Novel Defenses Against Data Poisoning in Adversarial Machine Learning

Sandamal Weerasinghe

Submitted in partial fulfilment of the requirements of the degree of

Doctor of Philosophy

Department of Electrical and Electronic Engineering
THE UNIVERSITY OF MELBOURNE

December 2019

Abstract

MACHINE learning models are increasingly being used for automated decision making in a wide range of domains such as security, finance, and communications. Machine learning algorithms are built upon the assumption that the training data and test data have the same underlying distribution. This assumption fails when (i) data naturally evolves, causing the test data distribution to diverge from the training data distribution, and (ii) malicious adversaries distort the training data (i.e., poisoning attacks), which is the focus of this thesis.

Even though machine learning algorithms are used widely, there is a growing body of literature suggesting that their prediction performance degrades significantly in the presence of maliciously poisoned training data. The performance degradation can mainly be attributed to the fact that most machine learning algorithms are designed to withstand stochastic noise in data, but not malicious distortions. Through malicious distortions, adversaries aim to force the learner to learn a model that differs from the model it would have learned had the training data been pristine. With the models being compromised, any systems that rely on the models for automated decision making would be compromised as well.

This thesis presents novel defenses for machine learning algorithms to avert the effects of poisoning attacks. We investigate the impact of sophisticated poisoning attacks on machine learning algorithms such as Support Vector Machines (SVMs), one-class Support Vector Machines (OCSVMs) and regression models, and introduce new defenses that can be incorporated into these models to achieve more secure decision making. Specifically, two novel approaches are presented to address the problem of learning under adversarial conditions as follows.

The first approach is based on data projections, which compress the data, and we examine the effect of the projections on adversarial perturbations. By projecting the training data to lower-dimensional spaces in selective directions, we aim to minimize the impact of adversarial feature perturbations on the training model. The second approach uses Local Intrinsic Dimensionality (LID), a metric that characterizes the dimension of the local subspace in which data samples lie, to distinguish data samples that may have been perturbed (feature perturbation or label flips). This knowledge is then incorporated into existing learning algorithms in the form of sample weights to reduce the impact of poisoned samples.

In summary, this thesis makes a major contribution to research on adversarial machine learning by (i) investigating the effects of sophisticated attacks on existing machine learning models and (ii) developing novel defenses that increase the attack resistance of existing models. All presented work is supported by theoretical analysis, empirical results, and is based on publications.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Sandamal Weerasinghe, December 2019

Acknowledgements

First and most importantly, I would like to express my sincere gratitude to my three remarkable supervisors Prof. Tansu Alpcan, Prof. Christopher Leckie, and Dr. Sarah M. Erfani, for their continuous support, patience and invaluable constructive criticism. I would also like to thank my advisory committee chair Prof. Marimuthu Palaniswami for his insightful comments and challenging questions, which motivated me to widen my research from various perspectives.

I am truly grateful to the University of Melbourne and the Northrop Grumman Corporation for providing financial support during my candidature. I also want to thank all my lab mates, including Javad, Ifrah, Bigi, Yasmeen, Ahvand, and Amin, for those wonderful moments. It was a great pleasure sharing a laboratory with all of you during the last three and a half years.

Finally, I want to express my deepest gratitude to my family and friends for their love, encouragement, and support. This accomplishment would not have been possible without them.

Preface

The outcomes of this thesis are published or under review for publication in the following journals and conferences. The author mainly did this thesis. However, the author benefited from his supervisors through group meeting sessions in which they provided technical comments and guidance.

Financial support provided by the University of Melbourne, including the Melbourne International Research Scholarship (MIRS) and the Melbourne International Fee Remission Scholarship (MIFRS), is gratefully acknowledged.

No portion of the work presented herein was conducted prior to my enrollment in the degree of Doctor of Philosophy at the University of Melbourne. Neither has any of it been submitted for any other qualification.

Chapter 3

- S. Weerasinghe, S. M. Erfani, T. Alpcan, and C. Leckie, "Support vector machines resilient against training data integrity attacks," *Pattern Recognition*, vol. 96, p. 106985, 2019

- P. S. Weerasinghe, S. M. Erfani, T. Alpcan, C. Leckie, and M. Kuijper, "Unsupervised Adversarial Anomaly Detection Using One-Class Support Vector Machines," in *23rd International Symposium on Mathematical Theory of Networks and Systems*, 2018, pp. 34–37

- S. Weerasinghe, S. M. Erfani, T. Alpcan, C. Leckie, and J. Riddle, "Detection of anomalous communications with SDRs and unsupervised adversarial learning,"

in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 469–472

Chapter 4

- **(under review)** S. Weerasinghe, T. Alpcan, S. M. Erfani, and C. Leckie, "Defending support vector machines against data poisoning attacks," *Submitted to Transactions on Information Forensics and Security*, 2020

Chapter 5

- **(under review)** S. Weerasinghe, S. M. Erfani, T. Alpcan, and C. Leckie, "Defending regression learners against poisoning attacks," *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020

Chapter 6

- S. Weerasinghe, T. Alpcan, S. M. Erfani, C. Leckie, P. Pourbeik, and J. Riddle, "Deep learning based game-theoretical approach to evade jamming attacks," in *International Conference on Decision and Game Theory for Security*. Springer, 2018, pp. 386–397

All the papers produced from this thesis are my original work. In all the publications, my contributions are: literature survey, theoretical analysis, problem formulation, algorithm development, programming in MATLAB/Python, performing experiments, results analysis, and manuscript writing.

The contributions of co-authors Prof. Tansu Alpcan, Dr. Sarah M. Erfani, and Prof. Christopher Leckie are: providing technical comments and guidance, discussions on results analysis, and manuscript proof-reading.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

DUE to the surge in the availability of rich data, machine learning algorithms are increasingly being utilized in a wide range of domains such as communications, smart grid, and security [7, 8]. Unlike in traditional programming, where programmers explicitly define the rules to obtain the corresponding outputs from the inputs, in machine learning, we expect the algorithms to learn the underlying mapping between inputs and outputs from the data itself, without being explicitly programmed. Machine learning algorithms build mathematical models based on the training data, which can subsequently be applied to unseen data samples (i.e., test data) with high accuracy.

The popularity of machine learning in many domains, inevitably, has given rise to the development of sophisticated attacks that undermine learning systems. We already witness primitive forms of adversarial attacks in applications such as spam filtering and virus detection, where attackers make progressive yet uncoordinated attempts to deceive machine learning-based systems. In the next decade, we expect adversarial attacks to increase in terms of complexity and frequency as more and more systems employ machine learning algorithms to support decision making.

Sophisticated attacks put systems that rely on machine learning for automated decision making at risk, with severe consequences. For example, machine learning systems are broadly adopted in power systems applications [9]. One such application is short term demand forecasting, where the power demand for the near future is predicted using past data to guarantee sufficient power supply. If an adversary compromises that forecasting system and forces it to predict a lower power demand than the expected power demand for a particular period, the power supplied during that period would be insuffi-

cient and may cause blackouts. Conversely, if the adversary forces the forecasting system to predict a higher power demand than the expected demand, there would be a surplus power generation that may overload the power distribution systems.

Adversaries may affect the learned models through careful manipulation of the training data or by accessing the learned models and changing their parameters. The latter approach assumes a powerful attacker and requires fundamentally different security solutions. This thesis will examine how attacks of the former type affect learning systems. Most machine learning algorithms are designed under the assumption that the training data and test data come from the same distribution [10]. In practice, however, due to the natural evolution of data, their distributions may diverge and degrade the classification performance of the learned models. Therefore, learning systems need to be retrained periodically to keep up with the drift in data. Unfortunately, it is this periodic retraining that allows adversaries to gradually inject malicious data into the training set that degrades the decision-making capabilities of the models [11].

In practice, such attacks can take place in situations where the attacker has an opportunity to introduce poisoned samples to the training process. For example, this can occur when data is collected using crowd-sourcing marketplaces, where organizations build data sets with the help of individuals whose authenticity cannot be guaranteed. Or when malware examples are collected using honeypots, which mimic likely targets of cyber attacks to lure attackers. In such scenarios, attackers can conceal adversarial samples among the normal data samples, which would later be used by the learners for training.

Adversarial attacks on learning systems through data manipulation can happen during the training phase or the testing phase. In a test time attack, known as an "evasion attack", the adversary manipulates selected data samples such that the predictions of the learning system for them before and after the manipulation would be different. Conversely, in a training time attack, known as a "poisoning attack", the adversary distorts a subset of training data to force the learner to learn an incorrect model with impaired prediction capabilities. The term "poisoning attack" is usually associated with attacks that distort the features of training data. But another form of training data poisoning is label altering attacks where adversaries modify the training labels of a carefully selected

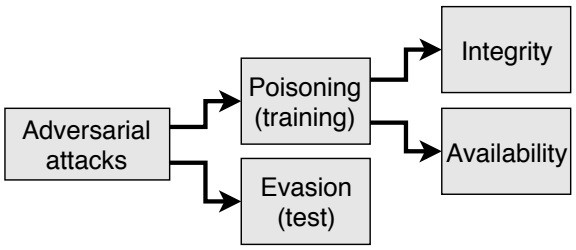


Figure 1.1: Adversarial attack categorization.

subset of data samples [12].

Poisoning attacks can be further divided into two categories based on the attacker's objective as "attacks on integrity" and "attacks on availability". In an integrity attack, the attacker distorts a portion of the attack data points in order to make the learning system learn a compromised decision boundary that exaggerates the region where innocuous data points lie. The compromised boundary would cause the learner to classify specific attack samples as innocuous ones during the evaluation phase. Conversely, in an availability attack, the attacker forces the learning system to classify innocuous samples as attack samples during the evaluation phase, denying them access to the resource protected by the learning system. Figure 1.1 shows a high-level hierarchy of adversarial attacks.

A sophisticated adversary has the capacity to conduct an attack in numerous ways. Due to the uncertainty that arises from the multiple ways an adversary could perform an attack, existing learning algorithms cannot be easily tailored to withstand such attacks [11]. In recent years, there has been an increasing interest in developing defense mechanisms for machine learning algorithms [13, 14, 15]. Although many machine learning algorithms can withstand stochastic noise in data by design, their performance degrades significantly in the presence of malicious adversarial attacks. Therefore it is crucial to develop new defenses that are designed with adversarial attacks in mind and incorporate them into existing machine learning algorithms.

The main aim of this thesis is to assess the effect of poisoning attacks on Support Vector Machines (SVMs) and regression models (i.e., altering features and/or labels) and design defense mechanisms that can be used to increase their attack resistance. Existing

research in the literature attempts to address this problem through the use of data filtering methods [16], augmented cost functions that consider possible adversarial attacks [17, 18], and metrics that differentiate poisoned samples from innocuous samples [19]. However, most defenses against adversarial attacks either rely on assumptions about the attacker or are tightly coupled to specific learning algorithms, thereby limiting their practicality and transferability. This indicates a need for new defenses against adversarial attacks that are transferable across different algorithms while making minimal assumptions regarding the attacker. To that end, we introduce several novel defenses in this thesis as follows.

The first part of this thesis is on a projection based defense against input perturbation attacks against SVMs and one-class SVMs (OCSVMs). While projecting training data to lower-dimensional spaces has been previously used as a way of reducing the computational expense of many algorithms, we demonstrate that projections in carefully selected directions can reduce the effects of input perturbation attacks as well.

The next part of the thesis is concerned with using Local Intrinsic Dimensionality (LID), which is a metric that gives the dimension of the subspace in the local neighborhood of each data sample to distinguish adversarial samples from normal samples [20, 21]. We develop a novel LID estimator that uses kernel matrix values and demonstrate its effectiveness against input perturbation attacks and label flip attacks on SVMs. Subsequently, we extend the LID based defense for regression models by introducing a LID based measure that compares the LID of each data sample with that of its neighborhood. We demonstrate that this defense results in an attack resilient regression estimator.

Finally, we investigate the effectiveness of adversarial distortions in a scenario where a transmitter in a radio environment is attempting to evade an active jammer. The transmitter uses adversarial distortions to support its anti-jamming efforts by deceiving the jammer's learning algorithms. In this work, we used a Long Short-Term Memory (LSTM) model as the learner [22]. We model the interaction between the transmitter and the jammer as a game, and use the game outcomes at every step of the game to decide the actions of the transmitter and jammer.

1.1 Research Questions

In this thesis, we develop novel defense algorithms against poisoning attacks to address the following research questions.

Question 1 - How will the outcomes of security games designed for adversarial learning in cybersecurity applications lead to robust machine learning systems?

- Explore the limitations of an adversary's knowledge about the learning algorithm, algorithm parameters, feature spaces, and training procedure.
- Explore the limitations of a learner's knowledge about the attack strategies.
- Design game theory-based techniques to discover attack strategies for adversaries and defense strategies for learners.

Question 2 - How will data reduction techniques impact the robustness of learning systems under adversarial settings?

- Analyze the advantage gained by the learner due to the unpredictability of using random projections.
- Assess and design metrics that define what a good projection is w.r.t to adversarial scenarios.

Question 3 - How can Local Intrinsic Dimensionality (LID) be used to defend against poisoning attacks?

- Explore new LID based measures that can identify adversarial samples from normal samples in different adversarial settings.
- Analyze ways in which LID can be incorporated into the learning process.

1.2 Research Contributions and Thesis Structure

This thesis aims to contribute to this growing area of research by exploring the impact of adversarial poisoning attacks on machine learning algorithms such as SVMs and regression models, and explores defenses that can be incorporated into such learners in order

to increase their attack resistance. It delivers the following contributions in four main chapters.

Chapter 3 - Projection Based Defense Against Training Data Integrity Attacks (Questions 1 and 2)

In Chapter 3, we investigate the use of dimension reduction through selective projections as a defense mechanism against attacks on integrity.

The main contributions are as follows:

1. We theoretically analyze the effects of adversarial distortions on the separating margins of binary SVMs and One-class SVMs (OCSVM) trained on data that has been nonlinearly projected to a lower-dimensional space.

2. We introduce a unique defense framework that incorporates nonlinear data projections to minimize the adversary's distortions, SVMs for learning, and game theory to predict the behavior of adversaries and take the necessary countermeasures.

3. We pose the problem of finding an appropriate defense mechanism as a game and find the Nash equilibrium solutions that give us insights into what the attacker may do and what precautionary strategy the learner should take.

Chapter 4 - LID Based Defense For SVMs Against Poisoning Attacks (Question 3)

In Chapter 4, we study how the theory of local intrinsic dimensionality (LID) can be incorporated into existing learning algorithms such as SVMs to minimize the effects of label flipping attacks as well as input perturbation attacks.

The main contributions are as follows:

1. We introduce a novel approximation of LID using kernel distances called *K-LID* that calculates the LID values of data samples in the hypothesis space (i.e., a high dimensional transformed space).

2. We then use the likelihood ratio of each data sample (i.e., how many times more likely their K-LID values are from the non-attacked K-LID distribution than the

attacked K-LID distribution) as a distinguishing factor and incorporate this knowledge into the SVM training process.

Chapter 5 - Defending Regression Learners Against Poisoning Attacks (Question 3)

In Chapter 5, we delve into regression learning problems by studying an LID based defense against optimization based poisoning attacks against linear regression models.

The main contributions are as follows:

1. We introduce a novel LID based measure called *Neighborhood LID ratio* that takes into account the LID values of a sample's k nearest neighbors.

2. We then extend the likelihood ratio based defense from Chapter 4, and introduce an N-LID based defense mechanism that makes no assumptions regarding the attack, yet can be used in conjunction with several existing learners such as ridge, LASSO, and neural network regression (NNR) models.

Chapter 6 - Adversarial Time Series Estimation (Question 1)

We extend our work to adversarial regression by investigating the effects of random adversarial perturbations on a time series estimation problem that uses long short-term memory (LSTM) models for predictions in Chapter 6. The chapter focuses on a jamming evasion application that uses a frequency hopping scheme.

The main contributions are as follows:

1. We consider a novel adversarial (deep) learning approach to jamming, where the transmitter and jammer attempt to mislead each other by maliciously distorting data used by the other player to make decisions.

2. We propose the integration of pseudo-random generators and online machine (deep) learning methods in the context of a frequency hopping scheme.

1.3 Publications

The research outcomes of this thesis are published or under review for publication in the following journals and conferences.

- Journals

  - S. Weerasinghe, S. M. Erfani, T. Alpcan, and C. Leckie, "Support vector machines resilient against training data integrity attacks," *Pattern Recognition*, vol. 96, p. 106985, 2019 (Chapter 3).

  - **(under review)** S. Weerasinghe, T. Alpcan, S. M. Erfani, and C. Leckie, "Defending support vector machines against data poisoning attacks," *Submitted to Transactions on Information Forensics and Security*, 2020 (Chapter 4).

  - **(under review)** S. Weerasinghe, S. M. Erfani, T. Alpcan, and C. Leckie, "Defending regression learners against poisoning attacks," *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020 (Chapter 5).

- Conferences

  - P. S. Weerasinghe, S. M. Erfani, T. Alpcan, C. Leckie, and M. Kuijper, "Unsupervised Adversarial Anomaly Detection Using One-Class Support Vector Machines," in *23rd International Symposium on Mathematical Theory of Networks and Systems*, 2018, pp. 34–37. (Chapter 3)

  - S. Weerasinghe, S. M. Erfani, T. Alpcan, C. Leckie, and J. Riddle, "Detection of anomalous communications with SDRs and unsupervised adversarial learning," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 469–472. (Chapter 3)

  - S. Weerasinghe, T. Alpcan, S. M. Erfani, C. Leckie, P. Pourbeik, and J. Riddle, "Deep learning based game-theoretical approach to evade jamming attacks," in *International Conference on Decision and Game Theory for Security*. Springer, 2018, pp. 386–397 (Chapter 6).

Chapter 2
Background and Literature Review

In this chapter, we present the background of our research, review relevant literature, and identify the limitations and gaps in the literature that we address in the following chapters. First, we give a brief introduction to machine learning and explain the main algorithms considered in this thesis. We then review the concept of adversarial learning by looking at the existing attack and defense algorithms. Finally, we also discuss time series estimation and applications of game theory in jamming applications.

2.1 Machine Learning

Machine learning algorithms can be divided in to two main categories based on their learning approaches as follows:

- **Supervised learning -** algorithms use the features of data samples as well as their corresponding labels during the training procedure. From the training data, supervised learning algorithms infer the relationship between the data samples and their respective labels. Supervised learning is mainly used in classification applications and for analyzing sequences of data. Support Vector Machines (SVMs), Artificial Neural Networks (ANN), logistic regression, and decision trees are popular supervised learning algorithms that are widely used for classification problems. Some notable applications of classification are image recognition, speech recognition, and medical diagnosis. Recurrent Neural Networks (RNN) and regression analysis models are used in applications where sequences of data such as time series are predicted.

- **Unsupervised learning -** algorithms take data samples without labels and attempt

to infer hidden patterns that govern the data distribution. Examples of unsupervised learning applications are clustering, outlier detection, and association rule mining [23].

2.1.1 Support Vector Machines

Support Vector Machines (SVMs) are among the most popular supervised learning techniques that are applied for classification tasks. The interpretability of the learned model has contributed to its popularity. Like many learning algorithms, SVMs assume stationary data distributions; that is, both the training data and the testing data are sampled have the same underlying distribution.

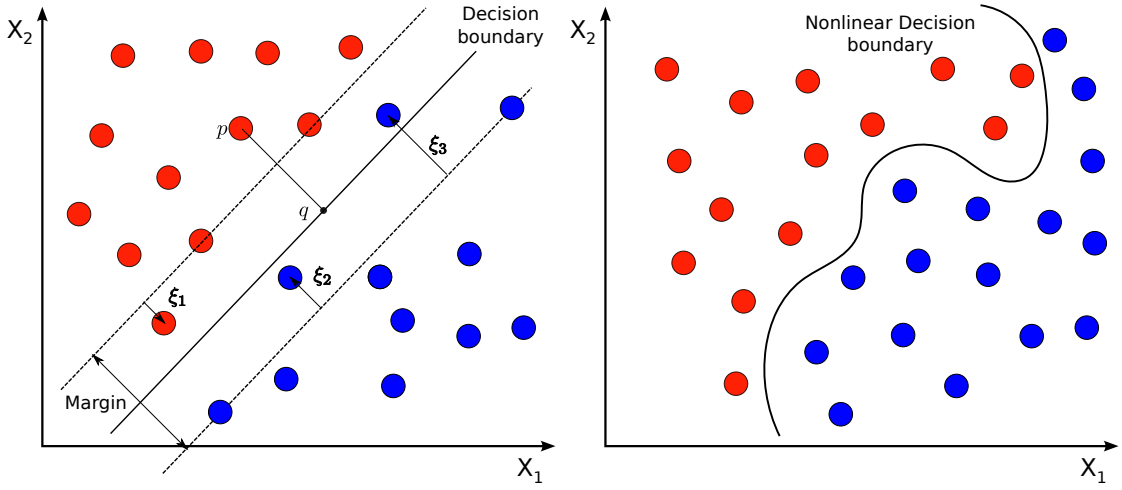
Take $X \in \mathbb{R}^{n \times d}$ and $y_i \in \{\pm 1\}$ for $i \in \{1, \ldots, n\}$ to be the training data. The SVM algorithm finds the hyperplane that separates the data from two classes with the maximum margin in some feature space. The hyperplane is parameterized by the equation $w^T x + b = 0$, where w is the $m \times 1$ normal vector to the hyperplane, and b is the bias (i.e., distance to the hyperplane from the origin).

The decision function of a SVM classifier is given as

$$h_{w,b}(x) = g(w^T x + b). \tag{2.1}$$

A sample x is assigned the label $+1$ if $g(x) \geq 0$ and -1 if $g(x) < 0$. Therefore the predicted label for a sample depends on its location w.r.t the hyperplane in the feature space.

The separating hyperplane of an SVM is significantly affected by the presence of outliers. As shown in Figure 2.1a, the SVM optimization problem is designed to tolerate a certain fraction of outliers. Cortes and Vapnik [24] introduced the possibility to allow certain samples to violate the separating hyperplane by so-called slack variables denoted by ξ_i. To penalize large slack values (i.e., ξ_i), a regularization constant C was introduced to the optimization problem. SVM learning can be formulated as the following convex



(a) A linearly separable dataset. (b) A non-linearly separable dataset.

Figure 2.1: The left figure shows the margin of an SVM on a linearly separable dataset with three outliers present. The figure on the right shows the margin of an SVM on a non-linearly separable dataset.

quadratic programming problem:

$$\begin{aligned} \arg\min_{w,b,\xi_i} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1,\ldots,n \\ & \xi_i \geq 0. \qquad\qquad\quad i = 1,\ldots,n \end{aligned} \tag{2.2}$$

Lagrange duality

The Lagrangian for the primal problem in Equation (2.2) can be constructed as

$$\mathcal{L}(w,b,\xi_i,\alpha,r) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{m}\alpha_i[y_i(w^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^{n} r_i\xi i, \tag{2.3}$$

where α and r are the Lagrange multiplier vectors.

By taking the derivatives of Equation (2.3) w.r.t w, we get

$$\begin{aligned} \nabla_w \mathcal{L}(w, b, \xi_i, \alpha, r) &= w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0, \\ w &= \sum_{i=1}^{n} \alpha_i y_i x_i. \end{aligned} \tag{2.4}$$

Similarly, by taking the derivatives w.r.t b, we get

$$\begin{aligned} \nabla_b \mathcal{L}(w, b, \xi_i, \alpha, r) &= 0 - \sum_{i=1}^{n} \alpha_i y_i - 0 = 0, \\ \sum_{i=1}^{n} \alpha_i y_i &= 0. \end{aligned} \tag{2.5}$$

By taking the derivatives w.r.t ξ_i, we get

$$\nabla_{\xi_i} \mathcal{L}(w, b, \xi_i, \alpha, r) = C - \alpha_i - r_i = 0. \tag{2.6}$$

By substituting the values from Equations (2.4), (2.5) and (2.6) in Equation (2.3), we obtain

$$\mathcal{L}(w, b, \xi_i, \alpha, r) \quad = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j. \tag{2.7}$$

The above objective function with the constraint $0 \leq \alpha_i \leq C$ derived from Equation (2.6) gives the dual optimization problem as follows:

$$\begin{aligned} \arg\max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, i = 1, \ldots, n \\ & \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{2.8}$$

Nonlinear SVMs

The above formulation finds the optimal separating hyperplane for linearly separable datasets (tolerating a few outliers). To allow for nonlinear decision surfaces, the data needs to be nonlinearly transformed to high dimensional spaces using nonlinear func-

tions known as kernels. Once transformed, the data can be linearly separated in the transformed space. Therefore, x_i in Equation (2.8) can be substituted by its high dimensional transformation $\phi(x)$. As transforming training data to a high dimensional space is an expensive operation, for positive definite kernels, we can use the kernel trick to reduce the computational overhead of calculating the nonlinear SVM margin. Kernel functions, usually denoted as $k(x, x') = \langle \phi(x), \phi(x') \rangle$, would lead to a decision function of the following form for a sample x'

$$h(x) = \operatorname{sgn}\Big(\sum_{i=1}^{n} \alpha_i y_i k(x', x_i) + b \Big). \tag{2.9}$$

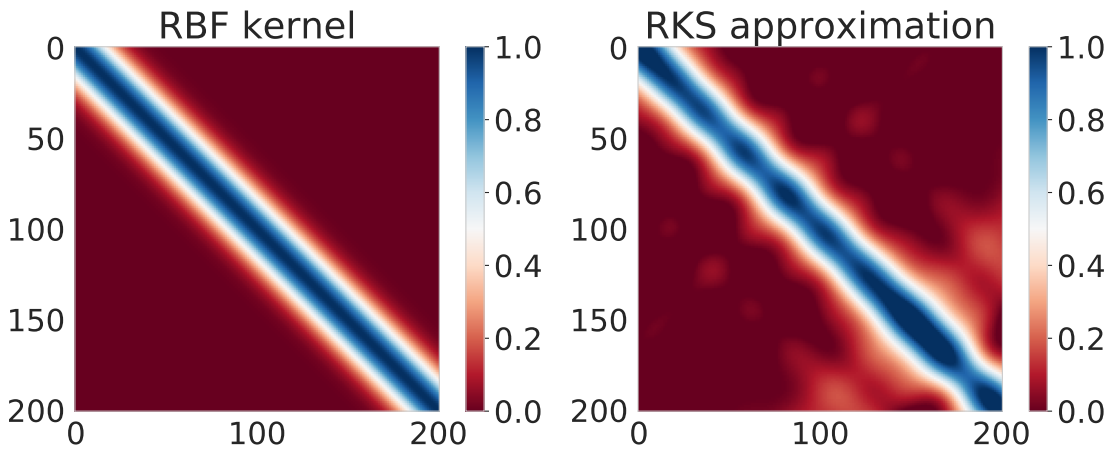
We use SVMs in Chapters 3 and 4 as the classification algorithm we reinforce against adversarial input perturbation and label flip attacks.

Randomized Kernels for SVMs

To improve the efficiency of kernel machines, Rahimi and Recht [25] embedded a random projection into the kernel formulation. They introduced a novel, data-independent method (RKS) that approximates a kernel function by mapping the dataset to a relatively low dimensional randomized feature space. Instead of relying on the implicit transformation provided by the kernel trick, Rahimi and Recht explicitly mapped the data to a low-dimensional Euclidean inner product space using a randomized feature map $z : \mathbb{R}^d \to \mathbb{R}^r$. The kernel value of two data points is then approximated by the dot product between their corresponding points in the transformed space z as

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \approx z(x) z(x'). \tag{2.10}$$

As z is a low dimensional transformation, it is more computationally efficient to transform data with z and train a linear SVM as the result is comparable to that of its corresponding nonlinear SVM. Refer to [25] for more details concerning the theory of kernel approximation. Figure 2.2 shows the actual radial basis function kernel (RBF) and the kernel matrix obtained using the RKS approximation.



(a) RBF kernel matrix.

(b) RKS approximation of the RBF kernel.

Figure 2.2: The RBF kernel matrix of a synthetic dataset and its RKS approximation.

More recently, the method introduced by Rahimi and Recht [25] has been applied to other types of kernel machines. Erfani et al. [26] introduced *Randomized One-class SVMs (R1SVM)*, an unsupervised anomaly detection technique that uses randomized, nonlinear features in conjunction with a linear kernel. The authors reported that R1SVM reduces the training and evaluation times of OCSVMs by up to two orders of magnitude without compromising the accuracy of the predictor. Erfani et al. [27] used a deep belief network (DBN) as a nonlinear dimension reduction algorithm to transform the high-dimensional data into a low-dimensional set of features. Subsequently, a OCSVM with a linear kernel is trained on the feature vectors produced by the DBN. Our work in Chapter 3 differs from these as we look at random projections as a defense mechanism for SVMs under adversarial conditions. To the best of our knowledge, no existing work adopts Rahimi and Recht's method to address adversarial learning for SVMs.

The RKS algorithm is at the core of the projection based defense algorithm we introduce in Chapter 3.

2.1.2 Anomaly Detection with SVMs

Anomaly detection is a subclass of machine learning that revolves around the problem of discovering patterns in data and identifying data points that behave significantly dif-

ferently from the learned patterns, which in some cases are caused by adversaries. These non-conforming data points are referred to as anomalies or outliers. Anomaly detection has numerous applications in a variety of domains, such as network intrusion detection, denial of service detection, credit card fraud detection, and spam filtering. It is a significant problem since the presence of anomalies may indicate malicious attacks that could disrupt mission-critical operations.

During the training process of an anomaly detection system, machine learning algorithms are applied to generate a model of the normal data in the absence of a malicious attack (noise may be present). In the detection process, the input data points are labeled as anomalies if they deviate significantly from the generated model. Subsequently, anomaly detection can detect previously unseen attacks. One Class Support Vector Machines (OCSVM), the outlier detection counterpart of SVMs [28], is useful in scenarios where the training data is unlabeled with a skewed distribution, such that there are many "normal" data samples and only a small percentage of "outliers".

There are two variants of OCSVMs, the algorithm proposed by Schölkopf et al. [29] and the Support Vector Data Description (i.e., SVDD) model proposed by Duin et al. [30]. The former algorithm separates the training data from the origin and attempts to find the separating hyperplane that gives the maximum offset from the origin. The SVDD method finds the smallest sphere that contains the training data. Under translation-invariant kernels, both these methods provide equivalent results. We make use of the OCSVM variant proposed by Schölkopf et al. [29] as the anomaly detection algorithm in Chapter 3.

2.1.3 Distributed SVMs

Distributed training of SVMs has been studied extensively with several variants including cascade SVM, incremental SVM, distributed parallel SVM, and consensus-based SVM used in practice [31]. The objective of these variants is to obtain a decision function that is close to the decision function obtained by a centralized SVM with access to all the data points. The approach used by Alpcan and Bauckhage [32] decomposes the centralized SVM optimization problem into a set of convex sub problems (one per node) with

the nodes having to exchange their respective support vectors (SVs) among each other through a fusion center at each iteration of training. A similar decomposed approach is proposed by Forero et al. [33] for linear SVMs, but instead of exchanging SVs, consensus constraints are added on the classifier parameters.

The latter consensus based DSVM is used in a series of works by Zhang and Zhu [34, 35] that aim to design defense strategies for DSVMs against adversaries. The authors use a game theoretic framework to capture the conflicting interests between the consensus-based DSVM learner and the attacker. Their proposed rejection-based defense strategy prevents updates to the learning models if the primal and dual variables change significantly after training with new data. Their proposed defense model assumes that the initial data it is given is clean and as the primal variables are used in the rejection criteria, this model cannot be used in conjunction with RBF kernels.

The framework by Alpcan and Bauckhage [32] is used in Chapter 4 as the distributed SVM framework for the distributed detection problem we consider.

2.2 Game Theory

Game theory is the study of mathematical models of interaction between intelligent, rational decision-makers, which we usually call players. Each player has his preference among a set of available alternatives. Game theory describes how these preferences change when players make different assumptions about how the other players would act. Once all the players perform their own actions, the game outputs a state. The utility function of each player measures their level of happiness in the given state. When a player is uncertain about which state of the world he faces, his utility is defined as the expected value of his utility function with respect to the appropriate probability distribution over states [36]. In Chapters 3 and 6, we use game theory to model the adversary-learner interaction as a game.

Non-cooperative games contain two or more utility-maximizing (cost-minimizing) players whose actions affect each other's utilities (costs). It should be noted that non-cooperative games are not applied exclusively to situations where the interests of dif-

ferent agents conflict. Adversarial games can be considered as extreme cases of non-cooperative games. In contrast, in cooperative games, the players intend to achieve an objective as a team. These are also known as coalition games. A security game is a non-cooperative two-player game played between a learner and an adversary that provides a base framework for modeling the adversary-learner interaction as a game. The equilibrium solutions obtained from the games can be used for formal decision making as well as to predict the adversary's behavior [37].

Zero-sum games represent situations of pure competition, where one player's gain (loss) is the negative of the other player's loss (gain). On the other side of the spectrum, we have non-zero-sum games, where there is no clear winner and loser. In either case, the players would be able to select their own actions in two ways. If a player has a unique strategy to choose a single action, it is known as a pure strategy. If a player randomizes over the set of available actions according to some probability distribution, it is known as a mixed strategy.

Equilibrium Concepts

- **Nash equilibrium** - The most popular solution concept found in the literature. A Nash equilibrium is a profile of strategy choices for each of the players in a two-player game, such that each player's strategy is the best response to the other player. The best response is a strategy that maximizes a player's payoff (minimizes cost), given the strategies of the other player. When in a Nash equilibrium, no player has an incentive to deviate unilaterally.

- **Min-max and max-min strategies** - In a two-player game, the min-max strategy of player one is the strategy that minimizes its worst-case cost in a situation where the other player plays the strategies that cause the most significant harm to it. The min-max value of the game for player one is the maximum loss (i.e., loss ceiling) guaranteed by a min-max strategy. Similarly, the max-min strategy of player two will guarantee a minimum payoff for it (i.e., gain floor). The strategies of the two players are known as security strategies as they provide a security level for the

players.

- **Stackelberg equilibrium** - In a two-player game where there is a definite order of play, the player that moves first (i.e., leader) can enforce its strategy on the player that follows (i.e., follower). The leader has to take into account all the possible responses of the follower, assuming that it plays rationally, and decide on the first move. If the follower's response to every strategy of the leader is not unique, the game could lead to a state where the leader is at a disadvantage due to the ambiguity of the follower's possible response.
- **Mini-max regret** - Consider a two-player game where, from player one's perspective, player two is not believed to be malicious, but entirely unpredictable. The mini-max regret strategy of player one allows it to minimize its worst-case loss (due to the unpredictability of the other player), rather than merely maximizing their worst-case payoff.

2.3 Adversarial Learning

Adversarial learning is a broad field, which covers a whole range of attacks across different machine learning algorithms [11, 38]. Adversarial attacks on learning systems can be divided into two main categories, poisoning attacks during training and evasion attacks during testing [39]. The mechanism used for generating adversarial samples depends on the objective of the adversary and the knowledge the adversary possesses about the learning system [40, 41, 17, 42]. As suggested by Papernot et al. [43], the objectives of an adversary who attacks a classification system may be as follows.

1. **Confidence reduction** - to reduce the confidence of the classifiers' outputs (increase class ambiguity).
2. **Misclassification** - alter a data point so that the classifier's output is different from that point's original class label.
3. **Targeted misclassification** - produce an artificial data point that the classifier would classify as a specific, predetermined class.

4. **Source/target misclassification** - alter a data point so that the classifier labels it as a specific, predetermined class that is different from the original class label.

2.3.1 Learning Under Adversarial Conditions

The problem of adversarial learning has inspired a wide range of research from the machine learning community, see [44] for a security evaluation of SVMs under adversarial conditions. Poisoning attacks, which alter the training data, can be carried out either by distorting the training data or by distorting the training labels. In label noise attacks, the attackers change the labels of a subset of the training data to increase the SVM's classification error [45, 12, 46].

For classification using binary SVMs, Biggio et al. [40] introduced an attack algorithm that finds the optimal attack point by maximizing the hinge loss of a binary SVM when tested on a validation set. Dalvi et al. [41] modeled classification as a game between the classifier and the adversary. They extend the naive Bayes classifier to optimally detect and reclassify distorted data points, by taking into account the adversary's optimal feature-changing strategy.

For anomaly detection, Kloft and Laskov [47] analyzed the effects of adversarial injections on the centroid anomaly detection (CAD) algorithm, which can be considered as a hard margin, hypersphere-based SVDD model [30] (SVDD is equivalent to OCSVM when the RBF kernel is used). As their work focuses on an online learning setting, they use different data replacement policies as the defense mechanism against integrity attacks. Previously, Rajasegarar et al. [48] used OCSVMs to detect anomalous secondary users that provide misleading observations in cognitive radio networks. They utilized anomaly detection in a scenario where a central node is attempting to determine if a spectrum is being utilized by a primary user or not, with one or many malicious users providing false information to force the central node to make an incorrect decision.

One approach for learning in the presence of poisoned training data is to identify and remove such samples prior to training. Steinhardt et al. [16] introduced a framework that uses an outlier detector prior to training in order to filter out poisoned data. They consider two scenarios, (i) where there is a clean outlier detector (trained independently

without being affected by the poisoned data), and (ii) where the outlier detector is also compromised. While the framework performs well in the first scenario, the authors claim that the attacker is able to subvert the outlier removal and obtain stronger attacks in the second scenario. Laishram and Phoha [49] introduced an algorithm that clusters the data in the input space and utilizes the distances among data points in the same cluster in the (input feature + label) space to identify the outliers. These works can be considered as a pre-processing step and could be used in conjunction with our proposed defenses to increase the attack resistance of SVMs further.

Vinh et al. [50] used multiple linear projections of the training data to train a single neural network. The random projections are used as a regularization mechanism to boost the accuracy of NNs under adversarial conditions. Wong et al. [51] use a nonlinear random projection technique to estimate the upper-bound on the robust loss for DNNs in the worst case that scales linearly in the size of the hidden units. The authors claim that the introduced robust training procedure results in networks with minimal degradation in detection accuracy. Although we use a significantly different nonlinear projection algorithm that is designed specifically for kernel based learners in Chapter 3, our defense framework yields similar benefits in terms of minimal accuracy degradation, and lower training times for SVMs as the authors claim their technique [51] does for DNNs.

2.3.2 SVMs in the Presence of Label Noise

In recent years, there has been an increasing body of literature on the problem of classification in the presence of label noise, see the work of Frénay and Verleysen [52] for a survey. Several prior works have shown that mislabeled instances significantly impact SVMs, and removing such samples reduces the complexity of SVMs [53]. In the works of Brodley and Friedl [54] and Libralon et al. [55], the authors show that removing flipped samples from training data results in SVMs with simpler decision boundaries.

Zhang and Yang [56] show that the performance of linear SVMs degrades significantly at merely 5% of flipped labels. Risk minimization under a particular loss function is said to be *label noise-robust* if the probability of misclassification of inferred models with label noise is identical to the same probability without label noise. Manwani and Sastry [57]

show that hinge loss is not robust to uniform label noise. Consequently, it leads to the conclusion that SVMs are not robust to uniform label noise.

2.3.3 Defense Mechanisms for SVMs

In order to embed a mechanism to consider possible label alterations into SVMs, Lin et al. [58] and An and Liang [59] use fuzzy memberships for each training data point to force data points to influence differently when calculating the separating hyperplane. In the work of Natarajan et al. [60], the authors suggest two approaches to modify loss functions to withstand random label noise during training. Liu and Tao [61] use traditional loss functions for classification tasks in the presence of random label noise by using importance reweighting.

Suykens and Vandewalle [18] introduce Least-Squares SVM (LS-SVM), where a quadratic loss function is used instead of the hinge loss, which results in a non-sparse solution to the optimization problem (all the training samples are assigned non-zero α values). The authors claim this approach prevents the SVM from over-relying on the contribution of specific samples. Label Noise Robust SVM (LN-SVM), proposed by Biggio et al. [45], assumes that the label of each training sample can be independently flipped with the same probability of μ. The probability of label flips is then incorporated into the kernel matrix. With this approach, each training sample is more likely to become a support vector.

Zhou et al. [17] introduced an Adversarial SVM (AD-SVM) model, which incorporated additional constraint conditions to the binary SVM optimization problem to thwart an adversary's attacks. Their model only supports data that is linearly separable and leads to unsatisfactory results when the severity of real attacks differs from the expected attack severity by the model. In Chapter 4, we focus on both types of training time attacks (i.e., label flipping and input perturbation) against SVMs and introduce one defense algorithm that can withstand both types of attacks.

2.4 Regression Analysis

2.4.1 Robust regression

In robust statistics, robust regression is well studied to devise estimators that are not strongly affected by the presence of noise and outliers [62]. Huber [63] uses a modified loss function that optimizes the squared loss for relatively small errors and absolute loss for relatively large ones. As the influence on absolute loss by outliers is less compared to squared loss, this reduces their effect on the optimization problem, thereby resulting in a less distorted estimator. The TheilSen estimator, developed by Thiel [64] and Sen [65] uses the median of pairwise slopes as an estimator of the slope parameter of the regression model. As the median is a robust measure that is not influenced by outliers, the resulting regression model is considered to be robust as well.

Fischler and Bolles [66] introduced random sample consensus (RANSAC), which iteratively trains an estimator using a randomly sampled subset of the training data. At each iteration, the algorithm identifies data samples that do not fit the trained model by a predefined threshold as outliers. The remaining data samples (i.e., inliers) are considered as part of the consensus set. The process is repeated a fixed number of times, replacing the currently accepted model if a refined model with a larger consensus set is found.

While these methods provide robustness guarantees against noise and outliers, in Chapter 5, we focus on malicious perturbations by sophisticated attackers that craft adversarial samples that are similar to normal data. Moreover, the aforementioned classical robust regression methods may fail in high dimensional settings as outliers might not be distinctive in that space due to high-dimensional noise [67].

2.4.2 Adversarial regression

The problem of learning under adversarial conditions has inspired a wide range of research from the machine learning community, see the work of Liu et al. [68] for a survey. Although adversarial learning in classification and anomaly detection has received a lot of attention [69, 14, 1, 3], adversarial regression remains relatively unexplored.

Most works in adversarial regression provide performance guarantees given several assumptions regarding the training data and noise distribution hold. The resilient algorithm by Chen et al. [70] assumes that the training data before adversarial perturbations, as well as the additive noise, are sub-Gaussian and that training features are independent. Feng et al. [71] provides a resilient algorithm for logistic regression under similar assumptions.

Xiao et al. [72] examined how the parameters of the hyperplane in a linear regression model change under adversarial perturbations. The authors also introduced a novel threat model against linear regression models. Liu et al. [73] use robust PCA to transform the training data to a lower-dimensional subspace and follow an iterative trimmed optimization procedure where only the n samples with the lowest residuals are used to train the model. Note that n here is the number of normal samples in the training data set. Jagielski et al. [74] use a similar approach for the algorithm "TRIM", where trimmed optimization is performed in the input space itself. Therefore they do not assume a low-rank feature matrix as done in [73]. Both approaches, however, assume that the learner is aware of n (or at least an upper bound for n), thereby reducing the practicality of the algorithms.

Tong et al. [75] consider an attacker that perturbs data samples during the test phase to induce incorrect predictions. The authors use a single attacker, multiple learner framework modeled as a multi-learner Stackelberg game. Alfeld et al. [76] propose an optimal attack algorithm against auto-regressive models for time series forecasting. In our work, we consider poisoning attacks against models that are used for interpolating instead of extrapolating. Nelson et al. [77] originally introduced RONI (Reject On Negative Impact) as a defense for adversarial attacks against classification models. In RONI, the impact of each data sample on training is measured, and samples with notable negative impacts are excluded from the training process.

To the best of our knowledge, no linear regression algorithm has been proposed that reduces the effects of adversarial perturbations in training data through the use of LID, as we do in Chapter 5. Most existing works on adversarial regression make strong assumptions regarding the attacker/attack, making them impractical, whereas, we propose a

novel defense that makes no such assumptions.

2.5 Jamming Attacks and Game Theory

The problem of jamming in wireless networks has inspired a wide range of research from the community, refer to Vadlamani et al. [78] for a recent survey on jamming attacks in wireless sensor networks. We focus on prior work that utilizes game theory for analyzing jamming problems. For example, Dabcevic et al. [79] use a game-theoretical approach to analyze jamming/anti-jamming behavior between cognitive radio systems. The authors formulate the jamming problem as a fictitious play between the jammer and the transmitter. Similar to most work in the literature that incorporates game theory to jamming problems, they consider channel hopping and changing power allocation (increasing the robustness of the signal) as anti-jamming strategies. Zhu et al. [80] formulated a non-cooperative game to model the interaction between wireless users and a malicious node that can act as a jammer and an eavesdropper. The authors also utilize a fictitious play-based algorithm to find mixed strategy Nash equilibrium solutions.

Spatial retreats are another form of anti-jamming strategy discussed by Xu et al. [81], where the participants, who are capable of moving, attempt to move out of a stationary jammer's range. Pietro and Oligeri [82] considered using directional antennas to minimize the interference coming from directions not involving the actual positions of the participants as a physical layer anti-jamming solution. Bhattacharya and Basar [83] and Zhu et al. [84], investigate jamming attacks by mobile attackers on mobile targets by modeling their interaction as zero-sum pursuit-evasion games.

Fixed jamming strategies are ineffective in cognitive radio networks (CRNs), where the transmitters and receivers are capable of changing their communication parameters such as frequency, band, and protocol on the fly. In CRNs, jammers are also likely to be equipped with cognitive radio technology, allowing them to change their attack strategies in response to the secondary users of the network [82, 85]. Wang et al. [85] modeled jamming attacks as a non-cooperative, zero-sum, stochastic game. At each stage of the game, the secondary users of the network observe the spectrum availability, the channel

quality and the jammer's strategy, and decide on their actions.

While machine learning techniques have been extensively used for intrusion detection, their usage in jamming applications is relatively recent. For example, Puñal et al. [86] use supervised learning to detect jamming attacks on IEEE 802.11 networks. The authors train a random forest classification model based on metrics collected from simulations and attempt to predict if a network is under a jamming attack.

In Chapter 6, we use stochastic channel hopping as the jamming evasion strategy and incorporate a state of the art time series prediction algorithm into the decision making process of the jammer. To the best of our knowledge, no existing work has explored this unique combination.

2.5.1 Chaotic Time Series Prediction

Cong and Songgeng [87] explored the use of chaotic systems to generate frequency hopping sequences. More recently, Hu et al. [88] proposed a pseudo-random sequence generator based on the Chen chaotic system. Short-term chaotic time series prediction is a well-researched problem [89, 90]. To make multi-step predictions on chaotic time series, Han et al. [91] proposed a variant of RNNs called *recurrent predictor neural network*, which attempts to reduce the number of redundant connections between nodes before training. More recently, particular interest has been put into predicting chaotic time series using Long Short-Term Memory networks (LSTMs). A knowledge-based prediction model combined with a machine learning-based prediction model is utilized by Pathak et al. [92] to predict chaotic systems. The hybrid method can make better predictions for multiple steps compared to each prediction model taken separately.

Chapter 3

Projection Based Defense Against Training Data Integrity Attacks

We first study a projection based defense for SVMs against adversarial perturbations. This chapter addresses the research questions 1 and 2 we introduced in Chapter 1.

SVMs are vulnerable to integrity attacks, where malicious attackers distort the training data in order to compromise the decision boundary of the learned model. With increasing real-world applications of SVMs, malicious data that is classified as innocuous may have harmful consequences. In this chapter, we present a novel framework that utilizes adversarial learning, non-linear data projections, and game theory to improve the resilience of SVMs against such training-data-integrity attacks. The proposed approach introduces a layer of uncertainty through the use of random projections on top of the learners, making it challenging for the adversary to guess the specific configurations of the learners. To find appropriate projection directions, we introduce novel indices that ensure the contraction of the data and maximize the detection accuracy.

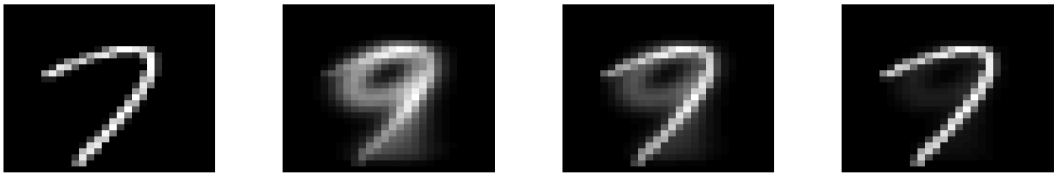
3.1 Introduction

SINCE their introduction, Support Vector Machines (SVMs) [24] have been successfully applied to a wide range of domains such as intrusion detection, image recognition, and bioinformatics [93]. Binary SVMs are primarily used for classification problems where the learner has access to labeled training data with balanced classes. In many SVM implementations, class weights are used to compensate for imbalanced classes. In contrast, its unsupervised counterpart, One-Class Support Vector Machines (OCSVMs) [28], are used for anomaly detection problems where the learner does not have access to training labels and the classes are significantly imbalanced. Binary SVMs and OCSVMs are

designed to withstand the effects of random noise in data [94]. However, their performance may degrade significantly when malicious adversaries deliberately alter the training data. It has become imperative to secure machine learning systems against such adversaries due to increased automation in many day to day applications. For example, if machine learning algorithms utilized in safety-critical environments (e.g., airports, power plants) are compromised by adversaries, it could result in loss of human lives.

Adversarial learning [11] is a broad field, which covers a whole range of attacks across different machine learning algorithms. Adversarial attacks on learning systems can be divided into two main categories, poisoning attacks during training and evasion attacks during testing [39]. In many applications, learned models are periodically updated using new batches of data in order to adapt to the natural evolution of data. This periodic updating provides an opportunity for adversaries to inject malicious data into the training process and carry out poisoning attacks. By injecting adversarial samples into the training dataset, the attackers aim to make the learner learn a decision boundary that is significantly different from the boundary it would have obtained if the dataset was not compromised. The mechanism used for generating adversarial samples depends on the objective of the adversary and the knowledge the adversary possesses about the learning system [40, 41, 17, 42].

Poisoning attacks can be further divided into two categories based on the attacker's objective: attacks on integrity and attacks on availability. Although the attacker's objectives are different, these attack types are closely related in terms of how the attacks are carried out. In an integrity attack, the attacker purposely distorts a portion of the attack/anomaly data points that are used for training. Such integrity attacks result in a compromised decision boundary that exaggerates the region where innocuous data points lie. Subsequently, during the evaluation phase, the learner classifies specific attack data points as innocuous ones, which can cause significant harm in mission-critical applications. In an availability attack, the attacker's objective is to force the learned model to classify innocuous samples as attack/anomaly samples during testing, denying them access to the resource protected by the learning system. This chapter focuses specifically on SVMs and OCSVMs and addresses the following key question: "Is it possible to make



(a) Anomaly (b) 0.2 (c) 0.5 (d) 0.8

Figure 3.1: A digit from anomaly class ('7') distorted by the adversary using different s_{attack} values to appear like a digit from the normal class ('9').

SVMs more resistant to poisoning attacks on integrity?".

Consider the illustrative example of digit '9' as the normal class and digit '7' as the anomaly class in an image anomaly detection setting (Figure 3.1). Assume a hypothetical anomaly detection algorithm that attempts to identify the smallest hypersphere that contains the images of digit '9'. The objective of the adversary in such a situation would be to maximize the radius of the minimum enclosing hypersphere. The adversary can achieve this by injecting data points (i.e., images) of digit '7' that are distorted to appear as digit '9' into the training set. Consider a parameter $s_{attack} \in [0, 1]$ that controls the severity of the attack. An image of digit '7' that closely resembles a digit '9' (small s_{attack}) would be considered as a *moderate attack*, whereas, digit '7' that actually resembles a '7' (large s_{attack}) would be considered a *severe attack*. As Figure 3.1 shows, after a less severe attack (e.g., 0.2), a digit '7' resembles a '9' visually, but as the attack severity increases, the digit tends to look like a '7' even though the learner considers it as a '9'. In practice, such attacks can be carried out in scenarios where the attacker has the opportunity to introduce distorted samples to the training process. For example, this can occur when data is collected using crowdsourcing marketplaces, where organizations build data sets with the help of a large group of people, or when malware examples are collected using honeypots which mimic likely targets of cyberattacks to lure attackers. In such scenarios, attackers can place adversarial samples among the normal data samples which would later be used by the learners for training.

Among adversarial defense techniques for SVMs, most works in the literature alter the optimization problem of SVMs in order to thwart an adversary's attacks [17, 41]. Meanwhile, recent works in the literature use nonlinear random projections to improve

the training and evaluation times of kernel machines, without significantly compromising the accuracy of the trained models [25, 26]. In this chapter, we show that under adversarial conditions, selective nonlinear projections can be leveraged as a defense technique for learners (SVMs/OCSVMs) as well. The learners gain an additional advantage due to the uncertainty that comes from the randomness of the projections. To the best of our knowledge, no existing work has explored the use of nonlinear random projections for adversarial defense.

The learner leverages the theory of low rank kernel approximation (using nonlinear projections) which facilitates large-scale, data-oriented decision making by reducing the number of optimization parameters and variables. As not all random projections result in low rank representations that mask the adversarial distortions, we introduce novel indices to identify prospective projection directions that could provide resistance against adversaries. In addition, we design security games [37] and model the adversary-learner interaction as non-cooperative, two-player, nonzero-sum games with the strategies and utility functions formulated around a nonlinear data-projection-based algorithm. The equilibrium solutions obtained from the games are used to predict the adversary's behavior and decide on advantageous configurations for the learner [95]. The main **contributions** of this work are summarized as follows:

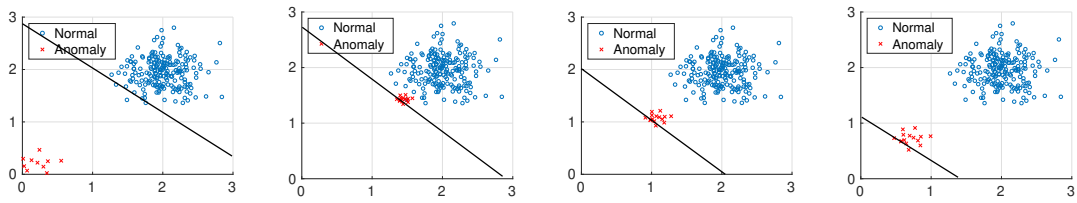
1. We theoretically analyze the effects of adversarial distortions on the separating margins of binary SVMs and OCSVMs trained on data that has been **nonlinearly projected** to a lower dimensional space. We provide an upper-bound for the length of the weight vector when there are no adversarial distortions, using the length of the weight vector when adversarial distortions are present. We prove a similar bound for OCSVMs under **linear projections** when the data is assumed to be linearly separable.
2. We introduce a unique framework that incorporates nonlinear data projections to minimize the adversary's attempts to mask their activities, SVMs for learning, and game theory to predict the behavior of adversaries and take the necessary countermeasures. As part of this framework, we introduce novel indices based on the Dunn's index [96] to identify suitable directions for nonlinear data projections.

3. We pose the problem of finding an appropriate defense mechanism as a game, and find the Nash equilibrium solutions that give us insights into what the attacker may do and what precautionary strategy the learner should take.
4. We show through numerical experiments conducted with benchmark data sets as well as OMNeT++ simulations that our proposed approach can (i) increase the attack resistance of OCSVMs (up to 13.5%) and SVMs (up to 14.1%) under adversarial conditions, and (ii) give the learner a significant advantage from a security perspective by adding a layer of unpredictability through the randomness of the data projection, making it very difficult for the adversary to guess the projected space used by the learner.

3.2 Problem Statement: Adversarial Learning Against Integrity Attacks

We consider an adversarial learning problem in the presence of a malicious adversary. The adversary's ultimate goal is to force the learner to accept a specially crafted adversarial point as innocuous after some learning iteration. To clarify, if we consider the innocuous class as the negative class, and the attack class as the positive class, the attacker would want to increase the false negative rate (FNR). To succeed in this, the attacker injects malicious training data in order to alter the decision boundary of the learner in a manner favorable to him/her. Subsequently, during the testing phase, it would be easier for the attacker to craft adversarial data points that still retain their harmful qualities, but are classified by the learner as innocuous. In the following section, we formalize the underlying problem using the attack strategy of the adversary.

Adversarial Attack Model: In the context of OCSVMs, the decision boundary (i.e., the separating hyperplane) is located closer to the normal data cloud and the undistorted anomalies lie close to the origin. The adversary would distort anomalies in order to shift them closer to the normal data cloud. Since the OCSVM algorithm considers all the data points in the training set to be from a single class, these distorted anomalies would shift the separating hyperplane in the direction of the attack points (towards the origin). In



(a) no attack. (b) $s_{attack} = 0.3$. (c) $s_{attack} = 0.5$. (d) $s_{attack} = 0.7$.

Figure 3.2: Training data distribution and separating hyperplane (black line) of a toy problem under different attack severities. 'o' (blue) denotes the undistorted data points and 'x' (red) denotes the data points distorted by the adversary. The OCSVM is trained considering the entire (unlabeled) dataset is from one class.

binary SVMs, the decision boundary lies between the data clouds of the two classes. Therefore, when the adversary distorts attack data and shift them closer to the normal data (with the labels flipped), the decision boundary would shift towards the attack data class.

The adversary is able to orchestrate different attacks by changing the percentage of distorted attack data points in the training dataset (i.e., p_{attack}) in addition to the severity of the distortion (i.e., s_{attack}). Figure 3.2 illustrates the data distributions when different levels of attack severities are applied to the anomaly data in a OCSVM problem. When there is no attack (i.e., 3.2a), the undistorted anomalies lie closer to the origin and the OCSVM disregards their contribution to the decision boundary by considering them as outliers. In the presence of an attack (e.g., 3.2c), the distorted anomalies are positioned much closer to the normal data cloud, and the OCSVM's decision boundary is influenced by their contribution. As s_{attack} increases, the anomaly data points are moved closer to the origin, reducing the gap between the origin and the separating hyperplane.

Let $X \in \mathbb{R}^{n \times d}$ be the training dataset and $T \in \mathbb{R}^{n \times d}$ be the distortions made by the adversary, making $X + T$ the training dataset that has been distorted (if the i^{th} data point is not distorted, T_i is a vector of zeros). It should be noted that the learner cannot demarcate T from $(X + T)$, otherwise the learner would be able to remove the adversarial distortions during training, making the problem trivial. The adversary has the freedom to determine T based on the knowledge it possesses regarding the learning system, although the magnitude of T is usually bounded due to its limited knowledge about the

learners' configuration, the increased risk of being discovered, and computational constraints.

The attack model in this work is inspired by the restrained attack model described in [17], where it is assumed that the adversary has the capability to move the i^{th} data point in any direction by adding a non-zero displacement vector $\kappa_i \in T$ to $x_i \in X$. It is also assumed that the adversary does not have any knowledge about the projection used by the learner. Therefore, all of the adversary's actions take place in the input space. The adversary picks a target x_i^t for each x_i to be distorted and moves it towards the target by some amount. Choosing x_i^t for each x_i optimally requires a significant level of computational effort and a thorough knowledge about the distribution of the data.

We assume that the attacker knows the distribution of the normal data used by the learner. While this does exaggerate the adversary's capabilities, in real-world applications there is a possibility to approximate this distribution based on domain knowledge or prior experience. For example, an attacker posing as a data supplier in a crowdsourcing marketplace could use the data samples of legitimate suppliers to approximate the distribution of normal data. This approach also tests our proposed defense framework in a worst-case scenario. Please note that the theoretical analyses we present in Section 3.4 do not depend on this assumption. In the analyses, the adversarial distortions added to the data can come from any poisoning attack algorithm under which Assumption 3.1 holds.

The attacker, similar to [17], uses the centroid of the normal data cloud in the training set as the target point for all anomaly data points that it intends to distort. A data point sampled from the normal data cloud or an artificial data point generated from the estimated normal data distribution could be used as alternatives. For each feature j in the input feature space, the adversary is able to add κ_{ij} to x_{ij} as follows where $s_{attack} \in [0, 1]$ controls the severity of the attack,

$$\kappa_{ij} = (1 - s_{attack})(x_{ij}^t - x_{ij}) \text{ and } |\kappa_{ij}| \leq |x_{ij}^t - x_{ij}|, \forall j \in d. \tag{3.1}$$

3.3 Defense Framework Against Integrity Attacks

Our novel defense framework consists of three main components: (1) a selective randomized projection using a novel metric that increases attack resistance by masking the adversary's distortions, (2) SVMs for learning and predicting, and (3) a game-theoretic model that supports defensive decision-making by considering the best responses of the players (Figure 3.3).

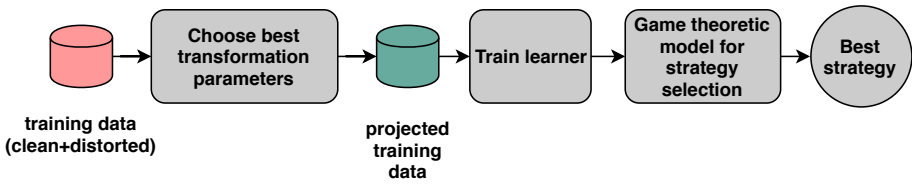
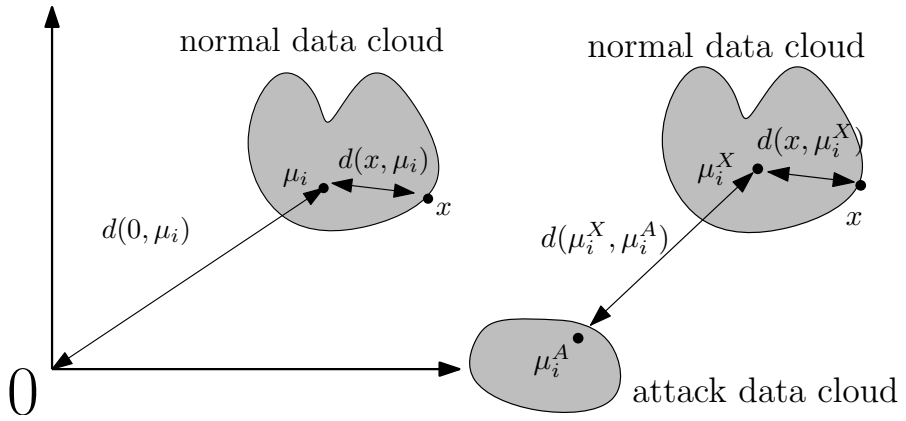


Figure 3.3: Flowchart of our proposed defense framework.

In order to increase the attack resistance of a learning system, the impact of adversarial inputs should be minimized. Therefore, at the heart of our framework we use a projection mechanism that projects data points to lower dimensional spaces in a manner that conceals the potential distortions of an adversary. Projecting a high dimensional dataset, using a carefully chosen projection matrix would preserve its pairwise Euclidean distances with high probability in the projected space [97]. Therefore, the properties of the original data distribution would be present in the projected dataset with only minor perturbations. Assuming the existence of a lower dimensional intrinsic subspace, the learner projects the data to a lower dimensional space using a projection matrix $A \in \mathbb{R}^{d \times r}$, i.e., $(X + T)A$. In this work, the learner trains a linear SVM in a low dimensional space where the data has been nonlinearly transformed using the algorithm introduced by Rahimi and Recht [25], instead of training a nonlinear SVM with a RBF kernel. Therefore, this direction is drawn from the Fourier transform of the shift invariant kernel being approximated. For the RBF kernel, A is sampled from $\mathcal{N}(0, 1)$.

By randomly drawing projection directions from some distribution, the learner also introduces a layer of uncertainty to the adversary-learner problem. For high dimensional datasets, this method gives the learner considerable flexibility to select the dimension to which the data is projected, as well as the direction. Therefore the learner gains a



(a) OCSVM. (b) Binary SVM.

Figure 3.4: Graphical illustrations of the compactness indices for OCSVMs and binary SVMs. The normal data cloud is shown in blue while the attack data cloud is shown in red.

significant advantage from a security perspective as this expands the search space for the adversary. But this unpredictability can also be seen as the main caveat of using random projections to reduce the dimensionality of data. While some random projections result in better separated volumetric clouds than the original ones, some projections result in the data from different classes being overlapped. As the learner cannot demarcate T from the training data, it is not possible to identify an ideal projection that conceals the adversarial distortions.

Thus, in a OCSVM problem, the learner would have to select a projection that contracts the entire training set (expecting the adversarial points to be masked by normal data) and separates the training data from the origin with the largest margin in the projected space. Therefore, motivated by a generalized version of the Dunn's index [96], we propose a compactness measure to rank suitable projection directions in a one-class problem. The learner draws multiple samples from $\mathcal{N}(0, 1)$ for the projection matrix A and ranks them using Equation 3.2. The projection direction A that gives the highest compactness value is considered as the projection that gives the best attack resistance as it gives a compact normal data cloud that is well separated from the origin. The compactness of projection P_i, where μ_i is the centroid of the projected training set, 0 is the origin

in the projected space, and the function d is the Euclidean distance, can be calculated as

$$c_i^1 = \frac{d(0, \mu_i)}{\left(\sum_{x \in P_i} d(x, \mu_i)\right)/n}. \tag{3.2}$$

In a binary SVM problem, the learner would have to select the projection that contracts the innocuous data and separates it from the attack data with the largest distance. Therefore, we devise a similar compactness measure where μ_i^X is the centroid of the projected innocuous data (contains the adversarial distortions as well) and μ_i^A is the the centroid of the projected attack data.

$$c_i^2 = \frac{d(\mu_i^X, \mu_i^A)}{\left(\sum_{x \in P_i} d(x, \mu_i^X)\right)/n}. \tag{3.3}$$

Following the linear projection, each i^{th} sample $(X+T)_i A$ is then nonlinearly transformed using the function

$$z((X+T)_i) = \frac{\sqrt{2}}{r} \cos\left(\sqrt{2\gamma}(X+T)_i A + b\right), \tag{3.4}$$

where γ is a parameter taken from the RBF kernel being approximated, r is the dimension to which the data is projected, d is the input space dimension and b is an r-dimensional vector whose elements are drawn uniformly from $[0, 2\pi]$ [25]. This transformation projects the data onto the interval $[0, 1]$. The approach used by the learner to identify suitable projection directions in a OCSVM problem is formalized in Algorithm 1 in terms of the random projection parameters A and b, the dimension of the projected dataset r and the adversary's data distortion strategy T.

The next component in our framework is the learning algorithm. Anomaly detection problems are addressed in this chapter using the OCSVM algorithm in [29], which separates the training data from the origin with a maximal margin in the feature space. For classification problems, the binary SVM algorithm introduced in [24] is used.

Algorithm 1 Identifying compact projections.

input $(X + T)$, r, number of samples N
$A^{'}, b^{'} \leftarrow null$ ▷ projection parameters
for $i \leftarrow 1, N$ **do** ▷ sample N random directions
  Draw A from $\mathcal{N}(0, , 1)$ ▷ sample A
  $c_i^1 \leftarrow calculate\_compactness((X + T)A)$ ▷ calculate compactness. (Eq. 3.2)
  if $c_i^1 > max\_compactness$ **then**
    $max\_compactness \leftarrow c_i^1$
    $A^{'} \leftarrow A$
  end if
end for
$[(X + T)^*, b'] \leftarrow z(X + T, A^{'})$ ▷ nonlinear projection (Eq. 3.4)
output $A^{'}, b^{'}$ ▷ Return best projection parameters

3.3.1 Game Theoretic Models to Identify Best Defense Strategies

In the final component of our framework, we pose the adversary-learner interaction as a bimatrix game. Using the formulated game, the learner can (i) predict the possible actions of the adversary, and (ii) decide what countermeasure to take in order to thwart the adversary's attempts. In this section, we present a game formulation that can be employed in adversarial conditions for anomaly detection. In Section 3.5.1, we demonstrate how the following game can be utilized in a real world application scenario.

In the following formulation we consider the adversary (M) and learner (L) to be the two players. The adversary is unaware of the learner's configuration and projections used, but it is capable of evaluating the learned model by sending adversarial samples during testing. Similarly, the learner is unaware of the details of the adversary's attack, but is able to simulate attacks during the training process. Since the adversary can vary the severity of the attacks, we choose different s_{attack} values (keeping p_{attack} constant) as the finite set of actions (a_M) available for the adversary. As the learner uses the projection based method to detect adversarial samples, the dimensions to which the data is projected will be used as the set of actions (a_L) available to the learner.

$$\begin{aligned} a_M &\in \{0, 0.3, 0.4, 0.5, 0.6\}, \\ a_L &\in \{20\%, 40\%, 60\%, 80\%, 100\%\}. \end{aligned} \tag{3.5}$$

A bimatrix game comprises two matrices, $G = \{g_{i,j}\}$ and $H = \{h_{i,j}\}$ where each pair

of entries $(g_{i,j}, h_{i,j})$ denotes the outcome of the game corresponding to a particular pair of decisions made by the players. These entries in the matrix are populated by the players' (adversary and learner) utility functions, $u_M : a_M, a_L \rightarrow \mathbb{R}$ and $u_L : a_M, a_L \rightarrow \mathbb{R}$. A pair of strategies $(g_{i^*,j^*}, h_{i^*,j^*})$ is said to be a non-cooperative Nash equilibrium outcome of the bimatrix game if there is no incentive for any unilateral deviation by any one of the players. While it is possible to have a scenario where there is no Nash equilibrium solution in pure strategies, there would always be a Nash equilibrium solution in mixed strategies [98].

Due to the adversary's ability to evaluate the model during testing (i.e., calculating the false negative rate (FNR)), we design u_M to reflect his/her desire to achieve false negatives and to penalize large adversarial distortions. This is because if the adversary greedily distorts the data, it would result in the distortions becoming quite evident and increase the risk of the attack being discovered. Similarly, the learner's utility function reflects a desire to achieve high classification accuracies, which is captured by the f-score. Note that an affine linear transformation of either of the utility functions would result in a strategically equivalent bimatrix game. All strategically equivalent games have the same Nash equilibria as shown by Basar and Olsder in Proposition 3.1 of [99]. The utility functions of the two players are, therefore, defined as

$$\begin{aligned} u_M(a_M, a_L) &= 1 + FNR - \frac{1}{2} s_{attack}, \\ u_L(a_M, a_L) &= f\text{-}score. \end{aligned} \tag{3.6}$$

3.4 Analysing the Impact of Adversarial Distortions on the Separation Margin

This section analyzes the effects of the adversary's distortions on the separation margin of OCSVMs and SVMs when the data is projected to low dimensional spaces. We investigate nonlinear projections as well as linear projections. Although the projection direction is randomized, the following analyses are conditional on the direction chosen by the defense framework. The margin of OCSVMs and SVMs is largely dependent on

the regularization parameter ν, therefore in the following analyses we take ν to be fixed to a value chosen prior to learning.

3.4.1 Nonlinear Projections

Attack Effectiveness on the OCSVM Margin

Let w_p^* be the primal solution of the OCSVM optimization problem in the projected space without adversarial distortions. Similarly, define w_{pt}^* as the primal solution in the presence of a malicious adversary. Since the learner cannot demarcate the distortions from the normal training data, it cannot empirically calculate $\|w_p^*\|_2$. Therefore, based on the assumptions given below, we analytically derive an upper-bound on $\|w_p^*\|_2$ of a OCSVM that has been trained on a nonlinearly projected undistorted dataset.

As the adversary distorts data in the input feature space, we can align any given dataset in such a way that any outliers present in the data would lie closer to the origin and the normal data would lie in the positive orthant. Such a projection would compel the adversary to make adversarial distortions in the direction of the normal data cloud (positive) as distortions in the negative direction would favor the learner (Figure 3.2).

Definition 3.1. *Let $X \in \mathbb{R}^{n\times d}$ be the matrix that contains the training data (normalized between $0-1$) and $T \in \mathbb{R}^{n\times d}$ the matrix that contains the adversarial distortions. Let $A \in \mathbb{R}^{d\times r}$ be the projection matrix where each element is an i.i.d. $\mathcal{N}(0,1)$ random variable. Define b as a $1 \times r$ row vector where each element is drawn uniformly from $[0, 2\pi]$. Using these variables, we define $C \in \mathbb{R}^{n\times r}$ (which is linearly separable [25]), where the element at row i column j takes the following form.*

$$\begin{aligned} C_{i,j} = \cos\Big(\Big[&\big(X_{i,1}+T_{i,1}\big)A_{1,j} + \big(X_{i,2}+T_{i,2}\big)A_{2,j} + \dots \\ &+ \big(X_{i,d}+T_{i,d}\big)A_{d,j}\Big] + b_{1,j}\Big). \end{aligned} \tag{3.7}$$

Similarly, we define the matrices C^X, C^T, S^X and S^T where the element at row i column

j is defined as,

$$
\begin{aligned}
C^X_{i,j} &= \cos\Big(\big[X_{i,1}A_{1,j} + X_{i,2}A_{2,j} + \cdots + X_{i,d}A_{d,j}\big] + b_{1,j}\Big),\\
C^T_{i,j} &= \cos\Big(\big[T_{i,1}A_{1,j} + T_{i,2}A_{2,j} + \cdots + T_{i,d}A_{d,j}\big]\Big),\\
S^X_{i,j} &= \sin\Big(\big[X_{i,1}A_{1,j} + X_{i,2}A_{2,j} + \cdots + X_{i,d}A_{d,j}\big] + b_{1,j}\Big),\\
S^T_{i,j} &= \sin\Big(\big[T_{i,1}A_{1,j} + T_{i,2}A_{2,j} + \cdots + T_{i,d}A_{d,j}\big]\Big).
\end{aligned}
$$

We address the anomaly detection problem using the OCSVM algorithm introduced by [29], which separates the training data from the origin with a maximal margin in the projected space. Following the above nonlinear transformation of data, the dual form of the OCSVM algorithm can be written in matrix notation as

$$
\underset{\alpha}{\text{minimize}} \quad \frac{1}{2}\alpha^T CC^T\alpha, \text{ s.t } 0 \leq \alpha \leq \frac{1}{\nu n} \text{ and } \mathbf{1}^T\alpha = 1. \tag{3.8}
$$

Assumption 3.1. *The distortions made by the adversary are small s.t. the small angle approximation* $\cos(\theta) = 1 - \frac{\theta^2}{2}$ *holds.*

This assumption is reasonable because small distortions decrease the risk of the adversary being discovered, therefore a rational adversary would refrain from conducting attacks with significant distortions. We empirically validate this assumption using Local intrinsic dimensionality (LID) (refer Section 4.3.2 for details). LID is a metric that characterizes the dimension of the local subspace in which data samples lie, and recent works demonstrate that LID can be used to identify adversarial samples [100].

First, we distort a subset of data samples from the MNIST dataset under different s_{attack} values. We then calculate the probability density functions of the LID values of normal samples and poisoned samples. If Assumption 3.1 holds, the statistical distance between the two distributions should increase as the attack severity increases. We measure the statistical distance between the probability distributions using the Kullback-Leibler divergence (for two distributions P and Q, $D_{KL}(P||Q) = \sum_i P(i)\log(P(i)/Q(i))$).

As Figures 3.5a and 3.5b depict, the pdfs of the LID values of normal samples and poisoned samples have two distinguishable distributions. Furthermore, Figure 3.5c shows

that the KL divergence between the two pdfs increases as the attack severity increases. It can thus be suggested that Assumption 3.1 is reasonable in real-world scenarios.

Theorem 3.1. *Let r be the dimension to which the data is projected using the method in (3.7). Then, if Assumption 3.1 holds, the length of the weight vector w_p^* of a OCSVM is bounded above by*

$$\|w_p^*\|_2 \leq \|w_{pt}^*\|_2 + \frac{3\sqrt{r}}{2}. \tag{3.9}$$

Proof: Let $\tilde{\alpha}$ be the vector achieving the optimal solution in the projected space when adversarial distortions are present. Then, the solution for the primal problem in the projected space with adversarial distortions, defined as $\|w_{pt}^*\|_2$, can be obtained as

$$\left\|w_{pt}^*\right\|_2 = \left\|\tilde{\alpha}^T C\right\|_2. \tag{3.10}$$

Using the cosine angle-sum identity on the matrix defined by equation 3.7 (the symbol $\odot$ denotes the Hadamard product for matrices),

$$\left\|w_{pt}^*\right\|_2 = \left\|\tilde{\alpha}^T\left(C^X \odot C^T\right) - \tilde{\alpha}^T\left(S^X \odot S^T\right)\right\|_2. \tag{3.11}$$

Using the reverse triangle inequality we obtain

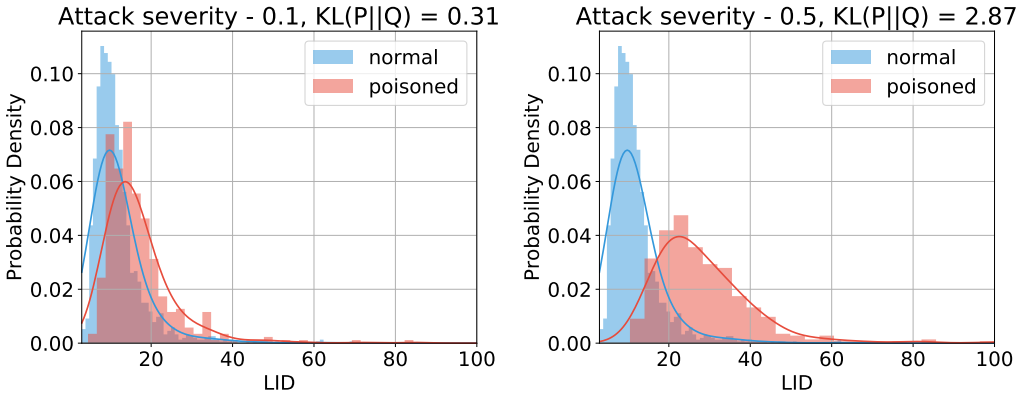
$$\left\|w_{pt}^*\right\|_2 \geq \left\|\tilde{\alpha}^T\left(C^X \odot C^T\right)\right\|_2 - \left\|\tilde{\alpha}^T\left(S^X \odot S^T\right)\right\|_2. \tag{3.12}$$

From the constraint conditions of the OCSVM problem (3.8), we get $\mathbf{1}^T\tilde{\alpha} = 1$. Also, as $\sin(\theta) \in [-1, 1]$ the inequality can be further simplified as,

$$\left\|w_{pt}^*\right\|_2 \geq \left\|\tilde{\alpha}^T\left(C^X \odot C^T\right)\right\|_2 - \sqrt{r}. \tag{3.13}$$

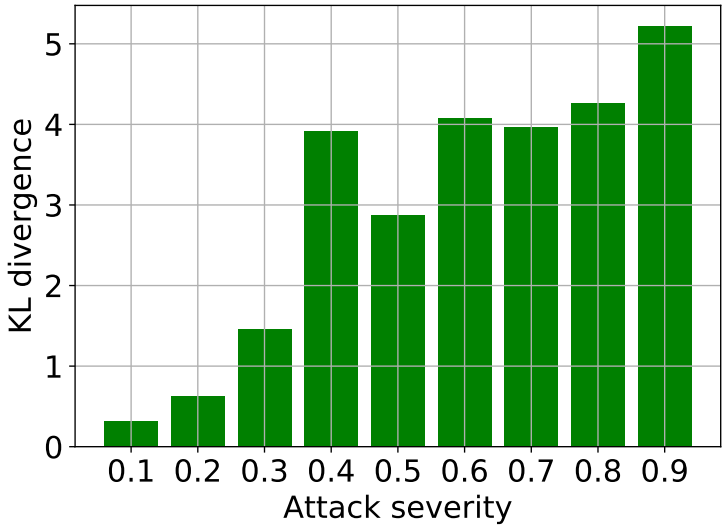
Due to Assumption 3.1, using small-angle approximation on C^T, we obtain

$$\left\|w_{pt}^*\right\|_2 \geq \left\|\tilde{\alpha}^T\left(C^X \odot \left(1 - \frac{TA \odot TA}{2}\right)\right)\right\|_2 - \sqrt{r}. \tag{3.14}$$



(a) $s_{attack} = 0.1$.

(b) $s_{attack} = 0.5$.



(c) KL divergence as s_{attack} is increased.

Figure 3.5: The top figures show the probability density functions of the LID values of normal samples and poisoned samples. The bottom figure demonstrates how the KL divergence changes as s_{attack} is increased.

Applying the reverse triangle inequality to the first term on the right hand side, we obtain

$$\left\|w_{pt}^*\right\|_2 \geq \left\|\tilde{\alpha}^T C^X\right\|_2 - \left\|\tilde{\alpha}^T \Big(C^X \odot (\frac{TA \odot TA}{2}) \Big) \right\|_2 - \sqrt{r}. \tag{3.15}$$

As the training data is normalized between (0 – 1), the maximum distortion magnitude that can be achieved is 1. Also, for any θ, $\cos(\theta) \leq 1$. Therefore, $C^X \odot (\frac{TA \odot TA}{2})$ on the right hand side can be replaced by a $n \times r$ matrix where each entry is $\frac{1}{2}$. Also, $\mathbf{1}^T \tilde{\alpha} = 1$, the inequality can be further simplified as,

$$\left\|w_{pt}^*\right\|_2 \geq \left\|\tilde{\alpha}^T C^X\right\|_2 - \frac{\sqrt{r}}{2} - \sqrt{r}. \tag{3.16}$$

Since the optimization problem is a minimization problem, as shown in (3.8), the optimal solution for the OCSVM without any distortion (i.e., α^*) would give a value less than or equal to the value given by $\tilde{\alpha}$. Thus,

$$\begin{aligned} \left\|\alpha^{*,T} C^X\right\|_2 &\leq \left\|w_{pt}^*\right\|_2 + \frac{3\sqrt{r}}{2}, \\ \left\|w_p^*\right\|_2 &\leq \left\|w_{pt}^*\right\|_2 + \frac{3\sqrt{r}}{2}. \end{aligned} \tag{3.17}$$

Attack Effectiveness on the binary SVM Margin

We address the classification problem using the ν-SVC algorithm, which uses the parameter ν to adjust the proportion of outliers, similar to the OCSVM algorithm [101]. Similar to the bound on OCSVMs, we analytically derive an upper-bound on $\|w\|_2$ of a binary SVM that has been trained on a nonlinearly projected undistorted dataset.

Definition 3.2. *The dual form of the ν-SVC classification algorithm is defined as,*

$$\underset{\alpha}{minimize} \quad \frac{1}{2}\alpha^T Y C C^T Y \alpha, \; s.t \; 0 \leq \alpha \leq \frac{1}{n}, \; \mathbf{1}^T Y \alpha = 0 \; and \; \mathbf{1}^T \alpha \geq \nu, \tag{3.18}$$

where Y is a $n \times n$ diagonal matrix that contains the labels.

Theorem 3.2. *If Assumption 3.1 holds, the length of the weight vector w_p^* of a binary SVM is bounded above,*

$$\|w_p^*\|_2 \leq \|w_{pt}^*\|_2. \tag{3.19}$$

The proof follows the same steps followed in Section 3.4.1 for OCSVMs. Using the constraint condition $\mathbf{1}^T Y\alpha = 0$ of the optimization problem instead of $\mathbf{1}^T\alpha = 1$ in Equation 3.16 leads to a trivial upper-bound for the binary SVM problem that is independent of the projection dimension r.

3.4.2 Linear Projections

Attack Effectiveness on the OCSVM Margin

The analysis adopts the approaches in [95] and [102] for anomaly detection using OCSVMs without labeled data. We use the length of the weight vector of a OCSVM (with a linear kernel) trained on a linearly projected, distorted dataset and present an upper-bound for the length of the weight vector of a OCSVM (with a linear kernel) trained in the input space of the data without any adversarial distortions.

Definition 3.3. *Let $V \in \mathbb{R}^{d\times d}$ be any matrix with orthonormal columns. Define $E := V^TV - V^TAA^TV$, and assume $\|E\|_2 < \epsilon, \epsilon \in (0, \frac{1}{2}]$ for a given A.*

Theorem 3.3. *Let w^* be the primal solution of the OCSVM optimization problem in the input feature space without adversarial distortions. Similarly, define w_{pt}^* as the primal solution in the presence of a malicious adversary in the projected space. Then the length of the weight vector w^* is bounded above by*

$$\|w^*\|_2^2 \leq \frac{(1+\lambda)}{|(1-\delta)|^2}\|w_{pt}^*\|_2^2, \tag{3.20}$$

where, $\delta := \dfrac{\|\alpha^T TA\|_2}{\|\alpha^T XA\|_2}$ *and* $\lambda = \dfrac{1}{2}\dfrac{\|E\|_2}{(1-\|E\|_2)}$.

Proof: Let $\tilde{\alpha}$ be the vector achieving the optimal solution in the projected space when adversarial distortions are present. Then, the optimization problem of the OCSVM in the projected space with adversarial distortions is given by

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2}\|\alpha^T(X+T)A\|_2^2, \\ \text{subject to} \quad & 0 \leq \alpha \leq \frac{1}{\nu n} \text{ and } \mathbf{1}^T\alpha = 1. \end{aligned} \tag{3.21}$$

As any dataset can be aligned in a manner that compels T to be positive and because $X \in [0,1]$, using the reverse triangle inequality we obtain,

$$|(1-\delta)|\|\alpha^T XA\|_2 \leq \|\alpha^T(X+T)A\|_2. \tag{3.22}$$

Let $Z_{pt}(\tilde{\alpha})$ be the optimal value of (3.21). Define next

$$Z_p(\tilde{\alpha}) := \|\tilde{\alpha}^T XA\|_2^2 \tag{3.23}$$

which is the optimal value without any malicious distortion $(T = 0)$. Then, it follows from (3.22),

$$Z_p(\tilde{\alpha}) \leq \frac{Z_{pt}(\tilde{\alpha})}{|(1-\delta)|^2}. \tag{3.24}$$

Let $Z(\alpha^*)$ denote the optimal value for the optimization problem in the input space. Although $\tilde{\alpha}$ is feasible, it is not an optimal solution. Therefore, by definition, $Z(\alpha^*) \leq Z(\tilde{\alpha})$.

Then, using the singular-value decomposition (SVD) of X, we get,

$$\begin{aligned} Z(\alpha^*) &= \frac{1}{2}\alpha^{*T}XX^T\alpha^*, \\ &= \frac{1}{2}\alpha^{*T}U\Sigma(V^TV)\Sigma U^T\alpha^*, \\ &= \frac{1}{2}\alpha^{*T}U\Sigma(E+V^TAA^TV)\Sigma U^T\alpha^*, \\ &= \frac{1}{2}\alpha^{*T}U\Sigma V^TAA^TV\Sigma U^T\alpha^* + \frac{1}{2}\alpha^{*T}U\Sigma E\Sigma U^T\alpha^*, \end{aligned} \tag{3.25}$$

where α^* is the vector achieving the optimal solution in the input feature space without

adversarial distortions and

$$\begin{aligned} Z_p(\tilde{\alpha}) &= \frac{1}{2}\tilde{\alpha}^T X A A^T X^T \tilde{\alpha}, \\ &= \frac{1}{2}\tilde{\alpha}^T U \Sigma V^T A A^T V \Sigma U^T \tilde{\alpha}. \end{aligned} \tag{3.26}$$

By substituting (3.26) into (3.25),

$$\begin{aligned} Z(\alpha^*) &= \frac{1}{2}\tilde{\alpha}^T U \Sigma V^T A A^T V \Sigma U^T \alpha^* + \frac{1}{2}\tilde{\alpha}^T U \Sigma E \Sigma U^T \tilde{\alpha} \\ &\leq \frac{1}{2}\tilde{\alpha}^T U \Sigma V^T A A^T V \Sigma U^T \tilde{\alpha} + \frac{1}{2}\tilde{\alpha}^T U \Sigma E \Sigma U^T \tilde{\alpha} \\ &= Z_p(\tilde{\alpha}) + \frac{1}{2}\tilde{\alpha}^T U \Sigma E \Sigma U^T \tilde{\alpha}. \end{aligned} \tag{3.27}$$

The second term of the above equation can be further analyzed by taking $Q = \tilde{\alpha}^T U \Sigma$ (note that $V^T V = I$)

$$\begin{aligned} \frac{1}{2}\tilde{\alpha}^T U \Sigma E \Sigma U^T \tilde{\alpha} &\leq \frac{1}{2}\|Q\|_2 \|E\|_2 \|Q\|_2, \\ &= \frac{1}{2}\|E\|_2 \|Q\|_2^2, \\ &= \frac{1}{2}\|E\|_2 \|\tilde{\alpha}^T U \Sigma V^T\|_2^2, \\ &= \frac{1}{2}\|E\|_2 \|\tilde{\alpha}^T X\|_2^2. \end{aligned} \tag{3.28}$$

Using the above result, (3.27) can be written as

$$Z(\alpha^*) \leq Z_p(\tilde{\alpha}) + \frac{1}{2}\|E\|_2 \|\tilde{\alpha}^T X\|_2^2. \tag{3.29}$$

The following steps result in a bound for the second term of the above equation.

$$\begin{aligned}
&|\tilde{\alpha}^T XX^T\tilde{\alpha} - \tilde{\alpha}^T XAA^T X^T\tilde{\alpha}| \\
=&|\tilde{\alpha}^T U\Sigma VV^T\Sigma U\tilde{\alpha} - \tilde{\alpha}^T U\Sigma VAA^T V^T\Sigma U\tilde{\alpha}|, \\
=&|(\tilde{\alpha}^T U\Sigma(VV^T - VAA^T V^T)\Sigma U\tilde{\alpha})|, \\
=&|\tilde{\alpha}^T U\Sigma E\Sigma U\tilde{\alpha}|, \\
\leq&\|E\|_2\|\tilde{\alpha}^T U\Sigma V\|_2^2, \\
=&\|E\|_2\|\tilde{\alpha}^T X\|_2^2.
\end{aligned} \tag{3.30}$$

The above inequality can be rewritten as

$$\left|\|\tilde{\alpha}^T X\|_2^2 - \|\tilde{\alpha}^T XA\|_2^2\right| \leq \|E\|_2\|\tilde{\alpha}^T X\|_2^2. \tag{3.31}$$

Thus, as shown in [102], using (3.23)

$$\begin{aligned}
\|\tilde{\alpha}^T X\|_2^2 &\leq \frac{1}{1-\|E\|_2}\|\tilde{\alpha}^T XA\|_2^2 \\
&\leq \frac{1}{1-\|E\|_2}Z_p(\tilde{\alpha})
\end{aligned} \tag{3.32}$$

From (3.24) and (3.29), by taking $\lambda = \frac{1}{2}\frac{\|E\|_2}{(1-\|E\|_2)}$,

$$\begin{aligned}
Z(\alpha^*) &\leq Z_p(\tilde{\alpha}) + \frac{1}{2}\frac{\|E\|_2}{(1-\|E\|_2)}Z_p(\tilde{\alpha}) \\
&= (1+\lambda)Z_p(\tilde{\alpha}) \\
&= (1+\lambda)\frac{Z_{pt}(\tilde{\alpha})}{|(1-\delta)|^2}
\end{aligned} \tag{3.33}$$

By definition, $w^* = \alpha^{*T} X$ and $w_{pt}^* = \tilde{\alpha}^T X$. Therefore, $Z(\alpha^*) = \frac{1}{2}\|w^*\|_2^2$, $Z_{pt} = \frac{1}{2}\|w_{pt}^*\|_2^2$.

$$\|w^*\|_2^2 \leq \frac{(1+\lambda)}{|(1-\delta)|^2}\|w_{pt}^*\|_2^2. \tag{3.34}$$

3.4.3 Discussion of the Theorems

In Theorems 3.1 and 3.2, we derive an upper-bound for the norm of the weight vector $\|w_p^*\|_2$ of a SVM/OCSVM that has been trained on projected undistorted data. The separating margin of a OCSVM is given by $\rho/\|w\|_2$, where ρ is the offset and w is the vector of weights. Similarly, in binary SVMs, the data from two classes are separated by the margin $2\rho/\|w\|_2$. This implies that a small $\|w\|_2$ corresponds to a large margin of separation. Therefore, the difference between $\|w_p^*\|_2$ and $\|w_{pt}^*\|_2$ is therefore an indicator of the attack's effectiveness.

In both cases, the strength of the adversary's attacks will be reflected in the value of the upper-bound (i.e., $\|w_{pt}^*\|_2$ component). Therefore the upper-bounds can be used to measure the impact of different attack algorithms on SVMs/OCSVMs in the same projected space (i.e., using the same projection direction A). For OCSVMs, the learner has the advantage to make the upper-bound of $\|w_p^*\|_2$ tighter by reducing the dimensionality of the dataset (i.e., r). Therefore, by projecting the data to low dimensional spaces, the learner is able to reduce the adversary's effects on the margin of separation (i.e., $\rho/\|w\|_2$) of a OCSVM. The upper-bound derived for binary SVMs is rudimentary, and the relationship between $\|w_p^*\|_2$ and $\|w_{pt}^*\|_2$ is as anticipated.

In Theorem 3.3, we derive an upper-bound on w^* (the primal solution of a OCSVM in the input feature space without adversarial distortions) using w_{pt}^* (the primal solution of a OCSVM in the presence of a malicious adversary in the projected space). This analysis assumes that the data is linearly separable, therefore a linear kernel is used in the OCSVM. From Theorem 3.3, we see that solving the OCSVM optimization problem using distorted data in the projected space results in a margin that is comparable to the margin of a OCSVM trained on undistorted data in the input feature space. The parameter δ is the ratio between the contribution from the distorted data to the objective function and the contribution from the normal data to the objective function. In the special case of $\delta = 1$, the upper-bound of $\|w^*\|_2^2$ diverges to infinity.

Table 3.1: Datasets used for training and testing with OCSVMs.

Dataset	Training size	Test size	Normal	Anomaly
MNIST	2,000	1,200	digit '9'	digit '8'
CIFAR-10	3,650	1,200	airplane	truck
SVHN	4,200	1,200	digit '8'	digit '0'

Table 3.2: Datasets used for training and testing with binary SVMs.

Dataset	Training size	Test size	Innocuous	Attack
MNIST	2,000	1,200	digit '9'	digit '1'
CIFAR-10	3,800	1,200	airplane	truck
SVHN	4,200	1,200	digit '8'	digit '0'

3.5 Evaluating the Detection Capability of the Defense Framework

We demonstrate the effectiveness of our novel defense framework on several datasets. This section describes how the datasets are obtained, pre-processing and other procedures of the experimental setup. We also show the applicability of our framework to real world security applications by simulating a particular network security application.

3.5.1 Experimental setup

Datasets: For experiments using binary SVMs, we choose data from two classes, considering one class as the innocuous class and another as the attack class. For experiments using OCSVMs, we generate single-class (unlabeled) datasets considering one of the original classes as the normal class, and a different one as the anomaly class. For each dataset, we create two test sets (with a normal to anomaly ratio of 5 : 1): (i) a clean test set (called test_C) with undistorted anomaly/attack data and normal data, (ii) a distorted test set (test_D) with its anomaly/attack points distorted. Tables 3.1 and 3.2 give the class and number of samples used in each training and test set.

Experimental setup: Attacks of different intensities are conducted (creating train_D) by varying the attack severity s_{attack} and attack percentage p_{attack}. In anomaly detection

problems, it is unlikely to find a large percentage of attack points within the training set, therefore we choose 5% for p_{attack} (percentage of distorted points in the training set) when using OCSVMs. For binary SVMs we choose 10%, 20%, 30% and 40% for p_{attack} as done in prior works in this area. We specifically choose the values 0.3, 0.4, 0.5 and 0.6 for s_{attack}. For comparison, we test all the attack scenarios against learners using the RBF kernel in the input feature space.

For nonlinear projections, we choose 20%, 40%, 60% and 80% of the input dimension as the dimensions to which the datasets are projected. The test sets are projected using the same parameters that give the highest compactness for the corresponding distorted training set. The learner then uses the projected training set to train a SVM model with a linear kernel, and the resulting model is evaluated using the test sets. For these experiments the ν parameter of the learners are kept fixed across all experiments conducted for each dataset. Since ν sets a lower bound on the fraction of outliers, it is crucial to keep its value fixed across different attack scenarios in order to evaluate the interplay between the adversarial distortions and the performance. As RBF kernels are used by the learners in the input space, we use the same *gamma* values in the low rank kernel approximation using Equation 3.4 in order to have identical kernel parameters. The ν and γ parameters for each dataset are found by performing grid searches on clean datasets.

Evaluation metric: For comparison purposes, we also train the learners using an undistorted training set (called train_C). We report the performance against test_C and test_D using the f-score (classification performance of the learners) and AUC (classification performance around the decision boundary). We also report the FNRs, which indicate the percentage of attack data points that are classified as normal data points by the learners.

3.5.2 Engineering Case Study: Identifying Adversaries from Radio Communication Signals

In this section we present an experimental evaluation of the developed framework for OCSVMs in the context of a real world security application: identifying rogue communication nodes using captured radio signals. We present the problem being addressed, data collection and pre-processing procedures followed by the learning model.

Application setting: In a given populated area, a multitude of individuals communicate with each other using a plethora of devices. While a majority of parties utilize such devices for innocuous, day-to-day activities (civilians), there may be a few malicious individuals (rogue agents) whose purpose is to cause harm and disrupt the lives of others. Even though these rogue agents would prefer to remain hidden, they would need to communicate electronically in order to plan and coordinate their activities. Communications are increasingly encrypted at various layers for privacy reasons. It is natural to assume that rogue agents prefer to conceal their radio communications among the civilian (background) radio traffic while enjoying the privacy protection provided by encryption systems.

Identifying rogue agents based on their wireless communication patterns is not a trivial task, especially when they deliberately try to mask their activities. An inherent assumption we make is that communication patterns of the rogue agents differ from those of regular background traffic to some degree, otherwise, the detection problem would be infeasible. Rogue agents would naturally prefer to avoid the standard networks used by civilians as there is a possibility for eavesdropping attacks, but using specialized communication equipment alone would highlight their presence if the radio spectrum was to be analyzed. Therefore, we can safely assume that rogue agents would strategically utilize specialized radio equipment as well as third party network infrastructure in order to avoid detection.

The above scenario can be posed as an anomaly detection problem where the learner creates a representation of normal data (i.e., civilians) using the data captured by the sensors and attempts to identify anomalies (i.e., rogue agents). In the aforementioned problem, if the rogue agents alter their communication patterns to resemble those of civilians to some extent during the initial stages of system deployment, they would be able to inject malicious data points into the dataset that will be used by the learner to create the anomaly detection model. As the learner cannot distinguish the radio signals of the rogue agents from those of the civilians, the learner would use the entire dataset collected by the sensors to train the anomaly detection model. This would result in a deformed representation of the normal data in the learned model. Therefore, during the evaluation

(operational) phase, the rogue agents would be able to evade the classifier without having to use identical communication patterns to those of the civilians.

Network Simulation and Data Collection: Simulations are performed using the INET framework for OMNeT++ [103] (datasets available at `https://github.com/sandamal/omnet_simulation`). In order to conduct a realistic simulation, signal attenuation, signal interference, background noise and limited radio ranges are considered. The nodes (civilians, rogue agents and listeners) are placed randomly within the given confined area. The simulator allows control of the frequencies and bit rates of the transmitter radios, their communication ranges, message sending intervals, message lengths, sensitivity of the receivers, minimum energy detection of receivers among other parameters. It is assumed that all nodes communicate securely, therefore the listeners are unable to access the content of the captured messages. Duration of reception, message length, inter arrival time (IAT), carrier frequency, bandwidth and bitrate are obtained as features using the data captured by the listeners.

Since the objective is to classify transmission sources, we consider the data received by the three closest listeners (using the power of the received signal) of each transmission source. The duration, message length and IAT of the messages received by each listener is averaged every five minutes, which results in 108 ($12 \times 3 \times 3$) features in total. Adding the latter three parameters (fixed for each transmission source) gives the full feature vector of 111 features. Using the collected data, we create two training datasets, train_C, train_D and two test datasets test_C and test_D. The two training datasets consist of 95% civilian data points and 5% rogue agent data points, while the two test datasets consist of 80% civilian data points and 20% rogue agent data points. In both train_D and test_D, the rogue agent data points are distorted (i.e., they deliberately changed their communication patterns to deceive the learner).

Learning model: We choose 20%, 40%, 60% and 80% of the input dimension as the dimensions to which the datasets are projected. By performing a grid search for parameters using only clean data, we set $\nu = 0.13$ and $\gamma = 0.009$. Identical parameter values are used in the OCSVM with a RBF kernel in the input space as well as in the OCSVMs that used

kernel approximations in the lower dimensional spaces.

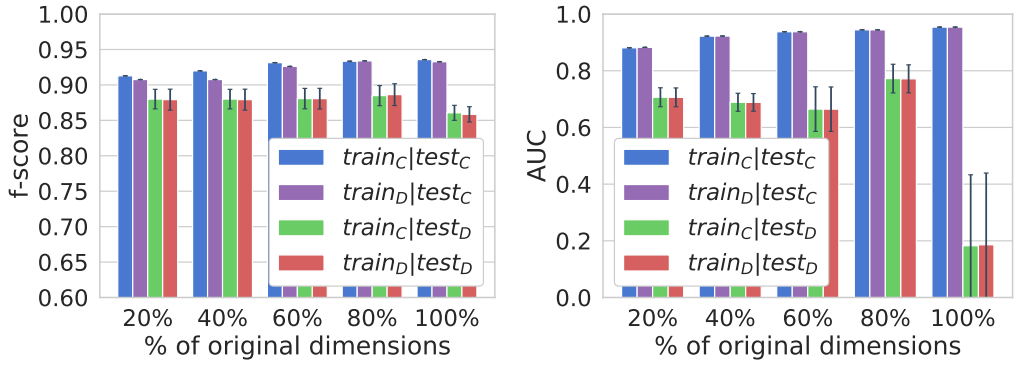
Game Formulation: Finally, we model the interaction between the rogue agents and the listeners as the bimatrix game explained in Section 3.3.1. Since the adversary can vary the severity of attacks during the initial stages of system deployment (i.e., data collection) by changing their communication parameters, we select four such communication patterns as the finite set of actions available for the adversary. If the adversary does not carry out an attack, we consider s_{attack} to be 0. If the rogue agents closely mimic the civilians (resulting in a small shift of the margin) we consider s_{attack} to be small. Conversely if rogue agents change their patterns to ones that are significantly different than those of the civilians, we consider s_{attack} to be larger. As the learner uses the projection based method to detect adversarial samples, the dimensions to which the data is projected will be used as the set of actions available for the learner.

$$a_M \in \{0, 0.3, 0.4, 0.5\},\ a_L \in \{20\%, 40\%, 60\%, 80\%, 100\%\}. \tag{3.35}$$

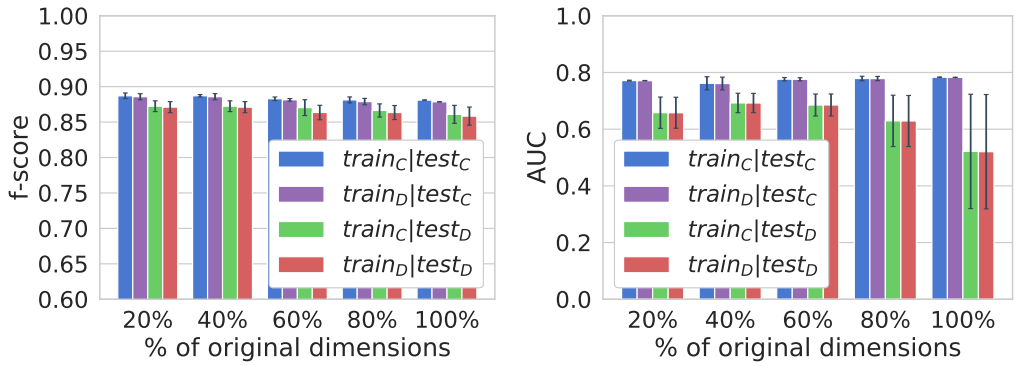
We present the outcome of the game in Section 3.6.3.

3.6 Results and Discussion

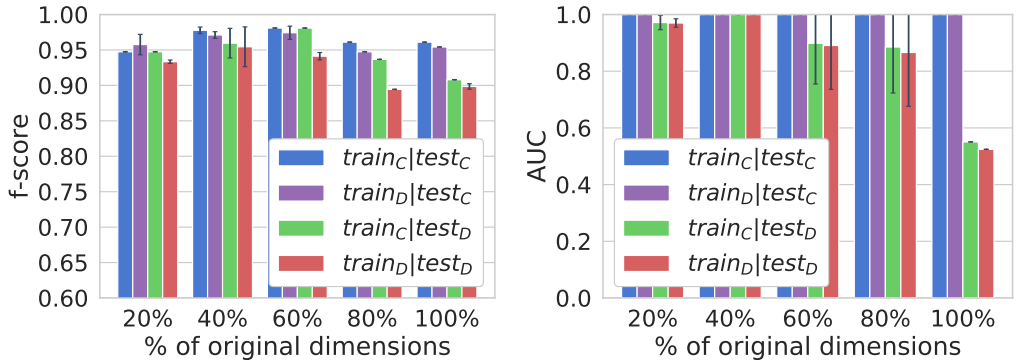
We demonstrate the effectiveness of the first part of the proposed defense mechanism (i.e., selective nonlinear projection) on three benchmark datasets. As most real world data are not linearly separable, we focus on the performance of nonlinear random projections using the indices introduced in Section 3.3 when an active adversary is conducting a directed attack by maliciously distorting the data. We extensively investigate the effectiveness of the defense framework under different attack configurations (i.e., s_{attack} and p_{attack} values) when the adversary is using the attack model described in Section 3.2. Then, we present the outcomes of the game presented in Section 3.3.1 using two datasets. Finally, to further evaluate the robustness added by the framework, we compare its effectiveness against other related attack/defense strategies in the literature.



(a) MNIST - f-score. (b) MNIST - AUC.

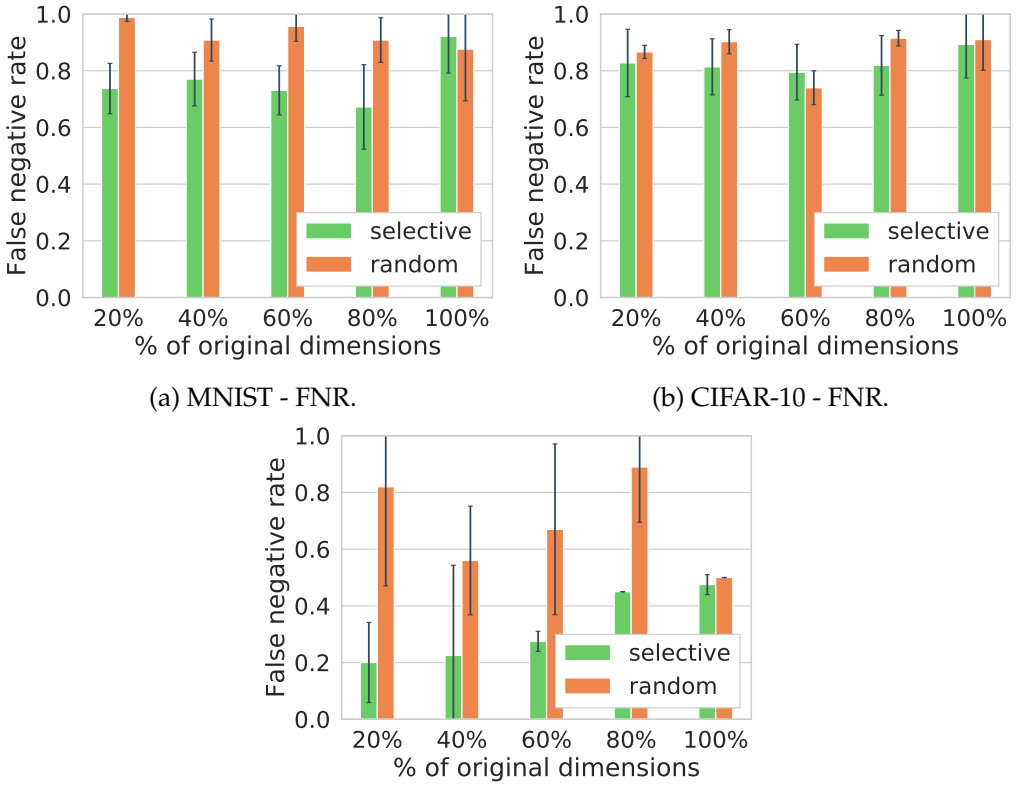


(c) CIFAR-10 - f-score. (d) CIFAR-10 - AUC.



(e) Omnet - f-score. (f) Omnet - AUC.

Figure 3.6: The performance of OCSVMs under attacks on integrity when the training takes place in different dimensional spaces. The left column compares the f-scores of OCSVMs trained on train_C and train_D against the two test sets: test_C and test_D. The right column shows the corresponding AUC values for each dataset.



(a) MNIST - FNR.

(b) CIFAR-10 - FNR.

(c) Omnet - FNR.

Figure 3.7: Comparison of the FNR values of OCSVMs under an integrity attack (i.e., trained on train_D and evaluated using test_D).

3.6.1 OCSVM results

Figures 3.6 and 3.7 present f-scores, AUC values and FNRs at different projected spaces under adversarial distortions. For f-score and AUC values, we present four results; when (i) trained using train_C, and tested with test_C; (ii) trained with train_C and tested with test_D; (iii) trained with train_D and tested with test_C; and finally (iv) trained with train_D and tested with test_D. We present the FNR values when the models are trained with train_D and tested with test_D.

Absence of an attack: First, in the absence of an attack (i.e., $\text{train}_C|\text{test}_C$), the classification performance of OCSVMs (i.e., f-score) trained on nonlinearly projected data are close to the performance of the OCSVM trained on the input feature space (2% lower on MNIST, 1% and 8% higher on CIFAR-10 and SVHN). In some cases (e.g., SVHN where the images contain parts of the adjacent digits, making the data noisy) we see that the

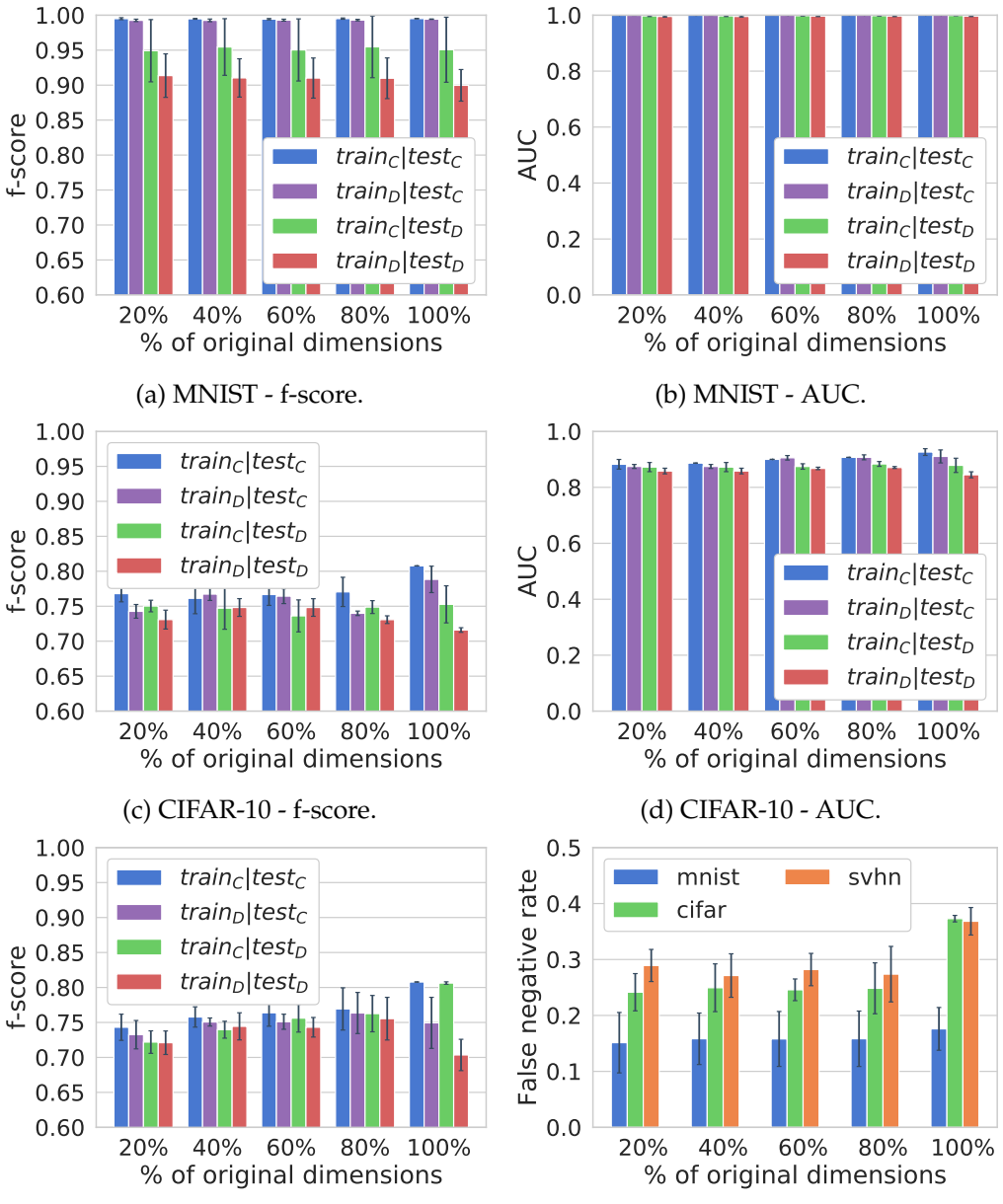
f-score can be higher in lower dimensional spaces than the input dimension OCSVM. We speculate that this occurs because a clearer separation can occur among data points from different classes when data is projected to a lower dimensional space, as shown in [26].

Under an attack on integrity: We observe that the f-scores of $\text{train}_D|\text{test}_D$ in all three datasets, across all the dimensions, are up to 1% less than the f-scores of $\text{train}_C|\text{test}_D$. Even though the difference is not large, this indicates that a OCSVM trained on clean data can identify adversarial samples better than a OCSVM trained on distorted data. Consequently this shows that OCSVMs are not immune to integrity attacks by design, and by carefully crafting adversarial data points, adversaries can increase the classification error of OCSVMs.

A comparison of the f-score in the $\text{train}_D|\text{test}_D$ case shows an increase in f-score when the proposed defense algorithm is used compared to an OCSVM in input space. The increased f-score confirms that by projecting data to a lower dimensional space using a carefully selected direction, we can identify adversarial samples that would not have been identifiable in the input space. This is further supported by Figure 3.7, which shows the average false negative rates of the OCSVMs under different levels of integrity attacks. We find that there is a significant improvement in detecting adversarial samples under the proposed approach compared to a OCSVM in input space (e.g., 7.25% on SVHN, 9.75% on CIFAR-10, and 24.87% on MNIST). The AUC values shown by the figures in the right column of Figure 3.6 further supports this. As reducing data to low dimensional spaces results in a loss of information, we believe that when the data dimensionality is reduced below a certain threshold, the performance would start to degrade.

Effectiveness of the compactness index: The effectiveness of the compactness index for selecting projection directions can be seen by the difference in FNRs in Figure 3.7. Although random projection directions have resulted in higher FNRs compared to selective projection directions in most test cases, it is possible for a randomly sampled direction to be one that minimizes the adversarial distortions. But the probability of obtaining such a direction will be low due to the large number of possible directions available for high dimensional datasets and would depend significantly on the distribution of the data clouds. An alternative approach to finding good directions would be to train an anomaly

detection model on every projected dataset and test its accuracy on a validation set. But the proposed index would be able to achieve this with much less computational burden.



(a) MNIST - f-score.

(b) MNIST - AUC.

(c) CIFAR-10 - f-score.

(d) CIFAR-10 - AUC.

(e) SVHN - f-score.

(f) FNRs.

Figure 3.8: The performance of binary SVMs under attacks on integrity when the training takes place in different dimensional spaces. The left column compares the f-scores of SVMs trained on train_C and train_D against the two test sets: test_C and test_D. Figures 3.8b and 3.8d show the corresponding AUC values for the MNIST and CIFAR-10 datasets. Figure 3.8f shows the FNR of binary SVMs under an integrity attack (i.e., trained on train_D and evaluated using test_D).

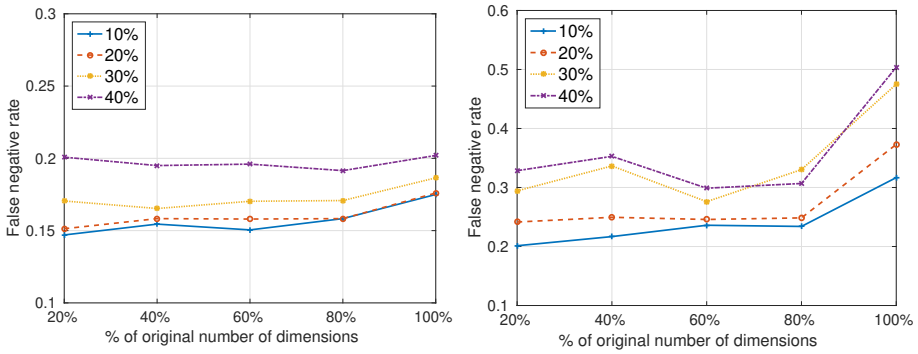
3.6.2 Binary SVM results

Absence of an attack: Figures 3.8a,3.8c and 3.8e present the f-score values at different dimensional spaces when the adversary distorts 20% (i.e., p_{attack}) of the training data. We show the average f-score along with the standard deviations when the attack severity varies from 0.3 to 0.6. Compared to the anomaly detection scenario using OCSVMs, the binary SVM trained on the input space with a RBF kernel outperforms the SVMs in the projected spaces when the training and testing data are clean (i.e., $\text{train}_C|\text{test}_C$) (e.g., 6.47% on SVHN, 4.64% on CIFAR-10, and less than 0.01% on MNIST).

Under an attack on integrity: Under adversarial conditions, the opposite can be observed with the defense framework giving better performance in terms of f-score (e.g., 5.19% on SVHN, 3.22% on CIFAR-10, and 1.39% on MNIST) (i.e., $\text{train}_D|\text{test}_D$). This characteristic is often found in defenses against adversarial examples where there is a trade-off between the learners' performance in the absence of attack, and their robustness under attack. Improving robustness often causes a decrease of the performance in the absence of attack. The resistance added against integrity attacks by our proposed approach is confirmed by Figure 3.8f, which shows the average false negative rates of the classifiers under the different attack severities. Again, we find that there is a significant improvement in detecting adversarial samples compared to the SVM in the input space (e.g., FNR reduction of 9.70% on SVHN, 13.15% on CIFAR-10, and 2.47% on MNIST).

Figures 3.8b and 3.8d show the AUC values for MNIST and CIFAR-10 that correspond to the f-score values in Figures 3.8a and 3.8c. Although the AUC values for CIFAR-10 show the effects of the adversary's attack on the performance of the SVM, the AUC values for MNIST fail to exhibit the attack's effects (where as the f-score values do). The AUC value represents the classification performance of a SVM around the separation boundary obtained from training, where as the f-score represents the performance of a SVM at the selected separation boundary. Therefore the AUC values can be considered as a supplementary result that supports the f-score and FNR results.

We also observe that for CIFAR and SVHN, when the dimension is reduced below 40% of the input dimension, the performance starts to degrade. We postulate that the explanation of this effect is the reduction in distance between classes with the dimension.



(a) MNIST. (b) CIFAR-10.

Figure 3.9: Comparison of the FNRs of binary SVMs when the adversary changes the percentage of distorted attack points in the training set (i.e., p_{attack}) from 10% to 40%.

As we reduce the dimension of the projection, we are able to reduce the effects of the adversarial distortions. But at the same time, there is a significant loss of useful information due to the dimensionality reduction. Due to the interplay between these two factors, the performance of SVMs reduces as we decrease the dimension beyond a dataset dependent threshold.

Effect of large percentages of attack data during training: Figure 3.9 depicts the FNRs of binary SVMs in different dimensional spaces when the adversary increases the percentage of distorted attack points in the training set from 10% to 40%. As expected, we see that when p_{attack} increases, the FNRs also increase. It should be noted that while an adversary theoretically has the ability to arbitrarily increase the number of attack points in the training set, in real world applications, greedily increasing the attack percentage would inevitably make the attack obvious.

3.6.3 Outcomes of the game

Figure 3.10 shows the payoff matrix of the adversary and learner when the deterministic game described in Section 3.3.1 is played on the MNIST dataset. By considering the best responses of both players, we obtain the Nash equilibrium solution to the game, which is $s_{attack} = 0.3$ and 20% of the original number of dimensions. We observe that these strategies are in fact the dominant strategies for both players. In a dominant strategy, each player's best strategy is unaffected by the actions of the other player, which gives a

		% of the original # of dimensions				
		20%	40%	60%	80%	100%
Attack severity	0	(1.50,0.867)	(1.63,0.848)	(1.66,0.845)	(1.37,0.854)	(1.40,0.859)
	0.3	**(1.62,0.838)**	(1.72,0.822)	(1.75,0.821)	(1.51,0.825)	(1.85,0.802)
	0.4	(1.48,0.847)	(1.59,0.829)	(1.63,0.827)	(1.36,0.837)	(1.80,0.802)
	0.5	(1.36,0.856)	(1.48,0.838)	(1.53,0.832)	(1.23,0.847)	(1.73,0.804)
	0.6	(1.27,0.862)	(1.39,0.843)	(1.42,0.838)	(1.14,0.855)	(1.58,0.812)

Figure 3.10: The utility matrix of the game played on MNIST, depicting the outcomes. The adversary is the row player and the learner is the column player and payoffs are displayed as (adversary utility, learner utility). The highlighted cell is the Nash equilibrium solution.

stronger outcome than the Nash equilibrium. Based on this result, we conclude that it is in the best interest of the learner to always project data to 20% of the original number of dimensions.

Similarly, Figure 3.11 shows the payoff matrix of the adversary and learner for the Omnet simulation dataset. As per Proposition 3.1 by Basar and Olsder [99], any deterministic bimatrix game that uses a positive affine transformation of the utility functions described in Section 3.3.1 used in this game would also have the same Nash equilibrium strategies for both players.

The Nash equilibrium solution to the game in this scenario is $s_{attack} = 0.4$ and 40% of the original number of dimensions. If the learner deviates from the Nash equilibrium solution unilaterally, there can be a reduction to its utility value up to 0.04. Similarly, if the attacker deviates from the Nash equilibrium solution unilaterally, the reduction to its utility would be up to 0.4. Therefore it is in the best interest of the learner to always project data to 40% of the original number of dimensions, and the attacker to use an attack severity of 0.4 in this particular problem. The average f-score values and FNRs are shown in Figure 3.6e and 3.7c for the different dimensions. The results show that the proposed framework can be successfully utilized in a practical application scenario such as this.

		% of the original # of dimensions				
		20%	40%	60%	80%	100%
Attack severity	0	(1.00,0.947)	(1.00,0.945)	(1.00,0.981)	(1.00,0.908)	(1.00,0.908)
	0.3	(0.95,0.935)	(0.85,0.974)	(1.10,0.938)	(1.30,0.894)	(1.35,0.896)
	0.4	(1.10,0.932)	**(1.25,0.935)**	(1.10,0.925)	(1.25,0.894)	(1.25,0.901)
	0.5	(1.10,0.913)	(1.15,0.933)	(1.75,0.830)	(1.75,0.844)	(1.75,0.844)

Figure 3.11: The utility matrix of the game played on simulation data, depicting the outcomes. The adversary is the row player and the learner is the column player and payoffs are displayed as (adversary utility, learner utility). The highlighted cell is the Nash equilibrium solution.

3.6.4 Comparison of defense framework

We compare the effectiveness of the proposed framework against related attack strategies and defense algorithms in the literature. First, we evaluate the performance of the algorithms when there is no attack present (NA). Then, we compare the performance against the *Restrained Attack (RA)* introduced by Zhou et al. [17], *Poisoning Attack (PA)* by Biggio et al. [40] and the *Coordinate Greedy (CG)* attack proposed by Li et al.[42]. We also compare the performance against the online Centroid Anomaly Detection (CAD) approach proposed by Kloft and Laskov [47] with the nearest-out replacement policy, LS-SVM proposed by Suykens and Vandewalle [18] and a Logistic Regression (LR) learner. The training and test datasets are generated in the same way as in our previous experiments. The regularization parameter C of LR was selected by performing a grid search on a clean data set.

The optimal strategy for the learner (i.e., 20% of the original number of dimensions) is selected using the game outcome in Section 3.6.3 and is used for all three datasets when the defense framework is used. Table 3.3 gives the FNR of each defense mechanism averaged over five cross validation sets. We see that the SVMs trained with the defense framework (SVM Fw and OCSVM Fw) have an increased ability to accurately detect adversarial samples during test time. For classification, the SVM with the framework (SVM Fw) has lower FNRs compared to the other learners in all test cases except for CIFAR-10 under PA and CG. Under PA, LS-SVM has a 1.9% lower FNR and under CG, LS-SVM has a 2.3% lower FNR. For anomaly detection, the OCSVM with the framework

Table 3.3: A comparison of the discrimination power (FNR) of the defense framework against different learners and attacks. The best results are highlighted in bold and columns corresponding to the proposed defense framework are highlighted in grey.

	Dataset	SVM	SVM Fw	LS-SVM	LR	OCSVM	OCSVM Fw	CAD
NA	MNIST	0.003	**0.002**	0.005	0.003	0.097	0.079	**0.000**
	CIFAR	**0.194**	0.200	0.198	0.200	0.210	0.212	**0.200**
	SVHN	0.210	0.209	**0.202**	0.204	0.290	**0.281**	0.285
RA	MNIST	0.122	**0.077**	0.084	0.146	0.694	**0.653**	1.000
	CIFAR	0.372	**0.237**	0.346	0.369	0.970	**0.835**	1.000
	SVHN	0.352	**0.234**	0.326	0.371	0.900	**0.790**	0.980
PA	MNIST	0.294	**0.203**	0.216	0.284	0.767	**0.675**	0.730
	CIFAR	0.364	0.223	**0.204**	0.361	0.814	**0.723**	0.863
	SVHN	0.384	**0.287**	0.321	0.410	0.863	**0.750**	0.886
CG	MNIST	0.417	**0.393**	0.396	0.426	0.797	**0.767**	0.818
	CIFAR	0.397	0.364	**0.341**	0.386	0.862	**0.784**	0.842
	SVHN	0.386	**0.298**	0.321	0.391	0.912	**0.845**	0.924

(OCSVM Fw) outperforms the other learners with consistently lower FNRs on all test cases.

We also observe that in the absence of an attack (i.e., NA), the performance of the SVMs with the defense framework are relatively less compared to the other learners. For classification, the SVM with no defense has a 0.6% lower FNR on CIFAR-10 and LS-SVM has a 0.7% lower FNR on SVHN. For anomaly detection, CAD has lower FNRs of 7.91% and 1.2% compared to OCSVMs with the defense framework when tested on MNIST and CIFAR-10. On SVHN however, the OCSVM with the framework has a 0.4% lower FNR. Note that the decrease in performance is relatively less compared to the advantage gained in the presence of an attack. This may be due to the fact that the RKS algorithm [25] only approximates the actual kernel matrix in a low dimensional space, therefore it is likely that some useful information is lost during the transformation. From this extensive comparison, we see that the defense framework is able to consistently reduce the FNRs across different attack strategies compared to other learners.

In summary, the above experiments demonstrate that (i) OCSVMs and SVMs are vulnerable to adversarial *attacks on integrity*, (ii) the performance in the projected spaces,

when there are no attacks, is comparable to that in the input space, but with less computational burden, and most importantly, (iii) by projecting a distorted dataset to a lower dimension in an appropriate direction we can increase the robustness of SVMs w.r.t. integrity attacks.

3.7 Summary

This chapter presents a theoretical and experimental investigation on the effects of integrity attacks that poison the training data and affect the learners in the course of training. We introduce a unique framework that combines nonlinear data projections using novel ranking indices that we introduce, together with SVMs and game theory. Through the network simulation we show that the flexibility of the proposed framework allows it to be applied to real world security applications. Our numerical analysis focuses on the performance of the proposed defense framework under adversarial conditions. The results suggest that SVMs and OCSVMs can be significantly affected if an adversary can manipulate the data on which they are trained. For each dataset, with very high probability, there is at least one dimensionality and projection direction that can accurately identify adversarial samples that would have been missed by a SVM or a OCSVM in the input space. Therefore, our approach can be utilized to make SVM based learning systems secure by (i) reducing the impact of possible adversarial distortions by contracting and separating data points from different classes in the projected space, and (ii) making it challenging for an adversary to guess the underlying details of the learner by making its search space unbounded through a layer of randomness.

However, the proposed defense cannot be used against adversarial attacks that flip a subset of training labels as finding a projection direction that separates data from the two classes is not possible in such a scenario. We look into label flipping attacks in the following chapter.

Chapter 4

LID Based Defense For SVMs Against Poisoning Attacks

In this chapter, we extend the SDR based classification problem described in Section 3.5.2 into a distributed adversarial learning problem. A cognitive network of SDRs can be used for distributed detection of anomalous communications by focusing on their statistical characteristics due to end-to-end encryption. This is not a trivial task because classifiers are vulnerable to targeted malicious attacks such as label flips and input perturbations. In this chapter, we introduce a novel defense based on Local Intrinsic Dimensionality (LID), which is a promising metric that characterizes the subspace of regions around the data samples against such malicious attacks.

We introduce a new approximation of LID called K-LID that uses kernel distance in the LID approximation, which allows LID to be calculated in a high dimensional transformed space. We use a weighted Support Vector Machine (SVM) distributed among the SDRs to defend against such attacks and achieve the original detection goal. The weighting mechanism uses K-LID as the distinguishing characteristic and de-emphasizes the effect of malicious data samples on the SVM decision boundary. Each sample is weighted on how likely its K-LID value is from the non-attacked K-LID distribution than the attacked K-LID distribution. This chapter addresses the research question 3 from Chapter 1.

4.1 Introduction

SOFTWARE-DEFINED radios (SDRs) with substantial computing and networking capabilities can be utilized as a low-cost scanner array for distributed detection of transmissions. We focus on the problem of identifying malicious transmission sources (rogue agents) who attempt to conceal their radio communications among the background (e.g., civilian) radio traffic (Figure 4.1). Due to the prevalence of encryption methods, identification of the transmission sources has to be based on their statistical charac-

teristics. Distinguishing such rogue agents' radio communications from background traffic requires the correlation of information in multiple dimensions such as time, frequency and communication protocols. For a network with a large number of SDR listeners, a distributed classifier is the obvious choice due to being scalable and efficient as it requires less communication overhead. Support Vector Machines (SVMs) [24] are proven supervised learning models that have been applied as classifiers in a wide range of security related applications and have several distributed variants (DSVMs) [33].

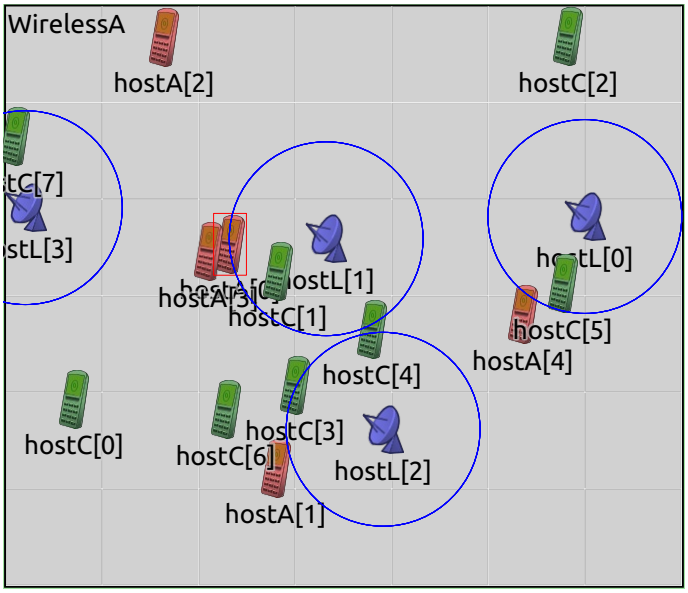
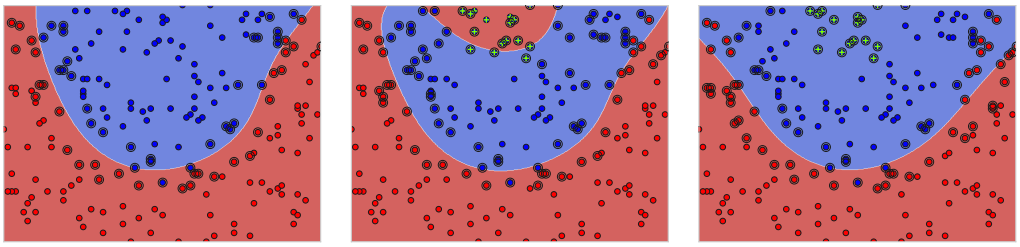


Figure 4.1: A representation of the SDR listeners (blue), civilians (green) and rogue agents (red) in OMNeT++ simulator.

Even though SVMs are able to withstand noisy training data by design, it has been shown that maliciously contaminated training data can degrade their classification performance significantly [17, 41, 40]. As Figures 4.2a and 4.2b show, carefully calculated manipulation of training data (labels or features) would result in a significantly different decision boundary compared to the boundary the SVM would have obtained if the data was pristine. This problem is aggravated in distributed detection settings as attackers have many potential entry points and even if one node is compromised, the effects of the attack can propagate through the entire detection network [34, 35]. Therefore it is important to design defense mechanisms against training data manipulations (i.e., input perturbations or flipping labels) that can subdue their effects.



(a) SVM - no attack. (b) SVM - alfa attack. (c) LID-SVM - alfa attack.

Figure 4.2: Results on a two dimensional synthetic dataset for an SVM with a RBF kernel ($C = 1$ and $\gamma = 0.5$). The decision boundary of the SVM trained in the absence of an attack is shown on the left. The middle figure shows the decision boundary of the SVM with no defense when a subset of labels are flipped (from blue to red) using the Adversarial Label Flip Attack (alfa) attack. The figure on the right shows the decision boundary of the LID-SVM in the presence of the alfa attack. The flipped samples are shown as green crosses and the support vectors are circled.

Prior work in the literature uses data sanitation (i.e., filtering out malicious data) as a defense mechanism when learning in the presence of maliciously altered data [16]. Data sanitation may seem like a trivial solution for applications such as image classification under label flipping attacks, where even human experts can detect the flipped samples. But for large data sets under input perturbation attacks filtering attacked samples becomes difficult. Expert filtering is also infeasible in other real-world applications such as communication/IoT systems where the data is high dimensional and cannot be transformed into a form that can be easily perceived by humans. Alternatively, forcing all the data samples to contribute to the decision boundary during training is a viable option [18, 45]. But eventually, as the percentage of attacked samples in the training data increases, their contribution would subdue the contribution of non-attacked samples.

We aim to address this problem using Local Intrinsic Dimensionality (LID), a metric that gives the dimension of the subspace in the local neighborhood of each data sample. Recent works in the literature show that LID can be used to identify the regions where adversarial samples lie [20, 21]. LID has been applied for detecting adversarial samples in Deep Neural Networks (DNNs) [100] and as a mechanism to reduce the effect of noisy labels for training DNNs [104]. *In this chapter, we propose a novel LID estimation called K-LID which can distinguish attacked samples from non-attacked samples based on the characteristics of*

the data itself. We then use this distinguishing metric to develop a distributed weighted SVM to be used by the SDR listeners to defend the surveillance network against training data manipulations. Note that the learner does not need to possess knowledge of the attacked samples, i.e., which samples are attacked and which are not, it only requires the probability distributions of K-LID values of attacked and non-attacked samples, which can be obtained using one of the many ways that we describe in Section 4.3.4. Figure 4.2c shows how the proposed defense mechanism can withstand sophisticated attacks with minimal deformations to the decision boundary.

Our main **contributions** are summarized as follows:

1. We present a scheme for distributed detection and classification of rogue transmissions using a SDR scanner array and distributed SVMs.

2. We introduce a novel defense strategy to increase the attack resistance of SVMs against label flipping attacks as well as input perturbation attacks.

3. The proposed defense uses a LID-based sample weighting mechanism that:
   - introduces a novel approximation of LID using kernel distances called *K-LID* that calculates the LID values of data samples in the hypothesis space (i.e., a high dimensional transformed space);
   - uses the likelihood ratio of each data sample (i.e., how many times more likely their K-LID values are from the non-attacked K-LID distribution than the attacked K-LID distribution) as a distinguishing factor and incorporate this knowledge into the SVM training process as weights.

4. We show through numerical experiments conducted with simulated and real-world data sets that our proposed approach can increase the attack resistance of SVMs against training time attacks by up to 10% on average.

5. We show experimentally that a distributed SVM detection system has 44% less communication overhead compared to a centralized SVM with only a 3.07% reduction in detection accuracy on average.

The remainder of the chapter is organized as follows. Section 4.2 formally defines the

problem being addressed followed by the defense methodology in Section 4.3. Section 4.4 describes the empirical analysis on several real-world datasets followed by the results and discussion. The concluding remarks of Section 4.5 conclude the chapter.

4.2 Problem Definition

We consider an adversarial learning problem in a distributed detection system that uses SVMs. The adversary's goal is to corrupt the classification model generated in the training phase to maximize the classification error of the detection system. This type of attack is referred to as a *poisoning availability attack* in the adversarial learning literature where the adversary's goal is to affect the prediction results indiscriminately, without seeking specific mis-predictions. Take $S := (X, y)$ to be the labeled training data where $X \in \mathbb{R}^{n \times d}$ and $y_i \in \{\pm 1\}$ for $i \in \{1, \ldots, n\}$.

To achieve its goal, the adversary either flips a fraction of training labels ($\tilde{S} := (X, \tilde{y})$) or perturbs the features of a fraction of data samples ($\tilde{S} := (\tilde{X}, y)$). When the learner trains on the contaminated data, it obtains a distorted decision boundary that is significantly different from the decision boundary it would have obtained if the data was pristine. In the following section, we introduce the different attack strategies an adversary may use against a SVM. In Section 4.3, we introduce our novel LID based defense models for SVMs against such training time attacks followed by empirical evidence in Section 4.4 to prove their effectiveness.

4.2.1 Adversarial Models

Label Flipping Attacks

Through a series of works, Xiao et al. [46, 45, 12] introduced several attack models that carefully select a subset of training labels to be flipped in order to maximize an SVM's classification error. The attack models assume that the attacker has perfect knowledge of the attacked system (white-box attack). This means that the attacker is aware of the SVM parameters C and γ and training data $S := (X, y)$. While the attacker's ability is

over-estimated, it can be considered as a worst case scenario for the defender. Moreover, relying on secrecy for security is considered as a poor practice when designing attack resilient learners [44].

The attack forces the learner to erroneously shift the decision boundary such that there is a significant deviation from a SVM trained on a non-flipped dataset. The attacker is restricted such that it is only allowed to flip the labels of the training data and only a maximum of L label flips are allowed. L bounds the attacker's capability and is fixed *a priori*. The main attack strategies we test against are:

- **Adversarial Label Flip Attacks (alfa):** take $\tilde{y}$ to be the contaminated training labels. The attack model can be considered as a search problem for $\tilde{y}$ that achieves the maximum difference between the empirical risk for classifiers trained on $\tilde{y}$ and y.
- **ALFA based on Hyperplane Tilting (alfa-tilt):** in this attack, the tilt in the separating margin before and after distorting the dataset is used as a surrogate to select the optimal label flips instead of classification error in alfa.
- **Farfirst:** samples that are furthest from the separating margin of the non-flipped SVM are flipped.
- **Nearest:** samples that are nearest to the separating margin of the non-flipped SVM are flipped.
- **Random:** a subset of samples are randomly selected from the training data and flipped.

Input Perturbation Attacks

For binary SVMs, Biggio et al. [40] introduced the poisoning attack algorithm **(PA)** that perturbs data points in feature space such that there is a maximal increase in the classification error. The authors assume that the attacker has perfect information, resulting in a worst-case analysis for the defenders. The authors address the problem of finding the best attack points by formulating an optimization problem that maximizes the learner's validation error. They empirically demonstrate that the gradient ascent algorithm can

identify local maxima of the non-convex validation error function. The main highlight of this work is that it can identify attack points in transformed spaces (using kernel methods).

In contrast, the restrained attack **(RA)** introduced by Zhou et al. [17] conducts attacks in input space. For each malicious data point x_i the adversary aims to alter, it picks a innocuous target x_i^t and perturbs x_i towards x_i^t. The amount of movement is controlled by two parameters, the discount factor (C_ξ) and the attack severity (f_{attack}).

Li et al.[42] introduced the Coordinate Greedy **(CG)** attack which is modeled as an optimization problem. The objective of the attacker is to make the perturbed data points appear as benign as possible to the classifier while minimizing the modification cost. For each malicious data point x_i, the attacker iteratively chooses a feature and greedily updates it to incrementally improve the attacker's utility.

In the distributed detection system we are concerned with in this paper, an attacker would need to gain access to one or more of the listener nodes in order to carry out a label flipping attack. This may happen through the use of malware or unauthorized access. For input perturbation attacks, however, the attack can be carried out by altering the communication parameters of the rogue agents.

4.3 Defense Model for Distributed SVMs

In this section, we present the inner workings of our novel LID based defense algorithm for distributed SVMs. We also briefly introduce the theory of LID for assessing the dimensionality of data subspaces. We extend that framework by introducing carefully designed weights to improve the resistance against training time attacks.

The defense algorithm we propose consists of several components. First, the LID values of all the data samples are calculated using a novel LID approximation. Then, for each data sample, we calculate how many times more likely its LID value is from the LID distribution of non-attacked samples than the LID distribution of attacked samples, i.e., likelihood ratio (LR). Subsequently, we fit a smooth function to the LR values to be able to predict the LRs for unseen LID values. Finally, during SVM training, we weight each

sample by the LR function value corresponding to its LID value. In the following sections we describe in detail each of the aforementioned components.

4.3.1 Distributed Weighted SVMs

First, we present how SVMs can be distributed among multiple compute nodes following an existing DSVM framework [32].

In SVMs, in order to penalize large slack values (i.e., ξ_i), a regularization constant C was introduced to the optimization problem. C penalizes all training samples equally. In order to selectively penalize samples, we introduce a weight β_i for each sample [105]. A small β_i value would allow for a large ξ_i value and the effect of the sample would be de-emphasized. Conversely, a large β_i value would force ξ_i to be smaller and therefore the effect of the particular sample would be emphasized. The weighted SVM learning can be formulated as the following convex quadratic programming problem:

$$\begin{aligned} \underset{w,\xi_i,b}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\beta_i\xi_i \\ \text{subject to} \quad & y_i(\langle\, w, \Phi(x_i) + b)\rangle \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0. \qquad\qquad\qquad\qquad i = 1, \ldots, n \end{aligned} \tag{4.1}$$

Then the dual formulation of the problem take the form:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C\beta_i, i = 1, \ldots, n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0. \end{aligned} \tag{4.2}$$

In order to decompose the centralized SVM classification problem into sub problems, define a set of $\mathcal{M} := \{1, 2, \ldots, M\}$ distributed compute units with access to different subsets, S_i, $i \in \mathcal{M}$, of the labeled training data such that $S = \bigcup_{i=1}^{M} S_i$. Given this partition, define the vectors $\{\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(M)}\}$ with the i^{th} vector having a size of N_i. In order to devise a distributed algorithm, the SVM optimization problem is relaxed by substituting

the constraint $\sum_{i=1}^{n} \alpha_i y_i = 0$ by a penalty function $0.5MZ(\sum_{i=1}^{n} \alpha_i y_i)$, where $Z > 0$. Thus, the following constrained optimization problem is obtained:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & F(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ & - \frac{MZ}{2} \Big(\sum_{l=1}^{n} \alpha_l y_l \Big) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C\beta_i, i = 1, \ldots, n \end{aligned} \tag{4.3}$$

Note that the objective function $F(\alpha)$ is strictly concave in all its arguments and the constraint set is convex, compact and nonempty.

The convex optimization problem defined in Equation (4.3) is next partitioned into M sub-problems through Lagrangian decomposition. Therefore, the e^{th} unit's optimization problem becomes

$$\begin{aligned} \underset{\alpha_i^{(e)} \in [0, C\beta_i]}{\text{maximize}} \quad & F_e(\alpha) = \sum_{i \in S_e} \alpha_i - \frac{1}{2} \sum_{i \in S_e} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ & - \frac{MZ}{2} \Big(\sum_{i \in S_e} \alpha_i y_i \Big). \end{aligned} \tag{4.4}$$

The individual optimization problems of the units are interdependent. Therefore, they cannot be solved individually without exchanging information between all the processing units. We use a fusion center that collects and distributes support vectors (SVs) among the individual processing units similar to the work of Alpcan and Bauckhage [32].

4.3.2 Theory of Local Intrinsic Dimensionality (LID)

We present here the procedure used to obtain the weights β_i including the novel LID approximation that we introduce.

In the theory of intrinsic dimensionality, classical expansion models measure the rate of growth in the number of data samples encountered as the distance from the sample of interest increases [20]. As an example, in Euclidean space, the volume of an m-dimensional ball grows proportionally to r^m, when its size is scaled by a factor of r. From this rate of change of volume w.r.t distance, the expansion dimension m can be deduced

as:

$$\frac{V_2}{V_1} = \left(\frac{r_2}{r_1}\right)^m \Rightarrow m = \frac{\ln\left(V_2/V_1\right)}{\ln\left(r_2/r_1\right)}. \tag{4.5}$$

Expansion models of dimensionality have previously been successfully employed in a wide range of applications, such as manifold learning, dimension reduction, similarity search and anomaly detection [20, 106]. *In this chapter, we use LID to characterize the intrinsic dimensionality of regions where attacked samples lie, and create a weighting mechanism that de-emphasizes the effect of samples that have a high likelihood of being adversarial examples.* Refer to [20] for more details concerning the theory of LID. Transferring the concept of expansion dimension to the statistical setting of continuous distance distributions leads to the formal definition of LID [20] is given below.

Definition 4.1. *(Local Intrinsic Dimensionality).*
Given a data sample $x \in X$, let $R > 0$ be a random variable denoting the distance from x to other data samples. If the cumulative distribution function $F(r)$ of R is positive and continuously differentiable at distance $r > 0$, the LID of x at distance r is given by:

$$LID_F(r) \triangleq \lim_{\epsilon \to 0} \frac{ln\big(F((1+\epsilon)\cdot r)/F(r)\big)}{ln(1+\epsilon)} = \frac{r \cdot F'(r)}{F(r)}, \tag{4.6}$$

whenever the limit exists.

The last equality of Equation (4.6) follows by applying L'Hôpital's rule to the limits [20]. The local intrinsic dimension at x is in turn defined as the limit, when the radius r tends to zero:

$$\mathrm{LID}_F = \lim_{r \to 0} \mathrm{LID}_F(r). \tag{4.7}$$

LID_F describes the relative rate at which its cumulative distribution function $F(r)$ increases as the distance r increases from 0. In the ideal case where the data in the vicinity of x is distributed uniformly within a subspace, LID_F equals the dimension of the subspace; however, in practice these distributions are not ideal, the manifold model of data does not perfectly apply, and LID_F is not an integer [100]. Nevertheless, the local intrinsic dimensionality is an indicator of the dimension of the subspace containing x that would best fit the data distribution in the vicinity of x.

Estimation of LID

Given a reference sample $x \sim \mathcal{P}$, where $\mathcal{P}$ represents the data distribution, the Maximum Likelihood Estimator of the LID at x is defined as follows [106]:

$$\widehat{\mathrm{LID}}(x) = -\left(\frac{1}{k} \sum_{i=1}^{k} \log \frac{r_i(x)}{r_{\max}(x)} \right)^{-1}. \tag{4.8}$$

Here, $r_i(x)$ denotes the distance between x and its i-th nearest neighbor within a sample of k points drawn from $\mathcal{P}$, and $r_{\max}(x)$ is the maximum of the neighbor distances. The above estimation assumes that samples are drawn from a tight neighborhood, in line with its development from Extreme Value Theory. In practice, the sample set is drawn uniformly from the available training data (omitting x itself), which itself is presumed to have been randomly drawn from $\mathcal{P}$. Note that the LID defined in Equation (4.7) is the theoretical calculation, and that $\widehat{\mathrm{LID}}$ defined in Equation (4.8) is its estimate.

4.3.3 LID Calculation

When the training data X is large, computing neighborhoods with respect to the entire dataset X can be prohibitively expensive. The LID value of a sample x can be estimated from its k-nearest neighbor set within a randomly-selected sample (*mini-batch sampling*) of the dataset X [100]. Given that the mini-batch is chosen sufficiently large so as to ensure that the k-nearest neighbor sets remain in the vicinity of x, estimates of LID computed for x within the mini-batch would resemble those computed within the full dataset X. Therefore, in order to reduce the computational cost, we use *mini-batch sampling* in our experiments.

Calculating LID w.r.t. labels

As shown in Section 4.3.2, LID is usually concerned with the data X and is independent of the corresponding labels y. Since only the labels are maliciously flipped in a label flipping attack, the LID values of X, before and after the attack would remain the same. Therefore, we devise three label dependent LID variations as follows.

Take $S := (X, y)$ to be the full training data set with labels. Then, define $S^j := (X^j, y^j)$, where $j = \{\pm 1\}$ as the set of data samples that carry the same label j.

- **In-class LID**: For each $x_l \in X^j$, the LID is calculated w.r.t. $x_{h \neq l} \in X^j$, for $j = \{\pm 1\}$. In-class LID of a particular sample gives the dimension of the subspace w.r.t. the data distribution of the same class.

- **Out-class LID**: For each $x_l \in X^j$ the LID is calculated w.r.t. $\{x_h \in X \mid x_h \notin X^j\}$, for $j = \{\pm 1\}$. Out-class LID gives the dimension of the subspace in which a particular sample lies w.r.t. the data distribution of the opposite class.

- **Cross-class LID**: Define cross-class LID as the ratio between the in-class LID and the out-class LID.

In our experiments we use *cross-class LID* to highlight samples that likely to be adversarial samples.

Kernel LID calculation

For the above label dependent LID variations to give distinguishable LID distributions for attacked and non-attacked samples, the data clouds from the two classes need to be physically separated. LID in its original form is calculated using the Euclidean distance (although the underlying distance measure does not necessarily have to be Euclidean [100]) between samples in the input space $x \in \mathcal{R}^d$. As non-linearly separable datasets have data from both classes inter-weaved, the aforementioned LID values of attacked samples and non-attacked samples would have overlapping distributions.

To have two distinguishable LID distributions for attacked and non-attacked samples, X needs to be non-linearly transformed to a high dimensional space where the data from two classes are well separated. This approach is similar to the SVM training procedure, where the data is transformed to higher dimensional spaces where the data becomes linearly separable. In SVMs, instead of explicitly transforming data to the high dimensional space and calculating the dot product between samples, the kernel trick is used to implicitly obtain the dot products (i.e., $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$). While it is feasible when using

certain kernels to obtain the transformed $\Phi(x)$ values, for kernels such as the *radial basis function kernel* (RBF) it is not possible, and only the dot product can be obtained. Thus, calculating the LID in the hypothesis space using Euclidean distance is not possible.

The RBF kernel function non-linearly transforms the squared Euclidean distance between two data points x and x' as $K(x, x') = \exp(-\gamma\|x - x'\|^2)$ in order to obtain the corresponding kernel matrix entry. The RBF kernel function returns 1 for identical data points and 0 for dissimilar data points. As the theory of LID computes similarity using a distance function (4.8), we use the reciprocal of the kernel value and propose the following variation of LID called Kernel LID (*K-LID*).

$$\widehat{\text{K-LID}}(x) = -\left(\frac{1}{k}\sum_{i=1}^{k}\log\frac{t_i(x)}{t_{\max}(x)}\right)^{-1}. \tag{4.9}$$

Here, $t_i(x)$ denotes the distance between x and its i-th nearest neighbor calculated as $t_i(x) = (1/K(x, x_i)) - 1$ and $t_{\max}(x)$ denotes the maximum distance among the neighbor sample points. In the remainder of this chapter, we will use Equation (4.9) to estimate K-LID values.

The following theorem shows the relationship between K-LID values and LID values calculated using Euclidean distance in the input space. Define Δ as a Euclidean distance measure. Then, following the above transformation of distance, t can be written as $t = \frac{1}{\exp^{(-\gamma\Delta^2)}} - 1$.

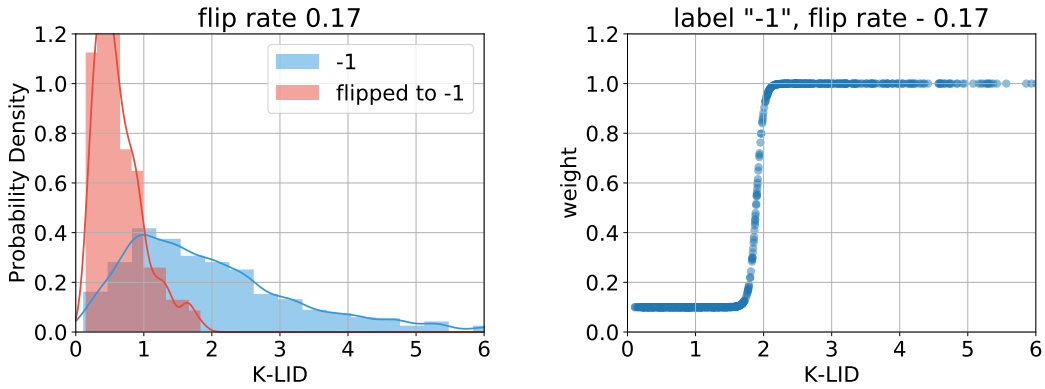
Theorem 4.1: The LID calculated using Euclidean distance in input space (i.e., LID_Δ) is proportional to the K-LID calculated using transformed distances (i.e., K-LID_t) as,

$$\text{LID}_\Delta = \frac{2\gamma\Delta^2\exp^{(\gamma\Delta^2)}}{\exp^{(\gamma\Delta^2)} - 1} \times \text{K-LID}_t. \tag{4.10}$$

Proof: The proof immediately follows from Theorem 3 of [107].

4.3.4 K-LID Distributions of Attacked and Non-Attacked Samples

We discuss here the intuition behind using K-LID to identify adversarial samples during training. Flipping the label of a data sample would give it a different class assignment



(a) K-LID pdf of MNIST.

(b) Weight assignment.

Figure 4.3: The probability density functions and the corresponding weight distributions when different γ values are used in the K-LID calculation.

from most of its close neighbors. Computing K-LID estimates with respect to its neighborhood from within those samples that share the same class assignment (i.e., In-class LID) would then reveal an anomalous distribution of the local distance to these neighbors. Similarly, the out-class LID (K-LID calculated w.r.t. samples that have the opposite class assignment) would also give different distributions for flipped and non-flipped samples. Consequently, the Cross-class LID, which combines the distinguishing powers of In-class LID and Out-class LID, would have the power to distinguish flipped samples from non-flipped samples.

Under input perturbation attacks, the perturbed samples would be in close proximity to other data samples that share the same class assignment, yet away from that data distribution by a moderate amount. Therefore, perturbed samples would have an anomalous distribution of the local distance to these neighbors, and would be highlighted by their cross-class K-LID values.

To build an SVM classifier resilient against label flipping attacks, we require the K-LID estimates of non-attacked samples and attacked samples to have distinguishing distributions. The two distributions can be obtained by simulating an attack and deliberately altering a subset of labels/data during training, by assuming the distributions based on domain knowledge or prior experience or by having an expert identify attacked and non-attacked samples in a subset of the dataset. Note that the learner *does not* need to be

aware of the type of attack being used, it only needs the K-LID distributions of attacked and non-attacked samples.

Through our experiments, we aim to demonstrate the distinguishing capability of the novel K-LID that we introduced. To that end, we use the following procedure to obtain the K-LID distributions of attacked and non-attacked samples while restating that any of the aforementioned procedures can be used well.

First, to identify a suitable hypothesis space that separates the data of two classes, the learner performs an exhaustive search for the ideal RBF gamma parameter (γ^*) by calculating the cross-class K-LID values for the training dataset under different γ^* values. *Note that this γ^* is used for obtaining K-LID values and is independent of the SVM training process*. The learner estimates the density functions of K-LID estimates for non-attacked samples and attacked samples of each class under each γ^* value using kernel density estimation methods [108]. Subsequently, the learner uses the Kullback-Leibler divergence (or colloquially KL distance) between the K-LID densities of non-attacked and attacked samples as the measure of goodness of each transformation and selects the γ^* value that yields the highest KL distance. The Kullback-Leibler divergence calculates the statistical distance between two probability distributions P and Q as $D_{KL}(P||Q) = \sum_i P(i) \log(P(i)/Q(i))$.

As Figure 4.3a shows, K-LID is a powerful metric that can give two distinguishable distributions for attacked and non-attacked samples. Although, ideally we like to see no overlap between the two distributions, in real-world datasets we see some percentage of overlap. Figure 4.3b shows the weight assignment function which gives weights to samples based on their K-LID values. In the following section, we explain how the weight assignment function is derived from the two K-LID distributions shown in Figure 4.3a.

Sample Weighting Scheme

Define p_n^j and p_f^j as the probability density functions of the K-LID values of non-attacked samples and attacked samples, respectively, for the two classes $j = \{\pm 1\}$. We assume there are two possible hypotheses, H_n^j and H_f^j. For a given data sample x' with the K-LID

estimate K-LID(x'). We may write the two hypotheses for $j = \{\pm 1\}$ as

$$\begin{aligned} H_n^j &: \text{K-LID}(x') \sim p_n^j, \\ H_f^j &: \text{K-LID}(x') \sim p_f^j, \end{aligned} \tag{4.11}$$

where the notation "K-LID(x') $\sim p$" denotes the condition "K-LID(x') is from the distribution p". To clarify, H_n^j is the hypothesis that K-LID(x') is from the distribution p_n^j and H_f^j is the hypothesis that K-LID(x') is from the distribution p_f^j. The *likelihood ratio* (LR) is usually used in statistics to compare the goodness of fit of two statistical models. For example, the likelihood ratio of a data sample x' with the K-LID estimate K-LID(x') is defined as

$$\Lambda^j(\text{K-LID}(x')) = p_n^j\big(\text{K-LID}(x')\big) / p_f^j\big(\text{K-LID}(x')\big). \tag{4.12}$$

$p(\text{K-LID}(x'))$ denotes the probability value corresponding to the K-LID of sample x from the probability distribution p. To clarify, $\Lambda^j(\text{K-LID}(x'))$ expresses how many times more likely it is that K-LID(x′) is under the K-LID distribution of non-attacked samples than the K-LID distribution of attacked samples. As the likelihood ratio value is unbounded above, we clip the LR values of all the samples at some dataset dependent threshold after obtaining the LR values for the entire dataset.

As there is a high possibility for p_n^j and p_f^j to have an overlapping region, there is a risk of overemphasizing (giving a higher weight to) attacked samples. To mitigate that risk, we only de-emphasize samples that are suspected to be attacked (i.e., low LR values). Therefore, we transform the LR values such that $\Lambda^j(\text{K-LID}(x)) \in (0.1, 1)$ for $j = \{\pm 1\}$. We set the lower bound for the weights to an arbitrary value close to 0 (0.1 in this case) as the weighted SVM would simply disregard any training points that carry the weight 0. The upper bound is set to 1 to prevent overemphasizing samples. Subsequently, we fit a hyperbolic tangent function to the transformed LR values in the form of $0.55 - 0.45 \times \tanh(az - b)$ and obtain suitable values for the parameters a and b. The scalar values 0.45 and 0.55 maintain the scale and vertical position of the function between 0.1 and 1. Finally, we use the K-LID value of each x_i and use Equation (4.13) to obtain the

Algorithm 2 LID-SVM Defense Algorithm

input $\tilde{S} = (X, \tilde{y})$ ▷ contaminated training data
output SVM weights β
for $j = \{+1, -1\}$ **do** ▷ for each class
    initialize K-LID$_f$, K-LID$_n$, Λ^j, β^j
    $\gamma^* \leftarrow$ search for gamma that best separates the two classes
    K-LID$_f \leftarrow$ k-lid-calc$(X, \tilde{y}, \gamma^*)$ ▷ k-lid of flipped data
    K-LID$_n \leftarrow$ k-lid-calc$(X, \tilde{y}, \gamma^*)$ ▷ k-lid of non-flipped data
    $\Lambda^j \leftarrow$ calculate likelihood ratio for each K-LID value
    fit function $g(z) = 0.55 - 0.45 \times \tanh(az - b)$ to Λ^j
    $\beta^j \leftarrow g(\text{K-LID}_X)$ ▷ obtain weight of $x_i \in X$ with $\tilde{y}_i = j$ using the K-LID of x
end for
$\beta \leftarrow \beta^{+1}$ and β^{-1}
LID-SVM = train classifier using$(X, \tilde{y}, \beta)$

corresponding weight β_i.

$$\beta_i = 0.55 - 0.45 \times \tanh(a\text{K-LID}(x_i) - b). \tag{4.13}$$

The high level procedure used to construct the LID-SVM under a label flipping attack is formalized in Algorithm 2.

4.4 Experimental Results and Discussion

This section describes how the datasets are obtained, pre-processed and other procedures of the experimental setup. We extensively investigate how the performance of LID-SVM holds against an increasing fraction of attacked training data, for each of the proposed attacks described in Section 4.2. Our code is available at `https://github.com/sandamal/lid-svm`.

4.4.1 Experimental Setup

Benchmark Datasets

We evaluate the effectiveness of LID-SVM on four real-world datasets used in [46]: Acoustic, Ijcnn1, Seismic and Splice as well as MNIST. Note that the high computational complexities of the attacks make it infeasible to be performed on larger datasets. We report

Table 4.1: Datasets used for training and testing.

Dataset	Training size	Test size	C	γ
MNIST	1,500	500	1.47	0.0197
Acoustic	500	500	1,024	0.0078
Ijcnn1	500	500	64	0.1200
Seismic	500	500	1,024	0.0078
Splice	500	500	1,024	0.0078
OMNeT	364	91	0.3969	0.7937

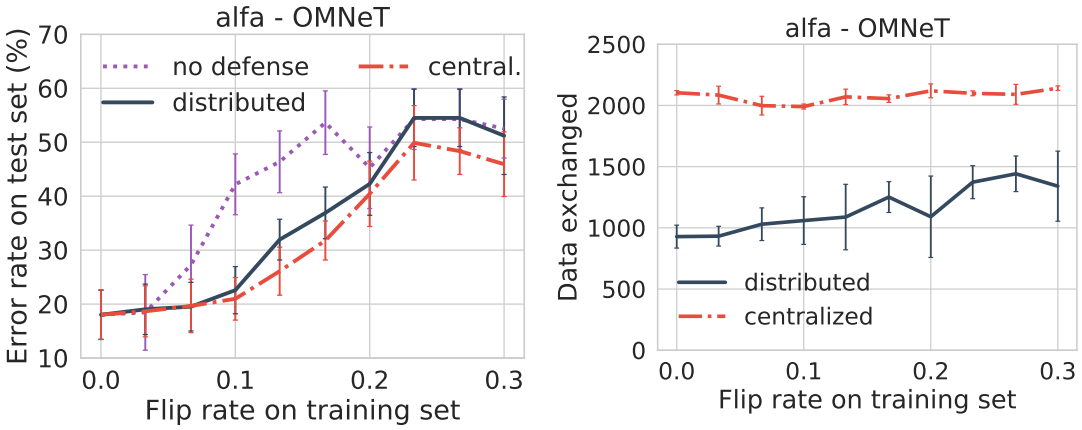
the performance of LID-SVM using the error rate (i.e., percentage of samples wrongly classified) on a separate test set, using 5-fold cross-validation. The average error rates are reported as the attack rate is increased from 0% to 30%. Table 4.1 gives the chosen parameters and the number of samples used in each training and test set.

For each dataset we compute the SVM hyper-parameters (i.e., C and γ) using a 5-fold cross-validation process. Kernel density estimation was performed using a Gaussian kernel with bandwidth set to 0.15 (found using a cross-validation approach to avoid overfitting). The effects of kernel density estimation and clipping of the LR values at a heuristic threshold are neutralized by the smooth function fitting (Equation (4.13)), therefore the algorithm is robust to these choices. For K-LID calculations, we tune k over the range $[10, 100)$ for a minibatch size of 100, following Ma et al. [100].

4.4.2 Results

Distributed detection vs. centralized detection

As SDRs have limited power and range, each SDR captures only the transmissions of nodes within its range. The objective of the cognitive network is for each SDR to have the ability to classify a new transmission source x as a rogue agent or a civilian in a distributed manner without communicating x itself to the other nodes. But to obtain a classifier that can identify all the types of transmitters in the given area, the information obtained by each listener would need to communicated to all the other SDR listeners during the training phase. This can be achieved in one of two ways: (i) have a fusion cen-



(a) Test error on OMNeT data.

(b) Data exchanged.

Figure 4.4: The classification performance of centralized LID-SVM and distributed LID-SVM and the amount of data exchanged in each case.

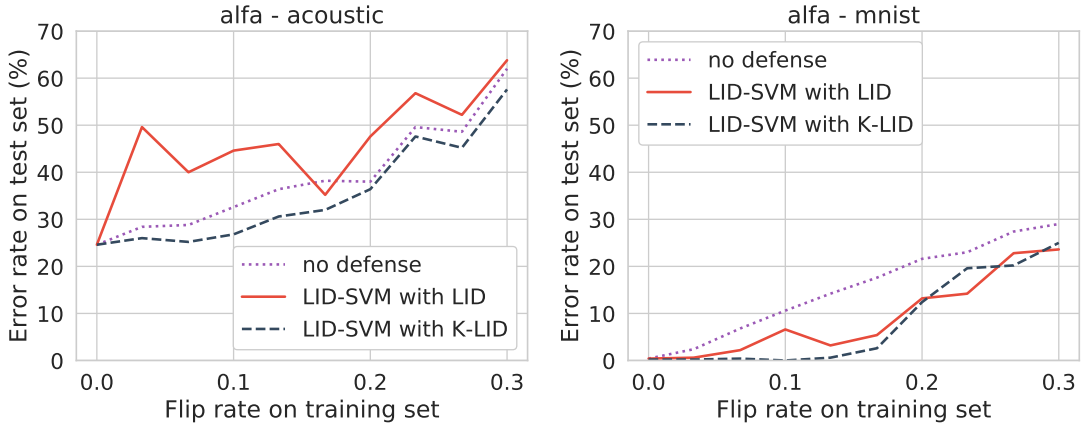
ter collect all the data from the listeners and train one classification model, and transfer the learned model back to the listeners (i.e., centralized solution), or (ii) have the listener SDRs (which have computing capabilities) compute classification models based on their local data, exchange the learned models with other listeners through one or more fusion centers followed by an update to their local models. For simplicity, we only consider the data exchange between the SDRs and the fusion centers (i.e., can be modeled as a distributed SVM with one fusion center). While having more than one fusion center improves the robustness of the overall system, it would increase the communication overhead due to the information exchange between the fusion centers.

Figure 4.4a shows the error rates on the test set when rogue agents carry out alfa attacks on the distributed SDR listeners with the flip rate increasing from 0% to 30%. We observe that the centralized SVM solution has lower error rates compared to the DSVM solution on average. We postulate that this is due to two main reasons, (i) the optimization problem used by the DSVM (Equation (4.3)) is a relaxation of the optimization problem of the centralized SVM (Equation (4.2)), and (ii) as shown by Amsaleg et al. [106], the MLE estimator of LID (Equation (4.8)) is not stable on small mini-batch sizes. In the DSVM setting, each SVM node trains on the data that it receives from civilians and rogue agents within its listening range, resulting in smaller dataset sizes. Therefore, the

resulting LID estimations would also be affected compared to the centralized learner.

Although the detection capability of the DSVM is less than the centralized SVM, we observe that its information exchange overhead is significantly less compared to the centralized SVM in this particular scenario. Figure 4.4b shows the number of data points exchanged at each flip rate for the two SVM solutions. In our experiment, we consider $M = 5$ listeners, therefore in the centralized SVM, the listeners would first transfer their data points to the central SVM, and the SVM would transfer the SVs back to all five nodes after the SVM training process. Afterwards, each node would be able to evaluate new data samples using the received SVs (note that for simplicity we are not considering the overhead of transmitting the α values). In the DSVM, each SVM node would use a random vector of α values to initialize the training process. Subsequently, after each training iteration, they would transfer their SVs to a central fusion node, which would distribute them among the other SVM nodes. Note that the DSVM iteratively trains only until the detection accuracy converges.

As Figure 4.4b shows, the DSVM can reduce the information exchange overhead by 44.4% on average with only a 3.07% reduction in average detection accuracy. These findings suggest that detection using a DSVM solution is indeed the better option for the SDR based detection network.



(a) LID vs K-LID on acoustic.

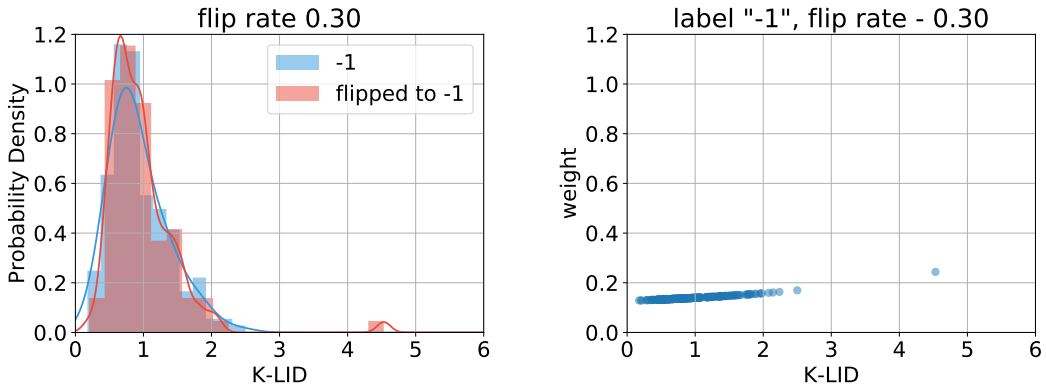
(b) LID vs K-LID on MNIST.

Figure 4.5: The classification performance of LID based LID-SVM and K-LID based LID-SVM on two datasets.

Performance of K-LID vs. Conventional LID

Figure 4.5 depicts the classification performance of K-LID based LID-SVM vs. conventional LID based LID-SVM. As expected, K-LID based LID-SVM has lower error rates on average. If data from the two classes are not linearly separable in the input space, conventional LID fails to give two distinguishable LID distributions with a low percentage of overlap. When the percentage of overlap is high, the defense algorithm assigns a low, uniform weight to almost all samples. Such a weight assignment would make the LID-SVM equivalent to an SVM with no defense (with a sub-optimal C value). Therefore, as seen in Figure 4.5a, LID based LID-SVM could perform worse than an SVM with no defense in some data sets. However, K-LID, which performs the LID calculation in higher dimensional transformed spaces where the data is separated, has sufficient separation power to obtain two distinguishable distributions with a low percentage of overlap.

When using the kernel based distance function, if the γ used is large, two data points are considered similar only if they are close to each other in the transformed space, conversely, if γ is small, even data points that are farther away from each other in the transformed space are considered similar. Therefore different γ values give distinct K-LID distributions with different percentages of overlap. When there is a high percentage of overlap it is not possible to obtain meaningful weights.



(a) pdf with 0.3 flip rate.

(b) weights with 0.3 flip rate.

Figure 4.6: The probability density function and the corresponding weight distribution at 30% flip rate.

K-LID Distributions as the Attack Rate Increases

Figure 4.6 shows the K-LID distributions of flipped and non-flipped samples in the Splice dataset when the flip rate is 30%. Having a flip rate of 30% assumes a very powerful attacker with significant influence. At high flip rates, the two distributions have a high overlap percentage and the likelihood ratio cannot be used to distinguish flipped samples from non-flipped ones, therefore the learner assigns a low, uniform weight to most samples. We believe that this increase in overlap percentage is the main reason why LID-SVM tends to have increasing error rates as the flip rate increases. As explained in Section 4.3.3, we estimate the LID using *mini-batch sampling*. For a flipped sample x_i, we speculate that as the flip rate increases, the probability of having other flipped samples within the randomly-selected subset increases and the LID calculation is influenced by other flipped samples that carry the same label y_i. A comprehensive investigation of the trade-offs among mini-batch size, LID estimation accuracy, and detection performance is an interesting direction for future work.

Under Label Flipping Attacks

We compare the performance of LID-SVM against LS-SVM [18] and LN-SVM [45] which have been shown to be effective against label flipping attacks. Table 4.2 gives the error rate of each defense mechanism averaged over all the flip rates considered (0% - 30%). Due to space limitations, graphs of only some experiments are shown.

Random label flips: The performance of the binary SVM without a defense against random label flips varies from dataset to dataset. We observe that it can retain near 1% error rate on MNIST, a mere 5% increase in error rate on Acoustic, and a 6% increase on OMNeT when the flip rate is increased up to 30%. On Ijcnn1, Seismic and Splice however there is a 16%, 10% and 18% increase in the error rates respectively. LID-SVM outperforms the other defense mechanisms in all of the datasets except for Seismic where LS-SVM has a 1.58% lower average error rate. On the other datasets however LID-SVM has lower average error rates (up to 12%) than the other two defenses. Figure 4.7 depicts how the error rates vary on MNIST, Ijcnn1, and OMNeT when the training flip rate

Table 4.2: Average error rates across all the flip rates of each defense algorithm against the five attacks considered. The best results are indicated in **bold** font.

	Dataset	SVM	LID-SVM	LS-SVM	LN-SVM
random	MNIST	0.64	**0.62**	12.83	5.17
	Acoustic	27.80	**26.35**	28.14	28.58
	Ijcnn1	16.79	**15.51**	16.36	19.60
	Seismic	25.45	24.79	**23.21**	36.97
	Splice	24.77	**20.28**	24.07	30.42
	OMNeT	29.19	**24.10**	27.85	31.08
farfirst	MNIST	12.93	**3.08**	16.72	18.54
	Acoustic	39.22	**35.01**	39.96	42.14
	Ijcnn1	34.42	29.91	**22.49**	38.21
	Seismic	33.16	**31.20**	31.48	35.00
	Splice	31.60	**28.29**	31.10	33.21
	OMNeT	36.75	32.10	**26.18**	34.88
nearest	MNIST	6.73	**1.56**	10.25	7.18
	Acoustic	26.76	25.94	26.69	**25.23**
	Ijcnn1	13.90	**12.38**	14.66	18.54
	Seismic	20.68	19.82	**18.97**	45.62
	Splice	22.79	**21.30**	22.50	39.11
	OMNeT	30.64	**26.42**	31.08	37.56
alfa	MNIST	14.34	**8.79**	16.76	17.00
	Acoustic	39.94	**36.87**	40.25	42.23
	Ijcnn1	32.02	29.01	**23.96**	35.08
	Seismic	30.85	**29.60**	30.31	33.42
	Splice	29.90	**26.80**	29.40	34.64
	OMNeT	41.23	31.98	**25.54**	33.41
alfa-tilt	MNIST	17.13	**6.64**	16.76	17.00
	Acoustic	43.87	**40.72**	43.78	43.03
	Ijcnn1	32.60	27.05	**26.00**	36.47
	Seismic	33.93	**30.26**	31.90	46.21
	Splice	30.47	**26.64**	30.58	43.28
	OMNeT	42.79	37.68	**22.81**	33.41

increases from 0% to 30%.

Naive adversarial label flips: We consider *farfirst* and *nearest* as naive attacks as the algorithms are relatively simpler compared to *alfa* and *alfa-tilt*. Although *farfirst* is simple, it can have a significant impact on an undefended SVM with error rates increasing by 30%

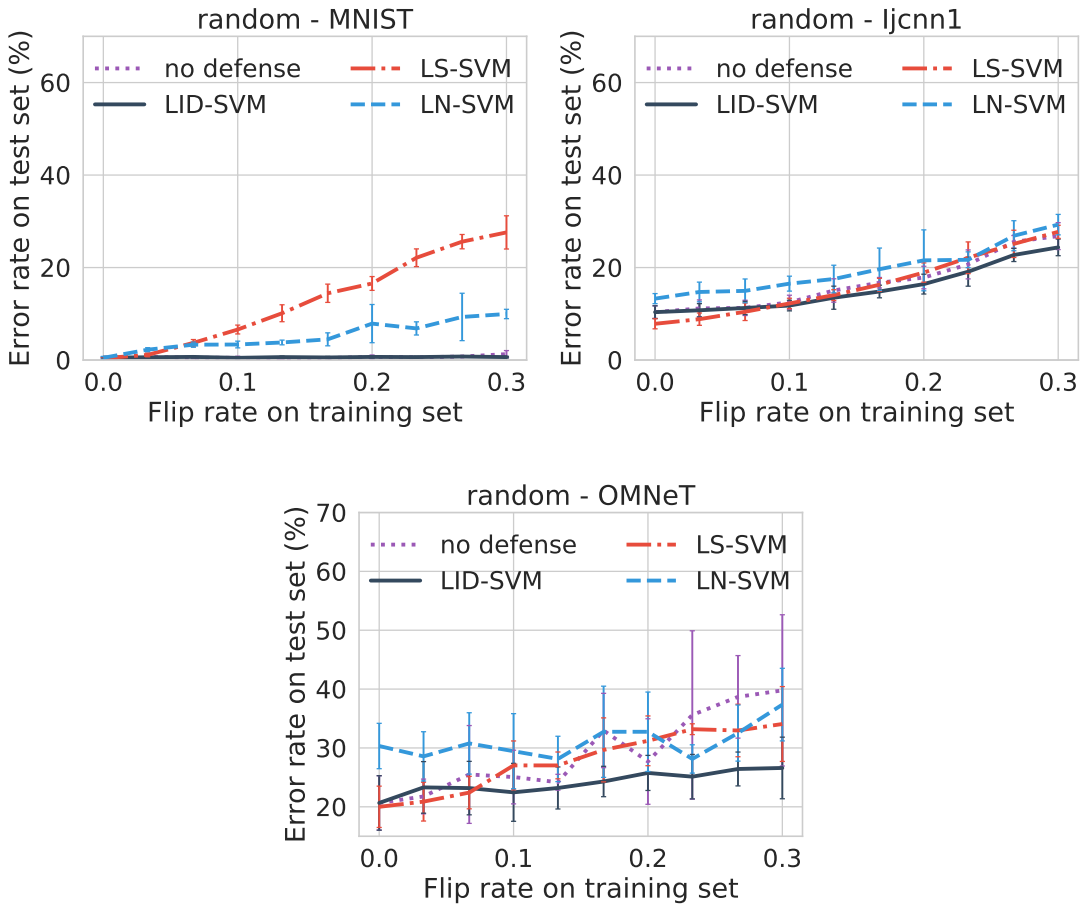


Figure 4.7: The average error rates of SVM, LID-SVM, LS-SVM, and LN-SVM ($\mu = 0.15$) against random label flip attacks on MNIST, Ijcnn1, and OMNeT when the training flip rate increases from 0% to 30%.

on MNIST, 13% on OMNeT, 29% on Acoustic, 48% on Ijcnn1, 27% on Seismic and 34% on Splice when the flip rate is increased to 30%. In *nearest*, the increase in error rates are 18% on MNIST, 13% on OMNeT, 3% on Acoustic, 8% on Ijcnn1, 1.4% on sesmic and 12% on Splice. In *farfirst*, the LID-SVM outperforms the other defenses in all scenarios except Ijcnn1 and OMNeT, where LS-SVM has 7.4% and 5.9%lower average error rates respectively. In *nearest*, LS-SVM outperforms LID-SVM on Seismic by 0.85% and LN-SVM outperforms LID-SVM on Acoustic by 0.7%. In all the other test datasets we observe that LID-SVM outperforms the other defenses by large margins (up to 15% against *farfirst* and up to 25% against *nearest*). Figure 4.8 shows how the error rates vary on MNIST, Acoustic,

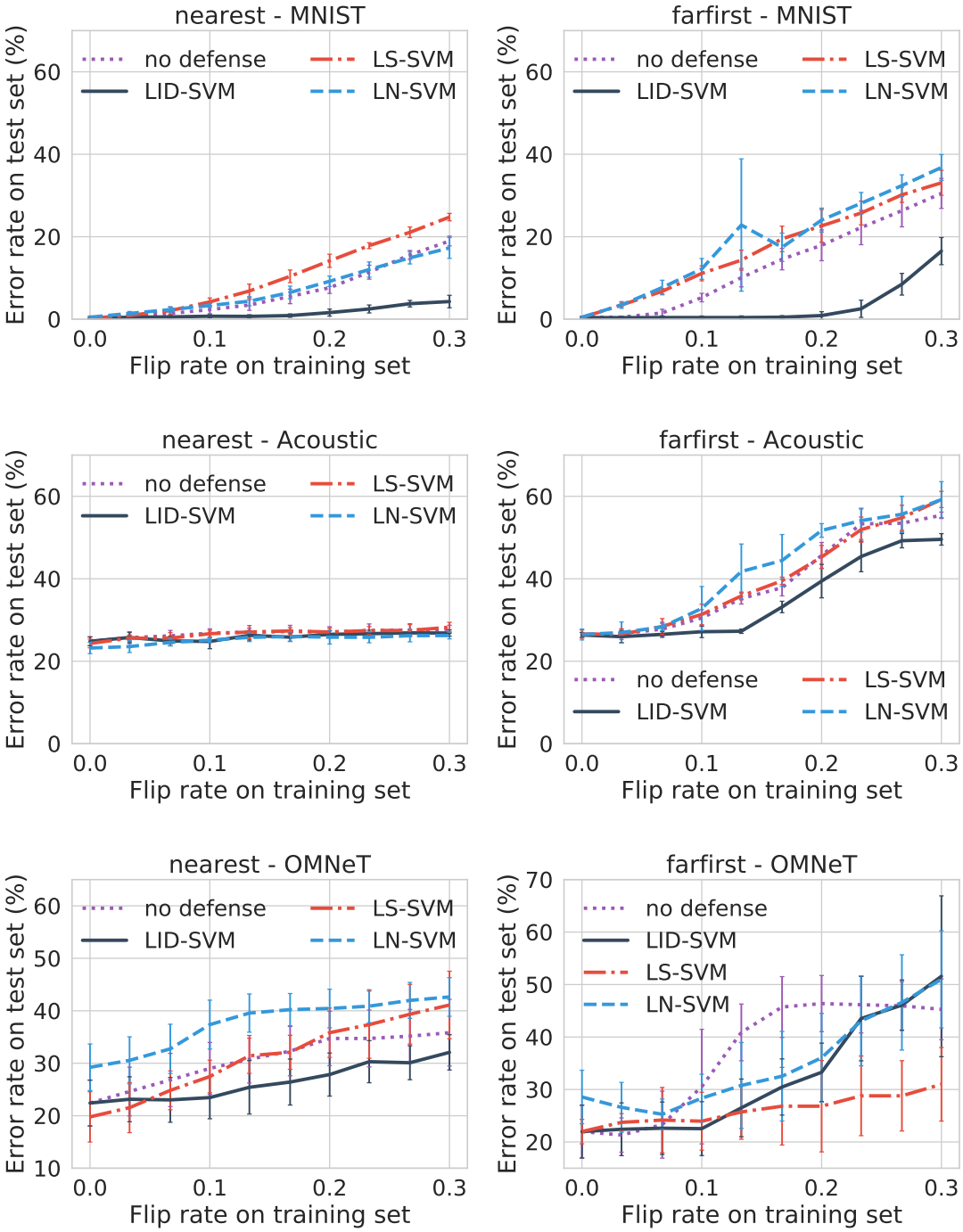


Figure 4.8: The average error rates of SVM, LID-SVM, LS-SVM, and LN-SVM ($\mu = 0.15$) against naive label flip attacks on MNIST, Acoustic, and OMNeT when the training flip rate increases from 0% to 30%.

and OMNeT when the training flip rate increases from 0% to 30%.

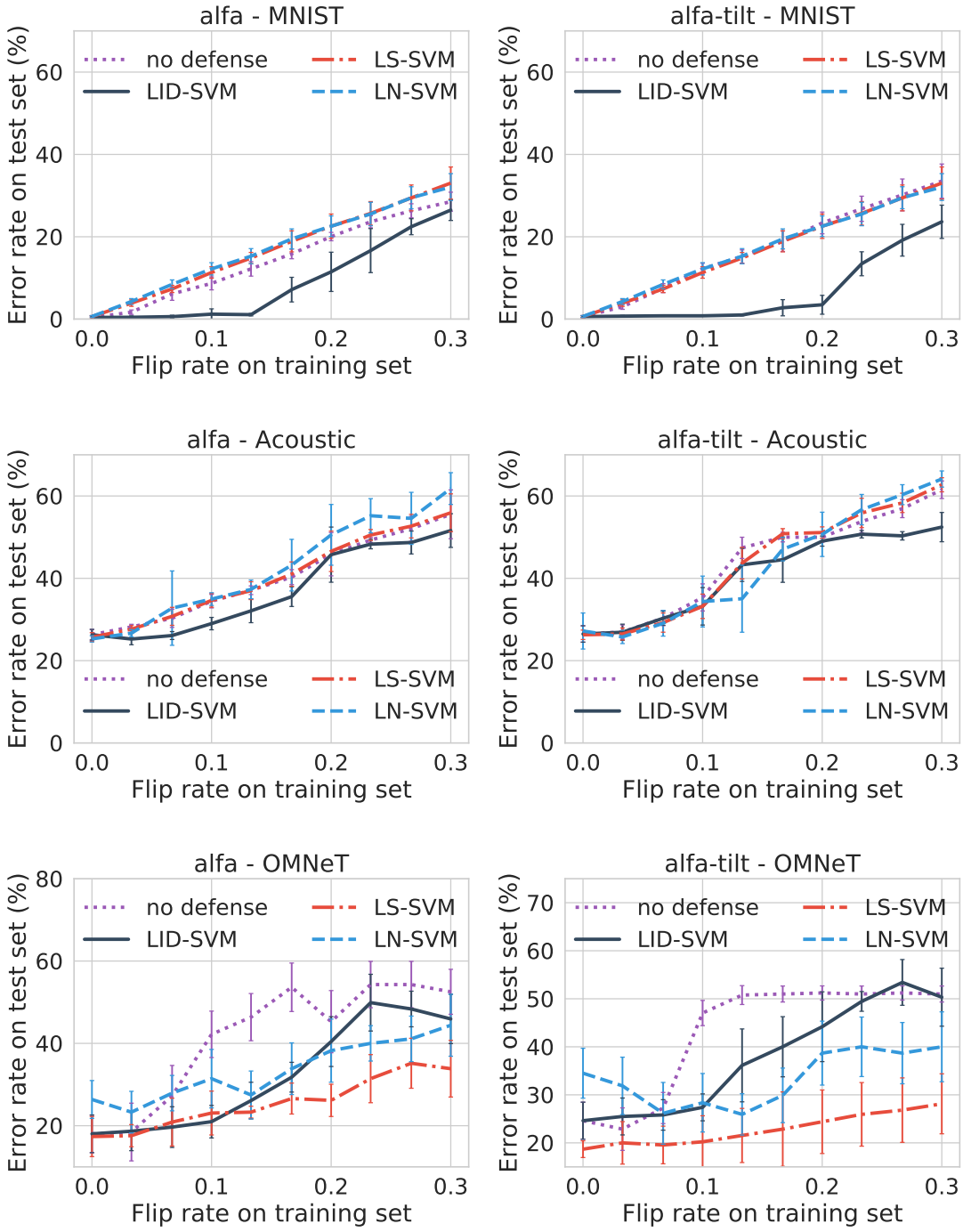


Figure 4.9: The average error rates of SVM, LID-SVM, LS-SVM and LN-SVM ($\mu = 0.15$) against sophisticated label flip attacks on MNIST, Acoustic, and OMNeT when the training flip rate increases from 0% to 30%.

Sophisticated adversarial label flips: Carefully selected adversarial label flips have a significant impact on the performance of SVMs. Against *alfa*, the error rates go up by 28% on MNIST, 34% on OMNeT, 29% on Acoustic, 40% on Ijcnn1, 24% on Seismic and 28% on Splice when the flip rate is increased from 0% to 30%. Against *alfa-tilt*, the error rate increases are 33% on MNIST, 35% on OMNeT, 35% on Acoustic, 39% on Ijcnn1, 33% on Seismic and 27% on Splice. Under alfa and alfa-tilt attacks, LS-SVM outperforms LID-SVM on the Ijcnn1 and dataset with 5.05% and 1.05% lower error rates respectively. Similarly, LS-SVM has 1% and 15% lower error rates on the OMNeT dataset under the same attacks. On all other datasets, the LID-SVM can significantly reduce the error rates compared to the other two defenses against all attack strategies. We observe that, on average, LID-SVM can achieve 1.3% to 10.5% lower error rates compared to the undefended SVM, 1.4% to 16.7% lower error rates compared to LN-SVM and up to 10.1% lower error rates compared to LS-SVM. Figure 4.9 shows the impact of *alfa* and *alfa-tilt* on three datasets.

Under Input Perturbation Attacks

We compare the performance of LID-SVM against LS-SVM [18] and CURIE [49] under input perturbation attacks. The two algorithms use different approaches to address the problem of learning under adversarial conditions. Table 4.3 gives the error rate of each defense mechanism averaged over all the poison rates considered (0% - 30%).

Similar to the *random* label flip attack, the performance of the binary SVM without a defense against input perturbation attacks varies from dataset to dataset. On MNIST, we observe that it is able to retain near 1% error rate under all three attack algorithms considered. On the other data sets however we see a considerable impact on the detection accuracy.

Under *PA*, the error rates increase by 22.3% on OMNeT, 3.7% on Acoustic, 10.5% on Ijcnn1, 7.2% on Seismic and 4.7% on Splice when the perturbation rate is increased from 0% to 30%. The LS-SVM outperforms LID-SVM on Ijcnn1 and OMNeT with 3.44% and 1.2% lower average error rates respectively. On the other data sets, however, LID-SVM outperforms the other defense algorithms by up to 2.9%.

Table 4.3: Average error rates across all the attack rates of each defense algorithm against the three attacks considered. The best results are indicated in **bold** font.

	Dataset	SVM	LID-SVM	LS-SVM	CURIE
PA	MNIST	0.68	0.65	**0.60**	0.64
	Acoustic	29.61	**28.12**	28.78	28.24
	Ijcnn1	16.42	15.51	**12.07**	16.36
	Seismic	22.52	**21.23**	22.11	22.33
	Splice	20.94	**18.65**	19.16	21.53
	OMNeT	30.38	29.47	**28.29**	30.44
RA	MNIST	0.84	0.78	**0.57**	0.73
	Acoustic	31.74	29.13	31.16	**28.66**
	Ijcnn1	15.63	12.86	**9.43**	13.80
	Seismic	24.77	**18.24**	18.61	20.14
	Splice	20.39	**17.16**	18.80	20.91
	OMNeT	**18.20**	18.89	19.32	18.86
CG	MNIST	0.68	0.66	**0.56**	0.68
	Acoustic	31.26	**28.42**	30.68	30.41
	Ijcnn1	16.07	**13.66**	15.72	14.99
	Seismic	23.43	**18.08**	19.24	20.49
	Splice	19.76	**16.09**	17.72	18.88
	OMNeT	18.02	18.70	**17.36**	18.90

Under *RA*, we see a 0.2% increase in error rate on MNIST, 0.9% on OMNeT, 14.3% on Acoustic, 10.3% on Ijcnn1, 5.4% on Seismic and 5.8% on Splice when the perturbation rate goes from 0% to 30%. Again we see LS-SVM outperforming LID-SVM on Ijcnn1 with a 3.4% lower average error rate. We also observe CURIE having a 0.47% lower average error rate on Acoustic. On the Seismic and Splice datasets, however, LID-SVM outperforms the other defenses by up to 3.8% lower error rates.

We observe that under *CG*, the LID-SVM can consistently outperform LS-SVM and CURIE with lower average error rates up to 2.8% on all the considered datasets except for OMNeT where LS-SVM has a 1.3% lower average error rate. Figure 4.10 shows the impact of *PA* and *RA* on Seismic, Acoustic, and OMNeT when the training perturbation rate increases from 0% to 30%.

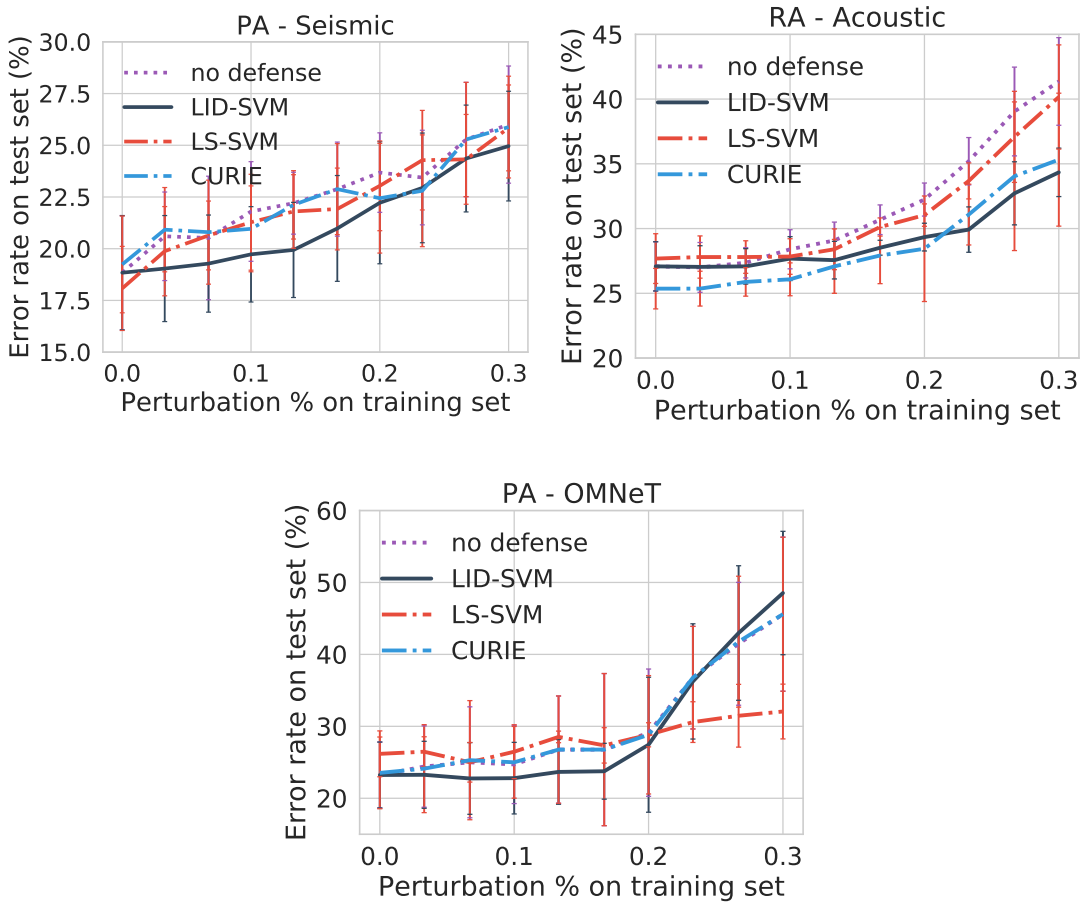


Figure 4.10: The average error rates of SVM, LID-SVM and LS-SVM against input perturbation attacks on three datasets when the attack percentage increases from 0% to 30%.

4.4.3 Discussion

We observe that the adversarial attacks such as *farfirst*, *alfa* and *alfa-tilt* increase the error rates of the tested learners considerably compared to random label flips. This shows that although SVMs may be able to handle label noise in some scenarios by design, they are not immune to adversarial label flip attacks, and by selectively flipping labels, adversaries can significantly increase the error rates. The *nearest* attack, where the labels of data points that are nearest to the separating hyperplane are flipped, has the least impact on the prediction accuracy of learners across all the considered test cases. We speculate that flipped labels near the hyperplane results in less movement/rotation in the margin compared to label flips that are farther away.

From the three input perturbation attacks we have considered in this chapter, *PA* [40] and *CG* [42] are unable to perform simultaneous optimization of multi-point attacks (i.e., collectively perturb data points such that there is a maximal increase in validation error). Furthermore, *PA* [40] attempts to find a reasonably good local maximum of the non-convex validation error surface, which may result in sub optimal attacks. Therefore we see in Table 4.3 that the impact of input perturbation attacks on SVMs is similar to that of *random* and *nearest* label flip attacks.

Although LS-SVM, LN-SVM and CURIE add some resistance to SVMs against training time attacks, LID-SVM is able to consistently reduce error rates across different attack strategies and datasets. LN-SVM and LS-SVM try to address the problem of learning under adversarial attacks by spreading the influence on the decision boundary more evenly across all samples (using heuristics), whereas in LID-SVM we make samples that are suspected to be attacked contribute less. From the extensive experiments conducted, we see that the novel K-LID calculation we introduce has the potential to distinguish attacked samples from non-attacked samples, and thereby subdue the malicious actions of adversaries.

CURIE attempts to learn under adversarial conditions by filtering out data points injected by the adversary using an algorithm based on clustering. The authors claim that attacked samples stand out from non-attack samples in (feature + label) space. As filtering is a pre-processing step that happens before training, CURIE can, in fact, be used in conjunction with LID-SVM to further improve the attack resistance.

A distributed SVM based learning solution allows for lower communication overhead without significantly compromising detection accuracy as shown in Section 4.4.2. Therefore it is ideal to be used in the SDR based cognitive radio network to detect rogue transmission sources. But the main drawback of the distributed learning system is that it exposes multiple entry points for attackers and an attack can propagate through the network even if a single node is compromised. But as demonstrated by the experimental results, using LID-SVM would facilitate secure distributed detection using SDRs while benefiting from the reduced communication overhead provided by the DSVM framework.

Although no significant differences in terms of running times were observed during the above experiments, further research could be conducted to determine the relative efficiency of the different defense algorithms. While mini-batch sampling is a tested method for improving the efficiency of LID-SVM [106, 100], there is room for significant improvement through parallelization.

In summary, the experiments demonstrate that (i) SVMs are vulnerable to adversarial label flip attacks and input perturbation attacks, (ii) LID values in the input space may not have sufficient distinguishing power when the data from the two classes are not linearly separable (whereas K-LID does), (iii) K-LID based LID-SVM can withstand label flipping attacks as well as input perturbation attacks (iv) de-emphasizing the effect of suspected samples gives better performance than methods that attempt to make all samples contribute to the decision process (e.g., LS-SVM and LN-SVM), (v) distributed detection using a DSVM framework has less communication overhead compared to a centralized learner under adversarial conditions.

4.5 Summary

In this chapter, we have addressed the challenge of increasing the attack resistance of SVMs against adversarial training time attacks. We observed that carefully crafted label flips and input perturbations can significantly degrade the classification performance of SVMs. We introduced three different label dependent variations of LID that can be used in situations with label flips and introduced a novel LID approximation (K-LID) that makes use of the kernel matrix to obtain the LID values. Using the K-LID, we proposed a weighted SVM (LID-SVM) and showed by testing against different attacks on several real-world datasets that it can be successfully utilized against label flip attacks as well as input perturbation attacks. While there were some instances where LS-SVM, LN-SVM, and CURIE outperformed LID-SVM, we observed that LID-SVM can achieve a higher level of stability across the different attacks and datasets considered in this evaluation. We observed that by using LID-SVM in a distributed manner, the learner can significantly reduce the communication overhead without sacrificing the classification accuracy.

However, the practicality of the likelihood ratio (LR) based weighting scheme is limited due to the learner's knowledge (regarding the attack/attacker) in real-world applications. To address this, we look at a defense mechanism that makes no assumption about the attack/attacker in the following chapter.

Chapter 5

Defending Regression Learners Against Poisoning Attacks

Regression algorithms are vulnerable to targeted malicious attacks such as training data poisoning, through which adversaries can manipulate the behavior of the prediction models. Previous works that attempt to address this problem rely on assumptions about the attack/attacker. In this chapter, we introduce a novel Local Intrinsic Dimensionality (LID) based measure called N-LID that compares a sample's LID to other samples in its neighborhood. We then show that N-LID can distinguish poisoned samples from normal samples and propose an N-LID based defense approach that makes no assumptions of the attacker. Through extensive numerical experiments with benchmark datasets, we show that the proposed defense mechanism outperforms state of the art defenses in terms of prediction accuracy and running time. This chapter addresses the research question 3 introduced in Chapter 1.

5.1 Introduction

LINEAR regression models are a fundamental class of supervised learning, with applications in healthcare and business [109, 110]. Recent works in the literature show that the performance of regression models degrades significantly in the presence of poisoned training data [74, 73, 72]. Through such attacks, adversaries attempt to force the learner to learn a prediction model with impaired prediction capabilities. Any application that relies on regression models for automated decision making could potentially be compromised and make decisions that could have serious consequences [111].

In practice, such attacks can take place in situations where the attacker has an opportunity to introduce poisoned samples to the training process. For example, this can occur when data is collected using crowd-sourcing marketplaces, where organizations build

data sets with the help of individuals whose authenticity cannot be guaranteed. Due to the size and complexity of datasets, it is infeasible to manually inspect and filter samples that are likely to be adversarial. Therefore, to address this problem, defense mechanisms that take adversarial perturbations into account need to be embedded into regression models.

Most works in the literature are related to robust regression, where regression models are trained in the presence of stochastic noise instead of maliciously poisoned data [63, 66]. Recently, however, several works have presented linear regression models that consider the presence of adversarial data samples. Jagielski et al. [74] and Liu et al. [73] present two such defense models that iteratively exclude data samples with the largest residuals from the training process. However, both require the learner to be aware of the number of normal samples in the training dataset, which can be considered as an overestimation of the learner's knowledge, making them impractical.

Local Intrinsic Dimensionality (LID) is a metric that gives the dimension of the subspace in the local neighborhood of each data sample [20, 21]. LID has been applied for detecting adversarial samples in Deep Neural Networks (DNNs) [100] and as a mechanism to reduce the effect of noisy labels for training DNNs [104]. In this chapter, we propose a novel LID-based defense mechanism that weights each training sample based on the likelihood of them being normal samples or poisoned samples. The resulting weight vector can then be used in conjunction with any linear regression model that supports a weighted loss function (e.g., weighted linear least squares function). Therefore the proposed defense mechanism can be used to make learners such as ridge regression, LASSO regression, elastic-net regression, and neural network regression (NNR) resilient against poisoning attacks.

We first introduce a novel LID measure called *Neighborhood LID ratio* (N-LID) that takes into account the LID values of the sample of interest as well as its k nearest neighbors. Therefore N-LID can identify regions with similar LID values and samples that have significantly different LID values than their neighbors. We then show that N-LID values of poisoned and normal samples have two distinguishable distributions and introduce a baseline weighting mechanism based on the likelihood ratio of each data sample's

N-LID (i.e., how many times more likely that the N-LID value is from the N-LID distribution of normal samples than the N-LID distribution of poisoned samples). Although the learners' capabilities are exaggerated, this paves the way for another N-LID based weighting scheme that assumes no knowledge of the attacker or the attacked samples. The latter defense has up to 76% lower mean squared error (MSE) compared to an undefended ridge model, without the increased computational costs associated with prior works.

As a real-world application of the proposed defense algorithm, we consider the following security application in the communications domain as a case study. Software-defined radios (SDRs) with considerable computing and networking capabilities can be utilized as an inexpensive scanner array for distributed detection of transmissions. Based on the captured transmission signals, the transmission sources are assigned a risk value $\tau \in [0, 1]$, based on the probability of them being malicious transmission sources (i.e., rogue agent). To clarify, background radio traffic (e.g., civilians) would have a τ value closer to zero, and a transmission source identified as a rogue agent would have a τ value closer to one. Due to the widespread presence of encryption methods, the risk association for transmission sources has to be done based on their statistical characteristics, without considering the transmitted information.

The task of assigning a risk value can be formulated as a regression problem where the learner creates an estimator based on data collected from known transmission sources. However, if the rogue agents deliberately try to mask their activities by altering their transmission patterns during the initial deployment of the SDR network (i.e., when the data is collected for training), they may force the learner to learn an estimator that is compromised (poisoning attack). Therefore, to prevent the estimator from being compromised, the learner uses a defense algorithm during the training phase. In Section 5.5.1, we present details of the simulation setup followed by empirical results demonstrating the usefulness of our proposed defense.

The main **contributions** of this work are:

1. A novel LID measure called *Neighborhood LID ratio* that takes into account the LID values of a sample's k nearest neighbors.

2. An N-LID based defense mechanism that makes no assumptions regarding the attack, yet can be used in conjunction with several existing learners such as ridge, LASSO and NNR.

3. Extensive numerical experiments that show N-LID weighted regression models provide significant resistance against poisoning attacks compared to state of the art alternatives.

The remainder of the chapter is organized as follows. Section 5.2 formally defines the problem being addressed followed by Section 5.3, where we introduce the defense algorithms. We then describe the existing attacks against regression models in Section 5.4. Section 5.5 includes a detailed empirical analysis of the attacks and defenses on several real-world datasets followed by the results and discussion. The concluding remarks of Section 5.6 conclude the chapter.

5.2 Problem statement

We consider a linear regression learning problem in the presence of a malicious adversary. Define the labeled pristine training data (i.e., prior to adversarial perturbations) as $S := (X, y)$, where $X \in \mathbb{R}^{n \times d}$ is the feature matrix and $y \in \mathbb{R}^{n \times 1}$ the corresponding response variable vector. We let $x_i \in \mathbb{R}^{d \times 1}$ denote the i^{th} training sample, associated with a corresponding response variable $y_i \in \mathbb{R}$ from y. Note that each $y_i \in [0, 1]$ for $i \in \{1, \ldots, n\}$ unlike in classification problems where the labels take discrete categorical values.

A linear regression model can be described by a linear function of the form $h_\theta(x_i) = \omega^T x_i + b$ that predicts the response variable $\hat{y}_i$ for each $x_i \in X$. The function h is parameterized by the vector $\theta = (\omega, b) \in \mathbb{R}^{d+1}$ consisting of the feature weights $\omega \in \mathbb{R}^d$ and the bias of the hyperplane $b \in \mathbb{R}$. The parameter vector θ is obtained by applying a learning algorithm, such as ridge regression (shown below), on S:

$$\theta^* = \arg\min_{\theta} \quad \frac{1}{n} \sum_{i=1}^{n} \left(y_i - h_\theta(x_i)\right)^2 + \lambda \|\omega\|_2^2. \tag{5.1}$$

Note that λ is a regularization parameter.

The adversary's goal is to maximize the regression model's prediction error for unseen data samples (i.e., testing phase). To succeed in this, the adversary introduces a set of poisoned data samples $\bar{X} \in \mathbb{R}^{p \times d}$ and labels $\bar{y} \in \mathbb{R}^{p \times 1}$ into the pristine training set S, resulting in a contaminated training dataset $\tilde{S} := (\tilde{X}, \tilde{y})$ where $\tilde{X} = X \cup \bar{X}$ and $\tilde{y} = y \cup \bar{y}$. Applying a learning algorithm on the contaminated training data would result in a compromised regression model as follows,

$$\tilde{\theta} = \underset{\theta}{\arg\min} \quad \frac{1}{n+p} \sum_{i=1}^{n+p} \left(y_i - h_\theta(x_i)\right)^2 + \lambda \|\omega\|_2^2, \tag{5.2}$$

where $y_i \in \tilde{y}$ and $x_i \in \tilde{X}$. Through $\bar{X}$ and $\bar{y}$, the adversary forces the learner to obtain a parameter vector $\tilde{\theta}$ that is significantly different from θ^*, which it would have obtained had the training data been unaltered.

To address this problem, we propose an LID based weighting scheme that can be incorporated into learning algorithms that optimize quadratic loss functions such as the Mean Squared Error in Equation (5.2). In *weighted least squares (WLS)*, a weight $\beta_i \in [0, 1]$ is assigned for each sample in order to discriminate and vary their influence on the optimization problem. A small β_i value would allow for a large residual value, and the effect of the sample would be de-emphasized. Conversely, a large β_i value would emphasize that particular sample's effect. The weighted regression learning problem is formulated as the following convex optimization problem:

$$\hat{\theta} = \underset{\theta}{\arg\min} \quad \frac{1}{n+p} \sum_{i=1}^{n+p} \beta_i \left(y_i - h_\theta(x_i)\right)^2 + \lambda \|\omega\|_2^2. \tag{5.3}$$

By carefully selecting a weight vector β, the learner can minimize the effects of the adversarial perturbations and recover the correct estimator $\hat{\theta} \approx \theta^*$.

Neural network based regression models (NNR) follow a similar gradient-based optimization algorithm that minimizes the MSE between the model prediction and the correct response variable. The algorithm uses a linear activation function for the output layer with a single neuron and backpropagation to compute the gradients. The sample weight-based defense can be used with NNR models by multiplying the loss of each sam-

ple by its corresponding weight to increase or decrease its importance in the optimization process.

5.3 Defense algorithms

We now present our novel LID based mechanism to obtain the weight vector β and show how it can easily be incorporated into existing regression algorithms.

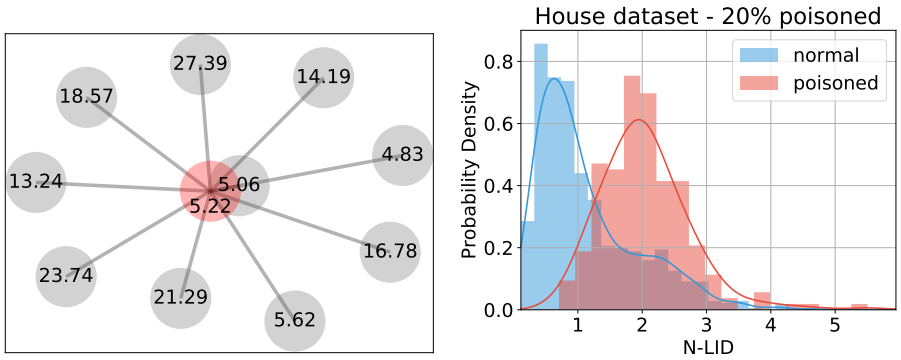
5.3.1 Neighborhood LID Ratio.

In Section 4.3.2, we first introduced the theory of Local Intrinsic Dimensionality (LID). We now introduce a novel LID based measure called neighborhood LID ratio (N-LID) and discuss the intuition behind using N-LID to identify adversarial samples during training. By perturbing the feature vector, the adversary moves a training sample away from the distribution of normal samples. Computing LID estimates with respect to its neighborhood would then reveal an anomalous distribution of the local distance to these neighbors. Furthermore, as $\widehat{\text{LID}}(x)$ is an indicator of the dimension of the subspace that contains x, by comparing the LID estimate of a data sample to the LID estimates of its nearest neighbors, we can identify regions that have a similar lower-dimensional subspace. More importantly, any samples that have a substantially different lower-dimensional subspace compared to its neighbors can be identified as poisoned samples.

Considering this aspect of poisoning attacks, we propose a novel LID ratio measure that has similar properties as the local outlier factor (LOF) algorithm introduced by Breunig et al. [19] as follows:

Definition 5.1. *(Neighborhood LID ratio)*
Given a data sample $x \in X$, the neighborhood LID ratio of x is defined as:

$$N\text{-}LID(x) = \frac{\frac{1}{k}\sum_{i=1}^{k} \widehat{LID}(x_i)}{\widehat{LID}(x)}. \tag{5.4}$$



(a) N-LID calculation of a poisoned sample

(b) N-LID distributions in House dataset.

Figure 5.1: The left figure shows the LID values of a poisoned sample (red) and its k nearest neighbors. The poisoned sample has an N-LID of 2.89. The right figure shows the N-LID distributions of the normal samples and poisoned samples when 20% of the data is poisoned.

Here, $\widehat{\text{LID}}(x_i)$ denotes the LID estimate of the i-th nearest neighbor from the k nearest neighbors of x. Figure 5.1a shows the LID estimates of a poisoned sample and its k nearest neighbors. As the poisoned sample's LID estimate is substantially different from its neighbors, the N-LID value calculated using Equation(5.4) highlights it as an outlier. As Figure 5.1b shows, N-LID is a powerful metric that can give two distinguishable distributions for poisoned and normal samples. Although ideally, we like to see no overlap between the two distributions, in real-world datasets, we see some degree of overlap.

5.3.2 Baseline Defense.

We now describe a baseline weighting scheme calculated using the N-LID distributions of poisoned and normal samples. Define p_n and p_a as the probability density functions of the N-LID values of normal samples and poisoned samples. For a given data sample x_i with the N-LID estimate N-LID(x_i), we define two possible hypotheses, H_n and H_a as

$$\begin{aligned} H_n &: \text{N-LID}(x_i) \sim p_n, \\ H_a &: \text{N-LID}(x_i) \sim p_a, \end{aligned} \tag{5.5}$$

where the notation "N-LID(x_i) $\sim p$" denotes the condition "N-LID(x_i) is from the distribution p". To clarify, H_n is the hypothesis that N-LID(x_i) is from the N-LID distribution of normal samples and H_a is the hypothesis that N-LID(x_i) is from the N-LID distribution of poisoned samples.

The *likelihood ratio* (LR) is usually used in statistics to compare the goodness of fit of two statistical models. We define the likelihood ratio of a data sample x_i with the N-LID estimate N-LID(x_i) as

$$\Lambda(\text{N-LID}(x_i)) = p_n(\text{N-LID}(x_i))/p_a(\text{N-LID}(x_i)), \tag{5.6}$$

where $p(\text{N-LID}(x_i))$ denotes the probability of the N-LID of sample x_i w.r.t the probability distribution p. To clarify, $\Lambda(\text{N-LID}(x_i))$ expresses how many times more likely it is that N-LID(x_i) is under the N-LID distribution of normal samples than the N-LID distribution of poisoned samples.

As there is a high possibility for p_n and p_a to have an overlapping region (Figure 5.1b), there is a risk of emphasizing the importance of (giving a higher weight to) poisoned samples. To mitigate that risk, we only de-emphasize samples that are suspected to be poisoned (i.e., low LR values). Therefore, we transform the LR values such that $\Lambda(\text{N-LID}(x)) \in [0, 1]$. The upper bound is set to 1 to prevent overemphasizing samples.

Subsequently, we fit a hyperbolic tangent function to the transformed LR values (z) in the form of $0.5(1 - \tanh(az - b))$ and obtain suitable values for the parameters a and b. The scalar value of 0.5 maintains the scale and vertical position of the function between 0 and 1. By fitting a smooth function to the LR values, we remove the effect of noise and enable the calculation of weights for future training samples. Finally, we use the N-LID value of each x_i and use Equation (5.7) to obtain its corresponding weight β_i.

$$\beta_i = 0.5(1 - \tanh(a\text{N-LID}(x_i) - b)). \tag{5.7}$$

5.3.3 Attack Unaware Defense.

The LR based weighting scheme described above assigns weights to training samples based on the probabilities of their N-LID values. This approach requires the learner to be aware of the probability density functions (PDFs) of normal and poisoned samples. The two distributions can be obtained by simulating an attack and deliberately altering a subset of data during training, by assuming the distributions based on domain knowledge or prior experience or by having an expert identify attacked and non-attacked samples in a subset of the dataset. However, we see that the N-LID measure in Equation (5.4) results in larger values for poisoned samples compared to normal samples (Figure 5.1b). In general, therefore, it seems that a weighting scheme that assigns small weights to large N-LID values and large weights to small N-LID values would lead to an estimator that is less affected by the poisoned samples present in training data.

Considering this aspect of N-LID, we choose three weighting mechanisms that assign $\beta = 1$ for the sample with the smallest N-LID and $\beta = 0$ for the sample with the largest N-LID as shown in Figure 5.2. Note that there is a trade-off between preserving normal samples, and preventing poisoned samples from influencing the learner. The concave weight function attempts to preserve normal samples, which causes poisoned samples to affect the learner as well. In contrast, the convex weight function reduces the impact of poisoned samples while reducing the influence of many normal samples in the process. The linear weight function assigns weights for intermediate N-LID values linearly without considering the trade-off.

We obtain the weight functions by first scaling the N-LID values of the training samples to $[0, 1]$. Take v_i as the scaled N-LID value of sample x_i. Then, for the linear weight function, we take $\beta_i = 1/v_i$ as the weight. For the concave weight function we use $\beta_i = 1 - v_i^2$ and for the convex weight function we use $\beta_i = 1 - (2v_i - v_i^2)^{0.5}$. We choose these two arbitrary functions as they possess the trade-off mentioned in the paragraph above. Any other functions that have similar characteristics can also be used to obtain the weights.

The main advantage of our proposed defense is that it is decoupled from the learner; it can be used in conjunction with any learner that allows the loss function to be weighted.

Algorithm 3 N-LID LR Defense Algorithm

input $\tilde{S} = (\tilde{X}, \tilde{y})$ ▷ contaminated training data
output uninfluenced ridge regression model
initialize N-LID$_a$, N-LID$_n$, Λ, $\beta = [\,]$
N-LID$_a \leftarrow$ n-lid-calculation$(\tilde{S})$ ▷ calculate n-lid values of poisoned samples
N-LID$_n \leftarrow$ n-lid-calculation$(\tilde{S})$ ▷ calculate n-lid values of normal samples
$\Lambda \leftarrow$ calculate likelihood ratio for each N-LID value
fit function $g(z) = 0.5(1 - \tanh(az - b))$ to Λ to find a and b
$\beta \leftarrow g(\text{N-LID}(x))$ ▷ obtain weight β_i of each $x_i \in X$ using the N-LID of x_i
N-LID LR ridge = train-weighted-ridge-model$(\tilde{X}, \tilde{y}, \beta)$

Algorithm 4 N-LID CVX Defense Algorithm

input $\tilde{S} = (\tilde{X}, \tilde{y})$ ▷ contaminated training data
output uninfluenced ridge regression model
initialize N-LID, $v, \beta = [\,]$
N-LID $\leftarrow$ n-lid-calculation$(\tilde{S})$ ▷ calculate n-lid values of all samples
$v \leftarrow$ scaler(N-LID) ▷ scale the N-LID values to $[0, 1]$
$\beta \leftarrow 1 - (2v - v^2)^{0.5}$ ▷ obtain weight β_i of each x_i using the chosen convex function
N-LID CVX ridge = train-weighted-ridge-model$(\tilde{X}, \tilde{y}, \beta)$

In Section 5.5, we demonstrate the effectiveness of our proposed defense by incorporating it into a ridge regression model and an NNR model.

The high level procedures used to obtain uninfluenced ridge regression models under adversarial conditions are formalized in Algorithms 3 and 4 for the baseline defense (N-LID LR) and the convex weight function based defense (N-LID CVX).

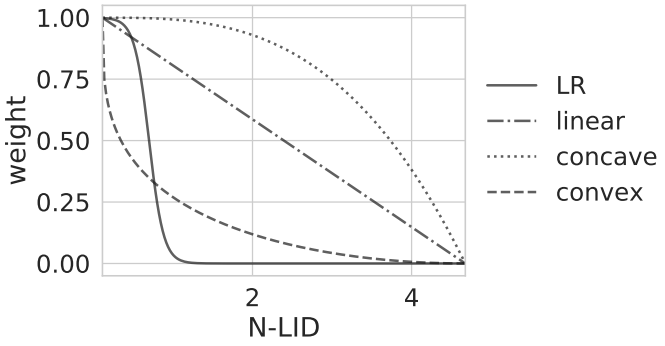


Figure 5.2: The N-LID based weighting schemes.

5.4 Threat models

In this work, we assume a learner that does not have access to a collection of pristine data samples. The training data it uses may or may not be manipulated by an adversary. The learner chooses its defense algorithm and regression model, without knowing details of the attack employed by the adversary.

The attacks being considered are *white-box* attacks, with the attacker knowing the pristine training data and any hyper-parameters of the learning algorithm. While these assumptions exaggerate the capabilities of a real-world attacker, it allows us to test the performance of defense algorithms under a worst-case scenario. Moreover, relying on secrecy for security is considered as a poor practice when designing defense algorithms [44].

We employ the attack algorithm introduced by Jagielski et al. [74] against linear regression models, which is an adaptation of the attack algorithm by Xiao et al. [72]. The latter attack algorithm is used in prior works on adversarial regression as the threat model [73]. While giving a brief description of the attack algorithm, we refer the readers to [74] for more details.

The underlying idea is to move selected data samples along the direction that maximally increases the MSE of the estimator on a pristine validation set. Unlike in classification problems, the attacker can alter the response variable y as well. We use similar values used by Jagielski et al. [74] for the hyper-parameters that control the gradient step and convergence of the iterative search procedure of the attack algorithm. The line search learning rate (η), which controls the step size taken in the direction of the gradient, is selected from a set of predefined values by evaluating the MSE values on a validation set.

The amount of poisoning is controlled by the *poisoning rate*, defined as $p/(p+n)$, where n is the number of pristine samples, and p is the number of poisoned samples. The attacker is allowed to poison up to 20% of the training data in line with prior works in the literature [74, 72].

The Opt attack algorithm selects the initial set of data points to poison randomly from the training set. The corresponding response variables $\bar{y}$ are initialized in one of

two methods: (i) $\bar{y} = 1 - y$ and (ii) $\bar{y} = \text{round}(1 - y)$. The first approach, known as Inverse Flipping (IFlip), results in a less intense attack compared to the latter, known as *Boundary Flipping* (BFlip), which pushes the response values to the extreme edges. In our experiments, we test against attacks that use both initialization techniques.

We also considered the optimization based attack by Tong et al. [75], but it was unable to exact a noticeable increase in MSE even at a poisoning rate of 20%. Therefore we do not present experimental results for it.

5.5 Experimental results and discussion

In this section, we describe the datasets used and other procedures of the experimental setup. We extensively investigate how the performance of N-LID weighted regression models holds against an increasing fraction of poisoned training data.

5.5.1 Case study: risk assignment to suspicious transmissions

We provide here a brief description of the simulations that were conducted to obtain the data for the case study that we are considering in this chapter. Refer to Chapter 3 for additional information regarding the security application in SDR networks. We use the same simulation setup introduced in Section 3.5.2 for classification for the regression problem in this chapter. Therefore the dataset contains the same 111 features. The response variables (i.e., the risk of being a rogue agent) are assigned based on the positions of the data samples in the feature space.

5.5.2 Benchmark datasets.

We use the following combination of relatively high dimensional and low dimensional regression datasets in our experimental analysis. In particular, The House and Loan datasets are used as benchmark datasets in the literature [74]. Table 5.1 provides the training and test set sizes, the regularization parameter λ of the ridge regression model, and the line search learning rates η of the attacks considered for each of the datasets.

Table 5.1: Description of the datasets.

Dataset	# of features	Training size	Test size	λ	η of BFlip	η of IFlip
House	274	840	280	0.0373	0.01	0.01
Loan	88	840	280	0.0273	0.05	0.05
OMNeT	111	502	125	0.0002	0.01	0.01
Grid	12	840	280	0.0173	0.01	0.30
Machine	6	125	42	0.0223	0.03	0.03

House dataset: The dataset uses features such as year built, floor area, number of rooms, and neighborhood to predict the sale prices of houses. There are 275 features in total, including the response variable. All features are normalized into $[0, 1]$. We fix the training set size to 840, and the test set size to 280.

Loan dataset: The dataset contains information regarding loans made on *Lending-Club*. Each data sample contains 89 features, including loan amount, term, the purpose of the loan, and borrower's attributes with the interest rate as the response variable. We normalize the features into $[0, 1]$ and fix the training set and test set size as above.

Grid dataset: Arzamasov et al. [112] simulated a four-node star electrical grid with centralized production to predict its stability, w.r.t the behavior of participants of the grid (i.e., generator and consumers). The system is described with differential equations, with their variables given as features of the dataset. In our work, we use the real component of the characteristic equation root as the response variable for the regression problem.

Machine dataset: The dataset by Kibler et al. [113] is used to estimate the relative performance of computer hardware based on attributes such as machine cycle time, memory, cache size, and the number of channels.

5.5.3 Experimental setup.

For each learning algorithm, we find its hyperparameters using five-fold cross-validation on a pristine dataset. For LID calculations we tune k over $[10, 100)$ as previously done by Ma et al. [100]. Kernel density estimation for LR calculation was performed using a Gaussian kernel with the bandwidth found using a cross-validation approach to avoid

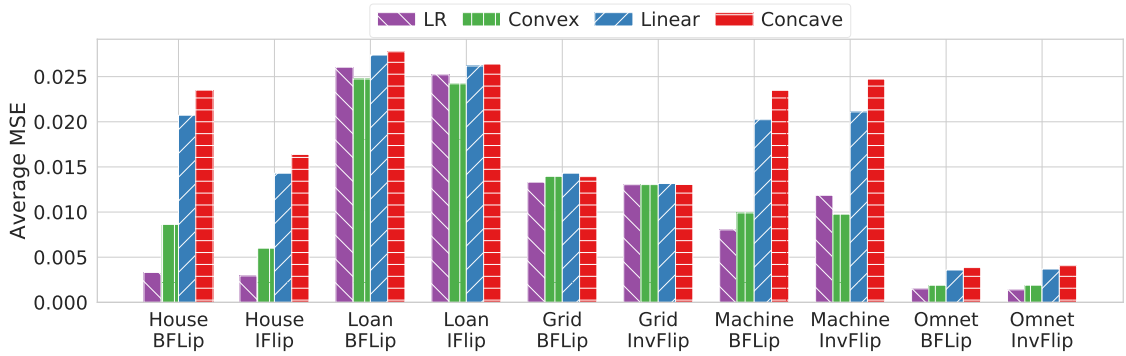


Figure 5.3: The average MSE for each weighting scheme under the two attack configurations (BFlip and IFlip) for the benchmark datasets and the OMNeT case study.

overfitting. The effects of kernel density estimation are neutralized by the smooth function fitting (Equation (5.7)); therefore, the algorithm is robust to these choices.

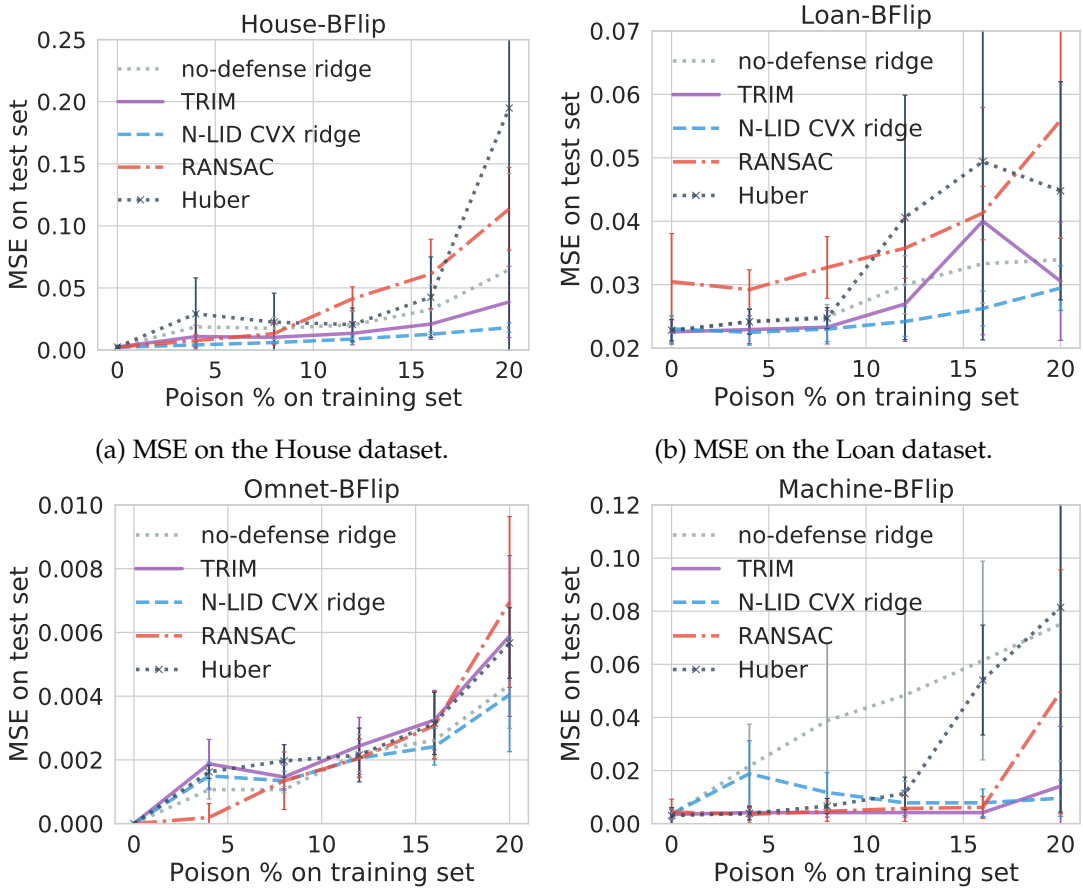
For each dataset, we create five cross-validation sets. We present the average MSE of the cross-validation sets for each learner when the poisoning rate increases from 0 to 20. While increasing the poisoning rate over 20% is feasible, it is unrealistic to expect a learner to withstand such amounts of poisoned data, and it also assumes a powerful attacker with significant influence over the training data. As the existence of such an attacker is unlikely, we limit the poisoning rate to 20%, similar to prior works in the literature.

5.5.4 Results and discussion.

Attacker unaware defense

Figure 5.3 compares the average MSE for the three attacker unaware weighting schemes and the LR based weighting scheme that we introduced in Section 5.3.3. Of the three attacker unaware weighting schemes, we see that the convex weight function has 58% lower average MSE in House, 9% in Loan, 51% in Machine, and 47% in OMNeT. This is not surprising as the convex weight function is the closest to the ideal LR based weight function (Figure 5.2). The result suggests that it is favorable to the learner to reduce the weights of suspected adversarial samples even if there is an increased risk of de-emphasizing normal samples as well. This is to be expected with linear regression data

where there is high redundancy; losing a portion of normal data from training would not significantly impact the performance of the estimator. In future studies, it might be possible to investigate the performance of the three weighting functions in situations where data redundancy is low, such as non-linear regression problems. In the following sections, the term *N-LID CVX* refers to a ridge regression model with the convex weight function based defense.



(a) MSE on the House dataset.

(b) MSE on the Loan dataset.

(c) MSE on the OMNeT dataset.

(d) MSE on the Machine dataset.

Figure 5.4: The average MSE on the test set for each defense algorithm under the B-Flip Opt attack. The poisoning rate is increased from 0% to 20%.

Table 5.2: The percentage increase/decrease of average MSE of each defense algorithm compared to a ridge regression model with no defense. The best results are indicated in **bold** font. The rows corresponding to the defense algorithms introduced in this work are highlighted in grey.

	House		Loan		OMNeT		Grid		Machine	
	BFlip	IFlip	BFlip	IFlip	BFlip	IFlip	BFlip	IFlip	BFlip	IFlip
N-LID LR ridge	**-87.27**	**-84.24**	-7.53	-4.78	**-20.53**	**-29.36**	-4.29	-0.86	-80.64	-71.63
N-LID CVX ridge	-66.83	-67.74	**-12.16**	**-8.55**	0.30	-3.52	-2.84	-0.13	-76.12	-76.63
TRIM	-38.13	-70.43	-1.59	-0.72	31.76	6.83	-6.74	-0.97	**-86.24**	-61.96
RANSAC	53.26	61.83	33.36	22.63	20.36	61.64	-5.92	-1.05	-70.24	**-83.37**
Huber	99.81	46.97	22.22	23.93	28.61	7.24	**-7.18**	**-2.14**	-35.55	-39.60

In the absence of an attack

As stated in Section 5.4, we assume that the learner is unaware of the attacker; therefore the training data may or may not be manipulated by the adversary. Hence it is possible to have a scenario where the learner utilizes a defense algorithm when an attack is not present.

It is common for adversarial learning algorithms to sacrifice performance in the absence of an attack to improve their attack resilience in the presence of attacks. Improving attack resilience often results in decreased performance in the absence of an attack and vice versa. However, as Figure 5.4 shows, all the adversarial/robust learners considered in the experiments perform similarly to a learner with no built-in defense when the poisoning rate is zero. This result may be explained by the fact that there is a high level of redundancy in the linear regression datasets. Therefore even if a learner removes/de-emphasizes a subset of samples during training even when there is no attack, the remaining data samples are sufficient to obtain an accurate estimator.

In the presence of an attack

First, we consider the performance of the undefended ridge learner, as shown in Figure 5.4. We observe that the MSE values increase by 25.6% and 14.56% on the House dataset, 0.50% and 0.27% on the Loan dataset, 0.46% and 0.01% on the Grid dataset, and 20.75% and 20.15% on the Machine dataset under BFlip and IFLip Opt attacks respectively. We

observe a similar trend in the OMNeT dataset, as well. These results suggest that (i) undefended regression learners can be severely affected by poisoning attacks, and (ii) among the two attack configurations considered, BFlip is more potent than IFlip as the initial movement of the response variable y is more aggressive compared to IFlip.

We now compare the performance of an N-LID LR weighted ridge model and N-LID CVX weighted ridge model against TRIM [74], RANSAC [66] and Huber [63] under poisoning attacks. Table 5.2 compares their average MSE values with the average MSE of a ridge regression model with no defense (shown as the percentage increase or decrease).

First, we consider the performance of the defenses on the three datasets with a relatively large number of dimensions (i.e., House, Loan, and OMNeT). Considering the experimental evidence on the performance of the robust learners (i.e., Huber and RANSAC), we see that they are not effective at defending against poisoning attacks when the dimensionality of the data is high. In fact, their performance is significantly worse than an undefended ridge regression model. The average MSE of RANSAC is up to 61.83% higher on the House dataset, up to 33.36% higher on the Loan dataset and up to 61.64% higher on the OMNeT dataset when compared with an undefended ridge model. Huber also increases the average MSE by 99.81%, 23.93%, and 28.61%, respectively, for the three datasets. This finding is consistent with that of Jagielski et al. [74] who reported similar performance for robust learners when used with high-dimensional datasets.

This behavior of Huber and RANSAC may be due to high dimensional noise. In high dimensional spaces, poisoned data might not stand out due to the presence of high dimensional noise [67], thereby impairing their performance. Moreover, it should be noted that these defenses are designed against stochastic noise/outliers, not adversarially poisoned data. In an adversarial setting, a sophisticated adversary may poison the data such that the poisoned samples have a similar distribution to normal data, thereby, reducing the possibility of being detected.

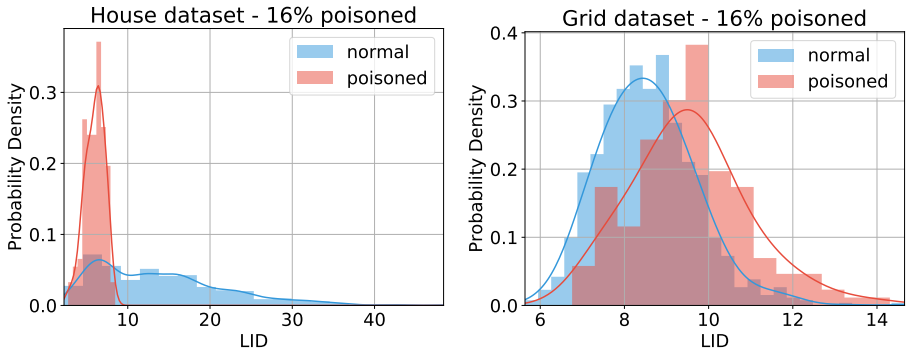
In contrast, TRIM, a defense that is designed against poisoning attacks, has up to 70.43% and 1.59% lower MSE values compared to an undefended ridge model on the House and Loan datasets. Interestingly, on the OMNeT dataset, we observe 31.76% and 6.8% higher MSE values for TRIM. Readers may also notice that the prediction perfor-

mance of TRIM shown in [74] is different from what we have obtained. Analyzing the provided code revealed that in their experiments the poisoned training dataset is created by appending the poisoned samples $\bar{X}$ to the matrix X containing the pristine samples (resulting in $\tilde{X} \in \mathbb{R}^{(n+p)\times d}$). During the first iteration of TRIM, the first n rows are selected to train the regression model, which happens to be the pristine data in this case. To avoid this issue, in our experiments, we randomly permute the rows of $\tilde{X}$ before training with TRIM.

The N-LID based defense mechanisms consistently outperformed the other defenses against all the attack configurations on the three datasets with a large number of dimensions. On the House dataset, N-LID LR weighted ridge model has the best prediction performance (87.27% lower MSE) with the N-LID CVX weighted ridge model being closely behind (67.74% lower MSE). A similar trend is present on the OMNeT dataset, where the N-LID LR weighted ridge model has a 29.36% lower average MSE. On the Loan dataset, we observe that N-LID CVX weighted ridge model has the best performance with a 12.16% lower average MSE. It is interesting to note that the performance of the N-LID CVX weighting mechanism, which makes no assumptions of the attack, is on par with the N-LID LR baseline weighting scheme in the majority of the test cases considered.

Turning now to the experimental evidence on the performance of the defenses on the two datasets with a small number of dimensions (i.e., Grid and Machine), we see that all the defenses considered can outperform the ridge regression learner without a defense. What stands out in Table 5.2 is that the robust regression learners (Huber and RANSAC) that performed worse than an undefended ridge regression model on the three datasets with a large number of dimensions, have significantly lower MSE values on the two datasets with the smaller number of dimensions. Huber has the best prediction performance on the Grid dataset even exceeding the N-LID based defenses. It has up to 7.18% and 39.60% lower MSE values, respectively, for the two datasets. The average MSE value of RANSAC is 5.92% lower on the Grid dataset while having an 83.37% lower MSE on the Machine dataset.

N-LID based defenses perform consistently across all the datasets considered when

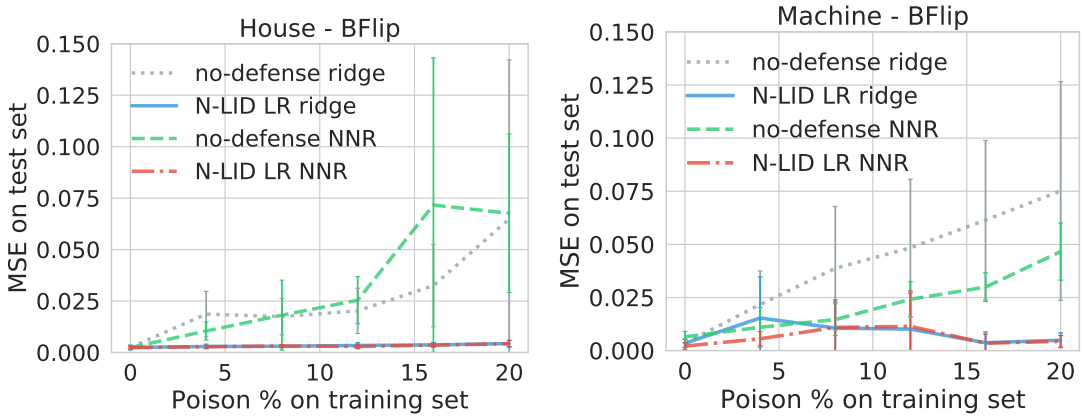


(a) LID distributions in House dataset. (b) LID distributions in Grid dataset.

Figure 5.5: The LID distribution of poisoned and normal samples when 16% of the training data is poisoned.

compared against an undefended ridge regression model. However, their performance improvement is relatively smaller on Grid and Machine. The lack of performance on the two datasets with a small number of dimensions may be explained by Figure 5.5, which shows the LID (not N-LID) distributions of normal and poisoned samples in two datasets with different dimensions. The LID estimate of a sample is the dimension of the true low-dimensional subspace (at least approximately) in which the particular sample lies (Section 4.3.2). In other words, it describes the optimal number of features needed to explain the salient features of the dataset. For datasets that have a small number of features, the range of values LID could take is also narrow (as LID is less than the actual number of dimensions). Therefore, if the majority of the features of a particular dataset are salient features, the resulting LID distributions of normal and poisoned samples would have a large overlap (Figure 5.5b). Overlapping LID distributions lead to overlapping N-LID distributions, thereby impairing the performance of LID based defenses on such datasets.

In contrast, datasets with a large number of dimensions may allow for a wider range of LID values. In such high-dimensional spaces poisoned and normal samples would exhibit distinguishable LID distributions as shown in Figure 5.5a (unless a majority of the features are salient features as explained above). Moreover, in such spaces where most data points seem equidistant, the N-LID based defenses have good performance because they characterize poisoned samples from normal samples using the true low-dimensional manifold of the data. However, the issue of obtaining distinguishable LID



(a) MSE on the House dataset.

(b) MSE on the Machine dataset.

Figure 5.6: The average MSE of NNR and ridge models when the poisoning rate is increased from 0% to 20%.

distributions for poisoned and normal samples for datasets with a high percentage of salient features is an intriguing one that could be usefully explored in further research.

Practical implications of poisoning attacks

We now turn to the practical implications of the poisoning attacks on the case study considered in this chapter. In a security application such as the one discussed, having the decision variable change by even a small amount could have dire consequences. We observe that the BFlip attack changes the assigned risks of 10.78% of the transmission sources by 50% when a ridge regression model without a defense is used (at 20% poisoning rate). This means a transmission source that should ideally be assigned a high-risk probability may be assigned a significantly lower risk value or vice versa. Nearly half of all the transmission sources (45.51%) have their assigned risks altered by at least 10%. However, the N-LID LR defense model resulted in none of the transmission sources having their assigned risk increased by over 10%.

Computational complexity

Table 5.3 compares the average training time (in seconds) for the learners considered in the experiments. Because ridge regression with no defense has the lowest training time,

Table 5.3: The average training time (s) as a factor of the training time of a ridge regression model with no defense. The best results are indicated in **bold** font. The rows corresponding to the defense algorithms introduced in this work are highlighted in grey.

	House		Loan		OMNeT		Grid		Machine	
	BFlip	IFlip	BFlip	IFlip	BFlip	IFlip	BFlip	IFlip	BFlip	IFlip
N-LID LR ridge	14.85	17.31	47.36	42.21	30.46	28.08	127.13	88.98	199.11	198.27
N-LID CVX ridge	**11.30**	**12.88**	**33.83**	**30.04**	**22.91**	**19.52**	86.43	60.85	**9.89**	**9.78**
TRIM	19.55	23.32	39.74	41.87	33.87	31.47	83.83	90.47	15.49	11.04
RANSAC	35.65	40.31	74.62	68.32	41.48	39.28	124.35	73.19	128.26	95.18
Huber	122.18	149.28	182.52	167.81	207.86	169.31	**23.24**	**11.96**	19.50	15.08

we report the training times of the other algorithms as a factor of its training time. It is apparent from this table that N-LID CVX ridge has the lowest training time in eight of the ten experiments considered. Its training times are only 11.30 times higher on the House dataset, 33.83 times higher on the Loan dataset, 86.43 times higher on the Grid dataset, 9.89 times higher on the Machine dataset, and 22.91 times higher on the OMNeT dataset. TRIM, being an iterative method, has higher training times compared to N-LID CVX ridge in the majority of the test cases. But its computational complexity is significantly lower compared to RANSAC, which is another iterative method.

Of the learners considered, Huber has the most variation in average training time. We observe that on the three datasets with a large number of dimensions, its training times are up to 207.86 times higher compared to an undefended ridge model. But on the two datasets with a small number of dimensions, its training times are significantly less (with 11.96 being the lowest among all the defenses on the Grid dataset).

The main advantage of the N-LID based methods is that they are not iterative; the N-LID calculation, albeit expensive, is only done once. Therefore they exhibit significant savings in computation time while incurring little or no loss in prediction accuracy. Given the results of the datasets with a small number of dimensions (i.e., machine and grid), the N-LID based methods also appear to be invariant to the number of dimensions of the data, unlike the robust regression methods we have considered.

Note that the computational complexity of N-LID would scale poorly to the number

of training data samples due to its k-nearest neighbor based calculation. However, the N-LID calculation can be made efficient by estimating the LID of a particular data sample within a randomly selected sample of the entire dataset. We refer the readers to the work of Ma et al. [100] for further information about this process known as minibatch sampling. Furthermore, calculating the LIDs within each minibatch can be parallelized for improved performance.

Transferability of the defense

We now investigate the transferability of the proposed defense by applying it to a fundamentally different learner, neural network regression (NNR). Neural network based regression models follow a gradient-based optimization algorithm similar to Ridge regression that minimizes the MSE between the model prediction and the correct response variable. The algorithm uses a linear activation function for the output layer with a single neuron and backpropagation to compute the gradients. The sample weight-based defense can be used with NNR models by multiplying the loss of each sample by its corresponding weight to increase or decrease its importance in the optimization process.

First, we run the Opt attack against ridge regression and obtain a poisoned dataset. Then we train an NNR model on the poisoned data. The average MSE of the NNR model with no defense is 25.34% higher on the House dataset, 12.29% higher on the Loan dataset, 5.61% higher on the OMNeT dataset, 19.11% higher on the Grid dataset, and 46.67% lower on the Machine dataset compared to a ridge regression model with no defense. This result may be explained by the fact that an NNR model (which is a non-linear estimator) is too complex for a linear regression problem; therefore, it overfits the perturbed training data, resulting in poor prediction performance on an unperturbed test set.

However, our aim is to demonstrate that our proposed defense can be applied to different learning algorithms, not to compare the linear ridge regression model with a NNR model. To that end, we observe that if the loss function of the NNR is weighted using the N-LID LR weighting scheme we introduce, it can maintain a near 0% increase in MSE when the poisoning rate is increased to 20%. These findings demonstrate the

flexibility of our defense mechanism.

In summary, these results show that (i) robust learners are unable to withstand carefully crafted adversarial attacks on datasets where the number of dimensions is large, whereas adversarial learners can, (ii) N-LID based defenses that de-emphasize the effects of suspected samples consistently perform better in terms of lower MSE values and running times compared to other defenses, (iii) although N-LID CVX makes no assumptions of the attack, its performance is comparable to the baseline N-LID LR defense, and (iv) the N-LID based defenses can successfully be used in conjunction with different algorithms such as NNR models.

5.6 Summary

This chapter addresses the problem of increasing the attack resistance of linear regression models against data poisoning attacks. We observed that carefully crafted attacks could significantly degrade the prediction performance of regression models, and robust regression models are unable to withstand targeted adversarial attacks on datasets with a large number of dimensions. We introduced a novel LID measure that took into account the LID values of a data sample's neighbors and introduced several weighting schemes that alter each sample's influence on the learned model. Experimental results suggest that the proposed defense mechanisms are quite effective and significantly outperform prior art in terms of accuracy and computational costs. Further research should be undertaken to investigate the effects of poisoning attacks on non-linear regression models and how N-LID would perform in such scenarios.

Chapter 6
Adversarial Time Series Estimation

Software-defined radios (SDRs) with substantial cognitive (computing) and networking capabilities provide an opportunity for malicious individuals to jam the communications of other legitimate users. Channel hopping is a well known anti-jamming tactic used to evade jamming attacks.

In this chapter, we model the interaction between a transmitter, who uses chaotic pseudo-random patterns for channel hopping, and a sophisticated jammer, who uses advanced machine learning algorithms to predict the transmitter's frequency hopping patterns as a non-cooperative security game. We investigate the effectiveness of adversarial distortions in such a scenario to support the anti-jamming efforts by deceiving the jammer's learning algorithms. The optimal strategies in the formulated game indicate how adversarial distortions should be used by the players at every step of the game to improve their outcomes. The studied jamming/anti-jamming scenario combines chaotic time series generators, game theory, and online deep learning. This chapter addresses the research question 1 from Chapter 1.

6.1 Introduction

RECENT advances in software-defined radios (SDRs), cognitive networking technologies, and the increasing availability of low-cost hardware, have resulted in most applications becoming dependent on wireless networks for their regular operations. Inevitably, this has given adversaries new opportunities to conduct attacks and harm systems that rely on wireless networks. One of the most common forms of attacks in wireless networks is a jamming attack. Adversaries can utilize cheap and compact transmitters and receivers [114] to scan the transmission channels in a particular area and disrupt the communication between two or more legitimate parties by causing interference or collisions at the receiver side (i.e., denial of service attack).

This chapter focuses on a particular scenario where stochastic channel hopping is

utilized as an evasion strategy in the presence of a sophisticated jammer. Consider a unit immobilized in a contested environment with its location unknown to friendly search and rescue units in the area. Assume that the immobilized unit (T) still has the capability to transmit signals using its radio transmitter. Multiple friendly units in the area (receivers R) attempt to locate and rescue the immobilized unit (T) by using radio direction finding (RDF). The interaction between the jammer and the transmitter is visualized in Figure 6.1. To successfully triangulate the location of T, the search and rescue units need to establish communication with it via one of n pre-defined channels. All units are equipped with software-defined radios (SDRs), which allows them to switch the transmission/receiver channels on the fly. We assume that transmitters and jammers only choose one channel at a given time to increase their broadcast as well as jamming range. This assumption is justified by the fact that these are small, low powered mobile SDR units.

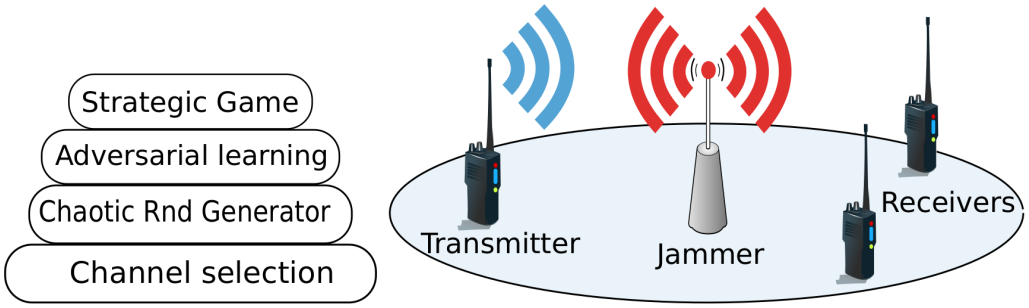


Figure 6.1: The interaction between the Transmitter, Jammer, and Receivers depicted as a multi-layered SDR system.

The problem of locating the transmitter T becomes challenging if an intelligent jammer J attempts to disrupt the communication through interference. Anti-jamming tactics for wireless networks have been an area of research for some time. For example, [81] propose channel surfing/hopping and spatial retreats as possible strategies for evading jamming attacks. In channel hopping, the signal transmitter would proactively switch the transmission channel according to some pattern (shared with the receiver). In contrast, in spatial retreats, the transmitter and receiver/s would attempt to move out of the jammer's effective transmission range. As the immobilized transmitter cannot physically move out of the jammer's transmission range, it could proactively switch between the n channels according to some static probability distribution p (i.e., channel hopping)

to evade the jammer.

We assume the jammer has spectrum sensing capabilities, which allows it to identify the channel the transmitter is using. By observing the channels over a finite period of time, the jammer would be able to approximate the static probability distribution p of the channels used by the transmitter with high probability, allowing it to interrupt transmissions more frequently. Therefore, the probability distribution over the n channels would need to be changed with time to successfully evade the jammer [115].

Changing the probability distribution in a truly random manner is not feasible as there needs to be a mechanism for the intended receivers (which is attempting to locate T) and for the transmitter to synchronize their channel-hopping schemes. Without a synchronized scheme, the probability of the transmitter and receivers using the same channels simultaneously would be low, thereby delaying the rescue. Therefore, the transmitter needs to change the probability distribution $p^T(k)$ over time step k according to some pseudo-random patterns. The receivers, who share the same pseudo-random generator keys as the transmitter, would be able to tune their receivers to the correct channels at any given time step. However, the jammer could attempt to learn the underlying pseudo-random patterns used by the transmitter by training a machine learning-based prediction model on the observed probability distributions.

If the jammer can closely predict the transmitter's pseudo-random generator functions, it will have the ability to disrupt the transmissions successfully. As the transmitter's frequency hopping pattern becomes evident with time, the jammer cannot learn the statistics of generator functions beforehand (i.e., offline learning). We propose a scheme in which the jammer learns the underlying patterns in an online setting (i.e., updating the prediction model as new data becomes available). This allows the transmitter to deceive the jammer into learning a false representation of the generator functions by introducing adversarial distortions on top of the generated probability distributions. Similarly, to prevent the transmitter from realizing that the jammer has successfully learned its generator functions (and react by switching the pseudo-random generators), the jammer would add adversarial distortions on top of its predictions. In this chapter, we apply a game-theoretic approach to the interaction between a transmitter and a sophisticated

jammer where the two players attempt to mislead each other by maliciously distorting data used by the other player to make decisions. We use several chaotic time series as generator functions for the transmitter and use state of the art Recurrent Neural Network (RNN) as a learner for the jammer and compare the effectiveness of the learner against a benchmark predictor. The combination of games, pseudo-random generators, and deep learning have not been used in jamming applications to the best of our knowledge.

The main contributions of the chapter are highlighted as follows:

- A novel adversarial (deep) learning approach to jamming, where the transmitter and jammer attempt to mislead each other by maliciously distorting data used by the other player to make decisions.
- Integrated use of pseudo-random generators and online machine (deep) learning methods in the context of a frequency hopping scheme.
- An overarching security game modeling the decision making in the context of adversarial (deep) learning and deception by the transmitter and receiver.

The chapter is organized as follows. Sections 6.2 and 6.3 formally define the problem and the methodology used. These are followed by Sections 6.4 and 6.5, which include the game formulation and simulation results, and the remarks of Section 6.6 conclude the chapter.

6.2 Problem Definition

We consider the interaction between a sophisticated jammer and a transmitter, where the transmitter T is attempting to establish communication with several receivers R while being interfered by the jammer J. The transmitter and the jammer can transmit over n distinct radio channels. At the network level, the only action available to T (J) is choosing a single channel out of the n available channels to transmit (cause interference on). The jammer may successfully disrupt the communication between the transmitter and the receivers by decreasing the *signal-to-interference-plus-noise* ratio (SINR) at the receivers.

However, this would then give away the location of the jammer, which may be an unacceptable risk. In this work, we assume that the transmitter/jammer would only use one channel at any given time to transmit/jam on.

Instead of using just one specific channel to transmit signals over a specific time interval (which would make jamming trivial for J), the transmitter switches between the available channels according to some probability distribution p^T. If the transmitter chooses a static probability distribution p^T throughout, the jammer can approximate p^T by observing the channels used by T over a specific time period. To prevent the jammer from approximating p^T, and thereby successfully carry out jamming attacks, the transmitter should periodically change p^T. The manner in which $p^T(k)$ is changed over time cannot be purely random, as that would make it very difficult for the receivers to listen to the correct channel at any given time to successfully locate T. But if the changing mechanism is easily perceived, the jammer would learn the new $p^T(k)$ with minimal effort. Therefore, the transmitter utilizes a chaotic pseudo-random number generator function $g_i(k) : \mathbb{R} \to \mathbb{R}$ for each channel $c_i, i \in [1..n]$ in order to create probability distributions that change periodically. Then, the probability of selecting channel c_i during the k^{th} interval is given by

$$p_i^T(k) = \frac{g_i(k)}{\sum_{j=1}^{n} g_j(k)}. \tag{6.1}$$

Making the probability distribution over the n channels during the k^{th} interval $p^T(k) = [p_1^T(k), p_2^T(k), \ldots, p_n^T(k)]$.

Note that the transmitter T has the freedom to decide the duration of each time interval in which a particular $p^T(k)$ is used. If the interval duration changes with each $k \in \mathbb{Z}_{\geq 0}$, the jammer can use a *change point detection* algorithm to identify that the probability distribution has changed and react accordingly [116, 117]. Therefore, without loss of generality, we assume that the time interval during which a particular probability distribution is used is fixed.

The jammer can observe the channel usage of the transmitter and approximate the probability distribution during the k^{th} interval by creating a histogram ($\hat{p}^T(k)$). In order to successfully interrupt T's transmission, the jammer would have to know in advance the probability distribution T would use in the next time interval. Therefore, the jammer

attempts to learn $g_i, i \in [1, n]$ by training prediction models for each g_i as new observations $\hat{p}^T(k)$ become available. For example, by using the observed $\hat{p}^T$ values from intervals 1 to $k-1$, the jammer would be able to predict the possible $p^T(k)$ distribution. The jammer attempting to learn the underlying generator functions can be posed as an *online time series forecasting problem*. The jammer then acts as the learner and attempts to learn the underlying function in an online setting as the transmitter's strategies become observable.

The transmitter, who also has spectrum sensing capabilities, observes the channels utilized by the jammer during the k^{th} interval and attempts to approximate the probability distribution J uses over the n-channels for jamming ($\hat{p}^J(k)$). If the observed $\hat{p}^J(k)$ does not diverge from $p^T(k)$ significantly (i.e., $\hat{p}^J(k) \approx p^T(k)$), it implies that the jammer has closely predicted $p^T(k)$ based on the transmitter's previous probability distributions. To hinder the online learners of the jammer, the transmitter T can mislead the jammer by introducing adversarial distortions to each $p_i^T(k)$. By adding adversarial distortions to each of the probabilities, the transmitter expects the jammer to learn a prediction model that is different from the actual generator functions used by the transmitter and predict a strategy $p^J(k)$ that is significantly different from $p^T(k)$. But the transmitter cannot greedily add significantly large distortions as the receivers who are attempting to locate the transmitter would be unaware of these distortion functions, and would continue to use the original pseudo-random generator functions to decide the channels they would listen to. As the transmitter's main objective is to be located without delay, adding adversarial distortions would have a detrimental effect.

If the jammer successfully predicts the p^T values over a period of time, the transmitter can react by either increasing the adversarial distortion intensity or by switching to different generator functions. If the transmitter switches the generator functions, the jammer would have to restart the learning processes. If the transmitter increases the adversarial distortions, it will make learning the true generator functions harder. Therefore, to prevent the transmitter from deviating from the usual generator patterns, the jammer could periodically add adversarial distortions to its own strategies p^J to mislead the transmitter into believing that the jammer has not learned the generator functions. Therefore, the in-

teraction between that jammer and transmitter has a two-way obfuscation nature where both players attempt to confuse each other.

The following specific assumptions are made to reduce the complexity of the transmitter-jammer interaction as a starting point and formalize it as a game:

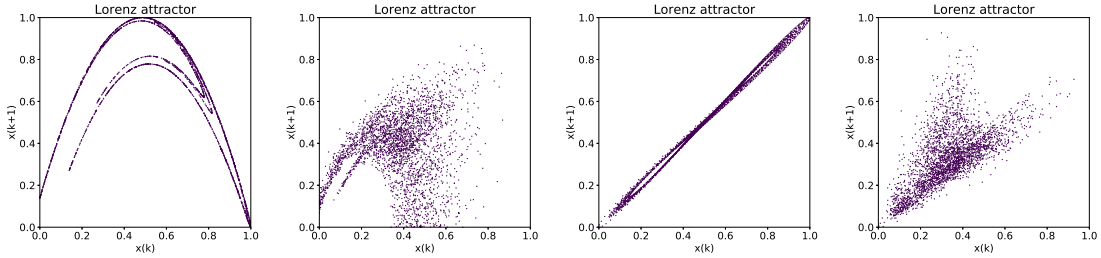
- We assume that both players can transmit using the same set of channels.
- The jammer and transmitter will only utilize a single channel at a time (with maximum power to maximize the range).
- The n channels are non-overlapping, therefore jamming on channel c_i would not cause interference on channel c_j where $i \neq j$.
- Jamming is modeled as a discrete event; it will either completely disrupt the communications or not.
- The transmitter has the flexibility of choosing different generators for each channel c_i. But once transmissions begin, the generator assignments are assumed to be fixed.
- While the transmitter can also decide the duration of the time interval to keep a particular probability distribution (i.e., $\Delta t \in [T_1, T_2], \Delta t \in \mathbb{Z}_{\geq 0}$), we assume the duration, in seconds, to be fixed for every time interval.

6.3 Methodology

This section presents the techniques used for generating the probability distributions and the learning algorithm used by the jammer.

The observations of the players, $\hat{p}^T$ and $\hat{p}^J$, are a combination of the other players' genuine probability distribution, p^J and p^T, and their respective adversarial distortions, d^T and d^J. As introducing adversarial distortions can lead to probability values becoming negative or greater than one, the resulting vector is projected onto the probability simplex as follows:

$$\tilde{p}_i^T(k) = \left[p_i^T(k) + d_i^T(k)\right]_p, \tag{6.2}$$



(a) original Henon (b) normalized (c) original Lorenz (d) normalized

Figure 6.2: The original phase graphs of the Henon and Lorenz attractors and the decorrelated graphs after combining with the other time series.

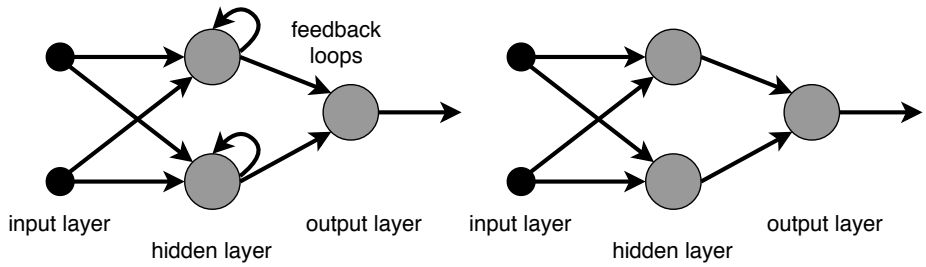
where $[.]_P$ is the projection onto the probability simplex Δ defined as

$$\Delta := \left\{ p \in \mathbb{R}^n : \sum_{i=0}^{n} p_i = 1 \text{ and } 0 \leq p_i \leq 1, \ \ \forall i \right\}. \tag{6.3}$$

6.3.1 Chaotic time series

While hardware-based random number generators are available, especially for non-civilian usage, pseudo-random generators are essential in the above scenario, for there needs to be a synchronization method (through pre-shared information) between the transmitter and the listeners. As a starting point, we select several chaotic time series, such as Rossler attractor, Lorenz attractor, and Henon attractor as the generator functions for the transmitter. Even though chaotic time series appear to unpredictable and show divergent behavior, they are governed by well-defined nonlinear equations [118].

At every step of the game, we obtain the corresponding time series values from each generator function and derive the probability distribution of the transmitter by normalizing the values (6.1). Figure 6.2 shows the phase diagrams, where the times series value at time $k+1$ is plotted against the time series value at time k, for Henon and Lorenz attractors. The phase diagrams indicate that the normalization process, which creates dependency among the chaotic attractors, makes each time series decorrelated. Therefore each time series appears to be more random. Due to this loss of correlation between adjacent time series values, the learning (prediction) task of the jammer becomes harder.



(a) A simple representation of Recurrent Neural Networks - RNN

(b) A simple representation of Feed Forward Neural Network

Figure 6.3: Difference between a traditional neural network and a Recurrent Neural Network (RNN).

6.3.2 Learning Algorithm

Deterministic dynamical systems generate chaotic time series. Therefore, to predict future values, the jammer has to learn the underlying nonlinear mappings of the time series. In the particular application scenario, we are concerned, the time series have to be learned solely from the past observations, without prior knowledge of the dynamical system.

As the future values of a time series depend on its previous values, we choose RNNs as the learners for the jammer. Unlike traditional feed-forward neural networks, RNNs have feedback connections within the layers (see Figure 6.3). It is these feedback connections that allow the past experience to be taken into account when predicting the subsequent steps in time series. We use Long Short Term Memory (LSTM) networks, a variant of RNNs that is not affected by the vanishing gradient problem as the jammer's learning algorithm [22].

6.4 Game Formulation

This section describes the transmitter-jammer interaction, modeled as a two-player static game between the jammer J and transmitter T, repeated over discrete time k. The myopic players interact over a sequence of steps (can be finite or infinite), and at each step, they solve the static game and determine the actions they would play during that time step. Although the games at each time step are independent, the learners of the jammer evolve

with time, enabling more accurate predictions as time goes on.

Since both players are equipped with SDRs, the possible actions available to both players at the network layer can be defined by choosing one of the n transmission channels to maximize range. For the transmitter, the probability distributions over the n channels would depend on the output of the generators as well as its adversarial distortions. Since the generator values are given at each step of the game, we focus on the mechanism used to generate adversarial distortions to formulate the game actions of T. Similarly, the jammer's probability distributions over the n channels would depend on the output of the learning algorithms as well as its own adversarial distortions. As the outputs of the learning algorithms are beyond the jammer's control, we focus on its adversarial distortion mechanism to formulate its game actions.

6.4.1 Player actions

Using adversarial distortions for the deception of the other player comes with consequences. For the transmitter, adding adversarial distortions means using a probability distribution over the n channels that is different from what the listeners are using. This leads to unsuccessful radio transmissions (without the jammer's influence) as the listeners would be listening on different channels at a given time. For the jammer, it means using a probability distribution that is different from what the learning algorithm predicts, as a form of deception. Using a distorted probability distribution results in more unsuccessful jamming attempts as both players would be using different probabilities over the channels.

The action sets of both players are defined as different distortion severity values using an arbitrary uniform quantization:

- transmitter: $a^T = \{0.1, 0.2, \ldots, 0.9\}$
- jammer: $a^J = \{0, 0.1, 0.2, \ldots, 0.9\}$.

The distortion severity value at the k^{th} time step is used to determine the adversarial distortion vectors $d^T(k)$ and $d^J(k)$ for each player obtained by sampling from a uniform distribution with different support sets. For example, the jammer would decide $a^J(k)$

and create the adversarial distortion vector $d^J(k)$ by randomly sampling from a uniform distribution over the interval $[-a^J(k), a^J(k)]$.

Since $d^T\left(a^T(k),k\right)$ and $d^J\left(a^J(k),k\right)$ are functions of the players' actions, the distorted probability distributions in (6.2), $\tilde{p}^T\left(a^T(k),k\right)$ and $\tilde{p}^J\left(a^J(k),k\right)$, also become dependent on the actions of the players.

Adversarial Deviation

Adversarial deviations are scalar values ($\phi^T\left(a^T(k),k\right)$ and $\phi^J\left(a^J(k),k\right)$) that indicate how much the originally intended probability distributions $p^T(k)$ and $p^J(k)$ deviate from the actually played distributions $\tilde{p}^T\left(a^T(k),k\right)$ and $\tilde{p}^J\left(a^J(k),k\right)$. As one possible metric, we choose root-mean-square error (RMSE) to measure the adversarial deviation caused by the distortions added during the k^{th} time step:

$$\phi^T\left(a^T(k),k\right) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(p_i^T(k) - \tilde{p}_i^T\left(a^T(k),k\right)\right)^2}, \tag{6.4}$$

$$\phi^J\left(a^J(k),k\right) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(p_i^J(k) - \tilde{p}_i^J\left(a^J(k),k\right)\right)^2}. \tag{6.5}$$

6.4.2 Utility functions

At each step of the game, the players need to make two decisions (i) the probability distribution over the n channels, and (ii) the amount of adversarial distortions to introduce. As Figures 6.4 and 6.5 depict, the two players use different approaches to address them. The transmitter obtains its undistorted probability distribution from the generator functions and uses the static game to decide the best adversarial distortion severity to use. The jammer observes the channel usage of the transmitter (i.e., $\hat{p}^T$) and trains n prediction models (as there is a unique generator function for each channel) to predict the next probability distribution the transmitter may use. Subsequently, similar to the transmitter, it decides on the severity of adversarial distortions based on the best response from the proposed game.

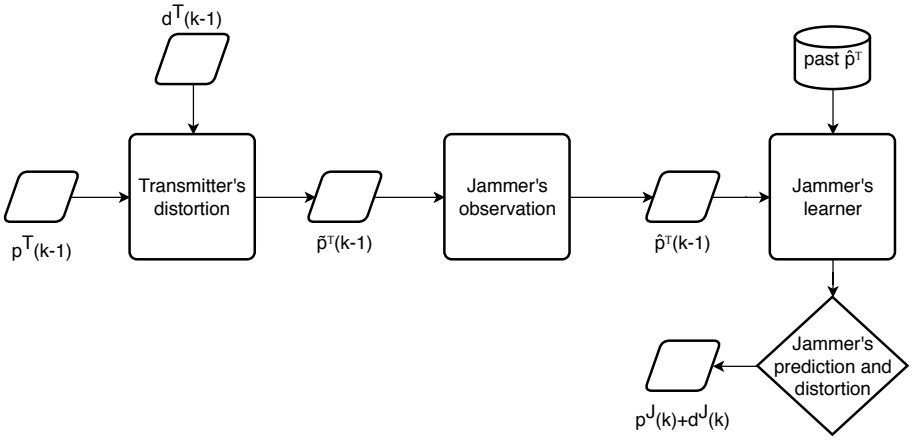


Figure 6.4: The decision making process of Jammer based on Transmitter's observed actions.

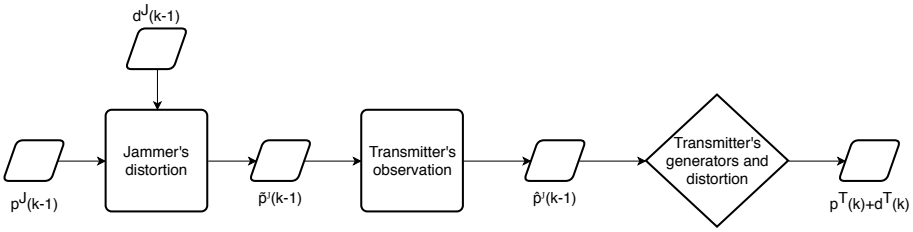


Figure 6.5: The decision making process of Transmitter based on Jammer's observed actions.

Note that the observed probabilities, $\hat{p}^T(k)$ and $\hat{p}^J(k)$, are obtained by counting the number of times the other player uses each of the n channels during the k^{th} time interval, i.e., creating a histogram. The observed probability distributions estimate the actually played probability distributions i.e., $\tilde{p}^T(a^T(k),k)$ and $\tilde{p}^J(a^J(k),k)$ closely as the length of the time interval increases. Since the observed probabilities depend on the distorted probability distributions, we define the observation function l as:

$$\hat{p}(k) = l(p(k), a(k), k), \tag{6.6}$$

making each $\hat{p}^T(k)$ and $\hat{p}^J(k)$ dependent on $a^T(k)$ and $a^J(k)$ respectively.

Utility of the transmitter:

For the transmitter to evade jamming, the jammer's estimation of transmitter's strategy $\hat{p}^T(k)$ should be significantly different from the actual generated strategy $p^T(k)$ during the k^{th} interval. Since $\hat{p}^T(k)$ is merely the observation of $\tilde{p}^T(k)$, the transmitter can also calculate it. We use the Kullback-Leibler divergence to calculate the statistical distance between two probability distributions as:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{6.7}$$

The transmitter's utility function is defined as:

$$\begin{aligned} U^T(k) &= D_{KL}\big(\hat{p}^T(k)||p^T(k)\big) - \phi^T\big(a^T(k),k\big) \\ &= D_{KL}\big(l(p^T(k),a^T(k),k)||p^T(k)\big) - \phi^T\big(a^T(k),k\big). \end{aligned} \tag{6.8}$$

Utility of the jammer:

Large adversarial distortions from the transmitter would make the jammer's learning algorithms learn incorrect representations of the transmitter's generator functions. Therefore, by adding adversarial distortions to its predicted values, the jammer expects to deceive the transmitter into using less severe adversarial distortions. The jammer, hence,

attempts to make $\tilde{p}^J(k)$ diverge from $\hat{p}^T(k)$ using its own adversarial distortions at the cost of jamming efficiency.

The jammer's utility function is defined as:

$$U^J(k) = D_{KL}\big(\hat{p}^T(k)||\tilde{p}^J\big(a^T(k),k\big)\big) - \phi^J\big(a^J(k),k\big). \tag{6.9}$$

In the above formulations ϕ^T and ϕ^J are the adversarial deviations explained in Section 6.4.1 that penalize large adversarial distortions.

6.4.3 Nash Equilibrium solution of the game

A bi-matrix game is represented by two $(m \times n)$ matrices, $A = \{a_{i,j}\}$ and $B = \{b_{i,j}\}$ where each pair of entries $(a_{i,j}, b_{i,j})$ denotes the outcome of the game corresponding to a particular pair of decisions made by the players. These entries in the matrix are populated by the players' utility functions, U^J and U^T. A pair of strategies $(a_{i^*,j^*}, b_{i^*,j^*})$ is said to be a non-cooperative Nash equilibrium outcome of the bi-matrix game if there is no incentive for any unilateral deviation by any one of the players. While it is possible to have a scenario where there is no Nash equilibrium solution in pure strategies, there would always be a Nash equilibrium solution in mixed strategies [98].

A player is said to use a mixed strategy when it chooses to randomize over a set of finite actions. Therefore, a mixed strategy can be defined as a probability distribution that gives each of the available actions a likelihood of being selected. At each step k of the game, the two players decide on their actions by computing the Nash equilibrium solutions in mixed strategies using their corresponding utility matrices. For example, Figure 6.6 shows the combined utility matrix for the two players when $k = 60$. In this particular step of the game, the best responses of both players yield a pure strategy Nash equilibrium solution, which is $a^J = 0.2$ and $a^T = 0.1$.

		transmitter - a^T			
		0.1	0.2	0.3	0.4
Jammer - a^J	0	(0.050,0.017)	(0.028,0.023)	(0.082,0.022)	(0.034,0.003)
	0.1	(0.038,0.005)	(0.019,0.001)	(0.073,0.016)	(0.009,0.011)
	0.2	**(0.096,0.049)**	(0.030,0.013)	(0.041,0.006)	(0.008,0.013)
	0.3	(0.048,0.083)	(0.091,0.012)	(0.062,0.000)	(0.004,0.005)
	0.4	(0.013,0.015)	(0.020,0.015)	(0.042,0.006)	(0.089,0.011)

Figure 6.6: The utility matrix of the game depicting the outcomes. The jammer is the row player and the transmitter is the column player and payoffs are displayed as (jammer utility, transmitter utility).

6.5 Simulation Results

In the simulations, for simplicity, the number of channels available for the transmitter and jammer is set to be three. Therefore, three time series are used to generate the probabilities for each of the channels from the transmitter's perspective. We use three popular chaotic time series, Rossler attractor, Lorenz attractor, and Henon attractor as the generator functions (i.e., $g_i(k), i \in [1, 2, 3]$) for the three channels of the transmitter.

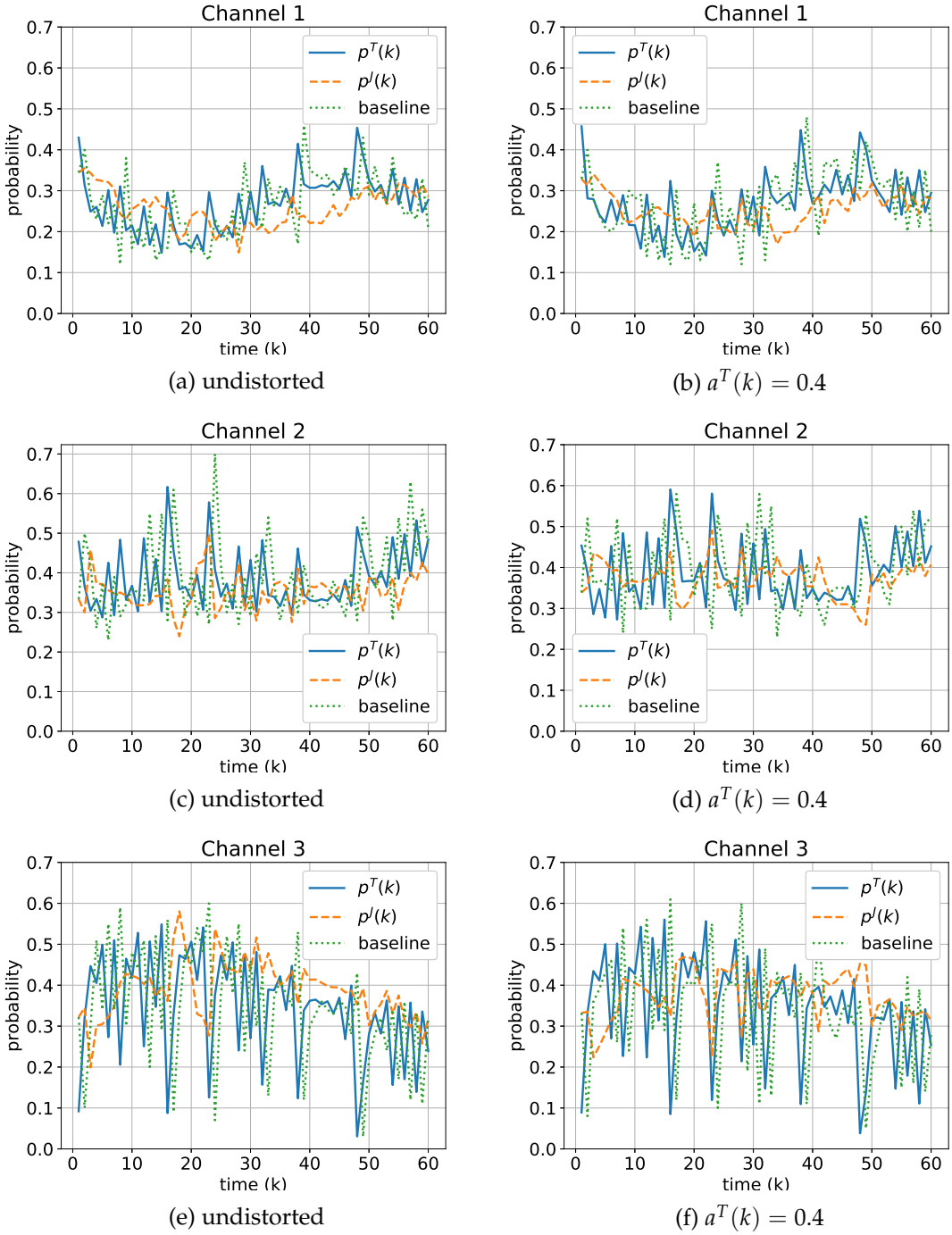
At the start of the game (i.e., $k = 1$ and $k = 2$), where there are no prior observations, the jammer uses normalized probability distributions sampled from $\mathcal{U}(0, 1)$ to use over the n channels. Subsequently, after the learners commence the learning process, the predicted probability distributions $p^J(k)$ are used. During the initial stages of the game, where there are very few observations, the predictions of the online learning algorithms would not be accurate as machine learning-based prediction models need sufficient data to learn the correct underlying patterns. After a certain threshold, the models are expected to stabilize and provide accurate predictions.

We train a four neuron LSTM model for each channel c_i using the observed probability distributions $\hat{p}^T$ as the training data. The models are trained in an online setting where they learn as new observations become available. The performance of the LSTMs is compared against a baseline persistence algorithm, which predicts the observation of the previous time step as the prediction for the current time step (i.e., $p^j(k) = \hat{p}^T(k-1)$). Since it's not possible to provide an output for $k = 1$, the algorithm provides a normal-

ized random probability distribution sampled from $\mathcal{U}(0, 1)$.

Figure 6.7 shows the probability distributions used by the transmitter, the jammer, and the baseline algorithm over the three channels under two conditions. The left column shows the probability distributions when the transmitter is not introducing distortions ($a^T(k) = 0, \forall k \in \mathbb{Z}_{\geq 0}$), and the right column shows the distributions when $a^T(k)$ is fixed at 0.4 for all time steps. The transmitter uses the Rossler attractor for channel 1, Lorenz attractor for channel 2, and Henon attractor for the third channel. As Figure 6.7 shows, the Henon attractor exhibits more randomness compared to the former two attractors. Comparing the undistorted probability distributions with the distorted probability distributions show that by introducing adversarial distortions, the transmitter can make each time series behave more abruptly, deviating from the patterns governed by the underlying equations.

Figure 6.8 shows the root-mean-square error (RMSE) between the transmitter's distorted probability distribution ($\tilde{P}^T(k)$) and the jammer's predicted probability distribution ($p^J(k)$), and the baseline persistence algorithm under different adversarial distortion severities. As the LSTMs are trained online, during the early stages of the game, they will not be able to make accurate predictions due to the lack of data. But as more data becomes available, the LSTMs are able to learn the underlying patterns of the data make more accurate predictions. Therefore, we show the RMSE values for the last 50% of the time series data as they reflect the true prediction capabilities of the LSTMs. We observe that the performance of the LSTMs in predicting the time series is comparatively higher than the baseline persistence algorithm. The performance differences are quite significant on channels 2 and 3, where the two chaotic functions used appear more random compared to that of channel 1. In summary, the above initial experiments demonstrate that (i) the transmitter can introduce more randomness to the time series by introducing adversarial distortions, (ii) the LSTMs' predictions are more accurate than the baseline persistence algorithm in all test cases.

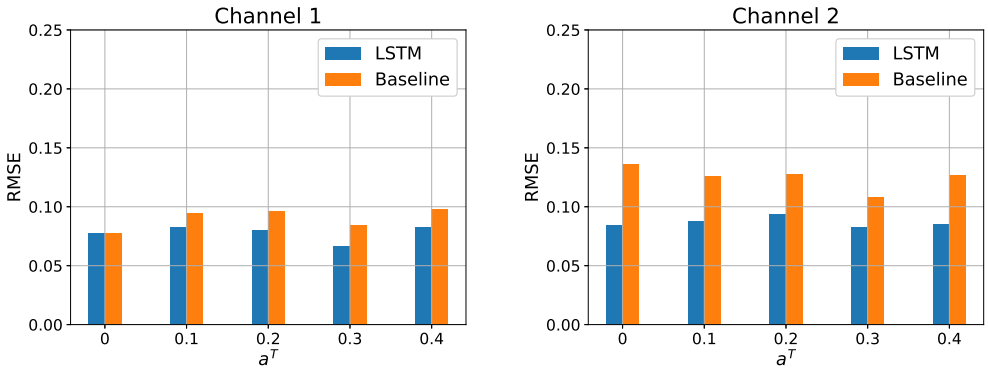


(a) undistorted (b) $a^T(k) = 0.4$

(c) undistorted (d) $a^T(k) = 0.4$

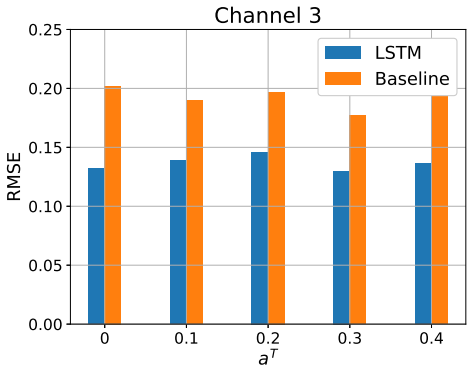
(e) undistorted (f) $a^T(k) = 0.4$

Figure 6.7: The probability distributions used by the transmitter, jammer with the deep learner, and a jammer with the baseline algorithm on the three channels. Figures on the left show the probability values when the transmitter is not using adversarial distortions, while the figures on the right show the probability values when $a^T(k)$ is set to 0.4.



(a) Prediction error on channel 1

(b) Prediction error on channel 2



(c) Prediction error on channel 3

Figure 6.8: RMSE values of the LSTMs and the baseline algorithm for the probability distributions used on channels 1 to 3. The RMSE is calculated for the last 50% of each time series.

6.6 Summary

This chapter models the interaction between a sophisticated jammer and a transmitter in a search and rescue scenario as a stochastic game. The transmitter, who has a pre-shared key with the rescuers, uses stochastic channel hopping according to some pseudo-random generators as the anti-jamming mechanism. The jammer attempts to learn the above pseudo-random generators by using online machine (deep) learning methods. To prevent the jammer from learning the pseudo-random generators, the transmitter introduces adversarial distortions. Similarly, to deceive the transmitter into thinking that the jammer has not been able to learn the underlying patterns, the jammer also distorts its jamming patterns on purpose.

The empirical results suggest that the transmitter can hinder the learning capabilities of the jammer by using adversarial distortions. The prediction performance of the LSTMs against different adversarial distortion levels is higher than the baseline persistence algorithm.

Chapter 7
Conclusion and Future Work

In this thesis, we investigated the performance degradation of machine learning algorithms such as SVMs and regression estimators in the presence of maliciously distorted training data and features. In this chapter, we first summarize the contributions of this thesis and their impact. Then, we discuss some limitations of the work and potential directions for future work.

7.1 Summary of Contributions

THE aim of this thesis was to investigate the effects of adversarial perturbations on existing machine learning algorithms and introduce novel defenses to increase their attack resistance. The numerous experiments conducted in this thesis confirmed that algorithms such as SVMs, OCSVMs, and regression algorithms, although used widely, are significantly affected by adversarial perturbations. Another finding to emerge from this study is that although defense algorithms that make assumptions regarding the attacker/attack increase the attack resistance of learners, their practicality may be limited.

In Chapter 3, we commenced our investigation on the effects of poisoning attacks on machine learning algorithms by exploring the use of data projections to lower-dimensional spaces as a defense against poisoning attacks. First, we considered the use of randomly drawn directions to project the data. Although the randomness provided a layer of security due to its unpredictability, we found that it was not sufficient to avert the effects of a sophisticated attacker.

Therefore we introduced two ranking indices that measure the goodness of a projection direction in order to guide the transformation to lower-dimensional spaces. We then incorporated the indices into a framework that uses SVMs and game theory. Through

experiments on several datasets, we discovered that SVMs and OCSVMs could be significantly affected if an adversary can manipulate the data on which they are trained. However, it was shown that by using the projection-based defense framework, the above algorithms became resilient to integrity attacks.

We also theoretically analyzed the effects of adversarial perturbations on the separating margins of separation of SVMs and OCSVMs. Following this analysis, we introduced an upper-bound for the length of the weight vector when there are no adversarial distortions, using the length of the weight vector when adversarial distortions are present.

The results in Chapter 3 show:

- SVMs and OCSVMs can be significantly affected if an adversary can manipulate the data on which they are trained.
- For each dataset, there is at least one dimensionality and projection direction that can accurately identify adversarial samples that would have been missed by an SVM or an OCSVM in the input space.
- Game theory can be used in adversarial learning applications to predict the behavior of an attacker and select a defense strategy for the learner.

The next chapter of the thesis was concerned with using the Local Intrinsic Dimensionality (LID) of data samples to distinguish adversarial samples from normal samples and thereby subvert malicious attacks. In this work, we considered the impact of two types of poisoning attacks, namely, label flip attacks and input perturbation attacks, and their implications on SVMs. We found that the LID estimation that uses Euclidean distances did not exhibit a strong capability to distinguish adversarial samples from normal samples. Therefore we introduced a novel LID approximation (K-LID) that makes use of the kernel matrix to obtain the LID values. We also introduced three label dependent variations of LID that can be used in situations where an attacker alters the training labels.

We then demonstrated that a weighted SVM that uses K-LID to de-emphasize the effects of suspected adversarial samples on the learned model was able to minimize the

impact of an adversary's attacks compared to the state of the art alternatives. We also observed that the proposed defense algorithm could be deployed in a distributed manner with significant reductions in communication overhead.

The results in Chapter 4 show:

- Sophisticated label flip attacks can inflict significant damage on SVMs compared to attacks that randomly flip labels.
- LID estimation in input space is not sufficient to distinguish poisoned samples from normal samples.
- However, K-LID, which calculates the LIDs using kernel matrix values, has sufficient distinguishing power.
- De-emphasizing the effect of suspected samples on the learned model gives better performance than methods that attempt to make all samples contribute to the decision process.

In Chapter 5, we first delved into learning linear estimators in the presence of poisoned training data. We found that maliciously perturbed training samples were able to degrade the prediction capabilities of regression models significantly. Most importantly, it was discovered that robust regression models, designed to withstand stochastic noise, were unable to withstand the effects of poisoned data in datasets with a large number of dimensions.

To circumvent the poisoning attacks, we introduced a new LID based measure called N-LID, which considers the LID values of the sample of interest as well as its k nearest neighbors. As N-LID was able to distinguish poisoned samples from normal samples, we incorporated this measure into ridge regression and neural network regression models as sample weights. Through experiments on several datasets, we demonstrated that this defense algorithm not only had lower MSE values but also was highly efficient in the majority of the test cases. We also introduced N-LID CVX, a defense that makes no assumptions of the attack, proving that it is possible to have defenses that are practical and effective against poisoning attacks.

The results in Chapter 5 show:

- Sophisticated poisoning attacks can reduce the accuracy of linear estimators.
- Robust regression learners are unable to withstand poisoning attacks on datasets with a large number of dimensions, whereas adversarial learners can.
- The proposed N-LID based defenses outperform other defenses considered in terms of prediction accuracy and running times in most of the test cases.
- N-LID CVX, which makes no assumptions of the attack, performs similarly to the baseline N-LID LR defense.

Finally, in Chapter 6, we considered adversarial distortions to support anti-jamming by deceiving a jammer's learning algorithms in a game-theoretic framework. In the search and rescue scenario considered, the transmitter used a stochastic channel hopping scheme based on pseudo-random generators as the anti-jamming mechanism. The jammer, in order to predict the transmitter's channel usage, used online machine learning methods. In this work, the transmitter utilized adversarial distortions to prevent the jammer from learning the pseudo-random generators.

The results in Chapter 6 show:

- Through adversarial distortions, the transmitter can introduce randomness to the time series and thereby increase the probability of evading the jammer.
- The LSTM learner had accurate predictions compared to the baseline persistence algorithm in all test cases under adversarial distortions.

In summary, this thesis has provided a more in-depth insight into the problem of training machine learning algorithms in the presence of adversarial perturbations. At present, we see primitive forms of adversarial attacks being carried out against machine learning-based systems such as spam filters and antivirus software. As machine learning is being utilized in many domains for decision making, in the near future, we can expect to see more sophisticated attacks, such as the ones we have considered in this thesis in

practice. Therefore, the insights gained from this thesis may be of assistance to defend such machine learning-based decision-making systems in the future.

7.2 Future Research Directions

The scope of this thesis was limited in terms of the number of machine learning algorithms and the types of attacks considered. Therefore, there are several questions in need of further investigation, as explained below.

Although we used projecting to lower-dimensional spaces as a defense against integrity attacks in Chapter 3, we note that the same defense cannot be used against label flipping attacks that we considered in Chapter 4. In a label flipping attack, the adversary strategically flips a subset of training labels, resulting in certain data samples from a particular class having labels of the opposite class. After such an attack, it is not possible to find a projection that separates data points from the two classes.

In Chapters 3 and 6 we considered Nash equilibrium as the game-theoretic solution concept for the adversary-learner interactions that we considered. However, there are other solution concepts that are weaker/stronger than the Nash equilibrium. Since solution concepts explain the decisions made by the players, using further solution concepts such as Stackelberg, mini-max regret, and dominant strategy equilibrium would provide new insights into this problem. Equilibrium solutions can be used to predict the behavior of adversaries in cybersecurity applications; therefore, further research that explore solution concepts beyond the Nash equilibrium would provide new insights into designing robust learning systems.

In our work that used LID for distinguishing adversarial samples from normal samples (i.e., Chapters 4 and 5), we introduced a novel LID estimator called K-LID (kernel LID) and a LID based measure called N-LID (neighborhood LID ratio). Both metrics resulted in attack resilient learners. Designing new LID estimators and LID based measures could lead to more secure machine learning algorithms; therefore, it is worth investigating.

In Chapter 5, we focused on the impact of poisoning attacks on linear regression mod-

els. From our investigation, we found that due to the amount of redundancy present in linear datasets, defense algorithms can remove/de-emphasize a large subset of the training data and still retain a high prediction accuracy. However, for non-linear datasets, we expect the amount of redundancy present in data to be relatively small. Therefore, a natural progression of this work is to analyse the effects of poisoning attacks on non-linear regression models and how N-LID would perform in such scenarios.

In Chapters 4 and 5, we considered poisoning attacks where the attacker's goal was to affect the prediction results indiscriminately, i.e., decrease the overall prediction capabilities. However, as explained in Chapter 1, in attacks on integrity and attacks on availability, the attacker's goal is to cause specific mispredictions. In practice, the latter attack types may have a larger impact on learners compared to general poisoning attacks. This would be a fruitful area for further work.

Finally, in Chapter 6, we considered a transmitter that uses a channel hopping scheme based on pseudo-random generators to evade jamming. Pseudo-random generators produce sequences of numbers that seem random, yet have an underlying structure. While cryptographically secure pseudo-random number generators are better suited for achieving the security goals in application scenario considered, we opted to use pseudo-random generators as they allowed the transmitter to synchronize their channel hopping scheme with the receivers. Therefore, as future research, it would be interesting to compare stronger random number generators that are used in cryptography for this application. Formulating different games (e.g., over a finite horizon or with additional solution concepts) could be explored in further research.

More broadly, research is also needed to assess the transferability of the defense algorithms we introduced in this thesis across the available machine learning algorithms, such as deep neural networks. A further study could also investigate the effect of combined attacks (using several attack algorithms on different subsets of data at the same time) on existing machine learning algorithms and defenses. However, we speculate that the most potent attack would dominate and therefore obscure the effects of the other attacks. Further research could also be conducted to determine the effectiveness of the proposed defenses in other practical applications such as smart grids and intrusion de-

tection systems.

Bibliography

[1] S. Weerasinghe, S. M. Erfani, T. Alpcan, and C. Leckie, "Support vector machines resilient against training data integrity attacks," *Pattern Recognition*, vol. 96, p. 106985, 2019.

[2] P. S. Weerasinghe, S. M. Erfani, T. Alpcan, C. Leckie, and M. Kuijper, "Unsupervised Adversarial Anomaly Detection Using One-Class Support Vector Machines," in *23rd International Symposium on Mathematical Theory of Networks and Systems*, 2018, pp. 34–37.

[3] S. Weerasinghe, S. M. Erfani, T. Alpcan, C. Leckie, and J. Riddle, "Detection of anomalous communications with SDRs and unsupervised adversarial learning," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 469–472.

[4] S. Weerasinghe, T. Alpcan, S. M. Erfani, and C. Leckie, "Defending support vector machines against data poisoning attacks," *Submitted to Transactions on Information Forensics and Security*, 2020.

[5] S. Weerasinghe, S. M. Erfani, T. Alpcan, and C. Leckie, "Defending regression learners against poisoning attacks," *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[6] S. Weerasinghe, T. Alpcan, S. M. Erfani, C. Leckie, P. Pourbeik, and J. Riddle, "Deep learning based game-theoretical approach to evade jamming attacks," in *International Conference on Decision and Game Theory for Security*. Springer, 2018, pp. 386–397.

[7] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[8] I. Kononenko, "Machine Learning for Medical Diagnosis: History, State of the Art and Perspective," *Artificial Intelligence in medicine*, vol. 23, no. 1, pp. 89–109, 2001.

[9] Y. Chen, Y. Tan, and D. Deka, "Is Machine Learning in Power Systems Vulnerable?" in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–6.

[10] W. Liu and S. Chawla, "A Game Theoretical Model for Adversarial Learning," in *2009 IEEE International Conference on Data Mining Workshops*, 2009, pp. 25–30.

[11] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial Machine Learning," in *4th ACM Workshop on Security and Artificial Intelligence*, 2011, pp. 43–58.

[12] H. Xiao, H. Xiao, and C. Eckert, "Adversarial Label Flips Attack on Support Vector Machines." in *20th European Conference on Artificial Intelligence*, 2012, pp. 870–875.

[13] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security & Privacy*, vol. 14, no. 3, pp. 68–72, 2016.

[14] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[15] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, 2018.

[16] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Advances In Neural Information Processing Systems*, 2017, pp. 3517–3529.

[17] Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi, "Adversarial Support Vector Machine Learning," in *18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1059–1067.

[18] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[19] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[20] M. E. Houle, "Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications," in *Similarity Search and Applications*, C. Beecks, F. Borutta, P. Kröger, and T. Seidl, Eds., 2017, pp. 64–79.

[21] Houle, Michael E., "Local intrinsic dimensionality II: multivariate analysis and distributional support," in *International Conference on Similarity Search and Applications*, 2017, pp. 80–95.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*, 1st ed. Boston, MA, USA: Auerbach Publications, 2011.

[24] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[25] A. Rahimi and B. Recht, "Random Features for Large-Scale Kernel Machines," in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 1177–1184.

[26] S. M. Erfani, M. Baktashmotlagh, S. Rajasegarar, S. Karunasekera, and C. Leckie, "R1SVM: A Randomised Nonlinear Approach to Large-Scale Anomaly Detection," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 432–438.

[27] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[28] M. M. Krell and H. Wöhrle, "New one-class classifiers based on the origin separation approach," *Pattern Recognition Letters*, vol. 53, pp. 93–99, 2015.

[29] P. B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support Vector Method for Novelty Detection," in *Advances in Neural Information Processing Systems (NIPS)*, 2000, pp. 582–588.

[30] D. M. J. Tax and R. P. W. Duin, "Support Vector Data Description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[31] D. Wang and Y. Zhou, "Distributed support vector machines: An overview," in *2012 24th Chinese Control and Decision Conference (CCDC)*. IEEE, 2012, pp. 3897–3901.

[32] T. Alpcan and C. Bauckhage, "A distributed machine learning framework," in *48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009*. IEEE, 2009, pp. 2546–2551.

[33] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[34] R. Zhang and Q. Zhu, "Secure and resilient distributed machine learning under adversarial environments," in *2015 18th International Conference on Information Fusion*. IEEE, 2015, pp. 644–651.

[35] R. Zhang and Q. Zhu, "A game-theoretic analysis of label flipping attacks on distributed support vector machines," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2017, pp. 1–6.

[36] K. Leyton-Brown and Y. Shoham, *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*, 1st ed. Morgan and Claypool Publishers, 2008.

[37] T. Alpcan and T. Basar, *Network Security: A Decision and Game-Theoretic Approach*, 1st ed., 2010.

[38] Y. Vorobeychik and M. Kantarcioglu, "Adversarial machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–169, 2018.

[39] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 387–402.

[40] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks Against Support Vector Machines," in *29th International Coference on International Conference on Machine Learning*, 2012, pp. 1467–1474.

[41] N. Dalvi, P. Domingos, S. Sanghai, D. Verma *et al.*, "Adversarial classification," in *10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2004, pp. 99–108.

[42] B. Li, Y. Vorobeychik, and X. Chen, "A general retraining framework for scalable adversarial classification," *arXiv preprint arXiv:1604.02606*, 2016.

[43] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," *CoRR*, vol. abs/1511.07528, 2015.

[44] B. Biggio, I. Corona, B. Nelson, B. I. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli, "Security evaluation of support vector machines in adversarial environments," in *Support Vector Machines Applications*. Springer, 2014, pp. 105–153.

[45] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Asian Conference on Machine Learning*, 2011, pp. 97–112.

[46] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomputing*, vol. 160, pp. 53–62, 2015.

[47] M. Kloft and P. Laskov, "Security analysis of online centroid anomaly detection," *Journal of Machine Learning Research*, vol. 13, pp. 3681–3724, 2012.

[48] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Pattern based anomalous user detection in cognitive radio networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, 2015, pp. 5605–5609.

[49] R. Laishram and V. V. Phoha, "Curie: A method for protecting SVM Classifier from Poisoning Attack," *arXiv preprint arXiv:1606.01584*, 2016.

[50] N. X. Vinh, S. Erfani, S. Paisitkriangkrai, J. Bailey, C. Leckie, and K. Ramamohanarao, "Training robust models using Random Projection," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 531–536.

[51] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter, "Scaling provable adversarial defenses," in *Advances in Neural Information Processing Systems*, 2018, pp. 8400–8409.

[52] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.

[53] D. F. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artificial Intelligence Review*, vol. 33, no. 4, pp. 275–306, 2010.

[54] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[55] G. L. Libralon, A. C. P. de Leon Ferreira, A. C. Lorena *et al.*, "Pre-processing for noise detection in gene expression classification data," *Journal of the Brazilian Computer Society*, vol. 15, no. 1, pp. 3–11, 2009.

[56] J. Zhang and Y. Yang, "Robustness of regularized linear classification methods in text categorization," in *26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, 2003, pp. 190–197.

[57] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1146–1151, 2013.

[58] C.-f. Lin *et al.*, "Training algorithms for fuzzy support vector machines with noisy data," *Pattern Recognition Letters*, vol. 25, no. 14, pp. 1647–1656, 2004.

[59] W. An and M. Liang, "Fuzzy support vector machine based on within-class scatter for classification problems with outliers or noises," *Neurocomputing*, vol. 110, pp. 101–110, 2013.

[60] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in Neural Information Processing Systems*, 2013, pp. 1196–1204.

[61] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 447–461, 2016.

[62] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. John Wiley & Sons, 2005, vol. 589.

[63] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics*. Springer, 1992, pp. 492–518.

[64] H. Thiel, "A rank-invariant method of linear and polynomial regression analysis, Part 3," in *Proceedings of Koninalijke Nederlandse Akademie van Weinenschatpen A*, vol. 53, 1950, pp. 1397–1412.

[65] P. K. Sen, "Estimates of the regression coefficient based on Kendall's tau," *Journal of the American Statistical Association*, vol. 63, no. 324, pp. 1379–1389, 1968.

[66] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[67] H. Xu, C. Caramanis, and S. Mannor, "Outlier-robust PCA: The high-dimensional case," *IEEE Transactions on Information Theory*, vol. 59, no. 1, pp. 546–572, 2012.

[68] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12 103–12 117, 2018.

[69] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[70] Y. Chen, C. Caramanis, and S. Mannor, "Robust sparse regression under adversarial corruption," in *International Conference on Machine Learning*, 2013, pp. 774–782.

[71] J. Feng, H. Xu, S. Mannor, and S. Yan, "Robust logistic regression and classification," in *Advances in Neural Information Processing Systems*, 2014, pp. 253–261.

[72] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *International Conference on Machine Learning*, 2015, pp. 1689–1698.

[73] C. Liu, B. Li, Y. Vorobeychik, and A. Oprea, "Robust linear regression against training data poisoning," in *10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 91–102.

[74] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[75] L. Tong, S. Yu, S. Alfeld, and Y. Vorobeychik, "Adversarial regression with multiple learners," *arXiv preprint arXiv:1806.02256*, 2018.

[76] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models," in *30th AAAI Conference on Artificial Intelligence*, 2016.

[77] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," *1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pp. 1–9, 2008.

[78] S. Vadlamani, B. Eksioglu, H. Medal, and A. Nandi, "Jamming attacks on wireless networks: A taxonomic survey," *International Journal of Production Economics*, vol. 172, pp. 76–94, 2016.

[79] K. Dabcevic, A. Betancourt, L. Marcenaro, and C. S. Regazzoni, "Intelligent cognitive radio jamming - a game-theoretical approach," *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 1, p. 171, Dec 2014.

[80] Q. Zhu, W. Saad, Z. Han, H. V. Poor, and T. Başar, "Eavesdropping and jamming in next-generation wireless networks: A game-theoretic approach," in *Military Communications Conference, 2011-MILCOM 2011*. IEEE, 2011, pp. 119–124.

[81] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel Surfing and Spatial Retreats: Defenses Against Wireless Denial of Service," in *3rd ACM Workshop on Wireless Security*, ser. WiSe '04, 2004, pp. 80–89.

[82] R. D. Pietro and G. Oligeri, "Jamming mitigation in cognitive radio networks," *IEEE Network*, vol. 27, no. 3, pp. 10–15, May 2013.

[83] S. Bhattacharya and T. Basar, "Optimal strategies to evade jamming in heterogeneous mobile networks," in *Workshop on Search and Pursuit-Evasion*, 2010.

[84] Q. Zhu, H. Li, Z. Han, and T. Basar, "A stochastic game model for jamming in multi-channel cognitive radio systems," in *2010 IEEE International Conference on Communications (ICC)*. IEEE, 2010, pp. 1–6.

[85] B. Wang, Y. Wu, K. J. R. Liu, and T. C. Clancy, "An anti-jamming stochastic game for cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 4, pp. 877–889, April 2011.

[86] O. Puñal, I. Aktaş, C. J. Schnelke, G. Abidin, K. Wehrle, and J. Gross, "Machine learning-based jamming detection for IEEE 802.11: Design and experimental evaluation," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, June 2014, pp. 1–10.

[87] L. Cong and S. Songgeng, "Chaotic frequency hopping sequences," *IEEE Transactions on Communications*, vol. 46, no. 11, pp. 1433–1437, 1998.

[88] H. Hu, L. Liu, and N. Ding, "Pseudorandom sequence generator based on the Chen chaotic system," *Computer Physics Communications*, vol. 184, no. 3, pp. 765–768, 2013.

[89] M. Casdagli, "Nonlinear prediction of chaotic time series," *Physica D: Nonlinear Phenomena*, vol. 35, no. 3, pp. 335–356, 1989.

[90] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Physical Review Letters*, vol. 59, pp. 845–848, Aug 1987.

[91] M. Han, J. Xi, S. Xu, and F.-L. Yin, "Prediction of chaotic time series based on the recurrent predictor neural network," *IEEE Transactions on Signal Processing*, vol. 52, no. 12, pp. 3409–3416, Dec 2004.

[92] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid Forecasting of Chaotic Processes: Using Machine Learning in Conjunction with a Knowledge-Based Model," *CoRR*, vol. abs/1803.04779, 2018.

[93] Y. Ma and G. Guo, *Support vector machines applications*. Springer, 2014.

[94] H. Xu, C. Caramanis, and S. Mannor, "Robustness and regularization of support vector machines," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1485–1510, 2009.

[95] T. Alpcan, B. I. P. Rubinstein, and C. Leckie, "Large-scale strategic games and adversarial machine learning," in *IEEE Conference on Decision and Control (CDC)*, 2016, pp. 4420–4426.

[96] J. C. Bezdek and N. R. Pal, "Some new indexes of cluster validity," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 301–315, 1998.

[97] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary Mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[98] J. Nash, "Non-cooperative Games," *Annals of Mathematics*, vol. 54, pp. 286–295, 1951.

[99] T. Basar and G. J. Olsder, *Dynamic noncooperative game theory*. Siam, 1999, vol. 23.

[100] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. N. R. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[101] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New Support Vector Algorithms," *Neural computation*, vol. 12, no. 5, pp. 1207–1245, May 2000.

[102] S. Paul, C. Boutsidis, M. Magdon-Ismail, and P. Drineas, "Random Projections for Linear Support Vector Machines," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 4, p. 22, 2014.

[103] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, 2008, p. 60.

[104] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. Erfani, S. Xia, S. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," in *35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80, 2018, pp. 3355–3364.

[105] X. Yang, Q. Song, and Y. Wang, "A weighted support vector machine for data classification," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, no. 05, pp. 961–976, 2007.

[106] L. Amsaleg, O. Chelly, T. Furon, S. Girard, M. E. Houle, K.-I. Kawarabayashi, and M. Nett, "Estimating local intrinsic dimensionality," in *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 29–38.

[107] M. E. Houle, "Dimensionality, discriminability, density and distance distributions," in *2013 IEEE 13th International Conference on Data Mining Workshops*. IEEE, 2013, pp. 468–473.

[108] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[109] H. C. Koh, G. Tan *et al.*, "Data mining applications in healthcare," *Journal of Healthcare Information Management*, vol. 19, no. 2, p. 65, 2011.

[110] I. Bose and R. K. Mahapatra, "Business data mining - a machine learning perspective," *Information & Management*, vol. 39, no. 3, pp. 211–225, 2001.

[111] R. H. Myers, *Classical and modern regression with applications*. Duxbury press Belmont, CA, 1990, vol. 2.

[112] V. Arzamasov, K. Böhm, and P. Jochem, "Towards concise models of grid stability," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–6.

[113] D. Kibler, D. W. Aha, and M. K. Albert, "Instance-based prediction of real-valued attributes," *Computational Intelligence*, vol. 5, no. 2, pp. 51–57, 1989.

[114] C. Laufer, *The Hobbyist's Guide to the RTL-SDR: Really Cheap Software Defined Radio*, 2015.

[115] K. Pelechrinis, C. Koufogiannakis, and S. V. Krishnamurthy, "On the efficacy of frequency hopping in coping with jamming attacks in 802.11 networks," *IEEE Transactions on Wireless Communications*, vol. 9, no. 10, pp. 3258–3271, 2010.

[116] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, 2013.

[117] Y. Kawahara and M. Sugiyama, "Change-point detection in time-series data by direct density-ratio estimation," in *2009 SIAM International Conference on Data Mining*. SIAM, 2009, pp. 389–400.

[118] G. Boeing, "Visual analysis of nonlinear dynamical systems: Chaos, fractals, self-similarity and the limits of prediction," *Systems*, vol. 4, no. 4, p. 37, 2016.