TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Sicherheit in der Informatik

Adversarial and Secure Machine Learning

Huang Xiao

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr. Helmut Seidl

Prüfer der Dissertation:

1. Prof. Dr. Claudia Eckert
2. Prof. Fabio Roli, Ph.D.

Die Dissertation wurde am 27.10.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 14.02.2017 angenommen.

Acknowledgements

First and foremost, I would like to thank my doctoral advisor Prof. Dr. Claudia Eckert. It has been a great honor to be one of her doctoral students, and this dissertation would be never possible without her insightful guidance and support. Over the many years, she impacted her wisdom to me with regards to both research and philosophy, generously and heartfully. Her remarkable views and visions on my research formed the most fundamental ground of the success of my doctoral work. During the tough time, I sincerely appreciate all her ideas, time, funding to make my Ph.D unique and fruitful. I also admire the excellent example she has provided as a successful researcher and professor, and she is the most open minded person I ever met in my life.

I would also like to appreciate all contributions and supervisions from other peers during my doctoral study. Dr. Thomas Stibor and Dr. Hasan Ibne Akram, they were my first mentors providing me immense guidance and help in research of machine learning at the early stage of my Ph.D. They are both great researchers and professionals who have built great success in their career. Also, in particular, I am thankful to two of my major collaborators, Dr. Han Xiao and Dr. Battista Biggio. It was absolutely my fortune and honor to be able to ever work with them. I appreciate all of their contributions, comments, critics and motivations. Both of them are brilliant researchers and collaborators. Of course, there are a number of other colleagues and friends, who ever helped and supported my doctoral work, they are certainly of equal importance and I appreciate them all disregarding that I can not list all the names here.

Moreover, I would also thank Prof. Fabio Roli from University of Cagliari, previous colleagues from Alibaba Inc., and also Prof. Michael Berstein from Stanford University. Without their help and recognition, I would never have the opportunities to conduct my visiting research in their organization and universities. Strongly encouraged and supported by my advisor Prof. Dr. Claudia Eckert, the visiting research experience has indeed opened my mind and broadened my vision.

Lastly but obviously not the least, I would thank my parents, who give me both life and best education, and my friends, who ever trusted and comforted me. Their love, patience and encouragement motivated me countless times, and this dissertation will also serve as a gift to thank them for their kindness.

Abstract

The advance of machine learning has enabled establishments of many automatic systems, leveraging its outstanding predictive power. From face recognition to recommendation systems and to social network relationship mining, machine learning found its rising attention from both researchers and practitioners in many different domains. Data-driven technologies based on machine learning facilitate the process of decision making extensively in our daily tasks. However, are the smart systems and their algorithm design really reliable enough for researchers and practitioners to trust? The fact is that very few researchers in this domain would ever consider the security of a learning algorithm, such that their systems can be easily subverted or at least compromised by some really intelligent and adaptive adversarial users.

In this work, we present the state-of-art study of a recent emerging research area named as Adversarial Machine Learning thoroughly, it investigates the vulnerabilities of current learning algorithms from the perspective of an adversary both in theory and practice. We show that several state-of-art learning systems are intrinsically vulnerable under carefully designed adversarial attack methods. We firstly propose a theoretical framework of the adversarial learning and the taxonomy of different potential threats. And we extend our study by introducing two main streams of adversarial attacks in several recent works, *i.e.*, causative attacks and exploratory attacks. Moreover, we suggest some countermeasures against adversarial actions, which inspire discussion and consideration of constructing more secure and robust learning algorithms. At last, we explore a research direction to reason the adversarial attacks using causal relationship, which is regarded as an effective approach for security analysis of any learning system. We conclude the entire work with a fruitful discussion and specify a list of open problems along with associated future works.

Zusammenfassung

Der Fortschritt des maschinellen Lernens hat wegen seiner hervorragenden Vorhersagekraft viele automatischen Systemen aktiviert. Von der Gesichtserkennung zu Empfehlungssystem und zu den sozialen Netzwerksbeziehungen Mining, maschinelles Lernen fand seine steigende Aufmerksamkeit von beiden Forscher und Praktiker in verschiedenen Bereichen. Datenbetriebene Technologien auf Basis des maschinellen Lernens erleichtern den Prozess der Entscheidungsfindung umfangreich in unseren alltäglichen Aufgaben. Jedoch sind die intelligente Systeme und deren Algorithmensentwürfe ausreichend zuverlässig für Forscher und Praktiker? Die Tatsache ist, dass nur wenige Forscher in diesem Bereich die Sicherheit eines Lernalgorithmus in Betracht nehmen werden, so dass ihre Systeme leicht verdorben oder zumindest von einigen intelligenten und adaptiven feindlichen Benutzern kompromittiert werden.

In dieser Arbeit präsentieren wir den aktuellen Stand der Forschungsgebiet von feindlichen maschinellen Lernen, welche ein neu erscheinendes Thema ist. Das Thema untersucht die Schwachstellen der aktuellen Lernalgorithmen aus der Sicht eines Gegners sowohl in Theorie als auch in Praxis. Wir zeigen, dass viele moderne Lernsysteme eigentlich von den sorgfältig konfigurierten Angriffsmethoden gefährdet werden. Wir schlagen zunächst einen theoretischen Rahmen von feindlichen maschinellen Lernen vor, und auch die Taxonomie der verschiedenen potenziellen Bedrohungen. Und dann erweitern wir unsere Studie durch die Einführung von zwei Hauptangriffsmodels in verschiedener Arbeiten, *d.h.* kausative Angriffen und Probeangriffen. Darüber hinaus empfehlen wir einige Gegenmaßnahmen gegen die feindliche Aktionen, die endgültig Diskussion und Konstruktion sicherer und stabiler Lernalgorithmen begeistern. Endlich untersuchen wir eine neue Forschungsrichtung, um die Angriffe mit kausale Beziehung zu erklären, die als ein wirksames Konzept für die Sicherheitsanalyse des Lernsystems betrachtet wird. Wir schließen diese Arbeit mit einer fruchtbaren Diskussion ab, und auch eine Liste von offenen Problemen zusammen mit zugehörigen künftigen Arbeiten wird gegeben.

Contents

List of Figures

List of Tables

Publications

Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *ICML'15*, Lille, France, July 2015.

Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Journal of Neurocomputing, Special Issue on Advances in Learning with Label Noise*, August 2014.

Huang Xiao and Claudia Eckert. Indicative support vector clustering with its application on anomaly detection. In *IEEE 12th International Conference on Machine Learning and Applications (ICMLA'13)*, Miami, Florida, USA, December 2013.

Huang Xiao, Han Xiao, and Claudia Eckert. Opars: Objective photo aesthetics ranking system. In *34th European Conference on Information Retrieval (ECIR'13)*, Moscow, Russia, March 2013.

Chapter 1

Introduction

Adversarial learning is a subfield of machine learning, where the learning problem is formulated on a dataset possibly tainted by a set of adversarial samples. By adversarial, it means that there exists a number of counterproductive actors who attempt to compromise the legitimate process of machine learning in oder to gain profit of certain kind. The uncertainty of prediction introduced by machine learning algorithms invokes a new dimension of security vulnerability for modern data-driven systems. Moreover, data is becoming the main driver to promote predictive services, *e.g.*, spam-filtering, mobile voice assistant, therefore the performance of a learning model is highly dependent on the quality of data sources. Not surprisingly, alternation on data sources is in many cases inevitable, *e.g.*, insider frauds, false manipulation, natural aging of devices. To this end, adversarial users may subvert the learning systems by manipulating the data sources. In this work, we walk through the adversarial learning in both theory and practice, eventually, our goal is to advocate the secure and robust design of machine learning methods, especially when they are applied in real work scenarios.

1.1 Motivation

Machine learning is nowadays becoming one of the most requisite components in many IT systems. In spite of its preponderant advantages, many learning-based systems may undergo some inherent deficiencies, which are sometimes utilized by malicious users to achieve their own good. Intrinsic vulnerabilities of learning algorithms are obviously overlooked by researchers in the machine learning community. Very few works in this area would ever consider the adversarial side of their proposed learning methods. Indeed, the demanding techniques of cracking a learning algorithm has prevented most of the malicious attempts until recent works on adversarial learning. They [77, 76, 12, 137] showed that there exists a number of intriguing properties of learning algorithms that will cause failure of their legitimate functions.

Recently two enlightening examples on deep neural networks [122, 94] highlight the

adversarial aspects of learning models. Deep networks are regarded as highly expressive

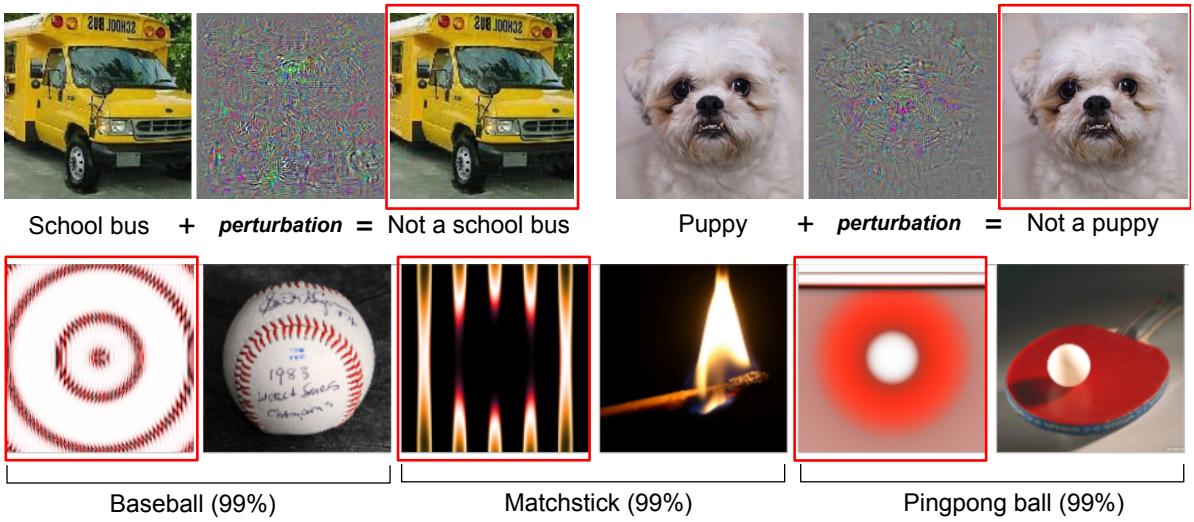


Figure 1.1: Adversarial examples evading deep neural networks. In the top row [122], inperceptible adversarial images generated with additive perturbation will be completely misclassified by the same DNNs. At the bottom row [94], unrecognizable images generated by evolutionary algorithm are classified with 99% confidence as recognizable images.

models that are widely applied in images recognition, voice recognition and other learning tasks. However, it seems that the expressive power may either overlook some blind spots in the sample space, or be overconfident of some clearly wrong examples. At the top row of Fig. 1.1, two original images (school bus and the puppy on the left side) can be correctly classified with high confidence. After a certain level of perturbation, two distorted images that inperceptible to human eyes are classified with 0% accuracy of being their correct classes. In contrary, at the bottom row, images that unrecognizable to human eyes are instead classified with 99% confidence as their original images (baseball, matchstick and pingpong ball on the right side). They obviously share some common features in the pixel level, which are considered as the most discriminative ones by the deep networks. In both examples, red framed images in Fig. 1.1 are the misclassified adversarial images that can be generated by a simple *maximizing-classification-error* process. The inherent properties of DNNs are decisive, since simply adoption of such models may lead to catastrophic results. For instance, a security guard system using DNN-based face detection might get evaded by those generated adversarial faces.

Another example on evading spam-filtering appears more frequently in practice. Since nowadays more and more spam filters are applying machine learning algorithms to detect malicious emails, *e.g.*, naive Bayes is a prevalent binary classifier for spam detection. Incentive of economic profit of promoting advertisement or similar information drives adversaries to obfuscate their spams as legitimate emails such that their content will

not be classified as spams. Recent works [77, 136] developed obfuscation techniques to successfully bypass statistical spam filters. In Fig. 1.2, we provide an illustration of how to obfuscate spams with good words to eventually bypass the detection.

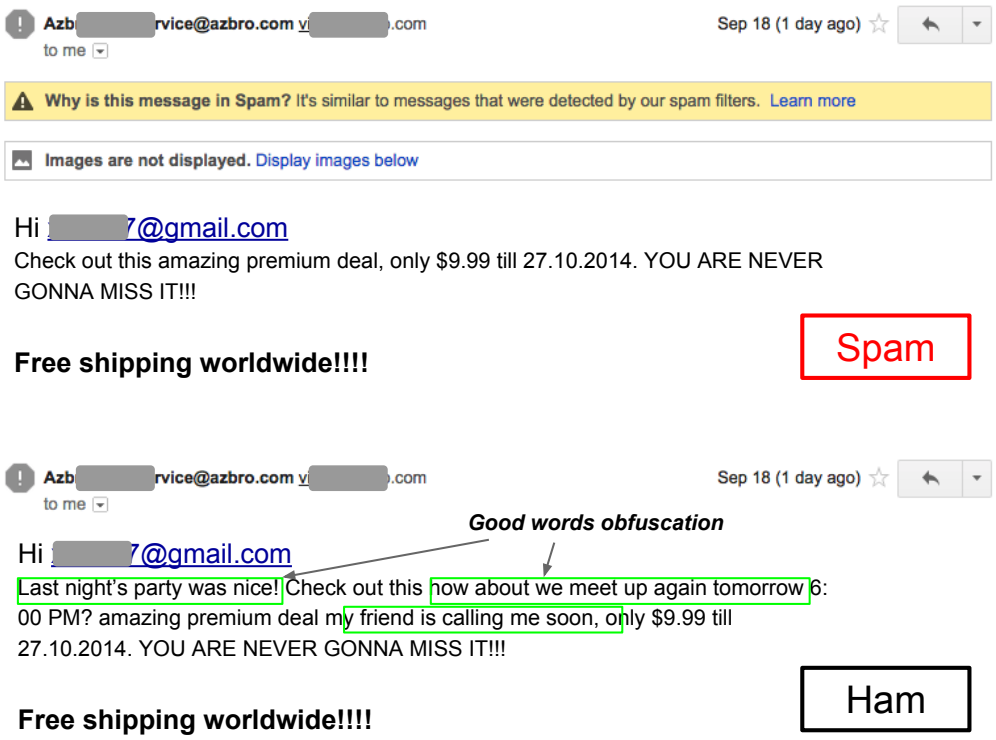


Figure 1.2: A spam email is obfuscated with additive good words such that it can bypass the spam filter as a ham. Green boxed text demonstrates the good words for obfuscation.

To this end, adversarial attempt on compromising a highly complex learning system seems promising to put the machine learning algorithms into a consideration that they might not be as formidable as originally designed. Therefore, we highlight the adversarial machine learning as a rising research area, which engages to provide a deeper understanding of learning algorithms from the perspective of adversaries. Eventually, more secure and robust learning systems are to be delivered in real-world scenarios.

1.2 Problem Statement

In this dissertation, we seek answers to following questions, that are posted considerably to present initiatives of studying the adversarial learning problems.

- **What is adversarial learning and how is it formulated?**
  Given the adversarial examples previously, the first problem we ought to clarify

is what is exactly the adversarial learning, and how we can formally formulate it. Conventional machine learning research has less concern on the security side of the algorithms, however in adversarial learning, we concentrate on problems of the possible exploits intrinsically embedded in the algorithms. Therefore a unified theoretical language of the adversarial learning is desirable to provide the foundation for further research of this area.

- **Are the adversarial attacks realistic and how would they be understood?** Although we may construct solid definition and formulation for the adversarial learning, there exists an enormous amount of interest to inspect the possibilities of different kind of attacks. The adversarial side of machine learning algorithms would drive researchers to investigate the methodologies of launching real attacks to find the blind spots. More importantly, insights of the adversarial attacks are expected to provide a comprehensive categorization of adversarial attacks.

- **Given the attacks in real scenarios, what are the possible countermeasures?**
  At last, secure and robust machine learning algorithms are supposed to be realized to counter the adversarial effect, given the fact that the learning systems are not always reliable under the extremely crafty and adaptive adversaries. Beyond defining and understanding the adversarial learning problems, countermeasures are to be delivered with respect to various vulnerabilities.

The aforementioned problems are subtle and essential for the study of adversarial learning, and we believe that the answers build the backbone of this research area. Although we would not be able to disclose every aspects of adversarial learning, we show in the sequel of this dissertation that our answers and contributions to the proposed problem statements are regarded as fruitful and indispensable for further research in this area.

1.3 Organization

For the remainder of this dissertation, we organize our work as follows to provide instructions of how to read this dissertation. In Chapter.2, we briefly introduce the background and preliminary knowledge of both machine learning and adversarial learning. More specifically we give a high level introduction of learning problems, which are shadowed by the adversarial impact with related works. Following it in Chapter.3, a comprehensive theoretical framework is proposed in formal language. We give formal notations and definitions to construct a general adversarial framework. And more importantly we provide a thorough taxonomy of adversarial attacks in different axises. We end this chapter by proposing several common attack examples with attack modelings respectively. Then our work is separated in two main parts, in Chapter.4 & 5 we focus on how

to design and implement real adversarial attacks on machine learning algorithms, and in Chapter.6 several robust and secure learning algorithms are proposed to give a guidance on how to enhance the security of modern learning systems. For the attacks, mainly causative attack and exploratory attack are studied with real examples. And for the secure learning, several different strategies of modeling robust learners are given. Besides, we stress on the importance of reasoning adversarial attacks for the security analysis. Finally in Chapter.7 we conclude this dissertation with discussions, and moreover we present clearly our contributions to this area. At last, a list of open problems during our work are presented to give future research directions of adversarial and secure machine learning.

Chapter 2

Background

Last decades, the revolution of computing has changed almost all the domains fundamentally. From personal computers to medical diagnosis devices, the advance of computation enables us to generate functions and solutions almost in a unlimited way. Among all these advances, Artificial Intelligence has finally evolved with a number of predictable progress in real world, in favor of the high performance computing systems and exponentially increasing data volume. Although traditional methodologies, like logical induction and expert systems, are still widely applied in many areas. The ideas of compiling rules of thumb and constructing expertise knowledge pool are relatively outdated, and replaced gradually with the principal of dealing with uncertainty. Respecting this, machine learning has gained most of the attention not only from researchers but also practitioners in industry, due to its profound capability of discovering uncertainties directly from observed data and its associated predictive power.

It is unarguable nowadays that machine learning is renowned of solving problems in a probabilistic way. For instance a speech recognition program identifies the semantic meaning of a human speech with a certain probability, instead of hard assertion. This probabilistic perspective of human intelligence is an enzyme to the birth of machine learning , which is designed as being capable to embody the intrinsic mechanism of predicting uncertainty. Especially with the boost of available training data, patterns can be constantly drawn from the empirical information. Tremendous effort is devoted in developing efficient and stable machine learning algorithms for tasks like classification, regression, and clustering. For instance, Support Vector Machine is extensively deployed in many systems because of its simplicity in structure and theory. Decision trees are always on top rank of best performing algorithms out of its boosting effect and straightforward interpretation. Deep networks are now striking almost all the learning scenarios together with its incredible predicting power. They all indeed benefit largely from rapid growth of computing resource and data explosion. Cloud computing brings even deeper changes in the traditional learning systems, which are often then distributed as a service. For example, Google prediction API provides interfaces to users to develop their own

spam filters or recommendation systems. IBM Watson offers both all-in-one solutions to industrial partners and cloud services to developers. machine learning based systems are now fast deployed almost everywhere in both industry and daily life.

However, the learning systems are facing with problems that they are not always reliable enough to provide stable and robust results. Similar to most of information systems, there exists potential security vulnerabilities and exploits within the learning algorithms themselves. Recent research work on Adversarial Machine Learning reveals certain security threats against the learning systems by investigating the weakness and robustness of machine learning algorithms. It shows that predictive models could deliver unreliable results when they are compromised by some highly skilled adversaries. Therefore, the data driven technologies built upon machine learning are still expected to involve rigorous research on their robustness, so as to be resilient enough against the adversarial impact.

2.1 Introduction of Machine Learning

In this section, we present readers a short introduction of machine learning . We are not intent to give a thorough analysis of the machine learning research itself, but to construct a backbone of the very basic concept of machine learning related to this work. There are tons of textbooks and literature of introducing much more details of machine learning, which is obviously far beyond the scope of our discussion. For advanced readers who are already familiar with machine learning, we recommend to proceed to the next sections.

The mechanism of discovering patterns and regularities from observations has been studied over centuries. Dated back to several decades ago, a well defined area named as Pattern Recognition was established to solve certain problems automatically facilitating computer algorithms. Up until then the complexity of real world scenario of pattern discovery required tremendous effort of researchers and practitioners to compile a large set of rules, in order to solve a relatively simple recognition task. Considering the learning task in Fig.2.1 to recognize the *MNIST* digits 3 [73], it would resort to partition the original image of size $28 * 28$ in 16 sub-matrices to feed custom rules and decisions as demonstrated. The images of digits have literally infinite variety of shapes due to the uncertainty brought by human hand writing. In this term, computer programs relying on custom logical rules could be extremely limited.

The influential work [97] leveraged the recent advance of probability theory and finally brought the probabilistic view to artificial intelligence. And machine learning has finally become an important research line built upon the foundation of probability theory, decision theory and information theory. The key concept of machine learning is to predict unseen instances from available observations, instead of coding rules based on domain knowledge. A general learning problem is thus given as follows.

Example 1. *Given a data set of n observations* $\mathcal{D} = \left\{(x_i, t_i) \mid x_i \in \mathcal{R}^d \text{ and } t_i \in \mathcal{R}^m\right\}_{i=1}^{n}$,

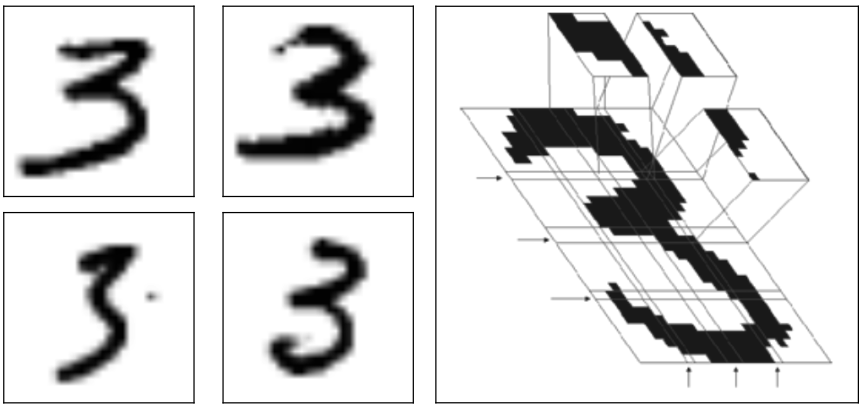


Figure 2.1: Examples of various *MNIST* digit images 3, which are partitioned in 16 sub-matrices to feed custom rules and decisions in order to be classified into correct classes.

where x_i is a n-dimensional input feature vector and t_i is a m-dimensional output. We find the most possible learning model f from the data set $\mathcal{D}$, such that the model f preserves the best predictive power on unseen data $\mathcal{X}^*$ i.i.d.*drawn from the same probability distribution.*

The target of any machine learning algorithm is to find the optimal learning function $f(x)$ in a hypothesis function space $\mathcal{F}$ which is usually constrained under certain conditions. The model relies on the observations *i.e.* training dataset to compass the searching in $\mathcal{F}$, where the f performs well on already seen data sample. However to generalize the predictive power of f, we also expect it to work effectively on unseen data *i.e.* the test dataset. According to existence of output vector t_i in each observation, machine learning algorithm can be categorized in three modes, namely, supervised, unsupervised, and semi-supervised.

- **Supervised learning** refers to those learning problems where supervised information is available directly in the training dataset, such as class labels, numerical outputs and so on. The supervised information is supposed to be reliable enough to guide the learning model to find its optimal form.

- **Unsupervised learning**, compared to supervised learning, does not require the availability of supervised information such as class labels or numerical outputs. It depends more on the intrinsic structure of the data space itself to accomplish tasks of clustering, feature extraction and so on.

- **Semi-supervised learning** is yet another important research area in machine learning . More practically in real applications, entirely labeled training datasets are rarely seen, however a minority of data samples can be annotated with least

effort. Therefore, the semi-supervised learning refers to the learning tasks where we can combine both the supervised and unsupervised information.

A recent developed paradigm in machine learning community named as self-taught learning [101] is yet another sub-domain aside aforementioned ones. Distinguished from semi-supervised learning, self-taught learning uses labeled data from desired classes while leveraging unlabeled data from other different but related classes. This technique is currently widely applied in images, sound and text, and is beyond the scope of discussion in this work. The taxonomy of machine learning based on whether the dataset contains supervised information plays a central role in designing a valid learning algorithm. Furthermore, the format of outputs determines what the learning algorithm can do on unseen samples, such as classification, regression and clustering.

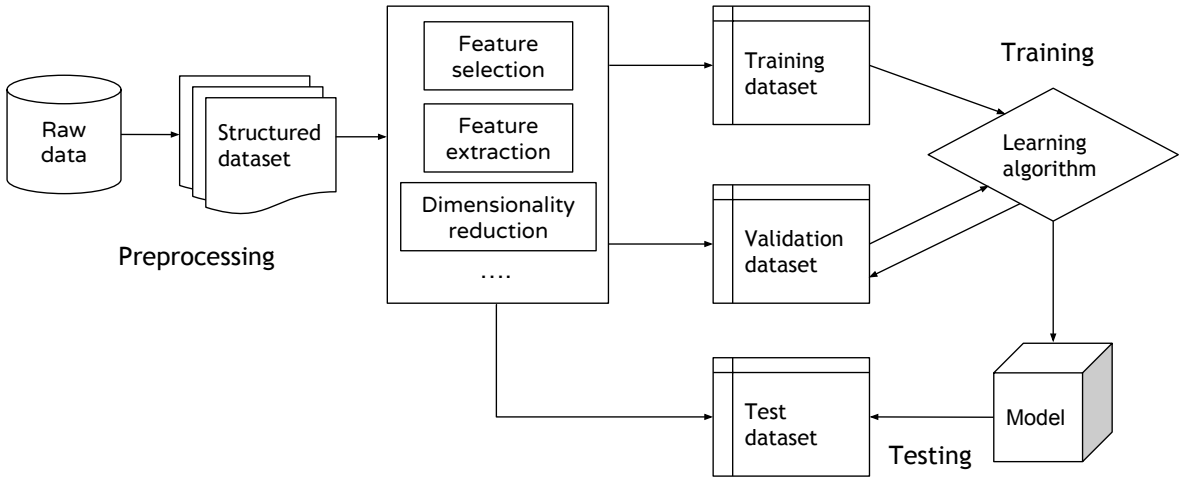


Figure 2.2: A commonly used learning process diagram from raw data input to a predictive model. The process contains data preprocessing, training model, and testing on unseen data input.

Apparently, learning algorithms rely heavily on the dataset to find an optimal learning model. From raw data to a predictive model, we can summarize the learning process in a number of stages. A commonly adopted learning process is depicted in Fig.2.2. In real world applications, data can be generated constantly from a certain mechanism, *e.g.* sensors on mobile devices, speech signals, video stream and so on. Diversity of data sources increases the complexity of data processing at the first place. In order to transform the raw data to a computer recognizable format, usually a large amount of preprocessing effort must be done in prior. However, even after the data is processed and structured, there is still a need of feature engineering including feature selection, feature extraction, and dimensionality reduction, to construct the most informative subset or projection of the original dataset. And the post-processed dataset is then subsampled in three part, namely, training set, validation set, and test set. Training data is firstly

fed to the learning algorithm to learn a predictive model, which is tuned on a separate validation set to adjust its learning parameters. Finally the predictive model is used to predict new unseen test dataset that is of central interest in practice.

As long as a problem can be defined as a learning problem, like recognizing handwritten digits we have seen in Fig.2.1, it can be broke down to a combination of representation, evaluation and optimization [40]. Representation is defined from the perspective of dataset. The data representation decides how is the data input recognized by the learning algorithm, and furthermore on the choice of proper learning algorithm. For instance, the data can be represented as a tree like structure if we treat the tree leaves as decisions. Or if the correlation among the features is acknowledged, then a graphical model is probably a good choice. Raw data is processed to meet the requirement of selected data representation, and then an evaluation criteria is determined to measure the goodness of the learning model f, which should be guaranteed to converge to the (sub-)optimal solution by choosing a suitable optimization technique. A schema of learning algorithms are summarized in these three components in Table.2.1

Representation	**Evaluation**	**Optimization**
Instance based	Accuracy or error rate	Greedy search
• Support vectors	Precision and recall	Gradient descent
• K-nearest neighbor	Squared error	Conjugate gradient
Tree based	F-1 score	Newton method
• Decision trees	K-L divergence	Stoc. gradient descent
• Random forest	Likelihood	Linear programing
Graphical models	Cross entropy	Quadratic programing
• Bayesian networks		

Table 2.1: A schema of learning algorithms. A learning problem can be described as an effective combination of each from Representation, Evaluation and Optimization.

Examples are only listed in Table.2.1 to briefly describe that a learning problem can be tackled by choosing a combination of these three components. Most of machine learning algorithms can be analyzed in this way to gain insight of its design. However, we will also show that the vulnerabilities are also well known under adversarial impact.

2.2 Learning under Adversarial Impact

Recent fast pacing development of machine learning has started to build integrated predictive models in many systems. A diversity of industries are facing a revolutionary change on products and services they can provide. Leveraging the burst of big data,

systems rely now heavily on the data and the learning algorithms applied on the dataset. However, the products and services provided by the learning based automatic process might not be regarded as reliable as users expect. Remarking the fact that machine learning is a highly sophisticated technique predicting the uncertainty in the essence of data observations, threats on the predictive models arise naturally with the emerging malicious attempts on valuable assets, especially with dramatically increasing adoption of machine learning systems. This coincides with a set of defined problems in the domain of information security. Indeed, adversarial machine learning can be viewed as a conjunction of machine learning and information security. The objective of this alliance is to investigate vulnerabilities of existing learning systems, and to design robust and secure machine learning algorithms under the existence of adversarial intention.

Anomalies vs. Adversaries

Consider a home environment monitor mounted with a set of sensors, such as infrared transducer, humidity sensor, the data collected by the device is commonly superposed with global noise which is produced by environmental dynamics. This can usually be referred as anomalies when the observations are not matching one of the previously found patterns. However, the sensors can also be manipulated under a certain purpose to produce unrecognized patterns to the monitor, which is eventually subverted by a series of malicious updates in data stream. This is named as adversarial noise, which is underestimated by many researchers and practitioners. An enlightening example showed in [122] crafted adversarial images generated by various neural networks have a 0% accuracy, while randomly generated noisy images produce nearly 50% as random guess. Being aware of the existence of anomalies, research areas such as anomaly detection, learning with noises are well defined. Unfortunately the distinction between anomalies and adversaries is less clearly stated until recent highlights on adversarial learners.

Adversarial Learner

Traditional security applications, for instances anti-virus program, network intrusion systems (IDS), rely on experts codifying rules and knowledge bases to automate the detection [43, 96]. These techniques will not only cost demanding, but also fail in cases with novelties. The semantics (behaviors) of internal states of instances, such as malware and intrusions, were clearly overlooked due to the high complexity and lack of automation tools. The evolving characteristics of security threats invoke thus semantic analysis of vulnerabilities, where machine learning is leveraged to construct the behavior-based detectors, especially to handle novel abnormalities. Pfoh *et al.*[100] employed support vector machine (SVM) with string kernel to sequentially predict malicious programs by investigating system call traces. Theoretical work in [109] built a foundation for recent advance of anomaly detection based on a margin-based classifier. In intrusion detection systems, much effort is also devoted by many researchers incorporating machine learning

[2, 75, 71, 85, 133, 146]. Numerous security tools and techniques based on machine learning are developed to integrate with traditional applications, and meanwhile adversaries also adapt their set of skills with the advance of learning systems. Assets protected behind the security guards are confidential and sensitive, which inspire the reconfiguration of the techniques of modern intruders. As a matter of fact, reverse engineering on learning systems is emerging as a new threat, possibly being neglected for the moment, in many domains.

D. Lowd and C. Meek [76] introduced an adversarial classifier reverse engineering (*ACRE*) method to learn information about a linear classifier in order to construct effective adversarial attack on spam filters. Another adversarial compromise on statistical spam filters is also proved as effective by filtering or introducing bag of words to bypass the spam detectors [78]. Another simple attack on statistical spam filters [136] is presented as well to leverage the nature of statistical learning. These efforts are nowadays not rare due to their own interests of gaining profits as adversaries. However, even the adversaries do not need to manipulate the learning models on a specific subset of sample, the models can also be inferred by a simple interaction of queries and answers [3]. Another interesting work [53] introduces a novel attack called "Bayes vs. Bayes" on statistical spam filters. It learns a second spam filter from the queries to the objective spam filter, and use the inferred second classifier to craft delicate spams to bypass the original spam filter. Although reverse engineering of security applications is mostly desired by malicious users, either to disable the functions (Denial-of-Service attack) or to gain sensitive information from the systems, recent research along this direction shows also compromises on other general-purpose learning models like classification and clustering. Support vector machines (SVMs) have been extensively studied in [138] [140] on its vulnerability against label noises, where various strategies of label flips attacks are performed on SVMs to compromise its classification effectiveness. Biggio *et al.* [21] has shown that single-linkage hierarchical clustering is especially sensitive to deliberate attack attempts, and more over they proposed a general framework to identify such attacks. Attacks on complete-linkage hierarchical clustering method is also extended with an ad-hoc poisoning schema, and shown as effective also on real-world data [10]. Not only some of the classification and clustering methods are threatened, but also the embedded feature selection methods are also sensitive against nasty noise. By maximizing the generalization error of the embedded feature selection model, the variance of selected subsets of features is clearly increased, so as the classification error [137]. It is shown that an indirect poisoning attack on generalization error might finally infect the feature selection stability in such adversarial environment. A recent exploit is even found in deep neural networks (*DNN*), which is regarded as the stat-of-art predictive model. The intriguing properties of *DNN* enable a certain type of compromise such that by overlaying hardly perceptible noise on images, they will be classified as a total different class [122]. On the other hand, researchers [94] also applied evolutionary algorithm to generate unrecognizable images to human, which are however recognized by *DNN* as belonging to certain class with 99.99% confidence. In a recent work of studying properties

of adversarial examples [51], it is found that the adversarial examples even generalize well across multiple learning models, if they are performed on similar tasks, *e.g.*, image recognition. This implies that the adversarial impact can potentially be transfered from one model to another without even knowing its statistical properties.

Malicious attempts on learning systems are now fast pacing towards novel cyber security threats, not only targeting traditional security applications like spam filters, malware detection, intrusion detection systems, but also on those advanced intelligent agents or automation services adopting machine learning. The attack vector studied in literature is closely related with intrinsic algorithmic design of the learning algorithms, which usually assume the data coming with a stationary distribution. In this term, the assumption of stationary data distribution makes the hypothesis that both training and test datasets are always drawn from the same distribution, which is obviously a strong constraint in real world. Considering the fact that data collected in real-time is usually disturbed at least by natural malfunction, e.g., aged sensors, background noise, and so on. The stationary assumption is barely true under the disturbance, especially when there exists carefully designed adversarial noise. Learning under adversarial impact is thus a highly demanding research area compounded with the fast developing data-driven technologies.

Robust Learner

Robustness of a system refers to the capacity of tolerating disturbance to deliver reliable and stable system performance. The property of robustness is highly desirable in machine learning under the adversarial impact. Without proper countermeasure against crafty adversaries, learning systems will be compromised to incur significant damage. Apart from security applications prohibiting systems from being abused, a robust learner is supposed to guarantee a reliable predictive ability with or without potential threats. Also being ware of the intrinsic uncertainty in learning models, robust learning is fundamentally different as those techniques applied in traditional information security.

To our knowledge, earliest noticeable work on robust learning was proposed by Kearns and Li [63], which centralized on the Probably Approximately Correct (PAC) learning model [126, 127]. They proved that a normal functional learner needs to bound the amount of manipulated training samples by $\epsilon/(1+\epsilon)$ to perform correctly, where ϵ is a permitted classification error by the learner. Similar works were also presented in [112, 27, 4] on PAC model. This associates the learnability and stability of a learning problem, which is a central challenge in learning theory. Also in the work of [18], manipulated label noises are mitigated by involving a simple kernel correction in the SVMs, assuming a certain level of noises. Similarly, [120] introduced a uniform prior on the label noises (flips) to enhance the robustness of learning SVMs from sloppily labeled training data. Some other techniques leading to the robustness will borrow loss function, *e.g.*, Huber's loss, from robust statistics, thus the function is less sensitive to outliers [55]. In probabilistic graphical model, [141] introduced a latent variable to represent ground truth, which generates a noisy version of observation. By maximizing

the log-likelihood, we can learn the distribution of the latent variable as well as the noisy level of multiple observers. Despite the robustness of Support Vector Clustering, [139] integrated additional user labels to reweigh the regularization coefficient C, so as to provide even more stable results on outlier detection. Indeed, they all can be considered as different types of regularization, which is a main technique of generalizing learning models. Moreover, adversarial examples can also be injected back to training samples [51, 84, 50], that will be trained to provide a robust model which is more insensitive to adversarial noises.

Although many efforts are devoted in dealing with noises or outliers as in aforementioned literature, we see that the robust learner is inevitably facing challenges from the adversarial noises. Assuming the discrepancy of the distribution of the training and test datasets, [48] employed a minimax optimization strategy over a defender-attacker game setting to counteract the adversarial effect introduced by worst-case feature deletion, so as for the defender to deliver a robust classifier. This game-theoretic perspective is naturally bridged with the adversarial machine learning. [35] has adapted the classifier dynamically according to adversary's attacking strategy. In their model, they configured a proper game setting by assuming that both parties (defender and attacker) have complete information of each other, thus the learner can always find an optimal solution dynamically adjusting to the adversary's action. Apparently the game between the learner and the adversary is of central importance in adversarial machine learning, we need a certain level of assumption on the game setting, for instance, one-shot game or repeated game. Involving game theory in the learning theory under adversarial impact is also a promising research direction.

Relation to Overfitting

In machine learning, overfitting refers to the problem of inconsistency of a learner's performance on training data and unseen data. It usually comes along with a dominating level of noises in observed data samples, such that the underlying pattern in the observations is less perceptible. We can always decompose the observations in two parts, namely the underlying pattern hidden in the samples and the stochastic (or adversarial) noises on observations. A robust and secure learner ought to seek an optimal solution to generalize its predictive ability not only on the training set, but also on unseen data samples in the future. This is called generalization in machine learning literatures. In order to generalize a model, it is wise not to apply a too complex estimator on training data with noises, so that the potential perturbation on observations will not dominate the model. Especially in adversarial environment, noises are no longer uniformly distributed, instead, they are delicately designed to achieve a certain malicious attempt. Intuitively we can image the complexity of data space is more sophisticated under adversarial impact, the learner will thus tend to increase its complexity to adjust to the data space. Therefore, it is easier to get overfitting.

2.3 Conclusion

In this chapter, we describe the background of adversarial and secure machine learning mainly in two parts. Firstly, machine learning as a well defined research area has found its rapid advancement in recent years, especially with the explosion of data generated and the marching development of computational power. However, data-driven technologies based on machine learning heavily rely on the data observations and the algorithmic design, which can easily fall into system deficiency under potential adversarial attacks. Therefore, learning under adversarial impact is now highly demanded while designing learning systems. Then we introduce the related work previously done in adversarial learning. A number of effort is devoted in revealing vulnerabilities of learning algorithms from the perspective of an adversary. And also in terms of defender (learner), the robustness of a learner is intensively associated the learnability of a certain learning problem, which is widely studied since a long time. However, a robust learner does not necessarily provide stable performance when there exists a set of adversarial noises carefully designed to contaminate the system. The interaction between the attacker and the defender invokes a yet explored sub-domain where we borrow the game-theoretic point of view to formulate the adversarial learning problem. To summary, this potential threat introduced by adversarial learners is thus a formidable problem for applications of data-driven technologies.

Chapter 3

Adversarial Learning Theory

Learning under adversarial impact is recently highly demanded due to the fast pacing advancement of machine learning based technologies. Previous work has shown that there is a potential threat from some highly skilled adversaries, who can explore the vulnerabilities of various learning systems. So far security threats are found mainly in the domain of information security, however, the adversaries are not limited to only violate security, but also the learnability of a certain algorithm. For instance, an adversary might attempt to inject a certain amount of carefully crafted images to an image recognition program, such that the program will not correctly classify certain images to their corresponding class. In this case, we say the learnability of current algorithm is compromised due to the increased complexity of data space.

In this chapter, we aim to provide a theoretical framework of adversarial machine learning, so as to build a solid foundation for security analysis of diverse machine learning algorithms. As discussed previously, learning algorithms are not always reliable due to a potential threat on the data gathered from various procedures, a thorough analysis from the perspective of a profound adversary gives important insight of the targeted system, which should be taken care of during the algorithm design. This might be the only possible and most effective methodology to indicate weakness of learners, and to deliver necessary patches respectively to enhance the robustness and security of the learning models. Similar to the domain of traditional information security, adversarial learning can be represented as a game between the defender (learner) and the attacker (adversary) in a game-theoretic setting, which is evolving over time. Typically, a learner is designed to discover patterns from historical data automatically, and to predict unseen data samples. However, driven by a certain incentive, e.g., financial gain, acquisition of confidential information, the adversary is able to discover an attack path to subvert the prediction either on a specific subset of data instances or on a general learning problem. We will categorize different types of attacks accordingly, and describe the attack scenario in details. Besides, we shape our theoretical framework to be easily extensible to those adversarial attacks that are not yet found, and also their countermeasures can

be developed efficiently.

3.1 Arms Race Problem

In security, there are analogues to the arms race problem for describing a game-theoretic setting between multiple parties, who usually compete over a certain assets to achieve their own goals respectively, *e.g.*, arms race between malware writers and anti-virus writers or spammers and anti-spam detectors. Similarly in adversarial machine learning, it also involves at least two parties, namely the learner and the adversary. Driven by opposite objectives of the learners and adversaries, a "reactive" arms race problem is commonly launched by the party of the very intelligent and adaptive adversaries. By "reactive" it refers to a unconscious situation to the learners, who usually naively sticks to the stationary assumption until the adversaries explore some vulnerabilities in the design of algorithm. Damages caused by the adversaries are supposed to be discovered rapidly by the learners, and are to be countered usually by either retraining on new data excluding malicious ones, or adding or removing a certain number of features that are intrinsically sensitive to those attacks. Even the most efficient countermeasures would have a significant delay of response time, which gives adversaries enough time to obtain what they aim for.

The "reactive" arms race problem exists commonly in security engineering. A common approach widely used in its domain, especially in cryptography is *security by obscurity*. It protects confidential information by keeping it under an encrypted shell, and the malicious users are not supposed to get through to the protection. Although this mechanism is still valuable in the domain of adversarial learning, we advocate and investigate another paradigm of securing a learning system, that is, instead of reactively responding to attacks, the system should be secured proactively in the design phase, so called "proactive" arms race problem. Following the paradigm of *security by design*, the learners are now aware of the possible breakthrough of the stationary assumption, and can anticipate potential attacks by modeling the adversaries. On one hand, such proactive security strategy offers the system higher level of security by taking potential vulnerabilities into account before designing the learning algorithm, on the other hand, the adversaries would spend much more effort to explore the systems before they are detected. The robustness is then guaranteed for a longer time and even without much effort of human intervention. In Fig.3.1 we depicted the concepts of both reactive and proactive arms race problems, which are of central role for the adversarial learning. On the left hand side, the "reactive" arms race problem is formulated in four stages. Firstly, the adversary examines an existing learning system's potential vulnerabilities driven by a certain incentive, e.g., economic gain, privacy. By analyzing the system, the adversary devise an effective attack and perform on the system. Then the attack may cause suspicious system behaviors which are then discovered by the learner. Finally, the learner responses to the attack by retraining on new data or refining features accord-

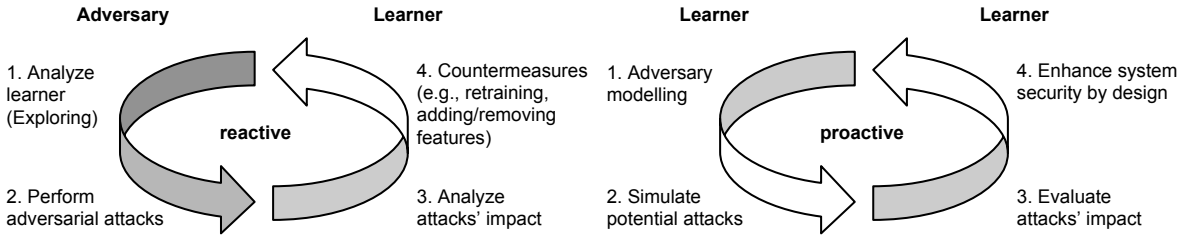


Figure 3.1: The concept of both "reactive" and "proactive" arms race problems. The arms race problem is iterative over time between the parties of adversary and learner. Both modes of arms race problem can be formed in four stages describing a full cycle from attacking to defending.

ingly. In proactive mode on the right hand side, the learner assumes the existence of adversaries and models the adversaries before the real attacks happen. By simulating the potential attacks, the learners can evaluate its negative impact on current system, whose security is then enhanced again at the design phase. Although in proactive arms race the adversary is not explicitly shown, we expect the learners' capability of modeling the potential adversaries highly depends on the properties of the adversaries, who are extremely adaptive during the iterations of the arms races.

Indeed the arms race problem in adversarial learning environment is still iterative and evolving over time. The concept diagram in Fig.3.1 illustrates a full cycle of adversarial learning and implies that the game between adversary and learner can be either one-shot game or iterated game. Although the importance of equilibrium in this game-theoretic setting can not be neglected, we focus on one-shot game in this work as atomic game to achieve balance. Recent work [25] has explored properties of Nash-equilibrial prediction models, and shown that under certain conditions we do find a unique Nash equilibrium between the adversary and the learner. Nevertheless iterated games and equilibrial of various adversarial games remain as an important research direction for the future. Finally built on the arms race problem defined, we summarize the problems to be investigated in this work as follows,

- **Analysis of targeted learner process:** a targeted learner process, e.g., classifier, must be firstly thoroughly studied and carefully analyzed, so as to model a potential adversary based on the knowledge of the learner process. During the analysis, theoretical properties of the learning algorithm will be a central issue.

- **Adversary modeling:** potential attacks are modeled targeting a particular learner process that evaluated thoroughly. And the impact of attacks is also to be examined.

- **Design of countermeasures:** countermeasures against adversarial attacks are integrated directly at the design phase. The learning algorithms assume a possibly

non-stationary data distribution and can adapt to concept shift, they are designed as being more robust and more resilient to uncertainty of data sampling.

3.2 A General Framework of Adversarial Learning

To study the adversarial learning, we need to define a formal language describing the learning problems under adversarial environment. A general framework is desired to provide a high level unified description of the adversarial learning problem so that particular problems can be well analyzed using a common language. Since adversarial learning is an extension of current machine learning research, we will follow and advocate the adversarial learning theory aligning the current available modeling methodology. We start with a list of notation which will be used all through this work to provide a convenient reference for different symbols.

List of Notation	
$\mathfrak{D}$	Space of data distribution
$\mathcal{F}$	Hypothesis space of learners
$\mathbb{P}_{tr}$, $\mathbb{P}_t$	Data distribution of training dataset, test dataset
$\mathcal{D}_{tr}$, $\mathcal{D}_{\text{vd}}$	Training dataset, test dataset
$S(\mathcal{D})$	Feature set of dataset $\mathcal{D}$
x, t	A data sample and its response
$\mathcal{Y}$	The label set
f	The learner
f^*	The best learner
$f_{\mathcal{A}}$	The learner compromised by attack $\mathcal{A}$
H	Evaluation function of learner f
ℓ	Loss function
Ω	Regularization term
λ	Smoothness factor in regularization
K	An adversarial attacker
$\mathbb{K}_f$	Space of possible attackers on learner f
$\mathcal{W}$	Objective function of an attacker
$\mathcal{A}$	Attack strategy (a subset of attack samples)
$\mathcal{A}^*$	Best attack strategy
$\boldsymbol{\Theta}$	Space of attacker's knowledge of learner
$\boldsymbol{\theta}$	Attacker's knowledge of learner
$\mathcal{C}(\mathcal{A})$	Cost function of attack strategy $\mathcal{A}$
$\Phi(\mathcal{A})$	Space of possible attacks
n, m	Number of training dataset, test dataset

Table 3.1: List of notations for the adversarial learning framework.

3.2.1 The Learning process

A learning process is defined as a system component which finds patterns out of seen data samples automatically, and form a predictive model for prediction. For convenience, sometimes we also refer it as learner, classifier or defender. Note that we use the terms learner, classifier, and defender interchangeably to represent the learning process through out this paper. In adversarial learning, the learner is the targeted component. By altering the data distribution for training or testing, adversaries explore the potential vulnerable spots of the learner, such that the learner performs a compromised predictive capability on a general learning problem or a specific set of samples.

Definition 1. *Given a training data set $\mathcal{D}_{tr} \in \mathbb{R}^d$ consisting of n samples drawn from a d-dimensional training data distribution $\mathbb{P}_{tr} \in \mathfrak{D}$, where $\mathfrak{D}$ is the space of d-dimensional data distribution. By selecting a proper evaluation function H, a learning process is defined as selecting the best learner f^* from the hypothesis learner space $\mathcal{F}$, such that the evaluation function is minimized.*

$$f^* = \arg\min_{f \in \mathcal{F}} H\left(f\left(\mathcal{D}_{tr}\right)\right)$$

Typically the evaluation function H is selected to be a cost function which evaluates the misclassification rate in classification or mean squared error in regression problem. We do have seen other equivalent types of cost function H, such as the margin $\|w\|$ in SVM or negative log-likelihood in *MLE*. The cost function can be chosen according to the particular learning problem, however it is reasonable to assume that the selected cost function H should also maximize the predictive capacity of f^* on unseen data, which is commonly referred as its generalization ability in machine learning literature. Generalization can be achieved by adding a regularization term in the cost function H, which is used to control the complexity of the learner to prohibit it from overfitting. This can then be formulated as a common procedure (regularized) empirical risk minimization (ERM) in the statistical learning theory. It is currently a known framework in machine learning unifying different types of learning procedures. Many learning scheme can be directly or indirectly converted as being belong to the ERM model [129].

Definition 2. *Empirical Risk Minimization Given a training data set $(x, t) \in \mathcal{D}_{tr}$, we define a loss function ℓ to access the empirical risk between the real response t and its evaluation by learner $f(x)$, the best learner is found by minimizing the empirical risk.*

$$f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{(x,t) \in \mathcal{D}_{tr}} \ell\left(t, f(x)\right) + \lambda \Omega(f) \tag{3.1}$$

The regularization term $\Omega(f)$ is controlled by the weight λ indicating the smoothness of the expected learner. Note that the cost function H is now replaced with two parts, *i.e.*, the loss function ℓ accessing the empirical risk between observed response t of an

input x and its evaluation by f, and the regularization term Ω controlling the complexity of f. In some literature [137], we also call Eq. (3.1) as generalization error. Assuming the stationary assumption, where both training data and test data are drawn from the same distribution, the best learner can be found by solving a well defined optimization problem in the *ERM* model. However in real world, the stationary assumption may be broken by adversaries, and *ERM* may deliver a compromised learner without considering the shift of data distribution.

Prediction Note that the response t in Eq. (3.1) can be empty in unsupervised learning, however in prediction, the response $\hat{t}$ exists as long as an unseen sample $\hat{x}$ is given to the best learner f^*. For example in classification or clustering, the response t represents the label of corresponding class (cluster), while in regression it is a numerical output.

$$\hat{t} = f^*(\hat{x}), \;\; \hat{x} \in \mathbb{P}_t,$$

where $\mathbb{P}_t \in \mathfrak{D}$ is the test data distribution, and it does not necessarily belong to the same distribution family as $\mathbb{P}_{tr}$. Given a test dataset $(x_i, t_i)_{i=1}^m \in \mathcal{D}_{\text{vd}}$, the performance of a learner f^* is evaluated by a loss function ℓ_t defined as,

$$\mathcal{E}_t = \frac{1}{m} \sum_{i=1}^{m} \ell_t(t_i, f^*(x_i)) \tag{3.2}$$

Normally the loss function ℓ_t is chosen differently as ℓ, and more closely related to the predictive task, such as classification error rate, mean absolute squared error. In adversarial learning, this loss function can also be freely chosen by adversaries to achieve their own goal. For instance, an adversary is only interested at maximizing classification error on a certain data subset.

3.2.2 Adversary Modeling

An adversary is supposed to be intelligent and adaptive to the learning system, and can introduce modifications on either training data or test data. Such modifications will typically alter the original data distribution which makes the stationary assumption fail. By contaminating partially the training data or test data, the adversary K is able to maximize his gain by misleading the learner to a certain level. Therefore, in adversarial learning we model the adversary K in this way, such that the adversary achieves his objective the most.

Definition 3. *An adversary is defined as a vector encoded in a 3-tuple $K = (\mathcal{A}^*, f, \boldsymbol{\theta}) \in \mathbb{K}_f$, where $\mathcal{A}^*$ is an optimal subset of data samples causing a maximal downgrade of the learner f, and $\boldsymbol{\theta}$ encodes the attacker's knowledge about learner f. $\mathbb{K}_f$ is the space of possible attackers against learner f.*

Given the definition of the adversary K, the optimal attack against the learner f is found by discovering an optimal data subset $\mathcal{A}^*$ bounded by the adversary's knowledge

$\boldsymbol{\theta}$ of the learner. We characterize the attacker's capability by assuming that an initial set of samples $\mathcal{A}$ is given, and that it is modified according to a space of possible modifications $\Phi(\mathcal{A})$. Given the attacker's knowledge $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and a set of manipulated attacks $\mathcal{A}^* \in \Phi(\mathcal{A})$, the attacker's goal can be characterized in terms of an objective function $\mathcal{W}(\mathcal{A}^*, \boldsymbol{\theta}) \in \mathbb{R}$ which evaluates how effective the attacks $\mathcal{A}^*$ are. The optimal attack strategy can be thus given as:

$$\begin{array}{rl} \max_{\mathcal{A}^*} & \mathcal{W}(\mathcal{A}^*; \boldsymbol{\theta}) \\ \text{s.t.} & \mathcal{A}^* \in \Phi(\mathcal{A}) \\ & \mathcal{C}(\mathcal{A}^*) \leq \tau \end{array} \tag{3.3}$$

The adversary leverages partially the knowledge $\boldsymbol{\theta}$ about the targeted learner and inject a set of modification on dataset $\mathcal{A}$ upper bounded by a cost function $\mathcal{C}$. Indeed the selection process of optimal subset $\mathcal{A}^*$ from the space $\Phi(\mathcal{A})$ is difficult and quite often strong $\mathcal{NP}$-hard [140]. Many work has been done in efficiently selecting the (sub)optimal subset either from training data distribution $\mathbb{P}_{tr}$ or from the test data distribution $\mathbb{P}_t$. And we name the optimal subset $\mathcal{A}^*$ as malicious or adversarial noise, which deviates far from regular anomalies or noise by unintentional system behavior, and is delicately crafted by very particular adversaries.

3.2.3 Evaluation of Attacks

Adversarial attacks on learners are evaluated by the objective function $\mathcal{W}$ in Eq. (3.3), where the attackers will achieve their maximum of defined target. The objective function $\mathcal{W}$ can be freely chosen according to attacker's real interest, however in adversarial environment, it resorts to a common goal of producing responses of input data as closely as they expect. That is, the objective encoded in $\mathcal{W}$ is to manipulate the classifier f to generate expected predictions of input dataset. Since the evaluation of attacks depends highly on the objective of the attacker, we illustrate the process of evaluation in two simple examples.

Example 2. *A spammer intends to create a few advertising spam emails which will not be detected by the spam filter. We define the spam filter as a classifier f discriminating emails as legitimate ($y = -1$) or spam ($y = +1$), where y is the label of an email. The spammer composes a set of emails $\mathcal{A} = \{x\}_{i=1}^m$ and modifies the $\mathcal{A}$ according to his knowledge $\boldsymbol{\theta}$ about the spam filter f. Note that the knowledge of the classifier might be acquired by a limited number of queries.*

By modifying the set $\mathcal{A}$, the spammer expects the crafted emails to evade being detected as spams ($y = +1$), therefore the spammer defines an attack objective to evaluate the effect of the attack, such that

$$\begin{array}{rl} \mathcal{A}^* = & \arg\max_{\mathcal{A} \in \Phi(\mathcal{A})} \sum_{i=1}^m -f(x_i), \;\; x_i \in \mathcal{A} \\ s.t. & \mathcal{C}(\mathcal{A}) <= \tau \end{array} \tag{3.4}$$

Note that the spam filter $f(x)$ gives label $+1$ or -1. Maximizing the sum of negative responses of x_i equals classifying most of the emails as legitimate. Constrained by the knowledge of spam filter $\boldsymbol{\theta}$, the spammer is only supposed to add or remove a limited number of legitimate words in the emails, where these cost-sensitive modifications are obviously bounded by a cost function $\mathcal{C}$.

Example 3. *An image recommendation system f returns a ranking score given an image's feature vector x, which can be viewed as a regression function. Now an adversary attempts to cause the system to provide a degenerated service by introducing a certain amount of images. Suppose the adversary knows the type of the algorithm the system is currently running, and also gain access to part of the images on the server (in practice may be hard), at least, the adversary is able to produce some wrong ratings to the images.*

We define the adversary's knowledge of the classifier f as $\boldsymbol{\theta} = (f, \hat{\mathcal{X}}, S(\hat{\mathcal{X}}))$, where $\hat{\mathcal{X}} = \{(x_i, y_i)\}_{i=1}^{m}$ is part of the training images and $S(\hat{\mathcal{X}})$ encodes the feature set of $\hat{\mathcal{X}}$. The most effective attack is to maximize the generalization error on currently available data set $\hat{\mathcal{X}}$.

$$\begin{aligned} \mathcal{A}^* = \quad & \arg\max_{\mathcal{A}\in\Phi(\mathcal{A})} \tfrac{1}{m}\textstyle\sum_{i=1}^{m}(\|y_i - f_{\mathcal{A}}(x_i)\|)^2 + \lambda\|f_{\mathcal{A}}\|^2 \\ s.t. \quad & f_{\mathcal{A}} = \arg\min_{f\in\mathcal{F}} \tfrac{1}{n}\textstyle\sum_{(x,y)\in\mathcal{X}\cup\mathcal{A}} \ell\left(y, f(x)\right) + \lambda'\Omega(f) \\ & \mathcal{C}(\mathcal{A}^*) <= \tau \end{aligned} \tag{3.5}$$

where $f_{\mathcal{A}}$ is the optimal classifier learned on the contaminated training dataset $\{\mathcal{X} \cup \mathcal{A}\}$, and the optimal attack $\mathcal{A}^*$ is found by maximizing the generalization error on the partially available dataset $\hat{\mathcal{X}}$ for the attacker, assuming that the classifier is always optimal disregarding the existence of adversarial noise.

In Eq. (3.5), the optimal attack on the recommendation system solves a bi-level min-max optimization problem. Note that the optimum is not always guaranteed and often it is $\mathcal{NP}$-hard, however it represents the intrinsic arms race property of the adversarial learning problem. In one-shot games, we are interested at keeping one side constant and looking for the optimum at the other side.

3.3 Taxonomy of Adversarial Attacks

Built upon the general framework of adversarial learning, we describe the taxonomy of adversarial attacks in different dimensions. The goal of taxonomy for the adversarial learning is to classify various types of attacks based on their property and impact on current learning system, so that we can gain better understanding of the attackers' strategy and develop effective countermeasure against them.

For simplicity, we resort to a binary classification problem to illustrate the taxonomy of adversarial attacks. Given a training dataset $\mathcal{X}_{tr}$ and its associate labels $\mathcal{Y}_{tr} = \{+1, -1\}$, a binary classifier is a discriminative function f for input x to give either positive $(+1)$

label or negative (-1) label. We denote $S(\mathcal{X})$ as the feature set for the dataset $\mathcal{X}$. In practice, we appoint positive sample as the instance of common interest, e.g., in intrusion detection system, positive sample refers to intrusion (anomaly).

3.3.1 Security Violation

In adversarial learning, the attacker's goal usually associates with the security violation of the targeted systems. For instances, in email spam filter, attackers aim to create a batch of spam emails evading being detected as spams, which compromise the integrity of the email system. When the learners are compromised by those malicious attackers, the security will be violated under the adversarial attacks. Therefore we firstly define the attacker's goal in terms of the desired *security violation*, which can be an **integrity**, **availability**, or **privacy** violation. According to the affected range within the targeted system, we can further distinguish the *attack specificity* as being **targeted** or **indiscriminate** [7, 59, 16, 22].

Integrity is violated if malicious activities are performed without compromising normal system operation, *e.g.*, attacks that evade classifier detection without affecting the classification of legitimate samples. In the case of binary classification, the attackers create a number of positive samples, but recognized by f as being negative (legitimate samples), thus the integrity of the binary classifier is violated to allow malicious activities to bypass the detection. Note that the function on legitimate samples will not be influenced.

Availability is violated if the functionality of the system is compromised, causing a denial of service. More strictly, the reliability of the predictive service is now damaged, especially in machine learning, the prediction of unseen data naturally comes with uncertainty. The reliability of the service provided by the learning system is thus of significant importance. For classification and clustering, this respectively amounts to causing the largest possible classification error and to maximally altering the clustering process on the input data [59, 16, 22]. Under the impact of adversaries, the predictive services are generally degenerated while the availability is violated, instead of specifically on positive samples.

Privacy is violated if the attacker is able to obtain information about the system's users by reverse-engineering the attacked system. More generally, the information of the classifier itself can also be sensitive in order to bound the capacity of an adversary. To reverse engineer a learning system, it is wise to limit the information exposed to the attacker as much as possible. Feature set $S(\mathcal{X})$, user input data $\mathcal{X}$ and even the system responses to legitimate user queries should all be regarded as privacy.

Finally, the attack specificity is **targeted**, if the attack affects the selection of a *specific* subset of information, and **indiscriminate**, if the attack affects the selection of *any* information.

3.3.2 Adversarial Capability

Another axis of classifying adversarial attacks grounds on the attacker's capability. It mainly relies on the information acquired by the attacker, who can have different levels of knowledge of the system, according to specific assumptions on the points $(k.i)$-$(k.iii)$ described in the following.

$(k.i)$ **Knowledge of the training data** $\mathcal{D}_{tr}$: The attacker may have partial or full access to the training data $\mathcal{D}_{tr}$. If no access is possible, he may collect a *surrogate* dataset $\hat{\mathcal{D}_{tr}} = \{\hat{\boldsymbol{x}}_i, \hat{y}_i\}_{i=1}^{m}$, ideally drawn from the same underlying process p from which $\mathcal{D}_{tr}$ was drawn, *i.e.*, from the same source from which samples in $\mathcal{D}_{tr}$ were collected; *e.g.*, honeypots for malware samples [119].

$(k.ii)$ **Knowledge of the feature set** $\mathcal{S}(\mathcal{X})$: The attacker may partially or fully know how features are computed for each sample, before performing the attack. The acquisition of the feature information is now more and more realistic while the machine learning is getting more and more standardized to provide unified services, *i.e.*, to process text dataset we usually employ the numerical statistic *tf-idf* to convert a document to a machine-friendly feature vector. Thus we believe that the attacker can acquire the knowledge of the feature set $\mathcal{S}$ with bearable effort.

$(k.iii)$ **Knowledge of the learning algorithm** $h(\mathcal{D}_{tr})$: The attacker may know that a specific learning algorithm is used, along with a specific selection criterion $\mathcal{L}(\mathcal{D}_{tr})$; *e.g.*, the accuracy of a given classifier. Aforementioned trend on feature engineering also applies on the choice of learning algorithms. Especially some algorithms are more frequently used in certain application, *i.e.*, deep network for image recognition, Naive Bayes for document classification.

Depending on the knowledge level of the attacker, we can further classify the adversarial attacks along the axis of attacker's capacity. Namely we have the *Perfect Knowldge (PK)* attack and the *Limited Knowledge (LK)* attack.

Perfect Knowledge (PK). The worst-case scenario in which the attacker has full knowledge of the attacked system, is usually referred to as *perfect knowledge* case [16, 22, 20, 65, 24, 59, 7], and it allows one to empirically evaluate an upper bound on the performance degradation that can be incurred by the system under attack.

Limited Knowledge (LK). Attacks with *limited knowledge* have also been often considered, to simulate more realistic settings [16, 22, 12]. In this case, the attacker is assumed to have only partial knowledge of $(k.i)$ the data, *i.e.*, she can collect a surrogate dataset $\hat{\mathcal{D}}$, but knows $(k.ii)$ the feature set, and $(k.iii)$ the learning algorithm. He can thus replicate the behavior of the attacked system using the surrogate data $\hat{\mathcal{D}}$, to construct a set of attack samples. The effectiveness of these attacks is then assessed against the targeted system.

Although there exists arguments of how realistic are both *PK* and *LK* attacks, in common practice of computer security, we dare the erring of overestimating the attacker's capabilities rather than underestimating them.

3.3.3 Adversarial Influence

The adversarial influence defines how and to what extent the attacker can control the learning process. As for supervised learning, the attacker can influence both training and test phases, or only at test phase, respectively exercising a **causative** or **exploratory** influence (more commonly referred to as poisoning and evasion attacks) [7, 59, 16]. Although modifying data at test time does not affect the learning process directly, it may nevertheless influence the security of the corresponding classifier against evasion attacks at test time, as also highlighted in recent work [74, 131]. Therefore, evasion should be considered as a plausible attack scenario.

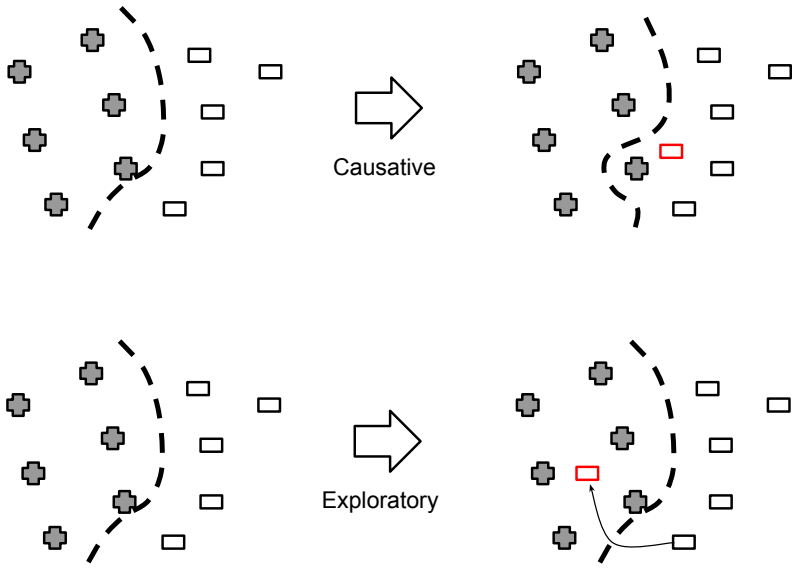


Figure 3.2: Causative and exploratory attacks on binary classification. On the top row, causative attack injects a negative sample into training dataset, which changes the classification boundary. On the bottom row, exploratory attack aims to find an equivocal spot in the data space to let a negative sample evade being detected, but the classification boundary stays untouched.

Causative (poisoning) attacks. the attacker is often assumed to control a small percentage of the training data $\mathcal{D}_{tr}$ by injecting a fraction of well-crafted attack samples. The ability to manipulate their feature values and labels depends on how labels are assigned to the training data; *e.g.*, if malware is collected via *honeypots* [119], and labeled with some anti-virus software, the attacker has to construct poisoning samples under the constraint that they will be labeled as expected by the given anti-virus. In this way, obviously, the training dataset is 'poisoned' and mislead the classifier to construct less effective predictive function.

Exploratory (evasion) attacks, the attacker manipulates malicious data at test time to evade detection. Clearly, malicious samples have to be manipulated without affecting their malicious functionality, *e.g.*, malware code has to be obfuscated without compromising the exploitation routine. In several cases, these constraints can be encoded in terms of distance measures between the original, non-manipulated attack samples and the manipulated ones [36, 76, 49, 123, 24, 12]. The attackers explore the boundary of targeted classifier and tries to mimic equivocal instances to get evaded.

We illustrate the causative and exploratory attacks on the case of binary classification in Fig. 3.2, which presents the fundamental difference of two types of adversarial attacks.

3.3.4 Summary

	Integrity	**Availability**
Exploratory		
- *targeted*	Misclassifying a specific subset of positive samples.	Generally harm the learner's prediction on any of a specific subset samples.
- *indiscriminate*	Misclassifying positive samples generally.	The learner's prediction on any samples is compromised.
Causative		
- *targeted*	Classifier is mis-trained on particular positive samples.	Classifier is mis-trained on particular subset of samples.
- *indiscriminate*	Classifier is mis-trained generally on positive samples	Classifier is mis-trained generally on all samples.

Table 3.2: Taxonomy of adversarial attacks in different axises.

We classify types of adversarial attacks mainly along three axises, *i.e.*, attacks on security violation, attacks according to adversaries' capacity and their impact on targeted systems. A clear taxonomy of potential adversarial attacks suggests constructive strategies of countering against those threats on targeted learning systems. Security-sensitive applications, *e.g.*, IDS, Anti-virus softwares, enhance their systems with focus on security violation by adversaries. Compromise on both integrity and availability would not only harm the applications, but also the assets under their protection would suffer from enormous lost. In order to protect against those potential threats, it is wise to follow the common practice in computer security, that is, rather than underestimating attackers we should overestimate their capacity and knowledge of the underlying systems, such that security engineers would put more effort to solidify the system design. At least the consolidation will force the adversaries to put more effort (e.g., time, skill, money) to crack their targets, that will help the defenders finally take the lead of the arms races.

Finally we summarize the taxonomy of adversarial attacks in Table 3.2. Note that all the attacks listed can either be a **Perfect Knowledge** attack or a **Limited Knowledge** attack depending on the knowledge level of the attacker.

3.4 Adversarial Attack Scenarios

We complete the adversarial learning theory by introducing some common attack scenarios, each of which aligns the taxonomy of the adversarial attacks. In practices, attackers will firstly define their goals, which are normally associated with the security violation. Besides, attackers specify whether they attempt to compromise the system generally (**indiscriminate**) or just fail the system on a particular set of data (**targeted**). In many cases, adversaries design their attack path either as causative or exploratory. In causative attack, mostly the classifier will be mis-trained on a contaminated (*poisoned*) training set, which will harm the availability of the system eternally. In exploratory attack, the attacker creates a particular set of carefully designed samples to bypass the classifier, where the integrity is often violated.

Indiscriminate Causative Integrity Attack

The indiscriminate causative integrity attack (*ICIA*) causes the classifier compromised on positive samples. Causative attacks usually come with a degeneration on the classifier itself, where the classifier is mis-trained on some poisoned dataset. Again, we employ the spam filter f as an example, which is a binary classifier.

Suppose we have an attacker K who gains a limited knowledge set of the spam filter $\boldsymbol{\theta} = (\mathcal{D}_p, S(\mathcal{D}_{tr}), h(\mathcal{D}_{tr}))$, where $\mathcal{D}_p \in \mathcal{D}_{tr}$ and $\mathcal{D}_{tr}$ is the emails used to train the spam filter. The attacker starts to inject a subset of malicious emails $\mathcal{A}$ as initial attack. Assuming the spam filter is not aware of the attack, the attacker will simulate the retraining on $\{\mathcal{D}_p \cup \mathcal{A}\}$ to get a compromised version f_A. The attacker evaluates the effect of attack $\mathcal{A}$ on a separate evaluation dataset $\mathcal{D}_{\text{vd}}$. To select the most effective attack $\mathcal{A}^*$, the false negative is to be maximized.

$$\mathcal{A}^* = \underset{\mathcal{A} \in \Phi(\mathcal{A})}{\arg\max}\, FN(y_i, f_{\mathcal{A}}(x_i)), \quad (x_i, y_i) \in \mathcal{D}_{\text{vd}} \tag{3.6}$$

Solving the optimization problem the attacker can find an optimal attack set $\mathcal{A}^*$ under the limited knowledge $\boldsymbol{\theta}$, which is required to retrain the poisoned classifier $f_{\mathcal{A}}$. The convergence and complexity of the attack depends on the objective function $FN(\cdot)$, where it refers to the false negative on the evaluation set $\mathcal{D}_{\text{vd}}$ in this case.

Indiscriminate Causative Availability Attack

On the other hand side, indiscriminate causative availability attack (*ICAA*) differs from *ICIA*) on the security violation. It does not only affects positive samples, but on its

general predictive ability on all other data samples. Following the Eq. (3.6), we replace the objective function $FN(\cdot)$ of the attacker with classification error.

$$\mathcal{A}^* = \arg\max_{\mathcal{A}\in\Phi(\mathcal{A})} \frac{1}{2m} \sum_{i=1}^{m} |y_i - f_{\mathcal{A}}(x_i)|, \;\; \|\mathcal{D}_{eval}\| = m \text{ and } (x_i, y_i) \in \mathcal{D}_{\text{vd}} \tag{3.7}$$

After injecting A^* in the dataset $\mathcal{D}_{tr}$ on the spam filter, the availability of the service is thus violated, since it does not provide robust prediction on the data from the evaluation set $\mathcal{D}_{\text{vd}}$, which is from the same distribution as $\mathcal{D}_{tr}$.

Targeted Exploratory Integrity Attack

Another main attack explores the boundary of a classifier, for instance, the targeted exploratory integrity attack (*TEIA*) explores a set of attack $\mathcal{A}$ consisting of a number of positive samples (spams), which are misclassified as legitimate. The exploratory attack does not have impact on the spam filter itself, therefore it does not necessarily require the information about feature set $\mathcal{S}(\mathcal{D}_{tr})$ and the learning algorithm $h(\mathcal{D}_{tr})$. Nevertheless, the attacker can submit sufficiently large number of queries $\mathcal{D}_q$ to the spam filter, and the responses $\boldsymbol{t}_q$ are as well available. Suppose the attacker creates a set of attacks $\Phi(\mathcal{A}) \in \mathcal{D}_q$, which are sent over to the spam filter. Optimal attack $\mathcal{A}^*$ is computed by maximizing the false negative on $\mathcal{A}^*$, which always carry malicious content designed by the attacker.

$$\mathcal{A}^* = \arg\max_{\mathcal{A}\in\Phi(\mathcal{A})} FN(y_i, f(x_i)), \;\; (x_i, y_i) \in \mathcal{A} \tag{3.8}$$

The only variable in *TEIA* is the attack samples $\mathcal{A}$. Since the number of permitted queries $\mathcal{D}_q$ is sufficiently large, this allows the attack samples $\mathcal{A}$ to explore the data space separating by the classifier f without much constraints. The attacker can introduce arbitrary modification on $\mathcal{A}$ as long as they carry the spam content (e.g., advertisement, attachment), and this is particularly realistic by adding a few legitimate words to obfuscate a normal email.

Targeted Exploratory Availability Attack

The targeted exploratory availability attack (*TEAA*) is less common than previous three attacks. Replacing the objective function in Eq. (3.8), we obtain the attack objective for *TEAA*.

$$\mathcal{A}^* = \arg\max_{\mathcal{A}\in\Phi(\mathcal{A})} \frac{1}{2m} \sum_{i=1}^{m} |y_i - f(x_i)|, \;\; \|\mathcal{A}\| = m \text{ and } (x_i, y_i) \in \mathcal{A} \tag{3.9}$$

The main difference with *TEIA* is that the attack $\mathcal{A}$ explores the data space in more directions. The attacks obfuscate themselves not only as normal samples but also as

malicious samples. This is very useful when the attacker aims to modify a legitimate email such that it will not be received by the recipients. If the network traffic is eavesdropped, the attacker can introduce arbitrary spam content to legitimate emails, so that the emails are misclassified as spams without directly modifying the spam filter. The consequence can be significantly harmful since the attackers might not be detected at all for quite a long time.

In summary we have introduced the most frequently four adversarial attacks, although there exists other combination of attacks, *e.g.*, causative attack can be targeted on a specific subset of samples, and also the exploratory attack might be also interested at dataset from a certain distribution generally. However, in most cases, we do expect the causative attack affects the classifier directly, and it leads to system degeneration generally on unseen data. For the exploratory attack, it explores mainly the boundary of the learner and will not affect the legitimate function at all, therefore it is more possibly targeted. In the next chapters we extend our analysis on adversarial learning from both causative and exploratory attacks, which constitute the foundation of this research field.

Chapter 4

Causative Attacks

In the taxonomy of adversarial attacks, we have shown that there exists two main streams of attacks invoking researchers' attention in recent years, namely the causative attack and exploratory attack. In this chapter we introduce the causative attack in details, more specifically, real adversarial attack examples on certain learning algorithms are expanded for deeper understanding the mechanism of causative attacks. We believe that only by knowing the attackers' strategy we would be able to depict an entire picture of the arms race games between the defenders and the attackers. In this proactive arms race problem, we take the role of an adversary and conduct effective attacks on targeted systems, as well as the valuable assets under protection.

Causative attacks (may also often be referred as poisoning attack) will fundamentally change the properties of the targeted classifier, whose prediction might fail specifically or generally on data samples. Recent years research in this field has successfully presented the proof-of-concept in many literatures. For instance, it has been shown that it is possible to gradually *poison* a spam filter, an intrusion detection system, and even a biometric verification system (more in general a classification algorithm) by exploiting their update mechanisms that enable the adversary to manipulate some portion of the underlying training data [90, 91, 106, 17, 14, 65, 66, 20, 13]; As advocated in a recent workshop [62], poisoning attacks are an emerging security threat for data-driven technologies, and could become the most relevant one in the coming years, especially in the so-called big-data scenario dominated by data-driven technologies. From a practical perspective, poisoning attacks are already a pertinent scenario in several applications. In collaborative spam filtering, classifiers are retrained on emails labeled by end users. Attackers owning an authorized email account protected by the same anti-spam filter may thus arbitrarily manipulate emails in their inbox, *i.e.*, part of the training data used to update the classifier. Some systems may even ask directly to users to validate their decisions on some submitted samples, and use their feedback to update the classifier (see, *e.g.*, PDFRate, an online tool for detecting PDF malware designed by [116]). Furthermore, in several cases obtaining accurate labels, or validating the available ground

truth may be expensive and time-consuming; *e.g.*, if malware samples are collected from the Internet, by means of honeypots, *i.e.*, machines that purposely expose known vulnerabilities to be infected by malware [119], or other online services, like VirusTotal,[1] labeling errors are possible. Work by [106] and [90] has shown the potential of poisoning attacks against PCA-based malicious traffic detectors and spam filters, and proposed robust techniques to mitigate their impact. More recently, [83] have demonstrated how to poison latent Dirichlet allocation to drive its selection of relevant topics towards the attacker's choice.

It is thus not absurd to assume that a privileged adversary may partially gain access to the training dataset, which may or may not be extremely confidential to the learning system. Although in fact, poisoning attacks by manipulation of the training data have been differently countered with data sanitization (*i.e.*, a form of outlier detection) [34, 90, 91], multiple classifier systems [11], and robust statistics [106]. And robust statistics have also been exploited to formally show that the *influence function* of SVM-like algorithms can be bounded under certain conditions [32]; *e.g.*, if the kernel is bounded. Despite the fact that this ensures some degree of robustness against small perturbations of training data, it may still be desirable to improve the security of learning algorithms against poisoning.

To study the poisoning (causative) attacks, we firstly investigate a theoretically solid defined learning model, namely Support Vector Machines (SVMs), and then we extend our analysis on embedded feature selection algorithms, which are also regarded as very solid machine learning technique combining feature selection and training. Causative attacks on these learning algorithms present a message that learning algorithms may not always deliver promising results on predicting unseen data if the stationary assumption does not hold any longer due to the perturbation induced by adversaries. Algorithms discussed in this chapter may represent a family of learning models, namely the *Tikhonov regularization*. It has been proved that many machine learning algorithms can be converted as some kind of regularization problem. Therefore we believe our state-of-art analysis of causative attacks will have larger impact on intelligent systems reasonably.

4.1 Causative Label Flips Attacks on SVMs

We start our first security analysis on Support Vector Machines (SVMs), which are regarded as one of the most frequently used classifiers[2] by practitioners. We investigate the vulnerability of SVMs to a specific kind of training data manipulation (Poisoning), *i.e.*, worst-case label noise. This can be regarded as a carefully-crafted causative attack where the partial labels of the training data are purposely reversed to maximize SVM's classification error. Note that we will intensively use the term *poisoning* instead of

[1] `http://virustotal.com`

[2] SVMs can also be applied for regression and clustering problem, namely Support Vector Regression and Support Vector Clustering.

causative, since it is more straightforward to show the essential of the attack itself, the term *Poisoning* indicates the training dataset is contaminated by some malicious activities. In our case, a portion of training labels are flipped.

While stochastic label noise has been widely studied in the machine learning literature, to account for different kinds of potential labeling errors in the training data [64, 45], only a few works have considered adversarial, worst-case label noise, neither from theoretical [26] nor practical perspective [19, 140]. In [64, 26] the impact of stochastic and adversarial label noise on the classification error have been theoretically analyzed for the *probably-approximately-correct* learning model, deriving lower bounds on the classification error as a function of the fraction of flipped labels η; In particular, the test error can be lower bounded by $\eta/(1-\eta)$ and 2η for stochastic and adversarial label noise, respectively. In recent work [19, 140], instead, we have focused on deriving more practical attack strategies to maximize the test error of an SVM given a permitted number of allowed label flips in the training data. Given that the cost of flipping training labels can be very expensive, it is reasonable to constrain the attacker with a budget of maximal number of label flipping. However, finding the worst (most effective) label flips is generally computationally demanding, we have devised proper heuristic methods to find approximate solutions more efficiently. By polluting the training data partially, the SVMs will retrain on the contaminated dataset and finally update its classification boundary which affects future predictions in general, therefore we categorize our label flips attack as a Indiscriminate Causative Availability Attack (*ICAA*). To our knowledge, these are the only works devoted to understanding how SVMs can be affected by adversarial label noise.

From a more practical viewpoint, the problem is of interest, since attackers may concretely have access and change some of the training labels in a number of cases. For instance, if feedback from end-users is exploited to label data and update the system, as in collaborative spam filtering, an attacker may have access to an authorized account (*e.g.*, an email account protected by the same anti-spam filter), and manipulate the labels assigned to her samples. In other cases, a system may even ask directly users to validate its decisions on some submitted samples, and use them to update the classifier (see, *e.g.*, `PDFRate`,[3] an online tool for detecting PDF malware [116]). The practical relevance of poisoning attacks has also been recently discussed in the context of the detection of malicious crowdsourcing websites that connect paying users with workers willing to carry out malicious campaigns (*e.g.*, spam campaigns in social networks) — a recent phenomenon referred to as *crowdturfing*. In fact, administrators of crowdturfing sites can intentionally pollute the training data used to learn classifiers, as it comes from their websites, thus being able to launch poisoning attacks [132].

We start our work on poisoning SVMs [19, 140] by describing our previously-defined attacks (Sects. 4.1.3 and 4.1.5), and then propose two heuristic approaches. One has been inspired from previous work on SVM poisoning [20] and incremental learning [28, 39],

[3] Available at: `http://pdfrate.com`

and makes use of a continuous relaxation of the label values to greedily maximize the SVM's test error through gradient ascent (Sect. 4.1.4). The other exploits a breadth first search to greedily construct sets of candidate label flips that are *correlated* in their effect on the test error (Sect. 4.1.6). As in [19, 140], we aim at analyzing the maximum performance degradation incurred by an SVM under adversarial label noise, to assess whether these attacks can be considered a relevant threat.

In this worst-case study, we assume that the attacker has perfect knowledge of the attacked system and of the training data, and we left the investigation on how to develop such attacks under limited knowledge of the training data for the future work. For simplicity, we further assume that the adversary share the same cost of flipping each label, independently from the corresponding data point. In practice, this might not be true for both positive and negative samples. We demonstrate the effectiveness of the proposed causative attacks by reporting experiments on both synthetic and real-world datasets (Sect. 4.1.7). Finally we conclude in Sect. 4.1.8 with a discussion on the contributions of this work, its limitations, and future research, also relationship to the application of other techniques, including semi-supervised and active learning.

4.1.1 Support Vector Machines and Notation

We revisit here structural risk minimization and SVM learning, and introduce the framework that will be used to motivate our attack strategies for adversarial label noise.

In risk minimization, the goal is to find a hypothesis $f : \mathcal{X} \to \mathcal{Y}$ that represents an unknown relationship between an input $\mathcal{X}$ and an output space $\mathcal{Y}$, captured by a probability measure P. Given a non-negative *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ assessing the error between the prediction $\hat{y}$ provided by f and the true output y, we can define the optimal hypothesis $f^\star$ as the one that minimizes the expected risk

$$R(f, P) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[\ell\left(f(\boldsymbol{x}), y\right)\right]$$

over the hypothesis space $\mathcal{F}$, *i.e.*, $f^\star = \arg\min_{f\in\mathcal{F}} R(f, P)$.

Although P is not usually known, and thus $f^\star$ can not be computed directly, a training dataset $\mathcal{D}_{\text{tr}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ of i.i.d. samples drawn from P are often available. In many cases a learning algorithm h can be used to find a suitable hypothesis. According to structural risk minimization [128], the learner h minimizes a sum of a regularizer and the empirical risk over the data:

$$h(\mathcal{D}_{\text{tr}}) = \arg\min_{f\in\mathcal{F}} \quad \left[\Omega(f) + C \cdot \hat{R}(f, \mathcal{D}_{\text{tr}})\right], \tag{4.1}$$

where the regularizer $\Omega(f)$ is used to penalize excessive hypothesis complexity and avoid overfitting, and the empirical risk $\hat{R}(f, \mathcal{D}_{\text{tr}})$ is given by

$$\hat{R}(f, \mathcal{D}_{\text{tr}}) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(f(\boldsymbol{x}_i), y_i\right), \tag{4.2}$$

and $C > 0$ is a parameter that controls the trade-off between minimizing the empirical loss and the complexity of the hypothesis. Given a reasonable large set of training dataset $\mathcal{D}_{\text{tr}}$, the empirical risk is upper bounded [129].

The SVM is an example of a binary linear classifier developed according to the aforementioned principle. It makes predictions in $\mathcal{Y} = \{-1, +1\}$ based on the sign of its real-valued discriminant function $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$; *i.e.*, $\boldsymbol{x}$ is classified as positive if $f(\boldsymbol{x}) \geq 0$, and negative otherwise. The SVM uses the hinge loss $\ell\left(f(\boldsymbol{x}), y\right) = \max\left(0, 1 - yf(\boldsymbol{x})\right)$ as a convex surrogate loss function, and a quadratic regularizer on $\boldsymbol{w}$, *i.e.*, $\Omega\left(f\right) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w}$. Thus, SVM learning can be formulated according to Eq. (4.1) as the following convex quadratic programming problem:

$$\min_{\boldsymbol{w}, b} \quad \tfrac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{i=1}^{n} \max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right) \quad . \tag{4.3}$$

In practice, the SVM generalizes the test error by trading-off between the empirical risk, and the regularization term that avoids overfitting. Geometrically the SVM separates the closest point (support vectors) of the two classes with the largest *margin* such that most of points are lying on the correct side divided by the margin. An important property of SVMs arises from their *dual* formulation, which only requires computing inner products between samples during training and classification, thus avoiding the need of an *explicit* feature representation. Accordingly, non-linear decision functions in the input space can be transformed as a linear case using *kernel* functions, *i.e.*, inner products in implicitly-mapped feature spaces. In this case, the SVM's decision function is given as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b,$$

where $k(\boldsymbol{x}, \boldsymbol{z}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{z})$ is the kernel function, and ϕ denotes the implicit feature mapping. The SVM's dual parameters $(\boldsymbol{\alpha}, b)$ are found by solving the dual *QP* problem:

$$\min_{0 \leq \boldsymbol{\alpha} \leq C} \qquad \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} - \mathbf{1}^\top \boldsymbol{\alpha} \ , \ \text{s.t. } \boldsymbol{y}^\top \boldsymbol{\alpha} = 0 \ , \tag{4.4}$$

where $\boldsymbol{Q} = \boldsymbol{y}\boldsymbol{y}^\top \circ \boldsymbol{K}$ is the label-annotated version of the (training) kernel matrix $\boldsymbol{K}$. The bias term b is computed from the corresponding Karush-Kuhn-Tucker (KKT) conditions, to satisfy the equality constraint $\boldsymbol{y}^\top \boldsymbol{\alpha} = 0$ (see, *e.g.*, [31]).

However, we are not only interested in how the hypothesis is chosen but also how it performs on a second validation or test dataset $\mathcal{D}_{\text{vd}}$, which may be possibly drawn from a different distribution $\tilde{P}$. We thus define the error measure

$$V_h\left(\mathcal{D}_{\text{tr}}, \mathcal{D}_{\text{vd}}\right) = \|f_{\mathcal{D}_{\text{tr}}}\|^2 + C \cdot \hat{R}\left(f_{\mathcal{D}_{\text{tr}}}, \mathcal{D}_{\text{vd}}\right) \quad , \tag{4.5}$$

which implicitly uses $f_{\mathcal{D}_{\text{tr}}} = h(\mathcal{D}_{\text{tr}})$. This function evaluates the structural risk of a hypothesis $f_{\mathcal{D}_{\text{tr}}}$ that is *trained* on $\mathcal{D}_{\text{tr}}$ but *evaluated* on $\mathcal{D}_{\text{vd}}$, and will form the foundation for

our label flipping attacks of training dataset poisoning. Moreover, since we are only concerned with label flips and their effect on the learner, for simplicity we use the notation $V_h(\boldsymbol{z}, \boldsymbol{y})$ to denote the above error measure when the datasets differ only in the labels $\boldsymbol{z}$ used for training and $\boldsymbol{y}$ used for evaluation; *i.e.*, $V_h(\boldsymbol{z}, \boldsymbol{y}) = V_h(\ \{(\boldsymbol{x}_i, z_i)\}\ ,\ \{(\boldsymbol{x}_i, y_i)\}\ )$.

4.1.2 Label Flips Attack Modeling on SVMs

In proactive arms race problem, we expect the classifier to simulate the adversarial behavior to analyze the learning algorithm by attacking it. To gain more insights on whether and to what extent the SVM may be affected by the presence of well-crafted mislabeled instances in the training data, we start to build an adversarial learning model. We assume the presence of an attacker whose goal is to cause a denial of service (availability), *i.e.*, to maximize the SVM's classification error generally, by changing the labels of L samples in the training set at most. Similarly to [140, 20], the adversarial learning problem can be modeled as follows.

In a higher level abstraction we assume there is some learning algorithm h known to the adversary that maps a training dataset into hypothesis space:

$$h : (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{F}$$

With a strong support of the stationary assumption, the training dataset $(\mathcal{X}, \mathcal{Y}) \in \mathcal{D}_{\text{tr}}$ follows a certain distribution which implies that there is at least one (sub-)optimal learning algorithm h^* that preserves the best generalization ability. Being disturbed by potential adversaries, the learning algorithm could be shifted within the hypothesis space $\mathcal{F}$, which could be distant from optimum. Although this mechanism could be applied on any learning algorithm, we consider the SVM as the objective, which has been already reasonably discussed above.

Consider that the learner is typically not aware of the threats induced by the adversary, who launches the attack with a simple and straightforward goal of maximizing the classification error (*i.e.*, the risk) generally on both seen and unseen data. By contaminating the training dataset, the hypothesis is selected based on the poisoned data drawn from a potentially adversarial data distribution $\hat{P}$ over $\mathcal{X} \times \mathcal{Y}$. However, the adversary's capability of manipulating the training data is bounded by forcing the adversarial data distribution $\hat{P}$ to be within a neighborhood of (*i.e.*, "sufficiently close to") the original distribution P, such that it will not deviate too much from the real observations. Also for a worst-case label flip attack, the attacker is restricted to only change the labels of training samples in $\mathcal{D}_{\text{tr}}$ and is only allowed to change at most L labels to achieve the maximal classification risk of h — L bounds the attacker's capability, and it is fixed a

priori. Thus, the problem can be formulated as

$$\begin{aligned} \boldsymbol{y}' = \arg\max_{\boldsymbol{z}\in\mathcal{Y}^n} \quad & R\left(f_{\boldsymbol{z}}(\boldsymbol{x}), P\right), \\ \text{s.t.} \quad & f_{\boldsymbol{z}} = h\left(\{(\boldsymbol{x}_i, z_i)\}_{i=1}^n\right) \ , \\ & \textstyle\sum_{i=1}^n \mathbf{1}_{z_i \neq y_i} \leq L \ , \end{aligned} \tag{4.6}$$

where $\mathbf{1}$ is the indicator function, which returns one if the argument is true, and zero otherwise. However, as with the learner, the true risk R cannot be assessed because P is also unknown to the adversary. As with the learning paradigm described above, the risk used to select $\boldsymbol{y}'$ can be approximated using by the regularized empirical risk with a convex loss. Thus the objective in Eq. (4.6) becomes simply $V_h\left(\boldsymbol{z}, \boldsymbol{y}\right)$ where, notably, the empirical risk is measured with respect to the true dataset $\mathcal{D}_{\text{tr}}$ with the original labels. For the SVM and hinge loss, this yields the following problem:

$$\begin{aligned} \max_{\boldsymbol{z}\in\{-1,+1\}^n} \quad & \tfrac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} + C\textstyle\sum_{i=1}^n \max(0, 1 - y_i f_{\boldsymbol{z}}(\boldsymbol{x}_i)) \ , \\ \text{s.t.} \quad & (\boldsymbol{\alpha}, b) = h_{SVM}\left(\{(\boldsymbol{x}_i, z_i)\}_{i=1}^n\right) \ , \\ & \textstyle\sum_{i=1}^n \mathbf{1}_{z_i \neq y_i} \leq L \ , \end{aligned} \tag{4.7}$$

where $f_{\boldsymbol{z}}(\boldsymbol{x}) = \sum_{j=1}^n z_j \alpha_j K(\boldsymbol{x}_j, \boldsymbol{x}) + b$ is the SVM's dual discriminant function learned on the tainted data with labels $\boldsymbol{z}$. Note that the difference in the tainted dataset lies in the labels $\boldsymbol{z}$ only, which is a measurable variable.

Nevertheless the optimization in Eq. (4.7) involves a subset selection problem of flipped labels $\boldsymbol{z}$ and a second level minimization problem for SVM learning, this is a $\mathcal{NP}$-hard problem which can not be easily solved. Therefore in the next sections we present a set of heuristic methods to find approximate solutions to the posed problem efficiently; in particular, in Sect. 4.1.3 we propose a fundamental label flips attack assuming the classifier does not change during the label flips, while the effectiveness of attack is evaluated afterwards under retraining. In Sect. 4.1.4 we present a similar but novel approach for adversarial label flips using continuous label relaxation of Problem. Furthermore in Sect. 4.1.5 we present an improved, modified version of the approach in [19], and in Sect. 4.1.6 we finally present another, novel approach for adversarial label flips that aims to invert clusters of labels that are 'correlated' in their adversarial effect.

4.1.3 Adversarial Label Flip Attack

We propose firstly a near-optimal label flip attack named Adversarial Label Flip Attack (`alfa`). It is formulated under the assumption that the attacker can maliciously manipulate the set of labels to maximize the empirical loss of the original classifier on the tainted dataset, while the classification algorithm preserves its generalization on the tainted dataset without noticing it. The consequence of this attack misleads the classifier

to an erroneous shift of the decision boundary, which most deviates from the untainted original data distribution.

As discussed above, given the 'clean' dataset $\mathcal{D}_{\text{tr}}$ with n labels $\boldsymbol{y}$, the adversary aims to flip at most L labels to form a set of tainted labels $\boldsymbol{z}$ that maximize the empirical risk $V_h(\boldsymbol{z}, \boldsymbol{y})$. Alternatively, we can pose this problem as a search for labels $\boldsymbol{z}$ that achieve the maximal discrepancy between the empirical risk evaluated on $\boldsymbol{z}$ and $\boldsymbol{y}$, respectively. The attacker's objective can thus be expressed as

$$\begin{aligned} \min_{\boldsymbol{z}\in\{-1,+1\}^n} \quad & V_h(\boldsymbol{z}, \boldsymbol{z}) - V_h(\boldsymbol{y}, \boldsymbol{z}) \ , \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^n \mathbf{1}_{z_i \neq y_i} \leq L \ . \end{aligned} \tag{4.8}$$

To solve this problem, we note that the $\hat{R}(f, \mathcal{D}_{\text{vd}})$ component of V_h is a sum of losses over the data points, and the evaluation set $\mathcal{D}_{\text{vd}}$ only differs in its labels. Thus, for each data point x_i, either we will have either the term $\ell(f(\boldsymbol{x}_i), y_i)$ or $\ell(f(\boldsymbol{x}_i), -y_i)$ contributing to the risk. Introducing an indicator vector $\boldsymbol{q} = \{q_i = +1 \text{ or } -1\}_{i=1}^n$, the attacker's objective can be rewritten as the problem of minimizing the following expression with respect to $\boldsymbol{q}$ and f:

$$\mathcal{W} = \arg\min_{\boldsymbol{q}} \left(\|f\|^2 + C \sum_{i=1}^{n} \begin{array}{l} (1-q_i) \cdot [\ell(f(\boldsymbol{x}_i), y_i) - \ell(f_{\boldsymbol{y}}(\boldsymbol{x}_i), y_i)] \\ +q_i \cdot [\ell(f(\boldsymbol{x}_i), -y_i) - \ell(f_{\boldsymbol{y}}(\boldsymbol{x}_i), -y_i)] \end{array} \right) \ .$$

In this expression, the dataset is effectively duplicated and either $(\boldsymbol{x}_i, y_i)$ or $(\boldsymbol{x}_i, -y_i)$ are selected for the set $\mathcal{D}_{\text{vd}}$. The q_i variables are used to select an optimal subset of labels y_i to be flipped for optimizing f.

When `alfa` is applied to the SVM, we replace the loss function ℓ with the hinge loss and the primal SVM formulation from Eq. (4.3). We denote with $\xi_i^0 = \max(0, 1 - y_i f_{\boldsymbol{y}}(\boldsymbol{x}_i))$ and $\xi_i^1 = \max(0, 1 + y_i f_{\boldsymbol{y}}(\boldsymbol{x}_i))$ the i^{th} loss of classifier $f_{\boldsymbol{y}}$ when the i^{th} label is respectively kept unchanged or flipped. Similarly, ϵ_i^0 and ϵ_i^1 are the corresponding slack variables for the new classifier f. The `alfa` attack framework can then be expressed as:

$$\begin{aligned} \min_{\boldsymbol{q},\boldsymbol{w},\boldsymbol{\epsilon}^0,\boldsymbol{\epsilon}^1,b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \begin{bmatrix} (1-q_i) \cdot (\epsilon_i^0 - \xi_i^0) \\ +q_i \cdot (\epsilon_i^1 - \xi_i^1) \end{bmatrix} \ , \\ \text{s.t.} \quad & 1 - \epsilon_i^0 \leq y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \leq \epsilon_i^1 - 1, \quad \epsilon_i^0, \epsilon_i^1 \geq 0 \ , \\ & \textstyle\sum_{i=1}^n q_i \leq L, \quad q_i \in \{0, 1\} \ . \end{aligned} \tag{4.9}$$

To avoid integer programming which is generally $\mathcal{NP}$-hard, the indicator variables $\{q_i\}$ are relaxed as continuous variable in $[0, 1]$. The minimization problem in Eq. (4.9) is then decomposed into two iterative sub-problems. First, by fixing $\boldsymbol{q}$, the summands $\xi_i^0 + q_i(\xi_i^0 - \xi_i^1)$ are constant, and thus the minimization reduces to the following QP

Algorithm 1: `alfa`

Input : Clean training set $\mathcal{D}_{\text{tr}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, maximum number of label flips L.
Output: Tainted training set $\mathcal{D}'_{\text{tr}}$.

Find $f_{\boldsymbol{y}}$ by solving Eq. (4.3) on $\mathcal{D}_{\text{tr}}$; `/* QP */`
foreach $(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{tr}$ **do**
  $\xi_i^0 \leftarrow \max(0, 1 - y_i f_{\boldsymbol{y}}(\boldsymbol{x}_i))$;
  $\xi_i^1 \leftarrow \max(0, 1 + y_i f_{\boldsymbol{y}}(\boldsymbol{x}_i))$;
  $\epsilon_i^0 \leftarrow 0$, and $\epsilon_i^1 \leftarrow 0$;
repeat
  Find $\boldsymbol{q}$ by solving Eq. (4.11); `/* LP */`
  Find $\boldsymbol{\epsilon}^0, \boldsymbol{\epsilon}^1$ by solving Eq. (4.10); `/* QP */`
until *convergence*;
$v \leftarrow$`Sort(`$[q_1, \ldots, q_n]$, "descend"`)`;
`/*` v `are sorted indices from` $n+1$ `*/`
for $i \leftarrow 1$ **to** n **do** $z_i \leftarrow y_i$;
for $i \leftarrow 1$ **to** L **do** $z_{v[i]} \leftarrow -y_{v[i]}$;
return $\mathcal{D}'_{\text{tr}} \leftarrow \{(\mathbf{x}_i, z_i)\}_{i=1}^n$;

problem:

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\epsilon}^0, \boldsymbol{\epsilon}^1, b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\left[(1-q_i)\epsilon_i^0 + q_i\epsilon_i^1\right] \ , \\ \text{s.t.} \quad & 1 - \epsilon_i^0 \leq y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \leq \epsilon_i^1 - 1, \quad \epsilon_i^0, \epsilon_i^1 \geq 0 \ . \end{aligned} \tag{4.10}$$

Second, fixing $\boldsymbol{w}$ and b yields a set of fixed hinge losses, $\boldsymbol{\epsilon}^0$ and $\boldsymbol{\epsilon}^1$. The minimization over (continuous) $\boldsymbol{q}$ is then a linear programming problem (LP):

$$\begin{aligned} \min_{\boldsymbol{q} \in [0,1]^n} \quad & C\sum_{i=1}^{n}(1-q_i)(\epsilon_i^0 - \xi_i^0) + q_i(\epsilon_i^1 - \xi_i^1) \ , \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^n q_i \leq L \ . \end{aligned} \tag{4.11}$$

After convergence of this iterative approach, the largest subset of $\boldsymbol{q}$ corresponds to the near-optimal label flips within the budget L. The complete `alfa` procedure is given as Algorithm 1.

4.1.4 ALFA with Continuous Label Relaxation (alfa-cr)

The underlying idea of the method presented in this section is to solve Eq. (4.7) using a continuous relaxation of the problem. In particular, we relax the constraint that the tainted labels $\boldsymbol{z} \in \{-1, +1\}^n$ have to be discrete, and let them take continuous real

values on a bounded domain. We thus maximize the objective function in Eq. (4.7) with respect to $\boldsymbol{z} \in [z_{\min}, z_{\max}]^n \subseteq \mathbb{R}^n$. Within this assumption, we optimize the objective through a simple gradient-ascent algorithm, and iteratively map the continuous labels to discrete values during the gradient ascent. The gradient derivation and the complete attack algorithm are reported in Sects. 4.1.4.

Gradient Computation

Let us first compute the gradient of the objective in Eq. (4.7), starting from the loss-dependent term $\sum_i \max(0, 1 - y_i f_{\boldsymbol{z}}(\boldsymbol{x}_i))$. Although this term is not differentiable when $yf_{\boldsymbol{z}}(\boldsymbol{x}) = 1$, it is possible to consider a subgradient that is equal to the gradient of $1 - yf_{\boldsymbol{z}}(\boldsymbol{x})$, when $yf_{\boldsymbol{z}}(\boldsymbol{x}) < 1$, and to 0 otherwise. The gradient of the loss-dependent term is thus given as:

$$\frac{\partial}{\partial \boldsymbol{z}} \left(\sum_i \max(0, 1 - y_i f_{\boldsymbol{z}}(\boldsymbol{x}_i)) \right) = -\sum_{i=1}^{n} \delta_i \frac{\partial v_i}{\partial \boldsymbol{z}} \ , \tag{4.12}$$

where $\delta_i = 1$ (0) if $y_i f_{\boldsymbol{z}}(\boldsymbol{x}_i) < 1$ (otherwise), and

$$v_i = y_i \left(\sum_{j=1}^{n} K_{ij} z_j(\boldsymbol{z}) \alpha_j(\boldsymbol{z}) + b(\boldsymbol{z}) \right) - 1 \ , \tag{4.13}$$

where we explicitly account for the dependency on $\boldsymbol{z}$. To compute the gradient of v_i, we derive this expression with respect to each label z_ℓ in the training data using the product rule:

$$\frac{\partial v_i}{\partial z_\ell} = y_i \left(\sum_{j=1}^{n} K_{ij} z_j \frac{\partial \alpha_j}{\partial z_\ell} + K_{i\ell} \alpha_\ell + \frac{\partial b}{\partial z_\ell} \right) \ . \tag{4.14}$$

This can be compactly rewritten in matrix form as:

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}} = \left(\boldsymbol{y}\boldsymbol{z}^\top \circ \boldsymbol{K} \right) \frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{z}} + \boldsymbol{K} \circ (\boldsymbol{y}\boldsymbol{\alpha}^\top) + \boldsymbol{y} \frac{\partial b}{\partial \boldsymbol{z}} \ , \tag{4.15}$$

where, using the numerator layout convention,

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{z}} = \begin{bmatrix} \frac{\partial \alpha_1}{\partial z_1} & \cdots & \frac{\partial \alpha_1}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \alpha_n}{\partial z_1} & \cdots & \frac{\partial \alpha_n}{\partial z_n} \end{bmatrix} , \ \frac{\partial b}{\partial \boldsymbol{z}}^\top = \begin{bmatrix} \frac{\partial b}{\partial z_1} \\ \cdots \\ \frac{\partial b}{\partial z_n} \end{bmatrix} , \text{ and simil. } \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}} \ .$$

The expressions for $\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{z}}$ and $\frac{\partial b}{\partial \boldsymbol{z}}$ required to compute the gradient in Eq. (4.15) can be obtained by assuming that the SVM solution remains in equilibrium while $\boldsymbol{z}$ changes smoothly. This can be expressed as an adiabatic update condition using the technique

introduced in [28, 39], and exploited in [20] for a similar gradient computation. Observe that for the *training* samples, the KKT conditions for the optimal solution of the SVM training problem can be expressed as:

$$\boldsymbol{g} = \boldsymbol{Q}\boldsymbol{\alpha} + \boldsymbol{z}b - 1 \begin{cases} \text{if } g_i > 0,\ i \in \mathcal{R} \\ \text{if } g_i = 0,\ i \in \mathcal{S} \\ \text{if } g_i < 0,\ i \in \mathcal{E} \end{cases} \tag{4.16}$$

$$h = \boldsymbol{z}^\top \boldsymbol{\alpha} = 0 \ , \tag{4.17}$$

where we remind the reader that, in this case, $\boldsymbol{Q} = \boldsymbol{z}\boldsymbol{z}^\top \circ \boldsymbol{K}$. The equality in conditions Eqs. (4.16)-(4.17) implies that an infinitesimal change in $\boldsymbol{z}$ causes a smooth change in the optimal solution of the SVM, under the constraint that the sets $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{E}$ do not change. This allows us to predict the *response* of the SVM solution to the variation of $\boldsymbol{z}$ as follows.

By differentiation of Eqs. (4.16)-(4.17), we obtain:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}} = \boldsymbol{Q}\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{z}} + \boldsymbol{K} \circ (\boldsymbol{z}\boldsymbol{\alpha}^\top) + \boldsymbol{z}\frac{\partial b}{\partial \boldsymbol{z}} + \boldsymbol{S}\,, \tag{4.18}$$

$$\frac{\partial h}{\partial \boldsymbol{z}}^\top = \boldsymbol{z}^\top \frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{z}} + \boldsymbol{\alpha}^\top\,, \tag{4.19}$$

where $\boldsymbol{S} = \mathrm{diag}(\boldsymbol{K}(\boldsymbol{z} \circ \boldsymbol{\alpha}) + b)$ is an n-by-n diagonal matrix, whose elements $S_{ij} = 0$ if $i \neq j$, and $S_{ii} = f_{\boldsymbol{z}}(\boldsymbol{x}_i)$ elsewhere.

The assumption that the SVM solution does not change structure while updating $\boldsymbol{z}$ implies that

$$\frac{\partial \boldsymbol{g}_s}{\partial \boldsymbol{z}} = 0, \ \ \frac{\partial h}{\partial \boldsymbol{z}} = 0, \tag{4.20}$$

where s indexes the *margin* support vectors in $\mathcal{S}$ (from the equality in condition Eq. (4.16)). In the sequel, we will also use r, e, and n, respectively to index the *reserve* vectors in $\mathcal{R}$, the *error* vectors in $\mathcal{E}$, and all the n training samples. The above assumption leads to the following linear problem, which allows us to predict how the SVM solution changes while $\boldsymbol{z}$ varies:

$$\begin{bmatrix} \boldsymbol{Q}_{ss} & \boldsymbol{z}_s \\ \boldsymbol{z}_s^\top & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial \boldsymbol{\alpha}_s}{\partial \boldsymbol{z}} \\ \frac{\partial b}{\partial \boldsymbol{z}} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{K}_{sn} \circ (\boldsymbol{z}_s \boldsymbol{\alpha}^\top) + \boldsymbol{S}_{sn} \\ \boldsymbol{\alpha}^\top \end{bmatrix} \ . \tag{4.21}$$

The first matrix can be inverted using matrix block inversion [79]:

$$\begin{bmatrix} \boldsymbol{Q}_{ss} & \boldsymbol{z}_s \\ \boldsymbol{z}_s^\top & 0 \end{bmatrix}^{-1} = \frac{1}{\zeta} \begin{bmatrix} \zeta \boldsymbol{Q}_{ss}^{-1} - \boldsymbol{v}\boldsymbol{v}^\top & \boldsymbol{v} \\ \boldsymbol{v}^\top & -1 \end{bmatrix}, \tag{4.22}$$

where $\boldsymbol{v} = \boldsymbol{Q}_{ss}^{-1}\boldsymbol{z}_s$ and $\zeta = \boldsymbol{z}_s^\top \boldsymbol{Q}_{ss}^{-1}\boldsymbol{z}_s$. Substituting this result to solve Eq. (4.21), one

obtains:

$$\frac{\partial \boldsymbol{\alpha}_s}{\partial \boldsymbol{z}} = \left(\frac{1}{\zeta}\boldsymbol{v}\boldsymbol{v}^\top - \boldsymbol{Q}_{ss}^{-1}\right)\left(\boldsymbol{K}_{sn} \circ (\boldsymbol{z}_s\boldsymbol{\alpha}^\top) + \boldsymbol{S}_{sn}\right) - \frac{1}{\zeta}\boldsymbol{v}\boldsymbol{\alpha}^\top , \tag{4.23}$$

$$\frac{\partial b}{\partial \boldsymbol{z}} = -\frac{1}{\zeta}\boldsymbol{v}^\top\left(\boldsymbol{K}_{sn} \circ (\boldsymbol{z}_s\boldsymbol{\alpha}^\top) + \boldsymbol{S}_{sn}\right) + \frac{1}{\zeta}\boldsymbol{\alpha}^\top . \tag{4.24}$$

The assumption that the structure of the three sets $\mathcal{S}, \mathcal{R}, \mathcal{E}$ is preserved also implies that $\frac{\partial \boldsymbol{\alpha}_r}{\partial \boldsymbol{z}} = \boldsymbol{0}$ and $\frac{\partial \boldsymbol{\alpha}_e}{\partial \boldsymbol{z}} = \boldsymbol{0}$. Therefore, the first term in Eq. (4.15) can be simplified as:

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}} = \left(\boldsymbol{y}\boldsymbol{z}_s^\top \circ \boldsymbol{K}_{ns}\right)\frac{\partial \boldsymbol{\alpha}_s}{\partial \boldsymbol{z}} + \boldsymbol{K} \circ (\boldsymbol{y}\boldsymbol{\alpha}^\top) + \boldsymbol{y}\frac{\partial b}{\partial \boldsymbol{z}} \ . \tag{4.25}$$

Eqs. (4.23) and (4.24) can be now substituted into Eq. (4.25), and further into Eq. (4.12) to compute the gradient of the loss-dependent term of our objective function.

As for the regularization term, the gradient can be simply computed as:

$$\frac{\partial}{\partial \boldsymbol{z}}\left(\frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{Q}\boldsymbol{\alpha}\right) = \boldsymbol{\alpha} \circ [\boldsymbol{K}(\boldsymbol{\alpha} \circ \boldsymbol{z})] + \frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{z}}^\top \boldsymbol{Q}\boldsymbol{\alpha} \ . \tag{4.26}$$

Thus, the complete gradient of the objective in Problem (4.7) is:

$$\nabla_{\boldsymbol{z}} V_h = \boldsymbol{\alpha} \circ [\boldsymbol{K}(\boldsymbol{\alpha} \circ \boldsymbol{z})] + \frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{z}}^\top \boldsymbol{Q}\boldsymbol{\alpha} - C\sum_{i=1}^{n} \delta_i \frac{\partial v_i}{\partial \boldsymbol{z}} \ . \tag{4.27}$$

The structure of the SVM (*i.e.*, the sets $\mathcal{S}, \mathcal{R}, \mathcal{E}$) will clearly change while updating $\boldsymbol{z}$, hence after each gradient step we should re-compute the optimal SVM solution along with its corresponding structure. This can be done by re-training the SVM from scratch at each iteration. Alternatively, since our changes are *smooth*, the SVM solution can be more efficiently updated at each iteration using an active-set optimization algorithm initialized with the $\boldsymbol{\alpha}$ values obtained from the previous iteration as a warm start [108]. Efficiency may be further improved by developing an ad-hoc incremental SVM under label perturbations based on the above equations. This however includes the development of suitable bookkeeping conditions, similarly to [28, 39], and it is thus left to future work.

Algorithm

Our attack algorithm for `alfa-cr` is given as Algorithm 2. It exploits the gradient derivation reported in the previous section to maximize the objective function $V_h(\boldsymbol{z}, \boldsymbol{y})$ with respect to continuous values of $\boldsymbol{z} \in [z_{\min}, z_{\max}]^n$. The current best set of continuous labels is iteratively mapped to the discrete set $\{-1, +1\}^n$, adding a label flip at a time, until L flips are obtained.

Algorithm 2: `alfa-cr`

Input : Untainted training set $\mathcal{D}_{\text{tr}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, maximum num. of label flips L, maximum num. of iterations N ($N \geq L$), gradient step size t.

Output: Tainted training set $\mathcal{D}'_{\text{tr}}$.

$\boldsymbol{z} \leftarrow \boldsymbol{y}$; `/* Initialize labels */`

$\boldsymbol{z}_{\text{best}} \leftarrow \boldsymbol{y}$;

$\{\boldsymbol{\alpha}, b\} \leftarrow$ learn SVM on $\mathcal{D}_{\text{tr}}$ (Eq. (4.3)) ;

$p \leftarrow 0$; `/* Number of current flips */`

$k \leftarrow 0$; `/* Number of iterations */`

while $p < L$ **do**

  $k \leftarrow k + 1$;

  `/* Compute gradient from Eq.` (4.12) `using current SVM solution */`

  $\boldsymbol{z} \leftarrow \boldsymbol{z} + t\nabla_{\boldsymbol{z}} V_h$;

  Project $\boldsymbol{z}$ onto the feasible domain $[z_{\min}, z_{\max}]^n$;

  $\{\boldsymbol{\alpha}, b\} \leftarrow$ update SVM solution on $\{\boldsymbol{x}_i, z_i\}_{i=1}^n$;

  if $V_h(\boldsymbol{z}, \boldsymbol{y}) \geq V_h(\boldsymbol{z}_{\text{best}}, \boldsymbol{y})$ **then**

    $\boldsymbol{z}_{\text{best}} \leftarrow \boldsymbol{z}$; `/* Best (soft) labels */`

  if $\mod(k, \lfloor N/L \rfloor) = 0$ **then**

    `/* Project the best (soft) labels to` p `(hard) label flips */`

    $p \leftarrow p + 1$; `/* Update flip count */`

    $\boldsymbol{z} \leftarrow$ Flip the first p labels from $\boldsymbol{y}$ according to the descending order of $|\boldsymbol{z}_{\text{best}} - \boldsymbol{y}|$;

return $\mathcal{D}'_{\text{tr}} \leftarrow \{(\mathbf{x}_i, z_i)\}_{i=1}^n$;

4.1.5 Hyperplane Tilting Attack (alfa-tilt)

We now propose a modified version of the adversarial label flip attack we presented in [19]. The underlying idea of the original strategy is to generate different candidate sets of label flips according to a given heuristic method (explained below), and retain the one that maximizes the test error, similarly to the objective of Eq. (4.7). However, instead of maximizing the test error directly, here we consider a surrogate measure, inspired by the work in [92]. It is shown that, under the *agnostic* assumption that the data is uniformly distributed in feature space, the SVM's robustness against label flips can be related to the change of the angle between the hyperplane $\boldsymbol{w}$ obtained in the absence of attack, and the one learnt on the tainted data with label flips $\boldsymbol{w}'$. Accordingly, the `alfa-tilt` strategy considered here, aims to maximize the following quantity:

$$\max_{\boldsymbol{z} \in \{-1,+1\}^n} \frac{\langle \boldsymbol{w}', \boldsymbol{w} \rangle}{||\boldsymbol{w}'||||\boldsymbol{w}||} = \frac{\boldsymbol{\alpha}'^{\top} \boldsymbol{Q}_{\boldsymbol{zy}} \boldsymbol{\alpha}}{\sqrt{\boldsymbol{\alpha}'^{\top} \boldsymbol{Q}_{\boldsymbol{zz}} \boldsymbol{\alpha}'} \sqrt{\boldsymbol{\alpha}^{\top} \boldsymbol{Q}_{\boldsymbol{yy}} \boldsymbol{\alpha}}} \ , \tag{4.28}$$

Algorithm 3: `alfa-tilt` [19]

Input : Untainted training set $\mathcal{D}_{\rm tr} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, maximum num. of label flips L, maximum num. of trials N, and weighting parameters β_1 and β_2.
Output: Tainted training set $\mathcal{D}'_{\rm tr}$.

$\{\boldsymbol{\alpha}, b\} \leftarrow$ learn SVM on $\mathcal{D}_{\rm tr}$ (Eq. 4.3) ;
for $i = 1, \ldots, n$ **do**
  $s_i \leftarrow y_i[\sum_{j=1}^n y_j \alpha_j K(x_i, x_j) + b]$
normalize $\{s_i\}_{i=1}^n$ dividing by $\max_{i=1,\ldots,n} s_i$;
$(\boldsymbol{\alpha}^{\rm rnd}, b^{\rm rnd}) \leftarrow$ generate a random SVM (draw $n+1$ numbers from a uniform distribution) ;
for $i = 1, \ldots, n$ **do**
  $q_i \leftarrow y_i[\sum_{j=1}^n y_j \alpha_j^{\rm rnd} K(x_i, x_j) + b^{\rm rnd}]$
normalize $\{q_i\}_{i=1}^n$ dividing by $\max_{i=1,\ldots,n} q_i$;
for $i = 1, \ldots, n$ **do**
  $v_i \leftarrow \alpha_i / C - \beta_1 s_i - \beta_2 q_i$
$(k_1, \ldots, k_n) \leftarrow$ sort $(v_1, \ldots, v_n)$ in ascending order, and return the corresponding indexes ;
$\boldsymbol{z} \leftarrow \boldsymbol{y}$;
for $i = 1, \ldots, L$ **do**
  $z_{k_i} = -z_{k_i}$
train an SVM on $\{\mathbf{x}_i, z_i\}_{i=1}^n$;
estimate the hyperplane tilt angle using Eq. (4.28) ;
repeat N times from 5, selecting $\boldsymbol{z}$ to maximize Eq. (4.28) ;
return $\mathcal{D}'_{tr} \leftarrow \{\boldsymbol{x}_i, z_i\}_{i=1}^n$;

where $\boldsymbol{Q}_{\boldsymbol{uv}} = \boldsymbol{K} \circ (\boldsymbol{u}\boldsymbol{v}^\top)$, being $\boldsymbol{u}$ and $\boldsymbol{v}$ any two sets of training labels, and $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ are the SVM's dual coefficients learnt from the untainted and the tainted data, respectively.

Candidate label flips are generated as explained in [19]. Labels are flipped with non-uniform probabilities, depending on how well the corresponding training samples are classified by the SVM learned on the untainted training set. We thus increase the probability of flipping labels of reserve vectors (as they are reliably classified), and decrease the probability of label flips for margin and error vectors (inversely proportional to α). The former are indeed more likely to become margin or error vectors in the SVM learnt on the tainted training set, and, therefore, the resulting hyperplane will be closer to them. This will in turn induce a significant change in the SVM solution, and, potentially, in its test error. We further flip labels of samples in different classes in a correlated way to force the hyperplane to rotate as much as possible. To this aim, we draw a random hyperplane $\boldsymbol{w}_{\rm rnd}$, $b_{\rm rnd}$ in feature space, and further increase the probability of flipping the label of a positive sample $\boldsymbol{x}^+$ (respectively, a negative one $\boldsymbol{x}^-$), if $\boldsymbol{w}_{\rm rnd}^\top \boldsymbol{x}^+ + b_{\rm rnd} > 0$

$(\boldsymbol{w}_{\text{rnd}}^{\top}\boldsymbol{x}^{-} + b_{\text{rnd}} < 0)$.

The full implementation of `alfa-tilt` is given as Algorithm 3. It depends on the parameters β_1 and β_2, which tune the probability of flipping a point's label based on how well it is classified, and how well it is correlated with the other considered flips. As suggested in [19], they can be set to 0.1, since this configuration has given reasonable results on several datasets.

4.1.6 Correlated Clusters Attack

Algorithm 4: `correlated-clusters`

Input : Untainted training set $\mathcal{D}_{\text{tr}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$, maximum number of label flips L, maximum number of iterations N ($N \geq L$).
Output: Tainted training set $\mathcal{D}'_{\text{tr}}$.

Let $err(\boldsymbol{z}) = \hat{R}\left(h(\{(\boldsymbol{x}_i, z_i\}), \mathcal{D}_{\text{tr}}\right)$;
$f_{\boldsymbol{y}} \leftarrow h(\mathcal{D}_{\text{tr}})$, $E_{\boldsymbol{y}} \leftarrow err(\boldsymbol{y})$;
$E^{\star} \leftarrow -\infty$, $\boldsymbol{z}^{\star} \leftarrow \boldsymbol{y}$;
`/* Choose random singleton clusters */`
for i=1...M **do**
  $j \leftarrow rand(1, n)$;
  $\boldsymbol{z}^{i} \leftarrow flip(\boldsymbol{y}, j)$;
  $E_i \leftarrow err(\boldsymbol{z}^{i}) - E_{\boldsymbol{y}}$;
  if $E_i > E^{\star}$ **then** $E^{\star} \leftarrow E_i$, $\boldsymbol{z}^{\star} \leftarrow \boldsymbol{z}^{i}$;
  for j=1...n **do**
    if $rand_{[0,1]} < L/n$ **then** $\Delta_{i,j} \leftarrow err(flip(\boldsymbol{z}^{i}, j))$;
    else $\Delta_{i,j} \leftarrow -\infty$;
`/* Grow new clusters by mutation */`
for t=1 ...N **do**
  $(i, j) \leftarrow \arg\max_{(i,j)} \Delta_{i,j}$ $\Delta_{i,j} \leftarrow -\infty$ $\boldsymbol{z}^{M+1} \leftarrow flip(\boldsymbol{z}^{i}, j)$;
  if $\|\boldsymbol{z}^{M+1} - \boldsymbol{y}\|_1 > 2L$ **then**
    Find best flip to reverse and flip it
  $E_{M+1} \leftarrow err(\boldsymbol{z}^{M+1}) - E_{\boldsymbol{y}}$;
  if $E_{M+1} > E^{\star}$ **then** $E^{\star} \leftarrow E_{M+1}$, $\boldsymbol{z}^{\star} \leftarrow \boldsymbol{z}^{M+1}$;
  for k=1...n **do**
    if $rand_{[0,1]} < L/n$ **then** $\Delta_{M+1,k} \leftarrow err(flip(\boldsymbol{z}^{M+1}, k))$;
    else $\Delta_{M+1,k} \leftarrow -\infty$;
  Delete worst cluster and its entries in E and Δ;
return $\mathcal{D}'_{\text{tr}} \leftarrow \{(\mathbf{x}_i, z_i^{\star})\}_{i=1}^{n}$;

Here, we explore a different approach to heuristically optimizing $V_h\left(\boldsymbol{z}, \boldsymbol{y}\right)$ that uses

a breadth first search to greedily construct subsets (or *clusters*) of label flips that are 'correlated' in their effect on V_h. Here, we use the term *correlation* loosely.

The algorithm starts by assessing how each singleton flip impacts V_h and proceeds by randomly sampling a set of P initial singleton flips to serve as initial clusters. For each of these clusters, k, we select a random set of mutations to it (*i.e.*, a mutation is a change to a single flip in the cluster), which we then evaluate (using the empirical 0-1 loss) to form a matrix Δ. This matrix is then used to select the best mutation to make among the set of evaluated mutations. Clusters are thus grown to maximally increase the empirical risk.

To make the algorithm tractable, the population of candidate clusters is kept small. Periodically, the set of clusters are pruned to keep the population to size M by discarding the worst evaluated clusters. Whenever a new cluster achieves the highest empirical error, that cluster is recorded as being the best candidate cluster. Further, if clusters grow beyond the limit of L, the best *deleterious* mutation is applied until the cluster only has L flips. This overall process of greedily creating clusters with respect to the best observed random mutations continues for a set number of iterations N at which point the best flips until that point are returned. Pseudocode for the correlated clusters algorithm is given in Algorithm 4.

4.1.7 Experiments

We evaluate the adversarial effects of various attack strategies against SVMs on both synthetic and real-world datasets. Experiments on synthetic datasets provide a conceptual representation of the rationale according to which the proposed attack strategies select the label flips. Their effectiveness, and the security of SVMs against adversarial label flips, is then more systematically assessed on different real-world datasets.

On Synthetic Datasets

To intuitively understand the fundamental concept of causative label flip attack strategies and differences of the proposed methods, we report here an experimental evaluation on two bi-dimensional datasets, where the positive and the negative samples can be perfectly separated by a linear and a parabolic decision boundary, respectively.[4] For these experiments, we learn SVMs with the linear and the RBF kernel on both datasets, using `LibSVM` [30]. We set the regularization parameter $C = 1$, and the kernel parameter $\gamma = 0.5$, based on some preliminary experiments on model selection. For each dataset, we randomly select 200 training samples, and evaluate the test error on a disjoint set of 800 samples. The proposed attacks are used to flip $L = 20$ labels in the training data (*i.e.*, a fraction of 10%), and the SVM model is subsequently updated on the tainted

[4]Data is available at `http://home.comcast.net/~tom.fawcett/public_html/ML-gallery/pages/index.html`.

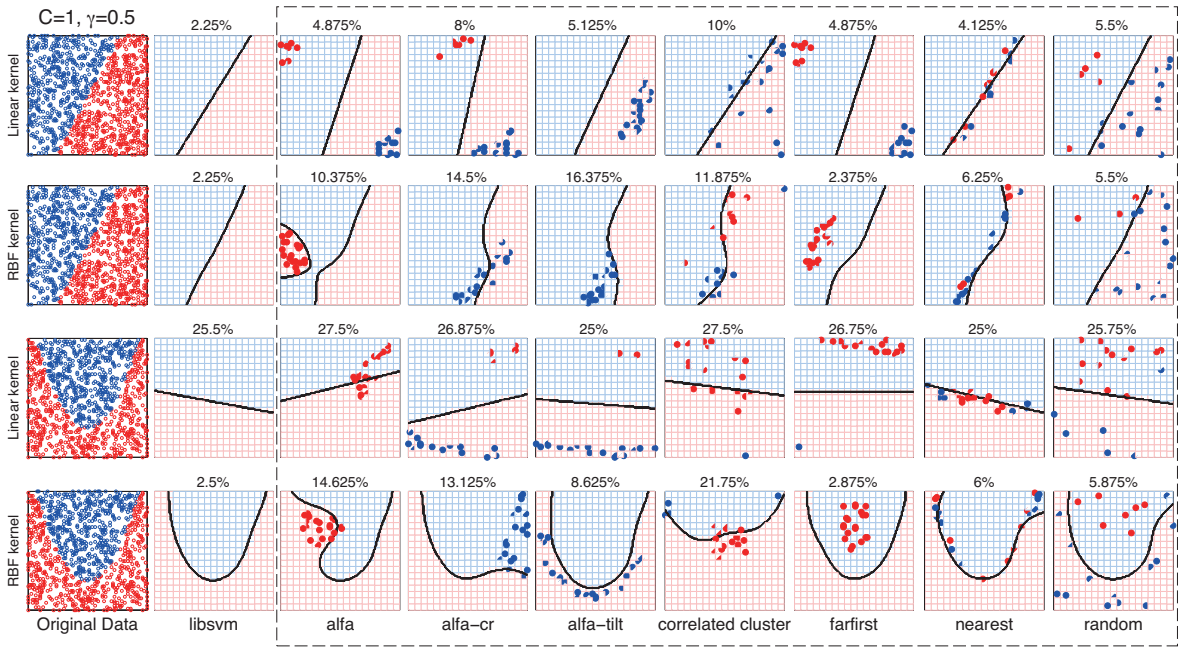


Figure 4.1: Results on synthetic datasets, for SVMs with the linear (first and third row) and the RBF (second and fourth row) kernel, trained with $C = 1$ and $\gamma = 0.5$. The original data distribution is shown in the first column (first and second row for the linearly-separable data, third and fourth row for the parabolically-separable data). The decision boundaries of SVMs trained in the absence of label flips, and the corresponding test errors are shown in the second column. The remaining columns report the results for each of the seven considered attack strategies, highlighting the corresponding $L = 20$ label flips (out of 200 training samples).

training set. Besides the four proposed attack strategies for adversarial label noise, further three attack strategies are evaluated as baselines for comparison, namely they are `farfirst`, `nearest`, and `random`. As for `farfirst` and `nearest`, only the labels of the L farthest and of the L nearest samples to the decision boundary are flipped. As for the `random` attack, L training labels are stochastically flipped. To mitigate the effect of randomization, each random attack selects the best label flips over 10 repetitions.

Results are reported in Fig. 4.1. First, note how the proposed attack strategies `alfa`, `alfa-cr`, `alfa-tilt`, and `correlated cluster` generally exhibit clearer patterns of flipped labels than those shown by `farfirst`, `nearest`, and `random`, yielding indeed higher error rates. In particular, when the RBF kernel is used, the SVM's performance is significantly affected by a careful selection of training label flips (*cf.* the error rates between the plots in the first and those in the second row of Fig. 4.1). This somehow contradicts the result in [32], where the use of bounded kernels has been advocated to improve robustness of SVMs against training data perturbations. The reason is that, in this case, the attacker does not have the ability to make unconstrained modifications to

the feature values of some training samples, but can only flip a maximum of L labels. As a result, bounding the feature space through the use of bounded kernels to counter label flip attacks is not helpful here. Furthermore, the security of SVMs may be even worsened by using a non-linear kernel, as it may be easier to significantly change (*e.g.*, "bend") a non-linear decision boundary using carefully-crafted label flips, thus leading to higher error rates. Amongst the attacks, `correlated cluster` shows the highest error rates when the linear (RBF) kernel is applied to the linearly-separable (parabolically-separable) data. In particular, when the RBF kernel is used on the parabolically-separable data, even only 10% of label flips cause the test error to increase from 2.5% to 21.75%. Note also that `alfa-cr` and `alfa-tilt` outperform `alfa` on the linearly-separable dataset, but not on the parabolically-separable data. Finally, it is worth pointing out that applying the linear kernel on the parabolically-separable data, in this case, already leads to high error rates, making thus difficult for the label flip attacks to further increase the test error (*cf.* the plots in the third row of Fig. 4.1).

On Real-World Datasets

We report now a more systematic and quantitative assessment of our label flip attacks against SVMs, considering five real-world datasets publicly available at the `LibSVM` website.[5] In these experiments, we aim at evaluating how the performance of SVMs decreases against an increasing fraction of adversarially flipped training labels, for each of the proposed attacks. This will indeed allow us to assess their effectiveness as well as the security of SVMs against adversarial label noise. For a fair comparison, we randomly selected 500 samples from each dataset, and select the SVM parameters from $C \in \{2^{-7}, 2^{-6}, \ldots, 2^{10}\}$ and $\gamma \in \{2^{-7}, 2^{-6}, \ldots, 2^{5}\}$ by a 5-fold cross validation procedure. The characteristics of the datasets used, along with the optimal values of C and γ as discussed above, are reported in Table 4.1. We then evaluate our attacks on a separate test set of 500 samples, using 5-fold cross validation. The corresponding average error rates are reported in Fig. 4.2, against an increasing fraction of label flips, for each considered attack strategy, and for SVMs trained with the linear and the RBF kernel.

The reported results show how the classification performance is degraded by the considered attacks, against an increasing percentage of adversarial label flips. Among the considered attacks, `correlated cluster` shows an outstanding capability of subverting SVMs, although requiring significantly increased computational time. In particular, this attack is able to induce a test error of almost 50% on the *dna* and *seismic* data, when the RBF kernel is used. Nevertheless, `alfa` and `alfa-tilt` can achieve similar results on the *acoustic* and *ijcnn1* data, while being much more computationally efficient. In general, all the proposed attacks show a similar behavior for both linear and RBF kernels, and lead to higher error rates on most of the considered real-world datasets than the trivial attack strategies `farfirst`, `nearest`, and `random`. For instance, when 20%

[5] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

Characteristics of the real-world datasets		
Name	Feature set size	SVM parameters
dna	124	Linear kernel $C = 0.0078$
		RBF kernel $C = 1, \gamma = 0.0078$
acoustic	51	Linear kernel $C = 0.016$
		RBF kernel $C = 8, \gamma = 0.062$
ijcnn1	23	Linear kernel $C = 4$
		RBF kernel $C = 64, \gamma = 0.12$
seismic	51	Linear kernel $C = 4$
		RBF kernel $C = 8, \gamma = 0.25$
splice	60	Linear kernel $C = 0.062$
		RBF kernel $C = 4, \gamma = 0.062$

Table 4.1: Feature set sizes and SVM parameters for the real-world datasets.

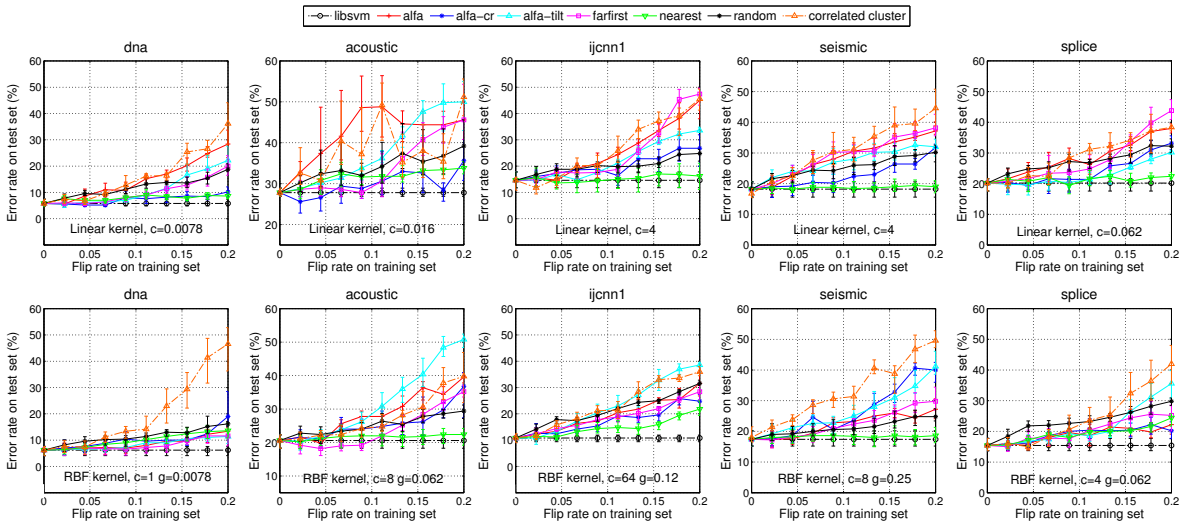


Figure 4.2: Results on real-world datasets (in different columns), for SVMs with the linear (first row) and the RBF (second row) kernel. Each plot shows the average error rate ($\pm$ *half* standard deviation, for readability) for each attack strategy, estimated from 500 samples using 5-fold cross validation, against an increasing fraction of adversarially flipped labels. The values of C and γ used to learn the SVMs are also reported for completeness (*cf.* Table 4.1).

of the labels are flipped, `correlated cluster`, `alfa`, `alfa-cr`, and `alfa-tilt` almost achieve an error rate of 50%, while `farfirst`, `nearest` and `random` hardly achieve an error rate of 30%. It is nevertheless worth remarking that *farfirst* performs rather well against linear SVMs, while being not very effective when the RBF kernel is used. This reasonably means that non-linear SVMs may not be generally affected by label flips that are *far* from the decision boundary.

To summarize, our results demonstrate that SVMs can be significantly affected by the

presence of well-crafted, causative adversarial label flips in the training data, which can thus be considered a relevant and practical security threat in application domains where attackers can tamper the training data.

4.1.8 Discussion

Although (stochastic) label noise has been well studied especially in the machine learning literature (*e.g.*, see [45] for a survey), to our knowledge few have investigated the robustness of learning algorithms against well-crafted, malicious label noise attacks. In this work, we have focused on the problem of learning with label noise from an adversarial perspective. In particular, we have discussed a framework that encompasses different label noise attack strategies, presented several causative attack strategies that can significantly worsen the SVM's classification performance on untainted test data, even if only a small fraction of the training labels are manipulated by the attacker.

An interesting future extension of this work may be to consider adversarial label noise attacks where the attacker has limited knowledge of the system, *e.g.*, when the feature set or the training data are not completely known to the attacker, to see whether the resulting error remains considerable also in more practical attack scenarios. Another limitation that may be easily overcome in the future is the assumption of equal cost for each label flip. In general, indeed, a different cost can be incurred depending on the feature values of the considered sample.

We nevertheless believe that this work provides an interesting starting point for future investigations on this topic, and may serve as a foundation for designing and testing SVM-based learning algorithms to be more robust against a deliberate label noise injection. To this end, inspiration can be taken from previous work on robust SVMs to stochastic label noise [121, 19]. Alternatively, one may exploit our framework to simulate a *zero-sum game* between the attacker and the classifier, that respectively aim to maximize and minimize the classification error on the untainted test set. This essentially amounts to re-training the classifier on the tainted data, *having knowledge* of which labels might have been flipped, to learn a more secure classifier. Different game formulations can also be exploited if the players use non-antagonistic objective functions, as in [24].

We finally argue that our work may also provide useful insights for developing novel techniques in machine learning areas which are not strictly related to adversarial learning, such as semi-supervised and active learning. In the former case, by turning the maximization in Eq. (4.6)-(4.7) into a minimization problem, one may find suitable label assignments $\boldsymbol{z}$ for the unlabeled data, thus effectively designing a semi-supervised learning algorithm. Further, by exploiting the continuous label relaxation of Sect. 4.1.4, one can naturally implement a *fuzzy* approach, mitigating the influence of potentially outlying instances. As for active learning, minimizing the objective of Eq. (4.6)-(4.7) may help identifying the training labels which may have a higher impact on classifier training, *i.e.*, some of the most informative ones to be queried. Finally, we conjecture

that our approach may also be applied in the area of structured output prediction, in which semi-supervised and active learning can help solving the inference problem of finding the best structured output prediction approximately, when the computational complexity of that problem is not otherwise tractable.

4.2 Poisoning Attacks on Embedded Feature Selection

Causative attack examples on SVMs in Sect.4.1 vividly show us that the potential adversaries can induce significant impact on current state-of-art classifier, *i.e.*, on binary SVMs. Label contamination can also be a threat to many supervised learning methods given that the adversaries' behavior is extremely adaptive even to a single learning model. A straightforward attack aiming at classification error can generate different variations of achieving their goals. For instances, to maximize SVM's classification error generally, adversaries can perform the label continuous relaxation, or force the hyperplane to tilt, they can also form so to speak the correlated clusters of labels, which is shown to be very counterproductive. The worst-case study on binary SVMs indeed disclose some vulnerability of learning models under causative attacks. However, is it only effective on SVMs? How does the causative attack perform on other learning methods. In this section, we move our focus on another widely adopted learning method, more precisely, on the embedded feature selection (*EFS*) algorithms.

In many data-driven applications, the challenge is that of inferring actionable knowledge from a large, usually high-dimensional data collection, to correctly prevent malware (*i.e.*, malicious software) infections or other threats. For instance, detection of malware in PDF files relies on the analysis of the PDF logical structure, which consists of a large set of different kinds of objects and metadata, yielding a high-dimensional data representation [116, 81, 80, 130]. Similarly, text classifiers for spam filtering rely on the construction of a large dictionary to identify words that are mostly discriminant of spam and legitimate emails [107, 82, 52, 105].

Due to the large number of available features, learning in these tasks is particularly challenging. Feature selection is thus a crucial step to mitigate the impact of the curse of dimensionality on classifier's generalization, and to learn efficient models providing easier-to-interpret decisions. In the last years, relevant work in the area of adversarial machine learning has proposed some pioneering methods for secure learning against particular kinds of attacks [7, 59, 16, 20, 12, 24, 49]. While the majority of work has focused on analyzing vulnerabilities of classification and clustering algorithms, only recent work has considered intrinsic vulnerabilities introduced by the use of feature selection methods. In particular, it has been shown that classifier evasion can be facilitated if features are not selected according to an adversary-aware procedure that explicitly accounts for adversarial data manipulation at test time [74, 131, 147]. Although these attacks do not directly target feature selection, but rather the resulting classification system, they highlight the need for adversarial feature selection procedures. Attacks that more ex-

plicitly target feature selection fall into the category of *poisoning attacks*. As we already know in this causative setting, the attacker has access to the training data, and possibly contaminates it to subvert or control the selection of the informative feature set.

Differently as strategies used in attacking SVMs, we now target feature selection algorithms, more precisely embedded feature selection algorithms. This is built on previously-proposed attack models for the security evaluation of supervised and unsupervised learning algorithms [16, 22, 20, 59, 7]. We then exploit causative attacks on popular embedded feature selection methods, including the so-called *least absolute shrinkage and selection operator* (LASSO) [124], *ridge* regression [58], and the *elastic net* [150]. (Sect. 4.2.2) Then we report experiments on PDF malware detection, assessing how poisoning affects both the process of feature selection and classification error (Sect. 4.2.3). Finally we conclude it by discussing our findings and contributions (Sect. 4.2.4), and sketching promising future research directions. Again in this work, we perform a indiscriminate causative availability attack ($ICAA$). We will see how it is designed differently as the causative label flips attack.

4.2.1 Embedded Feature Selection and Notation

Firstly we revisit the basics of (embedded) feature selection and its importance in machine learning. Moreover, we mainly introduce three embedded feature selection algorithms as the adversarial objective, namely LASSO, *ridge* regression and *elastic net*.

The curse of dimensionality prohibits many learning models generalizing well on high-dimensional dataset, while they usually can give a relatively better results for low-dimensional cases. Therefore methods that reduce feature size while does not compromise the predictive ability are highly desirable in those domains where they have to deal with high-dimensional data, *i.e.*, biological, document mining. It is reasonable to assume that many features are intrinsically not relevant to the learning problem we are challenging, therefore they can be safely removed without degrading the learning model itself. In many cases, feature selection methods are developed and deployed as a preliminary component for learning systems. The main principle of feature selection is to optimally choose a subset of features for a dataset that leads to a maximal generalization or a minimal expected risk. This associates closely with adversarial learning which instead maximizes the expected risk by changes values of certain features, thus furthermore changes the relevance of corresponding features.

Methodologically we categorize feature selection methods as three main type, namely, the filter methods, wrapper methods, and the embedded methods. They are fundamentally different in their way of interacting with underlying learning algorithms.

1. **Filter methods** do not incorporate any learning algorithm. They select the subset of features mainly based on the general criteria such as correlation with the output variable. They are computationally very efficient but not effective in highly complex data structure involves dependences. Therefore, they are mainly

served as a preprocessing step.

2. **Wrapper methods** literally wraps another learning algorithm to evaluate the quality of different subsets of features, disregarding the properties of the evaluating learning algorithm. Since the evaluation part is served as a black box, therefore it can be combined with any learning method in the end. Also due to its complexity of subset selection, wrapper methods are computationally expensive and in many cases get overfitting easily.

3. **Embedded methods** trades-off the computational demand and effectiveness on subset selection. The subset selection methods are 'embedded' as being also the learning methods, they are essentially not separable. Being widely studied and applied in practice, the embedded methods are now of our main interest to explore the adversarial properties when selecting feature subset.

We borrow the framework for feature selection in the work [70]. Given that a dataset drawn from a certain distribution $(\boldsymbol{x}, y) \in P$ and we assume a learning function (e.g., classification, regression) $f : \Lambda \times \mathbb{R}^n \to \mathbb{R}$ parameterized by $\boldsymbol{\alpha}$ such that we have the mapping $(\boldsymbol{\alpha}, \boldsymbol{x}) \longmapsto f(\boldsymbol{\alpha}, \boldsymbol{x})$. Moreover we denote an indicator vector $\boldsymbol{\sigma} \in \{0,1\}^n$, where variable $\sigma_i = 1$ indicates whether the i-th feature is present in the feature subset or $\sigma_i = 0$ for otherwise, $i = 1, \ldots, n$. Note that n here denotes the number of features but not the sample size. Now the embedded feature selection is to find the optimal solution $(\boldsymbol{\sigma}^*, \boldsymbol{\alpha}^*)$ that minimizes the expected risk

$$R(\boldsymbol{\alpha}, \boldsymbol{\sigma}) = \int \ell\left(f(\boldsymbol{\alpha}, \boldsymbol{\sigma} * \boldsymbol{x}), y\right) dP(\boldsymbol{x}, y) \tag{4.29}$$

where $*$ is the pointwise product and $P(\boldsymbol{x}, y)$ measures the probability of $(\boldsymbol{x}, y)$. Noticeably many features will be reduced to zero by multiplying the indicator variable $\sigma_i = 0$, therefore that produces a sparse model. In many learning problems, we place an additional constraint on the indicator vector such that $s(\boldsymbol{\sigma}) \leq \sigma_0$, where $s : [0,1]^n \to \mathbb{R}^+$ controls the sparsity of ultimate learning model. For example, if we choose the 0-*norm* as the measurement of the sparsity s, thus we have $s := \ell_0(\boldsymbol{\sigma}) \leq \sigma_0$, where the 0-*norm* counts the nonzero entries in the indicator vector $\boldsymbol{\sigma}$. Bear in mind that the distribution P is normally unknown, so as usual we are dealing with a training dataset $\mathcal{D}_{\text{tr}} = \{\mathcal{X}, \mathcal{Y}\}$, and rewrite the minimization problem in Eq. (4.29)

$$(\boldsymbol{\alpha}^*, \boldsymbol{\sigma}^*) = \arg\min R(\boldsymbol{\alpha}, \boldsymbol{\sigma}, \mathcal{X}, \mathcal{Y}) \quad \text{s.t.} \quad s(\boldsymbol{\sigma}) \leq \sigma_0. \tag{4.30}$$

Typically the minimization problem in Eq. (4.31) is very difficult to solve. However if both the R and s are convex, more specifically strict convex, we can rewrite the objective in Eq. (4.31) as

$$\min R(\boldsymbol{\alpha}, \boldsymbol{\sigma}, \mathcal{X}, \mathcal{Y}) + \lambda s(\boldsymbol{\sigma}) \tag{4.31}$$

Notably it coincides with the regularization problem introduced in Sect. 3.2.1, and there exists a unique $\lambda > 0$ such that $(\boldsymbol{\alpha}^*, \boldsymbol{\sigma}^*)$ solves the Eq. (4.31), and vice versa if we fix a $\lambda > 0$ then there exists one and only one σ_0 such that the optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{\sigma}^*)$ solve the minimization problem. The smoothness parameter λ in the regularization represents now a upper bound σ_0 of how many features can be maximally selected after training.

In this work we focus on a simple but extensively adopted linear model, namely LASSO (Least Absolute Shrinkage and Selection Operator) [124]. For comparison and completeness we also involve similar linear models such as *ridge* regression and *elastic net*. These algorithms perform feature selection by learning a linear function $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ that minimizes the trade-off between a loss function $\ell(y, f(\boldsymbol{x}))$ computed on the training data $\mathcal{D}$ and a regularization term $\Omega(\boldsymbol{w})$. The selection criterion $\mathcal{L}$ can be thus generally expressed as:

$$\min_{\boldsymbol{w},b} \mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i)) + \lambda \Omega(\boldsymbol{w}), \tag{4.32}$$

where λ is the trade-off parameter.[6] The quadratic loss $\ell(y, f(\boldsymbol{x})) = \frac{1}{2}(f(\boldsymbol{x}) - y)^2$ is used by all the three considered algorithms. As for the regularization term $\Omega(\boldsymbol{w})$, ideally, one would like to consider the ℓ_0-norm of $\boldsymbol{w}$ to exactly select a given number of features, which however makes the problem $\mathcal{NP}$-hard [88]. LASSO uses ℓ_1 regularization yielding a tighter convex relaxation to the ideal problem formulation. Ridge regression uses ℓ_2 regularization and the elastic net is a hybrid approach between the aforementioned ones, as it exploits a convex combination of ℓ_1 and ℓ_2 regularization, we list the different regularization for the three embedded methods as below.

$$\Omega(\boldsymbol{w}) = \begin{cases} \|\boldsymbol{w}\|_1^1 & \text{LASSO} \\ \|\boldsymbol{w}\|_2^2 & \text{Ridge regression} \\ \rho\|\boldsymbol{w}\|_1^1 + (1-\rho)\|\boldsymbol{w}\|_2^2 & \text{Elastic Net} \end{cases} \tag{4.33}$$

where $\rho \in (0, 1)$ is a strictly positive parameter controlling the sparsity. Eventually, if one is given a maximum number of features $k < d$ to be selected, the ones corresponding to the first k feature weights sorted in descending order of their absolute values can be thus retained, where the rest will be set to zero and the corresponding features are thus neglected.

4.2.2 A Framework of Poisoning Embedded Feature Selection

In this part we report a detailed case study on indiscriminate causative availability attacks ($ICAA$) against embedded feature selection algorithms mentioned above, including LASSO, ridge regression, and the elastic net.

[6] Note that this is equivalent to minimizing the loss subject to $\Omega(\boldsymbol{w}) \leq t$, for proper choices of λ and t [124].

In the considered setting, the **attacker's goal** is to maximally increase the classification error of these algorithms, by enforcing the selection of a wrong subset of features. As for the **attacker's knowledge**, we consider both PK and LK as discussed in Sect. 3.3. In the sequel, we consider LK attacks on the surrogate data $\hat{\mathcal{D}}$, since for PK attacks we can simply set $\hat{\mathcal{D}} = \mathcal{D}$. The **attacker's capability** amounts to injecting a maximum number of poisoning points into the training set. To estimate the classification error, the attacker can evaluate the same criterion $\mathcal{L}$ used by the embedded feature selection algorithm, on her available training set $\hat{\mathcal{D}}$, excluding the attack points (as they will not be part of the test data). The attack samples are thus kept outside from the empirical loss computation of the attacker, while they are clearly taken into account by the learning algorithm. Assuming that a single attack point $\boldsymbol{x}_c$ is added by the attacker, the **attack strategy** can be thus formulated as:

$$\max_{\boldsymbol{x}_c} \mathcal{W} = \frac{1}{m}\sum_{j=1}^{m} \ell\left(\hat{y}_j, f(\hat{\boldsymbol{x}}_j)\right) + \lambda\Omega(\boldsymbol{w}) \tag{4.34}$$

where it is worth remarking that f is learnt by minimizing $\mathcal{L}(\hat{\mathcal{D}} \cup \{\boldsymbol{x}_c\})$ (Eq. (4.32)), and thus depends on the attack point $\boldsymbol{x}_c$, as well as the corresponding $\boldsymbol{w}$ and b. The attacker's objective $\mathcal{W}$ (Eq. (4.34)) can be thus optimized by iteratively modifying $\boldsymbol{x}_c$ with a (sub)gradient-ascent algorithm, in which, at each step, the solution $\boldsymbol{w}, b$ is updated by minimizing $\mathcal{L}(\hat{\mathcal{D}} \cup \{\boldsymbol{x}_c\})$, *i.e.*, simulating the behavior of the feature selection algorithm on the poisoned data. Note that the parameters $\boldsymbol{w}, b$ estimated by the attacker are not generally the same ones estimated by the targeted algorithm. The latter will be indeed estimated by minimizing $\mathcal{L}(\mathcal{D} \cup \{\boldsymbol{x}_c\})$.

Gradient Computation. By calculating the partial derivative of Eq. (4.34) with respect to $\boldsymbol{x}_c$, and substituting $\ell(y, f(\boldsymbol{x}))$ and f with their expressions, one yields:

$$\frac{\partial \mathcal{W}}{\partial \boldsymbol{x}_c} = \frac{1}{m}\sum_{j=1}^{m} \left(f(\hat{\boldsymbol{x}}_j) - \hat{y}_j\right)\left(\hat{\boldsymbol{x}}_j^\top \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c} + \frac{\partial b}{\partial \boldsymbol{x}_c}\right) + \lambda \boldsymbol{r}\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c}\,, \tag{4.35}$$

where, for notational convenience, we set $\boldsymbol{r} = \frac{\partial \Omega}{\partial \boldsymbol{w}}$. Note that $\boldsymbol{r} = \mathrm{sub}(\boldsymbol{w})$ for LASSO, $\boldsymbol{r} = \boldsymbol{w}$ for ridge regression, and $\boldsymbol{r} = \rho\,\mathrm{sub}(\boldsymbol{w}) + (1-\rho)\boldsymbol{w}$ for the elastic net, being $\mathrm{sub}(\boldsymbol{w})$ the subgradient of the ℓ_1-norm, *i.e.*, a vector whose k^{th} component equals +1 (-1) if $w^k > 0$ ($w^k < 0$), and any value in $[-1, +1]$ if $w^k = 0$. As the subgradient is not uniquely determined, a large set of possible ascent directions should be explored, dramatically increasing the computational complexity of the attack algorithm. Further, computing $\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c}$ and $\frac{\partial b}{\partial \boldsymbol{x}_c}$ requires us to predict how the solution $\boldsymbol{w}, b$ changes while the attack point $\boldsymbol{x}_c$ is modified.

To overcome these issues, as in [28, 20], we assume that the Karush-Kuhn-Tucker (KKT) conditions under perturbation of the attack point $\boldsymbol{x}_c$ remain satisfied, similarly as the continuous label relaxation in Sect.4.1.4, *i.e.*, we adjust the solution to remain at

the optimum. At optimality, the KKT conditions for Eq. (4.32) with quadratic loss and linear f, are:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}^\top = \frac{1}{n}\sum_{i=1}^{n}\left(f(\hat{\boldsymbol{x}}_i) - \hat{y}_i\right)\hat{\boldsymbol{x}}_i + \lambda \boldsymbol{r}^\top = \boldsymbol{0}\,, \tag{4.36}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}\left(f(\hat{\boldsymbol{x}}_i) - \hat{y}_i\right) = 0\,, \tag{4.37}$$

where we transposed the first equation to have a column vector, and keep the following derivation consistent. If $\mathcal{L}$ is convex but *not differentiable* (*e.g.*, when using ℓ_1 regularization), one may express these conditions using subgradients. In this case, at optimality a necessary and sufficient condition is that at least one of the subgradients of the objective is null [23]. In our case, at optimality, the subgradient is uniquely determined from Eq. (4.36) as

$$\boldsymbol{r} = -\frac{1}{\lambda}\frac{1}{n}\sum_{i=1}^{n}\left(f(\hat{\boldsymbol{x}}_i) - \hat{y}_i\right)\hat{\boldsymbol{x}}_i^\top.$$

This allows us to drastically reduce the complexity of the attack algorithm, as we are not required to explore all possible subgradient ascent paths for Eq. (4.35), but just the one corresponding to the optimal solution.

Let us assume that the optimality conditions given by Eqs. (4.36)-(4.37) remain valid under the perturbation of $\boldsymbol{x}_c$. We can thus set their derivatives with respect to $\boldsymbol{x}_c$ to zero. After deriving and re-arranging in matrix form, one obtains:

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{\Sigma} + \lambda \boldsymbol{v} & \boldsymbol{\mu} \\ \boldsymbol{\mu}^\top & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c} \\ \frac{\partial b}{\partial \boldsymbol{x}_c} \end{bmatrix} &= -\frac{1}{n}\begin{bmatrix} \mathbf{M} \\ \boldsymbol{w}^\top \end{bmatrix}, \\
\text{where} \quad \boldsymbol{\Sigma} &= \frac{1}{n}\sum_i \hat{\boldsymbol{x}}_i \hat{\boldsymbol{x}}_i^\top \\
\boldsymbol{\mu} &= \frac{1}{n}\sum_i \hat{\boldsymbol{x}}_i \\
\mathbf{M} &= \boldsymbol{x}_c \boldsymbol{w}^\top + \left(f(\boldsymbol{x}_c) - y_c\right)\mathbb{I}
\end{aligned} \tag{4.38}$$

The term $\boldsymbol{v}$ yields zero for LASSO, the identity matrix $\mathbb{I}$ for ridge, and $(1-\rho)\mathbb{I}$ for the elastic net. The derivatives $\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c}$ and $\frac{\partial b}{\partial \boldsymbol{x}_c}$ can be finally obtained by solving the linear system given by Eq. (4.38), and then substituted into Eq. (4.35) to compute the final gradient.

Poisoning Feature Selection Algorithm. The complete poisoning attack algorithm is given as Algorithm 5, and an exemplary run on a bi-dimensional dataset is reported in Fig. 4.3. To optimize the attack with respect to multiple attack points, we choose to iteratively adjust one attack point at a time, while updating the current solution $\boldsymbol{w}, b$ at

Algorithm 5: Poisoning Embedded Feature Selection

Input : $\hat{\mathcal{D}}$ the (surrogate) training data; $\{\boldsymbol{x}_c^{(0)}, y_c\}_{c=1}^q$, the q initial attack points with (given) labels; $\beta \in (0,1)$; and σ, ε, two small positive constants.

Output: $\{\boldsymbol{x}_c\}_{c=1}^q$, the final attack points.

$p \leftarrow 0$;
repeat
  for $c = 1, \ldots, q$ **do**
    $\{\boldsymbol{w}, b\} \leftarrow$ train the classifier on $\hat{\mathcal{D}} \cup \{\boldsymbol{x}_c^{(p)}\}_{c=1}^q$.;
    Compute $\nabla\mathcal{W} = \frac{\partial \mathcal{W}(\boldsymbol{x}_c^{(p)})}{\partial \boldsymbol{x}_c}$ according to Eq. (4.35).;
    Set $\boldsymbol{d} = \Pi_{\mathcal{B}}\left(x_c^{(p)} + \nabla\mathcal{W}\right) - x_c^{(p)}$ and $k \leftarrow 0$.;
    repeat
      `/* line search to set the gradient step` η `*/`
      Set $\eta \leftarrow \beta^k$ and $k \leftarrow k + 1$;
      $\boldsymbol{x}_c^{(p+1)} \leftarrow \boldsymbol{x}_c^{(p)} + \eta\boldsymbol{d}$;
    until $\mathcal{W}(\boldsymbol{x}_c^{(p+1)}) \leq \mathcal{W}(\boldsymbol{x}_c^{(p)}) - \sigma\eta\|\boldsymbol{d}\|^2$;
  $p \leftarrow p + 1$;
until $|\mathcal{W}(\{\boldsymbol{x}_c^{(p)}\}_{c=1}^q) - \mathcal{W}(\{\boldsymbol{x}_c^{(p-1)}\}_{c=1}^q)| < \varepsilon$;
return $\{\boldsymbol{x}_c\}_{c=1}^q = \{\boldsymbol{x}_c^{(p)}\}_{c=1}^q$;

each step (this can be efficiently done using the previous solution $\boldsymbol{w}, b$ as a warm start). This gives the attack much more flexibility than a greedy strategy where points are added one at a time, and never modified after insertion. We also introduce a projection operator $\Pi_{\mathcal{B}}(\boldsymbol{x})$ to project $\boldsymbol{x}$ onto the feasible domain $\mathcal{B}$; *e.g.*, if features are normalized in $[0,1]$, one may consider $\mathcal{B}$ as the corresponding box-constrained domain. This enables us to define a feasible descent direction $\boldsymbol{d}$ within the given domain $\mathcal{B}$, and perform a simple line search to set the gradient step size η.

Descent in Discrete Spaces. If feature values are discrete, it is not possible to follow the gradient-descent direction exactly, as it may map the given sample to a set of non-admissible feature values. It can be however exploited as a search heuristic. Starting from the current sample, one may generate a set of candidate neighbors by perturbing only those features of the current sample which correspond to the maximum absolute values of the gradient, one at a time, in the correct direction. Eventually, one should update the current sample to the neighbor that attained the maximum value of $\mathcal{W}$, and iterate until convergence.

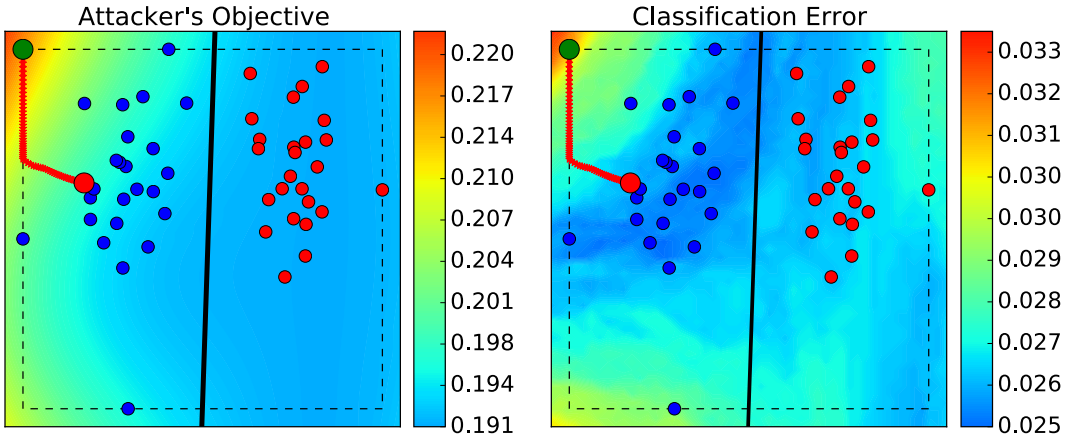


Figure 4.3: Poisoning LASSO. Red and blue points are the positive ($y = +1$) and negative ($y = -1$) training data $\mathcal{D}$. The decision boundary at $f(\boldsymbol{x}) = 0$ (for $\lambda = 0.01$, in the absence of attack) is shown as a solid black line. The solid red line highlights the path followed by the attack point $\boldsymbol{x}_c$ (*i.e.*, the magnified red point) towards a maximum of $\mathcal{W}(\boldsymbol{x}_c)$ (shown in colors in the *left* plot), which also corresponds to a maximum of the classification error (*right* plot). A box constraint is also considered (dashed black square) to bound the feasible domain (*i.e.*, the attack space).

4.2.3 Experiments

In this section, we consider an application example involving the detection of malware in PDF files, *i.e.*, one among the most recent and relevant threats in computer security [61]. The underlying reason is that PDF files are excellent carriers for malicious code, due to the flexibility of their logical structure, which allows embedding of several kinds of resources, including `Flash`, `JavaScript` and even executable code. Resources are simply embedded by specifying their type with *keywords*, and inserting the corresponding content in *data streams*. For instance, an embedded resource in a PDF may look like:

```
13 0 obj << /Kids [ 1 0 R 11 0 R ]
/Type /Page ... >> end obj
```

where keywords are highlighted in bold face. Recent work has promoted the use of machine learning to detect malicious PDF files (apart from legitimate PDFs), based on the analysis of their logical structure and, in particular, of the present keywords rather than the content of data streams [116, 81, 80, 130].

Experimental setup. In our experiments, we exploit the feature representation proposed by [81], where each feature simply denotes the number of occurrences of a given keyword in the PDF file. We collected 5993 recent malware samples from the *Contagio* dataset,[7] and 5951 benign samples from the web. As a preliminary step, following

[7] `http://contagiodump.blogspot.it`

the procedure described by [81], we extracted keywords from the first 1,000 samples in chronological order. The resulting 114 keywords were used as our initial feature set $\mathcal{X}$. We then randomly sampled five pairs of training and test sets from the remaining data, respectively consisting of 300 and 5,000 samples, to average the final results. To simulate LK attacks, we also sampled an additional set of five training sets (to serve as $\hat{\mathcal{D}}$) consisting of 300 samples each. We normalized each feature between 0 and 1 by bounding the maximum keyword count to 20, and dividing each feature value by the same value. This value was selected to restrict the attacker's capability of manipulating data to a large extent, without affecting generalization accuracy in the absence of attack.[8] We evaluate the impact of poisoning against LASSO, ridge and elastic net. We first set $\rho = 0.5$ for the elastic net, and then optimized the regularization parameter λ for all methods by retaining the best value over the entire regularization path [46, 99]. We evaluate our results by reporting the classification error as a function of the percentage of injected poisoning samples, which was increased from 0% to 20% (where 20% corresponds to adding 75 poisoning samples to the initial data). Furthermore, to understand how feature selection and ranking are affected by the attack, we also evaluate the consistency index originally defined by [69] to evaluate the stability of feature selection under random perturbations of the training data.

Kuncheva's Stability Index. Given two feature subsets $A, B \subseteq \mathcal{X}$, with $\|A\| = \|B\| = k$, $r = \|A \cap B\|$, and $0 < k < \|\mathcal{X}\| = d$, it is defined as:

$$I_C(A, B) = \frac{rd - k^2}{k(d - k)} \in [-1, +1]\,, \tag{4.39}$$

where positive values indicate similar sets, zero is equivalent to random selections, and negative values indicate strong anti-correlation between the feature subsets. The underlying idea of this stability index is to normalize the number of common features in the two sets (*i.e.*, the cardinality of their intersection) using a correction for chance that accounts for the average number of common features randomly selected out of k trials.

To evaluate how poisoning affects the feature selection process, we compute this index using for A a feature set selected in the absence of poisoning, and comparing it against a set B selected under attack, at different percentages of poisoning. To compare subsets of equal size k, for each method, we considered the first k features exhibiting the highest absolute weight values. As suggested by [69], all the corresponding pairwise combinations of such sets were averaged, to compute the expected index value along with its standard deviation.

Experimental results. Results are reported in Fig. 4.4, for both the PK and LK settings. No substantial differences between these settings are highlighted in our results, meaning that the attacker can reliably construct her poisoning attacks even without having access to the training data $\mathcal{D}$, but only using surrogate data $\hat{\mathcal{D}}$. In the absence of

[8] If no bound on the keyword count is set, an attacker may add an unconstrained number of keywords and arbitrarily influence the training process.

attack (*i.e.*, at 0% poisoning), all methods exhibit reliable performances and a very small classification error. Poisoning up to 20% of the training data causes the classification error to increase of approximately 10 times, from 2% to 20% for LASSO, and slightly less for elastic net and ridge, which therefore exhibit slightly improved robustness properties against this threat. For comparison, we also considered a *random* attack that generates each attack point by randomly cloning a point in the training data and flipping its label. As one may note from plots in the first column of Fig. 4.4, our poisoning strategy is clearly much more effective. Besides the impact of poisoning on the classification error,

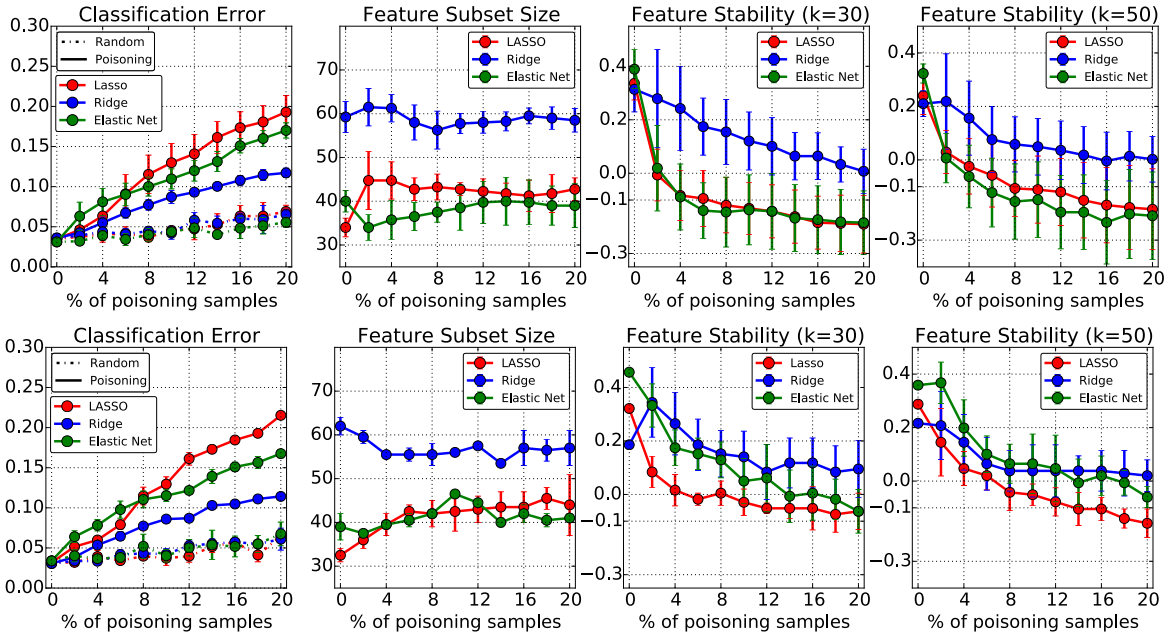


Figure 4.4: Results on PDF malware detection, for PK (*top row*) and LK (*bottom row*) poisoning attacks against LASSO, ridge, and elastic net, in terms of classification error (*first column*), number of automatically selected features (*second column*), and stability of the top $k = 30$ (*third column*) and $k = 50$ (*fourth column*) selected features, against an increasing percentage of injected poisoning samples. For comparison, we also report the classification error attained by all methods against random label-flip attacks (*first column*). All the reported values are averaged over five independent runs, and error bars correspond to their standard deviation.

the most significant result of our analysis is related to the impact of poisoning on feature selection. In particular, from Fig. 4.4 (third and fourth column), one can immediately note that the (averaged) stability index (Eq. (4.39)) quickly decreases to zero (especially for LASSO and the elastic net) even under a very small fraction of poisoning samples. This means that, in the presence of few poisoning samples, the feature selection algorithm performs as a *random* feature selector in the absence of attack. In other words, the attacker can almost arbitrarily control feature selection. Finally, it is worth remarking

that, among the considered methods, ridge exhibited higher robustness under attack. We argue that a possible reason besides selecting larger feature subsets (see plots in the second column of Fig. 4.4) is that feature weights are more evenly spread among the features, reducing the impact of each training point on the embedded feature selection process. We discuss in detail the importance of selecting larger feature subsets against poisoning attacks in the next section.

4.2.4 Discussion

We think that our work gives a two-fold contribution to the state of the art. The first contribution is to the field of adversarial machine learning. We are the first to propose a framework to evaluate the vulnerability of feature selection algorithms and to use it to analyze poisoning attacks against popular embedded feature selection methods, including LASSO, ridge regression, and the elastic net. The second contribution concerns the robustness properties of ℓ_1 regularization. Despite our results are seemingly in contrast with the claimed robustness of ℓ_1 regularization, it is worth remarking that ℓ_1 regularization is robust against *non-adversarial* data perturbations; in particular, it aims to reduce the variance component of the error by selecting smaller feature subsets, at the expense of a higher bias. Conversely, poisoning attacks induce a systematic *bias* into the training set. This means that an attacker may more easily compromise feature selection algorithms that promote *sparsity* by increasing the bias component of the error. The fact that ℓ_1 regularization may worsen performance under attack is also confirmed by [74], although in the context of evasion attacks. Even if the underlying attack scenario is different, also evasion attacks induce a specific bias in the manipulation of data, and are thus more effective against *sparse* algorithms that exploit smaller feature sets to make decisions.

4.2.5 Conclusion

In this work, we have provided a framework that allows one to model potential causative attacks against feature selection algorithms in a consistent way, making clear assumptions on the attacker's goal, knowledge and capabilities. We have exploited this framework to characterize the relevant threat of poisoning attacks against feature selection algorithms, and reported a detailed case study on the vulnerability of popular embedded methods (LASSO, ridge, and elastic net) against these attack. Our security analysis on a real-world security application involving PDF malware detection has shown that attackers can completely control the selection of reduced feature subsets even by only injecting a small fraction of poisoning training points, especially if *sparsity* is enforced by the feature selection algorithm.

This demands for the engineering of *secure* feature selection algorithms against poisoning attacks. To this end, one may follow the intuition behind the recently-proposed

feature selection algorithms to contrast evasion attacks, *i.e.*, to model interactions between the attacker and the feature selection algorithm [74, 131, 147]. Recent work on robust LASSO and robust regression may be another interesting future direction to implement secure feature selection against poisoning [87, 95]. From a more theoretical perspective, it may be of interest to analyze: (*i*) the impact of poisoning attacks on feature selection in relation to the ratio between the training set size and the dimensionality of the feature set; and (*ii*) the impact of poisoning and evasion on the bias-variance decomposition of the mean squared error. These aspects may reveal additional interesting insights also for designing secure feature selection procedures.

To this end we have seen causative attack examples on both SVMs and embedded feature selection methods. By sharing some common properties of their learning models (*i.e.*, regularization, sparsity), we indeed see some common disadvantages of applying a sparse model, and in terms of security consideration, simulating the behaviors of attackers presents us important messages of the demanding fact that designing a robust and secure machine learning algorithm is absolutely not an illusion, but a serious challenge for researchers in this domain.

4.3 Summary

Causatively subverting the learners (classifiers) is becoming a real threat in security-sensitive domains, *i.e.*, spam filter, intrusion detection system, as we have seen in previous works. Learning algorithms such as SVMs, embedded feature selection are compromised under certain causative attacks, which explore their associated data space and break the stationary assumption by inducing a maximal classification error.

In the example of poisoning SVMs, we studied carefully the SVMs in theory, where a margin-based learner is used to linearly classify data points in a potentially high-dimensional feature space. Although the adoption of kernel methods suggest robustness against random noises, they are not quite effective when there exists the adversarial noise generated by the proposed attacks. Consider that the adaptiveness of an adversary can be very flexible even with one single attack objective to maximize the classification error, therefore the classifier needs to thoroughly simulate different types of causative attacks, *e.g.*, an iterative *min-max* attack as in `alfa` or a gradient-based label continuous relaxation. A common practice of enhancing the security of SVM-based systems is to proactively conduct such attacks and take the weakness spots (*e.g.*, the attack points) into account. For instance, the learner discovers a relatively vulnerable set of data as suggested by the attack, then a regular anomaly detection process can be applied to mitigate their adversarial effect respectively.

For the case of poisoning embedded feature selection, the consequence could be even more catastrophic when the subset selection part was compromised, since it might be served as a preprocessing step for other learning methods, which are of course also compromised. Although the sparsity of any embedded methods indeed enables efficient

computation of a high-dimensional problem, recently it has been theoretically proved that the stability of a learning model will be forced to make concession when the sparsity is taking the priority [143]. For security engineering, it is thus meaningful to consider the adversarial attacks as real practice, since the instability introduced by the attackers will eventually harm the legitimate function of a sparse model, which is determinative in embedded feature selection. To counter against the causative attacks on feature selection, we have seen in real examples that the *elastic net* enhanced the robustness and security of 1-*norm* regularized LASSO by providing a hybrid regularization of both 1-*norm* and 2-*norm* term controlled by a trade-off parameter ρ. To this end, we might also suggest similar hybrid model for SVMs to increase the security level under adversarial impact. We will leave this also as a future work.

Again we stress on the impact introduced by causative attack which causatively degrades the legitimate function of a classifier. Therefore countermeasures are highly required and should call for attention from both designers of learning system and security applications.

Chapter 5

Exploratory Attacks

In previous chapters we have investigated the adversarial learning both in theory and especially in the case of causative attack, which is recognized as one of the two most important and threatening attacks. In the sequel, we introduce another type of adversarial attack, namely, the exploratory attack, or mostly called evasion attack, that describes immediately its attack mechanism by name. Hereinafter we will adopt the term *evasion attack* more frequently than *exploratory attack* bearing in mind that they are equivalent concepts.

In causative attacks the adversaries attempt to subvert the legitimate function of a learning system (learner) by introducing poisoning data points. The effective attacks subsume retraining of learners being unconscious of the existence of the malicious subset. The effort of heuristically approximating attack points involves computationally intensive searching of the vulnerable data space aligning with the training dataset. Although these attacks cause the distortion of the targeting learning functions with a more far-reaching damage *i.e.*, random guess of a spam-filtering process, normally they could be detected in a short response time by the defenders, observing the systems in malfunction. In contrary, evasion attacks disguise malicious samples as legitimate ones to evade the detection of learners, in consequence their actions could be potentially more stealthy and private, and also difficult to be recognized. In practice, evasion attacks are mostly targeted, but not generally subverting the legitimate function of any classifier. Driven by a certain incentive, *e.g.*, promoting product sales via spam emails, adversaries aim to obfuscate specific information in a mimicry of another, and eternally get classified as their expectation. The involvement of retraining the classifier is therefore not considered in evasion attack, since their impact on the classifier is normally insignificant until a large amount of instances get evaded, however this requires exhaustive effort from the adversaries to evade them, which in reality is not possible.

In spite of the well defined types of evasion attacks in Sect. 3.3, attackers are mostly interested only at evading a certain subset of positive instances from being detected by classifier, that is, most evasion attacks arise as a targeted evasion integrity attack *TEIA*.

Again, taking the binary spam filter as an example, attackers would disguise a couple of spams as legitimate emails by adding or removing some words with a certain cost bound, as studied in works [15, 36, 67, 76]. The incentive of evading instances of certain class is apparently appealing for attackers, given that learning-based systems are extensively deployed in many domains. We study the evasion attack hereinafter in details, more specifically, we introduce firstly the definition of evasion attack in formal language and extend it with a concrete evasion example introduced in the work [12].

5.1 Evasion Attacks at Test Time

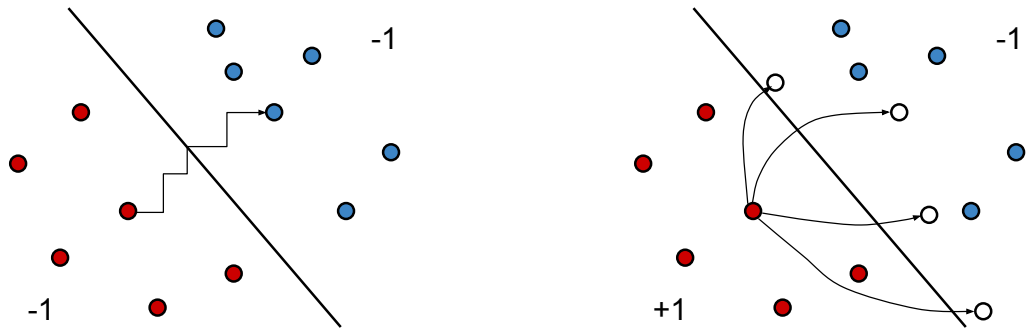
Previous work [36] in a game-theoretic setting preemptively examined the naive Bayes classifier, explored and patched the vulnerable spots in its data space, using a modified classifier to detect potentially malicious examples. Interestingly this pioneer work inspired its complementary works, where researchers start to reverse engineer classifiers by finding those blind spots. A first trial of reverse engineering a linear classifier was conducted by Lowd and Meek [76], they developed a near-optimal evasion attack on linear classifiers with both boolean and continuous input features.

Given a training dataset $\mathcal{D}_{\text{tr}}$ and its associated classifier f, an attacker tries to evade an targeted malicious sample x_a (*i.e.*, potentially a set of samples) to be classified as a negative (benign) sample. Denote an optimal data point x_a^* as the disguised version of x_a, namely by updating its corresponding features while keeping malicious content constant, we have $g(x_a^*) < 0$ and $g(x_a) > 0$ assuming $g(\cdot)$ is the discriminative function associated with f, *e.g.*, a hyperplane in $\mathcal{F}$. Here in the case of binary classification, f can be defined as an activation function, for instance, $f(g(x)) \in \{+1, -1\}$.

Recall the attack scenario given in Sect. 3.4, a targeted evasion integrity attack (*TEIA*) aims to maximize the false negative of the given set the positive attack points. Here we refine the definition of evasion attack for the binary classification problem in three axises: (**i.**) directedness (**ii.**) closeness and (**iii.**) adversarial cost.

Directedness defines the direction of the heuristics conducted by the attacker, it can either be directed or undirected as depicted in Fig. 5.1. For directed evasion attack, additional exemplary benign sample x^* is required to be known to the attacker, who eventually updates the malicious sample x_a towards x^*. Note that the evasion path is thus directed to mimicry the legitimate behavior of x^*. While for undirected evasion attack, attacker manipulates the sample x_a in an indeterministic way by probing the boundary of classifier, as investigated in [76]. It typically involves the reverse engineering of the classifier and is computationally expensive. Also in undirected evasion attack, the attacker does not mimicry any legitimate behavior as long as the the manipulated sample is classified as benign.

Closeness is a measurement evaluating how close is an attack point to its expected area. For instance in directed evasion attack, the closeness measures the distance between the optimal attack point and its exemplary benign sample, *e.g.*, $\|x_a^* - x^*\|$ uses



(a) Boolean features directed evasion

(b) Continuous features undirected evasion

Figure 5.1: Evasion attack examples for a binary classification problem. a) Direct evasion attack of a boolean valued positive sample that evades the hyperplane (classifier boundary) to mimicry the behavior of a certain benign sample. b) Undirected evasion attack of a continuous valued positive sample can go in multiple directions to get cross the hyperplane, heuristics of corresponding negative samples imply the reverse engineering of the binary classifier.

a euclidean distance to define the closeness of the attack. However in the undirected evasion attack, the closeness could be measured facilitating the so called instance of the minimal adversarial cost ($IMAC$) introduced in [76], and is served in undirected case as an exemplary benign sample. Note that closeness finally measures the effectiveness of an evasion attack, that is, the closer the optimal attack is to the exemplary instance, the more effective the evasion attack is.

Adversarial cost is a straightforward measurement of adversary's effort to achieve an effective evasion attack. For example in [76], they defined an minimal adversarial cost (MAC) of achieving a certain level of closeness. If an attacker successfully evades the classifier by a polynomial-many queries with a less than $(1-\varepsilon)MAC$ adversarial cost, we say the attack is ε-optimality. Similar concept is also proposed in [93] as a bounded cost ball $\mathfrak{B}^C$ measuring the distance between original attack point and its optimality.

In Fig. 5.1, we show two examples of evasion attacks. On the left plot the data space is defined by boolean features, initial positive sample is manipulated its boolean features to mimicry a targeted benign sample in a directed evasion setting. Note that the *zig-zag* evasion path only illustrates the discrete update of the boolean features. And the right plot instead shows a undirected evasion attack with continuous-valued features. Heuristics in different directions implies a reverse engineering of the classifier itself. Given the fact that, in practice, obtaining a legitimate instance as exemplary data is rational almost everywhere, and the computational demand for undirected evasion is normally prohibiting, and more importantly reverse engineering the classifier in most cases is not indispensable. Therefore in this work, we define, for the binary classification,

a directed evasion attack formally as follows. Without loss of generality, we assume an integrity attack.

Definition 4. *Given any set of positive attack points $\mathcal{A} = \{\boldsymbol{x}_i\}_{i=1}^m$, and at least one exemplary benign sample $\boldsymbol{x}^*$, a directed evasion attack finds a path towards $\boldsymbol{x}^*$ holding a malicious feature subset within a certain domain $\mathcal{M}$, such that it resides within a ε-close ball $\mathfrak{B}$ centered at $\boldsymbol{x}^*$, with a minimal adversarial cost bounded by τ. We call it as a ε-optimality evasion attack. The optimal attack points set $\mathcal{A}^*$ can be found by optimizing the cost function,*

$$\begin{aligned} \mathcal{A}^* &= \arg\min_{\mathcal{A}\in\Phi(\mathcal{A})} \tfrac{1}{m}\textstyle\sum_{i=1}^m L(x_i, x^*) + \lambda\mathcal{C}(\mathcal{A}) \\ s.t. \quad & \mathcal{C}(\mathcal{A}^*) \leq \tau \\ & \|x_i - x^*\|^2 \leq \varepsilon^2 \quad \forall i = 1, \ldots, m \\ & \left\{x^{\boldsymbol{d}}\right\} \subseteq \mathcal{M}, \quad \boldsymbol{d} \subseteq \{1, \ldots, m\} \end{aligned} \tag{5.1}$$

where $\Phi(\mathcal{A})$ encodes the set the heuristics of $\mathcal{A}$, L measures the closeness and $\mathcal{C}$ is the adversarial cost function. The malicious domain $\mathcal{M}$ carries malicious content encoded as feature subsets, which are pertained to $\mathcal{M}$ during the evasion attack. If the problem defined in Eq. (5.2) has at least one optimal solution (possibly many local minimum), then we say the attack is a ε-optimality evasion attack. Also we need to stress on the exemplary sample x^* which is not necessarily a benign sample, it can also be a legitimate distribution as long as x^* encodes certain benign behavior, *e.g.*, benign samples, data distribution, set of rules.

5.2 Example: Gradient-based Mimicry Evasion Attacks

To gain more insight of evasion attacks at test time, we revisit the work recently published by Biggio *et al.*[12] to guide through a possible idea of evading a binary classifier. Also we provide a discussion on the related work according to our formal definition, and point out possible open problems to be done in the future.

5.2.1 Problem Setting

For simplicity, again, authors [12] investigated their evasion attack on a binary classifier. Therefore, given a dataset $\mathcal{D}_{\text{tr}} = \{\mathcal{X}, \mathcal{Y}\}$, and $\mathcal{Y} = \{+1, -1\}$, a binary classifier $f : \mathcal{X} \to \mathcal{Y}$ is a mapping from input $\mathcal{X}$ to their labels $\mathcal{Y}$. Therefore its associated continuous discriminative function $g(x)$ defines a mapping: $\mathcal{X} \to \mathbb{R}$ such that $f(x) = +1$ if $g(x) > 0$, or $f(x) = -1$ if otherwise.

A reckless attacker was about to manipulate a positive attack point x, *i.e.*, $g(x) > 0$, to evade the classifier, originally a simple evasion strategy amounts to a undirected evasion attack, which is contented as far as $g(x^*) < 0$. This can easily be achieved by

a greedy heuristic method without much adversarial cost, however the optimal attack point expires when there is a small perturbation of the classifier, *e.g.*, retraining of the classifier. Therefore a more effective attack objective was proposed to have a high confident level of decision value $g(x) < 0$, that is, to minimize the decision function $g(x)$ bounded by a certain adversarial cost, which is defined by a distance function d.

$$d(x, x^*) < \tau$$

A distance function between the initial attack point x and the optimal one x^* describes the adversarial cost of manipulating the features of x to match x^*. Note that different features for different classes might induce different cost, hence in cost-sensitive scenarios, we can refer to a more general L_p cost function [93],

$$d(x, x^*) = \left(\sum_{i=1}^{d} c_i \|x_i - x_i^*\|^p \right)^{1/p}$$

For instances, if $p = 0$, the distance $d(x, x^*)$ measures the absolute number of different features between two instances (*nonzero* entries in $\|x - x^*\|$), while L_2 distance is simply the euclidean distance that is widely used in continuous-valued cases. In evasion attack, the L_p cost is bounded by a maximal adversarial cost τ, also note that the evasion attack in [12] is only guaranteed with a upper bound, that indicates the attacker will issue a pessimistic attack as much as possible.

5.2.2 Attack Modeling

As mentioned in Sect. 3.4 an evasion attack could have a relatively lower level of knowledge requirement from the attacker. Therefore authors [12] propose a limited knowledge attack where only feature set $\mathcal{S}(\mathcal{X})$ and a surrogate of dataset $\hat{\mathcal{D}}$ are available. The surrogate dataset is a slightly weak condition, since such kind of data can be obtained from an alternative data source which shares the same distribution as $\mathcal{D}_{\text{tr}}$. However knowing the exact feature representation $\mathcal{S}(\mathcal{X})$ normally is unrealistic and considered as optimistic in their work. Nonetheless, attacker is capable of defining an alternative feature representation $\hat{\mathcal{S}}$ over the surrogate dataset, which is then used to train an approximate discriminative function $\hat{g}$, given that direct access to g and $\mathcal{D}_{\text{tr}}$ is impossible. In order to obtain an approximation discriminative function $\hat{g}$ over the surrogate dataset $\hat{\mathcal{D}}$, a limited number of membership oracle queries to the classifier f is permitted. This assumption is considered as rational and reasonable since many classification systems eventually would not prevent users to submit system queries as a provided service. For convenience, we can simply assume that the source limit on the number of membership queries is always valid for the collected size of surrogate data samples. Evasion is then further defined with the objective over the approximate function $\hat{g}$ such that it is

minimized upper bounded by certain adversarial cost.

$$\begin{aligned} x^* = \quad & \arg\min_x \hat{g}(x) \\ \text{s.t.} \quad & d(x, x_0) \leq \tau \end{aligned} \tag{5.2}$$

where x_0 is the targeted malicious sample. Clearly, this is a undirected evasion attack which tries its best to move cross the hyperplane as distant as possible, being classified as benign. Noting that this is a nonlinear optimization problem and the discriminative function is not everywhere differentiable, the optimal attack points will very likely reach local optimums, where might be a flat area with $\Delta\hat{g} = 0$. In discriminative models, *e.g.*, one-class SVM, neural networks, the $\hat{g}$ associates with densities of data points, the flat density area may imply that that attack points fall in a sparse area but not the dense, as we can see in the Fig.5.2(a)[1], therefore they might either fail or be easily detected by any outlier removal process, given that the attack points deviate far from benign behavior.

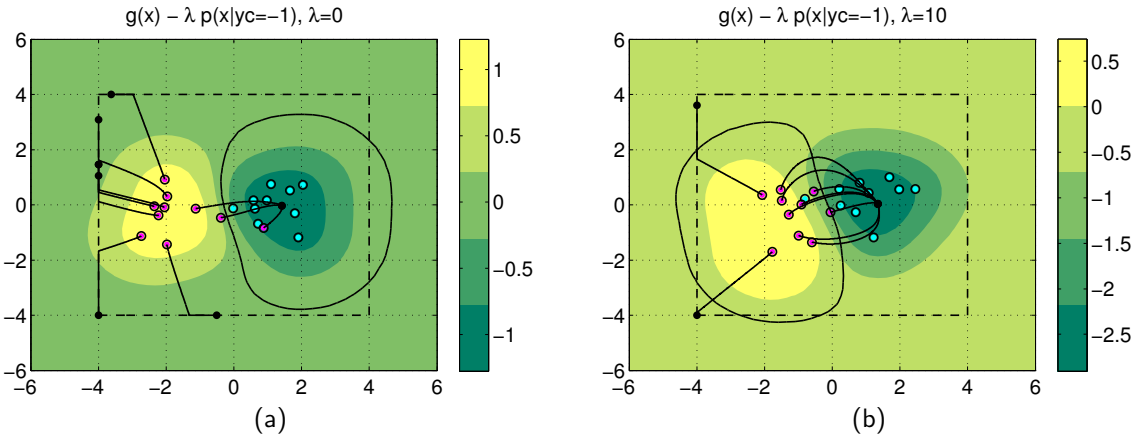


Figure 5.2: Gradient-based evasion attack on binary SVM with *RBF* kernel. a) the targeted attack points follow the gradient path to local optimums where the density is sparse, when the regularization term is set to zero. b) By setting the regularization parameter $\lambda = 10$, most of targeted attack points move to high density area of benign samples to achieve effective evasion attacks.

Observing the imperfection in the attack objective defined in Eq. (5.3), authors proposed to add an regularization favoring the attack points to move towards dense area of the surrogate dataset $\hat{\mathcal{D}}$. Using a *kernel density estimator* to compute the density of the attack point, finally it yields a modified objective function,

$$\begin{aligned} x^* = \quad & \arg\min_x \hat{g}(x) - \tfrac{\lambda}{n} \textstyle\sum_{f(x_i)=+1} k\left(\frac{x-x_i}{h}\right) \\ \text{s.t.} \quad & d(x, x_0) \leq \tau \end{aligned} \tag{5.3}$$

[1]The figures are reproduced by the open source code provided by the authors in: `https://comsec.diee.unica.it/adversarialib/`

The second regularization term in Eq. (5.4) penalizes when the attack point goes to low density area of benign samples. In Fig. 5.2(b), when the regularization parameter $\lambda = 10$, most of the targeted attack points get successfully evaded towards a dense area of the benign samples, and they form the mimicry evasion attacks at a distant and benign spot.

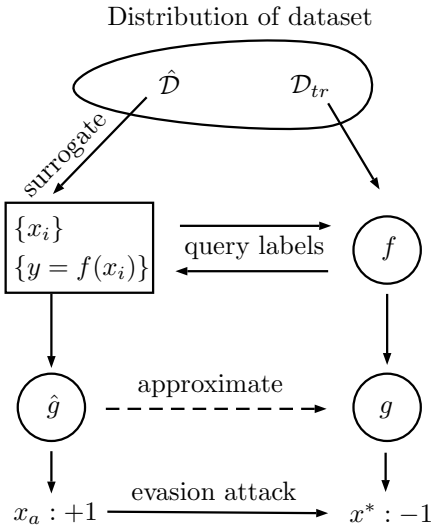


Figure 5.3: A surrogate dataset $\hat{\mathcal{D}}$ is drawn from the same distribution as training dataset $\mathcal{D}_{\text{tr}}$. An alternative decision function $\hat{g}$ is trained with labels queried from the targeted classifier f. The targeted attack point x_a finds its optimum x^* in benign class by evasion attack.

To this end, we see how different is the mimicry evasion attack from our definition in Def. 5.2. Although it seems a undirected evasion attack at the beginning, the regularization term given by the kernel density converts it as a directed evasion attack, where the closeness measures the density of the attack point x^*. Defining a compact ball $\mathfrak{B}$ over the benign samples, the objective in Eq. (5.4) forces the attack towards the center of the ball $\mathfrak{B}$. Since the attacker is only upper bounded at the adversarial cost, therefore it is not a minimal effort evasion attack as we defined in Def. 5.2, but a pessimistic attack to achieve high confidence level of being misclassified. We demonstrate the attack strategy in Fig. 5.3. Given the attacker's objective in Eq. (5.4), authors [12] applied a gradient-descent technique to find the optimum. The gradient of Eq. (5.4) resolves in two parts, namely $\Delta\hat{g}(x)$ and the gradient of the weighted kernel density, which depends on the kernel function only. If the $\hat{g}$ is differentiable, *e.g.*, SVM, neural networks, the solution can be found easily.

5.2.3 Discussion

To this end, we remark the difference of the presented evasion example and our definition shown in Def. 5.2. In the objective in Eq. (5.4), the adversary aims to update the positive attack point towards a dense area of benign samples, so that it gets a high level of confidence to be misclassified. However, the adversarial cost is merely upper bounded, that is, as long as the cost is less than some threshold τ, the evasion will not cease. This is effective in some cases, however, mostly we expect the adversaries to perform their attack to achieve maximal gain with minimal cost, as we propose in Def. 5.1. Additionally we also observed that the attacker can actually change any features during the gradient descent, this might be counterproductive in terms of adversaries, since there is no bound on the malicious content defined in domain $\mathcal{M}$. For instance, if adversaries inject malicious binaries in *PDF* files, there must be a subset of features representing the malicious binaries, which should be held within the domain $\mathcal{M}$, unless the evasion might be ill-defined.

5.3 Summary

In this chapter we introduce the evasion attack, without loss of generality *TEIA*, with formal definition and notations. Then a real example is given by the recent work presented in [12] for further analysis of evasion attack. Although evasion attacks could have variations for both availability and indiscriminate attacks, we believe that most evasion attacks performed by adversaries would be strongly interested at getting a certain subset of malicious data misclassified (evaded). Nonetheless other types of evasions can also be extended straightforwardly using the definition introduced previously.

Since more and more applications adopt learning-based components to provide predictive services, as we have seen in spam filtering, intrusion detection systems. Incentive to evade their detection also increasingly drives malicious users to disguise unintended content as legitimate information. A formal definition in Def. 5.2 provides more insights of analyzing this kind of attack, mainly from the axises of **i**) **directedness**, **ii**) **closeness**, and **iii**) **adversarial cost**. Consider that gathering legitimate data samples from targeted classifier is almost rational everywhere, directed evasion attack aiming at mimicry of legitimate behaviors is considered as a common strategy taken by adversaries. Besides, closeness and adversarial cost together measure the effectiveness of evasion attack, for instance in the gradient-based example we described, attacker attained a pessimistic attack towards a high density area of benign samples by utilizing the maximal adversarial cost. However, we remark that insidious adversaries would achieve their evasion goals with a minimal adversarial cost, other than fully utilizing the resource.

Evasion attack can be private and stealthy since it does not induce compromise of the targeted learner, therefore it is risky when such attacks can not be detected. In the gradient-based evasion example, we see that attack points move fast to the dense part of

benign samples, simulating legitimate behaviors, thus it is very difficult to be discovered by the current anomaly detectors. Although the evasion attack's impact is not as crucial as causative attack, since only a few test samples are infected, countermeasures against the attacks are still highly demanding. Due to the existence of causative and evasion attacks by highly crafty and adaptive adversaries, we believe that designing robust and secure learning algorithms is imperative.

Chapter 6

Secure Learning towards Robustness

Fast pacing development of data-driven machine learning techniques inflates nowadays many applications, for instance, spam filtering is a straightforward extension of binary classifier as a security tool to discover undesirable mails. Incentive of subverting those learning-based systems induces potentially destructive threats from those highly skillful and adaptive adversaries, who are capable of reverse engineering or evading the targeted systems. To this end, to secure the learning-based systems is currently a crucial task and open problem, considering that adversaries can be private and stealthy. Accordingly adversarial learning theory suggests a proactive strategy to face the arms race problems, the defenders exercise possible adversarial attacks at the first place before designing the learning algorithm, and then integrate their attack strategies as effective countermeasures.

In Sect. 2.2, we briefly introduced the robust learner and overfitting problem in machine learning context. In spite of adversarial or non-adversarial noise, robustness is a desirable property of any learning algorithm. For the sake of overfitting, learning algorithms are strictly evaluated by their generalization over unseen test dataset, whether they could provide reliable predictions with possible high level of perturbation. However, a secure learning algorithm demands even more rigorous level of robustness, such that even there exists a number of adversarial actions, the learning algorithm can still be reliable. Unlike stochastic (non-adversarial) noises in training dataset, adversarial samples are injected with targeted poisoning information to construct a ragged and instable data space, on which the learner is trained. Therefore, we understand the adversarial noise equivalently as the noise with a higher risk level. In our opinion, robust learning algorithms should not only be resilient to those normal data perturbation, but also effective under adversarial impact.

In this chapter, we extend our analysis of adversarial learning from the perspective of defender. Two robust learning algorithms are proposed in Sect. 6.1 and Sect. 6.2 to provide a certain level of security by involving prior knowledge about adversaries. Besides, another work on reasoning the attack also suggests another research direction

of adversarial learning, in Sect. 6.3 we adopted a Bayesian network approach to discover dependencies of factors and outcome, in order to causally analyze the precipitating factors of an abnormal behavior or an adversarial attack. Finally, we conclude our work with a discussion in Sect. 6.4.

6.1 Robust Support Vector Clustering with Indicative Labels

In previous chapters of causative attacks and exploratory attacks, adversarial attack strategies have been extensively studied by examples [138, 137, 12]. Causative attacks, or more specifically as poisoning attacks, subvert targeted classifiers, *e.g.*, SVMs, LASSO, such that legitimate function is degenerated. For evasion attacks, instead, legitimate function of classifiers is untouched, but nevertheless it generates incorrect predictions for those delicately manipulated malicious samples. Observing that adversaries usually manipulate their attack samples to locate at positions which might contradict the normal behavioral data patterns (secure patterns), therefore it is straightforward to firstly propose anomaly detection on the potentially disrupted training dataset. As a legitimate discipline to remove the effect introduced by anomalies, possibly adversarial noise, it is significant that the anomaly detection methods themselves are robust against outliers. Therefore in the sequel we investigate a robust version of a widely applied anomaly detection algorithm, namely support vector clustering (SVC), in a semi-supervised learning setting. In many learning scenarios, supervised learning is hardly applicable due to the unavailability of a complete set of data labels, while unsupervised models overlook valuable user feedback in an interactive system setting. Therefore we explore the semi-supervised learning with merely a small number of user indicated labels as supervised information, and apply the methodology on the SVC algorithm for anomaly detection. It is shown that the given labels can significantly improve the performance on detecting anomalies. Moreover, the partially labeled data proliferates the information without extra computational burden but significantly enhances the learning robustness against anomalies.

6.1.1 Related Work

Among a number of research works in the area of anomaly detection [29], unsupervised clustering methods are widely explored due to the fact that the input data does not always come along with labels. For instance, *Self-Organizing Maps* (SOM) [115] and K-means clustering [148] assume the compactness of normal data around its cluster center, whereas anomalies are sparsely settled. However, they do not support arbitrary data shape in the input space. The DBSCAN algorithm [44] detects arbitrary shapes of data in accordance with local densities, where outliers are supposed to be distributed in low density area and not belong to any cluster.

In recent years, kernel and spectral clustering methods [38] have invoked immense interest of researchers due to its non-parametric characteristics. Ben-Hur et.al [9] developed the support vector clustering (SVC) algorithm that discovers the smallest sphere in the feature space enclosing all the data points. With a delicate selection of parameters, the support vector clustering can naturally separate data samples into various classes. This support vector descriptive model shares the common core idea with the one-class support vector machine [111]. Unfortunately, the SVC has not yet been adapted to the semi-supervised mode. In many systems, we observe that some feedback can be contributed by users at a very low expense, which encourages the raise of the semi-supervised learning [149]. Especially in anomaly detection, it is highly expected that a least effort from users could improve the performance at a significant amount. The work in [56] introduces the semantic anomaly factor to measure the deviation an outlier behaves from the majority of its cluster members by inquiring their labels. Another well studied semi-supervised learning method is the semi-supervised support vector machine [113], which prevents the decision boundary from passing through high density area. A graph based semi-supervised method leverages the non-negative matrix factorization [54] to cluster data by minimizing distances of same labeled samples while maximizing distances of different labeled samples.

Despite the many work presented in this subfield of anomaly detection, very little is studied to enhance the robustness of detection process itself. Consider the adversarial learning setting, we believe that it brings enormous contribution when supervised information from legitimate users is available. For this reason, we present the semi-supervised version of support vector clustering, named as indicative support vector clustering (*iSVC*). Given a limited number of binary labels as legitimate or malicious, the support vector clustering algorithm can be guided to produce a more reliable and accurate boundary separating normal instances from anomalies. The *iSVC* reweighs part of the input data points leveraging the supervised information given by legitimate users, and then constructs the boundary alleviating the impact of anomalies on the hypersphere. To our knowledge, this is the first semi-supervised support vector clustering method to secure the anomaly detection process.

6.1.2 A Revisit of Support Vector Clustering

We revisit the theory and notation of Support Vector Clustering algorithm in this section, following the formulation introduced in Ben-Hur's work [9].

Given a data set of n points $\{x_i\}_{i=1}^n \subseteq \mathcal{X}$, it forms a d-dimensional real-valued input space with $\mathcal{X} \subseteq \mathbb{R}^d$. Defining a non-linear feature mapping Φ from $\mathcal{X}$ to a higher dimensional Hilbert space $\mathcal{H}$, the support vector clustering looks for the smallest sphere $\mathfrak{B} = \langle R, \boldsymbol{a} \rangle$ in $\mathcal{H}$ enclosing all the mapped data points, where R is the radius of the sphere and $\boldsymbol{a}$ is the center in $\mathcal{H}$. To allow soft margin, n slack variables $\{\xi_i\}_{i=1}^n$ are associated with each data point, such that the objective function is formulated as:

$$\begin{aligned} & \min R^2 + C\sum \xi_i \\ \text{subject to} \quad & \|\Phi(x_j) - \boldsymbol{a}\|^2 \leq R^2 + \xi_j,\ \forall j \\ & \xi_j \geq 0 \end{aligned} \tag{6.1}$$

To solve this problem, the corresponding Lagrangian form is introduced,

$$\begin{aligned} L = R^2 \quad & -\textstyle\sum_j \left(R^2 + \xi_j - \|\Phi(x_j) - \boldsymbol{a}\|^2\right)\beta_j \\ & -\textstyle\sum \xi_j\mu_j + C\sum \xi_j, \end{aligned} \tag{6.2}$$

where $\beta_j \geq 0$ and $\mu_j \geq 0$ are Lagrange multipliers with respect to the two constraints in Eq. (6.1), and C is the regularization parameter. Intuitively, the SVC algorithm minimizes the ball by permitting part of data lying outside of the boundary, but meanwhile penalizing their outlying. Setting the derivative of the Lagrange L to zero, it leads to the following observations,

$$\sum \beta_j = 1 \tag{6.3}$$

$$\boldsymbol{a} = \sum \beta_j \Phi(x_j) \tag{6.4}$$

$$\beta_j = C - \mu_j \tag{6.5}$$

The KKT conditions again require that

$$\xi_j\mu_j = 0 \tag{6.6}$$

$$\left(R^2 + \xi_j - \|\Phi(x_j) - \boldsymbol{a}\|^2\right)\beta_j = 0 \tag{6.7}$$

According to the Eq. (6.3)-(6.7), we have the following property of SVC,

Property 1. *Denote three disjoint set of points $\{\mathcal{S}, \mathcal{B}, \mathcal{R}\}$ are the sets for support vectors, bounded (error) support vectors, and reserved vectors respectively. A support vector $x_j \in \mathcal{S}$ lies on the sphere surface with $0 < \beta_j < C$, while a bounded support vector $x_j \in \mathcal{B}$ lies outside the sphere with $\beta_j = C$. For all the points lying inside the sphere $x_j \in \mathcal{R}$, the associated Lagrange multipliers $\beta_j = 0$.*

Eliminating the variables R, $\boldsymbol{a}$, and u_j, the Wolf dual form is that such as β_j are the only variables.

$$\mathcal{W} = \sum_j \Phi(x_j)^2\beta_j - \sum_{i,j} \beta_i\beta_j\Phi(x_i)\cdot\Phi(x_j) \tag{6.8}$$

$$\text{s.t.} \quad 0 \leq \beta_j \leq C,\ j = 1, \ldots, n. \tag{6.9}$$

The inner product of feature mapping $\Phi(x_i)\cdot\Phi(x_j)$ can be represented by a kernel function $k(x_i, x_j)$, for convenience, it is chosen as the widely adopted Gaussian kernel.

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2h^2}\right),$$

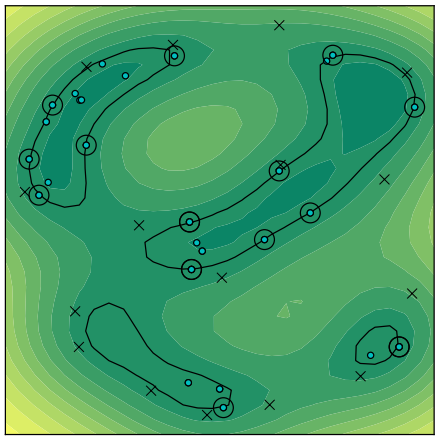


Figure 6.1: Support vector clustering on a 2-dimensional space, where data points are separated in 4 clusters by a nonlinear cluster boundary. Blue points are the reserved and support vectors, and the '×' marks the error vectors which reside out of the cluster bound.

where h is the kernel bandwidth controlling the smoothness of the clustering boundary. The Lagrangian W is then substituted as:

$$\mathcal{W} = \sum_j k(x_j, x_j)\beta_j - \sum_{i,j} \beta_j \beta_j K(x_i, x_j), \tag{6.10}$$

$$\text{subject to} \quad 0 \leq \beta_j \leq C, \; j = 1, \ldots, n. \tag{6.11}$$

Solving the above QP problem, we obtain the dual variables $\{\beta_j\}$ assigning the input data into the disjoint sets $\{\mathcal{S}, \mathcal{B}, \mathcal{R}\}$. Besides, the squared distance of any vector x to the ball center $\boldsymbol{a}$ can be computed as,

$$d(x)^2 = k(x, x) - 2\sum_j \beta_j k(x_j, x) + \sum_{i,j} \beta_i \beta_j k(x_i, x_j) \tag{6.12}$$

and the radius R is measured by any distance of support vectors to the center $\boldsymbol{a}$.

$$R = d(x_i), \qquad \exists\, x_i \in \mathcal{S}$$

Finally we obtain the cluster boundary by the radius R mapped back to the input space. An example of SVC on a 2-d toy sample is shown in Fig. 6.1, where points are separated by a nonlinear cluster boundary.

Cluster assignment

The SVC algorithm searches the smallest sphere enclosing all the points, but not classifies each point into its containing cluster. To assign the cluster labels, it requires a further geometrical analysis of the input data. For instance, two points x_i and x_j are connected with a line l, which is divided into several line segments. If any point on the segments stay outside the cluster boundary, it indicates that x_i and x_j belong to different clusters. Most efforts are dedicated to the cluster labeling process so as to improve the computational complexity. This prohibits the prevalence of SVC for clustering practitioners.

However, in anomaly detection, the cluster labeling process is not a necessity when anomalies are already identified as bounded support vectors. This characterizes the SVC as an ideal choice for anomaly detection. In the sequel, we neglect the cluster assignment of SVC, therefore convert it to an one-class classifier separating outliers from normal instances, moreover, we remark that the cluster assignment does not influence the robust enhancement to the SVC algorithm, since it is an isolated step to geometrically compute the cluster labels.

6.1.3 SVC Algorithm on Anomaly Detection

The SVC lends its advantages to anomaly detection being regarded as an equivalent one-class SVM classifier, if the kernel function $k(\boldsymbol{x}, \boldsymbol{y})$ only depends on $(\boldsymbol{x} - \boldsymbol{y})$ [110]. In this section, we explore its application for anomaly detection as well as its limitations.

The SVC forms an enclosed hypersphere separating the data into three sets, or generally speaking, two sets: $\{\mathcal{X}^+, \mathcal{X}^-\}$. We denote the positive set $\mathcal{X}^+$ for anomalies and the negative set $\mathcal{X}^-$ for normalities. The bounded support vectors (BSVs) correspond the set $\mathcal{X}^+$ which are distant from the ball center $\boldsymbol{a}$, and all the other points belong to the set $\mathcal{X}^-$, can either be a support vector or a reserved point. From the SVC optimality conditions in Eq. (6.3)-(6.7), we have following properties that hold for any anomaly $x_a \in \mathcal{X}^+$:

$$\xi_a > 0 \, , \beta_a = C \text{ and } \mu_a = 0$$

According to Eq. (6.3), the number of anomalies is upper bounded by $1/C$, where C is the penalty term.

$$n_a < 1/C$$

That is, we know that the proportion of anomalies in the input data P_a is also upper bounded by $\frac{1}{nC}$, Therefore, the penalty C and the sample size n determine a tolerance level for the outliers in the input data. Besides, the distance of an outlier x_a to the center $\boldsymbol{a}$ can be computed as:

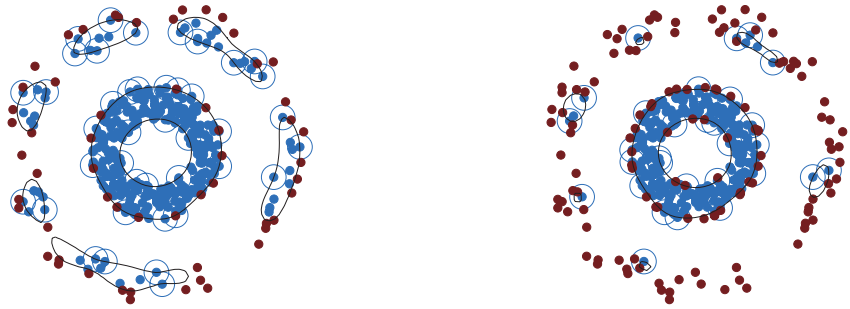
$$d(x_a) = k(x_a, x_a) - 2 \sum \beta_j k(x_a, x_j) + \sum \beta_i \beta_j k(x_i, x_j)$$

For a SVC hypersphere $\mathfrak{B} = \langle R, \boldsymbol{a} \rangle$, the third term in $d(x_a)$ is fixed, and for Gaussian kernel $k(x_a, x_a) = 1$. When the outlier x_a is distant from the center $\boldsymbol{a}$, it implies a small

value for $\sum_j \beta_j k(x_a, x_j)$, which is a weighted average of the Gaussian kernels on all the samples. When all the points are regarded as anomalies, i.e., $P_a = 1$ and $C = 1/n$, we have an approximated Parzen Window Estimate,

$$P(x_a) = \frac{1}{n} \sum_j k(x_a, x_j)$$

Thus, given a SVC hypersphere $\mathfrak{B} = \langle R, \boldsymbol{a} \rangle$, the further a point is away from the center $\boldsymbol{a}$, the lower is the density in comparison with others. This forms the fundamental idea of using SVC for anomaly detection. And moreover, the regularization parameter C sets a tolerance level to determine how many points can be maximally repelled out of the enclosing sphere, namely, identified as anomalies.



(a) $C = 0.1$ (b) $C = 0.05$

Figure 6.2: Given a kernel bandwidth $h = 1.8$, the effects of various regularization parameter C on the SVC cluster boundary and predicted anomalies, which comply with the data points residing at the low density area.

In Fig. 6.2, a set of points are distributed in two rings respectively in a 2-d space, where the inner ring is denser than the outer ring (anomalies). It is observed that an increased penalty term C (see Fig. 6.2a) imposes higher penalty on the outliers, and makes the sphere to be expanded to encircle more points. However, a mitigation of the penalty would possibly produce more unexpected outliers, see Fig. 6.2b. Consequently, this density-based SVC framework will head into a failure when the outliers are no longer sparse in the input space, which could be realistic when adversaries are aware of it. Furthermore, from Eq. (6.4), the center $\boldsymbol{a}$ of the sphere is a weighted mean of all the support vectors and bounded support vectors, and for all the β_j values of BSVs are set equally C. Therefore, for the robustness of the decision boundary, the hypersphere

$\mathfrak{B}$ is heavily biased by the anomalies, whose impacts should be excluded or at least alleviated. In adversarial setting, the SVC is formulated as a *Tikhonov regularization* problem, which could be vulnerable against causative attacks, which we have discussed in Sect. 4.1 and Sect. 4.2.

6.1.4 Indicative Support Vector Clustering

The study above reveals two critical problems of the SVC in anomaly detection applications. First, the sparsity of anomalies does not always hold, especially when the adversaries manipulates malicious samples actively to create undesirable set of anomalies. Most anomaly detection algorithms are not employable, if the anomalies form a cluster themselves. Second, the existence of anomalies will tamper with the center and radius of the hypersphere, the robustness of SVC algorithm is thus questionable when the adversarial impact of the outliers is overlooked. Therefore, we propose the semi-supervised version of the SVC algorithm by integrating user specified labels with high confidence.

Weighted Regularization for Robustness

The SVC generalizes the solution as a smallest sphere in the Hilbert space $\mathcal{H}$ by allowing a portion of points to be ruled out, but simultaneously penalizing their outlying. In accordance with the properties of SVC, the optimal ball center $\boldsymbol{a}$ is a weighted mean of all the SVs and BSVs under feature mapping Φ. Fragment the Lagrangian multipliers $\{\beta_j\}$ to three subsets as $\{\beta_{\mathcal{S}}, \beta_{\mathcal{B}}, \beta_{\mathcal{R}}\}$ in corresponds to $\{\mathcal{S}, \mathcal{B}, \mathcal{R}\}$ respectively, we have

$$\boldsymbol{a} = \sum_{x_s \in \mathcal{S}} \beta_{\mathcal{S}} \Phi(x_s) + \sum_{x_b \in \mathcal{B}} \beta_{\mathcal{B}} \Phi(x_b) + \sum_{x_r \in \mathcal{R}} \beta_{\mathcal{R}} \Phi(x_r),$$

where $0 < \beta_{\mathcal{S}} < C$, $\beta_{\mathcal{B}} = C$ and $\beta_{\mathcal{R}} = 0$. The bounded support vectors contribute even more than the support vectors on positioning the center. However, in anomaly detection or other noise-aware applications, BSVs are recognized as outliers or noises, to which a robust learning model should be more resilient. For robustness, a weighted regularization term to the original SVC objective function is thus proposed, the Lagrangian is now reformed as:

$$\begin{aligned} L = R^2 \quad & - \textstyle\sum_j \left(R^2 + \xi_j - \|\Phi(x_j) - \boldsymbol{a}\|^2\right) \beta_j \\ & - \textstyle\sum_j \xi_j \mu_j + \sum_j c_j \xi_j, \end{aligned} \tag{6.13}$$

Instead of equivalently penalizing all the input data with a constant C, the weighted regularization term $\sum c_j \xi_j$ in Eq. (6.13) treats each point individually. The objective is that, the larger the regularization coefficient c_j is, the less possible the point x_j would be driven away from the hypersphere, and vice versa. And the associated observation in Eq. (6.5) becomes:

$$\beta_j = c_j - \mu_j \tag{6.14}$$

In this way, the BSVs (anomalies) should have lower values of β_j, and the SVs or reserved points are optimized with higher values of β_j. Consequently, we alleviate the adversarial effect of the BSVs on the formation of the hypersphere $\mathfrak{B}$, and the SVs become the dominating factors leading to the decision boundary.

Integration of Indicative Labels

Assumption 1. *Anomalies are the patterns found to behave distinctly from the normal patterns, and similarly behaving instances are more likely hosted in the same cluster.*

To compute the regularization weights $\{c_j\}_{j=1}^n$, the supervised information from user given labels can be leveraged and integrated in addition to the unlabeled input data based on the assumption 1, that is, similar data patterns are more closely located to each other. Given a data set $\mathcal{X} \subseteq \mathbb{R}^d$, two supervised datasets are indicated by users,

$$\begin{aligned}\mathcal{X}^+ &= \{(x_a, y_a) \,|\, x_a \in \mathcal{X}, y_a = 1)\} \\ \mathcal{X}^- &= \{(x_r, y_r) \,|\, x_r \in \mathcal{X}, y_r = -1)\} \\ & a, r \in \{1, \ldots, n\}\end{aligned}$$

$\mathcal{X}^+$ is a labeled subset of $\mathcal{X}$ with only anomalies, while $\mathcal{X}^-$ is likewise a labeled subset with merely normal samples. By the Assumption 1, a user given label of an instance indicates the similarity of its neighborhood with a high confidence level, and can broadcast the supervised information over its neighborhood. More precisely, a given anomaly sample will increase the probability of its neighborhood of being anomalies, while a given legitimate sample will indicate that its neighborhood are more confident of being legit as well.

To leverage the favoring supervised information to obtain the regularization weights, an impact function g is defined in the input space $\mathcal{X}$ given the labeled sets $\mathcal{X}^+$ and $\mathcal{X}^-$.

$$g(x_i) = \frac{\sum_{x_a \in \mathcal{X}^+} k(x_a, x_i)}{\sum_a \mathbf{1}^+(x_a, x_i)} + \frac{\sum_{x_r \in \mathcal{X}^-} k(x_r, x_i)}{\sum_a \mathbf{1}^-(x_r, x_i)} \tag{6.15}$$

where the kernel function is taken as a similarity measurement, note that we use Gaussian kernel,

$$k(x, x_i) = \exp\left(\frac{\|x - x_i\|^2}{-2h^2}\right)$$

And $\mathbf{1}^+$ and $\mathbf{1}^-$ are both indicator functions defined as

$$\mathbf{1}^+(x_a, x_i) = \begin{cases} 1 & \|x_a - x_i\| \leq 2h \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{1}^-(x_r, x_i) = \begin{cases} -1 & \|x_r - x_i\| \leq 2h \\ 0 & \text{otherwise} \end{cases}$$

We define the radius of the affected neighborhood as $2h$ for simplicity, which means that the impact of the given labeled data is bounded. The bounds of the indicator functions $\mathbf{1}^+$ and $\mathbf{1}^-$ can also be configured individually so that we have flexibility on controlling the degree of impact by anomalies or normal samples. Besides, the similarity measurement $k(x, x_i)$ is proportional to a Gaussian distribution $\mathcal{N}(x, h)$. Therefore, the impact function $g(x_i)$ represents the probability of x_i being an anomaly affected by $\mathcal{X}^+$ or a normal instance affected by $\mathcal{X}^-$. Note that normally it is expected that a data point can either be influenced by anomalies or by normal instances, but not by both. If it was the case, there could be a potential overlap between the two classes of instances in the input data. This would probably result in a decline of the cluster performance due to the ambiguous description of the data set. However, this can be overcome by carefully selecting an impact bound of h, which will trade off the performance of final algorithm as well.

Using the impact function $g(x_j)$, the regularization weights c_j can be computed

$$c_j = \begin{cases} c_0 \cdot \frac{1-g(x_j)}{1-\exp(-2)} + \frac{1}{n} \cdot \frac{g(x_j)-\exp(-2)}{1-\exp(-2)} & \text{if} \quad g(x_j) > 0 \\ c_0 \cdot \frac{1-|g(x_j)|}{1-\exp(-2)} + \frac{|g(x_j)|-\exp(-2)}{1-\exp(-2)} & \text{if} \quad g(x_j) < 0 \end{cases} \tag{6.16}$$

where c_0 is the initial value of c_j and n is the sample size.

On the one hand, when the value of the impact function $g(x_i)$ is positive, the sample x_i is more likely recognized as an anomaly affected mainly by the given anomaly set $\mathcal{X}^+$. For example, when x_i infinitely approaches a given anomaly, we have $g(x_i) = 1$ and $c_i = 1/n$, which implies the probability of x_i being an anomaly: $P_a(x_i) = 1$. Similarly when x_i lies on or out of the bound of $2h$, we have $g(x_i) = \exp(-2)$ and $c_i = c_0$ which is not affected at all as the initial value. Consequently, the weighted penalties on the affected neighborhood of the given anomalies are reconfigured between $1/n$ and c_0.

On the other hand, when the point x_i is mainly affected by the given normal set $\mathcal{X}^-$, the value of $g(x_i)$ is negative, and it is more confident to be recognized as a normal instance. In extreme case, the x_i approximates a given normal instance infinitely, the probability of being an anomaly is zero, we can then set the penalty c_i as 1. Again, when a point lies out of the bound $2h$, it stays unaffected with the initial configuration of c_0. Consequently, the neighborhood of the given normal samples is penalized more heavily between c_0 and 1. Together with the constraints on the labeled samples, the Wolfe dual form becomes:

$$\begin{aligned} \mathcal{W} = & \sum_j k(x_j,x_j)\beta_j - \sum_{i,j} \beta_j \beta_j k(x_i, x_j) \\ \text{s.t.} \quad & 0 \leq \beta_j \leq c_j \\ & \beta_{\mathcal{X}^+} = c_{\mathcal{X}^+} \\ & 0 \leq \beta_{\mathcal{X}^-} < c_{\mathcal{X}^-} \end{aligned} \tag{6.17}$$

The weighted regularization term and the additional constraints do not change the convexity. Solving this dual problem by maximizing $\mathcal{W}$, we obtain the robust SVC boundary

providing a more reliable separation of anomalies from normal instances.

Complexity Analysis

Note that the iSVC algorithm only requires an additional computation of the regularization weights $\{c_j\}$. In the computation of c_j, the similarity measurements are represented by the kernel matrix K. Hence, we only need to compute the impact function for each point, the additional complexity is then $\mathcal{O}(n)$. Nevertheless, the problem can still be solved in polynomial time without noticeable computational burden.

6.1.5 Experiments

In this section, we evaluated the iSVC algorithm both on synthetic data and real-world data. The required parameters for iSVC involve the kernel bandwidth h, an initial penalty constant c_0 and user given positive and negative label sets that correspond to anomalies and normal instances respectively. The bandwidth h controls the smoothness of the cluster boundary and also the number of the support vectors. For simplicity, we assume an optimal h without explicitly evaluating it in experiments. Note that this can be done by the cross-validation. For the initial value of c_0, a good estimate could be derived from the proportion of the outliers in input data set, namely, $c_0 = 1/n_{\mathcal{B}}$, where n_{bsv} is the number of outliers in data set. In the end, only a small number of labels are supposed to be given by users and we show that the involvement of the supervised information indeed improves the performance of support vector clustering significantly on anomaly detection.

On Synthetic Data

To illustrate the mechanism and capability of iSVC algorithm, experiments were conducted firstly on a 2-d synthetic data set. It contains two normal classes of compacted Gaussians, each of which has 250 samples. Another set of 150 anomaly samples forms a sparse ring encircling the normal instances. The experiments were performed in three distinct configurations of given labels , that is, 1) only anomalies were given by users. 2) only normalities were given by users. 3) both anomalies and normalities were indicated. All the labeled data were randomly sampled from the input data set.

In Fig. 6.3, the iSVC clustering results given different label sets are shown in a 3x3 grid. Each row has the same bandwidth h and the initial c_0, and the first column of the results gives the cluster boundaries of original SVC algorithm on the synthetic data. Note that the bandwidth h is configured with a global value over all the experiment sessions. In Fig. 6.3a, a moderate value of C only discloses a minority of the actual anomalies, while a handful of normal instances are also recognized as anomalies due to their low densities. Clearly, the SVC algorithm fails when the densities of data samples are not explicitly separable between normalities and anomalies. In the middle of the

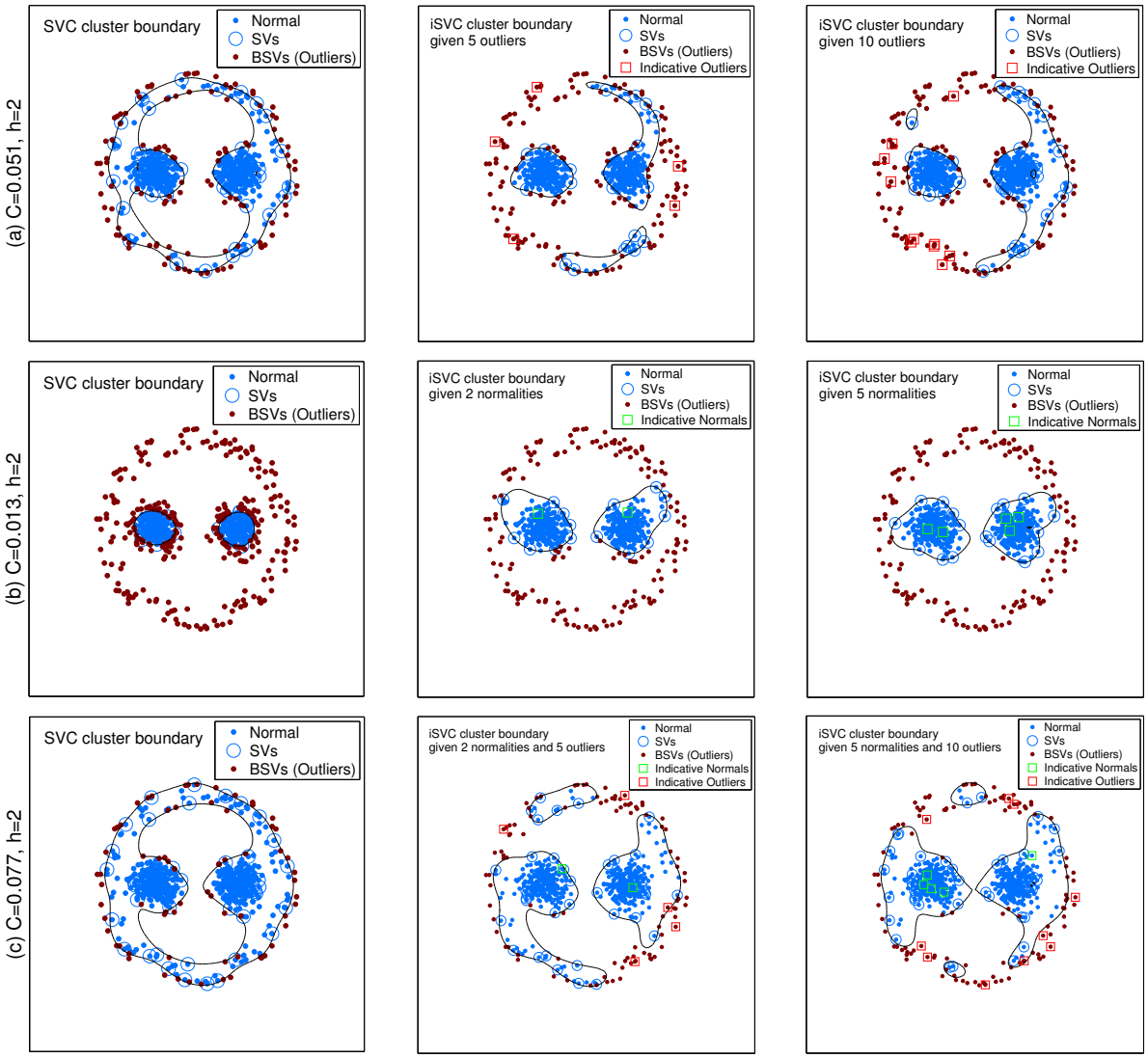


Figure 6.3: Experiment on a 2-d synthetic dataset. Given different sets of supervised labels, the cluster boundaries of SVC and iSVC changed accordingly and also the predicted anomalies. Note the blue points represent normal samples, and red ones are the predicted outliers.

first row, 5 given anomalies are evenly distributed along the ring. We observe that more anomalies are discovered under the supervised information. On the rightmost, more anomalies up to 10 are given, however, part of which are redundantly located on the left half ring. Consequently, the anomalies on the right half ring are completely unaffected. In the Fig. 6.3b, a smaller c_0 produces an excessive discovery of anomalies with a significant false positive. Since the normal instances are more compacted in

the center, supervised information can be passed on its neighborhood more efficiently. On the rightmost of the second row, only 5 given normal instances produce a perfect separation of the anomalies from the normal samples. In Fig. 6.3c, it does not present a promising cluster boundary for anomalies, however, the normal instances are mostly detected under a small number of given normal labels. Additional anomaly labels are required to achieve a perfect separation.

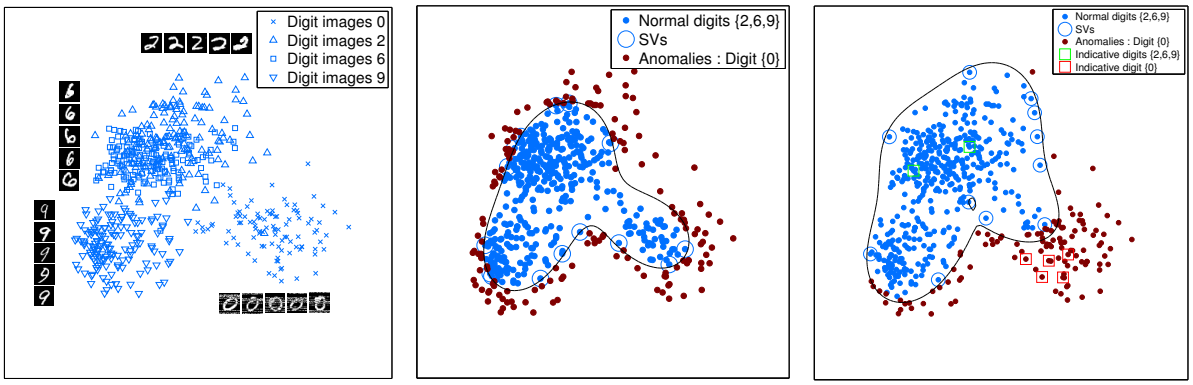
In summary, we observe that the iSVC outperforms the SVC algorithm by introducing additional supervised information of user given labels. Under the compactness of normal instances, the requirement on the number of indicative normalities is usually less rigorous than indicative anomalies. However, when the anomalies form certain clusters themselves instead of being sparsely distributed, iSVC can easily generate a promising result by only given a small number of labels. Note that it is also crucial to introduce a certain sampling mechanism to actively select labels such that iSVC leverages the supervised information effectively to avoid redundancy. We leave that as a future research work.

On Real-world Data

The evaluation of the iSVC algorithm was then performed on real-world data sets. Again, we assume an optimal kernel bandwidth h for each experiment session without explicitly estimating it. The experiments show that iSVC improves the original SVC algorithm and outperforms other semi-supervised binary classifier, and also presents its potential to related practitioners who intent to secure their anomaly detection process against adversarial users.

MNIST Digit Images. We firstly selected four digit sets from the MNIST digit images data set [72]: $\{0, 2, 6, 9\}$. To fabricate a similar anomaly-aware scenario, 100 images of digit $\{0\}$ were manually tampered with a couple of horizontal noisy lines. For digits $\{2, 6, 9\}$, we sampled 150 images out of each class as normal instances. Note that each digit image is a 8-bit grey image with a dimension of 28x28. For demonstration, we applied *PCA* firstly on the input data to visualize its two main principal components. In the reduced dimension, the tampered digit images $\{0\}$ are relatively distant from other digits images. In Fig. 6.4b, SVC algorithm generated several false positives and false negatives as well. Given 2 normal images and 5 anomalies in Fig. 6.4c, iSVC produced a better estimate of the cluster boundary setting the anomaly digits $\{0\}$ apart from other legitimate digits.

A quantitative evaluation of the iSVC algorithm was then performed on the input MNIST data set without PCA. The evaluation was repeated 10 times to reduce the effect of the randomness on choosing the sample labels. In Fig. 6.5, it is observed that a varied initial regularization constant C starting from an extremely small value close to 0 has a significant influence on the performance of SVC, while iSVC is much more reliable and relatively more independent of the regularization parameter by giving the indicative labels. The accuracy of iSVC is more stable and almost all over 90% with respect to various C (see top row). When we changed the proportion of the outliers in the input



(a) Original distribution (b) SVC cluster boundary (c) iSVC cluster boundary

Figure 6.4: MNIST digit images of $\{0, 2, 6, 9\}$ are demonstrated on its two main principals after applying PCA. Both SVC and iSVC are conducted on the reduced data set with the parameters $h = 2.5$ and $C = 0.012$.

data with C fixed as 0.03, the iSVC performed much better than the SVC as well. Note that, at such a small value of C, the recall is always 100% due to the excessive detection of anomalies, and the precision is poor because many normal instances were now classified as anomalies. Nevertheless, with only 5% randomly selected labels, the performance of iSVC on anomaly detection reveals its potential for practitioners, especially with a careful selection of indicative labels (see bottom row).

The WDBC Data. Finally we conducted an experiment on the *Wisconsin Diagnostic Breast Cancer (WDBC)* data set from UCI repository [5] to illustrate the empirical advantages of iSVC. The dataset contains in total 569 samples, out of which 212 are malignant and 357 are benign. For comparison, we employed the semi-supervised fast linear SVM solver [113] as a baseline method, denoted as *S3VM*. Note that *S3VM* requires an additional parameter indicating the ratio of outliers in the unlabeled dataset, and this is normally not available in practice. To approximate a value of the ratio, we estimated it from the labeled data instead of the unlabeled data. Suppose the labeled data are actively sampled according to a certain distribution, this reflects the underlying probability of positive samples. The empirical results are shown in Table. 6.1.

We started with a very small value of regularization parameter C which should identify excessive anomalies by iSVC. Given some indicative labels, iSVC outperforms the original SVC algorithm and behaves much more stable than *S3VM*. Especially when imbalanced labels are given, e.g., only giving 5% positive or 5% negative labels will completely exterminate the function of *S3VM*.

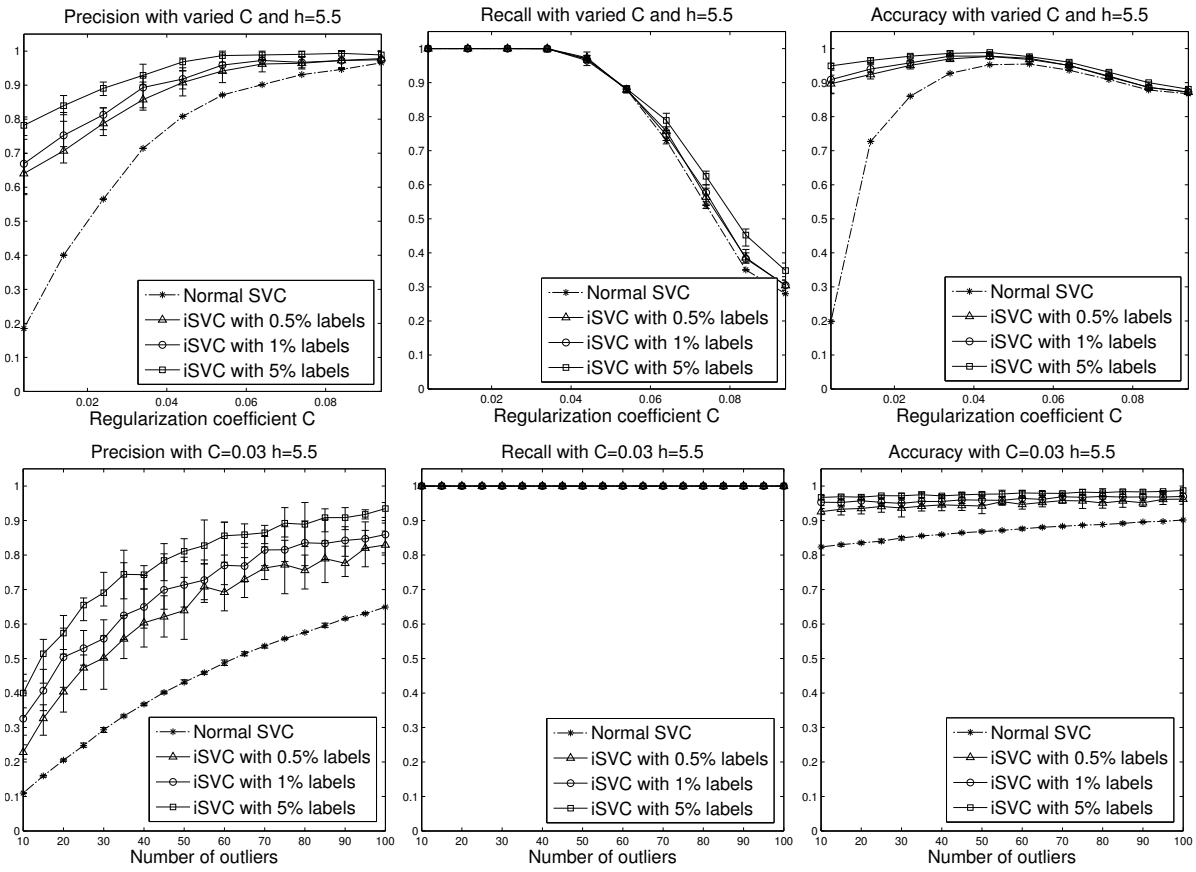


Figure 6.5: Quantitative evaluation for iSVC and SVC on the MNIST digit images data set without *PCA*, with respect to different regularization parameter C and different ratios of anomalies in the input data. For iSVC, different level $\{0.5\%, 1\%, 5\%\}$ of indicative labels are also given.

6.1.6 Summary

The iSVC algorithm is proposed to circumvent the inherent deficiencies in SVC that is not capable of integrating additional supervised information. By indicating a small number of sample labels, the iSVC exhibits its prominent performance on anomaly detection, more generally, on semi-supervised binary classification problem. The clustering model is guided to produce a more robust decision boundary under the supervised information, especially when there exists adversarial users who carefully craft poisoning or evading examples to form its own cluster. In previous introduction of various adversarial attacks, we have seen that potential attackers are capable of reverse engineering the classifier and leading it to misclassify certain samples, in order to achieve their own gain. In future, we will extend our work along three dimensions. Firstly we will extend the model to automatically estimate the optimal bandwidth and actively select informative

	WDBC dataset with $\mathbf{h = 0.9, C = 0.001}$			
	Accuracy	F_1-measure	FPR	FNR
SVC	63.4%	50.5%	28.6%	50%
	5% *positive* and 0% *negative* labels			
iSVC	**90.5**%	**87.0**%	**6.4**%	14.6%
S3VM	39.2%	55.0%	96.9%	**0**%
	0% *positive* and 5% *negative* labels			
iSVC	**89.8**%	**87.7**%	14.8%	2.4%
S3VM	64.9%	10.7%	**0**%	94.3%
	5% *positive* and 5% *negative* labels			
iSVC	**92.3**%	**89.8**%	**7.0**%	8.9%
S3VM	88.4%	86.3%	17.4%	**1.9**%
	10% *positive* and 10% *negative* labels			
iSVC	**93.9**%	**91.9**%	**6.4**%	5.6%
S3VM	92.4%	90.6%	10.4%	**2.8**%

Table 6.1: Empirical results on the WDBC dataset given different proportions of indicative labels of both positive and negative samples, comparing to the baseline method *S3VM*.

data samples for labeling. Second, the iSVC based anomaly detection model should integrate adversarial attacks and dynamically adapt to the adversarial noise. Finally we expect our model to be able not only to discover the adversarial users, but also to causally reason their malicious behaviors.

6.2 Objective Learning from Multiple Sources

Malicious impact incurred by non-gaussian or adversarial noises can be mitigated by involving a certain level of supervised information. Given a set of labels with high confidence, it is shown in Sect. 6.1 that the anomaly detection can be secured against outliers even when they formulate an abnormally behaving cluster. To counter adversarial attacks, additional information is required to guide the learners through the entanglement of the attackers. We present another mechanism of protecting against the malicious users from tampering legitimate functions of learning systems. More precise, we leverage responses from multiple sources for inference of ground truth and the expertise of those sources, some of which might possibly be the adversaries.

Nowadays machine learning techniques are increasingly employed in multiple applications of social web services. A common property of these services is that the information is retrieved and shared among a large number of Internet users. For instance, the Wikipedia enables any user to edit and evaluate the articles online and possibly requests feedbacks from them, *i.e.*, "trust-worthy", "objective", "complete" and "well-written". The quality of service is thus not guaranteed since it is publicly available for any user, and adversaries with certain incentive of those services are of potential threats. Besides, other platforms could leverage the wisdom of crowds to accomplish a common task or project. For example, the Amazon Mechanical Turk[1] offers service to those requesters to publish and maintain a set of assignments, which can be tackled by a group of hired users (not necessarily legitimate users). Galaxy Zoo[2] is another web service to allow users to annotate astronomical images. The way of performing a task collaboratively is called crowdsourcing that collects pieces of response or information from multiple, possible a large number of, users to accomplish a more complex function, *e.g.*, automatic recognition of galaxies from astronomical images. In machine learning and data mining research, this associates with challenges of *learning from multiple sources* [33], especially the security of inferring reliable predictions under potential adversarial users' actions is of central interest. In spite of these web service, learning from multiple sources finds its wide applications also in other domains. For instance, the sensor networks have been applied for reliable monitoring of remote and hostile environment. In [125], researchers used a 16-nodes sensor network mounted on a tree to monitor elevation from multiple weather fronts. However, the challenges arise if the collected sensor data is noisy or sometimes adversarial, *e.g.*, network outage, aging sensors, it might be counterproductive by simply averaging out as suggested by a common routine. Therefore, a mechanism of learning the environmental parameters and predicting the readings involving the additional information is required to perform a robust learning from multiple sources.

A number of studies have involved multiple sources, but mainly focus on classification problem, for instances, in previous work [37, 60, 117] researchers estimated error rates of various observers. In the sequel, we may refer sources interchangeably as labelers, experts, annotators or observers, they represent the same subject unless stated otherwise. In [33], the most informative samples are selected from multiple observers for model training, thus the model can be learned directly from this multiple source dataset, instead of separating the inference of ground truth and learning the classifier in two stages. Later, probabilistic models dealing with this learning problem are presented in [102, 103] with an assumption that expertise of annotators is independent of the given observations. This assumption is then relaxed in [134, 145] and extended to the active learning problem in [144]. In this work, we revisit the probabilistic graphical model presented in [141] both in theory and empirical experiments, and then present a real application to it, which provides an objective photo ranking function [142]. It is

[1] `https://www.mturk.com`
[2] `http://www.galaxyzoo.org/`

shown that the probabilistic graphical provides a unified regression function to correctly infer the ground truth and the expertise of multiple observers, using the less-parametric *Gaussian Process* framework. Given a good estimate of the expertise of observers, the adversarial impact can be mitigated with respect to the predicted ground truth.

6.2.1 Probabilistic Framework of Multiple Observers

Given N instances $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ where $\mathcal{X} \subseteq \mathbb{R}^L$ is the instance space, also the responses from M observers are denoted as $\mathcal{Y} = \{\boldsymbol{y}_{n,m}\}_{n=1,m=1}^{N,M}$ and $\mathcal{Y} \subseteq \mathbb{R}^D$. L and D are the dimensions of input instances and responses accordingly. Besides, a latent variable $\mathcal{Z} = \{\boldsymbol{z}_n\}_{n=1}^{N}$ denotes the unknown D-dimensional ground truth for the inputs $\mathcal{X}$. For compactness, the *bold*-typed $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ represent vectors in their instance space $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively.

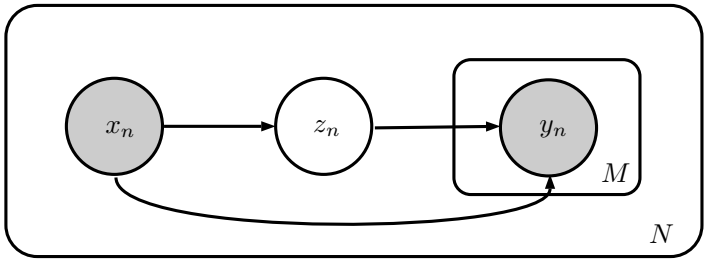


Figure 6.6: Graphical model of the learning problem from multiple observers. Instances $\mathcal{X}$ are associated with the unknown ground truth $\mathcal{Z}$, and the responses $\mathcal{Y}$ from M different observers are conditioned on both input instances and the latent ground truth. Note that only shaded variables are observed.

To formulate the learning problem from M observers, potentially with adversaries, a probabilistic graphical model is proposed in Fig. 6.6. The probabilistic graphical model proposes two problems to be solved, namely, it estimates the ground truth for all the input instances, and also the model captures the expertises functions for each observer, which might be a potential adversary. The deviation between the response $\boldsymbol{y}_m$ and $\boldsymbol{z}_m$ implies how much the m-th observer can misbehave from ordinary annotators.

The joint probability of the model can then be formulated as,

$$\begin{aligned} p(\mathcal{Y}, \mathcal{Z}, \mathcal{X}) =& p(\mathcal{Y} \mid \mathcal{Z}, \mathcal{X}) p(\mathcal{Z} \mid \mathcal{X}) p(\mathcal{X}) \\ \propto& \prod_{n=1}^{N} \prod_{m=1}^{M} \prod_{d=1}^{D} p(y_{n,m,d} \mid \boldsymbol{x}_n, z_{n,d}) p(z_{n,d} \mid \boldsymbol{x}_n), \end{aligned} \tag{6.18}$$

Note that $p(\boldsymbol{x}_n)$ is dropped since the $\mathcal{X}$ is treated as an empirical collection of instances, which hold constant while estimating the model. Besides, two assumptions are given in Eq. (6.18), firstly seen from the first product over D, each dimension of the ground

truth is independent but not identically distributed. Second, all observers are also independent, given by the product over M. To tackle this problem, previous work [134, 145, 102, 144, 103] adopted various parametric methods to estimate the conditional distribution accordingly. Here in [141], a nonparametric Gaussian process is proposed to capture the model. Moreover, the generative process of responses $\mathcal{Y}$ can be written in following two noise models,

$$z_{n,d} = f_d(\boldsymbol{x}_n) + \epsilon_n, \tag{6.19}$$

$$y_{n,m,d} = g_{m,d}(\boldsymbol{x}_n, z_{n,d}) + \xi_{m,d}, \tag{6.20}$$

where ϵ and ξ are both *i.i.d* Gaussian noise respectively. In the context of *Gaussian Process*, the goal is to find proper functionals for $\{f_d\}$ and $\{g_{m,d}\}$ for each independent output dimension. They are described as regression model and observer model respectively. In the adversarial learning setting, the regression model f_d generates stationary dataset from a certain ground truth $\mathcal{Z}$, while the observer model $g_{m,d}$ is supposed to encode the malicious behavior of the m-th adversary to produce effective adversarial samples $\mathcal{A} = \left\{\boldsymbol{x}_n, \boldsymbol{y}_{n,m}\right\}_{g_m}$, which break the stationary assumption.

Regression Model

The regression model f_d represents the mapping of an input instance to its corresponding ground truth. The function is defined with a Gaussian process governed by a covariance matrix, which is some nonlinear kernel. More precisely, the conditional distribution of $p(\mathcal{Z} \,|\, \mathcal{X})$ is given by,

$$p(\mathcal{Z} \,|\, \mathcal{X}) = \prod_{d=1}^{D} \mathcal{N}\left(\boldsymbol{z}_{:,d} \,|\, \mathbf{0}, \mathbf{K}_d\right), \tag{6.21}$$

where $\mathbf{K}_d$ is a $N \times N$ kernel depends on input $\mathcal{X}$. For each output dimension, there is a kernel function $k_d(x_i, x_j) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{0+}$ defined to produce the covariance matrix $\mathbf{K}_d$.

Observer Model

The observer model $g_{m,d}$ defines a mapping $g : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$ according to the graphical model in Fig. 6.6. A simplified linear mapping can be though of neglecting the dependence between input $\mathcal{X}$ and $\mathcal{Y}$, that is, the response is purely determined by a linear function of the unknown ground truth $\mathcal{Z}$. Therefore, a linear observer model is formulated to be,

$$p(\mathcal{Y} \,|\, \mathcal{Z}, \mathcal{X}) = p(\mathcal{Y} \,|\, \mathcal{Z}) = \prod_{m=1}^{M} \prod_{d=1}^{D} \mathcal{N}\left(\boldsymbol{y}_{:,m,d} \,\middle|\, w_{m,d}\boldsymbol{z}_{:,d} + \mu_{m,d}\mathbf{1}, \sigma_{m,d}^2\mathbf{I}\right), \tag{6.22}$$

where each observer is captured by $3 \times D$ parameters, namely, the $w_{m,d}, \mu_{m,d}, \sigma_{m,d} \in \mathbb{R}$. Together with Eq. (6.21) substituted back to Eq. (6.18), the optimal parameters can be estimated by maximizing the likelihood using optimization techniques such as L-BFGS quasi-Newton methods, since it does not have a close-form solution.

To estimate the ground truth, the posterior of $\tilde{\boldsymbol{z}}_{:,d}$ can be computed by $p(\boldsymbol{z}_{:,d} \,|\, \boldsymbol{y}_{:,:,d}, \boldsymbol{x})$, which is again a Gaussian. Moreover, prediction on new instance x_* can be derived from the joint distribution $[\tilde{\boldsymbol{z}}_{:,d}, z_{*,d}]^T$, that follows the Gaussian distribution,

$$\begin{bmatrix} \widetilde{\boldsymbol{z}}_{:,d} \\ z_{*,d} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_d & \boldsymbol{k}_*^\top \\ \boldsymbol{k}_* & k_d(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right), \tag{6.23}$$

where $\boldsymbol{k}_* := [k_d(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, k_d(\boldsymbol{x}_*, \boldsymbol{x}_N)]$. The best estimate of $z_{*,d}$ can be derived from the conditional distribution $p(z_{*,d} \,|\, \mathcal{X}, \widetilde{\boldsymbol{z}}_{:,d}, \boldsymbol{x}_*)$ by applying Bayes theorem.

However in many cases, linear observer model is too optimistic and can not describe complex dependencies, therefore, another Gaussian process is proposed replacing the linear mapping, that is, $g : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$ and the conditional distribution turns,

$$p(\mathcal{Y} \,|\, \mathcal{Z}, \mathcal{X}) = \prod_{m=1}^{M} \prod_{d=1}^{D} \mathcal{N}\left(\boldsymbol{y}_{:,m,d} \,|\, \mathbf{0}, \mathbf{S}_{m,d}\right), \tag{6.24}$$

where $\mathbf{S}_{m,d}$ is the covariance represented by a nonlinear kernel matrix. For each output dimension d and each observer m, there is another kernel function $s_{m,d}(\{\boldsymbol{z}_i, \boldsymbol{x}_i\}, \{\boldsymbol{z}_j, \boldsymbol{x}_j\})$ defined over $\{\mathcal{Z}, \mathcal{X}\}$ to capture the covariance matrix $\mathbf{S}_{m,d}$. The parameter estimation is similar as described for the linear observer model. Moreover in practice, a linear observer model is firstly estimated as an initialization for the nonlinear observer, to prevent the regression model f_d from getting too trivial while the observer model $g_{m,d}$ getting too complicate.

6.2.2 Synthetic Experiment

Synthetic experiment is conducted to demonstrate the performance of the proposed model. Given a nonlinear function $f(x) := sin(6x)sin(\frac{x}{2})$, an one-dimensional synthetic dataset is randomly sampled from the function. The training set $\mathcal{X}$ consists of 30 points uniformly distributed in $[0, 2\pi]$, and test instances are obtained from bins of $[0, 2\pi]$ with equal size 0.05, totally 126 points. Given the random samples from f, four observers are simulated with monotonic functions $g_m(f(\cdot))$ plus a Gaussian noise. For this monotonic setting, a perfect observer produces exactly the same outputs as ground truth, that is, $g_m(f(x)) \equiv f(x)$. The synthetic ground truth and observers are illustrated in Fig. 6.7. For comparison, two baselines based on Support Vector Regression and Gaussian Process Regression are employed to take average over all the observers, and are noted as `SVR-AVG` and `GPR-AVG`. Besides, the linear observer model (`LOB`) is also considered as a baseline method. Finally, the performance is measured by the mean absolute normalized error (MANE) and the Pearson correlation coefficient (PCC). For

the MANE, both the observed value and predicted value are normalized in $[0, 1]$ and then the mean absolute error is measured. And the PCC is mainly employed to indicate the correlation between the ground truth and the observers. When the PCC is close to 1, it means that the observer has a high positive correlation with the ground truth.

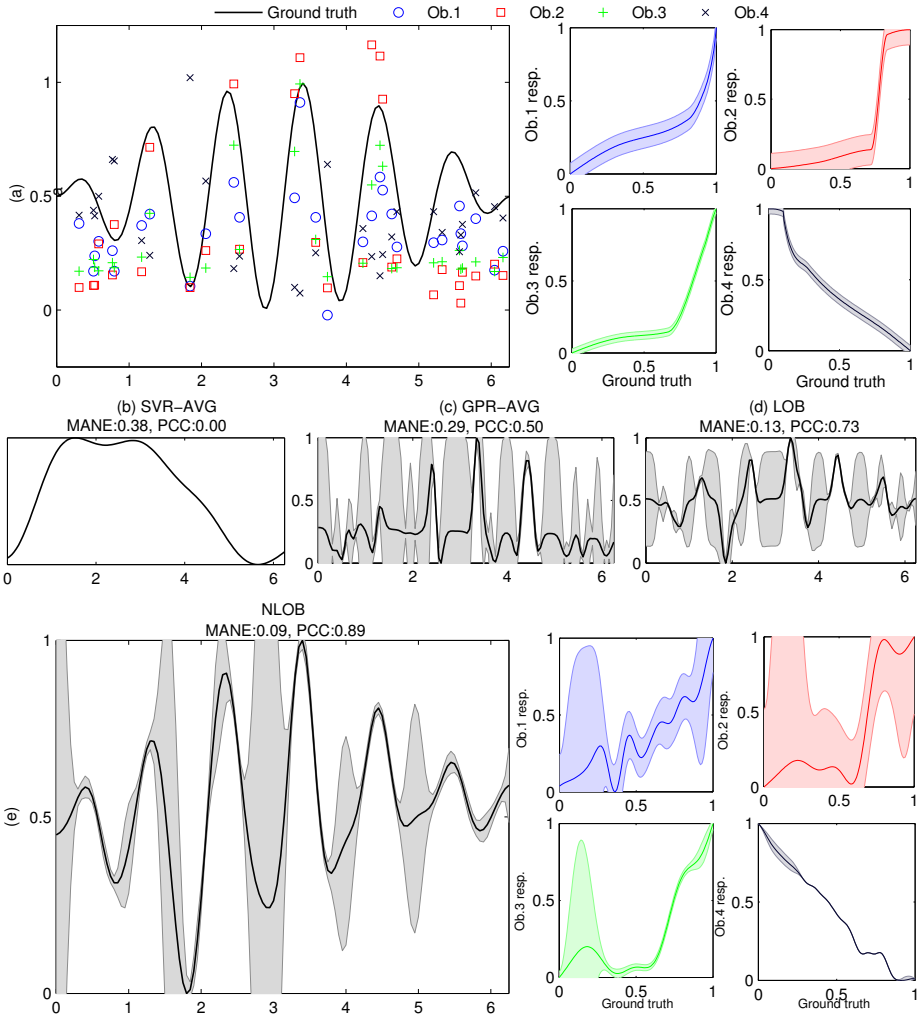


Figure 6.7: (a) Synthetic data and ground truth, and also Responses from observers are marked in different colors. On the right, four observers are simulated by monotonic functions $\{g_m\}$, and shaded area illustrates the pointwise variance. Note that the 4-th adversarial observer misbehaves to produce counterproductive observations in opposite to the ground truth. (b, c, d) Estimated ground truth on the test set using baselines `SVR-AVG`, `GPR-AVG` and `LOB`. (e) Estimated ground truth and observer models by nonlinear observer model `NLOB`.

We see from Fig. 6.7 that the nonlinear observer model has successfully recovered the ground truth function and also the observer functions. Noticeably, no extra prior

knowledge is provided except for the multiple source dataset. In addition, both the `LOB` and `NLOB` models outperform other baseline regression methods, which follow the trivial averaging strategy. Since the learning problem deals with a highly nonlinear compound sinusoid function, the `NLOB` model performs the best out of the others. Especially, the response function of the 4-th observer counters the predicted ground truth, which implies that the 4-th observer is a potential malicious user who attempts to give negative responses. More importantly, the adversarial effect is mitigated while estimating the regression function for the unknown ground truth, however other baseline methods are heavily compromised by the 4-th observer.

6.2.3 OPARS: Objective Photo Aesthetics Ranking System

We built a real application leveraging the predictive power of the probabilistic model defined in [141], namely, an Objective Photo Aesthetics Ranking System (*OPARS*) [142]. As the perception of beauty is subjective across individuals, evaluating the objective aesthetic value of an image is a highly biased task in image retrieval system. Unlike current online photo sharing services that take the average rating as the aesthetic score, the *OPARS* integrates various ratings from multiple users, and jointly estimate the objective scores (latent ground truth) and the expertise of different users. In the front-end, users are requested to rate images selected by an active learning process. And the multiple observer probabilistic model introduced above is employed in the back-end to integrate these ratings for predicting the aesthetic value of images. Moreover as suggested before, the *OPARS* is reliable and resilient to malicious users, who try to tamper the image ranking by sending adversarial ratings.

System Architecture: The image ranking system follows a browser-server architecture. The front-end establishes a user-friendly web interface facilitating the process of image browsing and rating. The back-end contains a set of functional components including the database access and storage procedure, user authentication, score prediction and so on. The design of the architecture is depicted in Fig. 6.8.

User Authentication: A password-based authentication protocol is adopted for convenient access control, see Fig. 6.9 (a). In addition, users will be prompted to provide individual demographic information for research purpose, which serves as prior knowledge in our regression model.

Continuous-valued Rating: Authenticated users are able to browse the image gallery and response with their ratings against the images. Instead of conventional discrete image ratings, *OPARS*, enables a continuous rating metric (ranging from 0 to 5). Additionally, it also records users' response time in milliseconds in the background.

Active Image Retrieval: One key contribution of *OPARS* is that the images presented to users are not selected randomly but in an active learning process. Recently, active learning [144] has gained significant attention in machine learning community. By introducing active learning in image rating system, the underlying predicting model provides more efficient and accurate results by only rating a small portion of images.

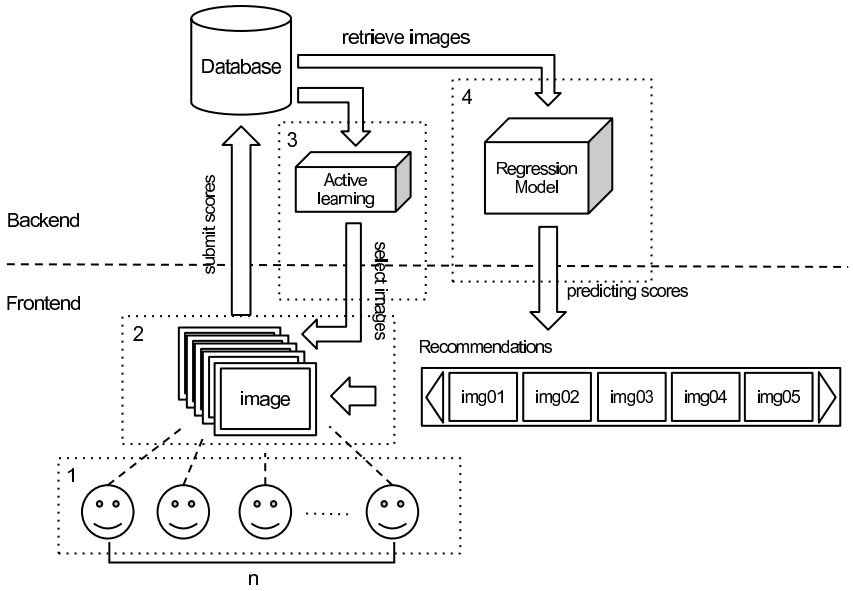


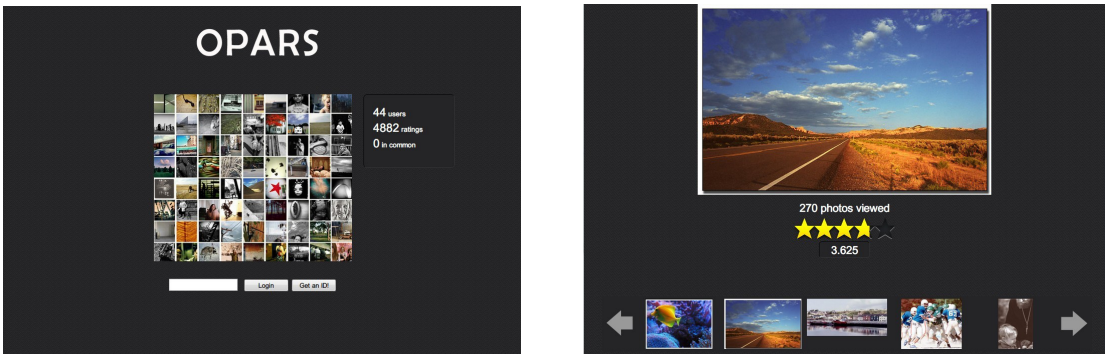
Figure 6.8: System architecture of the *OPARS*. It follows the browser-server architecture. Main system modules are framed in dotted lines. They are 1) User authentication module 2) Continuous-valued rating module 3) Active image retrieval module 4) Image recommendation module.

In *OPARS*, we adopt Gaussian process for active learning, where the unrated images with highest variances are selected for the next round evaluation. For the computational reasons, we call the active learning routine on every 10 rated images.

Image Recommendation: Note that conventional image rating systems take an average of ratings of an image as its objective score. In this way, individual expertise is ignored for evaluating the aesthetics of an image objectively. In our system, the underlying multiple observer regression model is capable of integrating users' expertise as prior knowledge and can successfully recover an objective score of an image, even if some users are adversarial or just biased. Note that images with highest predicted objective scores are recommended by the system. In total, we have received 42 registered users and 4839 ratings on images. Among these contributors, over 28 users have rated over 100 images, and 10 of them even reach a line of 200 images.

6.2.4 Summary

To secure a data-driven system against potential adversarial samples, we revisited the multi-observer probabilistic graphical model in [141], and moreover we built an application based on the model to provide an objective image ranking service. By introducing a latent ground truth variable, the observers' expertise together with the hidden ground truth can be estimated from the joint distribution in Eq. (6.18). An one-dimensional



(a) Home view (b) Main view

Figure 6.9: (a) The home view of the *OPARS* . The right panel displays fundamental statistics of current system. (b) The main view where user can browse image gallery by pressing arrow keys. Users' response time and ratings are recorded when they click on the rating bar. At the bottom, five images with highest objective aesthetic scores are recommended by the system.

synthetic experiment shows that the proposed graphical model outperforms other taking-average regression models, and gives better prediction of the ground truth even if there exists a potential adversary. This suggests a valuable probabilistic treatment to enhance the robustness of learning systems.

6.3 Reasoning the Attacks: A Copula Bayesian Network Approach

In Sect. 6.1 and Sect. 6.2, two methods are introduced to enhance the security and robustness of learning algorithms. For the Support Vector Clustering algorithm, a weighted regularization term is proposed to reduce the effect of adversarial samples, while in the multiple source scenario, a latent variable and graphical model is introduced to capture the real dependencies of ground truth and observers. Both treatments have introduced a certain level of prior knowledge into the learning algorithms, such that the adversarial impact can be mitigated without hurting the legitimate function of the systems. However, in practice, it is not enough to identify and mitigate the attacks, but also requires to reason the attacks. For instance, an online credit banking system might block the transaction of a credit card while it discovers a potential fraud, although it usually does not send a report reasoning the denial-of-service and why the fraud happens. To this end, we close our discussion of secure learning with a particular work on reasoning the attack causally, using a Copula Bayesian Networks approach.

Copula Bayesian networks (CBNs) are powerful models to represent multivariate continuous distributions, while coming along with a simplified parameter estimation strat-

egy. We propose a method for structure learning of CBNs, called *pi-CBN*, which is based on the partial inverse correlation matrix. We show that the method gives good estimations of CBNs and is able to outperform related approaches in the field of CBN-learning on several synthetic and real-world data sets. The results indicate that the *pi-CBN* is able to recover causal relations in less time than existing approaches, and moreover, results in more accurate models in terms of structural Hamming distance and log-probability.

6.3.1 Causality Discovery and Related Work

The identification of causal relationships remains an important topic in the analysis of the adversarial learning, in particular, the attacks have to be reasoned with causality relationship among features and responses. Bayesian Networks [57] (BNs) are powerful tools to uncover such relationships by offering a graphical structure along with conditional probability distributions/tables that reflect the interdependencies of included variables. Standard Bayesian network learning methods always include two main challenges: parameter learning and structure learning. Although a lot of effort has been put into this area, many problems remain, such as the computational effort required by many approaches, inaccuracies in structure inferences and non-trivial parameter estimation problems. Copula Bayesian networks (CBNs), recently proposed by Elidan ([41]), incorporate ideas from Copula theory to offer a simplified parameter estimation strategy. Thus, we develop a structure learning approach for CBNs. In a first step, we use Copula functions to reduce the prior parameters to be estimated by the segregation of univariate marginals from the multivariate distribution. Subsequently, we base the structure inference on the partial inverse correlation matrix (PICM). In experiments, we show that this strategy is faster than competing approaches and also results in Copula Bayesian networks (CBNs) of accurate node dependencies.

The main advantage of CBNs is the high flexibility of representing multivariate distributions by choosing various univariate marginals, meanwhile leveraging the graphical representation of BNs. In the multi-observer graphical model, the adversaries can be identified by recovering the dependencies between outputs and the latent ground truth, but the multivariate dependencies among the input features can never be learned. Prior work [41] has studied the basic problem of parameter learning for CBNs. It has shown that CBNs are also elegant models dealing with a large number of missing observations [42], where a lower bound for a log-likelihood function is proposed for efficient inference. However, the structure learning problem in CBNs has not ever been very well studied, which is considered as the most challenging problem of learning a Bayesian network. In terms of structure learning in BNs, two most commonly used algorithms are: First, the PC algorithm which belongs to the category of constraint based methods [118]. And second, scoring function based heuristic methods, especially based on the Bayesian information criterion (BIC) [57]. The PC algorithm starts from a completely connected graph, edges are removed if the corresponding independencies are given, which usually

requires a large amount of conditional independence tests. The BIC score based heuristic method generates the network structure in a greedy strategy, so the global minimum is not guaranteed. Other structure learning methods, such as the Sparse Candidate (SC) algorithm [47], best-first search [68], all suffer from either structural inaccuracy or heavy computational effort. To capture the causal relationship more efficiently using CBN, we suggest the use of the partial inverse correlation matrix, which largely reduces the amount of conditional independence tests so that the learning is extremely fast. Moreover, the estimated parameters of the Copula function (Gaussian Copula) can be used further as the input for the structure learning. Furthermore, even with a large amount of missing values in training data, it is still enough for a precise inference of structure. This differs impressively from other structure learning algorithms where significant statistics from data is required.

6.3.2 Copula Bayesian Network

We introduce the framework and notation of Copula Bayesian Network to lay the foundation of the proposed *pi-CBN* method. The Copula function is defined as a multivariate probability distribution within the domain of a N-dimensional unit hypercube. It can be selected as the prior for parameter estimation in Bayesian networks. A framework for Copula Bayesian networks [41] was proposed by combining Copula theory and BNs.

A Copula ([114], [89]) is a function C linking univariate marginals to generate a multivariate distribution. In domain $\mathcal{X} = \{X_1, \ldots, X_N\}$ consisting of an N real-valued random variables. Let $F_{\mathcal{X}}(x) \equiv P(X_1 \leq x_1, \ldots, X_N \leq x_n)$ be a cumulative joint probability distribution over $\mathcal{X}$ where $x = \{x_1, \ldots, x_n\}$ is an assignment of variables X.

Definition 5 (Copulas)**.** *Let $U_1, \ldots, U_N$ be real random variables marginally uniformly distributed over $[0, 1]$. A Copula function C is a cumulative joint probability function: $[0, 1]^N \rightarrow [0, 1]$.*

$$C(u_1, \ldots, u_N) = P(U_1 \leq u_1, \ldots, U_N \leq u_N)$$

From Definition 5, a Copula function C can be viewed as a probability function of points distributed in a N-dimensional unit hypercube. Copulas are important because of Sklar's theorem [114] as described below.

Theorem 1 (Sklar 1959)**.** *Let $F(x_1, \ldots, x_N)$ be any cumulative multivariate distribution over real-valued random variables, then there exists a copula function C such that*

$$F(x_1, \ldots, x_N) = C(F(x_1), \ldots, F(x_N)),$$

where $F(X_i)$ is marginal cumulative density distribution of variable X_i and if each $F(X_i)$ is continuous, then the Copula is unique.

Moreover, if $F(X_1, \ldots, X_N)$ has N-order partial derivatives, the joint density function can be obtained by

$$\begin{aligned} f(x) &= \frac{\partial^N C(F(x_1), \ldots, F(x_N))}{\partial F(x_1) \ldots F(x_N)} \prod_i f(x_i) \\ &= c(F(x_1), \ldots, F(x_N)) \prod_i f(x_i), \end{aligned} \tag{6.25}$$

where $c(F(x_1), \ldots, F(x_N))$ is called copula density function. Using Eq.(6.25), it is very easy to obtain a joint density distribution once the univariate marginals are estimated. This theorem gives the importance of Copulas that, for any multivariate distribution given its marginals, we can find a Copula distribution function to formulate its dependency structure taking univariate marginals as Copulas' arguments.

A simple example is the Gaussion Copula which is widely used because of its simplicity and practical applications. Suppose the correlation matrix Σ, a Gaussian Copula can be constructed simply by inverting Sklar's theorem [114]

$$C(\{F(x_i)\}) = \Phi_\Sigma\left(\phi^{-1}(F(x_1)), \ldots, \phi^{-1}(F(x_N))\right), \tag{6.26}$$

where ϕ is standard normal cumulative distribution, Φ_Σ is standard normal cumulative distribution with correlation matrix Σ. Using Sklar's Theo. (1) and Eq.(6.25), the multivariate Gaussian density distribution can be easily obtained. The correlation matrix Σ is the only parameter to be estimated when univariate marginals are known from data observations.

Moreover, in Bayesian networks, the Markov property allows the network to be split into local terms where a variable is only conditioned on its parents so that the joint probability distribution can be expressed as a product of a collection of local conditional probability distributions. This factorization can also be applied to Copulas to form the conditional Copula density functions.

Remark 1 (Conditional Copula)**.** *Let x denote a variable and $\mathbf{y} = \{y_1, \ldots, y_K\}$ the parents of x, $f(x \mid \mathbf{y})$ the conditional density function and $f(x)$ the marginal density of x. Then there exists a Copula density function $c(F(x), F(y_1), \ldots, F(y_K))$ such that:*

$$f(x|\mathbf{y}) = R_c(F(x), F(y_1), \ldots, F(y_K)),$$

where R_c is the Copula ratio

$$\begin{aligned} &R_c(F(x),F(y_1), \ldots, F(y_K)) \\ &\quad\equiv \frac{c(F(x), F(y_1), \ldots, F(y_K))}{\int c(F(x), F(y_1), \ldots, F(y_K)) f(x) dx} \\ &\quad= \frac{c(F(x), F(y_1), \ldots, F(y_K))}{\frac{\partial^K C(1, F(y_1), \ldots, F(y_K))}{\partial F(y_1) \ldots \partial F(y_K)}} \end{aligned} \tag{6.27}$$

and R_c is defined to be 1 when $\mathbf{y} = \emptyset$.

This implies a parametric form of a conditional probability distribution $f(x \mid \mathbf{y})$ given a Copula density function $c(F(x), F(y_1), \ldots, F(y_K))$ and a marginal density function $f(x)$.

Remark 2 (Decomposition of Copulas)**.** *Given a directed acyclic graph $\mathcal{G}$ encoding conditional independencies over $\mathcal{X}$, and let $f(x) = c\left(F\left(x_1\right), \ldots, F\left(x_N\right)\right) \prod_i f\left(x_i\right)$ be the Copula density function which is strictly positive for every value of $\mathcal{X}$. If $f(x)$ is decomposable according to $\mathcal{G}$, then the Copula density $c\left(F\left(x_1\right), \ldots, F\left(x_N\right)\right)$ can also be decomposed according to $\mathcal{G}$*

$$c\left(F\left(x_1\right), \ldots, F\left(x_N\right)\right) = \prod_i R_{c_i}(F(x_i), \{F(\mathbf{Pa}_i)\}_k),$$

where c_i is a local Copula density function defined over each local term decomposed from $\mathcal{G}$ for each variable x_i, conditioned on its corresponding parents $\{\mathbf{Pa}_i\}_k$. Therefore, the Copula density function can be factorized in a similar way as a Bayesian network. Given the set of univariate marginals for all variables, an elegant framework for Copula Bayesian Networks was proposed.

Definition 6 (Copula Bayesian Network)**.** *A Copula Bayesian Network (CBN) [41] is a triplet $\mathcal{C} = (\mathcal{G}, \Theta_C, \Theta_f)$ encoding the joint density $f_{\mathcal{X}}(x)$. Θ_C is a set for all local Copula densities $c_i(F(x_i), \{F(\mathbf{Pa}_i)\}_k)$ and Θ_f is the set of parameters representing the univariate marginals $f(x_i)$. Then $f_{\mathcal{X}}(x)$ can be parameterized as*

$$f_{\mathcal{X}}(x) = \prod_i R_{c_i}(F(x_i), \{F(\mathbf{Pa}_i)\}_k) f(x_i)$$

Note that Copulas separate the choices of marginal parametric form from the joint probability distribution, thus more general and accurate non-parametric density estimations are allowed to be applied on univariate marginals. By sharing the global univariate marginals, the hypothesis space on parameters has been largely reduced, which enables efficient estimation of BN parameters.

6.3.3 The pi-CBN Algorithm

A correlation matrix neither reflects the causality nor the Markov properties of Bayesian Networks, it only quantifies the pairwise correlation of variables. In order to infer more information among variables, a more specific measure of correlations is needed. The inverse correlation matrix [135] provides more correlative information. It reflects directly the independences in terms of partial covariance which can be used to construct an independence graph satisfying the Markov properties. Given a p-dimensional vector X and a q-dimensional vector Y. The inverse variance matrix $var(X, Y)^{-1}$ is denoted by D written in block form,

$$D = \begin{pmatrix} D_{XX} & D_{XY} \\ D_{YX} & D_{YY} \end{pmatrix}$$

where $D_{YY} = var(Y \mid X)^{-1}$ is the reciprocal of a partial variance of Y given X. Scale the inverse variance matrix D as it has unit off-diagonals, each off-diagonal element in the inverse variance matrix D can be obtained by:

$$D_{ij} = \frac{-D_{ji}}{\sqrt{D_{ii} * D_{jj}}}$$

Moreover, it is the negative of the partial correlation between the two corresponding variables, conditioned on all the rest variables. A particular important property of the scaled inverse correlation matrix is that the off-diagonal block D_{XY} is zero if and only if $cov(Y \mid X) = 0$ and also $corr(Y \mid X) = 0$. In practice, the zero-approximation of an off-diagonal element D_{XY} in the inverse correlation matrix implies the conditional independence of variables X and Y given the rest. This constructs a moral graph. Leveraging the conditional independence information of each pair of variable, we can construct a moral graph, which is the undirected version of the original directed graph bridging any pair of nodes which converge to a common node. This converged node is also called *collider*, and the corresponding V-structure is depicted in Figure 6.10.

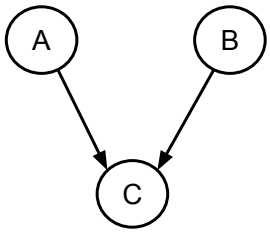


Figure 6.10: V-structure: Two non-adjacent nodes converging to a common node become correlated. In the corresponding moral graph, they are bridged by an additional edge connecting A and B.

For each edge in a moral graph, it requires determining whether it is a bridged edge due to colliders or not. Suppose there is an edge e_{ij} between node i and j, denote the Markov blanket of both nodes as M_{ij} and its potential colliders as C_{ij} forming triangles with e_{ij}. Given M_{ij}, nodes i and j are d-separated from the other nodes. The partial inverse correlation matrix regarding the Markov blanket M_{ij} encodes the conditional independences of i and j. If now the colliders for nodes i and j are excluded from M_{ij}, then the inverse correlation matrix should be changed accordingly where additional information introduced by colliders will be now removed, in other words, e_{ij} is a bridged edge. This can be done simply by comparing inverse correlation matrix regarding $\{M_{ij} \setminus C_{ij}\}$ with the one containing C_{ij}. Discovering the colliders will also orient nodes i and j towards its collider set C_{ij}. The *Detriangulation* is shown in Algorithm 6. Since a bridged edge could be due to multiple colliders, the algorithm

iterates through k-combinations of colliders (k starts from 1 to the maximal number of potential colliders) until the real colliders are found, unless it is not a bridged edge. Note that the algorithm could be computationally intensive when the graph is fully connected.

Algorithm 6: DetriangulateMoralGraph

Input : $\mathcal{G} = (V, E)$, Correlation Σ, threshold σ
Output: A partially directed graph $\mathcal{G}^{\prec}$

foreach $e = (p, q)$ *in* E **do**
  $C \leftarrow \{v_i \in V \mid v_i$ and e form triangle$\}$;
  for *colliders* $\leftarrow$ *each* k*-combination in* C **do**
    /* $k = 1, 2, \ldots, |C|$ */
    $N \leftarrow \{v_i \in V \mid (v_i, p) \in E$ or $(v_i, q) \in E\}$;
    $\hat{N} \leftarrow \{\mathrm{N} \setminus$ colliders$\}$;
    $R \leftarrow \Sigma_N^{-1}$;
    $S \leftarrow \Sigma_{\hat{N}}^{-1}$;
    /* Σ_S indicates the partial correlation matrix from Σ with respect to set S */
    if $S_{pq} < \sigma$ **then**
      if $R_{pq} > \sigma$ **then**
        $E \leftarrow \{E \setminus e\}$;
        orient p and q to all nodes in colliders;
        break;
return $\mathcal{G}^{\prec} \leftarrow \mathcal{G}$

Constraint Propagation. All the V-structures discovered from *Detriangulation* impose underlying constraints on the entire network. These constraints can be propagated through the network conforming to the following rules [98]:

- R_1: Orient $a - b$ into $a \to b$ whenever there is an arrow $c \to a$ and b and c are not adjacent. (No new V-structure).
- R_2: Orient $a - b$ into $a \to b$ whenever there is a path $a \to c \to b$. (Preserve the acyclicity)
- R_3: Orient $a - b$ into $a \to b$ whenever there are two chains $a - c \to b$ and $a - d \to b$ such that c and d are not adjacent. (Three-fork structure)
- R_4: Orient $a - b$ into $a \to b$ whenever there are two chains $a - c \to d$ and $c \to d \to b$ and c and b are not adjacent.

Moreover, R_4 is not required, if the starting orientation is limited to V-structures. After constraint propagation, a maximally directed acyclic graph is formed. It does not require all the edges being directed. All the rules conform to either preserving the acyclicity

or avoiding new V-structures. The pseudo-algorithm for constraint propagation is illustrated in Algorithm 7. It is very important to be aware of the order of applying these rules. The acyclicity should be the first and foremost of all the rules to be satisfied. Each time when an unknown edge is oriented, it should be followed by checking the acyclicity, otherwise a cyclic graph could be created unintentionally.

Algorithm 7: ConstraintPropagation

Input : A partially directed graph $\mathcal{G}^{\prec}$
Output: A maximally oriented graph $\mathcal{G}^{\prec}_{max}$

while $\mathcal{G}^{\prec}$ *is changed* **do**
  if *Edge(X, Y) is undirected and* $\exists$ *directed path from* X *to* Y **then**
    `/* preserve acyclicity */`
    set $X \rightarrow Y$;
    break ;
  if $\exists$ *node* X *where* $Y \rightarrow X \leftrightarrow Z$ **then**
    `/* no new collider */`
    set $Y \rightarrow X \rightarrow Z$;
    break ;
  if $\exists$ *an undirected edge* $X \leftrightarrow Y$ *and* $\exists$ *nonadjacent* Z *and* W
  § *such that* $X \leftrightarrow Z \rightarrow Y$ *and* $X \leftrightarrow W \rightarrow Y$ **then**
    `/* three-fork structure */`
    orient as $X \rightarrow Y$;
    break ;
return $\mathcal{G}^{\prec}_{max}$

Equivalent Class. The maximal DAG could possibly contain several equivalent graphs encoding the same joint probability distribution. Traditional structure learning methods will also end up with an equivalence class of networks. As a final step, it is worth converting the partial DAG (PDAG) to its equivalent completely oriented DAGs (CDAGs). This can be done using dynamic programming. Each undecided edge will be assigned an orientation manually, in the meantime, the constraints should always be updated and propagated. Thus it largely reduces the size of resulting networks and therefore is of great practical use. Finally, we present the *pi-CBN* algorithm combining the parameter estimation for Copulas in Algorithm 8. We adopted the Gaussian Copula. Note that, in Gaussian Copula, the estimated parameter Σ can work as the input for the partial inverse correlation matrix. Thus we gain a significant improvement in running time. Even though there is a large number of missing observations, the learned parameters still suffice for the structure learning by applying a robust nonparametric density estimation (*e.g.*, kernel density estimator) on the univariate marginals [42].

Algorithm 8: The *pi-CBN* Algorithm

Input : Dataset $\mathcal{X}$, threshold σ
Output: All equivalent DAGs $\{\mathcal{G}^*\}$

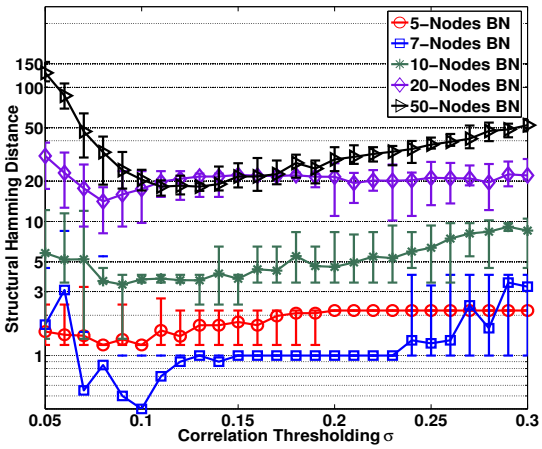
```
Construct fully connected graph G;
marginals ← estimate marginals for each variable;
Σ ← parameter estimation of Gaussian Copula;
Σ^{-1} ← inverse correlation matrix Σ;
foreach e = Σ^{-1}_{ij} do
    if e < σ then
        remove edge (i, j) from G;
set Moral graph Ĝ ← G;
G^≺ ← DetriangulateMoralGraph(Ĝ, Σ, σ);
G^≺_max ← ConstraintPropagation(G^≺);
{G*} ← get equivalent graphs of G^≺_max;
return {G*}
```

6.3.4 Experiments and Results

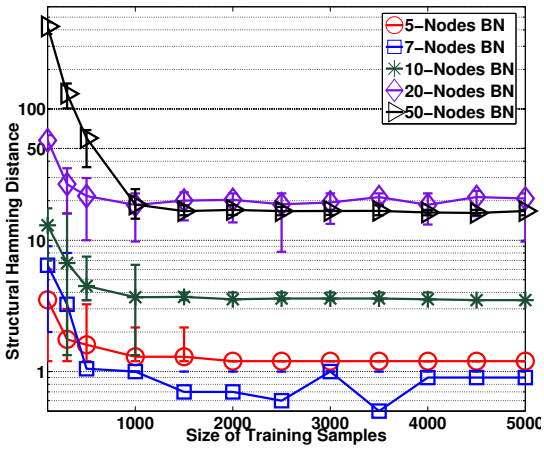
To empirically evaluate the algorithm, we firstly test it on the synthetic datasets to illustrate its capability and efficiency.

On Synthetic Datasets. The synthetic data sets comprise five synthetic networks with node size $\{5, 7, 10, 20, 50\}$ which were created and sampled with the Bayesian network toolbox (BNT) [86]. This was repeated 10 times to examine data variability effects. The size of the data sets (number of instances) ranges from 100 to 5000. Note that none of these networks represents a model of any real-world domain, they are merely created to represent ground truth causal relationships among variables. In order to compare the learned structures with the original networks, we use the Structural Hammming Distance (SHD) [1], which calculates the minimal number of operations converting one network into another. One important parameter of the *pi-CBN* algorithm is the correlation zero-approximation threshold σ (cf. Alg. 8) that controls the number of edges in the graph. However, it is not obvious how to fix that threshold appropriately. Therefore, the first experiment examines the resulting networks by varying σ. In Fig. 6.11(a) shows that SHD approaches a minimum when σ is around 0.1, in the meantime, it achieves a stabler performance with less variance on SHD. Therefore, for further experiments, we chose a proper σ value as 0.1. Another experiment addresses the algorithm's accuracy with increasing training size. In Fig. 6.11(b) it shows that the SHD decreases for larger datasets and reaches a promising value when the sample size exceeds 1000. Besides, for smaller sample size, the algorithm is not stable enough for high variances in SHD. Note that the SHD also correlates with the designed size of network, smaller networks are usually better identified than larger ones. In practice, a dataset with a sample size

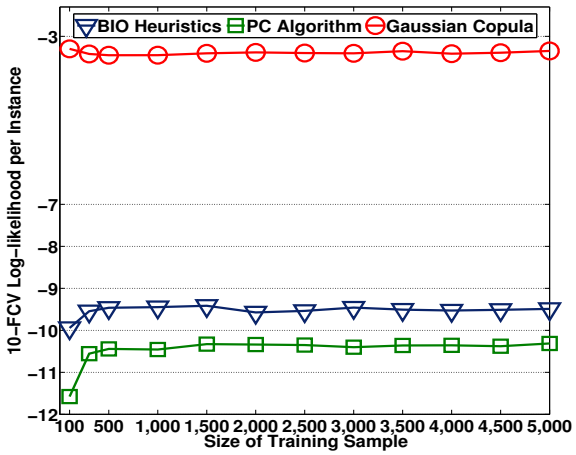
over 500 is considered as significant. To evaluate the parameter estimation, we ran experiments on datasets of different sizes using 10-fold cross validation. We computed the average log-probability per instance over all test folds.



(a) SHD *vs.* Threshold σ



(b) SHD *vs.* Training size



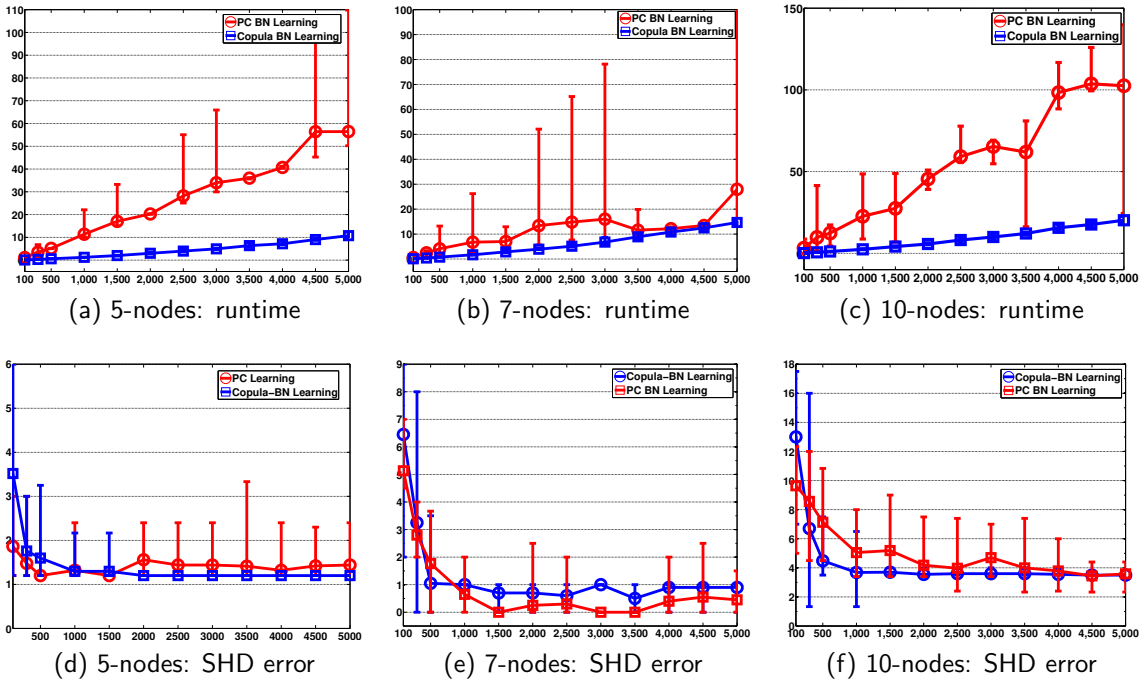
(c) Log-likelihood *vs.* Training size

Figure 6.11: (a) SHD against σ on five synthetic networks (node size 5, 7, 10, 20, 50, respectively) with training sample size 1000 using $pi - CBN$ method. (b) SHD against different training sample sizes on five synthetic networks using $(\sigma = 0.1)$. Sample sizes range from 100 to 5000. (c) Average log-probability per instance on test set against different training sample sizes (range from 100 to 5000) evaluated with 10 folds cross validation, comparing BIC-based heuristic BN Learning and PC Algorithm where Max-fan-in $= 5$ and $\alpha = 0.05$.

Let lp denote the log-probability per instance in a test set $\mathcal{D}_t$ of size m, then: $lp = \frac{1}{|\mathcal{D}_t|}\Sigma_{i=1}^{m} log(p(x_i))$, where x_i is the i-th instance in $\mathcal{D}_t$. PC algorithm and BIC score function based method were taken for comparison. Both of these two algorithms use

MLE for parameter estimation. Since BIC score heuristic method is computationally quite demanding for learning large networks, this experiment was only run on the 5-Nodes network. It is obvious from Fig. 6.11(c) that Copula-based model outperforms the other two on any training sample size.

The last experiment compares the *pi-CBN* algorithm with the PC algorithm on the synthetic networks of size 5, 7, and 10. We show the soundness of *pi-CBN* both in structure learning and computational efficiency. In Fig. 6.12 we give the runtimes and the SHD. The *pi-CBN* performs slightly better on structure learning than the PC algorithm for these small sized networks, and remarkably better for the runtime. Moreover, *pi-CBN* is also more stable than PC, especially when the dataset size is above 1000.



(a) 5-nodes: runtime (b) 7-nodes: runtime (c) 10-nodes: runtime

(d) 5-nodes: SHD error (e) 7-nodes: SHD error (f) 10-nodes: SHD error

Figure 6.12: Runtime (in seconds) and SHD error against different training sample sizes (range from 100 to 5000) on three synthetic networks (size 5,7,10), for *pi-CBN* correlation thresholding $\sigma = 0.1$, for the PC algorithm max-fan-in $= 5$ and thresholding $\alpha = 0.05$.

Reasoning the Attacks. In the end, we illustrate the qualitative performance of using *pi-CBN* to reason a certain type of attack. For this purpose we chose *KDD99* Intrusion Detection Challenge dataset [8] as a benchmark to evaluate our algorithm. Although the *KDD99* dataset is for now an relatively aged source, it is still valid and widely used in the domain of intrusion detection. We random sampled 1000 samples of both normal connections and the `back` Denial-of-Service attacks as our training dataset. The feature size is 41 and an additional class label indicating which kind of attack type

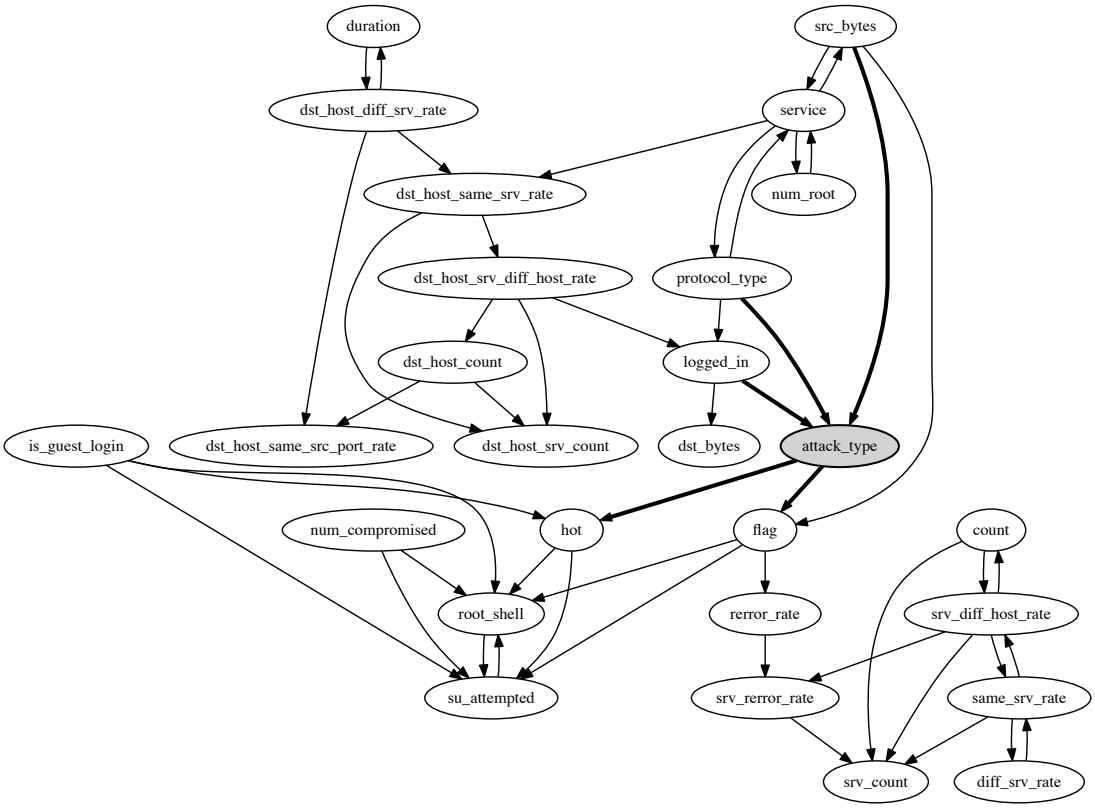


Figure 6.13: Causal reasoning of a DoS attack inferred from *KDD99* dataset using *pi-CBN* algorithm. The dataset only contains 1000 benign samples and 1000 `back` attack samples. Shaded node is the variable of our interest, and bold lines connect direct causes or consequences of a certain attack type.

it is, or just benign sample. Details of relevant features are explained in Table. 6.2. Note that we did not list all features but only those which are directly associated with the attack type. For all details of the features we refer to the description of the dataset[3].

The causal network is depicted in Fig. 6.13 without showing unconnected subgraphs. We shaded the `attack_type` node for our main concern, and most relevant features are connected with directed bold arcs. Preliminary observations present an intuitive explanation of differentiating a *DoS* attack from normal traffic. Most notably, the attack is directly induced by three factors, *i.e.*, the number of data bytes sent from source to destination, which protocol is used and also whether the users successfully logged in. Moreover, we notice that the `back` *DoS* attacks may affect the number of 'hot' indicators and the status flag in the end. In most cases it is very difficult to gain

[3]Kdd99 task description: `http://kdd.ics.uci.edu/databases/kddcup99/task.html`

expert knowledge to correctly evaluate the causal network, however, we believe that the causal relationship can interpret valuable information underlying various adversarial scenarios. Especially the *pi-CBN* can be regarded as a Bayesian model in one unit for prediction, inference, and reasoning.

Feature	Description
`attack_type`	If a connection is an intrusion or benign.
`src_bytes`	Number of data bytes from source to destination.
`protocol_type`	Type of the protocol, e.g. tcp, udp, etc.
`logged_in`	1 If successfully logged in; 0 otherwise.
`hot`	Number of 'hot' indicators.
`flag`	Normal or error status of the connection

Table 6.2: Relevant feature descriptions of *KDD99* dataset

6.3.5 Summary

To secure the learning systems against adversarial threats, we believe that one important aspect in real application is to reason the causality of the attacks, which can be typically encoded by a directed dependence graph. Therefore we explored and proposed a Copula-based Bayesian network methods, especially the Copula function captures the interdependencies of variables of dataset disregarding the explicit marginal distributions. Moreover the inverse Gaussian Copula covariance represents the conditional dependences by examining the near-zero entries. That leads to an efficient structure learning approach named *pi-CBN*. In empirical studies we show that the performance of *pi-CBN* gives both promising results on estimating the joint (conditional) distribution of the BN, and the causal relationship encoded in directed graphs. In addition to the probabilistic model for detecting suspicious activities, we leverage the derived Bayesian network to reason if an activity is a relevant threat, and due to which factors.

6.4 Discussion

From the perspective of defenders, the objective is firstly to provide reliable predictive functions to legitimate users. However, as a result of adaptive and craft adversaries aiming to compromise the learning systems, the defenders are seeking the robustness and security while designing the systems. Although the assumption about the adversaries is not always true, the proposed methods in this chapter explored and disclosed the possibilities to handle the suspicious behaviors more adaptively. For instance, the indicative SVC programs the user given information as a guidance to justify the regularization process, thus it drives the clustering to adapt both cases with or without adversarial samples. On the other hand, the probabilistic graphical model depicted in

Fig. 6.6 introduced a hidden latent variable of the ground truth, conditioned on which we can estimate the expertise of multiple observers (potentially very large), out of which the adversaries can be identified by inspecting the Pearson correlation coefficient with respect to the estimated ground truth. In both cases, additional prior knowledge such as the indicative labels, or proper assumption as the proposed graphical model are expected to overcome the adversarial effect introduced by those suspicious activities. The observations align with recent findings in deep neural networks as well, [51] suggest that the intriguing properties of adversarial examples are due to the linearity rather than the nonlinearity in high dimensional space, they also propose the adversarial training as a certain type of regularization to increase local constancy [84, 50], so that small perturbation on training samples do not lead to unexpected distortion of classifiers. On the other hand, if the prior knowledge and assumption are not properly proposed, the adversarial impact can not be easily countered, this leads to a future work of investigating the correctness of the adversarial hypothesis. Another aspect of securing the learning algorithms associates with the attack analysis. As we introduced, it is of great significance to gain insights of why and how an attack happens. To our knowledge, in literatures of adversarial learning, this is the first work of using Coupla Bayesian network to identify and reason malicious attacks. Future works in this domain involve the incremental version of estimating both parameters and structure of the network. Besides, we expect the Bayesian networks automatically mitigate the adversarial effect in an iterated game setting.

Chapter 7

Conclusion

At last, we conclude this work as follows. Firstly, a brief overview of the main work presented in previous chapters, especially with corresponding final comments and discussion. Moreover, we state the contributions of this work explicitly to repeat the importance of this work. Besides, we suggest the open problems in the domain of adversarial learning and describe the future work for related researchers and practitioners. Finally, we end up the entire dissertation with some final words.

7.1 A Brief Retrospect

The fast pacing development of machine learning techniques enables many related applications to focus on solving problems based on large-scale data stream. The increasing adoption of machine learning applications also motivate adversaries with a certain purpose to gain profit from those learning-based systems. In Chapter.1, we firstly introduced a high-level description of machine learning in a formal language, and extended our discussion with both roles of adversarial learner and robust learner. The antagonistic actions between adversaries and defenders constitute the essential part of this work, namely, the adversarial and secure learning. Given a dataset drawn from a certain distribution, the learner aims to obtain a description of patterns from the data for future prediction tasks, however the adversary injects malicious samples driven by some incentive to either subvert the legitimate function of the learner, or let the crafted samples evade the learner's prediction. Since the stationary assumption is not guaranteed in real world, we believe the proposed antagonistic games between adversaries and defenders are realistic.

Then we walk through a comprehensive study of the adversarial learning theory in Chapter.2. The central learning problem in adversarial environment is well defined as arms races. In arms race problems, defender and attackers are framed up in a game-theoretic setting, that is, both parties seek an equilibrial with a certain level of knowledge for each other. Instead of reactive game setting, the learner is supposed to act in a proac-

tive mode, where the learner simulates the behavior of attack before it really happens. To follow up, we defined our adversarial learning problem in formal language with a high-level adversarial modeling. And this gives a complete taxonomy of adversarial attacks for better understanding different kinds of threats. More specifically, as defined in Table. 3.2, Causative attack and Exploratory attacks are two main streams of adversarial attacks. Both of them produce security violation of the targeted systems on integrity or availability. And note that the security compromise can either be targeted at a specific subset of data samples, or indiscriminately cause a general performance degrade.

Causative attacks aim to alter the legitimate function of the learner, in binary classification problem, the classification boundary is distorted by injecting some adversarial samples. In general, causative attacks are mostly indiscriminate, since the impact on classifier boundary compromises its predictions on unspecific unseen data. But by carefully modeling the adversary's objective, causative attacks can also be targeted. In Sect. 4.1, real examples of causative label flips attack on SVMs have shown the vulnerability of the binary SVM classifier. Given a small number of label flips, the SVM boundary is driven to produce increasing classification errors. Several attack strategies are applied to find the vulnerable area in the training data space. In one-shot game, the defender is retrained on the tainted dataset not realizing the existence of the adversarial attack points. Although the attack strategies are quite adaptive, the objectives are all commonly set to maximize the classification error. Similar causative attack can also be applied on the embedded feature selection methods, such as LASSO, Ridge regression and Elastic net as described in Sect. 4.2. As a preliminary learning stage, embedded feature selection combines the parts of subset selection and model learning in an inseparable unit. We shown that a gradient-based poisoning attack finds optimal attack points, which finally entice the learning model to select a problematic subset of features. Moreover, we empirically showed that there is a trade-off between sparsity and stability. Therefore, elastic net with a hybrid $\ell_1 + \ell_2$ regularization term might be a compromised but less risky embedded feature selection method. For exploratory attack, more often referred as evasion attack, we proposed a formal definition of evasion attack and also a general attack framework, then revisited an gradient-based evasion attack in [12]. The evasion attacks are mostly targeted and defined by three axises: (**i**.) directedness (**ii**.) closeness (**iii**.) adversarial cost. The evasion example we revisited falls into our formal definition of the evasion attack, but it is a pessimistic attack and does not hold a feature subset within a malicious domain.

On the other hand side, to proactively secure the learning algorithms, we provide different strategies to enhance the security and robustness of machine learning. More specifically, prior information and proper assumption about adversaries are involved to mitigate the malicious impact. The countermeasures are integrated in the optimization of the learning problems. In the example of Support Vector Clustering in Sect. 6.1, additional supervised labels are given by users, the labels are considered as prior knowledge with high confidence. Therefore, impact function is proposed to integrate the information, such that the process of regularization can be more robust and flexible. Comparing

to original SVC model and semi-supervised SVM, the indicative SVC gives promising results on anomaly detection, especially when the anomalies tend to dominate the scenario to mislead the classifier. Another idea introduced in Sect. 6.2 uses probabilistic graphical model to model a multi-observer scenario. Given a latent variable about unknown ground truth, the expertise of observers can be estimated by the conditional distribution, as well as the unknown ground truth. As long as the observers behave suspiciously, they are considered as potential adversaries. Moreover, an objective photo ranking system named *OPARS* is designed to provide objective image ranking function with an active selection process. It is shown that even there exists a number of biased users, the system can still provide objective recommendations of images. Finally in Sect. 6.3, we explored another line of research to reason the attacks using a Copula Bayesian network approach. The key idea is to discover the causal relationship among variables, which is encoded in a directed acyclic graph. We showed that Copula function captures multivariate correlations disregarding the marginal distributions explicitly and also enables efficient structure learning leveraging the partial inverse correlation matrix. Empirical experiments of the proposed *pi-CBN* algorithm showed that it outperforms other baselines, moreover, we showed a preliminary example of using *pi-CBN* to reason certain kind of attack.

7.2 Contributions

Adversarial learning derives itself from machine learning community as a closely related subfield. The goal of this research domain, also regarded as the goal of this work, is to provide a comprehensive understanding of the adversarial learning problem, and then advocate to enhance the security and robustness of any machine learning algorithm. We studied the adversarial learning problem from the perspectives of both adversaries and learners, and we believe this is the only way to secure the learning systems against potential threats, that incurred by those crafty and adaptive adversaries. In spite of previous existing research works in this domain, we believe that our contributions through out this dissertation are very fruitful and significant, they are mainly in three aspects.

On A Unified Adversarial Learning Framework

First and foremost, we defined and proposed a unified theoretical framework to describe adversarial learning problem in formal language. By targeting on a general statistical learning problem, we present the attack model in a high level objective function, which is shown as a proper definition in most cases. To better understanding the attacker's strategy, we further introduced the taxonomy of adversarial attacks carefully defined in three dimensions, namely, (**i**.) the security violation, (**ii**.) attacker's capacity, and (**iii**.) attacker's influence. Besides, we gave four most common attacks with examples described within our theoretical framework. Although in previous literatures [6, 7, 59]

the adversarial learning were literally well studied, we believe our work contribute a more comprehensive and thorough theoretical framework for this domain, especially our proposed framework is more justified with respect to those recently emerging adversarial threats [140, 12, 137].

Empirical Guidance of Adversarial Attacks

Guided by the theoretical framework of adversarial learning, we showed several empirical studies of real attacks on different algorithms. More specifically, novel label flips attacks on SVMs are conducted to subvert the classifier so as to cause a significant classification error on unseen data samples. We proposed various strategies to find optimal labels to be flipped, namely, by duplicating training set, continuous label relaxation, hyperplane tilting and correlated clusters. These are regarded as the most recent adversarial threats against SVMs, and more importantly our attacks can also be applied on similar learning algorithms in *Tikhonov* regularization family. Besides, we are the first researchers who explored the empirical attacks on embedded feature selection algorithms, such as LASSO, Ridge regression, and Elastic nets. By employing a gradient ascent technique, we found optimal poisoning points to compromise the feature selection algorithms. Given a small portion of poisoning data points, the feature selection process can be corrupted to produce a random guess. The causative attack on LASSO demonstrated empirically that stability will be compromised if sparsity is present. Another contribution is that we defined a general attack model for evasion attack. By revisiting the work [12], we highlighted its consistency of attack with our evasion framework. In all, we believed that our contribution on the empirical guidance of adversarial attacks serves the machine learning engineers and researchers as state-of-art threats, which needs to be mitigated during algorithm design. Also, the proposed attacks can also be served as evaluation tools to access the robustness and security of a certain learning system.

Securing the Learning Algorithms

Finally we contributed to provide thoughts and implementations to secure machine learning algorithms in several works. In the proactive arms race problem, designing a secure learning systems associates with involvement of adversaries. In our work present in Sect. 6.1 and Sect. 6.2 we proposed two robust learning algorithms, both of which counter adversarial effect by introducing some level of prior knowledge. Thus we believe that our work also present effective strategies of designing robust and secure learning algorithms. In the end, we show a preliminary work on reasoning the attacks. Using the Copula Bayesian Network, causal relationship among variables can be efficiently captured, and further interprets why and how an adversarial attack actually happens. To our knowledge, this is the first work in this domain, which not only identifies attacks (anomalies), but also causally explains their strategies. The causality discovery in this

aspect is considered extremely important and informative for related researchers and practitioners.

7.3 Open Problems and Future Work

In spite of our fruitful results presented in the field of adversarial learning, there are many open problems that are not yet investigated. Our research of adversarial learning is far beyond mature to provide insightful solutions to these issues. Nevertheless we list the open problem in this section and leave them as a main part of our future work in this direction. We also expect researchers can work on these problems to further push the advance of secure and robust machine learning.

1. **Is our adversarial theory able to capture all kinds of attacks?**
   The theoretical framework and taxonomy are both derived from statistical learning theory written in a regularization objective. However, it is not clear if our framework is able to capture all types of attacks. Especially, our taxonomy is defined in a binary classification case, without loss of generality. For regression and clustering problems, we will need to extend our framework to fit these situations.

2. **Can causative attack also be targeted?**
   As seen in Sect. 3.4 four most common attack scenarios are introduced with their objective respectively. However, for causative attack, is there also realistic adversarial attack which subverts the classifier only on a specific subset of samples? The targeted causative attack may lead the classification boundary to a particular alteration such that only a limited small area in the data space is affected. This clearly needs more complicate attack strategy, since the adversary attempts to manipulate the classifier more precisely.

3. **Can evasion attack also be indiscriminate?**
   Likely, evasion attack normally concentrates on a small subset of data that misclassified by the learners. Indiscriminate evasion attack aims to find common patterns of some malicious data, therefore given a general set malicious samples, they can be conducted to evade the classifier. This is relevant since any malicious information might be obfuscated by the indiscriminate evasion attack without being detected. In the future we will explore the possibility of this kind of attack.

4. **How can we derive the error bound of a particular attack?**
   In our adversarial learning theory, we have not yet investigated the error bound of any particular attack. However, for a given type of attack, we expect an error bound analysis for the targeted learning algorithm, such that we can prove theoretically how many attack points we need to subvert the learning system to what degree. This is an important research direction for the adversarial learning theory in the future.

5. **Are the label flip attacks on SVMs applicable on other learning models?**
   Different causative label flip attack strategies are presented on the binary SVMs classifier. In the future we will also examine their effectiveness on other learning algorithms, and try to derive some common properties of our attack strategies. We expect to capture some characteristics of the causative label flip attacks to enhance the security of supervised learning.

6. **What is the trade-off effect between sparsity and stability?**
   In poisoning embedded feature selection algorithms, we showed that the Elastic net algorithm is more robust against adversarial attacks, since it involves a hybrid of ℓ_1 and ℓ_2 regularization. One would be interested how does the parameter ρ controls the sparsity term affect the feature stability under our attack. A quantitative evaluation of this problem will be valuable and interesting for feature selection or other sparse models.

7. **How to design more efficient online poisoning attacks?**
   The causative attacks on both SVMs and embedded feature selection involve intensive computational overhead, all the methods need to search exhaustively in a hypothesis attack space to find the most effective attack points. In real scenarios, attackers may adapt their strategy according to the response of defenders, therefore they may prefer online algorithms to efficiently compute a set of optimal or at least suboptimal attack points to tamper the system in real time.

8. **How can the defender counter the attacks in real time?**
   Likely the defenders are supposed to be aware of the existence of the attacks, and take the counterproductive actions as variables of their learning function. That is, the defenders design the objective functions with respect to the attack strategies in real time, forming an iterated game setting.

9. **Is our evasion attack framework applicable on general classification systems?**
   It seems that discriminative model is more vulnerable than generative model, since the discriminative nature of the classifier does not account for the data distribution. Therefore, evasion attacks can more reliably identify the classification bound in the limit of probes. One research point is to further apply our defined evasion attack on different classification systems, *e.g.*, spam filtering, fraud detection systems.

10. **How to quantitatively evaluate attack's impact in a cost-sensitive scenario?**
    In most of our work, we assume that the cost of manipulating different samples is universal. However in practice, this is rarely true. Therefore we need to define a quantitative framework to evaluate the attack's impact in a cost-sensitive scenario. That is, manipulating different samples induces different adversarial cost.

11. **To increase the security of learning algorithms, how to identify those adversarial noise from regular sample noise?**
    To design a secure and robust learning system in the adversarial environment, we may apply a preliminary step to generate those adversarial noise, and then train on them to mitigate their effect on local perturbation. Different from those regular sample noise, the adversarial samples are assumed to present more explicit patterns in a malicious way, that is, the adversarial examples always follow the direction of gradient ascent to maximize the negative effect. This is still a research area yet to be explored.

12. **Is there a systematic approach to evaluate the security of a learning system?**
    As in the IT security, a systematic approach for accessing the security of a learning system is desirable. For instance, a learning system needs to be poisoned or evaded if and only if the attackers consume a certain amount of adversarial cost. When the adversarial cost is considered as unrealistic, we say the learning system is secure to provide reliable services.

13. **How can we transfer the causality to the security of a learning system?**
    The causal network gives us potential interpretation of a certain adversarial attack, so that security engineers can easily conclude which features are more relevant to the attack, and how they are mitigated. There is still a huge amount of efforts needs to be put in this direction. For instance, we can approximate a linear model [104] around an adversarial example to examine the potential cause of attacks.

7.4 Final Words

To conclude the entire work, we stress the importance of the adversarial learning again. Needless to say, machine learning is nowadays infiltrated in many industries as a service, however it is not as secure as other IT-based systems. The increasingly adopted data-driven technologies also drive adversaries with a certain incentive to gain profit, especially when the adversaries are educated with profound knowledge about machine learning. Adversarial learning is a young but fast developing area, researchers are facing intense competitions with malicious users. Despite the arms race problems, machine learning engineers have less concerns on developing robust and secure learning system, instead, they put their effort mainly on how to improve the prediction performance of the learning models, disregarding the potential threats. We hope our contributions to the community can invoke attentions from related researchers and practitioners, who would include security analysis while designing the learning systems.

Bibliography

[1] L. M. Acid, S.; de Campos. Searching for bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal Of Artificial Intelligence Research*, 18:445–490, 2003.

[2] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, K. V. Ch, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age*, pages 9–17, 2000.

[3] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[4] P. Auer. Learning nested differences in the presence of malicious noise. In *Proceedings of the 6th International Conference on Algorithmic Learning Theory*, ALT '95, pages 123–137, London, UK, UK, 1995. Springer-Verlag.

[5] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[6] M. Barreno, B. Nelson, A. Joseph, and J. Tygar. The security of machine learning. *Machine Learning*, 81:121–148, 2010.

[7] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proc. ACM Symp. Information, Computer and Comm. Sec.*, ASIACCS '06, pages 16–25, New York, NY, USA, 2006. ACM.

[8] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *SIGKDD Explor. Newsl.*, 2(2):81–85, Dec. 2000.

[9] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik. A support vector clustering method. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 724–727 vol.2, 2000.

[10] B. Biggio, S. Bulò, I. Pillai, M. Mura, E. Mequanint, M. Pelillo, and F. Roli. Poisoning complete-linkage hierarchical clustering. In P. Fränti, G. Brown, M. Loog, F. Escolano, and M. Pelillo, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 8621 of *Lecture Notes in Computer Science*, pages 42–52. Springer Berlin Heidelberg, 2014.

[11] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In C. Sansone, J. Kittler, and F. Roli, editors, *10th International Workshop on Multiple Classifier Systems (MCS)*, volume 6713 of *Lecture Notes in Computer Science*, pages 350–359. Springer-Verlag, June 2011.

[12] 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 H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Part III*, volume 8190 of *Lecture Notes in Computer Science*, pages 387–402. Springer Berlin Heidelberg, 2013.

[13] B. Biggio, I. Corona, B. Nelson, B. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli. Security evaluation of support vector machines in adversarial environments. In Y. Ma and G. Guo, editors, *Support Vector Machines Applications*, pages 105–153. Springer International Publishing, 2014.

[14] B. Biggio, L. Didaci, G. Fumera, and F. Roli. Poisoning attacks to compromise face templates. In *6th IAPR Int'l Conf. on Biometrics (ICB 2013)*, pages 1–7, Madrid, Spain, 2013.

[15] B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems for robust classifier design in adversarial environments. *Int'l J. Mach. Learn. and Cybernetics*, 1(1):27–41, 2010.

[16] B. Biggio, G. Fumera, and F. Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, April 2014.

[17] B. Biggio, G. Fumera, F. Roli, and L. Didaci. Poisoning adaptive biometric systems. In G. Gimel'farb, E. Hancock, A. Imiya, A. Kuijper, M. Kudo, S. Omachi, T. Windeatt, and K. Yamada, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 7626 of *Lecture Notes in Computer Science*, pages 417–425. Springer Berlin Heidelberg, 2012.

[18] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In *Journal of Machine Learning Research - Proc. 3rd Asian Conference on Machine Learning (ACML 2011)*, volume 20, pages 97–112, Taoyuan, Taiwan, November 2011.

[19] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In *Journal of Machine Learning Research - Proc. 3rd Asian Conf. Machine Learning*, volume 20, pages 97–112, November 2011.

[20] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In J. Langford and J. Pineau, editors, *29th Int'l Conf. on Machine Learning*, pages 1807–1814. Omnipress, 2012.

[21] B. Biggio, I. Pillai, S. R. Buló, D. Ariu, M. Pelillo, and F. Roli. Is data clustering in adversarial settings secure? In *AISec'13: Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security*, pages 87–98, Berlin, Nov. 2013. ACM, ACM.

[22] B. Biggio, I. Pillai, S. R. Bulò, D. Ariu, M. Pelillo, and F. Roli. Is data clustering in adversarial settings secure? In *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security*, AISec '13, pages 87–98, New York, NY, USA, 2013. ACM.

[23] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[24] M. Brückner, C. Kanzow, and T. Scheffer. Static prediction games for adversarial learning problems. *J. Mach. Learn. Res.*, 13:2617–2654, 2012.

[25] M. Brückner and T. Scheffer. Nash equilibria of static prediction games. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 171–179. 2009.

[26] N. Bshouty, N. Eiron, and E. Kushilevitz. Pac learning with nasty noise. In O. Watanabe and T. Yokomori, editors, *Algorithmic Learning Theory*, volume 1720 of *Lecture Notes in Computer Science*, pages 206–218. Springer Berlin Heidelberg, 1999.

[27] N. H. Bshouty, N. Eiron, and E. Kushilevitz. Pac learning with nasty noise. *Theor. Comput. Sci.*, 288(2):255–275, Oct. 2002.

[28] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS*, pages 409–415. MIT Press, 2000.

[29] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[30] C.-C. Chang and C.-J. Lin. LibSVM: a library for support vector machines, 2001.

[31] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[32] A. Christmann and I. Steinwart. On robust properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research*, 5:1007–1034, 2004.

[33] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. *JMLR*, 9:1757–1774, 2008.

[34] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *IEEE Symposium on Security and Privacy*, pages 81–95, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[35] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 99–108, New York, NY, USA, 2004. ACM.

[36] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 99–108, Seattle, 2004.

[37] A. Dawid and A. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, pages 20–28, 1979.

[38] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. KDD '04, pages 551–556, New York, NY, USA, 2004. ACM.

[39] C. P. Diehl and G. Cauwenberghs. Svm incremental learning, adaptation and optimization. In *International Joint Conference on Neural Networks*, pages 2685–2690, 2003.

[40] P. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, Oct. 2012.

[41] G. Elidan. Copula bayesian networks. In *The 24th Annual Conference on Neural Information Processing Systems(NIPS)*, pages 559–567, 2010.

[42] G. Elidan. Inference-less density estimation using copula bayesian networks. *Uncertainty in Artificial Intelligence (UAI)*, 26, 2010.

[43] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia. A behavioral approach to worm detection. In *Proceedings of the 2004 ACM Workshop on Rapid Malcode*, WORM '04, pages 43–53, New York, NY, USA, 2004. ACM.

[44] M. Ester, H. peter Kriegel, J. S, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[45] B. Frenay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5):845–869, 2013.

[46] J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2 2010.

[47] N. Friedman. Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm. In *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

[48] A. Globerson and S. Roweis. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 353–360, New York, NY, USA, 2006. ACM.

[49] A. Globerson and S. T. Roweis. Nightmare at test time: robust learning by feature deletion. In W. W. Cohen and A. Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning*, volume 148, pages 353–360. ACM, 2006.

[50] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[51] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

[52] P. Graham. A plan for spam, 2002.

[53] J. Graham-Cumming. How to beat an adaptive spam filte. *Presentation at the MIT Spam Conference*, Jan. 2004.

[54] N. Guan, X. Huang, L. Lan, Z. Luo, and X. Zhang. Graph based semi-supervised non-negative matrix factorization for document clustering. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 1, pages 404–408, 2012.

[55] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[56] Z. He, S. Deng, and X. Xu. Outlier detection integrating semantic knowledge. In *Advances in Web-Age Information Management*, volume 2419 of *LNCS*, pages 126–131. Springer Berlin Heidelberg, 2002.

[57] D. Heckerman. Tutorial on learning in bayesian networks. Technical Report MSR-TR-95-06, Microsoft, 1995.

[58] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, Feb. 1970.

[59] L. Huang, A. D. Joseph, B. Nelson, B. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *4th ACM Workshop on Artificial Intelligence and Security (AISec 2011)*, pages 43–57, Chicago, IL, USA, 2011.

[60] S. Hui and S. Walter. Estimating the error rates of diagnostic tests. *Biometrics*, pages 167–171, 1980.

[61] IBM. The x-force threat and risk report.

[62] A. D. Joseph, P. Laskov, F. Roli, J. D. Tygar, and B. Nelson. Machine Learning Methods for Computer Security (Dagstuhl Perspectives Workshop 12371). *Dagstuhl Manifestos*, 3(1):1–30, 2013.

[63] M. Kearns and M. Li. Learning in the presence of malicious errors. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 267–280, New York, NY, USA, 1988. ACM.

[64] M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22(4):807–837, 1993.

[65] M. Kloft and P. Laskov. Online anomaly detection under adversarial impact. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 405–412, 2010.

[66] M. Kloft and P. Laskov. Security analysis of online centroid anomaly detection. *Journal of Machine Learning Research*, 13:3647–3690, 2012.

[67] A. Kolcz and C. H. Teo. Feature weighting for improved classifier robustness. In *Sixth Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, 2009.

[68] R. E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41–78, July 1993.

[69] L. I. Kuncheva. A stability index for feature selection. In *Proc. 25th IASTED International Multi-Conference: Artificial Intelligence and Applications*, AIAP'07, pages 390–395, Anaheim, CA, USA, 2007. ACTA Press.

[70] T. Lal, O. Chapelle, J. Weston, and A. Elisseeff. Embedded methods. In I. Guyon, M. Nikravesh, S. Gunn, and L. Zadeh, editors, *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 137–165. Springer Berlin Heidelberg, 2006.

[71] A. Lazarevic, A. Ozgur, L. Ertoz, J. Srivastava, and V. Kumar. A comparative study of anomaly detection schemes in network intrusion detection. In *In Proceedings of the Third SIAM International Conference on Data Mining*, 2003.

[72] Y. Lecun and C. Cortes. The mnist database of handwritten digits.

[73] Y. Lecun and C. Cortes. The MNIST database of handwritten digits. 2009.

[74] B. Li and Y. Vorobeychik. Feature cross-substitution in adversarial classification. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2087–2095. Curran Associates, Inc., 2014.

[75] Y. Liao and V. R. Vemuri. Using text categorization techniques for intrusion detection. In *Proceedings of the 11th USENIX Security Symposium*, pages 51–59, Berkeley, CA, USA, 2002. USENIX Association.

[76] D. Lowd and C. Meek. Adversarial learning. In *Proc. 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 641–647, Chicago, IL, USA, 2005. ACM Press.

[77] D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *Second Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, 2005.

[78] D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *In Proceedings of the Second Conference on Email and Anti-Spam (CEAS*, 2005.

[79] H. Lütkepohl. *Handbook of matrices*. John Wiley & Sons, 1996.

[80] D. Maiorca, I. Corona, and G. Giacinto. Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, ASIA CCS '13, pages 119–130, New York, NY, USA, 2013. ACM.

[81] D. Maiorca, G. Giacinto, and I. Corona. A pattern recognition system for malicious pdf files detection. In P. Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 7376 of *Lecture Notes in Computer Science*, pages 510–524. Springer Berlin Heidelberg, 2012.

[82] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proc. AAAI Workshop on learning for text categorization*, pages 41–48, 1998.

[83] S. Mei and X. Zhu. The security of latent dirichlet allocation. In *18th Int'l Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2015.

[84] T. Miyato, S. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing by virtual adversarial examples. *CoRR*, abs/1507.00677, 2015.

[85] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707, 2002.

[86] K. Murphy. Bayes net toolbox for matlab.

[87] N. M. Nasrabadi, T. D. Tran, and N. Nguyen. Robust lasso with missing and grossly corrupted observations. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1881–1889. Curran Associates, Inc., 2011.

[88] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, April 1995.

[89] R. B. Nelsen. *An Introduction to Copulas*. Springer Science+Business Media, Inc., 2006.

[90] 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. In *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, Berkeley, CA, USA, 2008. USENIX Association.

[91] B. Nelson, M. Barreno, F. Jack Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Misleading learners: Co-opting your spam filter. In *Machine Learning in Cyber Trust*, pages 17–51. Springer US, 2009.

[92] B. Nelson, B. Biggio, and P. Laskov. Understanding the risk factors of learning in adversarial environments. In *4th ACM Workshop on Artificial Intelligence and Security*, AISec '11, pages 87–92, Chicago, IL, USA, 2011.

[93] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. D. Tygar. Query strategies for evading convex-inducing classifiers. *J. Mach. Learn. Res.*, 13:1293–1332, 2012.

[94] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition 2015*, Boston, Massachusetts, US, Dec.

[95] N. Nguyen and T. Tran. Robust lasso with missing and grossly corrupted observations. *IEEE Trans. Inf. Theor.*, 59(4):2036–2058, 2013.

[96] V. Paxson. Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 3–3, Berkeley, CA, USA, 1998. USENIX Association.

[97] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[98] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[99] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[100] J. Pfoh, C. Schneider, and C. Eckert. Leveraging string kernels for malware detection. In *Proceedings of the 7th International Conference on Network and System Security*, Lecture Notes in Computer Science. Springer, June 2013.

[101] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 759–766, New York, NY, USA, 2007. ACM.

[102] V. Raykar, S. Yu, L. Zhao, A. Jerebko, C. Florin, G. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: Whom to trust when everyone lies a bit. In *Proc. 26th ICML*, pages 889–896. ACM, 2009.

[103] V. Raykar, S. Yu, L. Zhao, G. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *JMLR*, 11:1297–1322, 2010.

[104] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[105] G. Robinson. A statistical approach to the spam problem. *Linux J.*, 2003(107):3, 2003.

[106] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference*, IMC '09, pages 1–14, New York, NY, USA, 2009. ACM.

[107] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. *AAAI Technical Report WS-98-05, Madison, Wisconsin*, 1998.

[108] K. Scheinberg. An efficient implementation of an active set method for svms. *Journal of Machine Learning Research*, 7:2237–2257, 2006.

[109] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, July 2001.

[110] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, July 2001.

[111] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection, 2000.

[112] R. A. Servedio. Smooth boosting and learning with malicious noise. *J. Mach. Learn. Res.*, 4:633–648, Dec. 2003.

[113] V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear svms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 477–484, New York, NY, USA, 2006. ACM.

[114] A. Sklar. *Fonctions de repartition a n dimensions et leurs marges*, volume 8. Publications de l'Institut de Statistique de L'Universite de Paris, 1959.

[115] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski. Clustering approaches for anomaly based intrusion detection. *Proceedings of intelligent engineering systems through artificial neural networks*, 2002.

[116] C. Smutz and A. Stavrou. Malicious pdf detection using metadata and structural features. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 239–248, New York, NY, USA, 2012. ACM.

[117] D. Spiegelhalter and P. Stovin. An analysis of repeated biopsies following cardiac transplantation. *Statistics in Medicine*, 2(1):33–40, 1983.

[118] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, 2000.

[119] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 2002.

[120] G. Stempfel and L. Ralaivola. Learning svms from sloppily labeled data. In C. Alippi, M. M. Polycarpou, C. G. Panayiotou, and G. Ellinas, editors, *ICANN*, volume 5768 of *Lecture Notes in Computer Science*, pages 884–893. Springer, 2009.

[121] G. Stempfel and L. Ralaivola. Learning svms from sloppily labeled data. In *Proceedings of the 19th International Conference on Artificial Neural Networks: Part I*, ICANN '09, pages 884–893, Berlin, Heidelberg, 2009. Springer-Verlag.

[122] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, pages 1–10, Dec. 2013.

[123] C. H. Teo, A. Globerson, S. Roweis, and A. Smola. Convex learning with invariances. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1489–1496. MIT Press, Cambridge, MA, 2008.

[124] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[125] M. Tubaishat and S. Madria. Sensor networks: an overview. *Potentials, IEEE*, 22(2):20–23, 2003.

[126] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, Nov. 1984.

[127] L. G. Valiant. Learning disjunction of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'85, pages 560–566, San Francisco, CA, USA, 1985. Morgan Kaufmann Publishers Inc.

[128] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[129] V. N. Vapnik. An overview of statistical learning theory. *Trans. Neur. Netw.*, 10(5):988–999, Sept. 1999.

[130] N. Šrndić and P. Laskov. Detection of malicious pdf files based on hierarchical document structure. In *Proceedings of the 20th Annual Network & Distributed System Security Symposium (NDSS)*. The Internet Society, 2013.

[131] F. Wang, W. Liu, and S. Chawla. On sparse feature attacks in adversarial learning. In *IEEE Int'l Conf. on Data Mining (ICDM)*, pages 1013–1018. IEEE, 2014.

[132] G. Wang, T. Wang, H. Zheng, and B. Y. Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *23rd USENIX Security Symposium (USENIX Security 14)*, San Diego, CA, 2014. USENIX Association.

[133] L. Wehenkel. Machine learning approaches to power-system security assessment. *IEEE Expert*, 12(5):60–72, Sep 1997.

[134] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proc. 23rd NIPS*, volume 22, pages 2035–2043, 2009.

[135] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, Ltd, 2008.

[136] G. Wittel and S. Wu. On attacking statistical spam filters. In *Proc. of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, July 2004.

[137] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In *ICML'15*, Lille, France, July 2015.

[138] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli. Support vector machines under adversarial label contamination. *Journal of Neurocomputing, Special Issue on Advances in Learning with Label Noise*, August 2014.

[139] H. Xiao and C. Eckert. Indicative support vector clustering with its application on anomaly detection. In *IEEE 12th International Conference on Machine Learning and Applications (ICMLA'13)*, Miami, Florida, USA, December 2013.

[140] H. Xiao, H. Xiao, and C. Eckert. Advsersarial label flips attack on support vector machines. In *20th European Conference on Artificial Intelligence (ECAI)*, Montepellier, France, August 2012.

[141] H. Xiao, H. Xiao, and C. Eckert. Learning from multiple observers with unknown expertise. In *Proceedings of 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Gold Coast, Australia, April 2013.

[142] H. Xiao, H. Xiao, and C. Eckert. Opars: Objective photo aesthetics ranking system. In *34th European Conference on Information Retrieval (ECIR'13)*, Moscow, Russia, March 2013.

[143] H. Xu, C. Caramanis, and S. Mannor. Sparse algorithms are not stable: A no-free-lunch theorem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(1):187–193, Jan 2012.

[144] Y. Yan, R. Rosales, G. Fung, and J. Dy. Active learning from crowds. In *Proc. 28th ICML*, 2011.

[145] Y. Yan, R. Rosales, G. Fung, M. Schmidt, G. Hermosillo, L. Bogoni, L. Moy, J. Dy, and P. Malvern. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proc. AISTATS*, 2010.

[146] D.-Y. Yeung and C. Chow. Parzen-window network intrusion detectors. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 385–388 vol.4, 2002.

[147] F. Zhang, P. Chan, B. Biggio, D. Yeung, and F. Roli. Adversarial feature selection against evasion attacks. *IEEE Transactions on Cybernetics*, PP(99):1–1, 2015.

[148] Y. Zhou, H. Yu, and X. Cai. A novel k-means algorithm for clustering and outlier detection. FITME '09, pages 476–480, Washington, DC, USA, 2009. IEEE Computer Society.

[149] X. Zhu. Semi-supervised learning literature survey, 2006.

[150] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320, 2005.