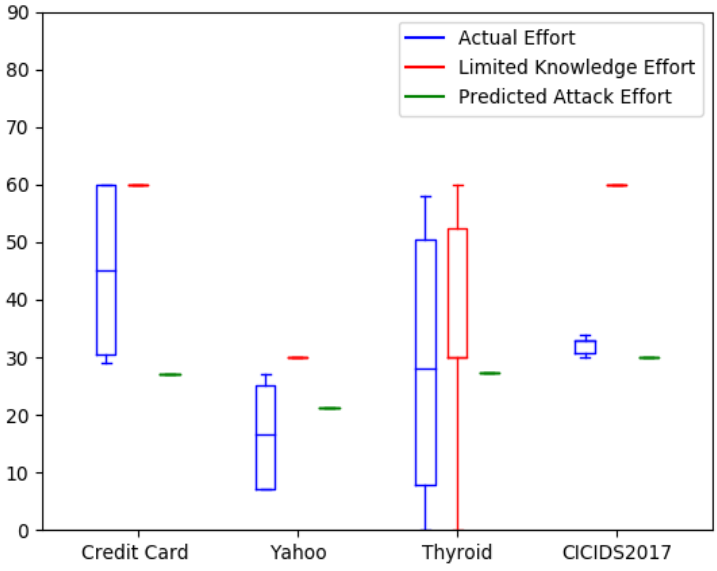

Figure 4.3. Adversary's Effort - Thyroid, Credit card, Yahoo & CIDSC'17 datasets

The results presented in Figures 4.2 & 4.3 validate that the effort required by the adversary is a worst case bound and he can always do better than the worst case. However, the efforts predicted for the adversary in a perfect knowledge scenario is comparable to the efforts required by the adversary in the real world situation. Thus, the model in the perfect knowledge scenario can be used as a reasonable indicator of the vulnerability of the method to the poisoning/obfuscation attack. Another interesting observation is that the adversary on an average needs to control only 6% of the training dataset to carry out a targeted attack.

4.3.2 Analyzing the Evasion Rate

We now assess the impact of this attack on DBSCAN by analyzing how the evasion rate of an adversary changes with the increase in the attack size. Figure 4.4, 4.5, 4.6 illustrates the evasion rate for the various attack types and compares it to the rates predicted by the model.

An interesting observation from the graphs is that on an average, an adversary only needs to control 5% of the training data set to increase the chances of evasion from 0 to 80%. We also observe here, that the minimum cluster size (α) effects the size of the training set that the adversary needs to control. The reason is intuitive, with a smaller α the effort required to create a core object will be less as compared to a bigger α. We have assumed the minimum cluster size to be 30 or 4% of the training data size. Of note is that there seems to be a fairly clear percentage of the data the adversary needs to be able to poison, at which the attack is likely to succeed; this is fairly well matched by the predicted rate.

4.3.3 Analyzing the Vulnerability to the Attack

We have analyzed the adversary's efforts and the probability of the success of the attack in the previous attack. We now assess the vulnerability of this attack by analyzing the change in the false positive rate of the anomaly detection approach

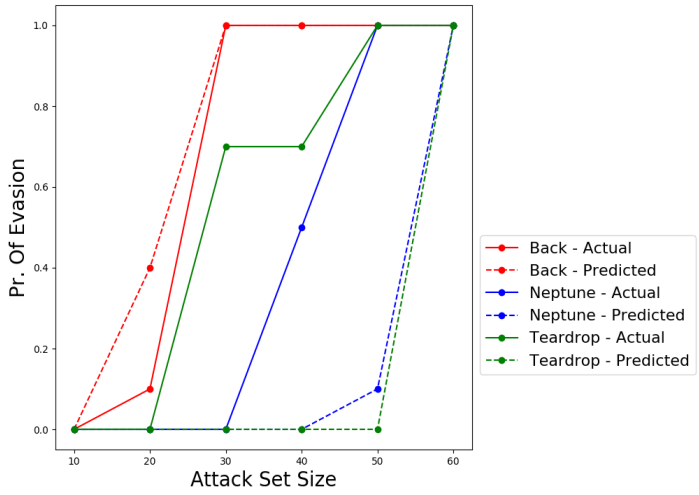

Figure 4.4. Probability of Evasion vs. Size of the Attack Set: KDD Cup'99 Dataset & dos attacks

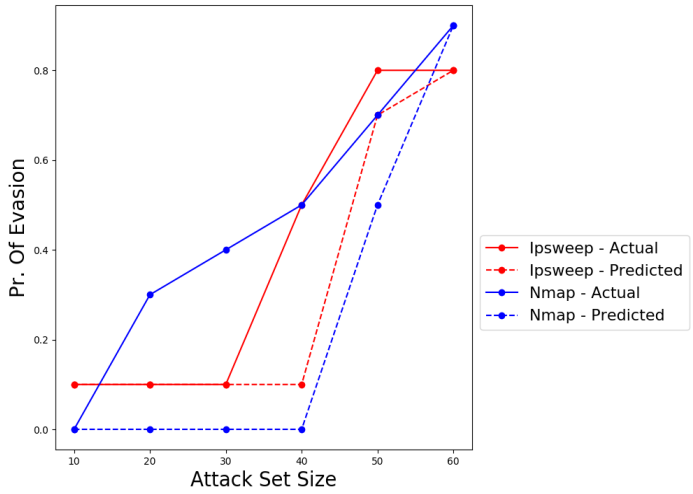

Figure 4.5. Probability of Evasion vs. Size of the Attack Set: KDD Cup '99 Dataset & u2r attacks

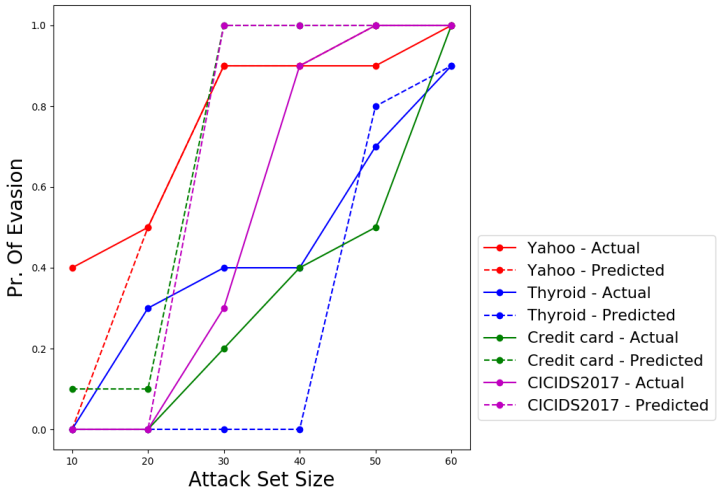

Figure 4.6. Probability of Evasion vs. Size of the Attack Set: Yahoo S5, Thyroid, IDS & Credit Card Anomaly Detection Dataset

Table 4.1.
Adversary's vulnerability to Attack for the Anomaly Detection Datasets

Attack Type	Original FPR	FPR Post Attack	Expected FPR
back (a1)	0.024	0.014	0.024
neptune (a2)	0.024	0.016	0.023
teardrop (a3)	0.024	0.013	0.023
nmap(a4)	0.017	0.003	0.015
ipsweep(a5)	0.017	0.004	0.015
Yahoo S5	0.01	0.007	0.00
Thyroid	0.04	0.04	0.035
Creditcard	0.016	0.013	0.001
CICIDS2017	0.004	0.004	0.001

and comparing it with the prediction of the model developed in Section 4.2.3. The changes in the FPR are presented in Table 4.1.

As it can be seen from Table 4.1, the actual and the expected false positive rates decrease with the increase in the attack points. This is counterintuitive because an attacker would want to increase the false positives so as to render the system unusable. However, our goal here is to perform a targeted attack which results in the expansion of the size of the cluster, ultimately leading to the false positives being included in the cluster resulting in a decrease in the false positive rate. Unfortunately, there is not a huge difference in the actual and the original false positive rates thereby letting the attacker carry out the attack stealthily without an obvious and easily detected increase in the false positive rate.

4.4 Detection of the Attack

In the previous section we have analyzed the efforts of the adversary required to subvert the DBSCAN anomaly detection approach and have predicted the vulnerability to the attack. Here, we present a method to detect the attack. Note: a simple approach to detect the attack would be to decrease the value of ϵ to make the bounds more tighter and as a consequence increasing the adversary's effort. However, this approach will lead to an increase in the number of false positives rendering the system useless.

To detect the attack points we propose to use random perturbation of data points. The intuition behind this is that since normal data points are in a high density region; perturbation by a small value $\eta \ll \epsilon$ is not going to make them an outlier. To understand why we use this heuristic, let us consider a normal data point p. It is a normal data point because it's distance from a core point is $\leq \epsilon$; and with high probability it is in a region of more uniform density as compared to the anomalous point a. This is because according to the attack strategy developed in 4.2.1, with high probability, there is a high narrow dense region between the anomalous point and the cluster C. This follows, from the fact that the attack points are added in a straight line between the anomalous point a and the point $p_i \in \mathcal{C}$ s.t. $dist(a, p_i) = min(dist(a, \mathcal{C}))$. Therefore, the attack points which are added are not randomly spread out but are concentrated in the narrow region as shown in Figure 4.1. Hence, if we perturb the anomalous point by a small value (η), the probability of it being misclassified will be higher than that for a normal point. A reader, could argue that the attacker could uniformly add points in the hypersphere centered at the anomalous point a, but then the attacker would require more effort and it will be easier to detect because a new cluster will be formed.

The results of applying our strategy and the effect on the change in the false positives and the amount of perturbation required is presented in 4.2. They indicate that we only need to perturb the data on an average 11% to classify the anomalous

Table 4.2.
Change in FPR when detecting the attack

Attack Type	FPR after Attack	FPR Post Detection	Perturbation Added to the Dataset
back (a1)	0.024	0.032	0.1
neptune (a2)	0.024	0.032	0.05
teardrop (a3)	0.024	0.068	0.03
nmap(a4)	0.0017	0.027	0.1
ipsweep(a5)	0.017	0.03	0.1
Yahoo S5	0.007	0.017	0.13
Thyroid	0.04	0.04	0.1
Credit card	0.013	0.021	0.1
CICIDS2017	0.004	0.05	0.3

points as outliers, however this methods leads to an increase in the number of false positives.

4.5 Conclusions

In this chapter, we have addressed the problem of evaluating the security of DBSCAN in adversarial conditions by providing an attack strategy and then a framework to bound the adversary's efforts and to assess the vulnerability to an attack. We have provided an optimal strategy in both perfect knowledge and limited knowledge scenarios and have proposed a model to estimate the effort of the adversary and vulnerability to the attack. We have demonstrated with real world datasets that DBSCAN is vulnerable to an obfuscation attack with a minimal amount of effort from

the adversary, and that we can effectively model the vulnerability to such attacks. Lastly, we have presented a detection strategy for identifying the poisonous points.

One of the main causes for the vulnerability of DBSCAN is because it solely relies on the distances between two points to determine if they will be included in a cluster or not, thus allowing for an efficient construction of an optimal policy for an obfuscation attack. We can also reasonably assume that density based clustering are more robust to availability attacks than to integrity attacks. Availability attacks render the system unusable by causing a high number of misclassifications (i.e., high false positives and high false negatives). Integrity attacks are the attacks that result in anomaly points being classified as normal points. This is evident as both the experiments and our model predict a decrease in the false positive rate with an increase in the true negative rate.

5 POISONING ATTACKS AGAINST NEAREST NEIGHBOR BASED TECHNIQUES (LOF) & DETECTION

5.1 Introduction

Nearest-neighbor based techniques for outlier detection broadly use two methods to compute a data instance's outlier score - the distance to its β^{th} nearest neighbor, where β is the user defined parameter and indicates the number of points to be included in the neighborhood of an object, or the relative density of each data instance. Local Outlier Factor (LOF) is a technique that uses relative density to compute its neighborhood. The main difference between LOF and DBSCAN is that LOF captures a local view of the data and hence can detect anomalies that are undetected by DBSCAN. The local density of a data instance is inversely proportional to the average density of its β neighbors. The outlier score in LOF is the ratio of its density to the average density of its β neighbors. For inliers, the LOF score is approximately 1 as its local density is approximately the same as of its neighbors. For outliers, the LOF score is greater than 1.

In this chapter we will address the problem of evaluating the security of nearest neighbor based anomaly detection in adversarial conditions and extend the model developed previously to predict the efforts of the adversary when the technique is Local Outlier Factor (LOF). LOF compares the local density of an object to its neighbors to identify outliers. The attack strategy will be similar to DBSCAN for poisoning the training dataset. Our goal will be to analyze the vulnerability of LOF to this attack.

5.1.1 Background

LOF calculates the anomaly score by comparing the ratio of the average of the densities of its β neighbors to its own distance, where β is a user defined parameter. To calculate the density of a data point, it finds the hypersphere that contains β closest neighbors of the data point and then divides the mass of that hypersphere (β) by the volume of that hypersphere. For the sake of completeness, we now define LOF.

Definition 5.1.1 *β-distance of an object o is the distance $dist(o, \beta)$ between the object o and it's β Nearest Neighbor.*

Definition 5.1.2 *β-distance neighborhood of an object o, $\mathcal{N}_\beta(o)$: contains every object $p \in \mathcal{D}$ (training set), whose distance from o is not greater than $dist(o, \beta)$.*

Definition 5.1.3 *Reachability distance of an object p w.r.t. o: $reachdist_\beta(o, p) = max(\beta - distance(p), dist(o, p))$.*

Definition 5.1.4 *Local Reachability Distance of o w.r.t. β:*

$$\ell rd_\beta(o) = \frac{1}{(\sum_{p \in N_\beta(o)} reachdist_\beta(o, p))/|(N_\beta(o)|)} \tag{5.1}$$

Definition 5.1.5 *Local Outlier Factor of o w.r.t β =*

$$LOF_\beta(o) = (\sum_{p \in N_\beta(o)} \frac{\ell rd_\beta(p)}{\ell rd_\beta(o)})/|N_\beta(o)| \tag{5.2}$$

Let $direct_{min}(o)$ denote the minimum reachability distance between o and p s.t. $p \in N_\beta(o)$. Similarly, let $direct_{max}(o)$ denote the corresponding maximum. Let $indirect_{min}(o)$ denote the minimum reachability distance between o and q s.t. $p \in N_\beta(o)$ and $q \in N_\beta(p)$. Let $indirect_{max}(o)$ denote the corresponding maximum. The following theorem gives a bound on the LOF value.

Theorem 5.1.1 *[24]. Let o be an object from the training set $\mathcal{D}$, and $1 \leq \beta \leq |\mathcal{D}|$. Then, it is the case that*

$$\frac{direct_{min}(o)}{indirect_{max}(o)} \leq LOF_{\beta}(o) \leq \frac{direct_{max}(o)}{indirect_{min}(o)}$$

Proof For proof, please see [24]. ■

We now give a lemma, that proves that the value of objects deep inside a dense region is approximately 1.

Lemma 5.1.1 *[24] Let the collection of objects in the dense region be $\mathcal{C}$. Intuitively, $\mathcal{C}$ is a cluster. Let reach-dist-min denote the minimum reachability distance of objects in $\mathcal{C}$ i.e., reach-dist-min=min $\{$reach-dist(o,p) $|o, p \in \mathcal{C}\}$. Let reach-dist-min be the corresponding maximum. Let θ be defined as (reach-dist-max/reach-dist-min - 1). Then for all objects $o \in \mathcal{C}$, such that all the β-nearest neighbors of o and p are in $\mathcal{C}$, it holds that 1/(1 - θ) $\leq LOF_{\beta}(o) \leq (1 + \theta)$.*

Proof See [24]. ■

5.1.2 Framework

We adapt the framework described in Chapter 3 to build the attack strategy:

- Adversary's Goal: The adversary wants to perform a targeted attack (a predetermined point unknown to the defender) i.e., misclassify the anomalous data instance.
- Adversary's Knowledge: We consider two scenarios (i) complete knowledge - adversary has access to the training data & parameters of the model, (ii) limited knowledge - adversary knows only the data distribution.
- Adversary's Capabilities: The capability of an adversary is causative, i.e., he can generate fake data points to disguise the anomaly point.
- Attack Strategy: Once the adversarýs goals, knowledge and capabilities are defined, we can develop an attack strategy. This is achieved by making the

neighborhood of the anomaly point denser so that it "looks-like" its neighbors thereby, ensuring that the average densities of a data instance's β neighbors and the density of the data instance are approximately equal. It is justified under the assumption that normal data lies in a dense area whereas anomalous data lies in a less dense area. Formally, the strategy can be defined as minimizing the number of attack points that have to be added so that the anomaly score calculated for the data instance will result in the instance being classified as a normal point. Let f be a function which takes the anomaly score (LOF for our purposes) as its inputs and returns a class label $\{-1, +1\}$, where -1 indicates an outlier and $+1$ indicates an inlier, $\mathcal{A}$ the set of attack points and a the anomaly point. The attack strategy can be formulated as -

$$min|A|, s.t. f(LOF(a)) = +1 \tag{5.3}$$

5.2 Attack Strategy & Assessing Risk

5.2.1 Adversary's Attack Strategy in Perfect Knowledge Scenario

We now present the attack strategy when the adversary has knowledge about the algorithm parameters and training data. We do so by adding attack points to increase the density of the anomaly points while ensuring that the density of the normal points does not increase. As a consequence, the average density of the neighbors becomes equal to the density of the anomaly point a. Note, a reader can argue that the simplest attack strategy is to add β points in the anomalous neighborhood. However, our goal is to estimate if we can do better than the worst case.

Before we present the attack strategy we define a hypersphere $\mathcal{R}$ as follows -

Definition 5.2.1 *Let there be a hypersphere centered at the anomaly point a with radius* $direct_{min}(a) - indirect_{max}(a)$.

We now prove that this region will exist. To prove that the region exists we just have to show that $direct_{min}(a) > indirect_{max}(a)$.

Lemma 5.2.1 *For any point a which is classified as an anomaly by LOF* $direct_{min}(a) > indirect_{max}(a)$.

Proof For a point to be classified as anomaly, it should be in a less dense region as compared to its neighbors. Therefore, $LOF_\beta(a) \geq 1$, which immediately follows from Lemma 5.1.1 and the fact that the average density of its neighborhood is higher than its density. Hence, from Theorem 5.1.1 it follows, that $direct_{min}(a) \geq indirect_{max}(a)$ as $LOF(a) \geq 1$. ■

The attack strategy is presented in Algorithm 6 and illustrated in Figure 5.1. Note, in the figure we have represented $direct_{min}$ as d_{min} and $indirect_{min}$ as i_{min}. Similarly, for $direct_{max}$ and $indirect_{max}$.

Algorithm 3: Adversary's attack strategy in Perfect Knowledge Scenario

Let $N_\beta(a) = p_1, p_2, ..., p_\beta$ be the β neighborhood of a as defined in Definition 5.1.2 sorted in nondecreasing order

Let p be the point s.t. $p \in N_\beta(a)$ and has the minimum reachability distance from i.e., $reachdist_\beta(a, p) = min\{reachdist(p_j, a) | p_j \in N_\beta(a)\}$

$i := 1$

while $LOF_\beta(a) \geq 1$ **do**

  Add attack point $a_i \in \mathcal{A}$ in $\mathcal{R}$ s.t. $dist(a_i, p) < direct_{min}(a)$

  $N_\beta(a) = N_\beta(a) - p_\beta + a_i$

  Update p according to the updated $N_\beta(a)$

  $i++$

end

We now give a proof of correctness for the above algorithm and then quantify the bound on the adversary's effort required to include the attack point in the cluster.

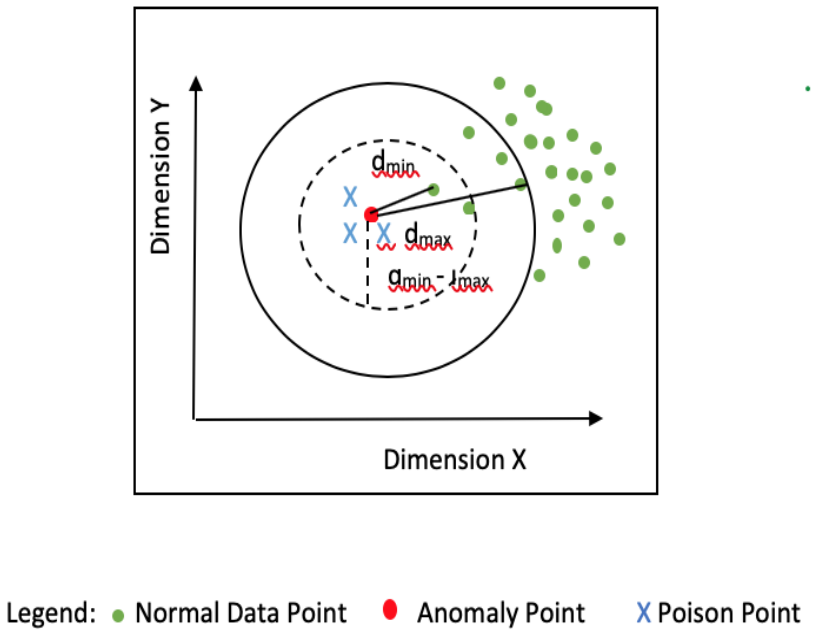

Figure 5.1. Illustration of the attack strategy

Proof of Correctness: To prove the correctness of the algorithm we first prove that the algorithm at each iteration increases the density of the anomaly point a by reducing the value of the local reachability density of the anomaly point. We are adding a point s.t. $dist(a, a_i) < dist(a, p_\beta)$. This, follows from the fact that $dist(a, a_i) < direct_{min}(a) \leq dist(a, p_\beta)$. Hence, at every iteration the density of the anomaly point decreases by p_β to $p_{\beta-1}$.

We now prove that the density of a point $p_i \in N_\beta(a)$ does not increase. Note, we are adding a point a_i s.t. $\forall p \in N_\beta(a), dist(a_i, p) \geq indirect_{max}(a)$. This, follows immediately from how we have defined $\mathcal{R}$. Hence, the density of a point in the high density region of $N_\beta(a)$ does not increase.

To carry out the attack stealthily we add the constraint that $dist(a_i, p) < direct_{min}(a)$. The intuition behind adding this constraint is to ensure that the attack points will not be false negatives by adding them in the region between the anomaly point and the cluster. Note - we have chosen to update the value of $LOF_\beta(a) = 1$ because of

Lemma 5.1.1.

We now quantify the effort required by the adversary. Let $|N_\beta^{\mathcal{D}\cup\mathcal{A}}(a)|$ be the neighborhood of a which contains points from the dataset $\mathcal{D}$ and the attack set $\mathcal{A}$. Further, let us assume that $\ell rd_{\beta-avg}(a)$ is the average local reachability of the β-neighborhood of the anomaly point a after attack points are added i.e., $\ell rd_{\beta-avg}(a) = \frac{\sum_{p_j \in N_\beta^{\mathcal{D}\cup\mathcal{A}_{(a)}}} \ell rd_\beta(p_j)}{|N_\beta^{\mathcal{D}\cup\mathcal{A}}(a)|}$. Let the initial local reachability distance of a before any attack point is added be $\ell rd_{\beta-ini}(a)$. Let the change in the $\ell rd_{\beta-ini}(a)$ due to the change in the β neighborhood of a at iteration i be represented by δ_i. Let $m * \delta = \sum_{i=1}^{m} \delta_i$, where m is the number of iterations or the number of attack points added. Therefore, $\ell rd_{\beta-fin}(a) = \ell rd_{\beta-ini}(a)$ - $m * \delta$.

Theorem 5.2.1 *Let $\ell rd_{\beta-avg}(a)$, $\ell rd_\beta(a)$, m , δ be defined as above, then* $|\mathcal{A}| = m \approx (\ell rd_{\beta-ini}(a) - \frac{\sum_{p_j \in N_\beta^{\mathcal{D}\cup\mathcal{A}_{(a)}}} \ell rd_\beta(p_j)}{|N_\beta^{\mathcal{D}\cup\mathcal{A}}(a)|})/\delta.$

Proof From Lemma 5.1.1 it follows that to make an anomaly point normal, the LOF of the point should be approximately 1. Therefore,

$$
\begin{aligned}
LOF_\beta(a) &= \frac{\sum_{p_j \in N_\beta^{\mathcal{D}\cup\mathcal{A}_{(a)}}} \frac{\ell rd_\beta(p_j)}{\ell rd_{\beta-fin}(a)}}{|N_\beta^{\mathcal{D}\cup\mathcal{A}}(a)|} \approx 1 \\
&= \frac{\sum_{p_j \in N_\beta^{\mathcal{D}\cup\mathcal{A}_{(a)}}} \ell rd_\beta(p_j)}{(\ell rd_{\beta-fin}(a)) * (|N_\beta^{\mathcal{D}\cup\mathcal{A}}(a)|)} \approx 1 \\
&= \frac{\sum_{p_j \in N_\beta^{\mathcal{D}\cup\mathcal{A}_{(a)}}} \ell rd_\beta(p_j)}{|N_\beta^{\mathcal{D}\cup\mathcal{A}}(a)|} \approx \ell rd_{\beta-fin}(a) \\
&= \frac{\sum_{p_j \in N_\beta^{\mathcal{D}\cup\mathcal{A}_{(a)}}} \ell rd_\beta(p_j)}{|N_\beta^{\mathcal{D}\cup\mathcal{A}}(a)|} \approx \ell rd_{\beta-ini}(a) - m * \delta \\
m &\approx (\ell rd_{\beta-ini}(a) - \frac{\sum_{p_j \in N_\beta^{\mathcal{D}\cup\mathcal{A}_{(a)}}} \ell rd_\beta(p_j)}{|N_\beta^{\mathcal{D}\cup\mathcal{A}}(a)|})/\delta
\end{aligned}
$$

■

5.2.2 Adversary's Attack Strategy in Limited Knowledge Scenario

We now bound the efforts of the adversary who has knowledge about the distribution of the data and its algorithm parameters but not the actual data. A brute force approach for making an anomaly point to look like normal will be to add β points in its neighborhood i.e., $|\mathcal{A}| = \beta$, thereby ensuring that the average reachability distance

of its neighborhood is the same as the reachability distance of the anomaly point. In this section, we aim to analyze whether an adversary can do better than the brute force approach with only limited knowledge about the data. Intuitively, we can use some of the existing normal data points in addition to the attack point to increase the density of the anomaly point so that it equals to the average density of its β-neighborhood. Note: we can reduce the attack from a multi-dimensional space to a single dimensional space because we are using the L_k metric to measure the distances. We assume, that there is one high dense region $\mathcal{C}$ which is a subset of $\mathcal{D}$ To do so, we make a few assumptions as follows -

1. The distances are drawn from a Gaussian distribution $Y \sim N(\mu, \sigma)$ with the distances within the range $(\mu \pm 6\sigma)$.

2. The anomaly point is at a distance $dist(a, \mu) = c_1 * \sigma$ from the mean μ of the distribution, where c_1 is some constant.

Let us assume that the local reachability distance i.e., the area of the β-neighborhood of the anomaly point "a" before attack points are added is $dist(a, \mu) \pm l_1$. After, we add the attack points let the new local reachability distance be $dist(a, \mu) \pm l_2$, where $l_2 < l_1$. The local reachability distance will decrease as we are increasing the density of the anomaly point, hence $l_2 < l_1$. Since l_2 is a constant, we can rewrite it as $l_2 = c_2 * \sigma$, where c_2 is some constant. Based on the above assumptions, we calculate the density of the region $\mathcal{S}$ which lies between $dist(a, \mu) \pm l_2$:

$$\mathbb{E}(|S|) = \mathbb{E}_{dist(a,\mu)-l_2}^{dist(a,\mu)+l_2}(Y) = \mathbb{E}_{(c_1-c_2)*\sigma}^{(c_1+c_2)*\sigma}(Y)$$
$$= |\mathcal{D}| * Pr(c_1 - c_2) \leq Y \leq (c_1 + c_2))$$
$$\mathbb{E}|\mathcal{S}| = |\mathcal{D}| * \frac{1}{2}[erf(\frac{(c_1 + c_2) - \mu)}{\sigma \quad \bar{(}2)}) - erf(\frac{(c_1 - c_2) - \mu)}{\sigma \quad \bar{(}2)})] \tag{5.4}$$

Theorem 5.2.2 *For any $|\mathcal{D}|$ and n, if an adversary has perfect knowledge about the training data, then $|\mathcal{A}| = \beta - \mathbb{E}|\mathcal{S}|$*

Proof This immediately follows from how we have defined S and that in the brute force case we require β points to be added. ■

Extension to Multiple Disjoint Clusters:

We have so far assumed that there is a single cluster for the adversary to be included in. We now extend our method to include disjoint clusters. Let there be k disjoint clusters and we assume that each cluster is drawn from a Gaussian Distribution $Y \sim N(\mu_1, \sigma_1), ..., Y \sim N(\mu_k, \sigma_k)$. The size of each cluster is $|\mathcal{C}_k|$, s.t. $\overset{i=k}{\underset{i=1}{}} |\mathcal{C}_k| = |\mathcal{D}|$ and $\{\mathcal{C}_i\} \cap \{\mathcal{C}_j\} = \emptyset$. We calculate the density of the region $\mathcal{S}$ which lies between $dist(a, \mu) \pm l_2$:

$$\mathbb{E}(|S|) = \overset{i=k}{\underset{i=1}{}} \mathbb{E}^{dist(a,\mu)+l_2}_{dist(a,\mu)-l_2}(Y) = \overset{i=k}{\underset{i=1}{}} \mathbb{E}^{(c_1+c_2)*\sigma}_{(c_1-c_2)*\sigma}(Y)$$

$$= \overset{i=k}{\underset{i=1}{}} |\mathcal{D}_i| * Pr(c_1 - c_2) \leq Y \leq (c_1 + c_2))$$

$$\mathbb{E}|\mathcal{S}| = \sum_{i=1}^{i=k} |\mathcal{D}_i| * \frac{1}{2}[erf(\frac{(c_1 + c_2) - \mu_i)}{\sigma_i \quad \bar{}(2)}) - erf(\frac{(c_1 - c_2) - \mu_i)}{\sigma_i \quad \bar{}(2)})] \tag{5.5}$$

Since, the clusters are disjoint we can assume that there will be only one cluster for which $Pr(c_1 - c_2) \leq Y \leq (c_1 + c_2)$ will not be 0. Therefore, our analysis will be similar to when there is one cluster with the difference being in the size of the cluster.

5.2.3 Vulnerability to the Attack

So far, we have developed an attack strategy and bounded the efforts required by the adversary in both the perfect knowledge and limited knowledge scenarios. We now assess the vulnerability of LOF under this attack strategy by analyzing how the false positives and false negatives change with the increase in the number of attack points. To estimate the change in the false positive we model the distance of the data points and the attack points from the centroid of the high dense region of the cluster $\mathcal{C}$ as being drawn from a Gaussian distribution. We state the assumptions formally below and then calculate the change in the false positives.

Assumptions:

1. The distances are drawn from a Gaussian distribution $Y \sim N(\mu, \sigma)$.

2. We assume that there is one adversary only.

3. A false positive is any point $p \in \mathcal{D}$ s.t. $LOF(p) > 1$.

Based on our assumptions, we can calculate the expected false positives as follows:

$$\mathbb{E}(FP) = \sum_{p \in \mathcal{D}} p * Pr(p_{fp}), \text{where} \tag{5.6}$$

$Pr(p_{fp})$ is the probability of d being a false positive

$$Pr(p_{fp}) = Pr(p \in \mathcal{D} | LOF(p) > 1) \tag{5.7}$$

To simplify the calculation of $Pr(p_{fp})$, we assume that there exists a point $c'\sigma$ s.t. all the data points sampled from the range $(\mu - c'\sigma, \mu + c'\sigma)$, are a part of the high density region or cluster $\mathcal{C}$. Since we are assuming that the distance between the data points and the centroid of the cluster are drawn from a Gaussian distribution, we can safely say that the density decreases as we move away from the mean of the Gaussian distribution. Let the anomaly point a be at a distance $dist(a, \mu) = c * \sigma$ from the mean μ of the distribution s.t. $\mu - (c * \sigma) < \mu - (c' * \sigma)$. The anomaly point a is classified as an anomaly because it is in a less dense region as compared to its neighbors. Hence, we can assume that $direct_{min} \geq (\mu - c' * \sigma) - (\mu - c * \sigma)$. This results in the attack points being added in the region $(c * \sigma, c' * \sigma)$, making it dense. Therefore,

$$\mathbb{E}(FP) = |\mathcal{D}| - (|\mathcal{D}| * Pr(c * \sigma \leq Y \leq c' * \sigma))$$

$$\mathbb{E}(FP) = |\mathcal{D}| - (|\mathcal{D}| * (\frac{1}{2}[erf(\frac{c'\sigma - \mu)}{\sigma \quad \overline{(2)}}) - erf(\frac{c\sigma - \mu)}{\sigma \quad \overline{(2)}})]) \tag{5.8}$$

We now predict the change in the false negative rate. We have designed the attack to ensure that the attack points will be classified as normal points, therefore the change in the false negative rate will be $|A| + 1$, where the 1 is added for the anomaly point.

5.3 Experiments

In the previous section we presented an attack strategy and analyzed the expected effort of an adversary in perfect knowledge and limited knowledge scenarios. We now evaluate the assumptions in our theoretical results by applying them to real world datasets. In our experimental scenario, we assume that there is one adversary whose goal is to disguise one anomaly point so as to evade detection. We study the effort required by the adversary in terms of the attack set size and then assess the vulnerability to the attack (note - we repeat this experiment for multiple adversary points, but we assume a single adversary at any given instant). Section 5.3 defines the experimental setup and Section 5.3.1, 5.3.2 & 5.3.3 discusses the results.

Experimental Setup

We have evaluated our methods on five anomaly detection datasets:

1. KDD Cup '99 Dataset [96] - From the KDD cup dataset we have chosen user to root (u2r) attacks (nmap and ipsweep) and denial of service (dos) attacks (back, neptune and teardrop), as these are the attacks where past anomaly detection efforts have shown success. For each attack type we have chosen the minimal feature set as given in [101]. We have randomly sampled 700 normal data points from the training data set ensuring that there were no duplicates. For each attack type we average out our results over 10 different instances. We have chosen β to be 20 and the False Positive Rate (FPR) to be 0.02 for every attack type except neptune for which the FPR is 0.05. These parameters ensure that the True Negative Rate (TNR) is 0.

2. Yahoo! S5 dataset [97] - From the Yahoo dataset we have chosen the A4 benchmark dataset which consists of both anomalies and outliers inserted at random positions. We have set $\beta = 20$ and the initial FPR for the Yahoo dataset is 0.03

3. Thyroid dataset [98]: We have set $\beta = 20$ and the initial FPR is 0.05.

4. Credit Card fraud detection dataset [99] from the Kaggle repository. We have set $\beta = 20$ and the initial FPR is 0.05.

5. CICIDS2017 Intrusion Detection Dataset [100] - This dataset, published by the Canadian Cybersecurity Institute and contains both benign and malicious traffic. From this dataset, we have chosen the DoS Hulk attack type and selected the features as proposed in [100]. We have chosen β to be 20 and the False Positive Rate (FPR) to be 0.03.

5.3.1 Estimation of Adversary's Effort

In this experiment we aim to estimate the effort required by the adversary in terms of the attack size. The results are presented in Figure 5.2 and 5.3 and demonstrate the number of attack points required for the poisoning attack. The results are also compared to the attack size predicted by the model developed in Section 3.2 in the Limited Knowledge Scenario. In the brute force case an attacker needs 20 points to disguise the anomaly point as we have set β to be 20.

The results indicate that with little effort an adversary can alter the decision boundary of the anomaly detection approach. We observe, that on an average an adversary only needs to control 1.4%(7.6) of the training data, which is significantly lower than the parameter β (in our case 20) to succeed with a targeted attack. The results also validate that the effort required by the adversary is comparable to the effort predicted by the model when an adversary has limited knowledge about the training data.

5.3.2 Analyzing the Evasion Rate

We now assess the impact of this attack on LOF by analyzing how the evasion rate of an adversary changes with the increase in the attack set size. Figure 5.4, 5.5

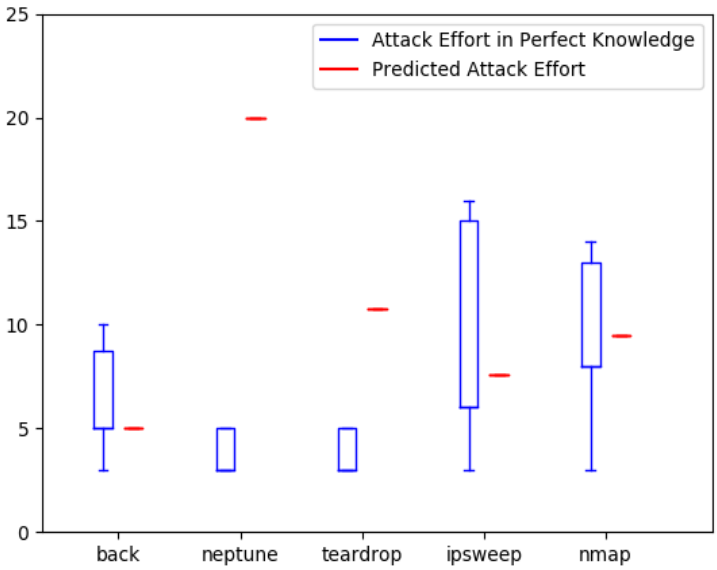

Figure 5.2. Adversary's Effort - KDD'99 Dataset

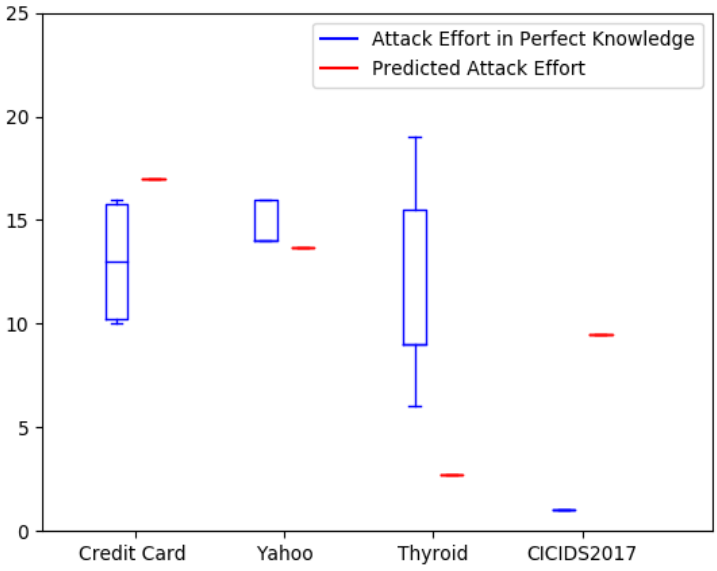

Figure 5.3. Adversary's Effort - Thyroid, Credit card, Yahoo & CICIDS2017 datasets

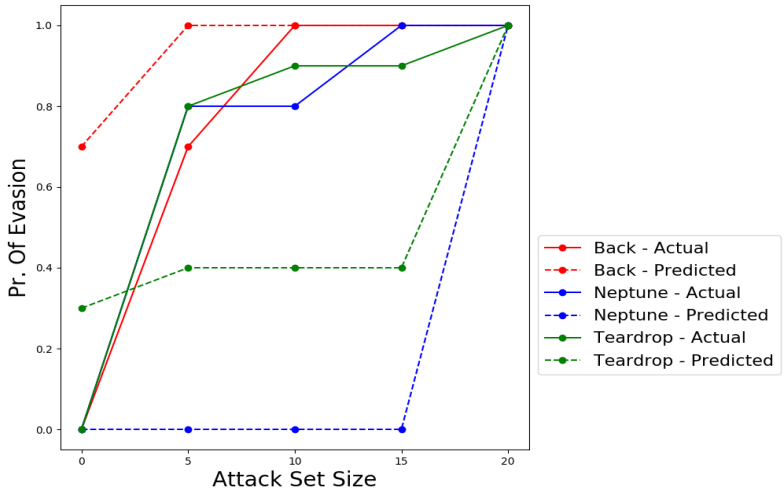

Figure 5.4. Probability of Evasion vs. Size of the Attack Set: KDD Cup '99 Dataset & dos attacks

& 5.6 illustrates the evasion rate for the various attack types and compares it to the rates predicted by the model.

As can be seen from the graphs, on an average, an adversary only needs to control 2.1% of the training data set to increase the evasion rate to 72%. This is because the size of the attack set is determined by the parameter β. A lower value of β indicates that an adversary needs to control only a smaller size neighborhood to ensure that the density of the anomaly point is approximately the same as the density of its neighborhood.

5.3.3 Analyzing the Vulnerability to the Attack

In this experiment, we assess the vulnerability to this attack by analyzing the change in the false positive rate of the anomaly detection approach and comparing it with the predictions made by the model developed in Section 5.2.3. Table 5.1 displays the results.

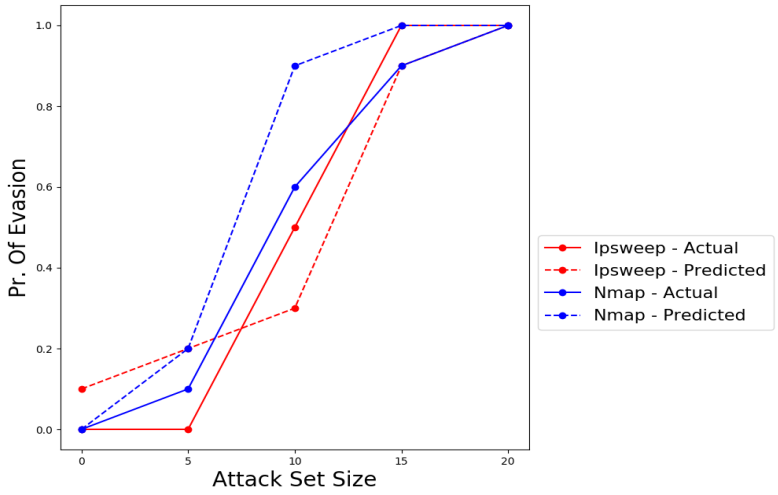

Figure 5.5. Probability of Evasion vs. Size of the Attack Set: KDD Cup '99 Dataset & u2r attacks

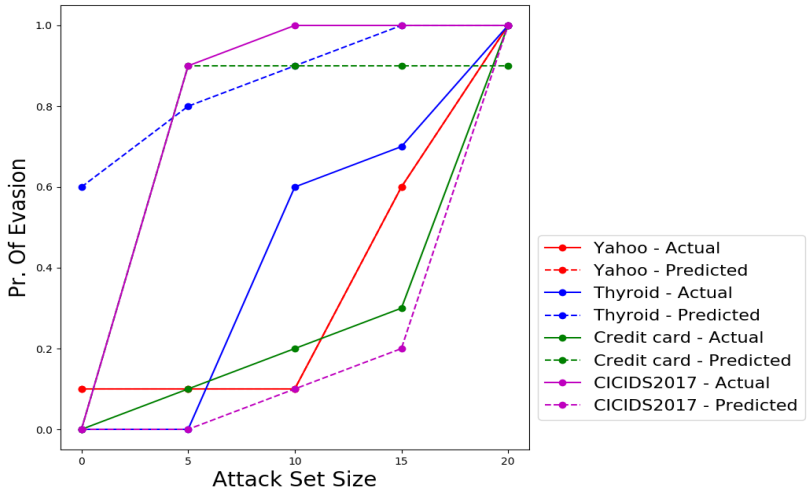

Figure 5.6. Probability of Evasion vs. Size of the Attack Set: Thyroid, Credit card, Yahoo & CICIDS2017 datasets

Attack Type	Original FPR	FPR Post Attack	Expected FPR
back (a1)	0.02	0	0.0185
neptune (a2)	0.05	0.05	0.0185
teardrop (a3)	0.02	0.014	0.017
nmap(a4)	0.02	0.007	0.017
ipsweep(a5)	0.02	0.011	0.016
Yahoo S5	0.03	0.026	0.021
Thyroid	0.05	0.033	0.041
Creditcard	0.02	0.021	0.015
CICIDS2017	0.11	0.08	0.105

Table 5.1.
Adversary's Vulnerability to Attacks

The actual and the expected false positive rates decrease with the increase in the number of attack points. The FPR decreases as a consequence of increasing the density of the anomalous point's neighborhood. An attacker would want to increase the false positives as compared to decreasing the false positive rate so as to render the system unusable. However, our goal here is to carry out a targeted attack requiring us to increase the density of the anomaly point so as to effectively disguise it.

5.4 Detection of the Attack

We have presented an attack strategy and have validated its success empirically. We now present a mechanism for detecting the attacks. To do so, we first define any given object o's sensitivity to its neighborhood as:

Definition 5.4.1 *Sensitivity to Neighborhood of an object o* ($\gamma_c(o)$) - *Let* $\{N_\beta(o)\} = p_1, p_2, ..., p_\beta$ *be the beta-distance neighborhood of o. We define a new neighborhood*

$\{N_{(\beta+c)}(o)\}$ which is formed by randomly replacing $c < \beta$ objects in $\{N_\beta(o)\}$, with $p_{(\beta+1)}, p_{(\beta+1)}, ..., p_{(\beta+c)}$ nearest neighbors. Let $LOF_{\beta+c}(o)$ denote the LOF value of object o w.r.t $(\beta + c)$ neighborhood. The sensitivity to its neighborhood is then calculated as:

$$\gamma_c(o) = (LOF_{\beta+c}(o) - LOF_\beta(o))/LOF_\beta(o) \tag{5.9}$$

We base our detection on the following hypothesis -

Hypothesis: An anomalous point whose density has been increased will have a higher sensitivity to its neighborhood as compared to a normal dense point.

This is based on the rationale that for a highly dense neighborhood the change in the neighborhood by $c < \beta$ objects should not have a significant impact on the LOF value as the object being added will be coming from a similar distribution, but replacing a (carefully placed) attack point will result in a change.

We assume that the attack points are in the region where $LOF_\beta(o)$ is greater than η which is a user defined threshold. This is reasonable, because the optimal attack strategy for an attacker is to minimize the effort required to misclassify the anomalous point. If an attacker wants to decrease the LOF value below a certain threshold it would need to significantly increase its effort i.e., number of attack points that need to be added. The algorithm to detect the attack and the anomalous points is presented in 4.

We now empirically validate our hypothesis by demonstrating the probability of an attack being detected. To do so, we have set up the experiments as described for the attacks. By experimental validation we have set the value of c = 3. The value of η is chosen such that there are around 3% of the objects in our vulnerable region. The results are presented in Table 5.2. They validate our hypothesis that the anomalous points are more sensitive to the changes in the neighborhood; on average, we can detect an adversary with 80% probability. We also observe that when the predictions made by the model in Section 5.2.2 are approximately equivalent to the actual effort we do significantly better at detecting attacks. For an adversary to defeat this attack

Algorithm 5: Detection of an Attack

Input : (c, η)

Output: Attack Points

Calculate the set $\{V\}, s.t. \forall o \in V, LOF(o) \geq \eta$ and o is classified as normal.

for *o in V* **do**
| Calculate $V_\gamma = \{\gamma_c(o)\}$

end

Sort V_γ in increasing order.

Let $\zeta :=$ Number of attacks points predicted by the model.

Investigate the top ζ points in V_γ as anomalous.

Table 5.2.
Probability of Detection

Attack Type	Probability of Detection
back (a1)	0.7
neptune (a2)	0.5
teardrop (a3)	0.5
nmap (a4)	1
ipsweep (a5)	0.8
Yahoo	0.9
Thyroid	1
Credit card	0.9
CICIDS2017	0.2

it requires significantly increasing the attack set size. As such, our estimates are a (loose) lower bound on the expected cost to the attacker. This is still a valuable risk analysis tool for the defender.

5.5 Conclusions

In this chapter, we addressed the problem of evaluating the security of anomaly detection approaches in adversarial conditions by developing an attack strategy and providing bounds on the effort required by the adversary in perfect knowledge scenario and limited knowledge scenario. We have then effectively modeled the vulnerability to these attacks by analyzing the change in the false positive rate and the false negative rates. Further, we have validated our model on real world datasets and have demonstrated that an insignificant amount of effort is required by the adversary for an integrity violation.

One of the main causes for the vulnerability of LOF is because only the density of the β-neighbors is used to determine whether a data instance is anomalous or not. We have exploited this property to show that increasing the density is an optimal attack strategy. We can also reasonably assume that anomaly detection approaches that rely on distance as a metric to classify data instances are more robust to availability attacks than to integrity attacks. Availability attacks are indiscriminate and render the system unusable by causing a high number of misclassifications. Integrity attacks are targeted and involve the misclassification of anomaly points as normal points. This is evident as both the experiments and our model do not show a substantial decrease in the false positive rate, however, the anomaly point is classified as a normal data instance.

6 POISONING ATTACKS AGAINST CLASSIFICATION BASED ANOMALY DETECTION (ONE-CLASS SVM) & DETECTION

6.1 Introduction

Classification based anomaly detection techniques learn a classifier from the given data instances and then classify a test instance as normal or anomalous. Examples include neural networks, bayesian networks, support vector machines (SVM) etc.

One-class SVM [30] was developed for anomaly detection and assumes that the training data belong to one-class, i.e., the normal class. Outliers are any data points that do not belong to this class. It does so by constructing a smooth boundary around a majority of the probability mass of the data. The algorithm proceeds by mapping the input data from the original feature space into a high dimensional feature space using kernel functions and then constructing a smooth boundary by finding a hyperplane to separate the training distance from origin. This formulation relates to two-class SVM, the only difference is that one-class SVM considers normal data to be further away from the origin whereas the anomalous data is closer to origin. Another formulation uses a hypersphere [58] to describe the data and then find the smallest hypersphere which contains the majority of the data.

In this chapter, we focus on one-class SVM and present the optimal attack strategy for a targeted attack and bounds on the adversary's effort. We then quantify the degradation in performance and vulnerability of one-class SVM under this attack.

6.1.1 Background

Here, we briefly review one-class svm as used for anomaly detection. A one-class SVM classifier for anomaly detection, inspired by the SVM classifier [102], was

proposed by Scholkopf et al. in [49]. The one-class classification problem is formulated as finding a hyperplane that separates the data instances (in feature space $\mathcal{X}$) from the origin and maximizes the distance from this hyperplane to the origin. This results in a binary function which captures region $\mathcal{D}$ in the feature space $\mathcal{X}$ where the probability density of the data lives. The classification problem is formulated as follows -

$$f(x) = \begin{cases} +1, \text{if} x \in \mathcal{D} \\ -1, \text{if} x \notin \mathcal{D} \end{cases} \tag{6.1}$$

where x is a data instance, and +1 indicates that it is a normal data point whereas -1 indicates anomalous. This hyperplane cannot be always found in the original feature space, thus a mapping function $\Phi : \mathcal{X} \rightarrow \mathcal{F}$, from $\mathcal{X}$ to a kernel space $\mathcal{F}$ is used. The problem to separate the data set from the origin is formulated as a quadratic optimization problem:

$$min(\frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i - \rho) \tag{6.2}$$

subject to

$$w * \Phi(x_i) \geq \rho - \xi_i, \xi_i \geq 0, i = 1, 2..., n$$

where w is a vector orthogonal to the hyperplane, ν represents the fraction of training data points that are allowed to be outliers or anomalies, x_i is the i^{th} training data point, n is the total number of data instances, $\xi_i = [\xi_1, .., \xi_n]$ is a vector of slack variables used to "penalize" the rejected data instances and ρ represents the margin, i.e., the distance of the hyperplane from the origin.

The decision function for a new data instance x is formulated as follows:

$$f(x) = sign(g(x)) \tag{6.3}$$

where,

$$g(x) = (w * \Phi(x) - \rho) \tag{6.4}$$

The optimization problem is solved by introducing Lagrange multipliers α_i which transforms the decision function to:

$$f(x) = (\ sign)(g(x)) = \sum_{i=1}^{i=n} \alpha_i \Phi(x_i, x) - \rho \tag{6.5}$$

The training data instances x_j for which $\alpha_j > 0$ are referred to as support vectors. ρ is calculated by exploiting the fact that for any α_j, s.t. $0 < \alpha_j < 1/(\nu n)$, the corresponding data instance x_j, satisfies:

$$\rho = w * \Phi(x_j) = \sum_{i=1}^{i=n} \alpha_i \Phi(x_i, x_j) \tag{6.6}$$

We state a proposition here that will be utilized to develop the attack strategy.

Proposition 6.1.1 *Assume the solution of Equation 6.2 satisfies $\rho = 0$. The following statements hold: [49]*

i. ν is an upper bound on the fraction of outliers.

ii. ν is a lower bound on the fraction of Support Vectors (SV).

iii. Suppose the data were generated independently from a distribution P(x) that does not contain discrete components. Suppose, moreover, that the kernel is analytic and non-constant. With probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of outliers.

6.1.2 Framework

We adapt the framework described in Chapter 3 to build the attack strategy:

- Adversary's Goal: The adversary wants to perform a targeted attack (a predetermined point unknown to the defender) i.e., misclassify the anomalous data instance.
- Adversary's Knowledge: We consider two scenarios (i) complete knowledge - adversary has access to the training data & parameters of the model, (ii) limited knowledge - adversary knows only the data distribution.
- Adversary's Capabilities: The capability of an adversary is causative, i.e., he can generate fake data points to disguise the anomaly point.

- Attack Strategy: Once the adversaryś goals, knowledge and capabilities are defined, we can develop an attack strategy. This is achieved by moving the decision boundary so that the anomalous point lies within the hyperplane, as illustrated in Figure 6.1. The hyperplane is moved closer to the origin to include the anomalous data instance. Formally the attack strategy can be stated as

$$min|A|s.t.f(a) = -1, \tag{6.7}$$

where a is the anomalous data instance

6.2 Attack Strategy & Assessing Risk

We have developed an attack model to develop an attack strategy and analyze the vulnerabilities of one-class SVM to the attack. In this section we develop two attack strategies - one in perfect knowledge scenario and one in limited knowledge scenario - and then develop a model to assess the vulnerability of the method to this attack. In this section, we assume the role of an adversary so that we can build an optimal attack strategy to help the defender understand the risk of an attack.

6.2.1 Adversary's Attack Strategy

In this section, we develop two different methods optimized and heuristic) for the attack strategy under the assumption that the adversary knows the training data set $\mathcal{X}$ and the one-class parameters of the model.

Adversary's Attack Strategy based on Optimization in Perfect Knowledge Scenario

For the optimized method, we first present a loss function based on the adversary's goal and then we use gradient descent to minimize the loss function.

For a data point x_a, to be classified as an outlier, $sign(f(x_a)) = -1$. The adversary can achieve his goal by adding data points so that the value of the function $sign(f(x_a))$

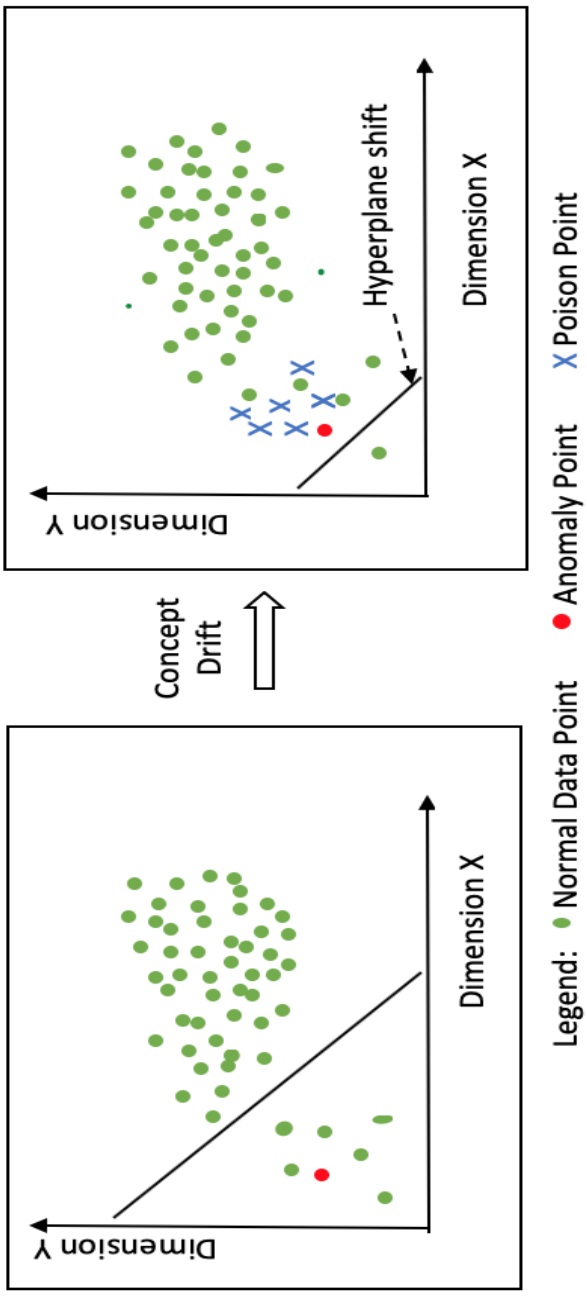

Figure 6.1. Illustration of the Adversary's Attack Strategy (Moving the hyperplane)

changes to +1. To do so, the attacker proceeds by drawing a validation data $\mathcal{X} = \{x_i\}_{i=1}^{n}$, training the one-class SVM on the data and then minimizing the loss function :

$$L_x = (g_{old}(x_a)) - g_{new}(x_a))^2 \tag{6.8}$$

where $g_{old}(x_a)$ is the value given by Equation 6.5 when the model is one-class on the untainted dataset and $g_{new}(x_a))$ is the value that the adversary wants for the anomalous point x_a. The adversary chooses the value of $g_{new}(x_a)$ as 0 because min(g(x)) for which f(x) = +1 is 0. The adversary then uses the gradient descent technique to find a_p which minimizes the value of the function L_x given in 6.8. Substituting $g_{new}(x_a) = 0$ and $g_{old}(x_a) = g(x)$ (for ease of use) in equation 6.8, the poison data point $a_p \in \mathcal{A}$ (where $\mathcal{A}$ is the set of poison points added by the adversary), at each iteration, can be calculated as:

$$a_{p+1} = a_p - 2 * \eta * g'(x) * g(x) \tag{6.9}$$

where η is the learning rate for the gradient descent technique and $g'(x)$ (substituting Eq. 6.6 in Eq. 6.5 to derive the derivative) is given by the following. (We only consider the set of support vectors as the value of $g(x)$ depends only on the support vectors(s), i.e., whose $0 < \alpha_i < 1/(\nu n)$.)

$$g'(x) = \sum_{i=1}^{i=s} \frac{\partial \alpha_i}{\partial a_p} * \Phi(x_p, x_a) + \alpha_p * \Phi'(x_i, x_a) - \sum_{i=1}^{i=s} \frac{\partial \alpha_i}{\partial a_p} * \Phi(x_i, x_j) - \alpha_p * \Phi'(x_p, x_j) \tag{6.10}$$

To calculate $\sum_{i=1}^{i=s} \frac{\partial \alpha_i}{\partial a_p}$, we replace the inner level optimization problem (calculating the hyperplane for one-class svm is a convex optimization problem) with the corresponding Karush-Kuhn-Tucker (KKT) constraints, which must hold at the global optimum:

$$g(x_k) = \begin{cases} \sum_{i=1}^{i=n} \alpha_i * \Phi(x_i, x_k) - \rho > 0 \text{ for } \alpha_k = 0 \\ \sum_{i=1}^{i=n} \alpha_i * \Phi(x_i, x_k) - \rho = 0 \text{ for } 0 < \alpha_k < \frac{1}{\nu n} \\ \sum_{i=1}^{i=n} \alpha_i * \Phi(x_i, x_k) - \rho < 0 \text{ for } \alpha_k = \frac{1}{\nu n} \end{cases} \tag{6.11}$$

Taking the derivative (and only considering the α_k values for which α_k is not constant),

$$\frac{\partial g(x_k)}{\partial x_p} = 0 = \sum_{i=1}^{i=s} \frac{\partial \alpha_i}{\partial a_p} * \Phi(x_i, x_k) + \alpha_p * \Phi'(x_p, x_k) - \\ - \sum_{i=1}^{i=s} \frac{\partial \alpha_i}{\partial a_p} * \Phi(x_p, x_j) - \alpha_p * \Phi'(x_p, x_j) \tag{6.12}$$

Rearranging Eq. 6.12, gives us:

$$\sum_{i=1}^{i=s} \frac{\partial \alpha_i}{\partial a_p} * (\Phi(x_i, x_k) - \Phi(x_i, x_j)) = \alpha_p * (\Phi'(x_p, x_k) - \Phi'(x_p, x_j)) \tag{6.13}$$

which can be rewritten for any i s.t. $0<\alpha_i<\frac{1}{\nu n}$ as:

$$Q_{ss} * \frac{\partial \alpha}{\partial a_p} = \alpha_p * Q'^{T}_{sXd} \tag{6.14}$$

where s denotes the set of α_i for which $g(i) = 0$. $Q_{ik} = [\Phi(x_i, x_k) - \Phi(x_i, x_j)]$, for $k = 1, ..., s$ and a given j. Similarly, $Q'_{kXd} = (\Phi'(x_p, x_k) - \Phi'(x_p, x_j))$, d is the dimension of the data point and $\frac{\partial \alpha}{\partial a_p} = [\frac{\partial \alpha_1}{\partial a_p}, ..., \frac{\partial \alpha_s}{\partial a_p}]$. This gives us:

$$\frac{\partial \alpha}{\partial a_p} = Q_{ss}^{-1} * \alpha_p * Q'^{T}_{sXd} \tag{6.15}$$

Substituting Eq. 6.15 in Eq. 6.10 gives us

$$g'(x) = (K_{sX1}) * (Q_{ss}^{-1} * \alpha_p * Q'_{sd}) - \\ (\alpha_p * (\Phi'(x_p, x_a) - \Phi'(x_p, x_j))) \tag{6.16}$$

where $K_i = (\Phi(x_i, x_a) - \Phi(x_i, x_j))$. The adversary's initial starting point is chosen to be the anomalous point, $a_0 = x_a$.

For linear, polynomial and RBF kernel, we give their derivation, as the loss function depends on their derivative.

1. Linear Kernel, given by $\Phi(a, a') = a^T a'$:

$$\Phi'(a, a') = a \tag{6.17}$$

2. RBF Kernel, given by $\Phi(a, a') = exp(-||a - a'||^2/2\sigma^2)$, where σ is the standard deviation; using chain rule -

$$\Phi'(a, a') = \Phi(a, a') * (1/\sigma^2) * (a - a') \tag{6.18}$$

3. Polynomial Kernel, given by $\Phi(a, a') = (1 + aa')^y$, for any y > 0, using chain rule:

$$\Phi'(a, a') = y * (1 + a^T a')^{y-1} * a \tag{6.19}$$

We now give an algorithm for generating the data points to poison the training set:

Algorithm 6: Adversary's Attack Strategy in Perfect Knowledge Scenario based on Optimization

---

Input : Training Set $\mathcal{X}$

Output: Attack Set $\mathcal{A}$

Calculate α_i using $\mathcal{X}$ and Equation 6.5

Initialize A := $\emptyset$ according to Equation 6.4

while *(curr <0)* **do**

  Initialize p = 1, $a_0 := x_a$ and curr := $g(a_0)$

  prev := curr

  do

    curr := prev := $g(a_0)$ according to Equation 6.4

    Update α_i using $\mathcal{X}$ and Equation 6.5

    Calculate a_{p+1} according to Equation 6.16 depending on the kernel that has been chosen

    curr := $g(a_0)$, calculated according to Equation 6.4

  while *(curr - prev) >δ and curr >prev and curr <0*;

  $\mathcal{X} := \mathcal{X} \cup a_p$

  $\mathcal{A} := \mathcal{A} \cup a_p$

end

Adversary's Attack Strategy based on Heuristic in Limited Knowledge Scenario

In the previous section we have developed a loss function that the adversary employs to generate adversarial points. We now develop a heuristic based strategy based on the intuition that if we add an attack data point a_i which is further away from the hyperplane as compared to the anomalous point x_a, it will be considered more of an outlier than x_a and that ν is an upper bound on the number of outliers in the training data as given in Proposition 6.1.1. The goal of the adversary is to change the decision boundary, which he does so by adding data points which are "more" outliers than the anomalous point. We give an informal proof for this heuristic - there will be two cases i) the number of outliers is ν, ii) the number of outliers is $< \nu$. For case i, if we add an attack point which is "more" of an outlier than the current outliers in $\mathcal{X}$, then the boundary decision will change to ensure that the number of outliers remain ν. As a consequence, at least one data point which was previously classified as outlier will be now classified as normal. We can keep iterating until the anomalous point x_a is classified as normal. For case ii, the effect will be the same but it will require more effort in terms of the attack set size. We now formally state in a proposition the upper bound on the adversary's effort.

Proposition 6.2.1 *Let $\mathcal{A}$ be the set of the attack points that are added to the training data set so that the anomalous point x_a misclassified as normal, i.e., $f(x_a) = +1$. Let $a_i \in \mathcal{A}$, satisfy the following two conditions -*

1. $d(\Phi(a_i)) \geq d(\Phi(x_a))$, *where* $d(\Phi(x)) = \frac{w^T \Phi(x) - \rho}{||w||_2}$.
2. $f(a_i) = -1$

Then $||A|| \leq \nu$.

Proof This proposition directly follows from the fact that ν is an upper bound on the number of outliers and the constraint that we have used to define how to generate a_i. ■

We now give a heuristic on how to generate the adversarial attack points -

$$a_i = a_{i+1} + \epsilon \frac{w^T \Phi(x) - \rho}{||w||_2}, a_0 = x_a \tag{6.20}$$

where ϵ is a user defined parameter which controls the amount of perturbation in the attack point. If the model parameters are known to the adversary, then the adversary can generate adversarial samples as given in Equation 6.20. The main advantage of this attack strategy is that it does not require optimization and hence it is faster to analyze than the strategy developed in the previous section.

6.2.2 Estimation Of Adversary's Effort

In the previous sections we have developed two attack strategies and have estimated an upper bound on the adversary's effort. In this section, we estimate the expected effort required by the adversary. In general, the expected effort required by the adversary should be less than the upper bound as there will be some existing normal data points in the training data that an adversary can use to change the decision boundary. To estimate the expected effort, we make the following assumptions -

1. Let $d(\Phi(x))$ be sampled from a normal distribution $Y \sim N(\mu, \sigma)$, with mean μ and σ estimated from the training data, within the range $\mu \pm 6\sigma$.

2. Let us assume that the anomaly point x_a is at a distance $d(\Phi(x_a)) = c_1 * \sigma$ from the mean μ of the distribution, where c_1 is some constant.

We are adding data points based on our heuristic developed in Section 6.2.1. Therefore, we will be adding points in the region $\mathcal{R} = (c_1 * \sigma, 6 * \sigma)$. The expected density of this region is given by:

$$\begin{aligned}
\mathbb{E}(|\mathcal{R}|) &= \mathbb{E}_{c_1*\sigma}^{6*\sigma}(Y) \\
&= |\mathcal{X}| * Pr(c_1 * \sigma) \leq Y \leq (6 * \sigma)) \\
\mathbb{E}(|\mathcal{R}|) &= |\mathcal{X}| * \frac{1}{2}[erf(\frac{(6 * \sigma) - \mu)}{\sigma \quad \bar{(2)}}) - erf(\frac{(c_1 * \sigma)) - \mu)}{\sigma \quad \bar{(2)}})]
\end{aligned} \tag{6.21}$$

The upper bound on the number of outliers is $\nu * |\mathcal{X}|$. Therefore, The Expected Effort by the adversary is -

$$\mathbb{E}(|\mathcal{A}|) = \nu * |\mathcal{X}| - \mathbb{E}(|\mathcal{R}|) \tag{6.22}$$

6.2.3 Vulnerability to the attack

So far, we have developed an attack strategy and have bounded the effort required by the adversary. We now assess the vulnerability of this attack under this attack strategy by analyzing how the false positives change with the increase in the number of attack points. A false positive is any error when an anomaly detector (incorrectly) rejects a benign input; we measure the change in this rate with the increase in the number of attack points.

To estimate the change in the number of false positives we model the $d(\Phi(x)$ function as being drawn from a Gaussian distribution. We now state the assumptions formally and calculate the change in the false positives.

Assumptions

1. The $d(\Phi(x))$ values are drawn from a Gaussian distribution $Y \sim N(\mu, \sigma)$.
2. We assume that there is one adversary only.
3. A false positive is any point $x \in \mathcal{X}$ s.t. $f(x) = -1$.

Based on our assumptions, we can calculate the expected number of false positives as follows:

$$\mathbb{E}(FP) = \sum_{x \in \mathcal{X}} x * Pr(x_{fp}), \text{where } Pr(x_{fp}) \text{ indicates the probability of x being a false positive} \tag{6.23}$$

$$Pr(x_{fp}) = Pr(x \in \mathcal{X} | f(x) < 0) \tag{6.24}$$

To simplify the calculation of $Pr(x_{fp})$, we assume that there exists a point $c'\sigma$ s.t. all the data points sampled from the range $(\mu - c'\sigma, \mu + c'\sigma)$, are part of the region

$\mathcal{S}$, where $f(x) = +1$. Let the anomaly point x_a is at a distance $d(\Phi(x_a)) = c * \sigma$ from the mean μ of the distribution, where c is some constant. The adversary expands the region to include the anomaly in the region S which results in a reduction of the false positives as more normal data points get included in the region S. As a consequence, $c\sigma > c'\sigma$. The change (decrease) in the expected number of false positives after the addition of attacks points is given by-

$$\mathbb{E}(FP_{dec}) = (|\mathcal{X}| * Pr(c' * \sigma \leq Y \leq c * \sigma))$$

$$\mathbb{E}(FP_{dec}) = (|\mathcal{X}| * (\frac{1}{2}[erf(\frac{c\sigma - \mu)}{\sigma \quad \bar{(}2)}) - erf(\frac{c'\sigma - \mu)}{\sigma \quad \bar{(}2)})]) \tag{6.25}$$

6.3 Experiments & Discussions

In the previous section we presented an attack strategy and have analyzed the impact of an adversary in perfect knowledge and limited knowledge scenarios. We also assessed the vulnerability to the attack by analyzing the change in the false positive rate. We now evaluate the assumptions in our theoretical results by applying them to real world datasets.

Experimental Setup

In our experimental scenario, we assume that there is one adversary whose goal is to disguise some anomaly point so as to evade detection. We study the effort required by the adversary in terms of the attack set size and then assess the vulnerability to the attacks (note - we repeat this experiment for multiple adversary points, but we assume a single adversary at any given instant). We have implemented our experiments on five anomaly detection datasets -

1. KDD Cup'99 intrusion detection dataset [96] From the KDD cup dataset we have chosen user to root (u2r) attacks (nmap and ipsweep) and denial of service (dos) attacks (back, neptune and teardrop) , as these are the attacks where past

anomaly detection efforts have shown success. For each attack type we have chosen the minimal feature set as given in [101].

2. Yahoo! S5 dataset [97] - The yahoo dataset consists of real and synthetic time-series representing the metrics of various Yahoo services. From the yahoo dataset we have chosen the A4 benchmark dataset which consists of both anomalies and outliers inserted at random positions.

3. Thyroid dataset [98] from the UCI machine learning repository.

4. Credit Card fraud detection dataset [99] from the Kaggle repository.

5. CICIDS2017 Intrusion Detection Dataset [100] - This dataset, published by the Canadian Cybersecurity Institute, contains network traffic which was captured over a period of 5 days in 2017. From this dataset, we have chosen the DoS Hulk attack type and selected the features as proposed in [100], as SVM has been shown to perform the best on it [103].

From all the data sets we have randomly sampled 700 normal data points from the training data set ensuring that there were no duplicates. For each attack type we average out our results over 10 different instances. The value of ν has been chosen so as to ensure that the True Positive Rate (TPR) is 1, i.e., all the adversarial points are initially classified as anomalous. The value for the ν parameter is given in Table 6.1 and the degree for the polynomial kernel has been set as 3.

6.3.1 Estimation of Adversary's Effort

In this experiment we aim to estimate the effort required by the adversary in terms of the attack size. The results are presented in Fig. 6.3.1, Fig. 6.3.1, Fig. 6.3.1 & Fig. 6.3.1. We give an average number of attack points required for the adversary to misclassify the anomalous point. The average number of points for the optimization attack for the Gaussian Kernel is 4, Linear Kernel is 3.4 and Polynomial

νparameter / Dataset	RBF Kernel	Polynomial Kernel	Linear Kernel
Credit card	0.01	0.03	0.03
KDD Cup - back	0.02	-	-
KDD Cup - ipsweep	0.02	0.03	0.03
KDD Cup - nmap	0.02	0.03	0.03
KDD Cup - neptune	0.02	0.08	0.05
KDD Cup - teardrop	0.02	0.08	0.05
Yahoo	0.01	-	-
Thyroid	0.01	-	-
CICIDS2017	0.01	-	-

Table 6.1.
The value of ν parameter for different kernels and datasets

is 8.74. The average number of attack points for the heuristic attack for the Gaussian Kernel is 6.3, Linear Kernel is 9.94 and Polynomial is 9.4. These results indicate that for an optimization attack an adversary only needs to control .76% of the dataset to misclassify his point with the range being 0.48% - 1.2%. For a heuristic attack an adversary needs to control only 1.22% with the range being 0.9% - 1.42% of the training data. One thing to note here is that the number of points is dependent on how far the anomalous point is from the hyperplane. If the adversary is close to the hyperplane then in the best case he needs to add only 1 poison point to cause a concept drift.

Another observation from the results is that the average number of points required in the perfect knowledge scenario is lower than the number of points required in the limited knowledge scenario. This validates the fact that optimization requires less effort in terms of the size of the attack set. However, in terms of computational resources the heuristic that we have developed in Section 6.2.1 is faster as it does not need any optimization and is a reasonable approximation to the attack as validated by the results. The results also show that the estimated no. of points for the attack is comparable to the effort required in the limited knowledge scenario attack. (Note - we haven't ran the experiments for the polynomial and the linear kernel on the thyroid and the yahoo dataset because they were already classifying the adversary point as benign.

6.3.2 Analyzing the Evasion Rate

We assess the impact of this attack on one-class SVM by analyzing how the evasion rate of an adversary changes with the increase in the attack set size. Fig. 6.6, 6.7, 6.8 illustrates the evasion rate for the various attack types for the RBF kernel. As can be seen from the graphs, on an average, an adversary only needs to control 1.14% with the maximum being 4.2% and the minimum 0.42% of the training data set to increase the evasion rate to 100%. These results validate that it is feasible to

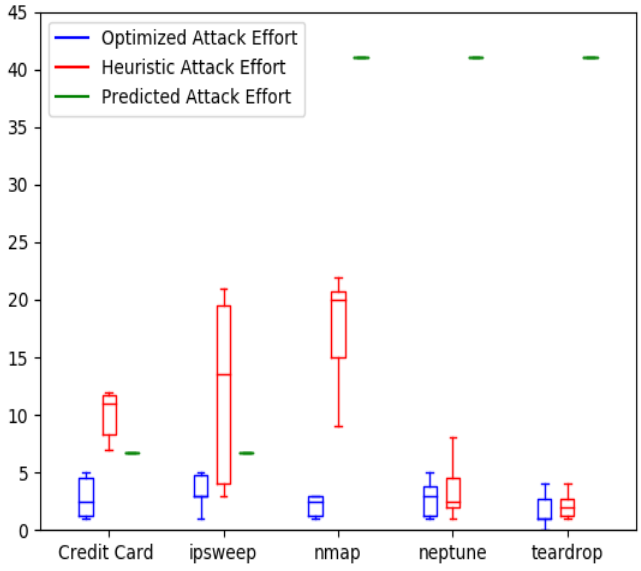

Figure 6.2. Adversary's Effort for Linear Kernel

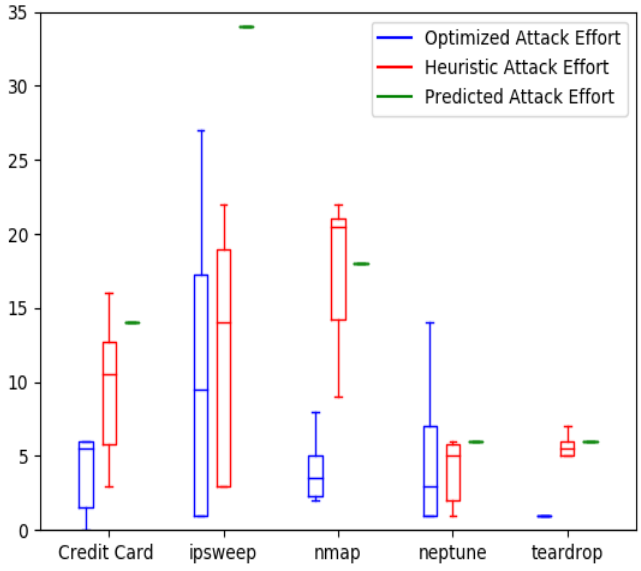

Figure 6.3. Adversary's Effort for Polynomial Kernel

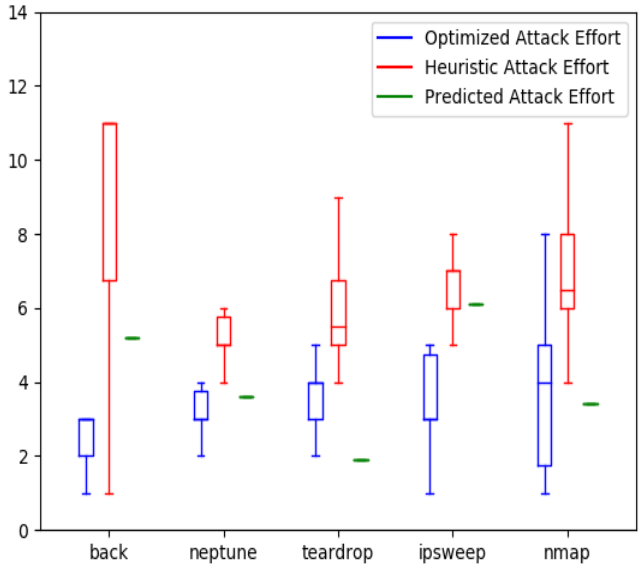

Figure 6.4. Adversary's Effort for Gaussian Kernel: KDD Cup Dataset

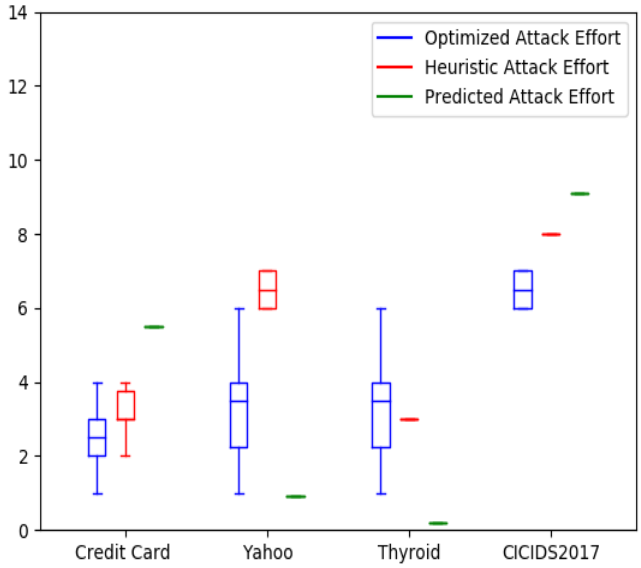

Figure 6.5. Adversary's Effort for Gaussian Kernel: Thyroid, Credit Card & CICIDS2017 Anomaly Detection Datasets

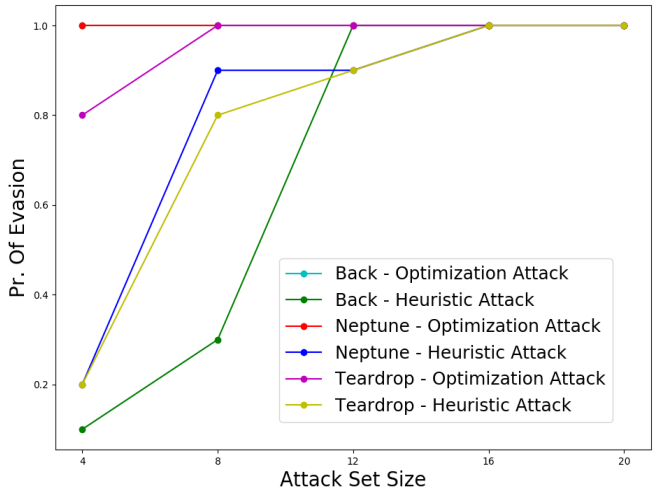

Figure 6.6. Probability of Evasion vs. Size of the Attack Set: KDD Cup'99 Dataset dos attacks

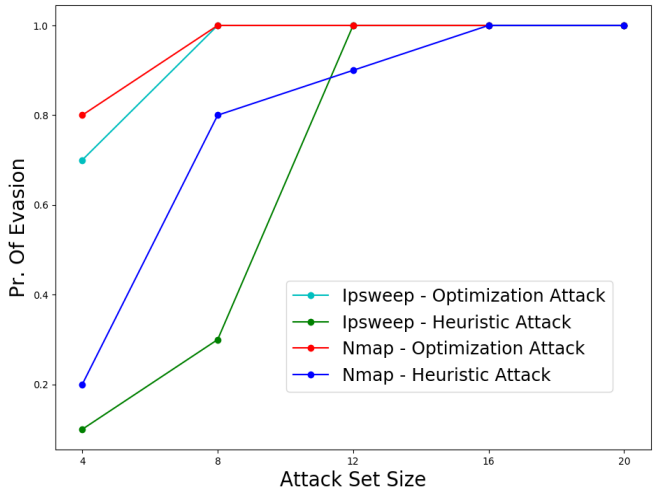

Figure 6.7. Probability of Evasion vs. Size of the Attack Set

attack the machine learning algorithm to cause a concept drift and does not require a substantial effort from the adversary to do so.

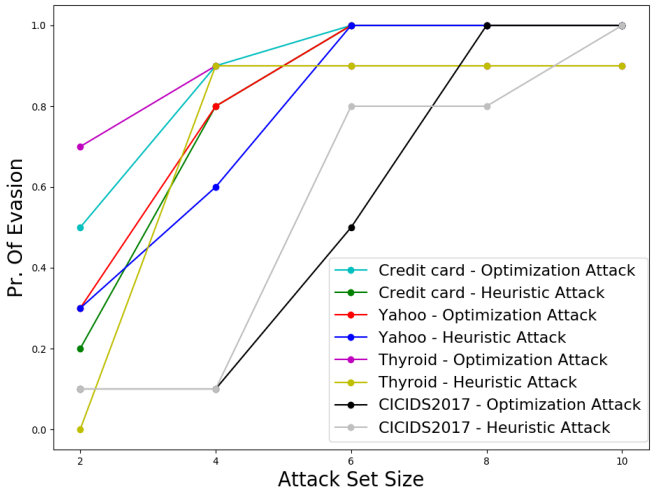

Figure 6.8. Probability of Evasion vs. Size of the Attack Set: Yahoo S5, Thyroid, Credit Card & CICIDS2017 Anomaly Detection Datasets

6.3.3 Analyzing the Vulnerability to the Attack

In this experiment, we assess the vulnerability to this attack by analyzing the change in the false positive rate of the anomaly detection approach for the optimization based attack strategy and the estimated false positives predicted according to the model. The results are presented in Table 6.2. We observe that the addition of attack points decreases the False Positive Rate. This happens because we are moving the hyperplane close to the origin; resulting in benign points that were originally misclassified as being classified correctly due to increasing the size of the capture region. This enables the attacker to carry out the attack stealthily.

6.3.4 Estimation of the effect of the parameter ν

In this experiment we estimate the effect of the parameter ν on the attack set size. As can be seen from the figure 6.9, there is a positive correlation between the value of ν and the attack set size. This is reasonable, as when we increase the value of ν, more data points will be rejected as outliers, which will result in the boundary being further moved away from the origin. This will increase the anomaly score of the anomalous data point, hence the adversary will have to increase its effort to misclassify the point as benign.

6.3.5 Comparison to Previous Work

In this experiment, we compare our attack strategy to an existing method. We specifically look at the poisoning attack strategy developed in [10] and show what our attack strategy reduces the effort of the attacker w.r.t the number of poison points required by the attacker. Their attack strategy aims to poison the dataset to maximize the hinge loss. We argue that this attack strategy is not suited for our attack model because it aims at maximizing the hinge loss i.e., increasing the number of false positives. As a consequence, this does not guarantee that our attacker will be

	RBF Kernel			Polynomial Kernel			Linear Kernel		
	Original FPR	FPR After Attack	Estimated FPR	Original FPR	FPR After Attack	Estimated FPR	Original FPR	FPR After Attack	Estimated FPR
Credit Card	0.011	0.008	0.002	0.020	0.016	0.011	0.009	0.011	0.006
Back	0.012	0.010	0.01	-	-	-	-	-	-
Ipsweep	0.016	0.015	0.01	0.048	0.021	0.046	0.009	0.008	0.009
Nmap	0.020	0.020	0.005	0.026	0.026	0.032	0.058	0.060	0.006
Neptune	0.015	0.012	0.005	0.009	0.004	0.008	0.059	0.058	0.06
Teardrop	0.012	0.009	0.007	0.009	0.004	0.003	0.059	0.058	0.06
Yahoo	0.009	0.008	0.003	-	-	-	-	-	-
Thyroid	0.010	0.009	0.006	-	-	-	-	-	-
CICIDS2017	0.010	0.05	0.001	-	-	-	-	-	-

Table 6.2.
Adversary's Vulnerability To Attack

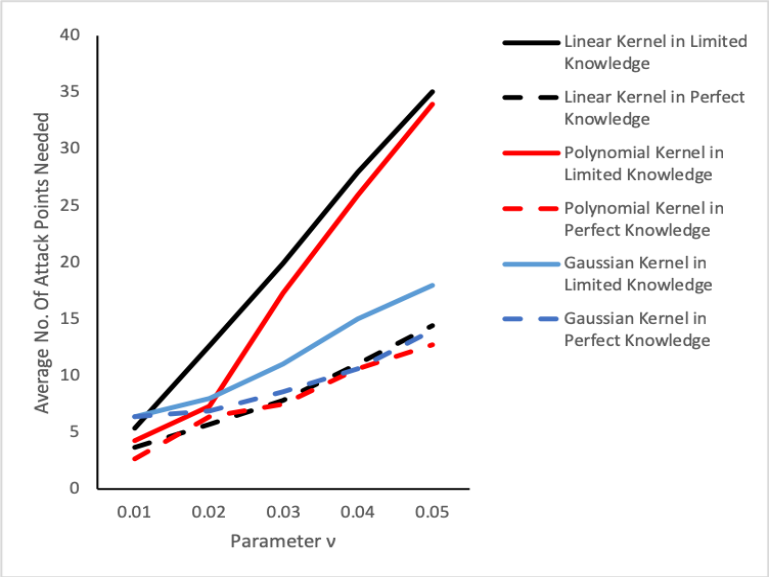

Figure 6.9. Average No. Of Attack Points needed vs. Size of ν

misclassified or that there will be a decrease in the false positives so that the defender will continue using the technique. Our comparison is presented in Table 6.3. We have compared the results for the Gaussian kernel, as for the Gaussian Kernel every attack was identified as malicious. As can see from the results, that the attack strategy presented in [10] is not always successful (the probability of success fo our strategy was 1.0.), and the effort required by the adversary is also more with the exception being nmap, yahoo and thyroid. An interesting observation is that for the nmap and yahoo dataset the adversary requires less effort in their strategy as compared to ours but the probability of success is significantly lower. Only 1 adversarial point was able to successfully evade the attack. For the thyroid dataset, the average number of attack points is higher for us because of one adversarial data instance for which we required more effort; but that same data instance could not evade detection using the attack strategy presented in [10].

6.4 Detection Techniques & Discussions

After presenting attack strategies and estimating the expected effort required by the adversary to subvert the anomaly detection technique, for completeness we now present a few detection strategies and open research directions which could help in building robust anomaly detection techniques.

6.4.1 Related Work in Detection Techniques

We look at a few existing approaches and evaluate if they can detect our attack or not:

1. Reject on Negative Impact (RONI) is a defense proposed by Barreno et al. [77] that measures the empirical effect of each training instance and eliminates from the training set those data points that have a negative impact on classification accuracy which exceeds a user defined threshold. This technique cannot be ex-

	Poisoning Attack Strategy of [10]		Our Optimization Attack Strategy
Dataset	Pr. Of Success	Average. No. Of Attack Points	Average No. Of Attack Points
Credit card	1.0	2.8	2.4
KDD Cup - back	0.5	6.4	2.7
KDD Cup - ipsweep	0.4	6.25	4.2
KDD Cup - nmap	0.1	3	4.9
KDD Cup - neptune	0.9	10.5	3.1
KDD Cup - teardrop	0.6	6.8	3.7
Yahoo	0.5	1	3.3
Thyroid	0.9	3.3	5
CICIDS2017	0.1	1	5.9

Table 6.3.
Comparison of our Attack Strategy to the one proposed in [10]

tended to anomaly detection because the goal of the adversary is not to decrease the classifier's inaccuracy but to misclassify its own anomalous points. If a classifiers inaccuracy goes below a user defined threshold then with high probability the defender is likely to retrain the model, change the model parameters or use a new technique. As we have seen from the experiments and the theoretical results, adding attack points lead to a decrease in the number of false positives, which is an improvement in the classifiers performance. Therefore, this defense will not detect the added poison points.

2. Ensemble Methods combines several classifiers into one to obtain better predictive performance. One of the ensemble methods is the bagging technique where each individual SVM is trained over a randomly chosen subset of the training set $\mathcal{X}$ via a bootstrap technique and the independently trained SVMs are aggregated in various ways such as majority voting. Ensemble methods have shown to be successful in evading adversarial attacks [104,105]. We explore this technique in Section 6.4.2 and modify it with the insights gained in Section 6.2.

6.4.2 Detection Technique based on Ensemble Methods

Limitation of Ensemble Technique

Here, we show that the naive ensemble technique is inefficient at classifying the anomalous point. The bagging technique generate c new training sets $\mathcal{X}_i$, each of size n', by sampling from $\mathcal{X}$ uniformly and with replacement. In the Table 6.4.2, we show for different values of n' and the optimization attack on the Gaussian Kernel, the probability that our anomalous point is detected. We have set c to be 10 and have combined the models by average voting. As we can see from the table, that the probability of detection has a mode value of 0.1 with its maximum for the Thyroid dataset when we are using 80% of the original training dataset. This is because on an average the number of poison points added are 0.5%, which is significantly small,

	n' = .95n	n' = .90n	n' = .85n	n' = .80n
Credit card	0.3	0.3	0.2	0.3
KDD Cup - back	0.0	0.1	0.1	0.1
KDD Cup - ipsweep	0.2	0.2	0.1	0.1
KDD Cup - nmap	0	0.1	0.1	0.2
KDD Cup - neptune	0.1	0.1	0.1	0.2
KDD Cup - teardrop	0.4	0.3	0.2	0.3
Yahoo	0.1	0.3	0.3	0.3
Thyroid	0.2	0.2	0.3	0.7
CICIDS2017	0.1	0.1	0.1	0.1

Table 6.4.
Pr. Of Detection for the Ensemble Approach

hence they are not excluded when we are training on c new datasets. We have run the detection method on the optimization attack strategy for the gaussian kernel.

Our Approach using Ensemble Technique

We observe from the attack strategy developed in Section 6.2, to cause an influence the attack points added are in a region which is closer to the hyperplane. This is because the adversary is trying to minimize effort, hence the anomalous points will be close to the hyperplane. If it is further away, then the adversary will have increased effort. We propose a modification in the ensemble technique to detect this attack. Let $g(x)$ be the value calculated according to Eq. 6.4 for $x = 1, ...n$. Sort the dataset $\mathcal{X}$ based on the absolute value of $g(x)$ in increasing order. We then divide $\mathcal{X}$ in two subsets $\mathcal{X}_1 = \mathcal{X}_{1,...,\tau}$, and $\mathcal{X}_2 = \mathcal{X}_{\tau,...,n}$, where τ is a user defined parameter; and use a different subset size for each subset to randomly sample with replacement. This gives a higher probability of removing the attack points while training the model, as

the optimization attack strategy generates points to moves the hyperplane between the origin and the anomalous point. This will result, with a high probability, that the attack points and the anomalous points to be relatively closer to the hyperplane as compared to the normal data instance. We now give an algorithm to detect the attack:

Algorithm 7: Detection Strategy

Input : Poisoned Data Set $\mathcal{X}$, Parameter to split the training set: τ, No. of training datasets: c, Weight for each subset: θ_1, θ_2

Output: Model based on Bagging

Calculate α_i using $\mathcal{X}$ and Equation 6.5

Calculate $g(\mathcal{X})$ according to Eq. 6.4

Let $\mathcal{X}$ be arranged so that it is in sorted in increasing order based on $abs(g(\mathcal{X})$

Split $\mathcal{X}$ in two subsets $\mathcal{X}_1 = \mathcal{X}_{1,\ldots,\tau}$, and $\mathcal{X}_2 = \mathcal{X}_{\tau,\ldots,n}$

for *iter in range(1, c)* **do**

  Randomly Sample $n_1' = \theta_1 * n$ samples from $\mathcal{X}_1$ and $n_2' = \theta_2 * n$ from $\mathcal{X}_2$

  Calculate α_i^{iter} using $\mathcal{X}_1$, $\mathcal{X}_2$ and Equation 6.5

end

Combine the models by average voting

Experiment & Results

We now present our results for the Gaussian Kernel based on the training dataset which has been poisoned using the optimization attack strategy. We have set c to be 10 as earlier, $\tau = 15$, $\theta_1 = 0.7$ so that on an average 4 (average no. of attack points) points are removed from this subset of the dataset. We have set the value of $\theta_2 = 0.99$, so that maximum number of normal data points are included in the training data. The results are presented in 6.4.2.

As we can see from the results, we are successful in detecting the target anomalous data point, but at the expense of an increase in the false positive rate.

	Pr. Of Detection	FPR
Credit card	0.9	0.020
KDD Cup - back	0.9	0.041
KDD Cup - ipsweep	0.9	0.0271
KDD Cup - nmap	0.7	0.056
KDD Cup - neptune	0.9	0.026
KDD Cup - teardrop	0.8	0.0191
Yahoo	0.6	0.0202
Thyroid	0.7	0.0224
CICIDS2017	0.6	0.0561

Table 6.5.
Pr. Of Detection for our Approach

6.5 Conclusions

In this work, we have analyzed the robustness of one-class SVM as an anomaly detection technique against poisoning attack. We have first presented two attack strategies and then have analyzed the risk of one-class SVM to those attacks by analyzing the expected effort required by the adversary to achieve his goal and then estimating the change in the number of False Positives. We were able to exploit the weakness of one-class SVM - its gradients could be easily calculated easily and that there is an upper bound on the fraction of the number of outliers to poison the dataset. We have also presented a detection strategy for the attack and have proved it to be more effective at detecting the attack.

7 TRANSFERABILITY OF ADVERSARIAL SAMPLES

7.1 Introduction

Transferability of adversarial samples is a property which allows the adversarial samples which are generated for one classifier, to be transferred to another classifier even if the classifiers are trained on disjoint datasets or have different architectures. Transferability of adversarial samples, gives the adversary a powerful tool to evade machine learning systems as the adversary does not require any knowledge of the system. An adversary can train a local model by issuing queries to the targeted model and then use the local model's parameters to craft adversarial samples that can subvert the targeted model. Transferability has been investigated mostly for evasion attacks.

In this chapter, we analyze whether the adversarial samples crafted for poisoning attacks on each anomaly detection technique transfer to the other anomaly detection technique.

7.2 Related Work

In this section, we briefly review the related work on transferability of adversarial samples. Transferability was first studied in [82] for evasion attacks, where they used this property to build a substitute model, whose parameters were used to craft adversarial samples, and then these samples were used as a target against the victim model. Kurkain et al. [106], further proved that adversarial samples transfer by various degrees. The samples which have a high resistance to adversarial training have a lower probability of transferring. Tramèr et al. [107] have shown that for

different models the adversarial subspace intersects, hence it is possible to transfer the samples.

Liu et al. [108] have empirically validated that targeted adversarial samples show resistance in transferring but the non targeted adversarial samples can transfer for large datasets and large models. They have used ensemble models to generate adversarial samples which can be transferred for a targeted attack. However, they have only focussed on evasion attacks and not on poisoning attacks. The concept of transferability has been successfully applied to even those neural networks that have been training on different domains [109]. They have proposed a generative framework to beat domain specific attack methods.

7.3 Results & Discussions

We now empirically validate for poisonous attacks, whether the adversarial samples transfer or not. We first discuss our experimental setup and then provide our results.

7.3.1 Experimental Setup

In our experimental scenario, the adversarial samples generated for DBSCAN (LOF and one-class SVM) are used to train LOF, one-class SVM (one-class SVM, DBSCAN for LOF and LOF, DBSCAN for one-class SVM) respectively. The hyper-parameters of the models are chosen to be the same as described in the Section 4.3 (5.3, 6.3) respectively. For every attack type, we assume that there are 10 different instances of the adversary. We then estimate the probability of transferring the adversarial samples by calculating the ratio of the number of instances which are predicted benign by the other anomaly detection methods to the total number of instances (10 in our case).

7.3.2 Results & Discussions

Our results are presented in Table 7.1, 7.2 & 7.3. The results depict the probability of the adversarial samples being transferred to another model, for each individual anomaly detection method (For the one-class SVM we have used the Gaussian Kernel as it had a True Negative Rate of 1 for every attack type). We now discuss our results for each individual anomaly detection method:

1. Transferability of Adversarial Samples generated for DBSCAN to LOF and one-class SVM : As we can see from the Table 7.1, the adversarial samples transfer to LOF but not to one-class SVM. The adversarial samples transfer to LOF because of two reasons - the number of attack points needed for DBSCAN (6%) is higher than LOF (1.4%), and the attack strategy for DBSCAN is to make the neighborhood dense by adding data points within ϵ radius. As the number of attack points added in DBSCAN is higher than the β parameter of LOF, this results in the anomalous point having a similar density as compared to its neighbors (attack points). Hence, the adversarial samples transfer from DBSCAN to LOF. However, the adversarial samples do not transfer from DBSCAN to one-class SVM. This indicates that the subspace in which we add the samples is different. One important observation to note here, is that even if the adversary increases his effort (DBSCAN needs on an average 6% while one-class SVM needs only 0.76%), it will not result in a misclassification of the anomalous point. This indicates that an adversary cannot add data points randomly to result in a misclassification but has to carefully craft the samples.

2. Transferability of Adversarial Samples generated for LOF to DBSCAN and one-class SVM : As we can see from Table 7.2 that the LOF adversarial samples do not transfer to DBSCAN and one-class SVM. The reason is intuitive for DBSCAN, it needs more adversarial samples than LOF, hence the anomalous point is classified as malicious. For one-class SVM the adversarial samples do

	Pr. of Attack Transfer to SVM	Pr. of Attack Transfer to LOF
Credit card	0	0.5
KDD Cup - back	0.1	1
KDD Cup - ipsweep	0.2	1
KDD Cup - nmap	0.1	1
KDD Cup - neptune	0	1
KDD Cup - teardrop	0	1
Yahoo	0.2	0.1
Thyroid	0.1	1
CICIDS2017	0.1	1

Table 7.1.
Pr. of Transferring DBSCAN adversarial samples to LOF & SVM

	Pr. of Attack Transfer to SVM	Pr. of Attack Transfer to DBSCAN
Credit card	0.9	0.1
KDD Cup - back	0	0
KDD Cup - ipsweep	0	0
KDD Cup - nmap	0.1	0
KDD Cup - neptune	0.1	0
KDD Cup - teardrop	0	0
Yahoo	0.2	0.1
Thyroid	0.5	0.1
CICIDS2017	0.1	0.1

Table 7.2.
Pr. of Transferring LOF adversarial samples to SVM & DBSCAN

not transfer, indicating that clustering techniques learn a different boundary as compared to classification techniques.

3. Transferability of Adversarial Samples generated for one-class SVM to DBSCAN and LOF: As we can see from the Table 7.3, the adversarial samples don't transfer to LOF and DBSCAN. The reason behind the adversarial samples not transferring to DBSCAN is that the effort required by the adversary is significantly more that the effort required for one-class SVM (6% vs 0.76%). Similarly for LOF, which needs an average of 1.4% of the training data. However, since the no. of attack points needed for one-class SVM is closer to LOF as compared to DBSCAN, the probability of transferring adversarial samples is higher for LOF.

	Pr. of Attack Transfer to LOF	Pr. of Attack Transfer to DBSCAN
Credit card	0	0
KDD Cup - back	0.5	0.1
KDD Cup - ipsweep	0.6	0.1
KDD Cup - nmap	0.3	0.4
KDD Cup - neptune	0.6	0.1
KDD Cup - teardrop	0.5	0
Yahoo	0.3	0
Thyroid	0.1	0
CICIDS2017	1	0

Table 7.3.
Pr. of Transferring SVM adversarial samples to LOF & DBSCAN

These results are also indicative of how we should be designing robust anomaly detection techniques. Since the adversarial samples do not transfer we can build heterogeneous systems for detection anomalies and are robust to adversarial samples.

7.4 Conclusions

In this chapter, we have investigated whether adversarial samples transfer for poisoning attacks. We have shown that they don't transfer with the exception being DBSCAN to LOF. One of the reasons for this is that the effort required for every anomaly detection technique is different. Another reason is that the attack strategy entails adding data points in different subspaces of the feature space.

8 CONCLUSIONS & FUTURE WORK

In this thesis, we have focussed on poisoning attacks, i.e., carefully crafted benign samples that an adversary injects into the training data to cause a concept drift in the anomaly detection method. We have presented a model to estimate the vulnerability of an anomaly detection method to an unknown attack, in particular the expected minimum number of poison samples the adversary would need to succeed. Such an estimate is a necessary step in risk analysis: do we expect the anomaly detection to be sufficiently robust to be useful in the face of attacks?

8.1 Summary of Main Results

8.1.1 Estimated Adversary's Effort

We have addressed the problem of evaluating the security of anomaly detection approaches in adversarial conditions by developing an attack strategy and providing bounds on the effort required by the adversary in perfect knowledge and limited knowledge scenario for DBSCAN (Chapter 4), LOF (Chapter 5) and one-class SVM (Chapter 6). We have demonstrated that the attack strategy is dependent on the underlying anomaly detection approach and varies for each individual approach. The maximum effort is required when trying to circumvent DBSCAN (6%) and minimum for one-class SVM (.76%).

8.1.2 Estimation of Vulnerability to an Attack

We have effectively modeled the vulnerability to these attacks by analyzing the change in the false positive rate and the false negative rates for DBSCAN (Chapter 4), LOF (Chapter 5) and one-class SVM (Chapter 6). We have demonstrated that

our attack strategy results in a decrease in the false positive, giving an advantage to the attacker as the defender will be unaware of the attack.

8.1.3 Detection of Attacks

We have developed mechanisms to detect the attack for DBSCAN (Chapter 4), LOF (Chapter 5) and one-class SVM (Chapter 6). The detection mechanism is based on the attacker's strategy and have shown that our detection mechanism is successful in detecting the attacks. We have demonstrated empirically that if the adversary's attack strategy is optimal, our detection mechanism can detect on an average 83% of the attacks.

8.1.4 Transferability of Attacks

In Chapter 7, we have demonstrated that targeted poisoning attacks do not transfer across the anomaly detection approaches. This results indicates, that to build a robust anomaly detection technique we should look at using heterogeneous ensemble anomaly detection approaches.

8.2 Future Work

While, we have presented a model to estimate the vulnerability to a poisoning attack our model can be expanded to include more scenarios. E.g., one of the main limitations of our model is that we haven't considered a setting where they are multiple adversaries and how they will affect the security of the anomaly detection approach. If multiple adversaries collude then it could lead to a decrease in the effort required by the adversaries. It will be interesting to investigate on how to model collusions between multiple adversaries. Another future work will be to investigate if a similar quantitative model can be developed for other methods for anomaly detection. Another limitation of our model is that we have assumed that the adversary knows

the anomaly detection technique. It would be interesting to test if uncertainty in the anomaly detection technique makes a poisoning/obfuscation attack significantly more difficult or easier to detect.

Another future extension of our work will be to design secure algorithms for anomaly detection. Designing an anomaly detection approach which assumes the presence of an adversary can be developed by applying statistically robust techniques or by modeling it as a game between the adversary and the defender [110]. This has been done for supervised prediction techniques like logistic regression where the game was modeled as a static prediction game [111]; it is not clear if the ideas in [111] would be useful for anomaly detection.

REFERENCES

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[2] Giancarlo Fortino, Roberta Giannantonio, Raffaele Gravina, Philip Kuryloski, and Roozbeh Jafari. Enabling effective programming and flexible management of efficient body sensor network applications. *IEEE Transactions on Human-Machine Systems*, 43(1):115–133, 2013.

[3] Ahmad Alzghoul, Magnus Löfstrand, and Björn Backe. Data stream forecasting for system fault prediction. *Computers & industrial engineering*, 62(4):972–978, 2012.

[4] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1):42–57, 2013.

[5] Maria-Luiza Antonie, Osmar R Zaiane, and Alexandru Coman. Application of data mining techniques for medical image classification. In *Proceedings of the Second International Conference on Multimedia Data Mining*, pages 94–101. Springer-Verlag, 2001.

[6] Luis Martí, Nayat Sanchez-Pi, José Manuel Molina, and Ana Cristina Bicharra Garcia. Anomaly detection based on sensor data in petroleum industry applications. *Sensors*, 15(2):2774–2797, 2015.

[7] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.

[8] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles A Sutton, J Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. *LEET*, 8:1–9, 2008.

[9] Patrick McDaniel, Nicolas Papernot, and Z Berkay Celik. Machine learning in adversarial settings. *IEEE Security & Privacy*, 14(3):68–72, 2016.

[10] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[11] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[12] Pavel Laskov and Marius Kloft. A framework for quantitative security analysis of machine learning. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, pages 1–4. ACM, 2009.

[13] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[14] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4):984–996, 2014.

[15] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[17] Igino Corona, Giorgio Giacinto, and Fabio Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, 239:201–225, 2013.

[18] Xueyuan Han, Thomas Pasquier, and Margo Seltzer. Provenance-based intrusion detection: opportunities and challenges. In *10th {USENIX} Workshop on the Theory and Practice of Provenance (TaPP 2018)*, 2018.

[19] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[20] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[21] Mete Çelik, Filiz Dadaşer-Çelik, and Ahmet Şakir Dokuz. Anomaly detection in temperature data using dbscan algorithm. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*, pages 91–95. IEEE, 2011.

[22] Tran Manh Thang and Juntae Kim. The anomaly detection by using dbscan clustering with multiple parameters. In *2011 International Conference on Information Science and Applications*, pages 1–5. IEEE, 2011.

[23] Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342. Australian Computer Society, Inc., 2005.

[24] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[25] Malak Alshawabkeh, Byunghyun Jang, and David Kaeli. Accelerating the local outlier factor algorithm on a gpu for intrusion detection systems. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, pages 104–110. ACM, 2010.

[26] Zengan Gao. Application of cluster-based local outlier factor algorithm in anti-money laundering. In *Management and Service Science, 2009. MASS'09. International Conference on*, pages 1–4. IEEE, 2009.

[27] Junshui Ma and Simon Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1741–1745. IEEE, 2003.

[28] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.

[29] Katherine Heller, Krysta Svore, Angelos D Keromytis, and Salvatore Stolfo. One class support vector machines for detecting anomalous windows registry accesses. 2003.

[30] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[31] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[32] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[33] Mohamed Idhammad, Karim Afdel, and Mustapha Belouch. Distributed intrusion detection system for cloud environments based on data mining techniques. *Procedia Computer Science*, 127:35–41, 2018.

[34] Chunhua Wang and Dong Han. Credit card fraud forecasting model based on clustering analysis and integrated support vector machine. *Cluster Computing*, pages 1–6, 2018.

[35] Mohamed Hegazy, Ahmed Madian, and Mohamed Ragaie. Enhanced fraud miner: credit card fraud detection using clustering data mining techniques. *Egyptian Computer Science Journal (ISSN: 1110–2586)*, 40(03), 2016.

[36] Klaus Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM transactions on information and system security (TISSEC)*, 6(4):443–471, 2003.

[37] Leonid Portnoy. *Intrusion detection with unlabeled data using clustering*. PhD thesis, Columbia University, 2000.

[38] Lishuai Li, Maxime Gariel, R John Hansman, and Rafael Palacios. Anomaly detection in onboard-recorded flight data using cluster analysis. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, pages 4A4–1. IEEE, 2011.

[39] Shahaboddin Shamshirband, Amineh Amini, Nor Badrul Anuar, Miss Laiha Mat Kiah, Ying Wah Teh, and Steven Furnell. D-ficca: A density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks. *Measurement*, 55:212–226, 2014.

[40] Zhenguo Chen and Yong Fei Li. Anomaly detection based on enhanced dbscan algorithm. *Procedia Engineering*, 15:178–182, 2011.

[41] Akash Saxena, Khushboo Saxena, and Jayanti Goyal. Hybrid technique based on dbscan for selection of improved features for intrusion detection system. In *Emerging Trends in Expert Applications and Security*, pages 365–377. Springer, 2019.

[42] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

[43] Daniel T Larose and Chantal D Larose. *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[44] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM international conference on data mining*, pages 243–254. SIAM, 2008.

[45] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 315–326. IEEE, 2003.

[46] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 535–548. Springer, 2002.

[47] Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 430–433. IEEE, 2004.

[48] Claudio De Stefano, Carlo Sansone, and Mario Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94, 2000.

[49] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[50] Sebastian Mika. Kernel fisher discriminants. 2003.

[51] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[52] Manuel Davy and Simon Godsill. Detection of abrupt spectral changes using support vector machines an application to audio signal segmentation. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–1313. IEEE, 2002.

[53] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[54] Javier Alvarez Cid-Fuentes, Claudia Szabo, and Katrina Falkner. Adaptive performance anomaly detection in distributed systems using online svms. *IEEE Transactions on Dependable and Secure Computing*, 2018.

[55] Santhosh Kelathodi Kumaran, Debi Prosad Dogra, and Partha Pratim Roy. Anomaly detection in road traffic using visual surveillance: A survey. *arXiv preprint arXiv:1901.08292*, 2019.

[56] Salah Zidi, Tarek Moulahi, and Bechir Alaya. Fault detection in wireless sensor networks through svm classifier. *IEEE Sensors Journal*, 18(1):340–347, 2018.

[57] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11-13):1191–1199, 1999.

[58] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[59] Hugo Martins, Luis Brito Palma, Alberto Cardoso, and Paulo Gil. A support vector machine based technique for online detection of outliers in transient time series. pages 1–6, 05 2015.

[60] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[61] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38. ACM, 2017.

[62] Hope Reese. Why microsoft?s tay ai bot went wrong. *Tech Republic*, 2016.

[63] Ellie Zolfaghharifard and Joseph Menn. Did kaspersky lab try to trick rivals into deleting harmless files? former employees say security firm created fake malware. *Daily Mail*, Aug 2015.

[64] Marius Kloft and Pavel Laskov. Security analysis of online centroid anomaly detection. *Journal of Machine Learning Research*, 13(Dec):3681–3724, 2012.

[65] Blaine Nelson and Anthony D Joseph. Bounding an attack's complexity for a simple learning model. In *Proc. of the First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML), Saint-Malo, France*, 2006.

[66] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68. ACM, 2006.

[67] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2009.

[68] James Newsome, Brad Karp, and Dawn Song. Paragraph: Thwarting signature learning by training maliciously. In *International Workshop on Recent Advances in Intrusion Detection*, pages 81–105. Springer, 2006.

[69] James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Security and privacy, 2005 IEEE symposium on*, pages 226–241. IEEE, 2005.

[70] Battista Biggio, Ignazio Pillai, Samuel Rota Bulò, Davide Ariu, Marcello Pelillo, and Fabio Roli. Is data clustering in adversarial settings secure? In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, pages 87–98. ACM, 2013.

[71] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning*, pages 97–112, 2011.

[72] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015.

[73] Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Bowei Xi. Adversarial support vector machine learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1059–1067. ACM, 2012.

[74] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647. ACM, 2005.

[75] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *CEAS*, volume 2005, 2005.

[76] Gregory L Wittel and Shyhtsun Felix Wu. On attacking statistical spam filters. In *CEAS*, 2004.

[77] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[78] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108. ACM, 2004.

[79] Pavel Laskov et al. Practical evasion of a learning-based classifier: A case study. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 197–211. IEEE, 2014.

[80] Fei Zhang, Patrick PK Chan, Battista Biggio, Daniel S Yeung, and Fabio Roli. Adversarial feature selection against evasion attacks. *IEEE transactions on cybernetics*, 46(3):766–777, 2016.

[81] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[82] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[83] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018.

[84] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[85] Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pages 353–360. ACM, 2006.

[86] Michael Brückner and Tobias Scheffer. Nash equilibria of static prediction games. In *Advances in neural information processing systems*, pages 171–179, 2009.

[87] Yan Zhou and Murat Kantarcioglu. Modeling adversarial learning as nested stackelberg games. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 350–362. Springer, 2016.

[88] Murat Kantarcıoğlu, Bowei Xi, and Chris Clifton. Classifier evaluation and attribute selection against active adversaries. *Data Mining and Knowledge Discovery*, 22(1-2):291–335, 2011.

[89] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 547–555. ACM, 2011.

[90] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017.

[91] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5764–5772, 2017.

[92] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *International workshop on multiple classifier systems*, pages 350–359. Springer, 2011.

[93] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pages 3517–3529, 2017.

[94] Battista Biggio, Konrad Rieck, Davide Ariu, Christian Wressnegger, Igino Corona, Giorgio Giacinto, and Fabio Roli. Poisoning behavioral malware clustering. In *Proceedings of the 2014 workshop on artificial intelligent and security workshop*, pages 27–36. ACM, 2014.

[95] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006.

[96] KDD Cup. Intrusion detection data set. *The UCI KDD Archive Information and Computer Science University of California, Irvine. DOI= http://kdd. ics. uci. edu/databases/kddcup99*, 1999.

[97] Yahoo! Webscope. Yahoo! Webscope dataset ydata-labeled-time-series-anomalies-v1_0 [http://labs.yahoo.com/Academic_Relations]. `"https://webscope.sandbox.yahoo.com/catalog.php?datatype=s"`, 2017. Online,Accessed: 2017-12-19.

[98] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[99] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 159–166. IEEE, 2015.

[100] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 2018.

[101] Adetunmbi A Olusola, Adeola S Oladele, and Daramola O Abosede. Analysis of kdd'99 intrusion detection dataset for selection of relevance features. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, pages 20–22, 2010.

[102] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[103] Ahmed Ahmim, Mohamed Amine Ferrag, Leandros Maglaras, Makhlouf Derdour, and Helge Janicke. A detailed analysis of using supervised machine learning for intrusion detection.

[104] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[105] Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple classifier systems for robust classifier design in adversarial environments. *International Journal of Machine Learning and Cybernetics*, 1(1-4):27–41, 2010.

[106] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[107] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[108] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[109] Muzammal Naseer, Salman H Khan, Harris Khan, Fahad Shahbaz Khan, and Fatih Porikli. Cross-domain transferability of adversarial perturbations. *arXiv preprint arXiv:1905.11736*, 2019.

[110] Murat Kantarcioglu, Bowei Xi, and Chris Clifton. Classifier evaluation and attribute selection against active adversaries. *Data Mining and Knowledge Discovery*, 22(1-2):291–335, January 2011.

[111] Michael Brückner, Christian Kanzow, and Tobias Scheffer. Static prediction games for adversarial learning problems. *Journal of Machine Learning Research*, 13(Sep):2617–2654, 2012.

VITA

Radhika Bhargava was born and raised in India where she obtained her B. Tech in Information Technology. She was admitted to Purdue University as a Ph.D. student in 2011 and received a M.S. in Computer Science from Purdue University in 2013. During her time at Purdue, she worked for Amazon and Google. Her research interests include adversarial machine learning, anomaly detection, secure multiparty computation, as well as game theory.