TECHNISCHEN UNIVERSITÄT MÜNCHEN

Lehrstuhl für Informatik XX

From Adversarial Learning to Reliable and Scalable Learning

Han Xiao

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

Doktors der Naturwissenschaften (Dr. rer. nat)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Helmut Seidl
Prüfer der Dissertation:	1. Univ.-Prof. Dr. Claudia Eckert
	2. Univ.-Prof. Dr. Daniel Cremers

Die Dissertation wurde am 14.08.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 02.03.2015 angenommen.

Abstract

Nowadays machine learning is considered as a vital tool for data analysis and automatic decision making in many modern enterprise systems. However, there is an emerging threat that adversaries can mislead the decision of the learning algorithm by introducing security faults into the system. Previous security research did not closely examined the vulnerabilities of the learning algorithms to adversarial manipulations. Understanding these threats is the only way to build robust learning algorithms for security-sensitive applications. This dissertation is organized in three parts. Each part contributes the new results in adversarial, reliable and scalable machine learning, respectively.

The first part of this dissertation studies how machine learning algorithms behave in the presence of the adversary. In particular, I provide analyses for the exploratory attack on convex-inducing classifiers and causative attack on support vector machines. Under the analyses are the tools from convex geometry and optimization theory. Using real-world data, I demonstrate the devastating impact of the attack algorithms on a newsletter classifier and a face recognition system.

The second part focuses on developing reliable learning algorithm that is resilient to the adversarial noise. I consider the problem of learning from multiple observers, in which each instance is associated with multiple but unreliable labels. To solve this problem, I develop a hierarchical Gaussian process model and consider the groundtruth label as a latent variable. The parameters of the model can be efficiently estimated by maximizing a posterior. The successful application of my method on the task of aesthetics score assessment would raise practitioners a great interest.

The third part concentrates on developing scalable online learning algorithms for security applications. I propose three systematic approaches for learning from large-scale data stream. The first method employs a set of Gaussian process models to perform real-time online regression. The second method is based on a variant of second-order perceptron to predict the upcoming label in a sequence. The last method provides a novel distributed learning framework for the client-server settings. It can learn from partially labeled data while minimizing the communication-cost over the network.

Zusammenfassung

Machinelles Lernen stellt heutzutage ein essentielles Tool für die Datenanlyse und automatische Entscheidungsfindung in vielen modernen Enterprisesystemen dar. Dadurch ergeben sich jedoch auch neuartige Angriffsvektoren. Besonders kritisch ist dabei, dass Angreifer durch das Ausnützen von Sicherheitslücken den Lernalgorithmus gezielt in die Irre führen können. Überraschenderweise werden solche Angriffe in der bestehenden Forschung jedoch kaum untersucht. Um sichere Lernalgorithmen entwickeln zu können ist aber ein eingehendes Verständnis dieser Angriffsformen nötig.

Die vorliegende Doktorarbeit ist in drei Teile gegliedert. Im ersten Teil wird untersucht wie sich Lernalgorithmen bei gezielter Manipulation durch den Angreifer verhalten. Basierend auf dem erworbenen Wissen werden dann im zweiten Teil der Arbeit zuverlässige Lernalgorithmen entwickelt, die immun gegenüber der Manipulationen durch den Angreifer sind. Schließlich werden im dritten Teil der Arbeit skalierbare Onlinelernalgorithmen für Sicherheitsanwendungen vorgestellt.

Acknowledgments

First and foremost I would like to thank my advisor, Professor Claudia Eckert, whose encouragement, guidance and support she has offered me throughout my graduate career. Moreover, the freedom given by Claudia allowed me to purse my own research interests. She shared the excitement when I had accomplishment and offered me encouragement when I was frustrated. Claudia made innumerable contributions to my development as a researcher and my ambitions to be a data scientist.

I would like to thank Professor Shou-De Lin for his invitation of a six-month research visit at National Taiwan University. Shou-De with his kindness and invaluable experience guided me to finish Chapter 11. I would also like to thank Phillip B. Gibbons for his suggestions, insights and revisions on Chapter 11. It is Shou-De and Phillip's dedication that made this chapter possible.

I would particularly like to thank Huang Xiao for his hard work and critical contributions to Chapter 6, Chapter 7 and Chapter 8. He is an extraordinary collaborator and friend. For his critical suggestions on Chapter 4, I would like to thank Thomas Stibor. I would also like to thank Professor Ping Luo for his useful feedback on Chapter 9 and Chapter 11. I would like to thank Professor Takehisa Yairi for the inspiring discussion on Chapter 7, and Nan Li for her feedback on Chapter 9. I thank Ruei-Bin Wang for his comments on Chapter 11. I thank Yu-Rong Tao for her collaboration and persistent hard work on some experiments in Chapter 9 and Chapter 10.

Many others have helped me over my graduate career. I cannot thank all these individuals enough for their support, but I would like to call attention to a few. I would like to thank Xin-Chang Liu and Cheetah Lin for being good friends who were always willing to listen and provided useful advices. In addition, I would like to thank Petra Lorenz and Alexander Lüdtke for their help and assistance of all kinds to my life and research career in Germany.

Finally, I offer my regards and blessings to all of those, especially my parents, who supported me in any respect during the completion of my dissertation. Without them, this work would not have been possible.

I gratefully acknowledge the support of my sponsors. Part of this work was supported in part by the HIVE (Hypervisorbasierte Innovative VErfahren zur Anomalieerkennung mit Hardwareunterstützung), which receives support from the German Federal Ministry of Education and Research under grants FKZ16BY1200D; and in part by National Science Council, National Taiwan University and Intel Corporation under grants NSC102-2911-I-002-001 and NTU103R7501. I would also like to thank China Scholarship Council for recognizing me the award of outstanding students abroad.

Publications

[1] Han Xiao and Claudia Eckert. Efficient Online Sequence Prediction with Side Information. *IEEE International Conference on Data Mining*, 2013.

[2] Han Xiao and Claudia Eckert. Lazy Gaussian Process Committee for Real-Time Online Regression. *AAAI Conference on Artificial Intelligence*, 2013.

[3] Han Xiao, Huang Xiao and Claudia Eckert. Learning from Multiple Observers with Unknown Expertise. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2013.

[4] Han Xiao, Huang Xiao and Claudia Eckert. Adversarial Label Flips Attack on Support Vector Machines. *European Conference on Artificial Intelligence*, 2012.

[5] Han Xiao and Thomas Stibor. Evasion Attack on Multi-Class Linear Classifier. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2012.

[6] Han Xiao and Thomas Stibor. Supervised Topic Transition Model for Detecting Malicious System Call Sequences. *SIGKDD workshop: Knowledge Discovery, Modeling and Simulation*, 2011. **(Best paper award)**

[7] Han Xiao and Thomas Stibor. Toward Artificial Synesthesia: Linking Pictures and Sounds via Words. *NIPS workshop: Next Generation Computer Vision Challenges*, 2010.

[8] Han Xiao and Thomas Stibor. Efficient Collapsed Gibbs Sampling For Latent Dirichlet Allocation. *Asia Conference on Machine Learning*, 2010.

Contents

List of Figures

List of Tables

Part I.

Introduction

If you know your enemies and know yourself, you can win a hundred battles without a single loss.
If you only know yourself, but not your opponent, you may win or may lose.
If you know neither yourself nor your enemy, you will always endanger yourself.

知彼知己，百戰不殆；
不知彼而知己，一勝一負；
不知彼，不知己，每戰必殆。

Sun Tzu — The Art of War

Chapter 1

Introduction

Building intelligent systems that can adapt to their environments and learn from their past experience has attracted many researchers from different domains, such as computer science, statistic, mathematics, physics, neuroscience and cognitive science. In recent decades, the research in developing learning algorithms has come a wide variety of industrial applications. People have witnessed many successful stories about how modern enterprises relied on machine learning algorithms enjoy great benefits from them. Nowadays, in many large-scale systems, machine learning is considered as a vital tool for data analysis and automatic decision making.

In the community of information security, researchers and engineers of have successfully deployed systems using machine learning and data mining for detecting suspicious activities, filtering spam, recognizing threats, etc. [6, 109]. These systems typically contain a classifier that separates *instances* into two classes, i.e. malicious and benign. Unfortunately, malicious instances that fail to be detected are inevitable for any known classifier. Also, there is an emerging threat that the *adversary* tries to mislead the decision of the classifier by manipulating instances [92, 7]. For example, spammers can add unrelated words, sentences or even paragraphs to the junk mail to avoid the detection of a spam filter [108]. They can also embed the text message in an image. Then, by adding varied background and distorting the image, the generated junk message can be difficult for OCR systems to identify but easy for humans to interpret [68]. Similarly for the host-based intrusion detection, an intruder can obfuscate an attack by inserting *No Operation* instructions or using synonymous system calls to avoid detection [150, 151, 162].

As a reaction to adversarial attempts, previous research has employed a cost-sensitive game theoretic approach to preemptively adapt the decision boundary of a learner by computing the adversary's optimal strategy [43]. Moreover, several improved spam filters that are more effective in adversarial environments have been proposed [68, 16].

Adversary and learner are *Yin* and *Yang* of information security. The car-and-mouse game between them pressures the machine learning researchers to investigate the vulnerability of the current learning algorithms in adversarial environments, which will be the first yet a significant step to improve them in the future. In addition, the improved learning algorithms should be reliable, in the sense that it should be resilient to the adversarial noise in the real-world environment. For instance, a spam filter should learn the correct decision even though the training data is noisily labeled. A recommendation system should be able to reflect the fair rating of a product without being subverted by the biased rating. Furthermore, the learning algorithms should be scalable

enough to handle real-time data. As an example, an intrusion detection system should be able to handle the large volume of data in today's network.

In this dissertation, I investigate both practical and theoretical aspect of adversarial, reliable and scalable machine learning algorithms for security applications and beyond. I summarize the following research problems that are covered in my dissertation.

Adversarial machine learning.

- What are the vulnerabilities of current learning algorithms?
- How can adversaries take advantage of them to design attack algorithm?

Reliable machine learning.

- How does unfaithful training data affects the learning algorithm?
- How can a learner learn from unfaithful training data?
- What are existing and emerging non-security applications where learning techniques can be used to against adversarial data?

Scalable machine learning.

- How to learn from users' behavioral data in real-time?
- How to learn from large-scale of system event logfile?
- How to learn in a client-server setting while minimizing the communication cost?

The answers to the above questions are presented in three parts of this dissertations. In each part, I first give a formulation of the problem, review related work, and propose the theory and methods. Proposed methods are evaluated on different data sets to demonstrate their effectiveness. In the remainder of this chapter, I further motivate the need for developing reliable and scalable machine learning algorithms by providing several high-level examples.

1.1. Motivation

Machine learning owes to its success to the enormous amount of data and to novel decision making algorithms. Spam filter, online advertisement, recommendation systems, consumer profiling, and many other Internet-related businesses crucially depend on machine learning algorithms. Unfortunately, the ubiquity of the Internet has also stimulated its abuse and the rise of sophisticated malicious adversaries. In this section, I will present three examples to show that data-driven technologies are exposed to the adversarial threats.

1.1.1. Spam Filter

Spam filtering is the most popular and successful example of machine learning application. It deals with the adversarial inputs directly. Typically, the automatic spam filtering function consists of an algorithm used to analyze the textual content of email messages. Figure 1.1 illustrates the general workflow of a spam filter.

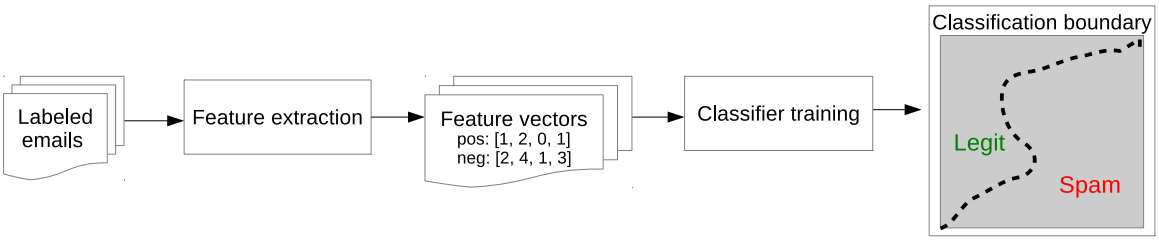


Figure 1.1.: General workflow of a spam filter. It is trained on a set of labeled email messages (containing both spam and non-spam) to construct a classification boundary. When it is deployed, the incoming message is first represented as a feature vector, and then it is mapped to the instance space for determining its label.

During the past fifteen years, spam filtering received much attention in the scientific community. But, as spam filters improved, spammers have also evolved. In the early days, the message body of spam consisted mostly of plain text without any explicit or malicious attempts to evade from detection. Nowadays, spams are carefully disguised to bypass these filters and specialized mimicry attacks are developed. All these efforts make it difficult for the filter to distinguish spam from legitimate emails. Figure 1.3 and Fig. 1.3 show two examples of the real-world spam.

Figure 1.2.: Two English spams. **(top)** The intended title is "this convention in June sincerely wants your attending". The spammer replaces some alphabets to unicode symbols which look similar, or **(bottom)** The intended title is "EIT 2014 ISTP Index".

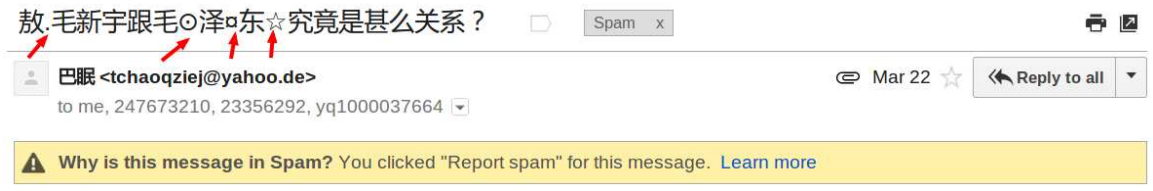


Figure 1.3.: A Chinese spam in which the spammer deliberately adds some alphabet between Chinese characters.

It can be observed from these two examples, that the spammer was trying to evade detec-

tion by adding some "noise" to the messages. Using more technical words, the adversary tries to introduce the feature noise to bypass the filter. This type of attack is called *exploratory attack*. Alternatively, the spammer could also imprint the decision by manipulating the training data of the spam filter, such as by randomly reporting spam as not-spam, or vice versa. By doing that, the adversary introduces the label noise to the training data. This attack is called *causative attack*. Illustrations of these two types of attack are depicted in Fig. 1.4 and Fig. 1.5.

One can clearly see that the assumption of faithful data that traditional machine learning methods based on is no longer valid, which raises the question of whether machine learning methods can be deployed at all in adversarial environments.

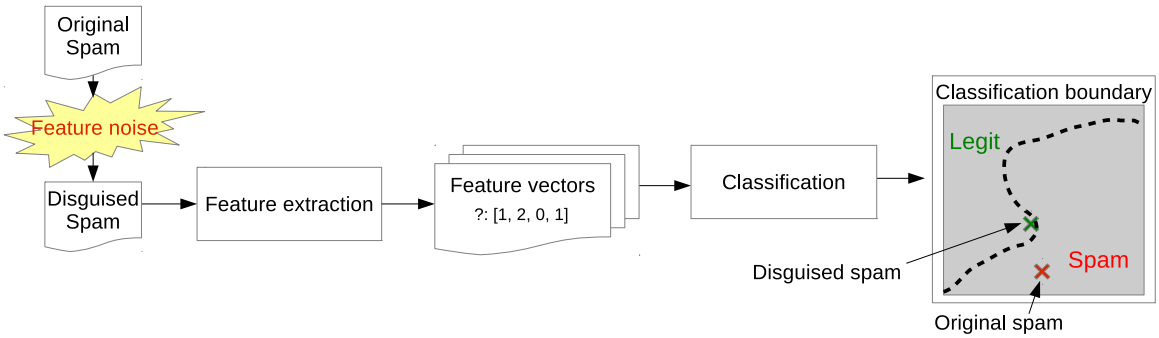


Figure 1.4.: Exploratory attack: introducing feature noise to the original spam. The first two dimensions of the feature vector (highlighted in red) are modified by the adversary. As a consequence, the original spam (red cross) becomes a legit mail (green cross) under the classification boundary.

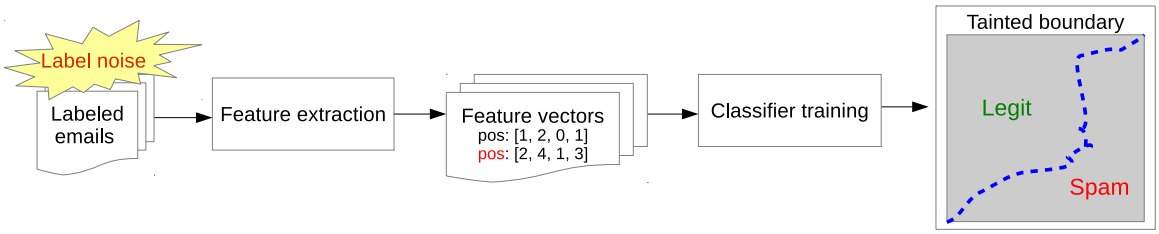


Figure 1.5.: Causative attack: introducing label noise to the training data. After introducing the label noise, the training data is contaminated. The classifier now produces a tainted decision boundary (in blue dashed line). Given this decision boundary, the original spam is no longer classified as a spam.

1.1.2. Social Recommendation Service

With recent developments in social networks, social media websites such as Facebook, Google+, Twitter, YouTube, Amazon use machine learning methods to recommend products, news, photos to meet users' specified interests. To do so, these services usually provide a feedback interface to allow users to submit ratings, post reviews or comments, favor items. This interactive interface is the major target for spammers. Unlike distributing email spams, social spammers try to create fake reviews, ratings and thus manipulate recommendations and rankings.

As an example, on a photo sharing website, each photo is rated by several users. The webmaster can collect those highly ranked photos and display them on the front page of the website. However, adversaries may give unfaithful ratings, either by deliberately "overestimating" or "underestimating" a photo. Adversaries may also target on the photos from a specific photographer to degrade his/her reputation. The problem is how to learn a faithful rating (or groundtruth) for each photo using a set of unfaithful ratings. Figure 1.6 illustrates this problem.

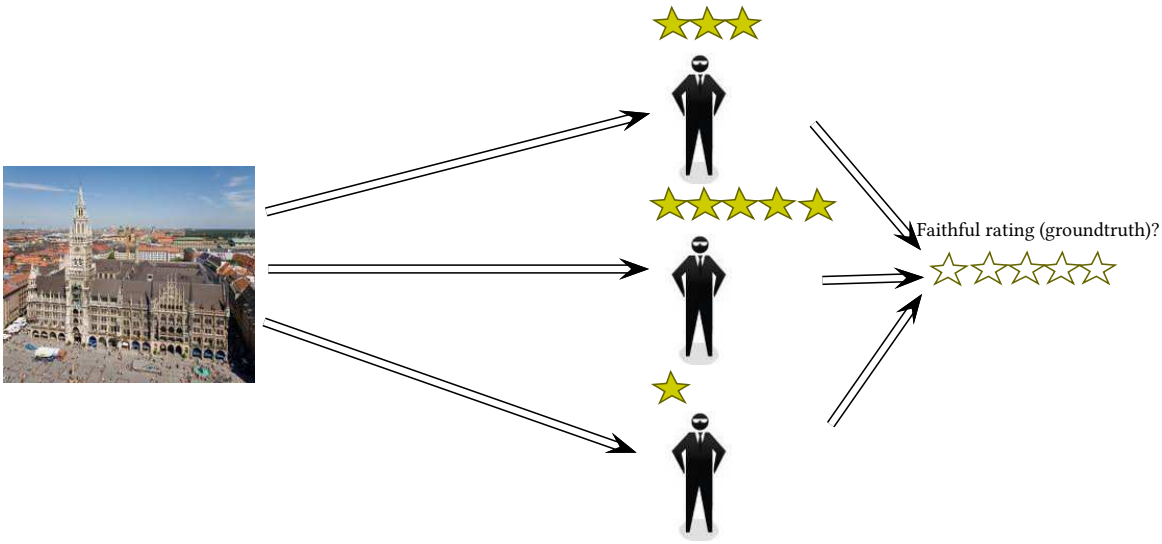


Figure 1.6.: Rating problem in a photo sharing website. Each rating may come from a faithful user or a social spammer who tries to manipulate the ranking in its favor.

The naive approaches such as "take the average" and "majority vote" completely ignore the difference on faithfulness between individuals, makes them inappropriate for solving this problem. Note that, the problem here is more difficult than detecting spams since the adversaries can blur their abuse behavior by giving random ratings. Sometimes it is even difficult for human to distinguish social spammers from normal users. Consequently, groundtruth data is scarce, making it an even more challenging problem.

1.1.3. Real-Time Anomaly Detection with Novel Input

The vast majority of security related applications can be constructed using simple rule-based detection methods. The advantages of using rule-based methods include the high-efficiency and low-cost. They are also intuitive for the domain experts to understand and maintain.

However, rules are not powerful enough to handle novel input samples or perform novel tasks. An example is system call prediction for anomaly detection. A snippet of system call sequence when running `evince` (a PDF viewer on Linux) is given below.

```
11:10:03 fcntl(13, FGETFL) = 0x8002 (flags ORDWR|OLARGEFILE)
11:10:03 fstat(13, -stmode=SIFREG|0644, stsize=0, ...) = 0
11:10:03 mmap(NULL, 4096, PROTREAD|PROTWRITE, MAPPRIVATE
|MAPANONYMOUS,-1,0) = 0x7f2ffa4f7000
11:10:03 lseek(13, 0, SEEKCUR) = 0
```

```
11:10:03 write(13, "¡?xml version=""1.0"" encoding=""UT"...,
159744) = 159744
11:10:03 write(13, "/bookmark:group¿"n ¡/bookm"..., 2772) = 2772
11:10:03 fstatfs(13, -ftype="EXT2SUPERMAGIC", fbsize=4096,
  fblocks=118094150, fbfree=95253262, fbavail=89252750,
  ffiles=30007296, fffree=29036163, ffsid=--298540496,
    -77351758, fnamelen=255, ffrsize=4096) = 0
11:10:03 lstat("/root/.local/share/recently-used.xbel", -
  stmode=SIFREG — 0600,stsize=162516, ...) = 0
11:10:03 fsync(13)                          = 0
11:10:04 close(13)                          = 0
11:10:04 munmap(0x7f2ffa4f7000, 4096)       = 0
11:10:04 rename("/root/.local/share/recently-used.xbel.GM3CEX",
  "/root/.local/share /recently-used.xbel") = 0
11:10:04 chmod("/root/.local/share/recently-used.xbel", 0600) = 0
11:10:04 write(10, ""1"0"0"0"0"0"0"0", 8) = 8
```

In the system call prediction task, the goal is to do real-time prediction of the next system call given all history information. Due to rich but huge amount of information the system events contained, manually inspecting the sequence and summarizing rules are very time-consuming and require profound expertise. Thus, the data-driven method can play a pivotal role here. By modeling previous information, machine learning algorithms can provide confidence intervals for their predictions.

Another example is user behavior prediction. Figure 1.7 depicts three users' mouse trajectory during online banking. The input data can be seen as a real-time two-dimensional data with infinite length. The goal is to learn from users' behavior in order to predict the next mouse movement. As different users tend to behave differently, it may help the security experts to recognize identity theft. Apparently, rule-based approaches are inappropriate in this problem.

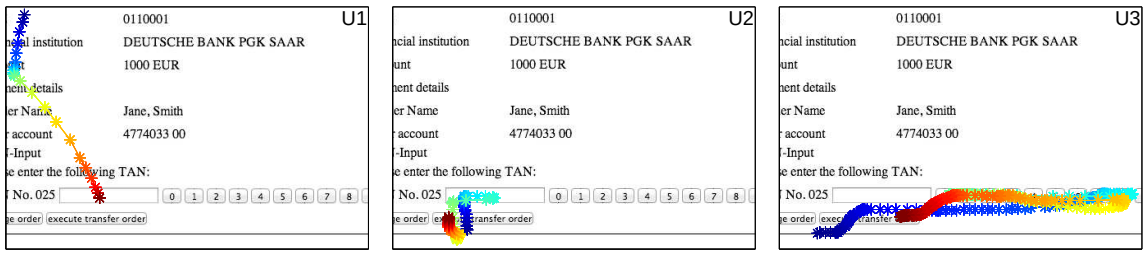


Figure 1.7.: Mouse trajectory from three different users while they are doing online transaction. Each column represents a user. The trajectory is illustrated by the color curve with ⋆, whose head is blue and tail is red.

An important lesson to be learned from these two examples is the necessity for a precise focus on the scalability of the learning algorithm. In the first example, a program could generate thousands of system calls per second. In the second example, an event monitor can capture hundreds of coordinates of the mouse in a second. Therefore, robust and scalable machine learning algorithms are required for efficiently handling large-scale of data in real-time.

1.2. Dissertation Organization

The outline of this dissertation is illustrated in Fig. 1.8. The remainder of this dissertation is organized into four parts. In the first part, I present the background and fundamental materials for this work. In Chapter 2, I briefly introduce preliminary knowledge required for understanding the dissertation. Then in Chapter 3, I introduce the adversarial learning problem. Two types of attack are highlighted, namely the exploratory and causative attack. I will give an simple example of cheating an linear classifier using the exploratory attack. The reader will get the basic idea of how the vulnerability of a machine learning algorithm can be exploited by the adversary.

In the second part, the focus is on analyzing the vulnerabilities of current learning algorithms. Chapter 4 and Chapter 3 study the exploratory attack where an adversary disguises malicious instances as benign by querying the classifier. Specifically, Chapter 4 elaborates the algorithm described in Chapter 3 and shows the effectiveness of the attack algorithm on the multi-class linear classifier. Chapter 5 further studies the exploratory attack algorithm on a broader family of convex-inducing classifiers. Unlike the idea used in Chapter 3 and Chapter 4, I develop a novel attack algorithm based on random walk. Chapter 6 focuses the attention on the causative attack, which aims to degrade the performance of the classifier by manipulating the training data.

In the third part, I investigate the problem of designing reliable machine learning algorithms against adversarial noise. Chapter 7 describes a probabilistic model for regression when there are multiple yet some unreliable observers providing continuous responses. The approach simultaneously learns the regression function and the expertise of each observer, allowing one to predict the ground truth and observers' responses on the new data. In Chapter 8, I focus on an important open problem in the content based image retrieval called *aesthetics assessment*. As image ratings from online communities have the substantial amount of disagreement among users, learning from those biased rating is challenge by its nature. I show that this problem can be solved by using the algorithm described in Chapter 7.

In the fourth part, I explore the large-scale learning problem and its application in security-sensitive domain. Chapter 9 and Chapter 10 focus on reducing the time-cost of the current learning algorithms. In Chapter 9, I present an online Gaussian process model for performing real-time regression tasks. The applicability of the proposed method is demonstrated by the mouse-trajectory prediction in an Internet banking scenario. Chapter 10 describes an efficient algorithm for sequence prediction. The algorithm allows one to incorporate the domain knowledge as side information to improve prediction, which is shown to be beneficial in tracing system call sequences. In Chapter 11, I pose a novel learning problem in client-server settings. Unlike the previous chapters, the goal is to reduce the communication-cost. The problem abstracts a scenario where a server receives potentially unlimited data from clients in a sequential manner, but only a small initial fraction of these data are labeled. Because communication bandwidth is restricted, each client is limited to sending the server only a small (high-priority) fraction of the unlabeled data it generates, and the server is limited in the amount of prioritization hints it sends back to the client. The goal is for the server to learn a good model of all the client data from the labeled and unlabeled data it receives.

In the final chapter, I conclude with a summary of the contributions of this dissertation and discuss important themes and open questions for the field of reliable and scalable learning in security-sensitive domains. Below, I outline the primary contributions I make in this dissertation.

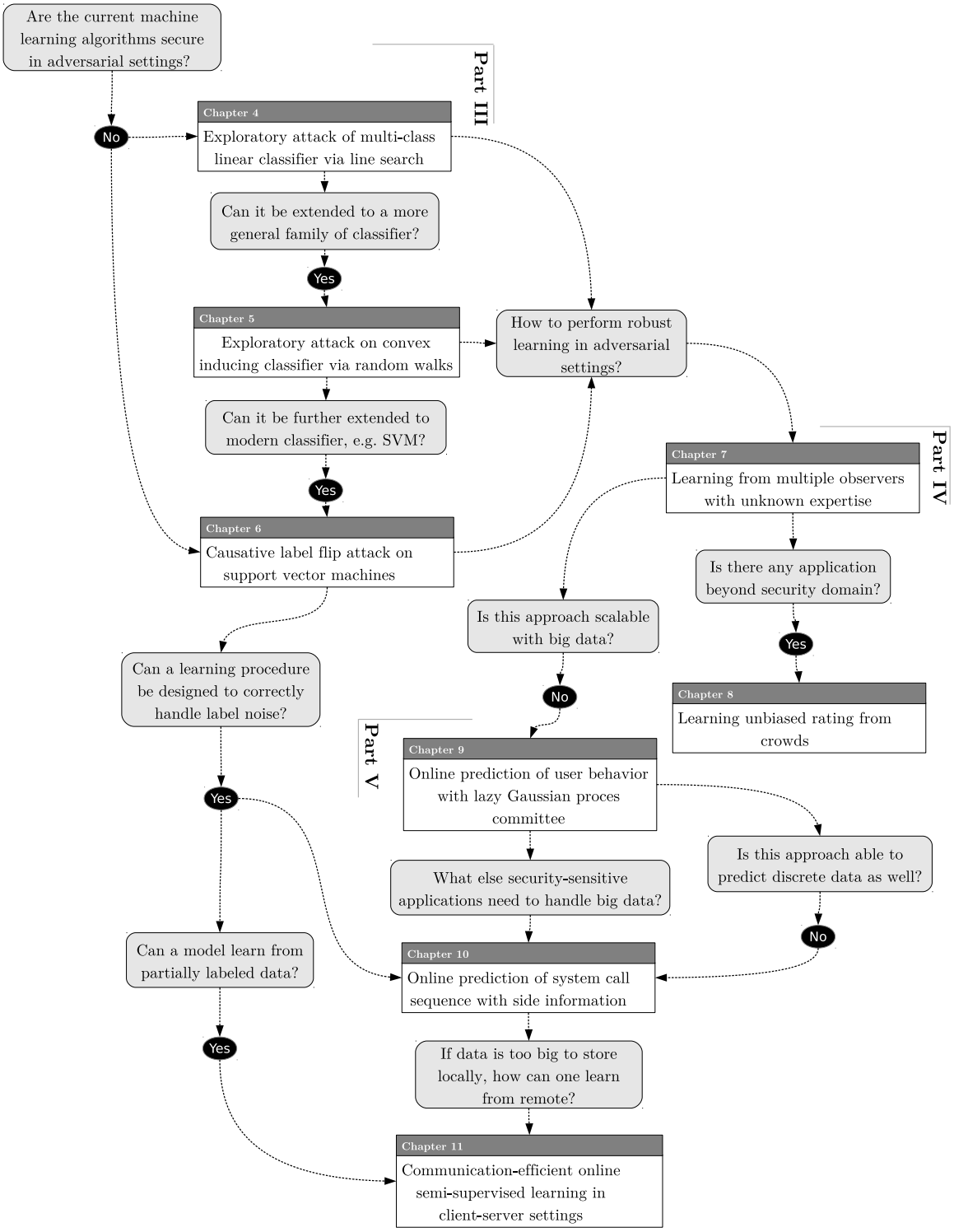


Figure 1.8.: The outline of this dissertation. Chapters are driven by a series of questions.

1.3. Contributions

In this dissertation, I examine a number of machine learning algorithms, assess their vulnerabilities, demonstrate real-world attacks against their learning mechanisms, and purpose several novel algorithms that can perform reliable and scalable learning in the real-world scenarios. Moreover, I provide the machine learning practitioners with a systematic methodology for developing robust and efficient machine learning systems for the security-sensitive applications. Furthermore, I also examine and answer theoretical questions about the adversarial strategies. This dissertation aims to bring together the interests from both the computer security and machine learning communities. The contributions of this thesis are summarized in the following paragraphs.

Identifying Vulnerabilities of Algorithms and Adversarial Capabilities The first major contribution of this dissertation is the theoretical analysis of the vulnerabilities. Traditional machine learning research were originally conceived under the assumption of faithful data and did not explicitly account for potential data manipulation by adversaries. I raise the attention of researcher about the vulnerabilities of some learning algorithms. I identify two types of adversaries' strategies, namely the exploratory attack and causative attack. For each type of attack, I provide a theoretical analysis using the knowledge of geometry analysis, optimization theory and probabilistic theory, and design attack algorithms that can be easily implemented in practice. By demonstrating the devastating impact of the attack algorithms on a newsletter classifier and a face recognition system, I underline the importance of reliable learning for the machine learning researchers and security analysts, which plays an essential role in motivating my next contribution.

Presenting Reliable Algorithm Resilient to Adversaries The second principal contribution I make in this dissertation is developing novel reliable learning algorithm for the security-sensitive environment. I formulate the problem of learning from multiple observers, where each instance is associated with multiple but unreliable labels. The designed algorithm is resilient to adversarial contamination in the data. The application of this method has a wide range of domains, from crowdsourcing platforms, photo rating websites to sensor networks. The successful application of my method on the task of aesthetics score assessment may raise practitioners great interest. The message sent from my dissertation is that, instead of using simple heuristics such as "take the average" and "majority vote", one can still learn from unreliable sources provided that the adversarial noise is appropriately modeled.

Presenting Online Algorithms for Large-Scale Data Stream My third contribution is the design efficient online learning algorithms for handling large-scale data. My online algorithms cover both the regression and classification task. Specifically, I point out the low efficiency of the original Gaussian process model and the Markov sequence predictor. Then, I provide approximated learning algorithms for improving the efficiency. I apply these algorithms in real-world security applications where the speed of prediction is the primary concern, such as Internet banking and system call tracing.

Establishing Distributed Learning Framework for Client-Server Settings My fourth contribution is proposing a novel learning problem for the client-server design. In particular, I consider a scenario where a distributed system consists of clients, a server, and a communication network.

The clients submit partially labeled training data to the server. The server learns a model from incoming data. I call this new setting as communication-efficient learning, which aims to reduce the communication cost over the network. This problem abstracts a common scenario where training data is too big to be stored locally and labeled completely. I show that some particular combination of techniques outperforms other approaches, and in particular, often outperforms (communication expensive) approaches that send all the data to the server.

Part II.

Background

I confess that I've been blind as a mole. But it's better to learn wisdom late than never know it at all.

Sherlock Holmes

Chapter 2

Preliminary Knowledge

I briefly introduce preliminary knowledge required for understanding this dissertation. Most of the content in this chapter are basic machine learning techniques that this dissertation builds upon. For a more thorough literature of machine learning, the reader should refer to a book such as [13].

2.1. Machine Learning

The problem of searching for patterns in data is a fundamental one and has a long and successful history. Machine learning is concerned with a vast field of techniques that automatically discover regularities in data. With the use of these regularities, people can take actions such as classifying the data into different categories for recognizing face or filtering spam.

For most practical applications, applying a learning algorithm directly to real-world objects is difficult because the learner cannot understand the structure of the object. Thus, the original input objects are processed to transform them into some new space of representative observations. This processing step is called *feature extraction*. The *feature* of an object represents the learner's view of the real world.

Typical application of machine learning is to predict unobserved state of the world based on its observed state. For example, given an email determine whether it is a spam or not. In this dissertation, I refer to the identity of the corresponding object as *label*. Note that the label can be either discrete or continuous, depending on the context of the application. In face recognition, the label could be the name of a face, or the gender of a face. It could also be the age of a face. In particular, if the label is discrete, the task is called *classification*. If the desired label consists of one or more continuous variables, then the task is called *regression*.

When the training data contains feature representations along with their corresponding labels, the problem is known as *supervised learning*. When the training data does not consist of any label, the problem is often known as *unsupervised learning*. For example, in the domain of text mining, the goal is to cluster those web pages that have similar topics. In other problem, the training data can be partially labeled. The learner is expected to learn from both labeled and unlabeled data simultaneously. This task is known as *semi-supervised learning*.

Each of the above tasks needs its own analyses and techniques, however, many of the key ideas are shared among all such problems. In the remaining of this chapter, I will provide a self-contained introduction to the important topics related to this dissertation, namely supervised learning, semi-

supervised learning, classification, regression, online learning and active learning. I will also cover some popular techniques in different learning settings, such as support vector machines and Gaussian process regression.

2.2. Supervised Learning

Supervised learning is a machine learning task of inferring a function from labeled training data. The name invokes the idea of a "supervisor" that instructs the learning system on the labels to associate with training examples.

Given a set of n training examples of the form $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is the feature vector of the i^{th} example and y_i is its label. The supervised learning task is to find a function $g : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. The function g is also called a hypothesis from a hypothesis space $\mathcal{H}$. Given an instance $\mathbf{x} \in \mathcal{X}$, the classification decision can thus be made according to the sign of $g(\mathbf{x})$.

In order to measure how well a function fits the training data, a loss function $V : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is defined. For a training example $(\mathbf{x}_i, y_i)$, the loss of predicting the value $\widehat{y}$ is $V(y_i, \widehat{y})$. Formally, the goal to find a classification hypothesis $g \in \mathcal{H}$ can be solved by the following optimization problem

$$g^* := \arg\min_g \gamma \sum_{i=1}^{n} V\left(y_i, g(\mathbf{x}_i)\right) + \|g\|_{\mathcal{H}}^2, \tag{2.1}$$

where γ is a fixed positive parameter for quantifying the trade off. Remark that the first term in Eq. (2.1) reflects the empirical loss of g on the training set, and the second term reflects the generalization ability of g.

2.2.1. Support Vector Machine

Support vector machine (SVM) is a supervised learning model for classification and regression analysis. SVM has a nonparametric nature, due to the fact that the model is represented by a set of training examples. The examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Formally, SVM projects the original training instances from the input space $\mathcal{X}$ to the *feature space* $\mathcal{F}$ by $\Phi : \mathcal{X} \to \mathcal{F}$. In general, SVM trained on a data set S has the form

$$f_S(\mathbf{x}) := \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b,$$

where K is a *Mercer Kernel* which satisfies the property $K(\mathbf{x}, \mathbf{x}_i) = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}_i)$ and $b \in \mathbb{R}$ denotes the bias. The classifier can be also rewritten as

$$f_S(\mathbf{x}) := \mathbf{w}^\top \mathbf{x} + b,$$

where $\mathbf{w} := \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i)$ and $\mathbf{w} \in \mathcal{F}$. Thus, the classification boundary of a SVM is a hyperplane in $\mathcal{F}$ with normal vector $\mathbf{w}$. Given the *hinge loss* function $V\left(y, f(\mathbf{x})\right) := \max(0, 1 - yf(\mathbf{x}))$,

Tikhonov regularization for a SVM is a constrained *quadratic programming* (QP) problem

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \gamma \sum_{i=1}^{n} \xi_i + \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n, \end{aligned} \tag{2.2}$$

where ξ_i represents the hinge loss of $(\mathbf{x}_i, y_i)$ resulting from the classifier f_S.

2.2.2. Gaussian Process Regression

While SVMs are frequently used in classification tasks, the Gaussian Process is often preferable in regression tasks. As in the supervised learning setting, the problem of regression aims to find a function estimation from the given labeled data. It is usually formulated as follows: given a training set $\mathcal{D} \coloneqq \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ of N pairs of input vectors $\mathbf{x}_n$ and noisy scalar outputs y_n, the goal is to learn a function f transforming an input into the output given by

$$y_n = f(\mathbf{x}_n) + \epsilon_n,$$

where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ and σ^2 is the variance of the noise. A *Gaussian process* is a collection of random variables, any finite number of which have consistent joint Gaussian distribution. *Gaussian process regression* (GPR) is a Bayesian approach which assumes a GP prior over functions. As a result the observed outputs behave according to

$$p(\mathbf{y} \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \mathcal{N}(\mathbf{0}, \mathbf{K}),$$

where $\mathbf{y} \coloneqq [y_1, \ldots, y_N]^\top$ is a vector of output values, and $\mathbf{K}$ is an $N \times N$ covariance matrix; the entries are given by a covariance function, i.e. $K_{ij} \coloneqq k(\mathbf{x}_i, \mathbf{x}_j)$. In this work, I consider a frequently used covariance function given by

$$k(\mathbf{x}_i, \mathbf{x}_j) \coloneqq \kappa^2 \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)\right) + \sigma^2 \delta_{ij}, \tag{2.3}$$

where κ denotes the signal variance and $\mathbf{W}$ are the widths of the Gaussian kernel. The last term represents an additive Gaussian noise, i.e. $\delta_{ij} \coloneqq 1$ if $i = j$, otherwise $\delta_{ij} \coloneqq 0$.

In the setting of probabilistic regression, the goal is to find a predictive distribution of the output y_* at a test point $\mathbf{x}_*$. Under GPR, the predictive distribution of y_* conditional on the training set $\mathcal{D}$ is also Gaussian

$$p(y_* \,|\, \mathcal{D}, \mathbf{x}_*) = \mathcal{N}\left(\mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{y}, k_* - \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{k}_*\right), \tag{2.4}$$

where $\mathbf{k}_* \coloneqq [k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_N)]^\top$ and $k_* \coloneqq k(\mathbf{x}_*, \mathbf{x}_*)$. One can observe that the training data is explicitly required at the test time in order to construct the predictive distribution, which makes GP a non-parametric method. The hyperparameters are $[\kappa^2, \sigma^2, \{\mathbf{W}\}]^\top$, where $\{\mathbf{W}\}$ denotes parameters in the width matrix $\mathbf{W}$. The optimal hyperparameters for a particular data set can be derived by maximizing the marginal likelihood function using a gradient based optimizer. For a more detailed background on GP, readers are referred to a textbook [129].

2.3. Online Learning

One of the extensions of traditional supervised learning is online learning. Online learning takes place in a sequence of consecutive rounds. On each round, the learner is given a question and is required to provide an answer to this question. The performance of an online learning algorithm is measured by the cumulative loss suffered by the prediction along the run on a sequence of question-answer pairs. Formally, let $(\mathbf{x}^{\langle t \rangle}, y^{\langle t \rangle})$ be the t^{th} example in a sequence. On round t, the algorithm first predicts the label of $\mathbf{x}^{\langle t \rangle}$ according to its current prediction rule. After that, the true symbol $y^{\langle t \rangle}$ is revealed and the algorithm suffers a loss, which reflects the degree to which its prediction was wrong. The algorithm then has the option to modify its prediction rule, with the explicit goal of improving the accuracy of its predictions for the rounds to come.

2.3.1. Passive-Aggressive Algorithm

Passive-Aggressive algorithm is a family of margin based online learning algorithm for various prediction tasks. The algorithm keeps updating the weight vector $\mathbf{w}$ and changes the classification hypothesis over time. Consider the binary classification problem as an example. The weight vector $\mathbf{w}^{\langle 1 \rangle}$ is initialized to $(0, \ldots, 0)$. On round t, the algorithm sets the new weight vector $\mathbf{w}^{\langle t+1 \rangle}$ to be the solution to the following constrained optimization problem,

$$\begin{aligned} \mathbf{w}_{t+1} &= \arg\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2, \\ \text{s.t.} \qquad & V(\mathbf{w}; (\mathbf{x}^{\langle t \rangle}, y^{\langle t \rangle})) = 0, \end{aligned}$$

where the loss V is defined by the following hinge-loss function,

$$V(y^{\langle t \rangle}, \mathbf{w}^\top \mathbf{x}^{\langle t \rangle}) \coloneqq \max(0, 1 - y^{\langle t \rangle} \mathbf{w}^\top \mathbf{x}^{\langle t \rangle}).$$

The resulting algorithm is passive whenever the hinge-loss is zero. In contrast, on those rounds where the loss is positive, the algorithm aggressively forces $\mathbf{w}^{\langle t+1 \rangle}$ to satisfy the zero-loss constraint.

2.4. Semi-Supervised Learning

Semi-supervised learning studies learning from both labeled and unlabeled examples. This learning setting focuses the real-world problems, where data is abundant the cost to label them is expensive. Because semi-supervised learning requires less human effort, it is of great interest both in theory and in practice. Some often-used methods include: EM with generative mixture models, self-training, co-training, transductive support vector machines and graph-based methods.

2.5. Active Learning

Active learning is a special case of semi-supervised machine learning in which a learning algorithm is able to interactively query an oracle to obtain the desired outputs at new data points. There are situations in which unlabeled data is abundant but manually labeling is expensive. In such a scenario, learning algorithms can actively query the oracle for labels. By carefully choosing the

examples, the number of examples to learn a concept can often be much lower than the number required in traditional supervised learning.

Formally, let T be the total set of all data under consideration. During each iteration i, the set T is broken up into three subsets:

- $T_{\mathrm{k}}^{\langle i \rangle}$: Data points where the label is known.
- $T_{\mathrm{u}}^{\langle i \rangle}$: Data points where the label is unknown.
- $T_{\mathrm{c}}^{\langle i \rangle}$: A subset of $T_{\mathrm{u}}^{\langle i \rangle}$ that is chosen to be labeled.

Most of the research in active learning involves the method to choose the optimal data points for $T_{\mathrm{c}}^{\langle i \rangle}$.

I have thus far briefly reviewed the basic concepts of machine learning. These concepts will be mentioned, used, extended and combined in my dissertation. For instance, in Chapter 7 I study the supervised learning framework and design an attack algorithm against SVM. In Chapter 9, Gaussian process regression is extended to the online setting for predicting users' behavior in real-time. In Chapter 11, online learning, active learning and semi-supervised learning are combined together for solving a novel distributed learning problem.

Chapter 3

Adversarial Machine Learning

Having reviewed some basic concepts of machine learning, I now introduce the problem of adversarial learning in this chapter. The problem of adversarial learning focuses on the vulnerability of machine learning algorithms in adversarial environments.

In general, the attack of a learning algorithm can be categorized into the following types by the adversarial purpose.

Exploratory attack The adversary discovers blind spots for malicious instances by querying the classifier.

Causative attack The adversary subverts the learning process of the classifier by manipulating the training data.

Figure 3.1 illustrates these two types of attack. For example, in an exploratory attack, a spammer can disguise a spam by adding unrelated words to evade the detection [107, 108, 173]. In a causative attack, the adversary flags every legitimate mail as spam to pollute the training data. Consequently, the spam filter trained on such data is likely to cause a false alarm and block all legitimate mails [120, 116]. Chapter 1 already showed some examples of adversarial attacks in real-world.

As a start, I will describe a simple exploratory attack problem on a linear classifier. The idea is to show the reader the idea of adversarial learning, and how the vulnerability (or "blind spots") of a learning algorithm can be used to attack itself. A more comprehensive study of the causative attack is presented in Chapter 6.

3.1. Problem Definition

Define $\mathbf{x} \in \mathcal{X}$ as an instance, where $\mathcal{X}$ is the instance space. $\mathbf{x}$ is represented by a vector variable with n dimensions, namely $\mathbf{x} = (x_1, \ldots, x_n)$. Denote x_i the i^{th} feature of the instance $\mathbf{x}$. Each instance can belong to one of two classes: positive (malicious) and negative (innocent), which are denoted by $\mathcal{X}^+$ and $\mathcal{X}^-$ respectively. Let the training set $\mathcal{S} \subset \mathcal{X}$ and the test set $\mathcal{T} \subset \mathcal{X}$ consist of both positive and negative instances.

I call a function $f : \mathcal{X} \to \{-1, 1\}$ as a *Boolean classifier*, or a *classifier* for short. I refer to $\mathcal{X}^+$ for which $f(x) = 1$ and $\mathcal{X}^-$ for which $f(x) = -1$. The goal of a classifier is to learn from $\mathcal{S}$ a

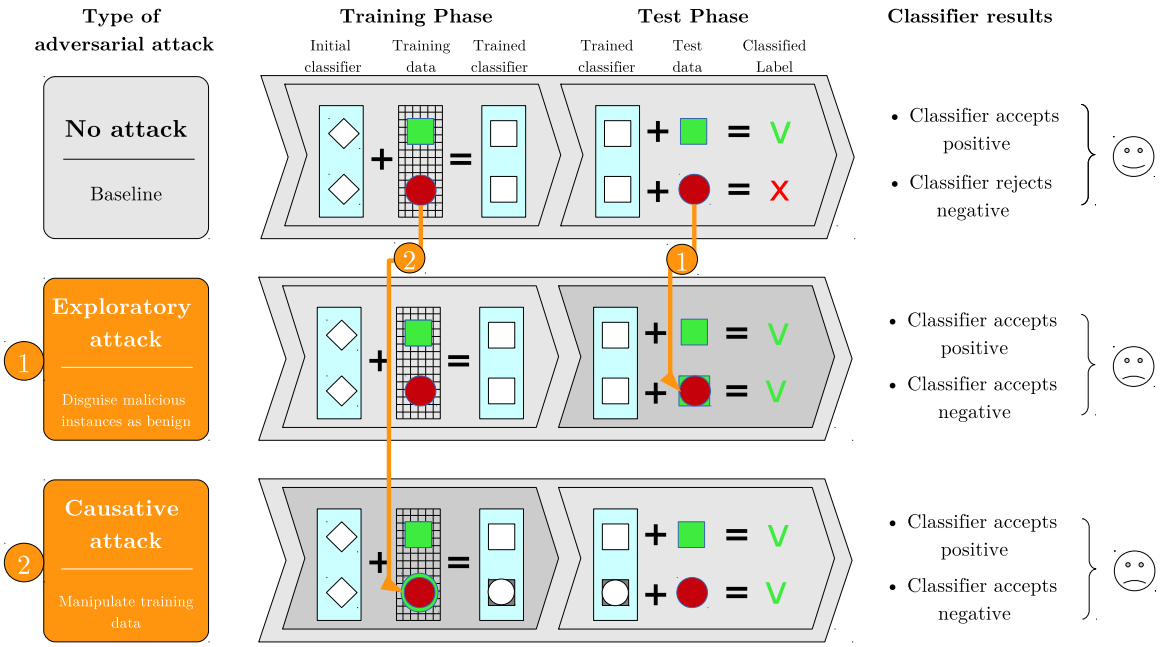


Figure 3.1.: Different types of adversarial attack on learning algorithms.

function $f(\mathbf{x})$ that can correctly predict new instances from T. Many successful security-sensitive applications rely on a well-performed classifier.

An *adversary* attempts to send malicious instances to the system while bypassing the classifier. For instance, a spammer can add "good" words or sentences to cheat the spam filter by decreasing the likelihood of detection. For adversary, some modifications to the spam are more cost-effective than others. I explain such differences on utility by an *adversarial cost function* $a(\mathbf{x}) \rightarrow \mathbb{R}^+$. It is assumed that adversaries have a base instance $\mathbf{x}^{\mathrm{a}}$ for which $f(\mathbf{x}^{\mathrm{a}}) = 1$ on hand. To evade detection, the adversary is interested in finding an instance $\mathbf{x}^*$ that most similar to $\mathbf{x}^{\mathrm{a}}$ but will be classified as $\mathcal{X}^-$. To measure the similarity between two instances, I define the adversarial cost function as

$$a(\mathbf{x}) = \sum_{i=1}^{n} a_i |x_i - x_i^{\mathrm{a}}|.$$

Note that, $a(\mathbf{x})$ is domain-dependent function. The positive scalars a_i represent the relative cost of changing each feature, allowing that some features may be more important than others (from adversaries' perspective of view)[1]. An illustrative example is depicted in Figure 3.2.

Finally, the task of finding $\mathbf{x}^*$ is formulated as

$$\mathbf{x}^* = \arg \min_{\mathbf{x}: f(\mathbf{x})=-1} a(\mathbf{x}).$$

In other words, the adversary searches for $\mathbf{x}^*$ by repetitively sending instances to the classifier f. For each instance, the adversary can observe its classification label.

[1]The positive scalar a_i is under an assumption that $\mathbf{x}^{\mathrm{a}}$ is the best instance as far as the adversary knows. That is, any changes to $\mathbf{x}^{\mathrm{a}}$ costs an utility loss.

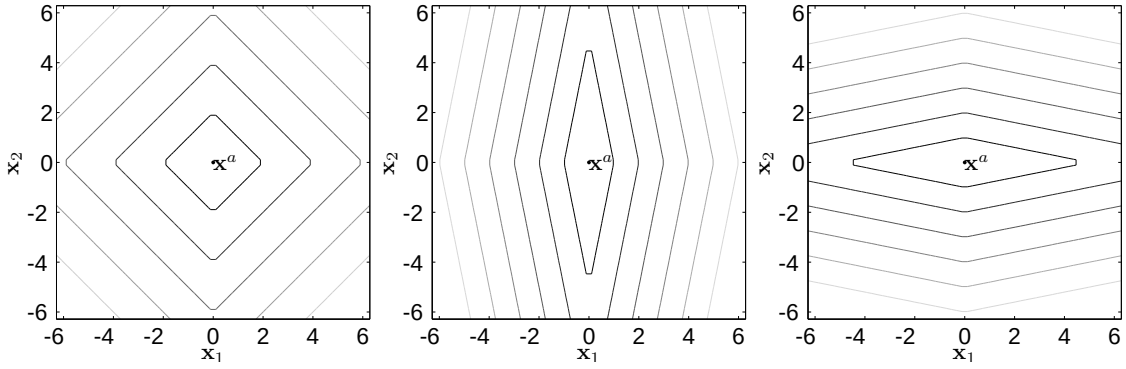


Figure 3.2.: Contour plots of linear cost function $a(\mathbf{x})$ with different value of a_i and $\mathbf{x}^{\mathrm{a}} = (0, 0)$ **(left)** an uniform linear cost function $|x_1| + |x_2|$, where both a_1 and a_2 are one. **(middle)** $5|x_1| + |x_2|$, where $a_1 = 5$ and $a_2 = 1$. **(right)** $|x_1| + 5|x_2|$, where $a_1 = 1$ and $a_2 = 5$.

3.2. A Case Study: Evading a Linear Classifier

Linear classification has become one of the most promising learning techniques for large sparse data with a huge number of instances and features. A linear classifier generates a weight vector $\mathbf{w}$ as the model. The decision function is

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^T\mathbf{x}).$$

In this section, I shall demonstrate an algorithm for evading a linear classifier using the adversarial cost function described in Section 3.1.

Considering the case in which all features are continuous. Intuitively, the adversary would like to search $\mathbf{x}^*$ in a direction such that he/she can quickly reach the boundary (large $|w_i|$) with minimum cost (small a_i). Thus, define a feature g with highest weight-to-cost ratio as

$$g = \arg \max_{i \in \{1,\cdots,n\}} \frac{|w_i|}{a_i}. \tag{3.1}$$

It can be shown that, for a linear classifier with continuous feature space , $\mathbf{x}^*$ can be found by changing only feature g in $\mathbf{x}^{\mathrm{a}}$. Let $\widehat{g}$ be the unit vector along dimension g, the instance of minimal adversarial cost is given by

$$\mathbf{x}^* = \mathbf{x}^{\mathrm{a}} + t\widehat{g}, \tag{3.2}$$

where t is the step length. Figure 3.3 shows two illustrative examples.

3.2.1. IMAC Algorithm

I have shown that $\mathbf{x}^*$ can be easily found by changing the value of x_g. However, in order to determine g one needs to first approximate the value of $\mathbf{w}$. Authors in [107] proposed an efficient algorithm for determining g and finding *I*nstance with *M*inimum *A*dversarial *C*ost (IMAC) $\mathbf{x}^*$. Specifically, given a positive instance $\mathbf{x}^+$ and a negative instance $\mathbf{x}^-$, the complete process is stated as follows:

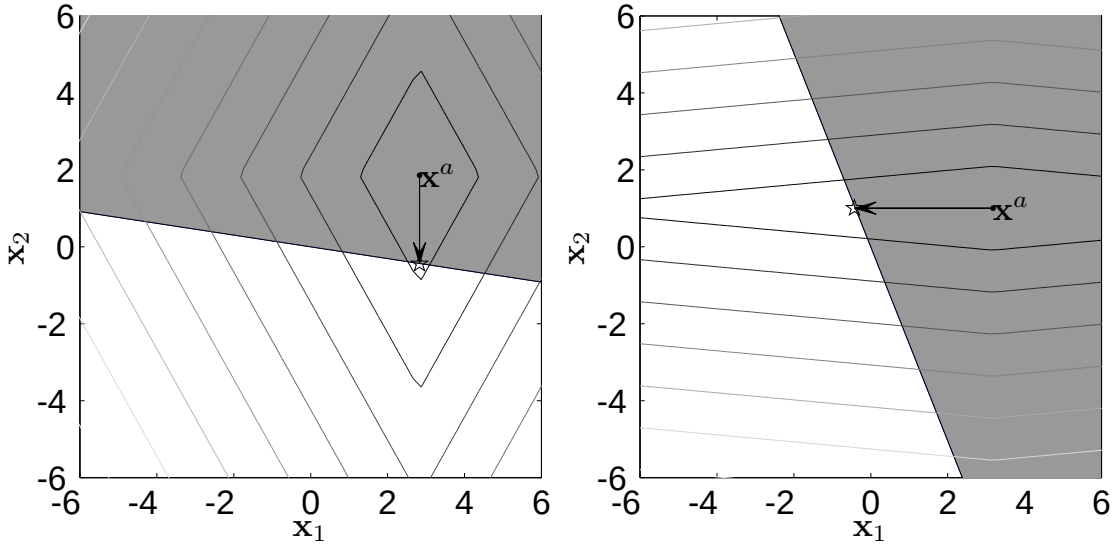


Figure 3.3.: Searching $\mathbf{x}^*$ (represented by a star) in a 2-dimensional space with a linear decision boundary. The shaded area represents the positive response of the classifier. The arrow follows the optimal searching direction. **(left)** The optimal searching direction is along x_2. **(right)** The optimal searching direction is along x_1.

1. Finding a pair of instances $\mathbf{s}^+, \mathbf{s}^-$ such that $\exists i \ \forall j \neq i, s_j^+ = s_j^-$ using $\mathbf{x}^+$ and $\mathbf{x}^-$.
2. Assessing the sign of w_i and letting $w_i = \text{sgn}(w_i)$.
3. Searching a negative instance close to the linear decision boundary.
4. Searching in every other dimensions $j \neq i$ and computing the relative value of weight w_j.
5. Computing g by (3.1) and finding $\mathbf{x}^*$ by (3.2).

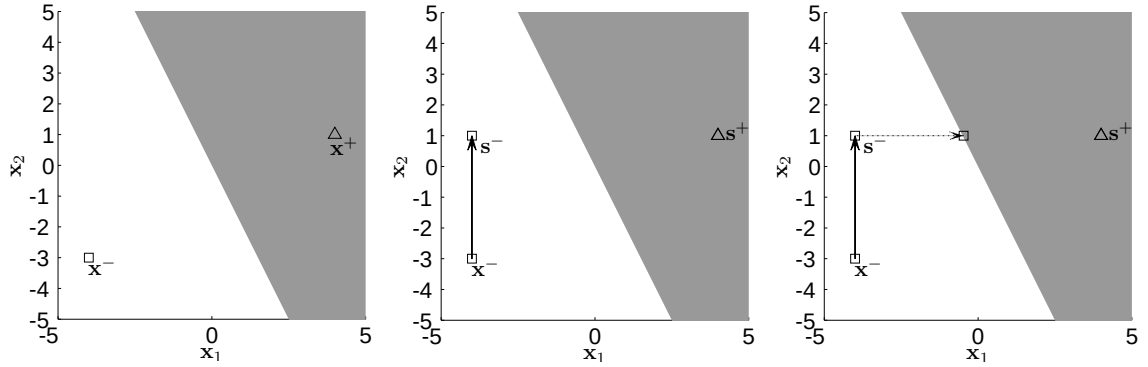
To intuitively demonstrate each step of the algorithm, an illustrative example on a 2-dimensional space is depicted in Fig. 3.4. For a linear classifier with continuous features and a linear cost function, this algorithm requires at most polynomially many queries for each step [107]. With a learned weight on hand, the adversary is now able to compute g by (3.1) and find optimal $\mathbf{x}^*$ by (3.2) using simple line search techniques.

Note that the algorithm can be easily scale on high-dimensional data.

Alternatively, one may consider to train an approximated linear classifier with queried instances and their corresponding returned labels. However, this approach does not consider the fact that, the data space is biased in practice. That is, if one just pick random points in the space, then all of them might end up being positive, or all of them might end up being negative. In this case, the resulting queried instances would be fairly uninformative.

3.2.2. Experiments

I implemented IMAC algorithm and evaluated its performance on a) the number of queries for approximating the weight; b) the adversarial cost of $\mathbf{x}^*$ given x^{a}. The number of queries for finding



(a) Initializing $\mathbf{x}^+$ and $\mathbf{x}^-$.

(b) Searching $\mathbf{s}^+, \mathbf{s}^-$ to assess the sign of w_1.

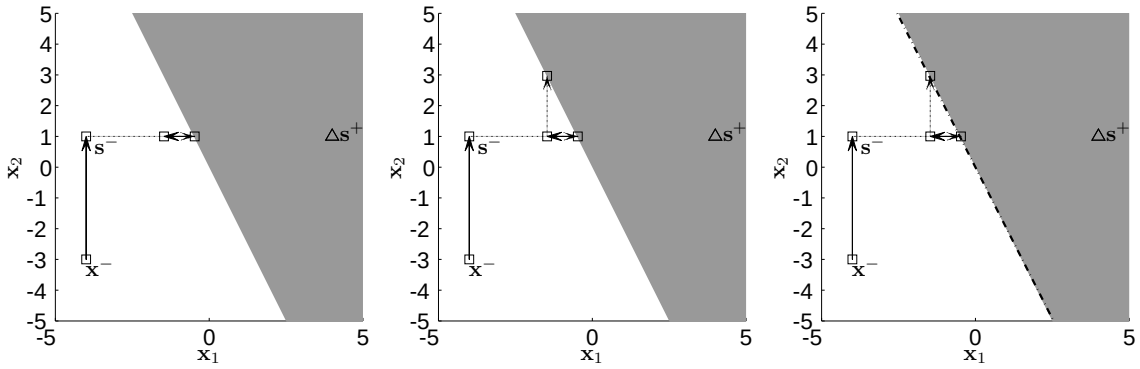
(c) Searching a negative instance close to the boundary.

Figure 3.4.: Algorithm proposed in [107] for approximating $\mathbf{w}$. **(a)** Assume the adversary has a positive instance $\mathbf{x}^+$ (denoted by a triangle) and a negative instance (denoted by a square) on hand, but has no idea about the linear decision boundary. The shaded area represents the positive response of a linear classifier. **(b)** The algorithm starts with $\mathbf{x}^+$ and changes feature values one at a time to match those of $\mathbf{x}^-$. At some point, the class of instance must change. The previous value and the current value of intermediate instance are set to $\mathbf{s}^-$ and $\mathbf{s}^+$, respectively. This step requires at most n test queries. **(c)** As $\forall j \neq i, s_j^+ = s_j^-$, a binary search along the dimension i will find a negative instance close to the decision boundary. Let ϵ be an approximation threshold, this step requires $O(\log(1/\epsilon + |s_i^+ - s_i^-|))$ test queries. The dotted line represents the line search operation. **(d)** The algorithm sets w_i to 1 or -1, and increases or decreases x_i by 1 until a negative instance is found. **(e)** The algorithm proceeds by searching in every other directions $j \neq i$ using a line search. This consists of increasing or decreasing each x_j exponentially until the class of $\mathbf{x}$ changes, and then bounding its exact value with a binary search. **(f)** Finally, the approximated $\mathbf{w}$ can be computed with the tangent rule. The dashed line shows the learned weight, which is almost identical to the ground-truth. The adversary is now able to compute g by (3.1) and find optimal $\mathbf{x}^*$ by using (3.2).

$\mathbf{x}^{\mathrm{a}}$ was not in the evaluation as it depends on the data. The target was a linear classifier with the form of $f(x) = \mathbf{w}\mathbf{x} - T$, where $\mathbf{w}$ and T is randomly generated. Note that the adversary did not have access to the original training data, but must come up with all of the queries itself. Thus, only a positive and a negative instance were used as the input of IMAC. To keep the setup as realistic as possible, I set the initial guess of feature set for adversary 10 times larger than the classifier virtually used. The size of feature set varied from 1000 to 10000. Each experiment was repeated for 10 times. Figure 3.5 shows the experimental result.

It can be observed from Fig. 3.5 that the performance of IMAC algorithm is stable as the number of features increasing. The algorithm can efficiently find low cost instances. Not surprisingly, fewer queries were required to sort through a smaller feature sets. Note, that in practice the adversary needs to keep down not only the number of total queries but also the number of positive queries. A salient rising of positive queries increases the likelihood of being detected.

To compare with the naive "reverse engineering" approach, I also train a perceptron linear

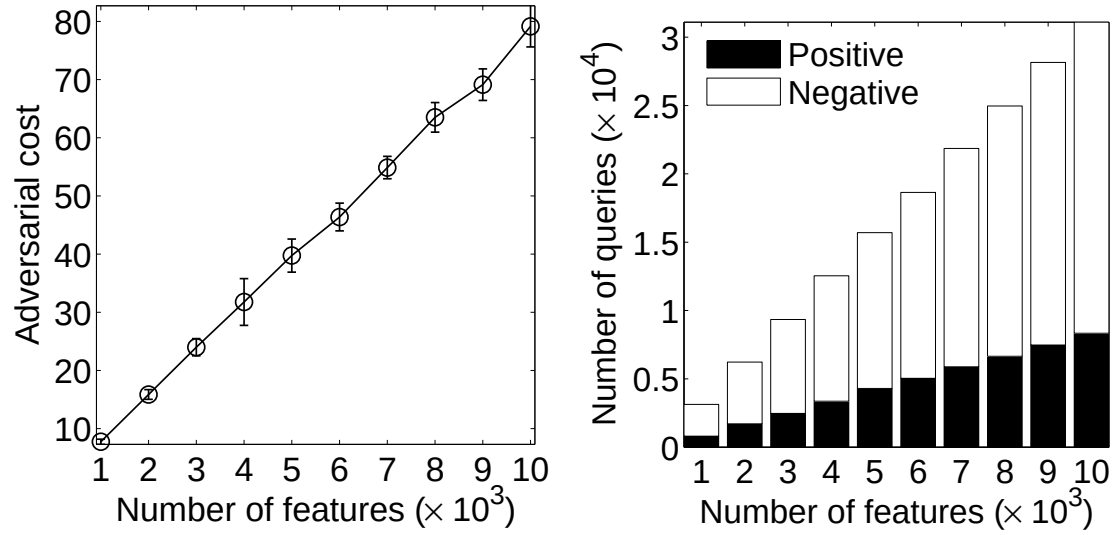


(a) Moving one unit away from the boundary.

(b) Searching in another dimension.

(c) Approximating $\mathbf{w}$ via tangent rule.

Figure 3.4. (cont.)



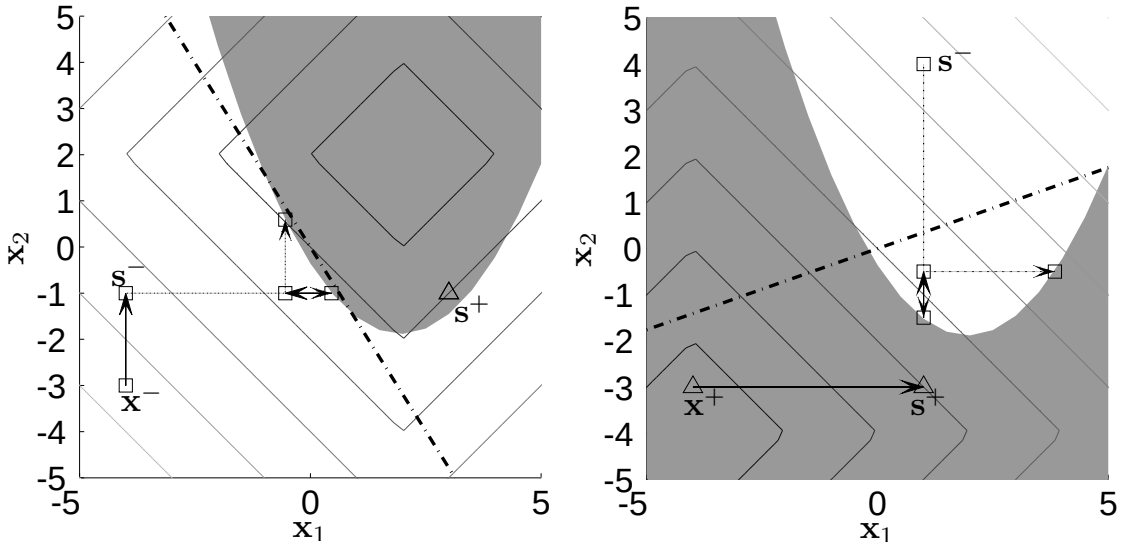
(d) Adversarial cost of $\mathbf{x}^{\mathrm{a}}$.

(e) Number of queries used for finding $\mathbf{x}^{\mathrm{a}}$.

Figure 3.5.: Result of IMAC algorithm on a random generated linear classifier. **(a)** Adversarial cost measure the distance between $\mathbf{x}^*$ and $\mathbf{x}^{\mathrm{a}}$. Lower cost is better. **(b)** The number of queries used for determining the weights. Bars shaded with black indicates the number of positive queries. In practice, it is important for an adversary to keep not only the number of total queries down, but also the number of positive queries.

classifier by randomly picking data points in space. The corresponding class labels are determined by querying $f(x)$. The training data also includes at least one positive instance: $\mathbf{x}^{\mathrm{a}}$. Unfortunately, as the dimension of feature space increasing, it become more difficult for the perceptron to approximate the target classifier. In fact, when the dimension of the feature space is greater than 500, the perceptron is unable to find any optimal instance, due to the fact that most of training instances fall into the negative class.

Though IMAC algorithm performs well on evading the linear classifier, it is not applicable to classifiers with a nonlinear decision boundary. An illustrative example of a common failure with nonlinear decision boundary is illustrated in Figure 3.6.



(a) Positive convexity.

(b) Negative convexity.

Figure 3.6.: Examples of IMAC algorithm on classifiers with nonlinear decision boundary. The shaded area denotes the positive response of classifier. **(a)** When the positive class is a convex, one can still find the optimal instance by changing one feature only. For instance, changing x_1 will lead to the optimum in this example. **(b)** When the negative class is convex, IMAC algorithm is not able to search the optimal cost instance. The approximate weights suggest that the optimal searching direction is along x_2. Unfortunately, searching along x_1 and x_2 are not optimal in this example.

In this chapter, readers have seen an example of adversarial learning. In particular, I explained a simple algorithm for conducting exploratory attack on linear classifiers. The following two chapters will provide a more comprehensive analysis of the exploratory attack problems. In particular, I will first extend the IMAC algorithm to attack multi-class classifiers in Chapter 4, and then study the attacking algorithm for more general convex-inducing classifiers in Chapter 5.

Part III.

Venerability of Learning Algorithms

In the practice of tolerance, one's enemy is the best teacher.

Dalai Lama

Chapter 4

Exploratory Attack of Multi-Class Linear Classifiers via Line Search

In this chapter, I continue the investigation of the exploratory attack. Specifically, I generalize the IMAC algorithm in Chapter 3 to the family of multi-class linear classifiers; e.g. linear support vector machines [35, 58, 94]. Multi-class linear classifiers have become one of the most promising learning techniques for large sparse data with a huge number of instances and features. I propose an adversarial query algorithm for searching minimal-cost disguised instances. I believe that revealing a scar on the multi-class classifier is the only way to fix it in the future. The contributions of this chapter are:

1. I generalize the problem of exploratory attack to the multi-class linear classifier, where the instance space is divided into multiple convex sets.

2. I prove that effective exploratory attack based on the linear probing is feasible under certain assumption of the adversarial cost. A description of the vulnerability of multi-class linear classifiers is presented.

3. I propose a query algorithm for disguising an adversarial instance as any other classes with minimal cost. The experiment on two real-world data set shows the effectiveness of the proposed algorithm.

This chapter is organized as follows. I pose the near-optimal exploratory problem for the multi-class linear classifier in Section 4.1. Then I describe theorems and algorithm for effectively searching evaded instances in Section 4.2 and Section 4.3, respectively. The experiment results on two real-world data sets are presented in Section 4.4. Section 4.5 concludes.

4.1. Problem Formulation

Let $\mathcal{X} = \{(x_1, \ldots, x_D) \in \mathbb{R}^D \mid L \leq x_d \leq U \text{ for all } d\}$ be the *feature space*. Each component of an *instance* $\mathbf{x} \in \mathcal{X}$ is a *feature* bounded by L and U which I denote as x_d. A basis vector of the form $(0, \ldots, 0, 1, 0, \ldots, 0)$ with a 1 only at the d^{th} feature terms $\boldsymbol{\delta}_d$. I assume that the feature space representation is known to the adversary, thus the adversary can query any point in $\mathcal{X}$.

4.1.1. Multi-Class Linear Classifier

The target classifier f is a mapping from feature space $\mathcal{X}$ to its response space $\mathcal{K}$; i.e. $f : \mathcal{X} \to \mathcal{K}$. I restrict the attention to *multi-class linear classifiers* and use $\mathcal{K} = \{1, \ldots, K\}, K \geq 2$ so that

$$f(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\ \mathbf{w}_k \mathbf{x}^{\mathrm{T}} + b_k, \tag{4.1}$$

where $k = 1, \ldots, K$ and $\mathbf{w}_k \in \mathbb{R}^D, b_k \in \mathbb{R}$. Decision boundaries between class k and other classes are characterized by $\mathbf{w}_k$ and b_k. I assume that $\mathbf{w}_1, \ldots, \mathbf{w}_K$ are linearly independent. The classifier f partitions $\mathcal{X}$ into K sets; i.e. $\mathcal{X}_k = \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = k\}$.

4.1.2. Attack of Adversary

As a motivating example, consider a text classifier that categorizes incoming emails into different topics; e.g. sports, politics, lifestyle, spam, etc. An advertiser of pharmacological products is more likely to disguise the spam as lifestyle rather than politics in order to attract potential consumers while remaining inconspicuous.

I assume the adversary's attack will be against a fixed f so the learning method of decision boundaries and the training data used to establish the classifier are irrelevant. The adversary does not know any parameter of f but can observe $f(\mathbf{x})$ for any $\mathbf{x}$ by issuing a *membership query*. In fact, there are a variety of domain specific mechanisms that an adversary can employ to observe the classifier's response to a query. Moreover, the adversary is only aware of an adversarial instance $\mathbf{y}^{\mathrm{m}}$ in some class, and has no information about instances in other classes. This differs from previous work which require at least one instance in each binary class [107, 117]. In practice, $\mathbf{y}^{\mathrm{m}}$ can be seen as the most desired instance of adversary; e.g. the original spam. The adversary attempts to disguise $\mathbf{y}^{\mathrm{m}}$ so that it can be recognized as other classes.

4.1.3. Adversarial Cost

I assume that the adversary has the access to an *adversarial cost function* $a(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{0+}$. An adversarial cost function measures the distance between two instances $\mathbf{x}, \mathbf{y}$ in $\mathcal{X}$ from the adversary's prospective. I focus on a linear cost function which measures the weighted ℓ_1 distance so that

$$a(\mathbf{x}, \mathbf{y}) = \sum_{d=1}^{D} e_d |x_d - y_d|, \tag{4.2}$$

where $0 < e_d < \infty$ represents the cost coefficient of the adversary associates with the d^{th} feature, allowing that some features may be more important than others. In particular, given the adversarial instance $\mathbf{y}^{\mathrm{m}}$, function $a(\mathbf{x}, \mathbf{y}^{\mathrm{m}})$ measures different costs of using some instances as compared to others. Moreover, I use $\mathcal{B}(\mathbf{y}, C) = \{\mathbf{x} \in \mathcal{X} \mid a(\mathbf{x}, \mathbf{y}) \leq C\}$ to denote the cost ball centered at $\mathbf{y}$ with cost no more than C.

In generalizing work [107], I alter the definition of *minimal adversarial cost* (MAC). Given a fixed classifier f and an adversarial cost function a I define the MAC of class k with respect to an instance $\mathbf{y}$ to be the value

$$\mathrm{MAC}(k, \mathbf{y}) = \min_{\mathbf{x}:\mathbf{x} \in \mathcal{X}_k} a(\mathbf{x}, \mathbf{y}), \quad k \neq f(\mathbf{y}).$$

4.1.4. Disguised Instances

I now introduce some instances with special adversarial cost that the adversary is interested in. First of all, instances with cost of $\mathrm{MAC}(k, \mathbf{y})$ are termed *instances of minimal adversarial cost* (IMAC), which is formally defined as

$$\mathrm{IMAC}(k, \mathbf{y}) = \{\mathbf{x} \in \mathcal{X}_k \,|\, a(\mathbf{x}, \mathbf{y}) = \mathrm{MAC}(k, \mathbf{y}), k \neq f(\mathbf{y})\}\,.$$

Ideally, the adversary attempts to find $\mathrm{IMAC}(k, \mathbf{x}^{\mathrm{A}})$ for all $k \neq f(\mathbf{y}^{\mathrm{m}})$. The most naive way for an adversary to find the IMAC is performing a brute-force search. That is, the adversary randomly samples points in $\mathcal{X}$ and updates the best found instance repetitively. To formulate this idea, I further extend the definition of IMAC. Assume $\widetilde{\mathcal{X}}$ is the set of adversary's sampled or observed instances so far and $\widetilde{\mathcal{X}} \subset \mathcal{X}$, I define *instance of sample minimal adversarial cost* (ISMAC) of class k with respect to an instance $\mathbf{y}$ to be the value

$$\mathrm{ISMAC}(k, \mathbf{y}) = \underset{\mathbf{x}:\mathbf{x}\in\widetilde{\mathcal{X}}\cap\mathcal{X}_k}{\operatorname{argmin}} \; a(\mathbf{x}, \mathbf{y}), \quad k \neq f(\mathbf{y}).$$

Note, that in practice the exact decision boundary is unknown to the adversary, thus finding exact value of IMAC becomes an infeasible task. Nonetheless, it is still tractable to approximate IMAC by finding ϵ-IMAC, which is defined as follows

$$\epsilon\text{-}\mathrm{IMAC}(k, \mathbf{y}) = \{\mathbf{x} \in \mathcal{X}_k \,|\, a(\mathbf{x}, \mathbf{y}) \leq (1+\epsilon) \cdot \mathrm{MAC}(k, \mathbf{y}), k \neq f(\mathbf{y}), \epsilon > 0\}\,.$$

That is, every instance in ϵ-$\mathrm{IMAC}(k, \mathbf{y})$ has the adversarial cost no more than a factor of $(1+\epsilon)$ of the $\mathrm{MAC}(k, \mathbf{y})$. The goal of the adversary now becomes finding ϵ-$\mathrm{IMAC}(k, \mathbf{y}^{\mathrm{m}})$ for all classes $k \neq f(\mathbf{y}^{\mathrm{m}})$ while keeping ϵ as small as possible.

4.2. Theory of Exploratory Attack

I describe the exploratory attack from a theoretical point of view. Specifically, by describing the feature space as a set of convex polytopes, I show that IMAC must be attained on the convex surface. Under a reasonable assumption of adversarial cost function, effective exploratory attack can be performed by linear probing. Finally, I derive bounds for quantitatively studying the vulnerability of multi-class linear classifiers to linear probing.

Lemma 1. *Let $\mathcal{X}_k = \{\mathbf{x} \in \mathcal{X} \,|\, f(\mathbf{x}) = k\}$, where the classifier f is defined in Eq.* (4.1)*. Then $\mathcal{X}_k$ is a closed convex polytope.*

Proof. Let $\mathbf{x}$ be a point in $\mathcal{X}_k$. As $\mathbf{x} \in \mathcal{X}$ it follows that

$$\mathbf{x}^{\mathrm{T}} \geq L \cdot \mathbb{1}_D \qquad \text{and} \qquad -\mathbf{x}^{\mathrm{T}} \geq U \cdot \mathbb{1}_D, \tag{4.3}$$

where $\mathbb{1}_D$ is a D-dimensional unit vector $(1, \ldots, 1)$. Moreover, since $f(\mathbf{x}) = k$, it follows that

$$\begin{pmatrix} \mathbf{w}_k - \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_k - \mathbf{w}_K \end{pmatrix} \mathbf{x}^{\mathrm{T}} \geq \begin{pmatrix} b_1 - b_k \\ \vdots \\ b_K - b_k \end{pmatrix}. \tag{4.4}$$

Thus, the foregoing linear inequalities define an intersection of at most $(K + 2D - 1)$ half-spaces. Denote $H_i^+ = \{\mathbf{x} \in \mathcal{X} \,|\, \widetilde{\mathbf{w}}_i \mathbf{x}^{\mathrm{T}} \geq \widetilde{b}_i\}$, where $1 \leq i \leq (K + 2D - 1)$. I have $\mathcal{X}_k = \bigcap_i H_i^+$, which establishes a half-space representation of convex polytope [76, 133]. □

Lemma 1 indicates that the classifier f decomposes $\mathbb{R}^D$ into K convex polytopes. Following the notations and formulations introduced in [76], I represent a hyperplane H_i as the boundary of a half-space ∂H_i^+; i.e. $H_i = \partial H_i^+ = \{\mathbf{x} \in \mathcal{X} \,|\, \widetilde{\mathbf{w}}_i \mathbf{x}^{\mathrm{T}} = \widetilde{b}_i\}$. Let $\mathcal{X}_k = \bigcap_{p=1}^{P_k} H_p^+$, where $\{H_1^+, \ldots, H_{P_k}^+\}$ is *irredundant*[1] to $\mathcal{X}_k$. Let $\mathcal{H}_k = \{H_1^+, \ldots, H_{P_k}^+\}$ be an irredundant set that defines $\mathcal{X}_k$, then $\mathcal{X}_k \subset \operatorname{int} \mathcal{X}$ provided that none half-space in $\mathcal{H}_k$ is defined by Eq. (4.3). Moreover, I define the p^{th} *facet* of $\mathcal{X}_k$ as $F_{kp} = H_p \cap \mathcal{X}_k$, and the *convex surface* of $\mathcal{X}_k$ as $\partial \mathcal{X}_k = \bigcup_{p=1}^{P_k} F_{kp}$.

Theorem 4.1. *Let $\mathbf{y}$ be an instance in $\mathcal{X}$ and $k \in \mathcal{K} \setminus f(\mathbf{y})$. Let $\mathbf{x}$ be an instance in IMAC$(k, \mathbf{y})$ as defined in Section 4.1.3. Then $\mathbf{x}$ must be attained on the convex surface $\partial \mathcal{X}_k$.*

Proof. I first show the existence of IMAC$(k, \mathbf{y})$. By Lemma 1, $\mathcal{X}_k$ defines a feasible region. Thus minimizing $a(\mathbf{x}, \mathbf{y})$ on $\mathcal{X}_k$ is a solvable problem. Secondly, $\mathcal{X}_k$ is bounded in each direction of the gradient of $a(\mathbf{x}, \mathbf{y})$, which implies that IMAC$(k, \mathbf{y})$ exists.

I now prove that $\mathbf{x}$ must lie on $\partial \mathcal{X}_k$ by contrapositive. Assume that $\mathbf{x}$ is not on $\partial \mathcal{X}_k$ thus is an interior point; i.e. $\mathbf{x} \in \operatorname{int} \mathcal{X}_k$. Let $\mathcal{B}(\mathbf{y}, C)$ denote the ball centered at $\mathbf{y}$ with cost no more than $a(\mathbf{x}, \mathbf{y})$. Due to the convexity of $\mathcal{X}_k$ and $\mathcal{B}(\mathbf{y}, C)$, yielding $\operatorname{int} X_k \cap \operatorname{int} \mathcal{B}(\mathbf{y}, C) \neq \emptyset$. Therefore, there exists at least one instance in $\mathcal{X}_k$ with cost less than $a(\mathbf{x}, \mathbf{y})$, which implies that $\mathbf{x}$ is not IMAC$(k, \mathbf{y})$. □

Theorem 4.1 restricts the searching of IMAC to the convex surface. In particular, when cost coefficients are equal, e.g. $e_1 = \cdots = e_D$, I can show that searching in all axis-aligned directions gives at least one IMAC.

Theorem 4.2. *Let $\mathbf{y}$ be an instance in $\mathcal{X}$ such that $\mathcal{X}_{f(\mathbf{y})} \subset \operatorname{int} \mathcal{X}$. Let P be the number of facets of $\mathcal{X}_{f(\mathbf{y})}$ and F_p be the p^{th} facet, where $p = \{1, \ldots, P\}$. Let $G_d = \{\mathbf{y} + \theta \boldsymbol{\delta}_d \,|\, \theta \in \mathbb{R}\}$, where $d \in \{1, \ldots, D\}$. Let $\mathcal{Q} = \{G_d \cap F_p \,|\, d = 1, \ldots, D, p = 1, \ldots, P\}$, in which each element differs from $\mathbf{y}$ on only one dimension. If the adversarial cost function defined in Eq.* (4.2) *has equal cost coefficients, then there exists at least one $\mathbf{x} \in \mathcal{Q}$ such that $\mathbf{x}$ is IMAC$(f(\mathbf{x}), \mathbf{y})$.*

Proof. Let H_p be the hyperplane defining the p^{th} facet F_p. Consider all the points of intersection of the lines G_d with the hyperplanes H_p; i.e. $\mathcal{I} = \{G_d \cap H_p \,|\, d = 1, \ldots, D, p = 1, \ldots, P\}$. Let $\mathbf{x} = \operatorname{argmin}_{\mathbf{x} \in \mathcal{I}} a(\mathbf{x}, \mathbf{y})$. Then $\mathbf{x}$ is the desired instance.

I prove that $\mathbf{x} \in \mathcal{Q}$ by contrapositive. Suppose $\mathbf{x} \notin \mathcal{Q}$, due to the convexity of $\mathcal{X}_{f(\mathbf{y})}$, the line segment $[\mathbf{x}, \mathbf{y}]$ intersects $\partial \mathcal{X}_{f(\mathbf{y})}$ at a point on another facet. Denote this point as $\mathbf{z}$, then $\mathbf{z}$ differs from $\mathbf{y}$ on only one dimension and $a(\mathbf{z}, \mathbf{y}) < a(\mathbf{x}, \mathbf{y})$.

Next, I prove $\mathbf{x}$ is IMAC$(f(\mathbf{x}), \mathbf{y})$ by contrapositive. Let $\mathcal{B}(\mathbf{y}, C)$ denote the *regular* cost ball centered at $\mathbf{y}$ with cost no more than $a(\mathbf{x}, \mathbf{y})$. That is, each vertex of the cost ball has the same distance of C with $\mathbf{y}$. Suppose $\mathbf{x}$ is not IMAC$(f(\mathbf{x}), \mathbf{y})$, then there exists $\mathbf{z} \in \mathcal{X}_{f(\mathbf{x})} \cap \operatorname{int} \mathcal{B}(\mathbf{y}, C)$. By Theorem 4.1, $\mathbf{z}$ and $\mathbf{x}$ must lie on the same facet, which is defined by a hyperplane H^*. Let $\mathcal{Q}^*$ be intersection points of H^* with lines $G_1, \ldots, G_D$; i.e. $\mathcal{Q}^* = \{G_d \cap H^* \,|\, d = 1, \ldots, D\}$. Then there exists at least one point $\mathbf{v} \in \mathcal{Q}^*$ such that $\mathbf{v} \in \operatorname{int} \mathcal{B}(\mathbf{y}, C)$. Due to the regularity of $\mathcal{B}(\mathbf{y}, C)$, I have $a(\mathbf{v}, \mathbf{y}) < a(\mathbf{x}, \mathbf{y})$. □

[1] Let $\mathcal{C}$ be a convex polytope such that $\mathcal{C} = \bigcap_{i=1}^{n} H_i^+$. The family $\{H_1^+, \ldots, H_n^+\}$ is called *irredundant* to $\mathcal{C}$ provided that $\bigcap_{1 \leq j \leq n, j \neq i} H_j^+ \neq \mathcal{C}$ for each $j = 1, \ldots, n$.

I now define special convex sets for approximating ϵ-IMAC near the convex surface. Given $\epsilon > 0$, the interior parallel body of $\mathcal{X}_k$ is $\mathcal{P}_{-\epsilon}(k) = \{\mathbf{x} \in \mathcal{X}_k \,|\, \mathcal{B}(\mathbf{x}, \epsilon) \subseteq \mathcal{X}_k\}$ and the corresponding exterior parallel body is defined as $\mathcal{P}_{+\epsilon}(k) = \bigcup_{\mathbf{x}\in\mathcal{X}_k} \mathcal{B}(\mathbf{x}, \epsilon)$. Moreover, the interior margin of $\mathcal{X}_k$ is $\mathcal{M}_{-\epsilon}(k) = \mathcal{X}_k \setminus \mathcal{P}_{-\epsilon}(k)$ and the corresponding exterior margin is $\mathcal{M}_{+\epsilon}(k) = \mathcal{P}_{+\epsilon}(k) \setminus \mathcal{X}_k$. By relaxing the searching scope from the convex surface to a margin in the distance ϵ, Theorem 4.1 and Theorem 4.2 immediately imply the following results.

Corollary 1. *Let $\mathbf{y}$ be an instance in $\mathcal{X}$ and $k \in \mathcal{K} \setminus f(\mathbf{y})$. For all $\epsilon > 0$ such that $M_{-\epsilon}(k) \neq \emptyset$, ϵ-IMAC$(k, \mathbf{y}) \subseteq \mathcal{M}_{-\epsilon}(k)$.*

Corollary 2. *Let $\mathbf{y}$ be an instance in $\mathcal{X}$ and ϵ be a positive number such that $\mathcal{P}_{+\epsilon}(f(\mathbf{y})) \subset \operatorname{int} \mathcal{X}$. Let P be the number of facets of $\mathcal{P}_{+\epsilon}(f(\mathbf{y}))$ and F_p be the p^{th} facet, where $p = \{1, \ldots, P\}$. Let $G_d = \{\mathbf{y} + \theta\boldsymbol{\delta}_d \,|\, \theta \in \mathbb{R}\}$, where $d \in \{1, \ldots, D\}$. Let $\mathcal{Q} = \{G_d \cap F_p \,|\, d = 1, \ldots, D, p = 1, \ldots, P\}$, in which each element differs from $\mathbf{y}$ on only one dimension. If adversarial cost function defined in Eq.* (4.2) *has equal cost coefficients, then there exists at least one $\mathbf{x} \in \mathcal{Q}$ such that $\mathbf{x}$ is in ϵ-IMAC$(f(\mathbf{x}), \mathbf{y})$.*

Corollary 1 and Corollary 2 point out an efficient way of approximating ϵ-IMAC with linear probing, which forms the backbone of our proposed algorithm in Section 4.3.

Finally, I consider the vulnerability of a multi-class linear classifier to linear probing. The problem arises of detecting convex polytopes in $\mathcal{X}$ with a random line. As one can easily scale any hypercube to a unit hypercube with edge length 1, the proof is restricted to the unit hypercube in $\mathbb{R}^D$.

Definition 1 (Vulnerability to Linear Probing)**.** *Let $\mathcal{X} = [0, 1]^D$, and $\mathcal{X}_1, \ldots, \mathcal{X}_K$ be the sets that tile $\mathcal{X}$ according to the classifier $f : \mathcal{X} \to \{1, \ldots, K\}$, with $K \geq 2$. Let G be a random line in $\mathbb{R}^D$ that intersects $\mathcal{X}$. Denote Z the number of sets intersect G, the vulnerability of classifier f to linear probing is measured by the expectation of Z.*

When $\mathbb{E}\, Z$ is small, a random line intersects small number of decision regions and not much information is leaked to the adversary. Thus, a robust multi-class classifier that resists linear probing should have a small value of $\mathbb{E}\, Z$.

Theorem 4.3. *Let f be the multi-class linear classifier defined in Eq.* (4.1)*, then the expectation of Z is bounded by $1 < \mathbb{E}\, Z < 1 + \frac{\sqrt{2}(K-1)}{2D}$.*

Proof. By Lemma 1, I have K convex polytopes $\mathcal{X}_1, \ldots, \mathcal{X}_K$. Let $\mathcal{F}$ be the union of all facets of polytopes. Observe that each time the line touches a convex polytope, it only touches its surface twice. The exit point is the entrance point for a new polytope, except at the end-point. Thus, the variable that we are interested in can be represented as

$$Z = |\mathcal{F} \cap G|,$$

where $|\cdot|$ represents the cardinality of a set. Obviously, $\mathbb{E}\, Z$ is bounded by $1 < \mathbb{E}\, Z < K$. I will give a tighter bound in the sequel.

Let $\mathcal{G}$ be the class of all lines of $\mathbb{R}^D$, and μ be the measure of $\mathcal{G}$. Following the notation introduced in [135], I denote the measure of $\mathcal{G}$ that meet a fixed bounded convex set $\mathcal{C}$ as $\mu(\mathcal{G}; \mathcal{G} \cap \mathcal{C} \neq \emptyset)$. Considering an *independent Poisson point process* on $\mathcal{G}$ intensity measure μ, let N be the number of lines intersecting $\mathcal{X}$. I emphasize that N is a finite number, so that one can label them

independently $G_1, \ldots, G_N$. It follows that $G_n, n = 1, \ldots, N$ are *i.i.d.*. Given a fixed classifier f, yielding

$$\begin{aligned}\mathbb{E}\sum_{n=1}^{N}|\mathcal{F}\cap G_n| &= \mathbb{E}\sum_{n=1}^{N}\left[P(N=n)\sum_{i=1}^{n}|\mathcal{F}\cap G_i|\right]\\ &= \sum_{n=1}^{N}\left[P(N=n)\cdot n\cdot\mathbb{E}|\mathcal{F}\cap G_1|\right]\\ &= \mathbb{E}\,N\cdot(\mathbb{E}\,Z).\end{aligned} \tag{4.5}$$

Remark that $G_1, \ldots, G_N$ follow the Possion point process, yielding $\mathbb{E}\,N = \mu(\mathcal{G}; \mathcal{G}\cap\mathcal{X}\neq\emptyset)$. Therefore I can rewrite Eq. (4.5) as,

$$\mathbb{E}\,Z = \frac{\mathbb{E}\sum_{n=1}^{N}|\mathcal{F}\cap G_n|}{\mu(\mathcal{G};\mathcal{G}\cap\mathcal{X}\neq\emptyset)}. \tag{4.6}$$

Next, I compute $\mathbb{E}\sum_{n=1}^{N}|\mathcal{F}\cap G_n|$. Let $M = |\mathcal{F}|$. Due to the convexity of $\mathcal{X}_k$, any given line can hit a facet no more than once. Therefore, I have

$$\begin{aligned}\mathbb{E}\sum_{n=1}^{N}|\mathcal{F}\cap G_n| &= \mathbb{E}\sum_{n=1}^{N}\sum_{m=1}^{M}|F_m\cap G_n|\\ &= \sum_{m=1}^{M}\mathbb{E}\Big|\Big\{n\in\{1,\ldots,N\}|F_m\cap G_n\neq\emptyset\Big\}\Big|\\ &= \sum_{m=1}^{M}\mu(\mathcal{G};\mathcal{G}\cap F_m\neq\emptyset).\end{aligned} \tag{4.7}$$

By substituting Eq. (4.7) into Eq. (4.6), it obtains

$$\mathbb{E}\,Z = \frac{\sum_{m=1}^{M}\mu(\mathcal{G};\mathcal{G}\cap F_m\neq\emptyset)}{\mu(\mathcal{G};\mathcal{G}\cap\mathcal{X}\neq\emptyset)}. \tag{4.8}$$

Assume that μ is translation invariant, by Cauchy-Crofton formula I can rewrite Eq. (4.8) as

$$\mathbb{E}\,Z = \frac{\sum_{m=1}^{M}A(F_m)}{A(\mathcal{X})}, \tag{4.9}$$

where $A(\cdot)$ denotes the surface area[2]. Note, that the numerator of Eq. (4.9) depends on the shape of each polytope and relates to the training method of classifier. Thus, it is difficult to compute the exact value of $\mathbb{E}\,Z$. Nonetheless, I can bound the expectation by using the fact $A(\mathcal{X}) < \sum_{m=1}^{M}A(F_m) < A(\mathcal{X}) + \sqrt{2}(K-1)$ (see [5] for the upper bound). Remark that the surface area $A(\mathcal{X})$ of a unit hypercube is $2D$. I yield

$$1 < \mathbb{E}\,Z < 1 + \frac{\sqrt{2}(K-1)}{2D},$$

[2]The surface area in $\mathbb{R}^D$ is the $(D-1)$-dimensional Lebesgue measure.

which concludes the proof.

□

I remark that Theorem 4.3 implies a way to construct a robust classifier that resists exploratory algorithm based on linear probing, e.g. by jointly minimizing Eq. (4.9) and the error function in the training procedure.

4.3. Algorithm for Approximating ϵ-IMAC

Based on theoretical results, I present an algorithm for deceiving the multi-class linear classifier by disguising the adversarial instance $\mathbf{y}^{\mathrm{m}}$ as other classes with approximately minimal cost, while issuing polynomially many queries in: the number of features, the range of feature, the number of classes and the number of iterations.

An outline of the searching approach is presented in Figs. 4.1 to 4.3. I use a $K \times D$ matrix ϵ for storing ISMAC of K classes and an array C of length K for the corresponding adversarial cost of these instances. The scalar value W represents the maximal cost of all optimum instances. Additionally, it is required a $K \times I$ matrix T for storing the searching path of optimum instances in each iteration. The k^{th} row of matrix ϵ is denoted as $\epsilon[k, :]$. I consider ϵ, T, C, W as global variables so they are accessible in every scope. After initializing variables, the main routine `MLCEvading` (Fig. 4.1 line 4) first invokes `MDSearch` (Fig. 4.2) to search instances that is close to the starting point $\mathbf{y}^{\mathrm{m}}$ in all classes and saves them to ϵ. Then it repetitively selects instances from ϵ as new starting points and searches instances with lower adversarial cost (Fig. 4.3 line 6–7). The whole procedure iterates I times. Finally, it obtains $\epsilon[k, :]$ as the approximation of ϵ-IMAC$(k, \mathbf{y}^{\mathrm{m}})$.

I begin by describing `RBSearch` in Fig. 4.3, a subroutine for searching instances near decision boundaries along dimension d. Essentially, given an instance $\mathbf{x}$, an upper bound u and a lower bound l, I perform a recursive binary search on the line segment $\{\mathbf{x} + \theta\boldsymbol{\delta}_d \,|\, l \leq \theta \leq u\}$ through $\mathbf{x}$. The effectiveness of this recursive algorithm relies on the fact that it is impossible to have $\mathbf{x}^{\mathrm{u}}$ and $\mathbf{x}^{\mathrm{l}}$ in the same class while $\mathbf{x}^{\mathrm{m}}$ is in another class. In particular, if the line segment meets an exterior margin $\mathcal{M}_{+\epsilon}(k)$ and ϵ-IMAC$(k, \mathbf{x})$ is the intersection, then `RBSearch` finds an ϵ-IMAC. Otherwise, when the found instance $\mathbf{y}$ yields lower adversarial cost than instance in ϵ does, Figure 4.4 is invoked to update ϵ. The time complexity of `RBSearch` is $\mathcal{O}(\frac{u-l}{\epsilon})$.

I next describe Fig. 4.2. Given $\mathbf{x}$ which is known as ISMAC$(k, \mathbf{y}^{\mathrm{m}})$ and the current maximum cost W, the algorithm iterates $(D-1)$ times on $\mathcal{P}_{+\epsilon}(\mathcal{X}_{f(\mathbf{x})})$ for finding instances with cost lower than W. Additionally, I introduce two heuristics to prune unnecessary queries. First, the searched dimension in the previous iteration of $\mathbf{x}$ is omitted. Second, I restrict the upper and lower bound of the searching scope on each dimension. Specifically, knowing W and $a(\mathbf{x}, \mathbf{y}^{\mathrm{m}}) = c$, I only allow `RBSearch` to find instance in $[x_d - \frac{W-c}{e_d}, x_d + \frac{W-c}{e_d}]$ since any instance lying out of this scope gives adversarial cost higher than W. This pruning is significant when I have obtained ISMAC for every class. Special attention must be paid to searched dimensions of $\mathbf{x}$ (see Fig. 4.2 line 5–7). Namely, if d is a searched dimension before the $(i-1)^{\mathrm{th}}$ iteration, then I relax the searching scope to $[x_d^{\mathrm{A}} - \frac{W-c}{e_d}, x_d^{\mathrm{A}} + \frac{W-c}{e_d}]$ so that no low-cost instances will be missed.

Theorem 4.4. *The asymptotic time complexity of the algorithm is* $\mathcal{O}(\frac{U-L}{\epsilon}DKI)$.

Proof. Follows from the correctness of the algorithm and the fact that the time complexity of `RBSearch` is $\mathcal{O}(\frac{u-l}{\epsilon})$. □

$(\epsilon, C) \leftarrow$`MLCEvading`$(\mathbf{y}^{\mathrm{m}}, \mathbf{e}, D, L, U, K, I, \epsilon)$:

for $k \leftarrow 1$ **to** K **do**
  $\epsilon[k,:] \leftarrow \mathbf{0}$, $T[k,:] \leftarrow \mathbf{0}$, $C[k] \leftarrow +\infty$;
$C[1] \leftarrow 0$;
`MDSearch`$(\mathbf{x}^{\mathrm{A}}, \mathbf{x}^{\mathrm{A}}, \mathbf{e}, 1, 0, D, L, U, 1, \epsilon)$;
for $i \leftarrow 2$ **to** I **do**
  for $k \leftarrow 2$ **to** K **do**
    `MDSearch`$(\epsilon[k,:], \mathbf{x}^{\mathrm{A}}, \mathbf{e}, k, C[k], D, L, U, i, \epsilon)$;

Figure 4.1.: Query algorithm for attacking multi-class linear classifiers

`MDSearch`$(\mathbf{x}, \mathbf{x}^{\mathrm{A}}, \mathbf{e}, k, c, D, L, U, i, \epsilon)$:

for $d \leftarrow 1$ **to** D **do**
  if $d \neq T[k, i-1]$ **then**
    $\delta \leftarrow \frac{W-c}{e_d}$;
    $u = \min\{U, x_d + \delta\}$, $l = \max\{L, x_d - \delta\}$;
    if $d \in \{T[k,1], \ldots, T[k, i-2]\}$ **then**
      if $x_d > x_d^{\mathrm{A}}$ **then** $l = \max\{L, x_d^{\mathrm{A}} - \delta\}$;
      ;
      else $u = \min\{U, x_d^{\mathrm{A}} + \delta\}$;
      ;
    $\mathbf{x}^{\mathrm{u}} \leftarrow \mathbf{x}$, $\mathbf{x}^{\mathrm{l}} \leftarrow \mathbf{x}$;
    $x_d^{\mathrm{u}} \leftarrow u$, $x_d^{\mathrm{l}} \leftarrow l$;
    if $f(\mathbf{x}^{\mathrm{u}}) \neq k$ **then** `RBSearch`$(x_d, u, \mathbf{x}, d, i, \epsilon)$;
    ;
    if $f(\mathbf{x}^{\mathrm{l}}) \neq k$ **then** `RBSearch`$(l, x_d, \mathbf{x}, d, i, \epsilon)$;
    ;

Figure 4.2.: Multi-dimensional search from ISMAC$(k, \mathbf{y}^{\mathrm{m}})$

In particular, if the adversarial cost function has equal cost coefficient, then there exists at least one innocent class $k \in \{2, \ldots, K\}$ such that I can find instance in ϵ-IMAC$(k, \mathbf{y}^{\mathrm{m}})$ with $\mathcal{O}(\frac{U-L}{\epsilon} DKI)$ queries.

4.4. Experiments

I demonstrate the algorithm[3] on two real-world data sets, the 20-newsgroups[4] and the 10-Japanese female face[5]. On the newsgroups data set, the task of the adversary was to evade a text classifier by disguising a commercial spam as a message in other topics. On the face data set, the task of

[3] A Matlab implementation is available at `http://home.in.tum.de/~xiaoh/pakdd2012-code.zip`
[4] `http://people.csail.mit.edu/jrennie/20Newsgroups/`
[5] `http://www.kasrl.org/jaffe.html`

```
RBSearch(l, u, x, d, i, ε):
x* ← x;
if u − l < ε then
    x*_d ← u;
    k ← f(x*), c ← a(x*);
    if c < C[k] then Update(x*, k, c, d, i);
    ;
x^u ← x, x^l ← x, x^m ← x;
x^u_d ← u, x^l_d ← l, x^m_d ← (u+l)/2;
if f(x^m) = f(x^l) then
    RBSearch(m, u, x, d, i, ε);
else if f(x^m) = f(x^u) then
    RBSearch(l, m, x, d, i, ε);
else
    RBSearch(l, m, x, d, i, ε);
    RBSearch(m, u, x, d, i, ε);
```

Figure 4.3.: Recursive binary search on dimension d

```
(ε, C, T, W) ←Update(x*, k, c, d, i):
ε[k, :] ← x*;
C[k] ← c;
T[k, i] ← d;
W ← max{C[1], ..., C[K]};
```

Figure 4.4.: Update ISMAC$(k, \mathbf{y}^{\mathrm{m}})$

adversary was to deceive the classifier by disguising a suspect's face as an innocent. I employed LIBLINEAR [58] package to build target multi-class linear classifiers, which return labels of queried instances. The cost coefficients were set to $e_1 = \cdots = e_D = 1$ for both tasks. For the groundtruth solution, I directly solved the optimization problem with linear constraints Eq. (4.3) and Eq. (4.4) by using the models' parameters. I then measured the average empirical ϵ for $(K-1)$ classes, which is defined as $\widehat{\epsilon} = \frac{1}{K-1}\sum_{k \neq f(\mathbf{y}^{\mathrm{m}})}\left[\frac{C[k]}{\mathrm{MAC}(k,\mathbf{y}^{\mathrm{m}})} - 1\right]$, where $C[k]$ is the adversarial cost of disguised instance of class k. Evidently, small $\widehat{\epsilon}$ indicates better approximation of IMAC.

4.4.1. Spam Disguising

The training data used to configure the newsletter classifier consisted of $7,505$ documents, which were partitioned evenly across 20 different newsgroups. Each document is represented as a vector with $61,188$ dimensions, where each dimension represents the number of occurrences of a word. The accuracy of the classifier on training data was 100% for every class. I set the category "misc.forsale" as the adversarial class. That is, given a random document in "misc.forsale", the adversary attempts to disguise this document as from other category; e.g. "rec.sport.baseball". Parameters of the algorithm were $K = 20, L = 0, U = 100, I = 10, \epsilon = 1$. The adversary was

restricted to query at most $10,000$ times. The adversarial cost of each class is depicted in Fig. 4.5 (left).

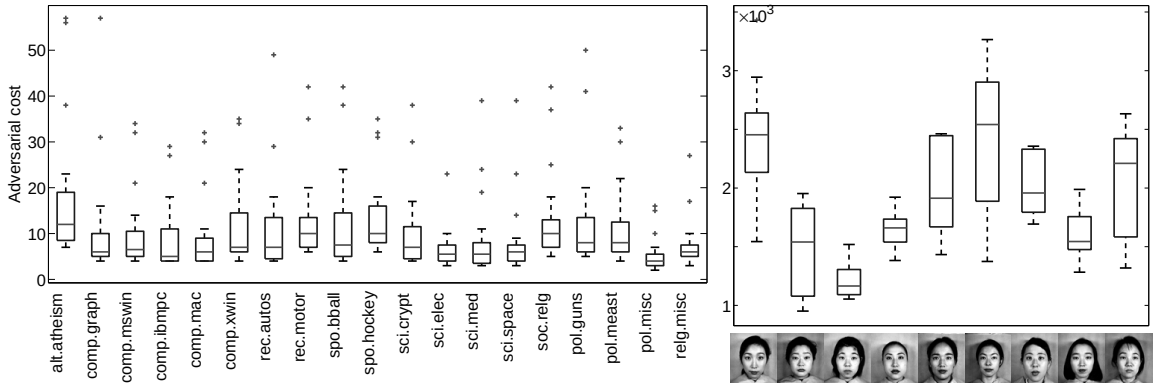


Figure 4.5.: Box plots for adversarial cost of disguised instance of each class. **(Left)** On the 20-newsgroups data set, I considered "misc.forsale" as the adversarial class. Note, that feature values of the instance are non-negative integers as they represent the number of words in the document. Therefore, the adversarial cost can be interpreted as the number of modified words in the disguised document comparing to the original document from "misc.forsale". The value of $\widehat{\epsilon}$ for 19 classes is 0.79. **(Right)** On the 10-Japanese female faces data set, I randomly selected a subject as the suspect. The box plot shows that the adversarial cost of camouflage suspicious faces as other subjects. The value of $\widehat{\epsilon}$ for 9 classes is 0.51. A more illustrative result is depicted in Fig. 4.6.

4.4.2. Face Camouflage

The training data contained 210 gray-scaled images of 7 facial expressions (each with 3 images) posed by 10 Japanese female subjects. Each image is represented by a 100-dimensional vector using principal components. The accuracy of the classifier on training data was 100% for every class. I randomly picked a subject as an imaginary suspect. Given a face image of the suspect, the adversary camouflage this face to make it be classified as other subjects. Parameters of the algorithm were $K = 10, L = -10^5, U = 10^5, I = 10, \epsilon = 1$. The adversary was restricted to query at most $10,000$ times. The adversarial cost of each class is depicted in Fig. 4.5 (right). Moreover, I visualize disguised faces in Fig. 4.6. Observe that many disguised faces are similar to the suspect's face by humans interpretation, yet they are deceptive for the classifier. This visualization directly demonstrates the effectiveness of our algorithm.

It has not escaped the notice that an experienced adversary with certain domain knowledge can reduce the number of queries by careful selecting cost function and employing heuristics. Nonetheless, the goal of this chapter is not to design real attacks but rather examine the correctness and effectiveness of our algorithm so as to understand vulnerabilities of classifiers.

4.5. Conclusion

Understanding the vulnerability of classifiers is the only way to develop resistant classifiers in the future. In this chapter, I showed that multi-class linear classifiers are vulnerable to the exploratory

Figure 4.6.: Disguised faces given by the algorithm to defeat a multi-class face recognition system. The original faces (with neutral expression) of 10 females are depicted in the first row, where the left most one is the imaginary suspect and the remaining 9 people are innocents. From the second row to sixth row, faces of the suspect with different facial expressions are fed to the algorithm (see the first column). The output disguised faces from the algorithm are visualized in the right hand image matrix. Each row corresponds to disguised faces of the input suspicious face on the left. Each column corresponds to an innocent.

attack and presented an algorithm for disguising the adversarial instance. In Chapter 5, I will generalize the exploratory attack problem to the family of classifier with convex-inducing decision boundaries.

Chapter 5

Exploratory Attack on Convex-Inducing Classifiers via Random Walks

In this chapter, I extend the exploratory attack problem to a more general family of classifiers, namely the *convex-inducing* classifier. A convex-inducing classifier partitions the input space into two sets and at least one of them is convex. Unlike the line search algorithms used in Chapter 3 and Chapter 4, this chapter formalizes the exploratory attack on convex-inducing classifiers as a ℓ_p-norm minimization problem. To solve this problem, I develop an algorithm based on random walks in the convex body. Underlying the algorithm is a sophisticated analysis, which combines tools from convex geometry, geometric tomography and probability theory.

This chapter is organized as follows. The problem of exploratory attack is formulated in Section 5.1. Previous work and methods are briefly recapitulated in Section 5.2. The proposed attack algorithm for finding the optimal disguised solution in the convex benign set is described in 5.3. Section 5.4 provides an theoretical study of the algorithm from a geometric perspective, which clearly elucidates the convergence rate, the upper bound of expected iterations and the required number of samples per iteration. Some heuristics and implementation issues are remarked in Section 5.5. Section 5.6 reports the experimental results on both synthetic data and real-world data. Discussions on detecting the exploratory attack are made in Section 5.7. Finally, conclusions are drawn in Section 5.8.

5.1. Problem Formulation

Let $\mathcal{X} \subseteq \mathbb{R}^D$ be the *input space*, and $\mathcal{Y} := \{-1, 1\}$ be the *response space*. Each *instance* $\mathbf{x} \in \mathcal{X}$ is represented as a D-dimensional vector. A classifier $f : \mathcal{X} \to \mathcal{Y}$ partitions $\mathcal{X}$ into two sets, the *benign* set $\mathcal{X}^- := \{\mathbf{x} \in \mathcal{X} \,|\, f(\mathbf{x}) = -1\}$ and the *malicious* set $\mathcal{X}^+ := \{\mathbf{x} \in \mathcal{X} \,|\, f(\mathbf{x}) = 1\}$. The adversary does *not* know the decision boundary of f, but can observe the response $f(\mathbf{x})$ via a *membership oracle*, that is, a procedure that reports whether $\mathbf{x}$ is malicious or not, but provides no other information. The adversary can query any instance in $\mathcal{X}$.

Let $\mathbf{y}^{\mathrm{m}} \in \mathcal{X}^+$ be a malicious instance blocked by the classifier. The cost function $g : \mathcal{X} \to \mathbb{R}_{0+}$ measures the adversary's effort of modifying $\mathbf{y}^{\mathrm{m}}$ as $\mathbf{x}$. In this chapter I focus on $g(\mathbf{x}) := \|\mathbf{x} - \mathbf{y}^{\mathrm{m}}\|_{\ell_p}$, where $p \in [1, \infty]$. As a motivating example, the ℓ_1-norm represents a cost measurement based on the edit distance for the email spam. Given $\mathbf{y}^{\mathrm{m}}$, the goal of the adversary is to find an

instance in $\mathcal{X}^-$ that gives the minimum cost. Formally, this problem can be formulated as

$$\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{y}^{\mathrm{m}}\|_{\ell_p} \qquad \text{subject to} \quad \mathbf{x} \in \mathcal{X}^-, \tag{5.1}$$

where $\mathcal{X}^-$ is specified by the membership oracle f.

Let $\mathbf{x}^*$ be a solution of (5.1), the goal of the adversary is to approximate $\mathbf{x}^*$ within a constant factor $\epsilon > 0$. More exactly, an attack algorithm accomplishing this should find an instance $\mathbf{x}$ that approximates $\mathbf{x}^*$ either with *absolute error*, i.e. $g(\mathbf{x}) - g(\mathbf{x}^*) \leq \epsilon$), or with *relative error* $\epsilon > 0$, i.e. $g(\mathbf{x}) \leq (1 + \epsilon)g(\mathbf{x}^*)$, using finite number of queries. An algorithm requiring less queries is more desirable for the adversary. I hereinafter say that a solution is an ϵ-*optimal* provided that it has the absolute error ϵ. Given some input, an algorithm is called a *polynomial algorithm* if it finds an ϵ-optimal solution using polynomial number of queries with respect to $\frac{1}{\epsilon}$ and the dimensions of the input space D.

I present a new evasion attack algorithm against convex-inducing classifiers whose $\mathcal{X}^-$ is convex and $\mathcal{X}^+$ is non-convex[1]. The main ingredient of the proposed algorithm is sampling by random walks in a sequence of progressively smaller convex bodies. Given an *absolute error* $\epsilon > 0$, the algorithm guarantees finding the ϵ-optimal solution (i.e., such that $g(\mathbf{x}) - g^* \leq \epsilon$) in polynomial time.

5.2. Related Work

Note that without any convexity assumption on $\mathcal{X}^+$ and $\mathcal{X}^-$, problem (5.1) is hard and no polynomial algorithm exists. Previous work relax this problem based on different assumptions, which can be summarized as follows.

- When both $\mathcal{X}^+, \mathcal{X}^-$ are convex, the decision boundary must be a hyperplane. Authors in [107] present a line search algorithm that approximates the normal of the decision hyperplane using membership queries. Once the hyperplane is determined, problem (5.1) is immediately solved by linear programming.
- When only $\mathcal{X}^+$ is convex and $p = 1$, an ϵ-optimal solution can be found by repetitively querying the vertices of ℓ_p-norm balls co-centered at $\mathbf{y}^{\mathrm{m}}$ till one vertex belongs to the benign set [117]. The algorithm can be implemented using line search techniques.
- When only $\mathcal{X}^-$ is convex and $p \in [1, \infty]$, the problem becomes minimizing a convex function over a convex set in $\mathbb{R}^D$. Authors in [117] present a centroid cutting plane (CCP) method that iteratively eliminates a halfspace through the approximated centroid of $\mathcal{X}^-$, which is inspired by the randomized cutting plane scheme proposed in [10].
- When only $\mathcal{X}^+$ is convex and $p \in (1, \infty]$, problem (5.1) can not be solved in polynomial time [117].

Learning with membership queries has been studied in many literature [3, 4], especially in the *active learning* scenarios [111, 156]. While active learning and exploratory attack are similar in their exploration of querying strategies, there are significant differences between these two

[1] When the adversary is unaware of which set is convex, they can combine the results from the proposed algorithm and the Multi-line search algorithm [117].

settings. First of all, in many active learning scenarios, queries are selectively drawn from a finite pool of instances. In the exploratory attack, however, the adversary can query any instance in the input space. Moreover, the objective of the adversary is not to approximate the decision function of the classifier. In other word, the classifier will still be largely unspecified to the adversary after the exploratory attack. Although active learning methods can be adopted for the solution of problem (5.1), the query complexity is much higher for general convex classes [128].

Inspired by the Grünbaum theorem [77], the cutting plane scheme is first proposed in [95] for solving convex programming problem and has been extensively studied in [104, 119]. Instead of using the centroid as the cutting point, different approximate centers have been studied in [141, 98], which are known as ellipsoid methods. In recent years, both cutting plane and ellipsoid methods have gained new interest motivated by the randomized algorithms [89, 88]. In particular, randomized cutting plane schemes based on random walks have been studied by several authors [10, 126, 41].

5.3. Algorithm

I focus on the non-trivial case in which $\mathcal{X}^-$ is convex and $p \in [1, \infty]$. In the spirit of randomized cutting plane scheme [10, 126, 41], the proposed algorithm alternates between cutting the convex body and performing random walks. As a consequence, the feasible region becomes smaller and smaller from iteration to iteration until an ϵ-optimal solution is found. The major difference between the proposed method and CCP method [117] is twofold. First, I do not approximate the centroid of the feasible region. The cutting point is the random sample with the minimum cost. Second, the cut is performed by intersecting the feasible region with ℓ_p-norm balls.

The main steps of the proposed method are summarized in Algorithm 5.1. Let the superscript $^{\langle k \rangle}$ denote the iteration number. Given an initial benign instance $\mathbf{x}^{\langle 0 \rangle} \in \mathcal{X}^-$, the algorithm starts with the set $\mathcal{P}^{\langle 0 \rangle} := \mathcal{X}^-$ and generates N random samples $S^{\langle 0 \rangle}$ in $\mathcal{P}^{\langle 0 \rangle}$. Denote the sample with the minimum cost as $\mathbf{x}^{\langle 0 \rangle}$ (see line 4). Then consider a new set $\mathcal{P}^{\langle 1 \rangle} \subset \mathcal{P}^{\langle 0 \rangle}$, which is obtained by cutting off a portion of $\mathcal{P}^{\langle 0 \rangle}$ with a cost ball $\mathcal{B}^{\langle 0 \rangle}$ centered at $\mathbf{y}^{\mathrm{m}}$ with radius $g(\mathbf{x}^{\langle 0 \rangle})$ (see line 6). The algorithm proceeds by walking randomly in $\mathcal{P}^{\langle 1 \rangle}$ until $S^{\langle 1 \rangle}$ contains N random samples (see lines 9 to 17). In the next iteration, the algorithm selects the minimum cost sample $\mathbf{x}^{\langle 1 \rangle}$ and cuts $\mathcal{P}^{\langle 1 \rangle}$ with $\mathcal{B}^{\langle 1 \rangle}$, which gives a smaller set $\mathcal{P}^{\langle 2 \rangle} \subset \mathcal{P}^{\langle 1 \rangle}$, and so forth. Remark that the intersection of two convex sets is convex, one obtain a sequence of convex sets

$$\mathcal{P}^{\langle k \rangle} \subset \mathcal{P}^{\langle k-1 \rangle} \subset \cdots \subset \mathcal{P}^{\langle 1 \rangle} \subset \mathcal{P}^{\langle 0 \rangle} := \mathcal{X}^-,$$

and a sequence of points $\mathbf{x}^{\langle k \rangle}$ having the property

$$g(\mathbf{x}^*) \leq g(\mathbf{x}^{\langle k \rangle}) \leq g(\mathbf{x}^{\langle k-1 \rangle}) \leq \cdots \leq g(\mathbf{x}^{\langle 1 \rangle}) \leq g(\mathbf{x}^{\langle 0 \rangle}).$$

Finally, the algorithm terminates when either all probing opportunities have been used up, or $\mathbf{x}^k$ is an ϵ-optimal solution; e.g. $g(\mathbf{x}^{\langle k-1 \rangle}) - g(\mathbf{x}^{\langle k \rangle}) < 10^{-3}\,\epsilon$. Figure 5.2(a,b) illustrates the k^{th} iteration of the algorithm.

The idea of using random walks to generate samples in the convex set has been studied in [55, 143]. Here, we employ a similar strategy used in the hit-and-run algorithm [143]. Roughly speaking, each random sample is generated in two steps. First, it selects a line L with a random direction through a starting point $\mathbf{x}$ in the convex body $\mathcal{P}$. Then the starting point moves to a new

Input : dimensions D, cost function g, initial instance $\mathbf{x}^{\langle 0 \rangle} \in \mathcal{X}^-$, number of samples N, maximum step length Ω, absolute error ϵ.
Output: disguised instance $\mathbf{x}^{\langle k \rangle}$

$k \leftarrow 0, \mathcal{P}^{\langle 0 \rangle} \leftarrow \mathcal{X}^-$;
$S^{\langle 0 \rangle} \leftarrow$ Randomly generate N samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in $\mathcal{P}^{\langle 0 \rangle}$;
repeat
  $\mathbf{x}^{\langle k \rangle} \leftarrow \underset{\mathbf{x}_i \in S^{\langle k \rangle}}{\operatorname{argmin}}\, g(\mathbf{x}_i)$; /* select the sample with minimum cost */
  $\mathcal{B}^{\langle k \rangle} \leftarrow \{\mathbf{x} \in \mathbb{R}^D \mid g(\mathbf{x}) \leq g(\mathbf{x}^{\langle k \rangle})\}$;
  $\mathcal{P}^{\langle k+1 \rangle} \leftarrow \mathcal{P}^{\langle k \rangle} \cap \mathcal{B}^{\langle k \rangle}$; /* cut the feasible region */
  $\mathbf{x}_0 \leftarrow \mathbf{x}^{\langle k \rangle}$;
  $S^{\langle k+1 \rangle} \leftarrow \emptyset$;
  for $i \leftarrow 1$ **to** N **do**
    for $j \leftarrow 1$ **to** N **do** $\alpha_j \sim \mathcal{N}(0, 1)$;
    ;
    $\mathbf{u} \leftarrow \sum_{\mathbf{v}_j \in S^{\langle k \rangle}} \alpha_j \mathbf{v}_j$; /* generate a random direction */
    $\omega \leftarrow \Omega$;
    repeat $\omega \sim \mathrm{Uniform}(0, \omega)$ **until** $\mathbf{x}_{i-1} + \omega \mathbf{u} \in \mathcal{P}^{\langle k+1 \rangle}$;
    ;
    $\mathbf{x}_i \leftarrow \mathbf{x}_{i-1} + \omega \mathbf{u}$; /* move to the new point */
    $S^{\langle k+1 \rangle} \leftarrow S^{\langle k+1 \rangle} \cup \mathbf{x}_i$;
  $k \leftarrow k + 1$;
until *stopping rule*;
return $\mathbf{x}^{\langle k \rangle}$;

Figure 5.1.: Exploratory attack on convex $\mathcal{X}^-$ by random walks

point chosen randomly from $\mathcal{P} \cap L$. However, if $\mathcal{P}$ is severely elongated, then the uniform random directions will rarely align with the long axis of $\mathcal{P}$. As a consequence, random samples may not be generated uniformly in $\mathcal{P}$, as shown in Figure 5.2(c). To address this problem, I use $S^{\langle k \rangle}$ to generate walking directions for $S^{\langle k+1 \rangle}$. Intuitively, if $S^{\langle k \rangle}$ is uniformly distributed in $\mathcal{P}^{\langle k \rangle}$, a direction vector given by the linear combination of $S^{\langle k \rangle}$ is more likely aligned with the long axis of $\mathcal{P}^{\langle k \rangle}$. Since the shape of $\mathcal{P}^{\langle k+1 \rangle}$ is similar to $\mathcal{P}^{\langle k \rangle}$, the direction vector aligns with the long axis of $\mathcal{P}^{\langle k+1 \rangle}$ as well, which makes $S^{\langle k+1 \rangle}$ uniformly distributed in $\mathcal{P}^{\langle k+1 \rangle}$. With a stronger analysis, one can show that this strategy always maintain the convex body in the near-isotropic position [89, 10].

5.4. Geometric Analysis

In this section, I analyze Algorithm 5.1 from a geometric perspective and establish some theoretical advances. Before explaining these results, the notations and assumptions need to be clarified. First, the input space has $D \geq 2$ as the problem is trivial for the case of one dimension. For any convex body $\mathcal{K} \in \mathbb{R}^D$, the shorthand notation $\mathrm{vol}(\mathcal{K})$ denotes the volume (i.e. the Lebesgue measure) of $\mathcal{K}$. Moreover, I assume that the set $\mathcal{X}^-$ is circumscribed in an Euclidean ball and contains a small

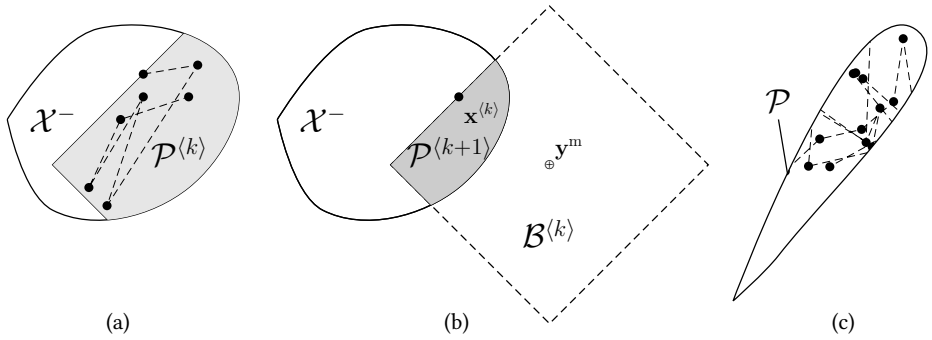


Figure 5.2.: An illustration of Algorithm 5.1 with $g(\mathbf{x}) := \|\mathbf{x} - \mathbf{y}^{\mathrm{m}}\|_{\ell_p}$. **(a)** Random samples are generated in $\mathcal{P}^{\langle k \rangle}$ using random walks. **(b)** The cut is performed with $\mathcal{B}^{\langle k \rangle}$ through the sample with the minimum cost, which results in a smaller convex set $\mathcal{P}^{\langle k+1 \rangle}$. **(c)** When the convex body $\mathcal{P}$ is not in the isotropic position, random samples generated by standard hit-and-run will not be uniformly distributed in $\mathcal{P}$.

Euclidean ball, otherwise no solutions of the problem may exist and the problem could not even theoretically be solved. Finally, the optimal solution is denoted as $\mathbf{x}^*$, which gives the minimum cost $g^* := \|\mathbf{x}^* - \mathbf{y}^{\mathrm{m}}\|_{\ell_p}$.

5.4.1. Main Results

First of all, I show that the algorithm produces a smaller and smaller convex body from iteration to iteration.

Theorem 5.1. *In the k^{th} iteration of Algorithm 5.1, the expected volume of $\mathcal{P}^{\langle k \rangle}$ is at most*

$$\mathbb{E}[\mathrm{vol}(\mathcal{P}^{\langle k \rangle})] \leq \left(\frac{D}{D+N}\right)^k \mathrm{vol}(\mathcal{X}^-).$$

The proof is given in Section 5.4.2. Theorem 5.1 suggests that the random walks is performed in a shrinking convex body, which guarantees the efficiency of the proposed algorithm. Observe that when N goes to infinity, nearly all volume is cut out and the remaining set is close to empty. This is perfectly consistent with the intuition. Comparing to CCP method that reduces volume by $\mathrm{vol}(\mathcal{P}^{\langle k \rangle}) \leq \frac{2}{3}\mathrm{vol}(\mathcal{P}^{\langle k-1 \rangle})$, the proposed method yields a deeper cut when the number of random samples $N > \frac{D}{2}$. The following corollary establishes the minimal number of samples required in each iteration to guarantee that the proposed method cuts deeper than the central-cut method with *arbitrarily high probability*.

Corollary 3. *Given a probability level $\xi > 0$, set*

$$N \geq 2.2 \ln \frac{1}{\xi}.$$

Then, in each iteration Algorithm 5.1 cuts off more volume than the central-cut method with probability at least $1 - \xi$.

Proof. Due to Theorem 1 in [77], the probability that the proposed algorithm cuts off less volume than the central-cut method is $(1-\frac{1}{e})^N$. By solving $(1-\frac{1}{e})^N \leq \xi$, one obtain $N \geq \ln \frac{1}{\xi} / \ln \frac{1}{1-\frac{1}{e}} \geq 2.2 \ln \frac{1}{\xi}$. □

The next theorem shows that the absolute error of $\mathbf{x}^{\langle k \rangle}$ is reduced iteratively, which allows one to bound the expected number of iterations for finding an ϵ-optimal solution.

Theorem 5.2. *Given an initial instance $\mathbf{x}^{\langle 0 \rangle} \in \mathcal{X}^-$, the expected absolute error in the k^{th} iteration is at most*

$$\mathbb{E}[g(\mathbf{x}^{\langle k \rangle}) - g^*] \leq \left(\frac{1}{N+1}\right)^{\frac{k}{D}} \mathbb{E}[g(\mathbf{x}^{\langle 0 \rangle}) - g^*].$$

The expected number of iterations to find an ϵ-optimal solution is at most

$$k = \left\lceil \frac{D}{\ln(N+1)} \ln \frac{g(\mathbf{x}^{\langle 0 \rangle}) - g^*}{\epsilon} \right\rceil.$$

The proof is given in Section 5.4.3. It is now clear that Algorithm 5.1 is a polynomial algorithm and the total number of iterations is linearly bounded by the dimensions of the input space. This bound is an important yet missing piece in previous work [117]. As an immediate implication, one can show that the optimal solution with *relative* error can be found in polynomial time as well.

Corollary 4. *Given $\epsilon > 0$, Algorithm 5.1 can compute an instance $\mathbf{x}$ such that $g(\mathbf{x}) - g^* \leq \epsilon\, g^*$ in polynomial time.*

Proof. Let b be a positive lower bound on g^*, i.e. $0 < b \leq g^*$. Due to Theorem 6 in [10], one can compute b in polynomial time. Then invoke Algorithm 5.1 with absolute error $\delta := \epsilon\, b$ and obtain an instance $\mathbf{x}$ satisfying $g(\mathbf{x}) - g^* \leq \epsilon\, b \leq \epsilon\, g^*$. □

5.4.2. Proof of Theorem 5.1

I begin by establishing a lemma concerning the volume ratio in one iteration. The proof relies on the techniques from convex geometry and geometric tomography.

Lemma 2. *Let $\mathcal{K} \subset \mathbb{R}^D$ be a convex body. Let $\mathbf{x}_1, \ldots, \mathbf{x}_N$ be N uniform random points in $\mathcal{K}$. Given a point $\mathbf{y} \in \mathbb{R}^D \setminus \mathcal{K}$, denote $\mathcal{P} := \{\mathbf{x} \in \mathcal{K} \mid \|\mathbf{x} - \mathbf{y}\|_{\ell_p} < \min_{1 \leq i \leq N} \|\mathbf{x}_i - \mathbf{y}\|_{\ell_p}\}$, where $p \in [1, \infty]$. Then*

$$\mathbb{E}\left[\frac{\text{vol}(\mathcal{P})}{\text{vol}(\mathcal{K})}\right] \leq \frac{D}{D+N}.$$

The corresponding second moment about zero is bounded by

$$\mathbb{E}\left[\left(\frac{\text{vol}(\mathcal{P})}{\text{vol}(\mathcal{K})}\right)^2\right] \leq \frac{2\,D^2}{2\,D^2 + 3\,D\,N + N^2}.$$

Proof. First consider the quotient of the volumes of $\mathcal{K}$ and $\mathcal{P}_v := \{\mathbf{x} \in \mathcal{K} \mid \|\mathbf{x} - \mathbf{y}\|_{\ell_p} < v\}$, where v is some positive real value such that $\mathcal{P}_v$ is nonempty. Let $H_{\min}$ be the separating hyperplane of $\mathcal{K}$ and a ℓ_p ball $\mathcal{B}_{\min} := \{\mathbf{x} \in \mathbb{R}^D \mid \|\mathbf{x} - \mathbf{y}\|_{\ell_p} \leq \min_{\mathbf{x} \in \mathcal{K}} \|\mathbf{x} - \mathbf{y}\|_{\ell_p}\}$ and supports both of them. Denote by $H_{\max}$ the hyperplane supports $\mathcal{K}$ and parallels to $H_{\min}$, as depicted in Figure 5.3(a). Let

$\mathbf{w}$ be their normal and h be the $\mathbf{w}$-breadth of $\mathcal{K}$. By rotating $\mathcal{K}$ and $\mathcal{P}_v$ and translating them to $\mathbf{y}$, one can assume without loss of generality that $\mathbf{w} = (1, 0, \ldots, 0)^\top$, i.e. $\mathbf{w}^\top \mathbf{x} = x_1$; and that $\mathbf{y}$ is the origin and $\mathbf{x}^* := \operatorname*{argmin}_{\mathbf{x} \in \mathcal{K}} \|\mathbf{x} - \mathbf{y}\|_{\ell_p} := \mathbf{0}$, as shown in Figure 5.3(b).

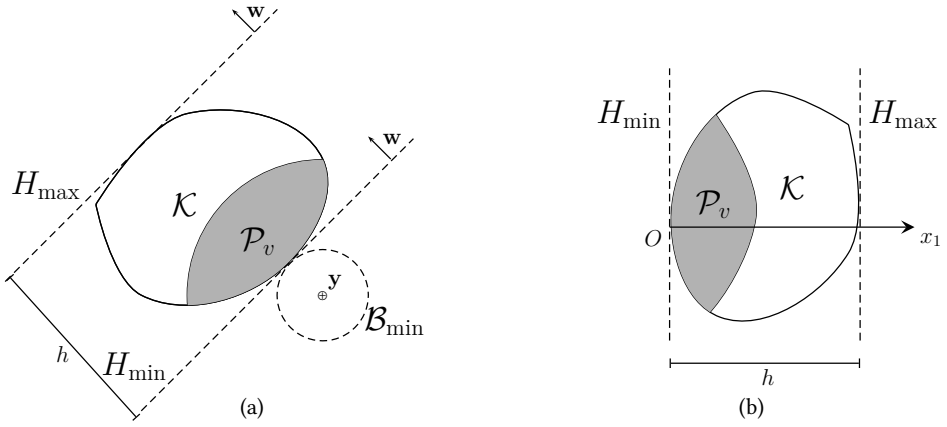


Figure 5.3.: **(a)** Finding two parallel hyperplanes that support $\mathcal{K}$. **(b)** Rotating and translating $\mathcal{K}$ and $\mathcal{P}_v$ until they are aligned with x_1 axis. In this example, the cost function $g(\mathbf{x}) := \|\mathbf{x} - \mathbf{y}^{\mathrm{m}}\|_{\ell_2}$. Gray area denotes $\mathcal{P}_v$ in both figures.

Next, construct a convex body $\mathcal{K}'$ that has the same volume as $\mathcal{K}$ and is symmetric with respect to x_1 axis. Define the function

$$\psi_{\mathcal{K}}(s) := \frac{1}{\mathrm{vol}(\mathcal{K})} \int_{\mathbf{x} \in \mathcal{K},\, \mathbf{w}^\top \mathbf{x} = s} \mathrm{d}s.$$

That is, $\psi_{\mathcal{K}}(s)$ represents the $(D-1)$ dimensional volume of $\mathcal{K}$ intersected with the hyperplane $H_s := \{\mathbf{x} \in \mathbb{R}^D \,|\, x_1 = s\}$, as a fraction of the volume of $\mathcal{K}$.

Consider the set $\mathcal{K}'$ obtained by replacing the cross-section $\mathcal{K} \cap H_s$ for every $s \in [0, h]$ by a $(D-1)$ dimensional ball of volume $\psi_{\mathcal{K}}(s)$ and centered at the point $(s, 0, \ldots, 0)^\top$. Consequently, the new set $\mathcal{K}'$ has the same volume as $\mathcal{K}$, and it is symmetric with respect to the x_1 axis. To show that $\mathcal{K}'$ is a convex set, let $\mathcal{S}_1$ and $\mathcal{S}_2$ be the cross-sections of $\mathcal{K}'$ at $(s_1, 0, \ldots, 0)^\top$ and $(s_2, 0, \ldots, 0)^\top$, respectively. By the Brunn-Minkowski inequality [136], for any $\alpha \in [0, 1]$ it holds that

$$\mathrm{vol}\left(\alpha\, \mathcal{S}_1 + (1-\alpha)\, \mathcal{S}_2\right)^{\frac{1}{D-1}} \geq \alpha\, \mathrm{vol}(\mathcal{S}_1)^{\frac{1}{D-1}} + (1-\alpha)\, \mathrm{vol}(\mathcal{S}_2)^{\frac{1}{D-1}}.$$

Denote $r_{\mathcal{K}'}(s)$ the radius of $\mathcal{K}' \cap H_s$, then the above inequality implies that $r_{\mathcal{K}'}(\alpha\, s_1 + (1-\alpha)\, s_2) \geq \alpha\, r_{\mathcal{K}'}(s_1) + (1-\alpha)\, r_{\mathcal{K}'}(s_2)$. That is, function $r_{\mathcal{K}'}$ is concave with respect to s. Hence $\mathcal{K}'$ is convex. I employ the same technique to construct a convex set $\mathcal{P}_t$ that has the same volume as $\mathcal{P}_v$ and is symmetric about x_1 axis. Observe that the surface of $\mathcal{P}_t$ intersects x_1 axis at the origin and some point $(t, 0, \ldots, 0)^\top$, where $t \in (0, h]$, as shown in Figure 5.4(a).

Define the set $\mathcal{K}'_{<t} := \{\mathbf{x} \in \mathcal{K}' \,|\, x_1 < t\}$. We immediately have

$$\frac{\mathrm{vol}(\mathcal{P}_v)}{\mathrm{vol}(\mathcal{K})} = \frac{\mathrm{vol}(\mathcal{P}_t)}{\mathrm{vol}(\mathcal{K}')} \leq \frac{\mathrm{vol}(\mathcal{K}'_{<t})}{\mathrm{vol}(\mathcal{K}')} \tag{5.2}$$

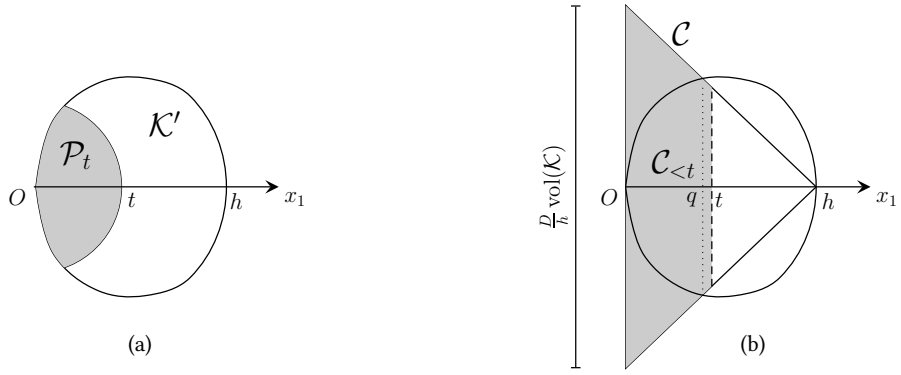


Figure 5.4.: **(a)** Constructing convex sets $\mathcal{K}'$ and $\mathcal{P}_t$ such that they have same volume as $\mathcal{K}$ and $\mathcal{P}_v$, respectively. Gray area denotes $\mathcal{P}_t$. **(b)** Constructing a convex cone $\mathcal{C}$ that has base area $\frac{D}{h}\mathrm{vol}(\mathcal{K}')$ and height h. Gray area denotes $\mathcal{C}_{<t}$.

Define a cone $\mathcal{C}$ with the base area $\frac{D}{h}\mathrm{vol}(\mathcal{K}')$ and the height h as illustrated in Figure 5.4(b). Using the cone volume formula, it yields $\mathrm{vol}(\mathcal{C}) = \mathrm{vol}(\mathcal{K}') = \mathrm{vol}(\mathcal{K})$. Define the set $\mathcal{C}_{<t} := \{\mathbf{x} \in \mathcal{C} \,|\, x_1 < t\}$, $\mathcal{C}_{\geq t} := \{\mathbf{x} \in \mathcal{C} \,|\, x_1 \geq t\}$ and $\mathcal{K}'_{\geq t} := \{\mathbf{x} \in \mathcal{K}' \,|\, x_1 \geq t\}$. Let q be the coordinate at which $\mathcal{K}'_{<q} \subseteq \mathcal{C}_{<q}$ and $\mathcal{C}_{\geq q} \subseteq \mathcal{K}'_{\geq q}$. Observe that for $t \leq q$ one has

$$\mathrm{vol}(\mathcal{K}'_{<t}) \leq \mathrm{vol}(\mathcal{C}_{<t}),$$

and for $t > q$,

$$\mathrm{vol}(\mathcal{K}'_{<t}) = \mathrm{vol}(\mathcal{K}') - \mathrm{vol}(\mathcal{K}'_{\geq t}) \leq \mathrm{vol}(\mathcal{C}) - \mathrm{vol}(\mathcal{C}_{\geq t}) = \mathrm{vol}(\mathcal{C}_{<t}).$$

Thus, for any $t \in (0, h]$ it holds that

$$\mathrm{vol}(\mathcal{K}'_{<t}) \leq \mathrm{vol}(\mathcal{C}_{<t}).$$

By using this fact, one can now rewrite (5.2) as

$$\frac{\mathrm{vol}(\mathcal{P}_v)}{\mathrm{vol}(\mathcal{K})} \leq \frac{\mathrm{vol}(\mathcal{K}'_{<t})}{\mathrm{vol}(\mathcal{K}')} \leq \frac{\mathrm{vol}(\mathcal{C}_{<t})}{\mathrm{vol}(\mathcal{K}')} = \frac{\mathrm{vol}(\mathcal{C}_{<t})}{\mathrm{vol}(\mathcal{C})} = 1 - \frac{(h-t)^D}{h^D}.$$

Consequently, for any $u \in [0, 1]$ it holds that

$$\begin{aligned}\Pr\left(\frac{\mathrm{vol}(\mathcal{P}_v)}{\mathrm{vol}(\mathcal{K})} \geq u\right) \leq \Pr\left(1 - \frac{(h-t)^D}{h^D} \geq u\right) &= \Pr\left(t \geq h - h\,(1-u)^{\frac{1}{D}}\right) \\ &= (1-u)^{\frac{1}{D}}. \end{aligned} \tag{5.3}$$

One can bound the expected quotient of volumes of $\mathcal{P}$ and $\mathcal{K}$ using (5.3)

$$\begin{aligned}\mathbb{E}\left[\frac{\mathrm{vol}(\mathcal{P})}{\mathrm{vol}(\mathcal{K})}\right] = \int_0^1 \Pr\left(\frac{\mathrm{vol}(\mathcal{P})}{\mathrm{vol}(\mathcal{K})} \geq u\right) \mathrm{d}u &= \int_0^1 \left[\Pr\left(\frac{\mathrm{vol}(\mathcal{P}_v)}{\mathrm{vol}(\mathcal{K})} \geq u\right)\right]^N \mathrm{d}u \\ &\leq \int_0^1 (1-u)^{\frac{N}{D}} \mathrm{d}u = \frac{D}{D+N}.\end{aligned}$$

Similarly, the second moment is bounded as

$$\begin{aligned}\mathbb{E}\left[\left(\frac{\mathrm{vol}(\mathcal{P})}{\mathrm{vol}(\mathcal{K})}\right)^2\right] = \int_0^1 \Pr\left(\left(\frac{\mathrm{vol}(\mathcal{P})}{\mathrm{vol}(\mathcal{K})}\right)^2 \geq u\right) \mathrm{d}u &= \int_0^1 \left[\Pr\left(\frac{\mathrm{vol}(\mathcal{P}_v)}{\mathrm{vol}(\mathcal{K})} \geq \sqrt{u}\right)\right]^N \mathrm{d}u \\ &\leq \int_0^1 (1-\sqrt{u})^{\frac{N}{D}} \mathrm{d}u \\ &= \frac{2\,D^2}{2\,D^2 + 3\,D\,N + N^2},\end{aligned}$$

which concludes the proof. □

Consider in k^{th} iteration where samples in previous $(k-1)^{\text{th}}$ iterations are given, then $\mathrm{vol}(\mathcal{P}^{\langle k-1\rangle})$ is uniquely determined. It follows from Lemma 2 that

$$\mathbb{E}[\mathrm{vol}(\mathcal{P}^{\langle k\rangle}) \,|\, \mathrm{vol}(\mathcal{P}^{\langle k-1\rangle})] \leq \frac{D}{D+N}\,\mathrm{vol}(\mathcal{P}^{\langle k-1\rangle}).$$

Taking expectation on both sides and applying the law of iterated expectations on the right-hand side, it yields

$$\mathbb{E}[\mathrm{vol}(\mathcal{P}^{\langle k\rangle})] \leq \frac{D}{D+N}\,\mathbb{E}[\mathrm{vol}(\mathcal{P}^{\langle k-1\rangle})].$$

Thus, Theorem 5.1 is proved by recursively applying the above relation for k times.

5.4.3. Proof of Theorem 5.2

I employ similar techniques used in Section 5.4.2 to show the error reduction in one iteration. Definitions of $\mathcal{K}$, $\mathbf{y}$, $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are followed from Lemma 2.

Lemma 3. *Let* $g' := \min_{1\leq i\leq N} \|\mathbf{x}_i - \mathbf{y}\|_{\ell_p}$, $g^* := \min_{\mathbf{x}\in\mathcal{K}} \|\mathbf{x}-\mathbf{y}\|_{\ell_p}$ *and* $h := \max_{\mathbf{x}\in\mathcal{K}} \|\mathbf{x}-\mathbf{y}\|_{\ell_p} - g^*$. *Then*

$$\mathbb{E}[g' - g^*] \leq \frac{h}{D}\,\mathrm{B}\left(N+1, \frac{1}{D}\right) \leq h\,\left(\frac{1}{N+1}\right)^{\frac{1}{D}},$$

with the corresponding second moment about zero

$$\mathbb{E}[(g'-g^*)^2] \leq \frac{2\,h^2}{D}\,\mathrm{B}\left(N+1, \frac{2}{D}\right) \leq h^2\,\left(\frac{1}{N+1}\right)^{\frac{2}{D}},$$

where $\mathrm{B}(\cdot,\cdot)$ *is the Euler Beta function.*

Proof. Construct $\mathcal{K}'$ and $\mathcal{P}_t$ as in Lemma 2. Consider the inverted cone $\mathcal{C}'$ with the base area $\frac{D}{h}\mathrm{vol}(\mathcal{K}')$ and the height h, as shown in Figure 5.5. Given $t \in (0, h]$, let q be the coordinate at

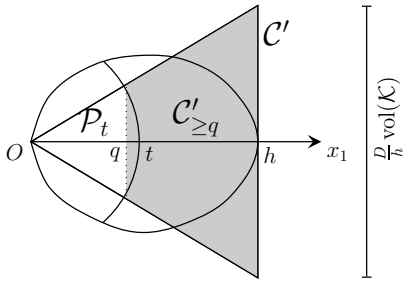


Figure 5.5.: An inverted convex cone $\mathcal{C}'$ has base area $\frac{D}{h}\mathrm{vol}(\mathcal{K}')$ and height h. Gray area denotes $\mathcal{C}'_{\geq q}$.

which $\mathcal{C}'_{<q} \subseteq \mathcal{P}_{<q}$ and $\mathcal{P}_{\geq q} \subseteq \mathcal{C}'_{\geq q}$, where $\mathcal{P}_{<q} := \{\mathbf{x} \in \mathcal{P}_t \,|\, x_1 < q\}$, $\mathcal{P}_{\geq q} := \{\mathbf{x} \in \mathcal{P}_t \,|\, x_1 \geq q\}$, $\mathcal{C}'_{<q} := \{\mathbf{x} \in \mathcal{C}' \,|\, x_1 < q\}$ and $\mathcal{C}'_{\geq q} := \{\mathbf{x} \in \mathcal{C}' \,|\, x_1 \geq q\}$. Let s be a random variable in $[0, h]$, it holds that

$$\Pr(s \geq t) = 1 - \frac{\mathrm{vol}(\mathcal{P}_t)}{\mathrm{vol}(\mathcal{K}')} \leq 1 - \frac{\mathrm{vol}(\mathcal{C}'_{<q})}{\mathrm{vol}(\mathcal{K}')} = \frac{\mathrm{vol}(\mathcal{C}'_{\geq q})}{\mathrm{vol}(\mathcal{C}')} = \frac{h^D - q^D}{h^D}.$$

Thus, the expected value can be bounded as

$$\begin{aligned}\mathbb{E}[g'] = \int_0^h \Pr(g' \geq t)\mathrm{d}t \leq \int_0^h \left(\frac{h^D - q^D}{h^D}\right)^N \mathrm{d}q &= \frac{h}{D}\mathrm{B}\left(N+1, \frac{1}{D}\right) \\ &\leq h\left(\frac{1}{N+1}\right)^{\frac{1}{D}},\end{aligned}$$

where the last inequality holds due to [2]. Similarly, the second moment about zero is given by

$$\begin{aligned}\mathbb{E}[(g')^2] = \int_0^h \Pr(g' \geq \sqrt{t})\mathrm{d}t \leq \int_0^h \left(\frac{h^D - q^{\frac{D}{2}}}{h^D}\right)^N \mathrm{d}q &= \frac{2\,h^2}{D}\mathrm{B}\left(N+1, \frac{2}{D}\right) \\ &\leq h^2\left(\frac{1}{N+1}\right)^{\frac{2}{D}},\end{aligned}$$

which concludes the proof. □

Consider in k^{th} iteration where samples at $1, \ldots, (k-1)^{\text{th}}$ iterations are given, then $g(\mathbf{x}^{\langle k-1\rangle})$ is uniquely determined. It follows from Lemma 3 that

$$\mathbb{E}[g(\mathbf{x}^{\langle k\rangle}) - g^* \,|\, g(\mathbf{x}^{\langle k-1\rangle})] \leq \left(\frac{1}{N+1}\right)^{\frac{1}{D}} [g(\mathbf{x}^{\langle k-1\rangle}) - g^*].$$

Taking the expectation on both sides and applying the law of iterated expectations, it obtains

$$\mathbb{E}[g(\mathbf{x}^{\langle k\rangle}) - g^*] \leq \left(\frac{1}{N+1}\right)^{\frac{1}{D}} \mathbb{E}[g(\mathbf{x}^{\langle k-1\rangle}) - g^*].$$

A recursive application of the above inequality gives

$$\mathbb{E}[g(\mathbf{x}^{\langle k \rangle}) - g^*] \leq \left(\frac{1}{N+1}\right)^{\frac{k}{D}} \mathbb{E}[g(\mathbf{x}^{\langle 0 \rangle}) - g^*].$$

Remark that the goal is to find $\mathbf{x}$ such that $\mathbb{E}[g(\mathbf{x}) - g^*] \leq \epsilon$. Thus, Theorem 5.2 is proved by substituting the left hand side as ϵ and then solving k.

5.5. Implementation Issues

I highlight some practical considerations for an efficient implementation. First of all, note that if the random walk starts from a point on the surface of the convex body (line 7 of Algorithm 5.1), then it is difficult to generate a feasible walking direction for the first step (see Figure 5.6). This problem becomes more severer in the high dimensional space. As a workaround, in each iteration the cut is performed through the sample with the second smallest cost. The smallest cost sample is thus in the interior of the convex body, which can be used as a good starting point. Second, I gradually reduce the maximum step length of random walks by setting $\Omega := 2\, g(\mathbf{x}^{\langle k \rangle})$ to avoid unnecessary queries. Finally, a membership query should only be sent to the defender's classifier if it can not be asserted by the adversary itself. Readers that interested in technical details are referred to my MATLAB implementation[2].

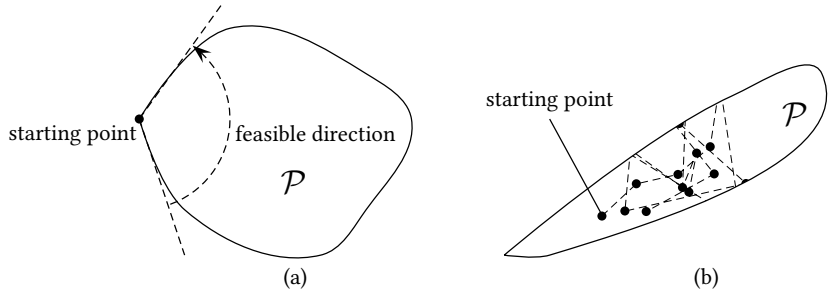


Figure 5.6.: **(a)** The starting point is close to the boundary. The arc represents all feasible walking directions. In the high dimensional space, it is extremely difficult to generate a feasible walking direction. **(b)** The convex set is not in the isotropic position. Random samples are not uniformly distributed in the set.

5.6. Experiments

The experiment section contains two parts. First, the attack algorithms were tested on synthetic classifiers. The goal was to compare the effectiveness and the efficiency of the proposed algorithm with CCP method under different dimensions, cost functions and convex sets. The second set of experiments was conducted on the newsgroups data, where I employed the algorithm for disguising a document to deceive a multi-class linear classifier.

[2] http://home.in.tum.de/∼xiaoh/convexattackmatlab.zip

5.6.1. Synthetic Examples

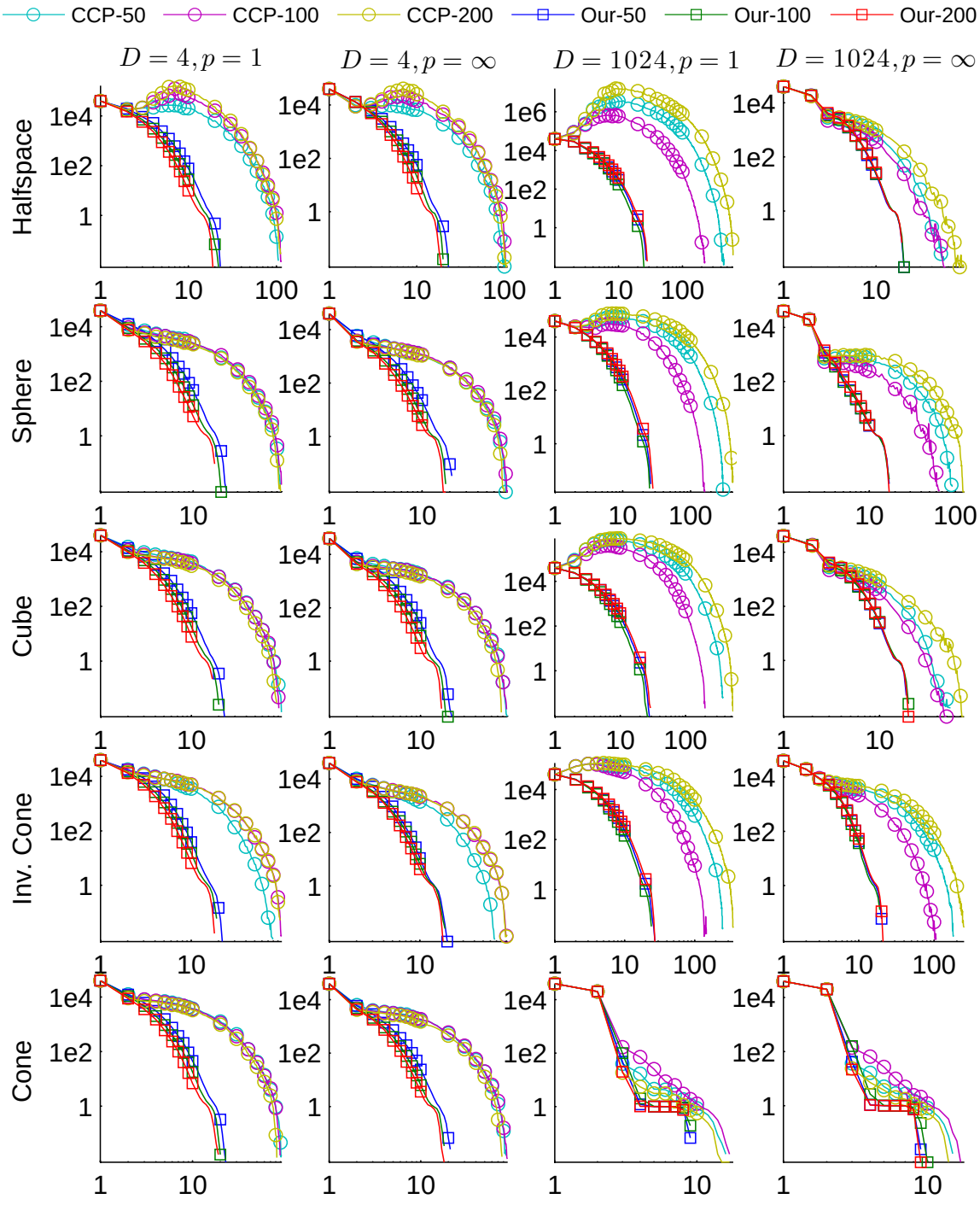


Figure 5.7.: ℓ_1 and ℓ_∞ cost as a function of iterations for 4 and 1,024-dimensional problems. From top to bottom, each row represents $\mathcal{X}^-$ with a special geometry structure. The experiment is repeated for 120 times and the average performance is reported.

I constructed five convex-inducing classifiers by setting the benign set of each classifier as follows:

Halfspace $\mathcal{X}^- := \{\mathbf{x} \in \mathbb{R}^D \,|\, x_1 \geq 0\}$;

Sphere $\mathcal{X}^- := \{\mathbf{x} \in \mathbb{R}^D \,|\, \|\mathbf{x} - (R, 0, \ldots, 0)^\top\|_{\ell_2} \leq R\}$;

Cube $\mathcal{X}^- := \{\mathbf{x} \in \mathbb{R}^D \,|\, \|\mathbf{x} - (R, 0, \ldots, 0)^\top\|_{\ell_\infty} \leq R\}$;

Cone $\mathcal{X}^- := \{\mathbf{x} \in \mathbb{R}^D \,|\, 0 \leq x_1 \leq 2\,R,\, \sum_{d=2}^D x_d^2 \leq x_1\}$;

Inverted cone $\mathcal{X}^- := \{\mathbf{x} \in \mathbb{R}^D \,|\, 0 \leq x_1 \leq 2\,R,\, \sum_{d=2}^D x_d^2 \leq 2\,R - x_1\}$,

where $R := 10,000$ in all settings. Letting the original malicious instance $\mathbf{y}^{\mathrm{m}} := (-1, 0, \ldots, 0)^\top$ and the initial benign instance $\mathbf{x}^{\langle 0 \rangle} := (2\,R, 0, \ldots, 0)^\top$. Hence, I had $\mathbf{x}^* = (0, 0, \ldots, 0)^\top$ by construction. The absolute error $\epsilon := 1$. I implemented CCP method with the same heuristics mentioned in Section 5.5. The number of samples generated in each iteration was set to 50, 100 and 200, respectively.

Figure 5.7 illustrates ℓ_1 and ℓ_∞ cost for problems in 4 and 1,024-dimensional spaces. Observe that the proposed method converges considerably faster than CCP in all settings. Typically, the proposed method reproduced 7 to 8 exact decimal digits for the cost value after 20 iterations, which is about 30 times faster than CCP. In addition, the total number of queries is significantly less than CCP. These advantages are more apparent in high dimensional space. Furthermore, when $\mathcal{X}^-$ is an unbounded convex set (e.g. "halfspace"), CCP shows a poor start and even increases the cost in the first ten iterations. This is due to the fact that CCP bounds the feasible region by the intersection of halfspaces. When the feasible region is not completely bounded, random walks can run away from the cost ball very easily. By contrast, the proposed algorithm shows a robust convergence rate by bounding the feasible region with norm balls.

5.6.2. On Real-World Data

This experiment was conducted on the 20-newsgroups data set[3], which contains 20,000 documents partitioned evenly across 20 different newsgroups. Each document was represented as a 61,188-dimensional vector. Each dimension denoted the number of occurrences of a word. A multi-class linear classifier was trained using LIBLINEAR package [58], which partitioned the input space into 20 convex sets. By considering one set as $\mathcal{X}^-$ and all other sets as $\mathcal{X}^+$, I obtained a binary classifier with the convex benign set. The adversarial cost function $g(\mathbf{x}) := \|\mathbf{x} - \mathbf{y}^{\mathrm{m}}\|_{\ell_1}$ was defined to represent the edit distance (in terms of word) between two documents. From all training instances, I randomly selected a document $\mathbf{y}^{\mathrm{m}} \in \mathcal{X}^+$ and applied the algorithm to find the disguised document in $\mathcal{X}^-$. The algorithm was terminated if no significant cost reduction is observed on $\mathbf{x}^{\langle k \rangle}$. This simulated an exploratory *targeted* attack. For instance, a Viagra seller will disguise the spam as lifestyle tips rather than IT news in order to attract potential consumers while remaining inconspicuous.

Figure 5.8 depicts the relative cost $g(\mathbf{x}^{\langle k \rangle})/g(\mathbf{x}^{\langle 0 \rangle})$ for each newsgroup, where $\mathbf{x}^{\langle 0 \rangle}$ is the initial benign instance. Generally, the cost depends on the location of $\mathbf{y}^{\mathrm{m}}$ and the shape of $\mathcal{X}^-$. On the one hand, we observe a document from "sci.electronics" can be disguised as "comp.os.ms-windows.misc" by only modifying 1.5% of its content, which corresponds to 14 words in the

[3] http://people.csail.mit.edu/jrennie/20Newsgroups/

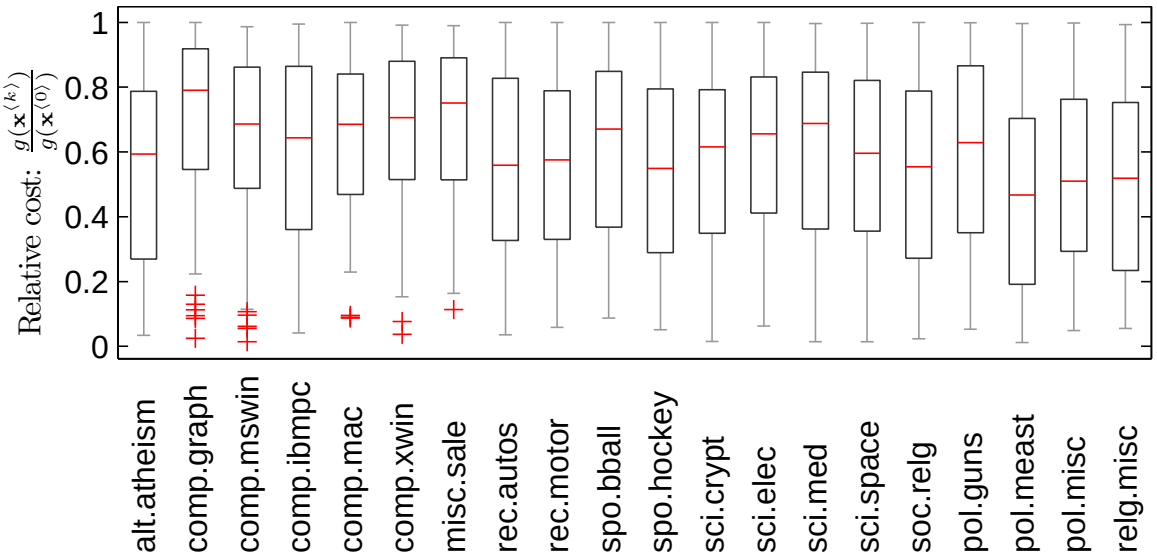


Figure 5.8.: Each column depicts the relative cost $g(\mathbf{x}^{\langle k \rangle})/g(\mathbf{x}^{\langle 0 \rangle})$ for disguising a malicious document as from the benign newsgroup labeled below. Smaller value is preferable for the adversary. The central mark is the median, the edges of the box are the 25^{th} and 75^{th} percentiles, For instance, the first box shows about 25% documents from other newsgroups can be disguised as "alt.atheism" by only changing 27% of their contents, and about 50% can be disguised by changing at most 60% of their contents. The experiment is repeated 100 times for each group.

document. On the other hand, some documents are so difficult to be disguised unless their contents are completely changed. The average number of queries is 606.8 for one exploratory attack.

5.7. Detecting Exploratory Attack

Finally, I discuss a possible defense mechanism against the proposed attack algorithm. The prerequisite is that the defender can identify the sender of each query (e.g. via IP address), then he can simply ignore all malicious probings from the attacker (e.g. using a blacklist) without shutting down the service completely.

Recall that random samples were generated in a smaller and smaller convex body. As a consequence, the queries from an adversary demonstrated a convergent trend on every dimension. Figure 5.9 displays a time series plot of an exploratory attack on 2-dimensional space. By contrast, queries from a regular user have no general tendencies. Therefore, the defender need to concern only the changes over time on one dimension. If a convergent trend is observed, then the classifier is under attack with a high probability. Note that the monitored dimension should be aligned with a long axis of the convex body in order to obtain salient observation. This can be done by selecting the feature with the largest range in benign training instances. Moreover, if the classifier is under series of exploratory attacks, then the monitored dimension will exhibit repeating convergent patterns. In this case, techniques such as autocorrelation and spectrum analysis from signal processing can be adopted for attack detection.

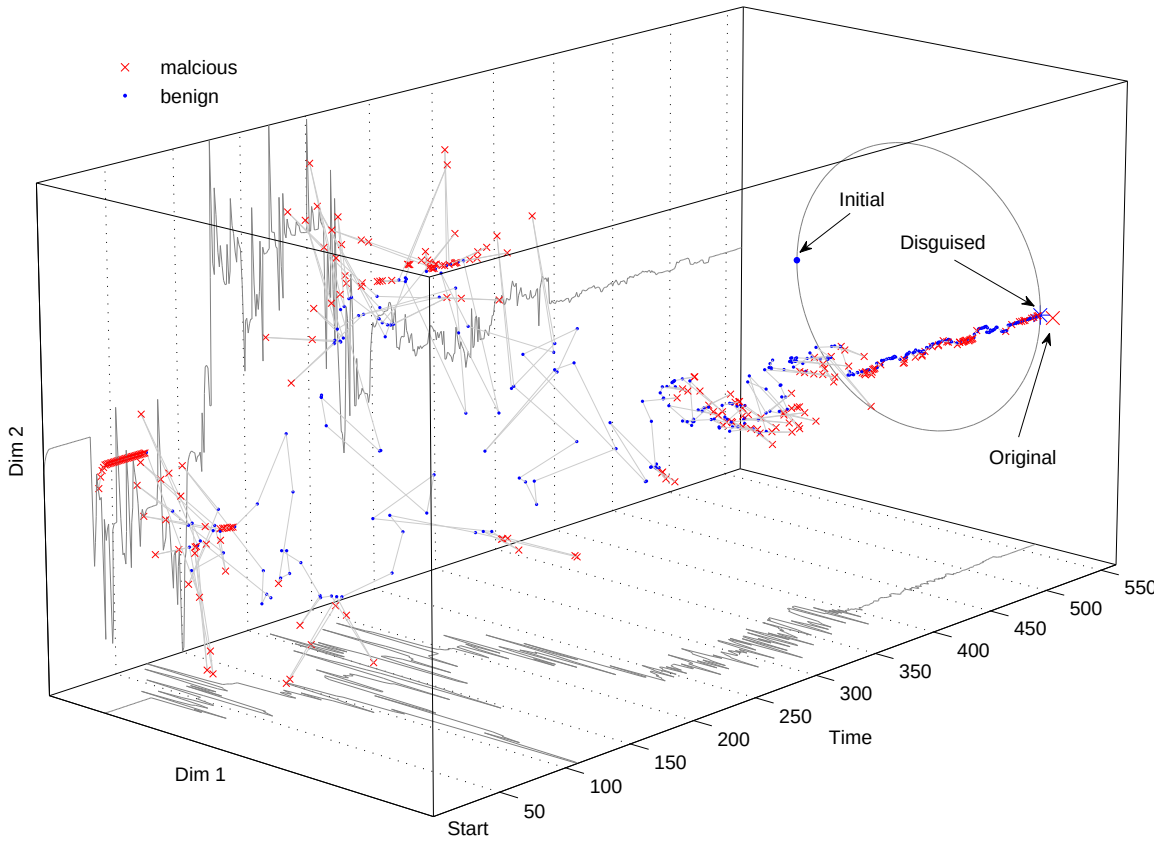


Figure 5.9.: A time series plot of an exploratory. The benign set is the interior of the circle. Although the plot is quite jogged, the convergent trend is evident on both dimensions.

5.8. Conclusion

In this chapter, I proposed an exploratory attack algorithm on the classifier with the convex benign set. The main ingredient of the algorithm is walking randomly in a shrinking convex body. The convergence property is thoroughly studied from a geometric perspective. Experimental results show that the proposed approach converges significantly faster than the previous centroid cutting plane method. A defense mechanism is also discussed.

The proposed algorithm works in much boarder situations. The cost function can has the form $g(\mathbf{x}) := \|\mathbf{A}\mathbf{x} - \mathbf{y}^{\mathrm{m}}\|_{\ell_p}$, where $\mathbf{A}$ is a $D \times D$ matrix. This allows some features may be more important than others. In this case the convergence rate depends on $\mathbf{A}$. Moreover, if the adversary is unaware of which set is convex, then he can simply run the proposed algorithm and previous methods and selects the best solution. For general classifiers without any convexity property, the proposed algorithm can be applied for searching local optimal solutions.

The efficiency of the proposed algorithm can be further improved if the adversary has prior knowledge about the geometry of the convex body. For instance, the adversary can exploit queried samples to train a local classifier. Then the geometry structure of the that classifier can be used to guide random walks in the most improving direction.

Chapter 6

Causative Label-Flip Attack on Support Vector Machines

As it is mentioned in Chapter 3, two types of attack that are interesting for my research, i.e. the exploratory and causative attack. In Chapter 4 and Chapter 5, I have presented a comprehensive analysis of the exploratory attack. In this chapter, I address the problem of causative attack. Recall that a *causative* attack aims to subvert the learning process of a model by controlling the training data [7]. For example, the adversary flags every legitimate mail as spam while the defender is gathering the training data. Consequently, the spam filter trained on such data is likely to cause a false alarm and may block all legitimate mails [120, 116].

The causative attack has recently attracted growing interest from the scientific community due to its long-lasting impact on learning algorithms. In general, if one attempt to harness human resources for training models, then the training data is in danger of contamination. Specifically, the adversary can carry out the causative attack either by introducing *feature noise* or *label noise* to the training data. Different types of feature noise have been extensively studied in several literature [48, 71, 107, 116]. However, little is known on how adversarial label noise is induced. Most of previous work either assume that labels are erased at random [22], or they restrict the underlying distribution of label noise to certain families without considering the attack strategy from the adversary's perspective [49, 97]. Recently, a label flips strategy based on heuristics is proposed to attack support vector machines (SVMs) [12].

This chapter focuses on a special causative attack called *adversarial label flips attack*. It abstracts a scenario where the adversary contaminates the training data through flipping labels in the supervised learning setting. More exactly, the adversary aims to find a combination of label flips under a given *budget* so that a classifier trained on such data will have maximal classification error. Motivated by Tikhonov regularization, I present an optimization framework for solving this problem. I then devise an algorithm for attacking support vector machine, which can be efficiently solved as two minimization problems. Experiments demonstrate that the attack maximally degrades the accuracy of SVMs with different kernels.

The rest of this chapter is organized as follows. The problem of adversarial label flips is described in Section 6.1. A framework for finding the near-optimal label flips is presented in Section 6.2. The algorithm for attacking SVMs is derived in Section 6.3, followed by experimental results on both synthetic and real-world data in Section 6.4. Section 6.5 provides conclusions and

discussions.

6.1. Problem Formulation

In the supervised classification problem, the training set of n instances is denoted as $S := \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y}\}_{i=1}^n$, with the *input space* $\mathcal{X}$ and the *label space* $\mathcal{Y} := \{-1, 1\}$. Given a *hypothesis space* $\mathcal{H}$ and a *loss function* V, the goal is to find a classification hypothesis $f_S \in \mathcal{H}$ by solving Tikhonov regularization problem

$$f_S := \arg\min_f \gamma \sum_{i=1}^{n} V\left(y_i, f(\mathbf{x}_i)\right) + \|f\|_{\mathcal{H}}^2, \tag{6.1}$$

where f_S denotes the classifier trained on S, and γ is a fixed positive parameter for quantifying the trade off. Remark that the first term in Eq. (6.1) reflects the empirical loss of f on S, and the second term reflects the generalization ability of f. Given an instance $\mathbf{x} \in \mathcal{X}$, the classification decision is made according to the sign of $f_S(\mathbf{x})$.

To express the label flips, I first introduce a set of variables $z_i \in \{0, 1\}, i = 1, \ldots, n$. Then replace y_i with $y_i' := y_i(1 - 2z_i)$ so that if $z_i = 1$ then the label is flipped $y_i' = -y_i$, otherwise $y_i' = y_i$. Denote $S' := \{(\mathbf{x}_i, y_i')\}_{i=1}^n$ the *tainted* training set, which shares the same instances as S but with some flipped labels. The adversary constructs S' in such a way that the resulting $f_{S'}$ yields maximal loss on some test set T. Thus, the problem of finding the near-optimal label flips can be formulated as

$$\max_{\mathbf{z}} \quad \sum_{(\mathbf{x},y)\in T} V\left(y, f_{S'}(\mathbf{x})\right), \tag{6.2}$$

$$\text{s.t.} \quad f_{S'} \in \arg\min_f \gamma \sum_{i=1}^{n} V\left(y_i', f(\mathbf{x}_i)\right) + \|f\|_{\mathcal{H}}^2, \tag{6.3}$$

$$\sum_{i=1}^{n} c_i z_i \le C, \quad z_i \in \{0, 1\}, \tag{6.4}$$

where $c_i \in \mathbb{R}_{0+}$ is the cost (or risk) of flipping label y_i from the adversary's viewpoint. Constraint Eq. (6.4) limits the total adversarial cost of label flips to C. Unfortunately, the above *bilevel* optimization problem is intrinsically hard due to the conflict and the interaction between Eq. (6.2) and Eq. (6.3). The conflict arises from the fact that for a given training set the defender learns a classifier with minimal empirical loss and good generalization ability, whereas the adversary expects that the classifier has maximal loss and poor generalization ability. That is, the beneficial outcome in one of them is associated with a detrimental outcome in another. Moreover, since any single flipped label may lead to a change to the classifier, the greedy strategy that flips labels based merely on the current classifier is ineffective. Essentially, the adversary has to evaluate each combination of label flips and selects the one that deteriorates the classifier the most.

As solving even the simplest linear bilevel problem is strong $\mathcal{NP}$-hard [161] and an exhaustive search on all combinations of flips is prohibitive, I resort to a relaxed formulation of finding the near-optimal label flips. In particular, I assume that the adversary only maximizes the empirical loss of the classifier on the original training set, yet indulges the defender in maximizing the generalization ability of the classifier. To obtain a set of label flips that jointly deteriorates the clas-

sifier's performance to the greatest extent, the adversary must foresee the reaction of the defender to the flipped labels. With these considerations in mind, I relax the original bilevel problem and present a loss minimization framework in the next section.

6.2. Label Flip Attack Framework

Let A and B be two sets of labeled instances, I first define an auxiliary loss function

$$g(B, f_A) := \gamma \sum_{(\mathbf{x},y)\in B} V\left(y, f_A(\mathbf{x})\right) + \|f_A\|_{\mathcal{H}}^2, \tag{6.5}$$

where f_A denotes the classifier trained on A. Note that the first term in Eq. (6.5) reflects the empirical loss incurred by f_A over the set B, which differs from Eq. (6.1).

To maximally degrade the classifier's performance, S' is selected in a way such that it has maximal loss under the original classifier f_S but yields minimal loss under the tainted classifier $f_{S'}$. The intuition is as follows: the adversary shifts the classification hypothesis so that the "terribly" mislabeled instances in S' asserted by the original classifier are now identified as "perfectly" labeled instances by the tainted classifier. With this strategy, the adversary can proactively cause the defender to produce a classifier whose loss is low on S' but high on S, which in turn has high loss on the test set. Formally, this idea can be represented as

$$\begin{aligned} \min_{\mathbf{z}} \quad & g(S', f_{S'}) - g(S', f_S), \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i z_i \leq C, \quad z_i \in \{0, 1\}. \end{aligned} \tag{6.6}$$

Remark that given *any* training set the defender *always* finds the optimal classifier by solving Tikhonov regularization problem. Thus, the first term in Eq. (6.6) reflects the defender's destined action on the training set S'. The second term quantifies the empirical loss on S' using the classifier f_S trained on the original set S, which represents the adversary's strategy of selecting instances with high loss.

I further refine the objective function and constraints of Eq. (6.6) for the algorithmic convenience. Denote U the expanded representation of S so that each instance in S is duplicated with a flipped label. Formally, the set $U := \{(\mathbf{x}_i, y_i)\}_{i=1}^{2n}$ is constructed as follows

$$\begin{aligned} (\mathbf{x}_i, y_i) &\in S, \quad i = 1, \ldots, n, \\ \mathbf{x}_i &:= \mathbf{x}_{i-n}, \quad i = n+1, \ldots, 2n, \\ y_i &:= -y_{i-n} \quad i = n+1, \ldots, 2n. \end{aligned}$$

I introduce an indicator variable $q_i \in \{0, 1\}, i = 1, \ldots, 2n$ for each element in U, where $q_i = 1$ denotes that $(\mathbf{x}_i, y_i) \in S'$, and $q_i = 0$ denotes that it is not. Replace S' by U and substitute

Eq. (6.5) into Eq. (6.6), one can rewrite the near-optimal label flips problem as

$$\begin{aligned} \min_{\mathbf{q},f} \quad & \gamma \sum_{i=1}^{2n} q_i \left[V\left(y_i, f(\mathbf{x}_i)\right) - V\left(y_i, f_S(\mathbf{x}_i)\right) \right] + \|f\|_{\mathcal{H}}^2, \\ \text{s.t.} \quad & \sum_{i=n+1}^{2n} c_i q_i \leq C, \\ & q_i + q_{i+n} = 1, \quad i = 1, \ldots, n, \\ & q_i \in \{0, 1\}, \quad i = 1, \ldots, 2n. \end{aligned} \tag{6.7}$$

The term $\|f_S\|_{\mathcal{H}}^2$ can be ignored as it is a constant with respect to the optimization variables. Indicator variables $q_{n+1}, \ldots, q_{2n}$ correspond to $z_1, \ldots, z_n$ in the previous bilevel formulations, respectively. The constraint $q_i + q_{i+n} = 1$ reflects that only one label can be chosen for the instance $\mathbf{x}_i$. Due to the acquiescence on the defender's behavior of maximizing the generalization ability of the tainted classifier, the conflicting objectives of the defender and the adversary are now incorporated into one minimization problem. Given a training set one can employ the above framework to compute the set of label flips that will jointly degrade the classifier's accuracy without exceeding a specified budget. Recall that SVMs can be considered as a special case of Tikhonov regularization, it is straightforward to develop an attack on SVMs subject to this framework, as the reader will see in the next section.

6.3. Attack on SVM

SVMs project the original training instances from the input space $\mathcal{X}$ to the *feature space* $\mathcal{F}$ by $\Phi : \mathcal{X} \to \mathcal{F}$. In general, SVMs trained on S has the form

$$f_S(\mathbf{x}) := \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b,$$

where K is a *Mercer Kernel* which satisfies the property $K(\mathbf{x}, \mathbf{x}_i) = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}_i)$ and $b \in \mathbb{R}$ denotes the bias. The classifier can be also rewritten as

$$f_S(\mathbf{x}) := \mathbf{w}^\top \mathbf{x} + b,$$

where $\mathbf{w} := \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i)$ and $\mathbf{w} \in \mathcal{F}$. Thus, the classification boundary of a SVM is a hyperplane in $\mathcal{F}$ with normal vector $\mathbf{w}$. Given the *hinge loss* function $V\left(y, f(\mathbf{x})\right) := \max(0, 1 - yf(\mathbf{x}))$, Tikhonov regularization for SVMs is a constrained quadratic programming (QP) problem

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \gamma \sum_{i=1}^{n} \xi_i + \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n, \end{aligned} \tag{6.8}$$

where ξ_i represents the hinge loss of $(\mathbf{x}_i, y_i)$ resulting from the classifier f_S. Denote $\epsilon_i := \max(0, 1 - y_i f_{S'}(\mathbf{x}_i))$ the hinge loss of $(\mathbf{x}_i, y_i)$ resulting from the tainted classifier $f_{S'}$. By plugging

Eq. (6.8) into Eq. (6.7), yields

$$\begin{aligned} \min_{\mathbf{q},\mathbf{w},\boldsymbol{\epsilon},b} \quad & \gamma \sum_{i=1}^{2n} q_i(\epsilon_i - \xi_i) + \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0, \quad i = 1, \ldots, 2n, \\ & \sum_{i=n+1}^{2n} c_i q_i \leq C, \\ & q_i + q_{i+n} = 1, \quad i = 1, \ldots, n, \\ & q_i \in \{0, 1\}, \quad i = 1, \ldots, 2n. \end{aligned} \tag{6.9}$$

Observe that Eq. (6.9) involves an integer programming problem which is in general $\mathcal{NP}$-hard. Therefore, I first relax it into a continuous optimization problem by allowing all q_i to take values between $[0, 1]$. Then I decompose Eq. (6.9) into two sub-problems and devise an iterative approach to minimize them alternatively. On the one hand, by fixing $\mathbf{q}$, the minimization over $\mathbf{w}, \boldsymbol{\epsilon}, b$ is reduced to the following QP problem

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\epsilon},b} \quad & \gamma \sum_{i=1}^{2n} q_i \epsilon_i + \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0, \quad i = 1, \ldots, 2n. \end{aligned} \tag{6.10}$$

On the other hand, by fixing $\mathbf{w}, b$ and using the computed $\boldsymbol{\epsilon}$ the minimization over $\mathbf{q}$ can be described as a linear programming (LP) as follows

$$\begin{aligned} \min_{\mathbf{q}} \quad & \gamma \sum_{i=1}^{2n} q_i(\epsilon_i - \xi_i) \\ \text{s.t.} \quad & \sum_{i=n+1}^{2n} c_i q_i \leq C, \\ & q_i + q_{i+n} = 1, \quad i = 1, \ldots, n, \\ & 0 \leq q_i \leq 1, \quad i = 1, \ldots, 2n. \end{aligned} \tag{6.11}$$

It is easy to see that by minimizing Eq. (6.10) and Eq. (6.11) the objective function Eq. (6.9) decreases monotonically. Note that ξ_i can be computed beforehand, the algorithm can be implemented efficiently with off-the-shelf QP and LP solvers. After the algorithm converges, I greedily select the largest subset of $\{q_{n+1}, \ldots, q_{2n}\}$ meeting the given budget and flip the corresponding labels. The complete procedure is summarized in Fig. 6.1, which I denote as ALFA.

6.4. Experiments

I demonstrate the label flips attack on SVMs with linear kernel and radial basis function (RBF) kernel using two sets of experiments. First, I employed some two-dimensional synthetic data to visualize the decision boundaries of SVMs under the label flips. The second set of experiments was conducted on ten real-world data sets, where I concentrated the influence of label flips on SVMs

Input : original training set S, adversarial cost $c_1, \ldots, c_n$, budget C, parameter γ
Output: tainted training set S' with flipped labels

Find f_S by solving Eq. (6.8) on S; /* QP */
foreach $(\mathbf{x}_i, y_i) \in U$ **do**
    $\xi_i \leftarrow \max(0, 1 - y_i f_S(\mathbf{x}_i))$;
    $\epsilon_i \leftarrow 0$;
repeat
    Find $q_1, \ldots, q_{2n}$ by solving Eq. (6.11); /* LP */
    Find $\epsilon_1, \ldots, \epsilon_{2n}$ by solving Eq. (6.10); /* QP */
until *convergence*;
$L \leftarrow$`Sort(`$[q_{n+1}, \ldots, q_{2n}]$, *"desc"*`)`;
`/* L is an array of sorted indices */`
for $i \leftarrow 1$ **to** n **do** $y'_i \leftarrow y_i$;
;
$j \leftarrow 1$;
while $\sum_{i=1}^{j} q_{L[i]} \leq C$ **do**
    $y'_{L[j]-n} \leftarrow -y_{L[j]-n}$; /* Flip label */
    $j \leftarrow j + 1$;
return $S' \leftarrow \{(\mathbf{x}_i, y'_i)\}_{i=1}^{n}$;

Figure 6.1.: Adversarial Label Flips Attack on SVMs (ALFA)

with respect to different budgets. In all experiments, the proposed ALFA was compared with the following three label flip strategies

- Uniform random flip: instances are uniformly chosen at random from the training set and their labels are flipped. This can be regarded as introducing label noise to the training set from the non-adversarial perspective.
- Nearest-first flip: instances that have small distances to the decision hyperplane in the feature space are first flipped. This corresponds to a thoughtless labeler who erroneously labels instances that are difficult to be distinguished.
- Furthest-first flip: instances that have large distances to the decision hyperplane in the feature space are first flipped. In this way, I can simulate a malicious labeler who deliberately gives wrong labels on instances that are easy to be distinguished.

The adversarial cost was set as $c_i := 1$ for $i = 1, \ldots, n$. Thus, given a budget C one can flip at most $\min(\lfloor C \rfloor, n)$ labels. Experiments were conducted as follows. First, I randomly selected the same number of instances from two classes and construct the training set and the test set, respectively. Second, the training set was tainted by performing different flip strategies. Third, I trained SVMs (with $\gamma := 1$) on the original training set and four tainted training sets. Finally, the classification error of each SVM was measured on the test set, respectively. As the test set is balanced, the worst performance of a classifier is with 50% error rate, which corresponds to the random guess. Hence, an error rate around 50% indicates an effective attack strategy on SVMs.

In the experiments, the convergence of ALFA typically occurred in $5 \sim 10$ iterations. On a training set with 300 instances, the MATLAB implementation[1] without special code-level optimization takes about 3 seconds for computing the near-optimal label flips[2].

6.4.1. Synthetic Examples

I generated linear and parabolic patterns in two dimensional space for this experiment. From each pattern, I selected 100 instances as the training set and 800 instances as the test set. Let $C := 20$, decision boundaries of SVMs under different flip strategies are illustrated in Fig. 6.2.

By comparing Fig. 6.2(b) with Fig. 6.2(f), one can clearly observe the dramatic changes on decision boundaries of SVMs under ALFA. For instance, the original decision plane of linear SVM on the parabolic pattern is almost tilted by 90 degrees under ALFA (see the 3^{rd} row of Fig. 6.2). Moreover, when ALFA is applied to SVMs with RBF kernel, the error rate increases from 3.2% to 32.4% on the linear pattern and 5.1% to 40.8% on the parabolic pattern. Not surprisingly, the nearest-first strategy is least effective due to the tolerance nature of soft-margin SVMs. While the furthest-first strategy increases the classification error as well, it is less compelling than ALFA. Further note that the performance of SVMs is quite stable under the uniform random label noise and the error rate hardly changes with 20 flipped labels, as shown in Fig. 6.2(c). This implies that previous robust learning algorithms based on the assumption of random label noise may be too optimistic as they underestimate the adversary's impact on the classifier's performance.

6.4.2. On Real-World Data

I continue the investigation of different flip strategies using 10 real-world data sets, which were downloaded from LIBSVM website. For each data set, I randomly selected 200 instances as the training set and 800 instances as the test set. As in practice the adversary usually controls only a small portion of the training data, I demonstrated the effectiveness of label flips with respect to different budgets, especially with low budget.

Figure 6.3 depicts the error rate of SVMs up to 60 label flips (i.e. $C := 1, \ldots, 60$). As expected, the error rate of SVMs increases with the growth of label flips. While SVMs sometimes show the resilience to the random label noise, the error rate significantly increases under ALFA and the furthest-first strategy due to their adversarial nature. The advantage of ALFA is most significant when SVMs are trained with RBF kernel. On many data sets, by flipping only 20 labels (i.e. 10% of training data) with ALFA the error rate of RBF-SVM rises to 50%, which is turned into the random guess. Moreover, I remark that ALFA is more cost-effective than the furthest-first strategy especially with small flips. When the number of flipped labels is large, ALFA keeps trapping SVMs with worst performance at 50% error rate. On the contrary, the furthest-first strategy increases the error rate over 50% (see Fig. 6.3(b) `a9a`,`connect-4`,`letter`), which in fact regains the predictive power of SVMs. This behavior is due to the fact that the proposed framework captures the classifier's reaction to flipped labels, whereas the furthest-first strategy merely considers the information about the current classifier.

[1] MATLAB implementation and more experimental results are available at `http://home.in.tum.de/~xiaoh`

[2] I tried an exhaustive search to find the groundtruth optimal label flips. For example, To obtain the optimal 20 label flips out of 300 training instances, the program has to check over 7×10^{30} combinations. Due to the extremely slow progress, I terminated the program after one month running on a 12-cores workstation.

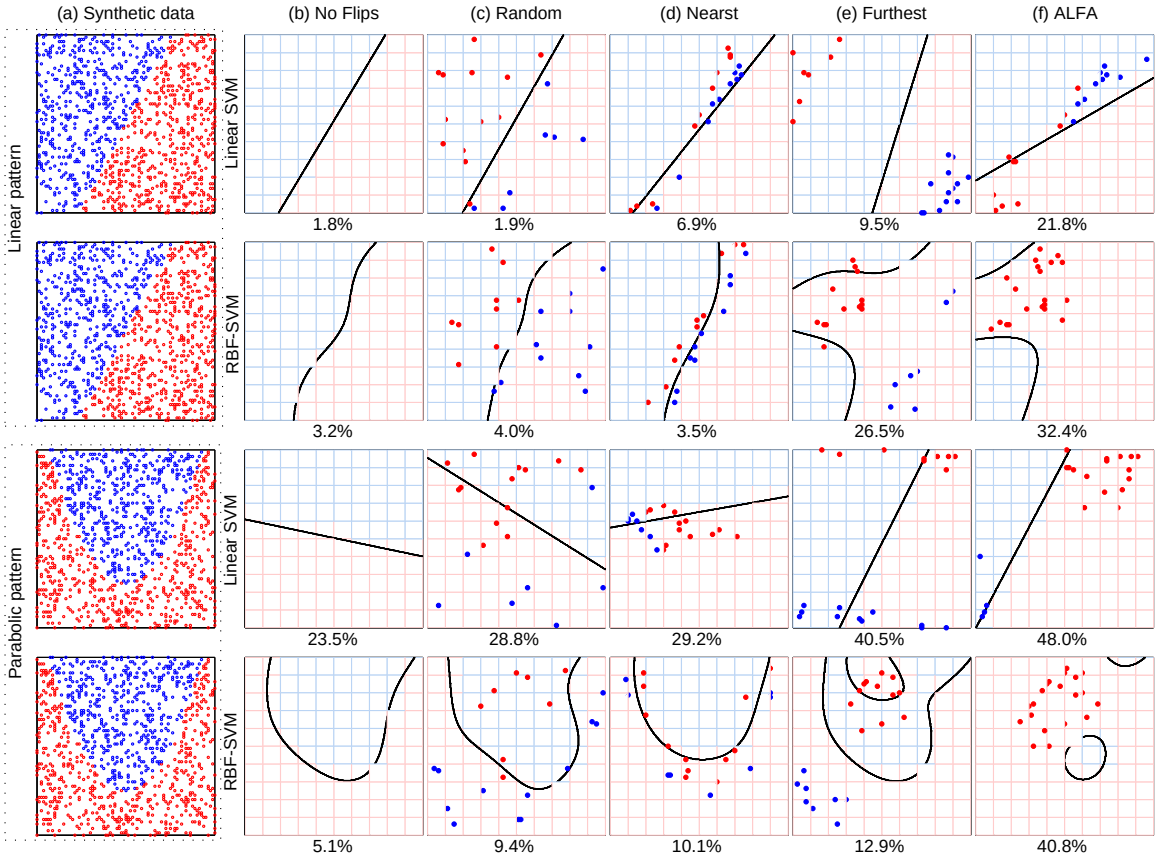
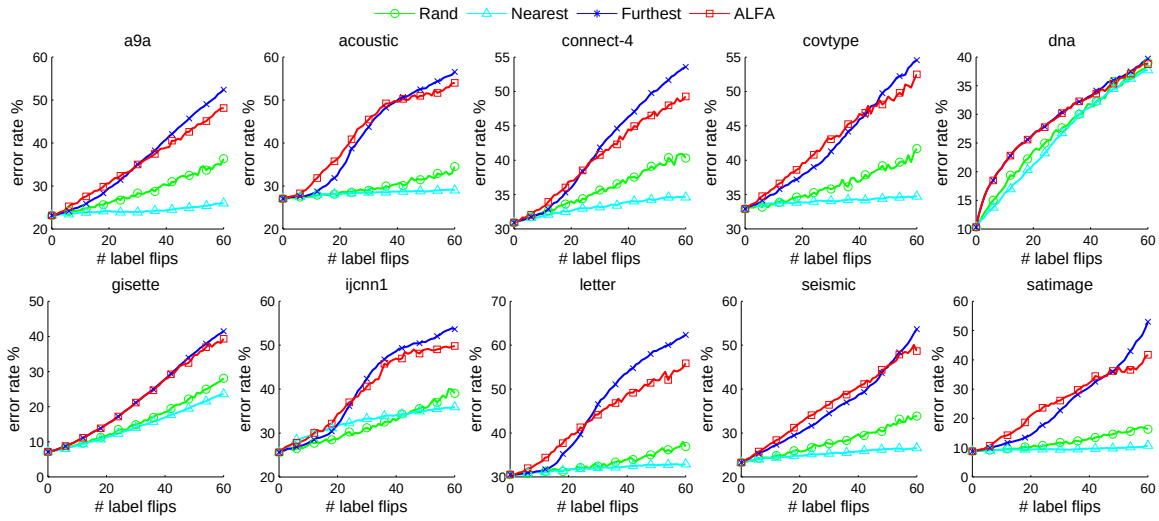
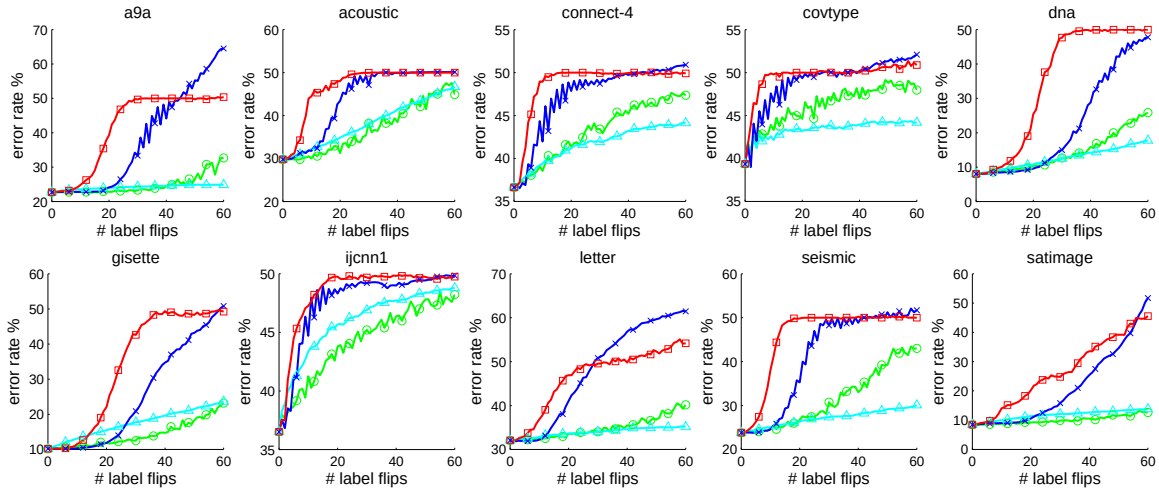


Figure 6.2.: Decision boundaries of SVMs under different flip strategies. The first and second rows illustrate results on the linear pattern, the third and fourth rows illustrate results on the parabolic pattern. For each strategy, the number of flipped labels is fixed to 20 (i.e. 20% of the training data). Each point represents an instance. Labels are denoted in red and blue. In each plot, decision regions of SVMs are shaded in different colors. Only flipped instances in the training set are highlighted. The percentage under each plot indicates the error rate of SVM measured on the test set, respectively. **(a)** The synthetic data generated for the experiment. **(b)** Decision boundaries of SVMs trained on the original training set without label flips. **(c)** Decision boundaries of SVMs under random label flips. **(d)** Decision boundaries of SVMs under nearest-first flip strategy. **(e)** Decision boundaries of SVMs under furthest-first flip strategy. **(f)** Decision boundaries of SVMs under ALFA.

From the perspective of a cost-averse adversary, it is also interesting to know the required budget for turning a SVM into a random guess. Table 6.1 shows the required percentage of label flips when the tainted SVM reaches 50% error rate on the test set. First of all, observe that the required percentage of label flips greatly depends on data sets, or how training instances are distributed in the feature space. Moreover, comparing with the linear kernel it is easier to taint SVMs with RBF kernel. This is because by mapping instances to the infinite dimensional feature space, instances are more sparsely distributed. Hence, flipping a label will result a significant change on the separating hyperplane. Furthermore, in both cases ALFA flips less labels than other strategies.



(a) Error rate of SVMs with linear kernel under different flip strategies.



(b) Error rate of SVMs with RBF kernel under different flip strategies.

Figure 6.3.: Error rate of SVMs as a function of the number flipped labels. Within each experiment, the training set consists of 200 instances (100 for each class) selected randomly. The adversary can flip at most 60 labels (i.e. 30% of the training data). The classification error is measured on 800 test instances with balanced labels. Results are averaged over 60 repetitions. Note that 50% error rate corresponds to the random guess.

For the linear kernel the required percentage of label flips is roughly stable with respect to the size of the training set. That is, the required flips rises linearly when the size of training set increases. On the contrary, for RBF kernel the required percentage increases as the training set becomes larger.

Finally, I adapted ALFA to attack the label noise robust SVM (LN-SVM) based on a simple kernel matrix correction [12]. The experiment indicates that, although LN-SVM shows resilience to the random noisy labels, it still greatly suffers from ALFA.

Table 6.1.: The percentage of flipped labels when a SVM reaches 50% error rate. Experiment is conducted on ten data sets with 100, 200 and 300 training instances, respectively. The classification error is measured on the randomly selected test set with 800 instances. From the adversary's viewpoint, smaller percentage value indicates a more cost-effective flip strategy as it requires lower budget. For each data set, the most effective strategy is highlighted with the boldface. Results are averaged over 60 repetitions.

	100				200				300			
Data sets	Rand.	Near.	Furt.	ALFA	Rand.	Near.	Furt.	ALFA	Rand.	Near.	Furt.	ALFA
SVM with linear kernel												
a9a	41.9	70.4	**29.5**	31.5	43.7	72.2	**27.1**	29.8	44.5	72.9	**26.7**	29.9
acou.	38.5	77.6	19.2	**17.1**	41.5	77.4	18.8	**17.3**	42.5	76.6	18.8	**17.4**
conn.	38.2	67.7	**27.7**	29.1	40.1	73.7	**24.4**	27.5	42.2	77.3	**21.4**	25.2
covt.	32.1	73.7	25.0	**23.8**	37.0	74.4	24.6	**22.6**	36.9	75.1	23.9	**21.7**
dna	**43.4**	47.6	50.7	47.8	**42.5**	51.6	45.8	44.2	43.5	54.6	**42.6**	43.2
gise.	47.7	56.6	43.7	**43.6**	47.0	61.8	**37.9**	37.9	47.6	63.8	**35.6**	35.6
ijcn.	33.9	62.6	26.5	**25.4**	37.9	72.7	21.5	**20.8**	38.2	76.4	19.7	**17.6**
lett.	36.7	80.6	**18.2**	19.0	40.2	82.6	**17.1**	18.6	41.5	82.1	**17.4**	19.1
seis.	38.7	73.8	26.3	**25.5**	40.7	71.3	**28.3**	28.7	41.3	70.7	28.8	**28.1**
sati.	44.5	70.5	**30.0**	32.2	45.4	70.3	29.8	**25.5**	46.4	69.2	30.6	**22.3**
SVM with RBF kernel												
a9a	21.6	65.3	12.8	**7.7**	31.5	74.9	18.8	**12.0**	36.1	76.1	20.4	**14.1**
acou.	6.3	14.7	4.1	**2.9**	16.3	36.8	10.2	**7.1**	22.6	52.7	13.7	**7.8**
conn.	7.2	33.8	3.7	**2.8**	18.5	68.8	8.7	**5.3**	25.2	76.2	12.3	**6.8**
covt.	2.5	13.2	1.8	**1.4**	6.6	55.8	4.3	**2.2**	11.6	71.2	7.3	**3.9**
dna	27.6	53.6	20.8	**11.6**	40.9	63.7	31.6	**17.0**	46.7	66.5	32.6	**19.2**
gise.	29.4	68.9	23.4	**14.1**	38.7	70.8	28.4	**17.8**	43.4	69.2	29.0	**19.3**
ijcn.	8.1	27.2	4.2	**3.5**	19.4	41.0	13.6	**8.4**	25.0	40.3	20.4	**10.4**
lett.	22.6	78.0	11.7	**8.0**	31.0	84.4	14.1	**10.9**	35.3	84.5	14.2	**11.9**
seis.	11.0	33.4	6.4	**4.3**	24.0	64.4	13.5	**7.4**	29.3	69.0	16.4	**9.6**
sati.	39.1	69.2	25.5	**23.7**	41.8	68.8	28.7	**22.3**	43.4	67.8	30.3	**23.3**

6.5. Conclusion

When the hope is to develop a robust learning algorithm under adversarial conditions, it is incumbent on the researchers to understand the adversary's strategy. Throughout this chapter, I have investigated the problem of adversarial label flips in the supervised learning setting, where an attacker contaminates the training data through flipping labels. I present an optimization framework for the adversary to find the near-optimal label flips that maximally degrades the classifier's performance. The framework simultaneously models the adversary's attempt and the defender's reaction in a loss minimization problem. Based on this framework, I develop an algorithm for attacking SVMs. Experimental results demonstrate the effectiveness of the proposed attack on both synthetic and real-world data set. While solving problems for adversaries may seem counterproductive, I believe that investigating the strategy of the adversary and the vulnerability of the defender is the only way to develop a robust learning algorithm in the future.

Comparing with the random label noise, the adversarial label noise has been shown to be more influential to the classifier's performance. Thus, the proposed framework can be used as a baseline for evaluating the robustness of a learning algorithm under the noisy condition. The framework can be also extended to the active learning and online learning settings, where labels

are usually committed by massive annotators with various motivations. Another relevant scenario is the crowdsourcing platform (e.g. Amazon's Mechanical Turk), where the labeled data can be obtained quickly from crowds of human workers. In such settings, the adversarial label noise is inevitable due to the limitation of quality control mechanisms. It would be interesting to formulate this learning problem as a n-player hybrid game, which contains both cooperative and non-cooperative players. By categorizing players into coalitions and modeling the worst-case behavior of each coalition, one may develop an algorithm that learns from good labelers yet shows resilience to malicious labelers. The algorithm in the next chapter, which is motivated by this idea, can effectively solve the problem of learning from multiple annotators/users/teachers, or more generally, "observers". In Chapter 7 and Chapter 8, the reader will reconsider adversarial learning from the defender's point of view. From now on, the focus of the dissertation will be shifted from examining the vulnerabilities to building reliable learning algorithms.

Part IV.

Reliable Learning Algorithms

One person's data is another person's noise.

K.C. Cole

Chapter 7

Learning from Multiple Observers with Unknown Expertise

In Chapter 6, I have demonstrated the label noise in the training data can significantly subvert the learning process of a model, thereby degrade its accuracy. As a consequence, it is natural for researchers to seek for more robust learning algorithms that are resilient to the adversarial noise. An intuitive solution is to harness not only a single label, but multiple labels for each instance to train the model. In this chapter, I realize this idea by proposing a hierarchical Gaussian process model that can effectively learn from multiple, but unreliable labels.

The motivation of building such model is also inspired by the recent advent of social web services. Nowadays, data can be shared and processed by a large number of users. As a consequence, researchers are faced with data sets that are labeled by multiple users. For example, Wikipedia provides a feedback tool to engage readers in the assessment of article quality based on four criteria, i.e. "trustworthy", "objective", "complete" and "well-written". The Amazon Mechanical Turk is an online system that allows the requesters to hire users from all over the world to perform crowdsourcing tasks. Galaxy Zoo is a website where visitors label astronomical images. While providing large amounts of cheap labeled data in a short time, these platforms usually have little quality control over users. Thus, the response of each user can vary widely, and in some cases may even be adversarial. A natural question to ask is how to integrate opinions from multiple users for obtaining an objective opinion. The commonly used "majority vote" and "take the average" heuristics completely ignore the individual expertise and may fail in the settings with non-Gaussian or adversarial noise. This casts a challenge of *learning from multiple sources* for the machine learning and data mining researchers [34].

Despite these web applications, one can find this problem in wide range of domains. Recently, *sensor networks* have been deployed for the scientific monitoring of remote and hostile environments. For example, researchers deployed a 16-node sensor network on a tree to study its elevation under different weather fronts [158]. Each node samples climate data at regular time intervals and the statistics are collected. Using sensor data in this manner presents many novel challenges, such as fusing noisy readings from several sensors, detecting faulty and aging sensors. Importantly, it is necessary to use the trends and correlations observed in previous data to predict the value of environmental parameters into the future, or to predict the reading of a sensor that is temporarily unavailable (e.g. due to network outages). However, these tasks may have to be performed with

only limited knowledge of the location, reliability, and accuracy of each sensor.

In this chapter, the labeler (including user, annotator and sensor) mentioned above is referred to as the *observer*. Given an *instance*, the label (e.g. annotation, reading) provided by an observer is called the *response*. Unlike the conventional supervised learning scenario, in the proposed setting each instance is associated with a set of responses, yet the *ground truth* is unknown as some responses may be subjective or come from unreliable observers. I concentrate on the regression problem with continuous responses from multiple observers. Specifically, the proposed method provides a principled way to answer the following questions:

1. How to learn a regression function to predict the ground truth precluding the prior knowledge of observers?

2. How to estimate the expertise of each observer without knowing the ground truth?

7.1. Related Work

There is a number of studies dealing with the setting involving multiple labelers, yet most of them focus on the classification problem. Early work such as [47, 85, 148] focus on estimating the error rates of observers. In the machine learning community, the problem of estimating the ground truth from multiple noisy labels is addressed in [145]. Instead of estimating the ground truth and learning the classifier separately, recent interest has shifted towards on learning classifiers directly from such data. Authors of [34] provide a general theory of selecting the most informative samples from each source for model training. Later, a probabilistic framework is presented by [130, 131] to address the classification, regression and ordinal regression problem with multiple annotators. The framework is based on a simple assumption that the expertise of each annotator does not depend on the given data. This assumption is infringed in [167, 176] and later is extended to the active learning scenario [175]. There are some other related work that focus on different settings [26, 172].

The above studies paid little attention to the regression problem under multiple observers, which is the main core of this chapter. Moreover, the proposed work differs from the related work in various aspects. First, I employ a less-parametric method, i.e. the *Gaussian process* (GP), to model the observers and the regression function. This allows one to associate the observer's expertise with both ground truth and input instance. Moreover, the proposed model is presented in an extensible probabilistic framework. The missing data and prior knowledge can be straightforwardly incorporated into the model.

The rest of this chapter is organized as follows. Section 7.2 formulates the problem and introduces a probabilistic framework. The framework consists of two parts. The regression model is introduced in Section 7.2.2. A linear and a non-linear observer model is proposed in Section 7.2.3 and Section 7.2.4, respectively. Section 7.3 reports the experimental results on both synthetic and real-world data sets. Conclusions are drawn in Section 7.4.

7.2. Problem Formulation

Denote the *instance space* $\mathcal{X} \subseteq \mathbb{R}^L$ and the *response space* $\mathcal{Y} \subseteq \mathbb{R}^D$ and the *ground truth space* $\mathcal{Z} \subseteq \mathbb{R}^D$. Given N instances $\mathbf{x}_1, \ldots, \mathbf{x}_N$ where $\mathbf{x}_n \in \mathcal{X}$, denote the *objective ground truth* for $\mathbf{x}_n$ as $\mathbf{z}_n \in \mathcal{Z}$. In the proposed setting, the ground truth is unknown. Instead, consider multiple

responses $\mathbf{y}_{n,1}, \ldots, \mathbf{y}_{n,M} \in \mathcal{Y}$ for $\mathbf{x}_n$ provided by M different observers. For compactness, the $N \times L$ matrix of instance $x_{n,l}$ is represented as $\mathbf{X} \coloneqq [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$. The $N \times M \times D$ tensor of observers' responses $y_{n,m,d}$ is denoted by $\mathbf{Y} \coloneqq [\mathbf{y}_{1,1}, \ldots, \mathbf{y}_{1,M}; \ldots; \mathbf{y}_{N,1}, \ldots, \mathbf{y}_{N,M}]$. The $N \times D$ matrix of ground truth $z_{n,d}$ is denoted by $\mathbf{Z} \coloneqq [\mathbf{z}_1, \ldots, \mathbf{z}_N]^\top$.

Given the training data $\mathbf{X}$ and $\mathbf{Y}$, the goal is threefold. First, it is of interest to get an estimate of the unknown ground truth $\mathbf{Z}$. The second goal is to learn a regression function $f : \mathcal{X} \to \mathcal{Z}$ which generalizes well on unseen instances. Finally, for each observer the goal is to model its *expertise* as a function of the input instance and the ground truth, i.e. $g : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$.

7.2.1. Probabilistic Framework

To formulate this problem from the probabilistic perspective, I consider the training data $\mathbf{X}$ and $\mathbf{Y}$ as random variables. The ground truth $\mathbf{Z}$ is unknown and hence is a latent variable. In general, the observed response $\mathbf{Y}$ depends both on the unknown ground truth and the instance. That is, observers may exhibit varying levels of expertise on different instances. On Wikipedia the assumption is particularly true for the novice readers, whereas the rating from an expert reader is consistent across different types of articles. Figure 7.1 illustrates the conditional dependence between $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ with a graphical model. As a consequence, the joint conditional distribution can be expressed as

$$\begin{aligned} p(\mathbf{Y}, \mathbf{Z}, \mathbf{X}) &= p(\mathbf{Z} \mid \mathbf{X}) p(\mathbf{Y} \mid \mathbf{Z}, \mathbf{X}) p(\mathbf{X}) \\ &\propto \prod_{n=1}^{N} \prod_{d=1}^{D} p(z_{n,d} \mid \mathbf{x}_n) \prod_{m=1}^{M} p(y_{n,m,d} \mid \mathbf{x}_n, z_{n,d}), \end{aligned} \tag{7.1}$$

where the term $p(\mathbf{X})$ is dropped as the other two conditional distributions are more interesting. There are two underlying assumptions in this model. First, each dimension of the ground truth is independent, but is not identically distributed. Second, all observers respond independently.

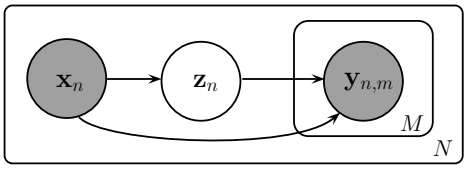


Figure 7.1.: Graphical model of instances $\mathbf{X}$, unknown ground truth $\mathbf{Z}$ and responses $\mathbf{Y}$ from M different observers. Only the shaded variables are observed.

[1]

Note that the first term in (7.1) indicates the probabilistic dependence between the ground truth and the input instance, whereas the second term characterizes the observers' expertise. Previous work have explored different parametric methods to model these two conditional distributions [167, 176, 130, 175, 131]. A distinguishing factor in this chapter is that, I employ the Gaussian process as the backbone to construct the model. Specifically, the generative process of $\mathbf{Y}$ can be

[1] While these are reasonable assumptions, they may not entirely true. For instance, in the article feedback of Wikipedia, an article scores high on "well-written" is often more "trustworthy". Moreover, readers share the same education background are more likely to produce similar ratings.

interpreted as follows

$$z_{n,d} = f_d(\mathbf{x}_n) + \epsilon_n, \tag{7.2}$$
$$y_{n,m,d} = g_{m,d}(\mathbf{x}_n, z_{n,d}) + \xi_{m,d}, \tag{7.3}$$

where ϵ and ξ is independent identically distributed Gaussian noise, respectively. Note that the choice of $\{f_d\}$ and $\{g_{m,d}\}$ characterizes the regression function and the observers, respectively. In particular, an ideal observer would have $g_{m,d}(z_{n,d}) = z_{n,d}$ on every d. Therefore, the goal can be understood as searching $\{f_d\}$ and $\{g_{m,d}\}$ given the training data. Intuitively, if two instances are close to each other in $\mathcal{X}$, then their corresponding ground truth should be close in $\mathcal{Z}$ through the mapping of $\{f_d\}$, which in turn restricts the searching space of $\{g_{m,d}\}$ when $\mathbf{Y}$ is known.

7.2.2. Regression Model

I first concentrate on Eq. (7.2) and represent functions $\{f_d\}$ by the Gaussian process with some non-linear kernel. Specifically, the conditional distribution of the ground truth given the training instances is assumed to be

$$p(\mathbf{Z} \,|\, \mathbf{X}) = \prod_{d=1}^{D} \mathcal{N}\left(\mathbf{z}_{:,d} \,|\, \mathbf{0}, \mathbf{K}_d\right), \tag{7.4}$$

where the d^{th} dimension of the ground truth is denoted as $\mathbf{z}_{:,d}$. I introduce a $N \times N$ kernel matrix $\mathbf{K}_d$ that depends on $\mathbf{X}$, where each element is given by the value of a composite covariance function $k_d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{0+}$, made up of several contributions as follows

$$k_d(\mathbf{x}_i, \mathbf{x}_j) := \kappa_{1,d}^2 \exp\left(-\frac{\kappa_{2,d}^2}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) + \kappa_{3,d}^2 + \kappa_{4,d}^2 \mathbf{x}_i^\top \mathbf{x}_j + \kappa_{5,d}^2 \delta(\mathbf{x}_i, \mathbf{x}_j). \tag{7.5}$$

The noise term ϵ in Eq. (7.2) is folded into the Kronecker delta function $\delta(\mathbf{x}_i, \mathbf{x}_j)$. The covariance function involves an exponential of a quadratic term, with the addition of a constant bias, a linear and a noise terms. For each dimension, the parameters need to be learned from the data are $\kappa_{1,d}, \ldots, \kappa_{5,d}$. Samples from this prior are plotted for various values of the parameters in Fig. 7.2.

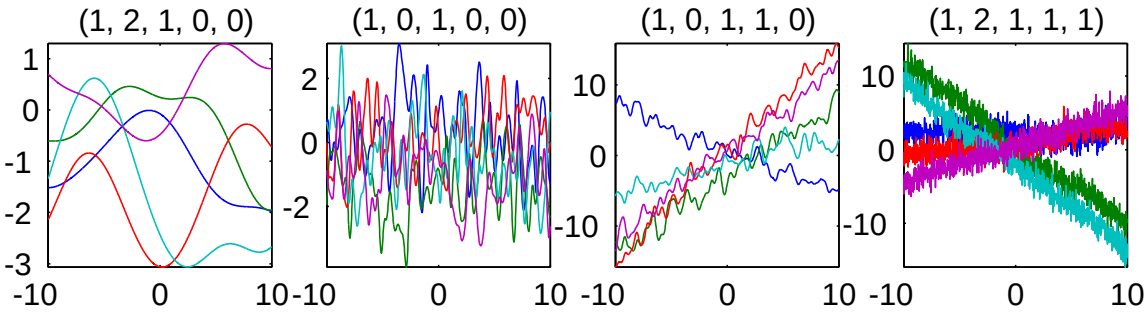


Figure 7.2.: Samples drawn from a Gaussian process prior defined by the covariance function Eq. (7.5). The title above each plot denotes the value of $(\kappa_{1,d}, \kappa_{2,d}, \kappa_{3,d}, \kappa_{4,d}, \kappa_{5,d})$. The samples are obtained using a discretization of the x-axis of 1000 equally spaced points.

7.2.3. Linear Observer Model

To model the observer's expertise, I now concentrate on (7.3) and assume that $\{g_{m,d}\}$ is a linear mapping from $\mathcal{Z}$ to $\mathcal{Y}$, which does not depend on the instance at all. Denote $\mathbf{y}_{:,m,d}$ the d^{th} dimension response of all training instances provided by the m^{th} observer. The second conditional distribution in (7.1) is assumed to be

$$p(\mathbf{Y} \mid \mathbf{Z}, \mathbf{X}) = p(\mathbf{Y} \mid \mathbf{Z}) = \prod_{m=1}^{M} \prod_{d=1}^{D} \mathcal{N}\left(\mathbf{y}_{:,m,d} \mid w_{m,d}\mathbf{z}_{:,d} + \mu_{m,d}\mathbf{1}, \sigma_{m,d}^2\mathbf{I}\right), \tag{7.6}$$

where $\mathbf{1}$ is an all-ones vector with length N and $\mathbf{I}$ is a $N \times N$ identity matrix. Each observer is characterized by $3 \times D$ parameters, i.e. $w_{m,d}, \mu_{m,d}, \sigma_{m,d} \in \mathbb{R}$.

Parameter Estimation

Now I can combine Eq. (7.6) with Eq. (7.4) and estimate the set of all parameters, i.e. $\boldsymbol{\Theta} \coloneqq \{\{\kappa_{1,d}, \ldots, \kappa_{5,d}\}, \{w_{m,d}\}, \{\mu_{m,d}\}, \{\sigma_{m,d}\}\}$, by maximizing the likelihood function $p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\Theta})$. In the linear observer model, the latent variable $\mathbf{Z}$ can be marginalized out, which yields

$$p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\Theta}) = \prod_{m=1}^{M} \prod_{d=1}^{D} \mathcal{N}\left(\mu_{m,d}\mathbf{1}, w_{m,d}^2\mathbf{K}_d + \sigma_{m,d}^2\mathbf{I}\right).$$

The maximum likelihood estimator of $\mu_{m,d}$ is given by $\widetilde{\mu}_{m,d} = \frac{1}{N}\sum_{n=1}^{N} y_{n,m,d}$. I hereinafter use the short-hand $\overline{\mathbf{y}}_{:,m,d} \coloneqq \mathbf{y}_{:,m,d} - \widetilde{\mu}_{m,d}\mathbf{1}$. As a consequence, the log-likelihood function is given by

$$\begin{aligned} F^{\text{LOB}} \coloneqq \log p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\Theta}) &= \sum_{m=1}^{M}\sum_{d=1}^{D} \log p(\mathbf{y}_{:,m,d} \mid \mathbf{X}, \boldsymbol{\Theta}) \\ &= \sum_{m=1}^{M}\sum_{d=1}^{D} -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{C}| - \frac{1}{2}\text{tr}\left(\overline{\mathbf{y}}_{:,m,d}^{\top}\mathbf{C}^{-1}\overline{\mathbf{y}}_{:,m,d}\right), \end{aligned} \tag{7.7}$$

where $\mathbf{C} \coloneqq w_{m,d}^2\mathbf{K}_d + \sigma_{m,d}^2\mathbf{I}$. To find the parameters by maximizing Eq. (7.7), I take the partial derivatives of F^{LOB} with respect to the parameters and obtain

$$\frac{\partial F^{\text{LOB}}}{\partial w_{m,d}} = w_{m,d}\text{tr}\left(\mathbf{B}\mathbf{C}^{-1}\mathbf{K}_d\right), \tag{7.8}$$

$$\frac{\partial F^{\text{LOB}}}{\partial \sigma_{m,d}} = \sigma_{m,d}\text{tr}\left(\mathbf{B}\mathbf{C}^{-1}\right), \tag{7.9}$$

$$\frac{\partial F^{\text{LOB}}}{\partial \kappa_{i,d}} = \sum_{m=1}^{M}\frac{1}{2}w_{m,d}^2\text{tr}\left(\mathbf{B}\mathbf{C}^{-1}\frac{\partial \mathbf{K}_d}{\partial \kappa_{i,d}}\right), \tag{7.10}$$

where $\mathbf{B} \coloneqq \mathbf{C}^{-1}\overline{\mathbf{y}}_{:,m,d}\overline{\mathbf{y}}_{:,m,d}^{\top} - \mathbf{I}$ and $\frac{\partial \mathbf{K}_d}{\partial \kappa_{i,d}}$ is a matrix of element-wise partial derivatives of Eq. (7.5) with respect to $\kappa_{1,d}, \ldots, \kappa_{5,d}$. As there exists no closed-form solution, I resort to L-BFGS quasi-Newton method to maximize F^{LOB}. Essentially, in each iteration the gradients are computed by Eqs. (7.8) to (7.10) and the parameters are updated accordingly.

Estimate of Ground Truth

Note that the ground truth $\mathbf{Z}$ is marginalized out from Eq. (7.7) and still remains unknown. To estimate the ground truth of all training instances, I need to find the posterior of $\mathbf{Z}$, i.e. $p(\mathbf{Z} \mid \mathbf{Y}, \mathbf{X}) = p(\mathbf{Y} \mid \mathbf{Z}, \mathbf{X})p(\mathbf{Z} \mid \mathbf{X})/p(\mathbf{Y} \mid \mathbf{X})$. By using the property of Gaussian distribution, one can show that the posterior of $\mathbf{z}_{:,d}$ follows $\mathcal{N}(\mathbf{u}, \mathbf{V})$, which is unfortunately intractable. Therefore, I aim to seek an approximate distribution close to the true posterior in the KL divergence sense. Take the d^{th} dimension of ground truth as example, the problem is equivalent to minimizing the KL divergence between the true posterior and an arbitrary function $q(\mathbf{z}_{:,d})$, namely

$$\mathrm{KL}[q(\mathbf{z}_{:,d}) \parallel p(\mathbf{z}_{:,d} \mid \mathbf{y}_{:,:,d}, \ldots, \mathbf{y}_{:,M,d}, \mathbf{X})] = -\mathcal{Q} + \log \prod_{m=1}^{M} p(\mathbf{y}_{:,m,d} \mid \mathbf{X}),$$

where $\mathcal{Q} \coloneqq \int \left(q(\mathbf{z}_{:,d}) \log \frac{p(\mathbf{y}_{:,1,d}, \ldots, \mathbf{y}_{:,M,d} \mid \mathbf{z}_{:,d}, \mathbf{X}) p(\mathbf{z}_{:,d} \mid \mathbf{X})}{q(\mathbf{z}_{:,d})} \right) \mathrm{d}\mathbf{z}_{:,d}$. One can see immediately that KL divergence will be minimized when $\mathcal{Q}$ is maximized. Substituting Eq. (7.6) and Eq. (7.4) into $\mathcal{Q}$ gives

$$\begin{aligned}
\mathcal{Q} &= \int q(\mathbf{z}_{:,d}) \left(\sum_{m=1}^{M} \log p(\mathbf{y}_{:,m,d} \mid \mathbf{z}_{:,d}) + \log p(\mathbf{z}_{:,d} \mid \mathbf{X}) - \log q(\mathbf{z}_{:,d}) \right) \mathrm{d}\mathbf{z}_{:,d} \\
&= \int q(\mathbf{z}_{:,d}) \Bigg(\sum_{m=1}^{M} \frac{1}{2\sigma_{m,d}^2} \left(2(1 + w_{m,d}) \overline{\mathbf{y}}_{:,m,d}^{\top} \mathbf{z}_{:,d} - (1 + w_{m,d})^2 \mathbf{z}_{:,d}^{\top} \mathbf{z}_{:,d} \right) \\
&\quad - \frac{1}{2} \mathbf{z}_{:,d}^{\top} \mathbf{K}_d^{-1} \mathbf{z}_{:,d} - \log q(\mathbf{z}_{:,d}) \Bigg) \mathrm{d}\mathbf{z}_{:,d} + \text{constant}. \qquad (7.11)
\end{aligned}$$

I now parameterize $q(\mathbf{z}_{:,d}) \coloneqq \mathcal{N}(\mathbf{z}_{:,d} \mid \mathbf{u}, \mathbf{A})$ as a Gaussian density function, which gives

$$\begin{aligned}
\int \mathbf{z}_{:,d} q(\mathbf{z}_{:,d}) \mathrm{d}\mathbf{z}_{:,d} &= \mathbb{E}[\mathbf{z}_{:,d}] = \mathbf{u} \\
\int q(\mathbf{z}_{:,d}) \log q(\mathbf{z}_{:,d}) \mathrm{d}\mathbf{z}_{:,d} &= -\frac{1}{2} \log \left((2\pi e)^N |\mathbf{A}| \right).
\end{aligned}$$

Substituting these back into (7.11) gives

$$\begin{aligned}
\mathcal{Q} = &\sum_{m=1}^{M} \frac{1}{\sigma_{m,d}^2} \left(2(1 + w_{m,d}) \overline{\mathbf{y}}_{:,m,d}^{\top} \mathbf{u} - (1 + w_{m,d})^2 (\mathrm{tr}(\mathbf{A}) + \mathbf{u}^{\top} \mathbf{u}) \right) \\
&- \mathrm{tr}(\mathbf{K}_d^{-1} \mathbf{A}) - \mathbf{u}^{\top} \mathbf{K}_d^{-1} \mathbf{u} + \log |\mathbf{A}| + \text{constant}. \qquad (7.12)
\end{aligned}$$

Finally by taking the partial derivative of (7.12) with respect to $\mathbf{u}$ and $\mathbf{A}$ and set them to zero respectively, which ends up with

$$\mathbf{u} = \mathbf{V} \left(\sum_{m=1}^{M} \frac{w_{m,d}}{\sigma_{m,d}^2} \overline{\mathbf{y}}_{:,m,d} \right), \quad \mathbf{V} = \left(\sum_{m=1}^{M} \frac{w_{m,d}^2}{\sigma_{m,d}^2} \mathbf{I} + \mathbf{K}_d^{-1} \right)^{-1}. \qquad (7.13)$$

The above computation is repeated D times on every dimension to obtain the estimate of ground truth $\widetilde{\mathbf{Z}}$.

Prediction on New Instance

Given a new instance $\mathbf{x}_*$, the goal is predicting the ground truth $\mathbf{z}_*$ by using the learned regression function. This can be derived from the joint distribution

$$\begin{bmatrix} \widetilde{\mathbf{z}}_{:,d} \\ z_{*,d} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_d & \mathbf{k}_*^\top \\ \mathbf{k}_* & k_d(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right), \tag{7.14}$$

where $\mathbf{k}_* \coloneqq [k_d(\mathbf{x}_*, \mathbf{x}_1), \ldots, k_d(\mathbf{x}_*, \mathbf{x}_N)]$. It turns out that $p(z_{*,f} \mid \mathbf{X}, \widetilde{\mathbf{z}}_{:,d}, \mathbf{x}_*)$ follows a Gaussian distribution. Hence, the best estimate for the ground truth is

$$\widetilde{z}_{*,d} = \mathbf{k}_* \mathbf{K}_d^{-1} \widetilde{\mathbf{z}}_{:,d}, \tag{7.15}$$

and the uncertainty is captured in its variance

$$\mathrm{var}(\widetilde{z}_{*,d}) = k_d(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_* \mathbf{K}_d^{-1} \mathbf{k}_*^\top. \tag{7.16}$$

As a consequence, the response from an observer can be also predicted by

$$\widetilde{y}_{*,m,d} = (1 + \widetilde{w}_{m,d}) \widetilde{z}_{*,d} + \widetilde{\mu}_{m,d}, \tag{7.17}$$

with variance $\widetilde{\sigma}_{m,d}$.

Priors on Parameters

Note that $w_{m,d}$ is an important indicator of the observer's expertise. On the one hand, a genuine observer would have $w_{m,d}$ close to 1, whereas an adversarial observer gives $w_{m,d}$ close to -1. On the other hand, we encourage $w_{m,d}$ to be a small value unless supported by the data. Without any knowledge on observers, one can only expect that $w_{m,d}$ takes value either around 1 or -1, which inspires the following penalty function

$$\text{penalty}(w_{m,d}) \coloneqq \begin{cases} \eta(w_{m,d} - 1)^2 & \text{if } w_{m,d} > 1; \\ 0 & \text{if } -1 \leq w_{m,d} \leq 1; \\ \eta(w_{m,d} + 1)^2 & \text{if } w_{m,d} < -1, \end{cases} \tag{7.18}$$

where η controls the value of penalty as shown in Fig. 7.3 (see "general"). When $w_{m,d}$ takes value between $[-1, 1]$, there is no penalty and the gradient is given by Eq. (7.8) directly. When $|w_{m,d}| > 1$ we penalize $w_{m,d}$ and keep it from being too large. This allows the model to search a reasonable solution for $w_{m,d}$ without over-fitting on the training data.

In the case that observers are highly reliable, the learned $w_{m,d}$ should be close to 1 and $\mu_{m,d}, \sigma_{m,d}$ close to 0. One can add a Laplacian prior for observers' parameters, which leads to an L_1 regularization. The penalty term induced by the Laplacian prior for $w_{m,d}$ is $-(\frac{1}{2}\log\lambda + \sqrt{\frac{2}{\lambda}}|w_{m,d} - 1|)$, where a smaller value of λ suggests that the observer is more reliable. The maximization of F^{LOB} can be carried out by computing the sub-gradient of $w_{m,d}, \mu_{m,d}$ and $\sigma_{m,d}$, respectively.

The relationship between observers can be incorporated into the model as well. For example, the demographic information of users or the geographic location of sensors can be represented as a $M \times M$ proximity matrix $\mathbf{P}$. In particular, one can expect two observers have similar pa-

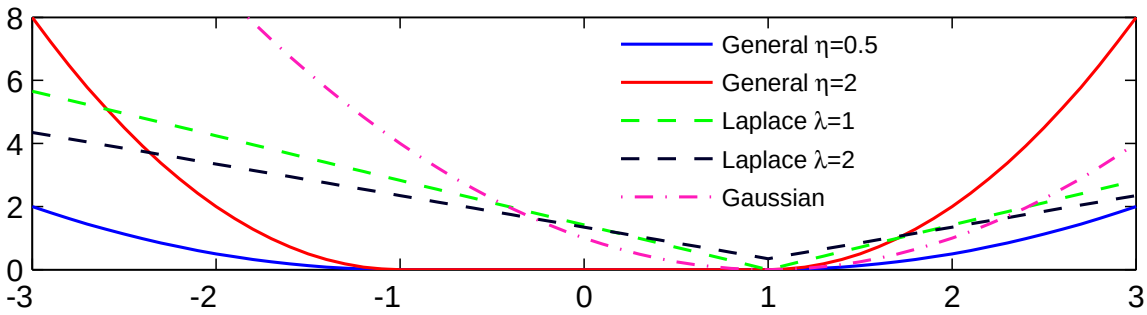


Figure 7.3.: Penalty functions of $w_{m,d}$ induced by different prior models. The "general" penalty function corresponds to Eq. (7.18). Similar penalty functions can be added to $\mu_{m,d}$ and $\sigma_{m,d}$ as well.

rameters if they are highly correlated in $\mathbf{P}$. Assuming $\mathbf{P}$ is a positive definite matrix, we can set the prior distribution of $\mathbf{w}_{:,d}$ set as $\mathcal{N}(\mathbf{w}_{:,d} \,|\, \mathbf{1}, \mathbf{P})$. As a consequence, I add a penalty term $-\sum_{d=1}^{D} \operatorname{tr}(\mathbf{w}_{:,d}^{\top} \mathbf{P} \mathbf{w}_{:,d})$ to Eq. (7.6). The gradient of $w_{m,d}$ is computed by Eq. (7.8) with an additional term $-2\mathbf{P}_{m,:}\mathbf{w}_{:,d}$. Figure 7.3 illustrates different penalty functions of $w_{m,d}$.

Table 7.1.: Penalty terms added to Eq. (7.6) under different prior models, where $K_\alpha(x)$ is the modified Bessel function of the second kind with order α and evaluated at x.

Prior	Parameters	Penalty Term
Gaussian	$\mathbf{P}$	$\sum_d \operatorname{tr}(\mathbf{w}_{:,d}^{\top} \mathbf{P} \mathbf{w}_{:,d})$
Laplace	λ	$-\sum_m \sum_d (\frac{1}{2} \log \lambda + \sqrt{\frac{2}{\lambda}} \lvert w_{m,d} \rvert)$
Inverse Gaussian	λ, η	$-\frac{1}{2} \sum_m \sum_d \log(w_{m,d}^2 + \lambda) + \log \left(K_1 \left(\frac{\sqrt{\lambda}\sqrt{w_{m,d}^2 + \lambda}}{\eta} \right) \right)$

Missing Responses

The model can be extended to handle the training data with missing responses. First of all, I partition the responses $\mathbf{Y} = (\mathbf{Y}^o, \mathbf{Y}^u)$, where $\mathbf{Y}^o$ represents the observed part and $\mathbf{Y}^u$ is the missing part of the responses. Consequently, the latent variables in the model consists of $\mathbf{Z}$ and $\mathbf{Y}^u$. The *expectation maximization* (EM) algorithm can be developed for estimating the model parameters. In the E-step, I fix the model parameter $\boldsymbol{\Theta}$ and compute the sufficient statistics of $\widetilde{\mathbf{Z}}$ by Eq. (7.13) and then update $\widetilde{\mathbf{Y}}^u$ by its prediction using Eq. (7.17). In the M-step, I use L-BFGS to maximize $\log p(\widetilde{\mathbf{Y}}, \widetilde{\mathbf{Z}} \,|\, \mathbf{X}, \boldsymbol{\Theta})$ and update $\boldsymbol{\Theta}$. The two steps are repeated until the likelihood reaches a local maximum.

7.2.4. Non-Linear Observer Model

The assumptions behind the linear observer model may not be appropriate in some scenarios. For instance, if the thermistor is being used to measure the temperature of the environment, due to the self-heating effect the electrical heating may introduce a significant error, which is known as a nonlinear function of the actual environment temperature. Moreover, the observers' responses

may depend on the input instance. With these considerations in mind, I propose a more sophisticated model which assumes that $\{g_{m,d}\}$ is a nonlinear mapping from $\mathcal{X} \times \mathcal{Z}$ to $\mathcal{Y}$. By representing $\{g_{m,d}\}$ as the Gaussian process, the second conditional distribution in (7.1) has the form of

$$p(\mathbf{Y} \,|\, \mathbf{Z}, \mathbf{X}) = \prod_{m=1}^{M} \prod_{d=1}^{D} \mathcal{N}\left(\mathbf{y}_{:,m,d} \,|\, \mathbf{0}, \mathbf{S}_{m,d}\right), \tag{7.19}$$

where $\mathbf{Y}$ is connected with $\mathbf{X}$ and $\mathbf{Z}$ by a $N \times N$ kernel matrix $\mathbf{S}_{m,d}$. The $(i,j)^{\text{th}}$ element in $\mathbf{S}_{m,d}$ is given by

$$\begin{aligned} s_{m,d}\left(\{\mathbf{z}_i, \mathbf{x}_i\}, \{\mathbf{z}_j, \mathbf{x}_j\}\right) :=& \phi_{m,1,d}^2 \exp\left[-\frac{\phi_{m,2,d}^2}{2}(z_{i,d} - z_{j,d})^2\right] \\ &+ \phi_{m,3,d}^2 + \phi_{m,4,d}^2 z_{i,d} z_{j,d} + \phi_{m,5,d}^2 \delta(z_{i,d}, z_{j,d}) \\ &+ \phi_{m,6,d}^2 \exp\left[-\frac{1}{2}\sum_{l=1}^{L} \eta_{m,l,d}^2 (x_{i,l} - x_{j,l})^2\right], \end{aligned}$$

where $x_{i,l}$ is the l^{th} dimension of the instance $\mathbf{x}_i$. This covariance function has a similar form as Eq. (7.5), but with the addition of an *automatic relevance determination* kernel on $\mathbf{X}$. By incorporating a separate parameter $\eta_{m,l,d}$ for each input dimension l, one can optimize these parameters to infer the relative importance of different dimensions of an instance from the data. One can see that, as $\eta_{m,l,d}$ becomes small, the response $y_{n,m,d}$ becomes relatively insensitive to $x_{n,l}$. This allows one to detect the dimensions of $\mathcal{X}$ that substantially affect the observer's response.

Parameter Estimation

The observer model in Eq. (7.19) can be combined with Eq. (7.4) to form the new model,

$$p(\mathbf{Y} \,|\, \mathbf{X}, \boldsymbol{\Theta}) = \int p(\mathbf{Y} \,|\, \mathbf{Z}, \mathbf{X}, \boldsymbol{\Theta}) p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\Theta}) \mathrm{d}\mathbf{Z},$$

where $\boldsymbol{\Theta} \coloneqq \{\{\kappa_{1,d}, \ldots, \kappa_{5,d}\}, \{\phi_{m,1,d}, \ldots, \phi_{m,6,d}\}, \{\eta_{m,l,d}\}\}$ is the set of model parameters to be inferred from the data. Unfortunately, such marginalization of $\mathbf{Z}$ intractable as the latent variable $\mathbf{z}$ appears nonlinear in the kernel matrix. Instead, I seek a *maximum a posterior* (MAP) solution by maximizing

$$\log p(\mathbf{Z}, \boldsymbol{\Theta} \,|\, \mathbf{Y}, \mathbf{X}) = \log p(\mathbf{Y} \,|\, \mathbf{Z}, \mathbf{X}, \boldsymbol{\Theta}) + \log p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\Theta}) + \text{constant}, \tag{7.20}$$

with respect to $\mathbf{Z}$ and $\boldsymbol{\Theta}$. Substituting Eq. (7.19) and Eq. (7.4) into Eq. (7.20) gives

$$\begin{aligned} F^{\text{NLOB}} \coloneqq \log p(\mathbf{Z}, \boldsymbol{\Theta} \,|\, \mathbf{Y}, \mathbf{X}) = -\frac{1}{2}\Bigg(&\sum_{d=1}^{D} \sum_{m=1}^{M} \left(\ln |\mathbf{S}_{m,d}| + \mathrm{tr}(\mathbf{S}_{m,d}^{-1} \mathbf{y}_{:,m,d} \mathbf{y}_{:,m,d}^{\top})\right) \\ &+ \sum_{d=1}^{D} \left(\ln |\mathbf{K}_d| + \mathrm{tr}(\mathbf{K}_d^{-1} \mathbf{z}_{:,d} \mathbf{z}_{:,d}^{\top})\right) \Bigg) + \text{constant}. \end{aligned}$$

The partial derivative of F^{NLOB} with respect to the latent variable is given by

$$\frac{\partial F^{\text{NLOB}}}{\partial \mathbf{z}_{:,d}} = \operatorname{tr}\left(\left(\mathbf{S}_{m,d}^{-1}\mathbf{y}_{:,m,d}^{\top}\mathbf{y}_{:,m,d}\mathbf{S}_{m,d}^{-1} - \mathbf{S}_{m,d}^{-1}\right)\frac{\partial \mathbf{S}_{m,d}}{\partial \mathbf{z}_{:,d}}\right) - \mathbf{K}_d^{-1}\mathbf{z}_{:,d}. \tag{7.21}$$

The gradients with respect to the parameters of kernel matrix can be likewise derived as in the linear observer model. Finally, these gradients are used in the L-BFGS algorithm for maximizing F^{NLOB}.

When the algorithm converges, the estimate of ground truth is directly given by the stationary point of F^{NLOB}. Predicting the response of a new instance can be carried out in the same way as in Eq. (7.13). Moreover, the estimation of the m^{th} observer's response is given by

$$\widetilde{y}_{*,m,d} = \mathbf{s}_* \mathbf{S}_{m,d}^{-1} \widetilde{\mathbf{y}}_{:,m,d},$$

where $\mathbf{s}_* \coloneqq [s_{m,d}(\widetilde{\mathbf{z}}_*, \widetilde{\mathbf{z}}_1, \mathbf{x}_*, \mathbf{x}_1), \ldots, s_{m,d}(\widetilde{\mathbf{z}}_*, \widetilde{\mathbf{z}}_N, \mathbf{x}_*, \mathbf{x}_N)]$.

Initialization

Note that seeking the MAP solution of $\mathbf{Z}$ and $\boldsymbol{\Theta}$ simultaneously may lead to a bad local optimum. Specifically, the model may stuck in a solution where $\{f_d\}$ is too trivial (e.g. close to a constant) and $\{g_{m,d}\}$ is too complicated (e.g. highly non-linear), which contradicts the intuition. To mitigate this problem, I first fit the training data with the linear observer model. The idea is to find an initial approximation of $\{f_d\}$ by restricting $\{g_{m,d}\}$ as linear. Then, I take $\widetilde{\mathbf{Z}}$ estimated by the linear observer model as the initialization of the ground truth, and train the nonlinear observer model to further refine $\{f_d\}$ and $\{g_{m,d}\}$.

7.3. Experiments

To evaluate the performance of the proposed algorithm on predicting the ground truth and the observers' responses, I set up two experiments. First, the effectiveness of the proposed models was demonstrated on the synthetic data. The second experiment was conducted on the real-world data. In both experiments, the ground truth was known and observers' responses were simulated by mapping the ground truth with some random nonlinear functions. As a consequence, the performance could be evaluated straightforwardly. Two metrics were considered here, i.e. the mean absolute normalized error (MANE) and the Pearson correlation coefficient (PCC). In MANE, I first re-scaled the actual value and its predicted value into $[0, 1]$ respectively, and then measured the mean absolute error. MANE value close to 0 or PCC value close to 1 indicates that the algorithm performs well. In particular, the expected MANE of a random predictor is 0.5.

The proposed linear observer model (LOB) and nonlinear observer model (NLOB) were compared with several baselines. Refer SVR and GPR as the Support Vector Regression and Gaussian Process Regression trained with the ground truth, respectively. I combined responses from multiple observers by taking the average and then used it as training labels. These two models were denoted as SVR-AVG and GPR-AVG, respectively. For a fair comparison, the covariance function of $\mathbf{x}$ of GPR and GPR-AVG was in the same composite form as in Eq. (7.5). In addition to these non-parametric methods, `Raykar` refers to the model in which both $p(\mathbf{Z} \mid \mathbf{X})$ and $p(\mathbf{Y} \mid \mathbf{Z})$ are Gaussian in the spirit of [131].

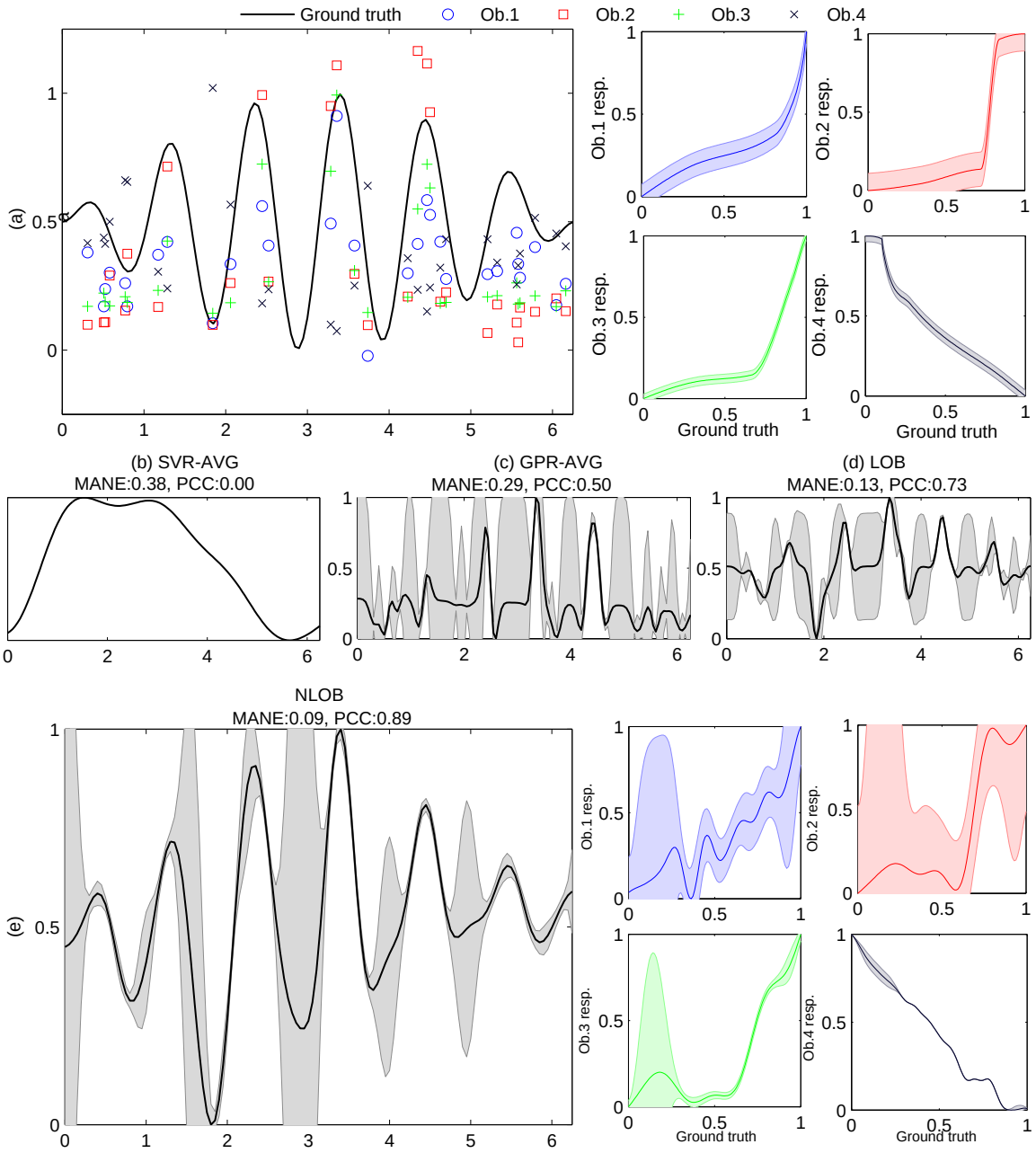


Figure 7.4.: **(a)** Synthetic data generated for the experiment. Responses from observers are represented by markers with different colors. The right panel illustrates randomly generated $\{g_m\}$ used for simulating four observers. Shaded area represents the pointwise variance. Note that the 4^{th} observer is *adversarial*, as his response tends to be the *opposite* of the ground truth. **(b, c, d)** Predicted ground truth on the test set by applying SVR–AVG, GPR–AVG and LOB, respectively. **(e)** Predicted ground truth and learned observer functions given by NLOB.

7.3.1. Synthetic Examples

To create one-dimensional synthetic data (i.e. $L \coloneqq 1$ and $D \coloneqq 1$), I set $f(x) \coloneqq \sin(6x)\sin(\frac{x}{2})$. The training instances $\mathbf{X}$ were generated by randomly sampling 30 points in $[0, 2\pi]$ from the uniform distribution. The test instances were obtained using a discretization of $[0, 2\pi]$ with equal space of 0.05, which results in 126 points. Four simulated observers were obtained by setting the corresponding $\{g_m\}$ as a random nonlinear monotonic function. For a training instance x, the m^{th} observer provides its response by $g_m(f(x))$ plus some Gaussian noise. An illustration of the synthetic data is depicted in Section 7.3. Figure 7.4(b, c, d, e) shows the results given by the baselines and the proposed method. Not surprisingly, taking the average of observers' responses is not an effective solution. In contrast, the proposed LOB and NLOB models outperform baseline methods significantly, which yield lower MANE and higher PCC. Moreover, the observers' functions learned by NLOB are very close to those predefined $\{g_m\}$ in Section 7.3.

7.3.2. On Real-World Data

I downloaded four real-world data sets from UCI Machine Learning Repository, i.e. AUTO, COMMUNITY, CONCRETE and WINE. On each data set, I randomly selected 500 instances and generated 20 observers in the same manner as in Section 7.3.1. The number of adversarial observers was fixed to 6. The experiment was conducted with 10-fold cross-validation. The prediction result of the ground truth and observers' responses was summarized in Table 7.2. It is notable that the proposed LOB and NLOB significantly outperform SVR/GPR-AVG and `Raykar` on inferring the ground truth. In general, additional improvements are observed when NLOB is used. Comparing it with the SVR/GPR column, one can see that the regression function learned by NLOB is almost as good as the one trained using the ground truth. I remark that the promising performance of NLOB is achieved by merely learning from a set of observers without any prior knowledge of their expertise and the ground truth. Furthermore, LOB and NLOB also show encouraging performance on predicting responses of observers, which can be proved useful in many applications such as the recommendation system.

Table 7.2.: Prediction of the ground truth and observers' responses. In each cell, the upper value is MANE, while PCC is at the bottom. For the ground truth and the average baselines we only report the best performance, where a superscript $^{\text{S}}$ denotes that the performance is achieved by SVR or SVR-AVG; for GPR and GPR-AVG we use the superscript $^{\text{G}}$. The best model on each data set is highlighted by bold font. Note that only LOB and NLOB can predict observers' responses.

Data set	Ground truth				
	SVR/GPR	SVR/GPR-AVG	`Raykar`	LOB	NLOB
AUTO	$0.19 \pm 0.05^{\text{G}}$	$0.21 \pm 0.07^{\text{G}}$	0.25 ± 0.08	0.26 ± 0.05	$\mathbf{0.20 \pm 0.04}$
	$0.84 \pm 0.07^{\text{G}}$	$0.63 \pm 0.43^{\text{G}}$	0.50 ± 0.22	$\mathbf{0.84 \pm 0.05}$	0.82 ± 0.08
COMMUNITY	$0.15 \pm 0.03^{\text{G}}$	$0.27 \pm 0.08^{\text{S}}$	0.22 ± 0.10	0.17 ± 0.03	$\mathbf{0.16 \pm 0.03}$
	$0.80 \pm 0.08^{\text{G}}$	$0.44 \pm 0.38^{\text{S}}$	0.70 ± 0.13	0.76 ± 0.04	$\mathbf{0.77 \pm 0.04}$
CONCRETE	$0.15 \pm 0.02^{\text{G}}$	$0.22 \pm 0.08^{\text{G}}$	0.20 ± 0.08	0.18 ± 0.07	$\mathbf{0.17 \pm 0.06}$
	$0.76 \pm 0.08^{\text{G}}$	$0.60 \pm 0.46^{\text{G}}$	0.66 ± 0.21	0.78 ± 0.11	$\mathbf{0.79 \pm 0.09}$
WINE	$0.20 \pm 0.06^{\text{G}}$	$0.30 \pm 0.05^{\text{S}}$	0.29 ± 0.06	0.27 ± 0.09	$\mathbf{0.25 \pm 0.07}$
	$0.67 \pm 0.12^{\text{G}}$	$0.52 \pm 0.30^{\text{G}}$	0.38 ± 0.19	0.58 ± 0.20	$\mathbf{0.61 \pm 0.17}$

Table 7.2. (cont.)

Data set	Observers' responses	
	LOB	NLOB
AUTO	0.26 ± 0.04	$\mathbf{0.25 \pm 0.09}$
	$\mathbf{0.75 \pm 0.05}$	0.70 ± 0.11
COMMUNITY	0.26 ± 0.04	$\mathbf{0.25 \pm 0.09}$
	$\mathbf{0.62 \pm 0.09}$	0.55 ± 0.15
CONCRETE	0.26 ± 0.04	$\mathbf{0.15 \pm 0.06}$
	0.66 ± 0.18	$\mathbf{0.72 \pm 0.15}$
WINE	0.32 ± 0.07	$\mathbf{0.24 \pm 0.07}$
	0.47 ± 0.18	$\mathbf{0.48 \pm 0.15}$

7.4. Conclusion

Motivated by the research problem raised at the end of Chapter 6, this chapter has investigated the regression problem under multiple observers providing responses that are not absolutely accurate. The problem involves learning a regression function and observers' expertise from such data without any prior information of the observers. Based on the Gaussian process, we propose a probabilistic framework and develop two models. The proposed approach provides an estimate of the ground truth and also predicts the responses of each observer given new instances. Experiments showed that the proposed method outperforms several baselines and leads to a performance close to the model trained with the ground truth.

There are many opportunities for further extending the model. One possible direction is to extend the model with *multiple kernel learning*. The idea is to let the algorithm pick or composite different covariance functions instead of fixing the combination in advance. As a consequence, the algorithm may learn complex fits for the observers by selecting multiple kernels in a data-dependent way. Moreover, it would be highly beneficial to design *active sampling* methods for selecting which instance and whose response should be learned next.

Chapter 8

Learning Unbiased Rating from Crowds

In this chapter, I apply the method described in Chapter 7 to solve a real-world problem. The task is to predict the aesthetics score of a given image. The *aesthetics* of a natural image is a measure of the perceived beauty of a visual stimulus.

In order to train such a model, a naive way is to acquire a set of images where each of them is labeled with an *objective* aesthetics score. However, the perception of beauty is subjective across individuals, thus such data set of score-image pairs is infeasible to obtain in practice. Fortunately, *subjective* aesthetics score can be easily collected with the help of photo sharing websites (e.g. Flickr, Photobucket, Photo.net), where each image is rated by a large number of users. Assuming that each image has a *universal* aesthetics value, which is an objective and consensus measure over all, or almost all, people. Having subjective scores from multiple users creates the possibility of learning the objective score.

The problem is that, scores provided by different users can vary widely due to their different educational backgrounds, personal tastes, psychological states, and so on. Furthermore, some users may even be adversarial by deliberately providing deceitful scores. These characteristics perfectly suit the method proposed in Chapter 7. I shall show that the proposed algorithm can quantitatively predict the universal aesthetics value in terms of score. The reader will see an interesting result on the real-world image data.

8.1. Related Work

Computational image aesthetics evaluation can be useful in various applications, such as content-based image retrieval and quality-based image management. In the recent past, there has been interest in this line of research [44, 91, 105, 113]. As the problem lends itself naturally to a regression setting once the training set is available, nearly all work concentrate on aesthetics features engineering. Another important yet often glossed over aspect is the training label of each image. Previous work used the average score of multiple users as a surrogate for the objective aesthetics score to train the regression model. However, this is fraught with danger as the expertise of each user is overlooked. Even though several aesthetics predictors trained in such manner have found appreciable success [106, 153], To the best of my knowledge, no previous work clearly considers the inherent subjectivity of this task.

8.2. Framework Illustration

I employ the same notations and the model formulation described in Chapter 7. Figure 8.1 illustrates the conditional dependence between $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ under the framwork.

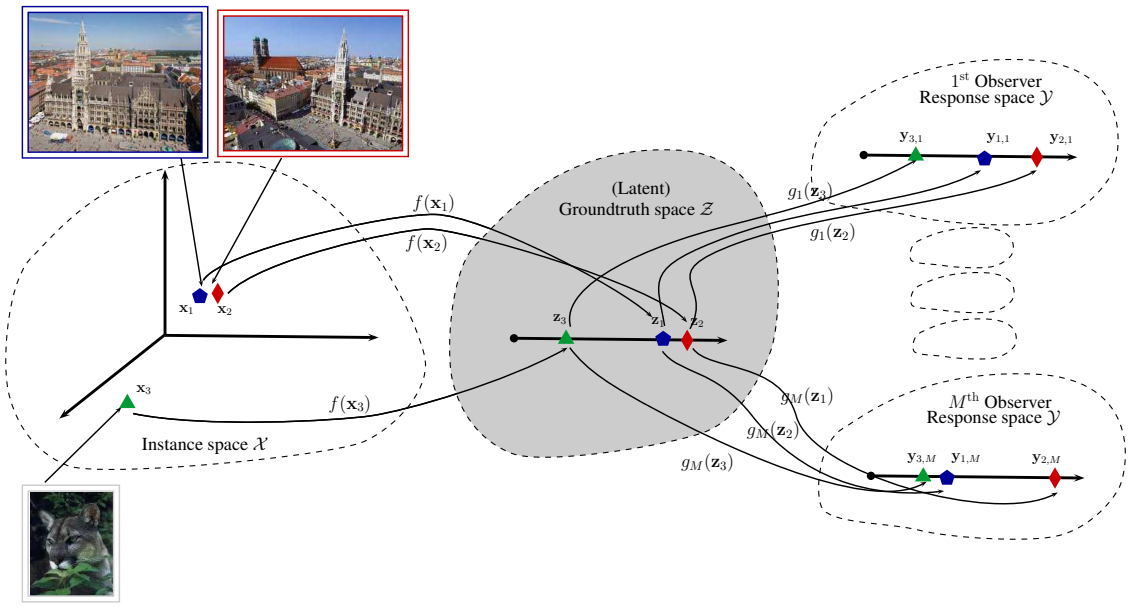


Figure 8.1.: Generative process of subjective aesthetics scores. Notations are followed from Chapter 7. Two photos of the city of Munich map to the similar place into the instance space, whereas the photo of cat is mapped to a place far away from the first two. Intuitively, if two instances are close to each other in $\mathcal{X}$, then their corresponding ground truth should be close in $\mathcal{Z}$ through the mapping of $\{f_d\}$, which in turn restricts the searching space of $\{g_{m,d}\}$ when $\mathbf{Y}$ is known.

8.3. Experiments

The experiment was conducted on the Photo.net data set [45]. As the original data set did not contain ratings from individual users, my colleague Huang implemented a Web interface that allows users to rate images by clicking on a continuous score bar (from 0 to 5). He collected ratings of 201 images from 10 different users, which results in a $201 \times 10(\times 1)$ response matrix $\mathbf{Y}$ without any missing value. Each image is represented as a 59-dimensional vector, which includes low-level perceptual information (e.g. exposure, contrast, sharpness, color saturation), some rules of thumb in photography as described in [44]. I trained GPR-AVG, `Raykar`, LOB, NLOB with 10-fold cross-validation, and then selected the best model to predict the objective aesthetics scores on the test set of 2733 unrated images.

Figure 8.2 illustrates five images with highest scores and five images with lowest scores predicted by different models. One can immediately see that there are some agreements on predicting high-score images among different models. Generally, panoramas of landscape win affection,

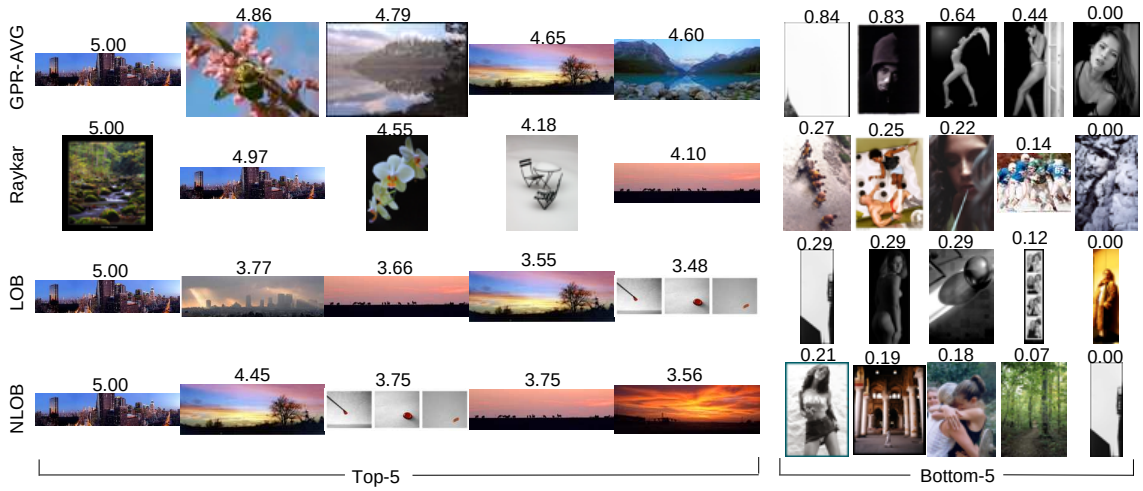


Figure 8.2.: Prediction on the test set with 2733 images. Each row shows top-5 (left) and bottom-5 (right) images for each model. The predicted objective aesthetics score is labeled above each image, respectively.

whereas portraits with drab and gray color are unpleasant and less attractive. Comparing to GPR–AVG and `Raykar`, the proposed LOB and NLOB lifted up panorama images, which makes their predicted high-score images more homogeneous.

To highlight the difference between four methods, I depict five scatter plots in Fig. 8.3. As both LOB and NLOB are based on the Gaussian process, it is not surprising that predictions of LOB and NLOB are more correlated with GPR–AVG than `Raykar`. Also, it is worth highlighting that scores predicted by LOB and NLOB are generally lower than those given by GPR–AVG. This can be concluded from the first and third plot of Fig. 8.3, where nearly all points lie in the upper triangle.

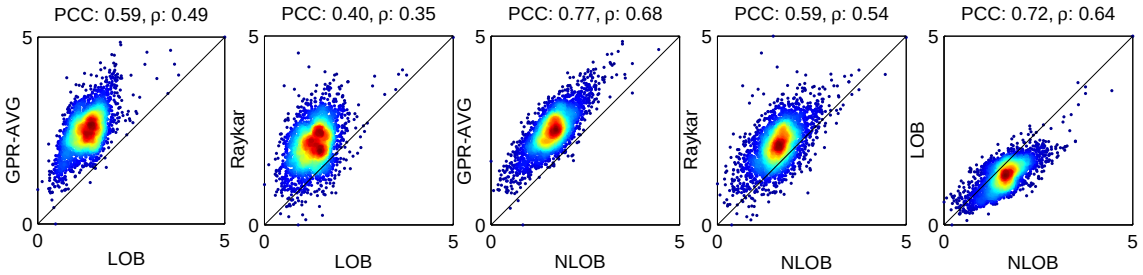


Figure 8.3.: Scatter plots of predicted objective aesthetics scores of 2733 test images, where the colors encode the density of the points. The title above represents the Pearson correlation (PCC) and Spearman correlation (ρ), respectively.

Finally, Fig. 8.4 shows scatter plots of predicted responses of 10 observers versus predicted objective aesthetics scores. Note that some users may exhibit varying levels of expertise on different images (e.g. the 3^{rd} and 10^{th} observer), whereas others produce rating that are highly correlated with the objective aesthetics scores (e.g. the 5^{th} and 9^{th} observer).

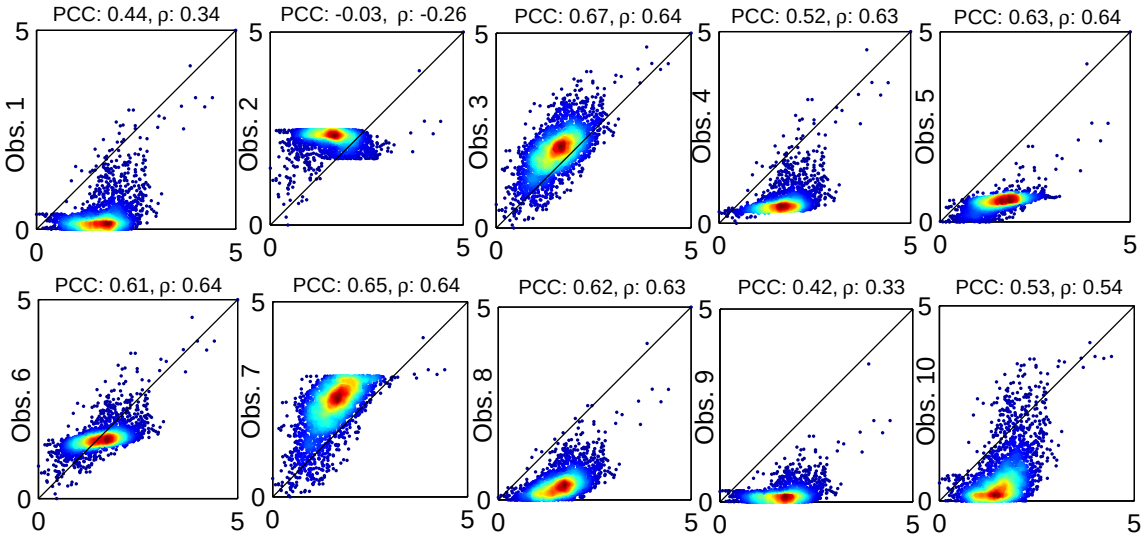


Figure 8.4.: Scatter plots of predicted observers' response of 2733 test images, where x-axis represents the predicted objective aesthetics scores.

Due to the lack of agreed upon standard evaluation metrics for this task, it is hard to say which model gives the best prediction. Nonetheless, I find the experimental results are interesting and consistent with the intuition.

8.4. Conclusion

Learning to evaluate the aesthetics score of an image can be beneficial in the content based image retrieval system. For instance, the learned aesthetics model can be incorporated into a content based image retrieval system to discriminate between visually similar images, giving higher priority to more pleasing query results. Unfortunately, the subjectivity of this task makes it extremely challenging for objective aesthetics assessment. Naively, one might take the average. But in fact this heuristic overlooks the difference between users hence it may fail in the settings with non-Gaussian or adversarial noise. In this chapter, I have demonstrated that the algorithm in Chapter 7 can be easily applied to solve this problem.

Even though social network services have been around for a long time, the interest in integrating unreliable opinions from anonymous users has just caught on until recently. This work reiterates the significance and usefulness of the the algorithm in Chapter 7 in learning unreliable information from multiple users. While very limited work has been published so far, this work is hoped to encourage more contributions in this field.

From the next chapter, the focus of the dissertation will be moved to improving the scalability of learning algorithms. The reader will see how current learning algorithms can be approximated to efficiently handle large-scale data.

Part V.

Scalable Online Learning Algorithms

Hofstadter's Law: It always takes longer than you expect, even when you take into account Hofstadter's Law

Douglas R. Hofstadter

Chapter 9

Online Prediction of User Behavior with Lazy Gaussian Process Committee

In this chapter, I present a novel Gaussian process approximation scheme for improving its scalability on large-scale real-time data. As it is introduced in Chapter 2, Gaussian process (GP) is a promising Bayesian method for non-linear regression and classification [129]. It has been demonstrated to be applicable to a wide variety of real-world statistical learning problems. An important advantage of GP over other non-Bayesian models is the explicit probabilistic formulation of the model, allowing one to assess the uncertainty of predictions as I showed in Chapter 7. In addition, since GP has a simple parameterization and the hyperparameters can be adjusted by maximizing the marginal likelihood, it is easy to implement and flexible to use.

However, GP is not always the method of choice especially for large date sets. GP has inherently dense representations in the sense that all training data is required for the prediction. The training procedure requires computation, storage and inversion of the full covariance matrix, which can be time-consuming. Furthermore, the Bayesian posterior update to incorporate data is also computationally cumbersome. These drawbacks prevent GP from applications with large amounts of training data that require fast computation, such as learning motor dynamics in real-time.

Much research in recent years has focused on reducing the computational requirements of GP on large data sets. Many of these methods are based on a small set of training inputs, which summarizes the bulk of information provided by all training data [144, 137, 146, 127]. Other methods make structural assumptions about the covariance matrix so that a GP can be decomposed into a number of smaller GPs [157, 121, 28].

In this chapter I present a novel approximation method called *lazy Gaussian process committee* (LGPC) for learning from a continuous data stream. As its name suggests, LGPC is based on a combination of multiple GPs, which is closely related to several previous work [157, 121, 28]. Unlike previous work, LGPC is updated in a "lazy" fashion in the sense that new training examples are directly allocated to a subset of GPs without adapting their hyperparameters. The problem of selecting a near-optimal subset is formulated as submodular optimization, allowing the training procedure to be carried out efficiently. Experiments showed that LGPC has comparable accuracy to the standard GP regression and outperforms several GP online alternatives. The simplicity and the efficiency of LGPC make it more appealing in real-time online applications.

As a direct application in information security, I applied LGPC to predict mouse-trajectory of a user in an Internet banking scenario. The intention was to predict user's hand movements in real-time, so as to offer support to security applications, such as recognizing identity theft or an abnormal funds transfer. Thus, the model's learning and prediction should be sufficiently fast. In the field of psychology and cognitive science, there has been abundant evidence that a motor dynamic of the hand can reveal the time course of mental process [1, 147]. Moreover, several studies have showed that computer mouse-trajectory can afford valuable information about the temporal dynamics of a variety of psychological process [149, 64, 63]. For instance, when moving their hand while making a decision, people may deviate more from a straight trajectory if there is a tempting alternative, making viable such measures as maximum deviation, curvature area, and switches in direction. Moreover, unintentional stress might manifest less smooth, more complex and fluctuating trajectories [42]. An online learning technique is necessary as it allows the adaption to changes in the trajectories. The effectiveness of the online mouse-trajectory learning confirms the applicability of LGPC.

The rest of the chapter is organized as follows. Section 9.1 briefly reviews the Gaussian process and previous work on sparse approximations. Section 9.2 introduces the proposed method. Experimental results are presented in Section 9.3. Section 9.4 concludes the chapter and points out some future directions.

9.1. Related Work

This section briefly reviews Gaussian process and previous attempts on reducing its computational complexity. The underlying problems that motivate this work are highlighted.

9.1.1. GP Regression

The problem of regression aims to find a function estimation from the given data, which is usually formulated as follows: given a training set $\mathcal{D} \coloneqq \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ of N pairs of input vectors $\mathbf{x}_n$ and noisy scalar outputs y_n, the goal is to learn a function f transforming an input into the output given by

$$y_n = f(\mathbf{x}_n) + \epsilon_n,$$

where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ and σ^2 is the variance of the noise. A *Gaussian process* is a collection of random variables, any finite number of which have consistent joint Gaussian distribution. *Gaussian process regression* (GPR) is a Bayesian approach which assumes a GP prior over functions. As a result the observed outputs behave according to

$$p(\mathbf{y} \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \mathcal{N}(\mathbf{0}, \mathbf{K}),$$

where $\mathbf{y} \coloneqq [y_1, \ldots, y_N]^\top$ is a vector of output values, and $\mathbf{K}$ is an $N \times N$ covariance matrix; the entries are given by a covariance function, i.e. $K_{ij} \coloneqq k(\mathbf{x}_i, \mathbf{x}_j)$. In this work, I consider a frequently used covariance function given by

$$k(\mathbf{x}_i, \mathbf{x}_j) \coloneqq \kappa^2 \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)\right) + \sigma^2 \delta_{ij}, \tag{9.1}$$

where κ denotes the signal variance and $\mathbf{W}$ are the widths of the Gaussian kernel. The last term represents an additive Gaussian noise, i.e. $\delta_{ij} \coloneqq 1$ if $i = j$, otherwise $\delta_{ij} \coloneqq 0$. Samples from this prior are plotted for various values of the parameters in Fig. 9.1.

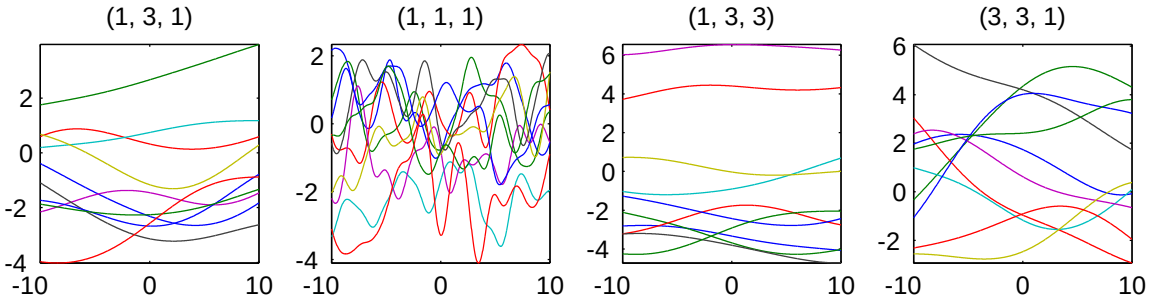


Figure 9.1.: Samples drawn from a Gaussian process prior defined by the covariance function (9.1). The samples are obtained using a discretization of the x-axis of 1000 equally spaced points. The text above each plot denotes the value of $\kappa^2, \sigma^2, \{\mathbf{W}\}$, respectively. In this example, the input $\mathbf{x}$ is one-dimensional. Hence, the parameter $\{\mathbf{W}\}$ is in fact a scalar value, which can be absorbed into κ.

In the setting of probabilistic regression, the goal is to find a predictive distribution of the output y_* at a test point $\mathbf{x}_*$. Under GPR, the predictive distribution of y_* conditional on the training set $\mathcal{D}$ is also Gaussian

$$p(y_* \mid \mathcal{D}, \mathbf{x}_*) = \mathcal{N}\left(\mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{y}, k_* - \mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{k}_*\right), \tag{9.2}$$

where $\mathbf{k}_* \coloneqq [k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_N)]^\top$ and $k_* \coloneqq k(\mathbf{x}_*, \mathbf{x}_*)$. One can observe that the training data is explicitly required at the test time in order to construct the predictive distribution, which makes GP a non-parametric method. The hyperparameters are $[\kappa^2, \sigma^2, \{\mathbf{W}\}]^\top$, where $\{\mathbf{W}\}$ denotes parameters in the width matrix $\mathbf{W}$. The optimal hyperparameters for a particular data set can be derived by maximizing the marginal likelihood function using a gradient based optimizer. For a more detailed background on GP, readers are referred to the textbook [129].

It should be noted that in each iteration the computation of the likelihood and the derivatives involves inversion of a matrix of size $N \times N$, which requires time $\mathcal{O}(N^3)$. Once the inversion is done, inference on a new test point requires $\mathcal{O}(N)$ for the predictive mean and $\mathcal{O}(N^2)$ for the predictive variance. Thus, a simple implementation of GPR can handle problems with at most a few thousands training examples, which prevents it from real-time applications dealing with large amounts of data.

9.1.2. GP Approximations

The sparse representation of data has been studied exhaustively [96]. It has been shown that the bulk of information provided by all training inputs can be summarized by a small set of inputs, which is often known as *inducing inputs* or support vectors. By assuming additional dependency about the training data given the inducing inputs, various sparse GP approximations were derived, such as the subset of regressors [144], projected latent variables [137] and sparse GP with pseudo-inputs [146]. A unifying view of these sparse GP methods was presented in [127].

It should be noted that the selection of inducing inputs does leave an imprint on the final solution. Loosely speaking, the selection can be carried out either in a passive (e.g. random) or active fashion. An extensive range of proposals were suggested to find a near-optimal choice for inducing inputs, such as posterior maximization [144], maximum information gain [137], matching pursuit [93] and pseudo-input via continuous optimization [146]. In particular, sparse online GP [40] was developed by combining the idea of a sparse representation with an online algorithm, allowing the inducing inputs (basis vectors) to be constructed in a sequential fashion.

An alternative approach for speeding up GPR is Bayesian committee machine (BCM) introduced by [157]. Loosely speaking, the original training data is partitioned into parts, where each part is maintained by a GP. The computational cost is thus significantly reduced due to much smaller number of training examples within each GP. BCM provides a principled approach to combining Bayesian estimators trained on different data sets for the prediction. Inspired by this idea, some other work have been focused on combining multiple GPs for regression [121, 28].

In the online setting, a straightforward application of the approaches mentioned above is impeded by two obstacles. First, it is inefficient to optimize hyperparameters every time a new training example is presented. Second, adding new training examples to the model may cause non-smooth fluctuations in the marginal likelihood function and its gradients, meaning that a smooth convergence is not guaranteed [137]. Although several heuristics can be adapted to alleviate this problem, there is no reliable way of learning hyperparameters. Moreover, as training examples are presented sequentially rather than in batch, selecting inducing inputs from a data stream in a far-sighted fashion becomes extremely challenging. Furthermore, the inducing inputs selection and the hyperparameters estimation are somewhat undermined by each other [146], which may adversely affect the quality and the efficiency of online regression.

9.2. LGPC for Online Regression

The basic idea of LGPC is straightforward. Instead of training a single GP using all the training data, I partition the data and allocate it to a committee consisted of Q independent GPs with different hyperparameters. That is, each GP maintains a subset of the training data, which is denoted as $\mathcal{D}_1, \ldots, \mathcal{D}_Q$ respectively, where $\mathcal{D}_q \coloneqq \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$ and T is the maximum number of training examples of each GP, which can be set in accordance with the available computational power. This idea is illustrated in Fig. 9.2

Intuitively, each GP in the committee corresponds to an interpretation of the relationship between input and output. For predicting the output y_* of a query point $\mathbf{x}_*$, the outputs from all GPs are combined together. Under the independence assumption, it obtains $p(\mathbf{y}_1, \ldots, \mathbf{y}_Q \,|\, \mathcal{D}) = \prod_{q=1}^{Q} p(\mathbf{y}_q \,|\, \mathcal{D}_q)$. Thus, the predictive distribution can be approximated as

$$\widehat{p}(y_* \,|\, \mathcal{D}, \mathbf{x}_*) = c \times \frac{\prod_{q=1}^{Q} p(y_* \,|\, \mathcal{D}_q, \mathbf{x}_*)}{[p(y_*)]^{Q-1}}, \tag{9.3}$$

where c is a normalization constant. The posterior predictive probability densities are simply multiplied. Note that since the posterior probability densities is multiplied, one has to divide by the priors $Q-1$ times. Readers may find out that LGPC has a similar predictive distribution as in BCM. However, as the reader will see in the next section, their training procedures are completely different.

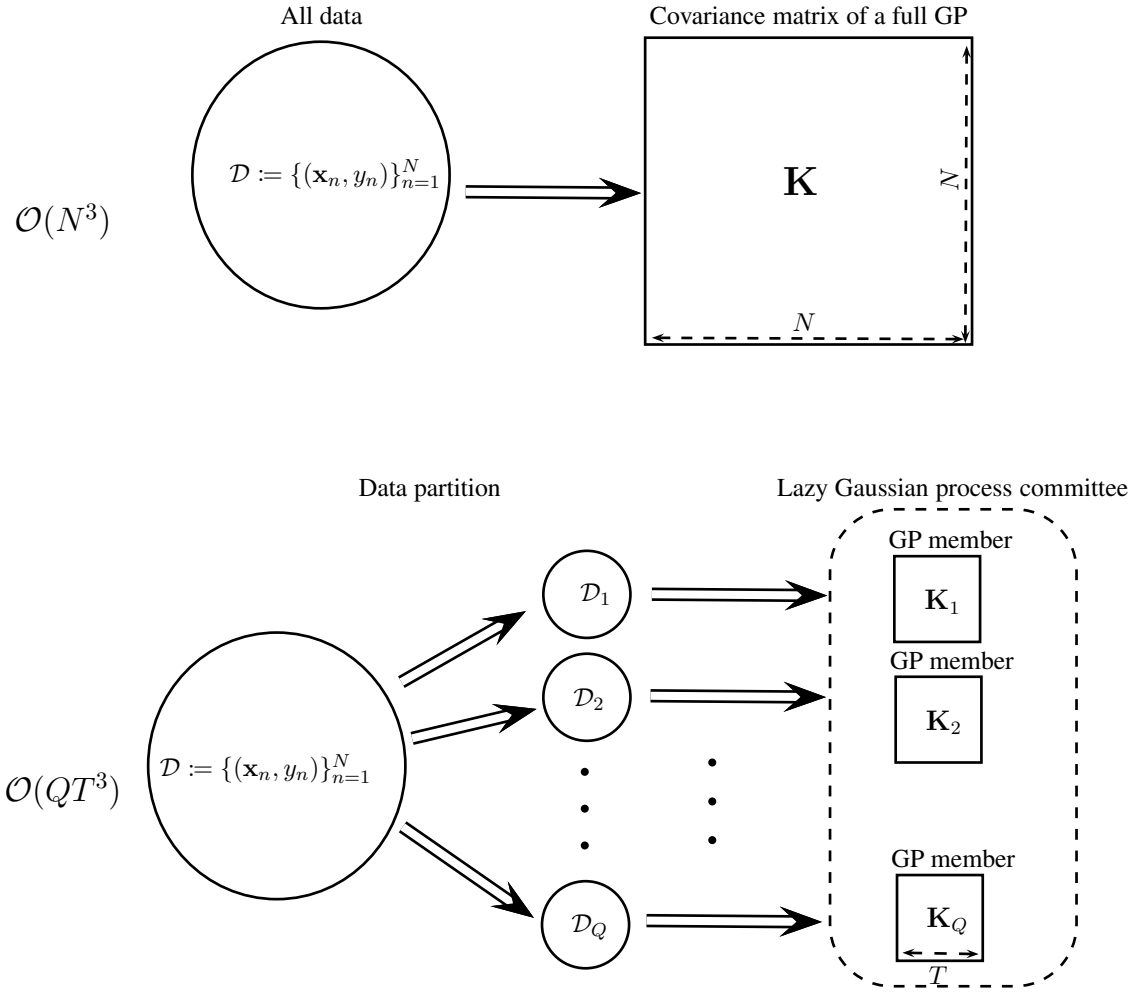


Figure 9.2.: The basic idea of LGPC: decomposing a large training data set into small sets. One each small data set, an individual GP is trained, and together they form a GP committee. **(Top)**: original GP regression model. **(Bottom)**: data partition in the proposed LGPC model.

This section describes LGPC in three parts, namely the allocation of new training examples, the incremental update and the predictions of query points. Note that all hyperparameters of LGPC are constants during online learning.

9.2.1. Allocation of New Training Examples

Denote all GPs in the committee as $\mathcal{Q} \coloneqq \{1, \ldots, Q\}$. Given a new training example $(\mathbf{x}_{N+1}, y_{N+1})$, I select a subset $\mathcal{A} \subseteq \mathcal{Q}$ and allocate the new example to their data collection, respectively. Denote

$\mathcal{D}_q^{\langle N \rangle}$ as the training examples allocated to the q^{th} GP at time N, the update rule is formalized as

$$\mathcal{D}_q^{\langle N+1 \,|\, \mathcal{A} \rangle} := \begin{cases} \mathcal{D}_q^{\langle N \rangle} \cup \{(\mathbf{x}_{N+1}, y_{N+1})\} & \text{if } q \in \mathcal{A}; \\ \mathcal{D}_q^{\langle N \rangle} & \text{otherwise.} \end{cases}$$

On the one hand, if $\mathcal{A}$ contains only one element, meaning that only one GP is updated each time, then the information provided by the new training example may not be well utilized. On the other hand, if one let $\mathcal{A} := \mathcal{Q}$, then all GPs must update their corresponding Gram matrix to include the new training example (no matter whether such inclusion will contribute to the prediction of the committee or not), which can degrade the efficiency and the quality of the prediction. Thus, it needs an *active* selection policy to choose at most S GPs from the committee, such that their data inclusion can yield the maximal improvement for prediction. Fig. 9.3 illustrates this idea.

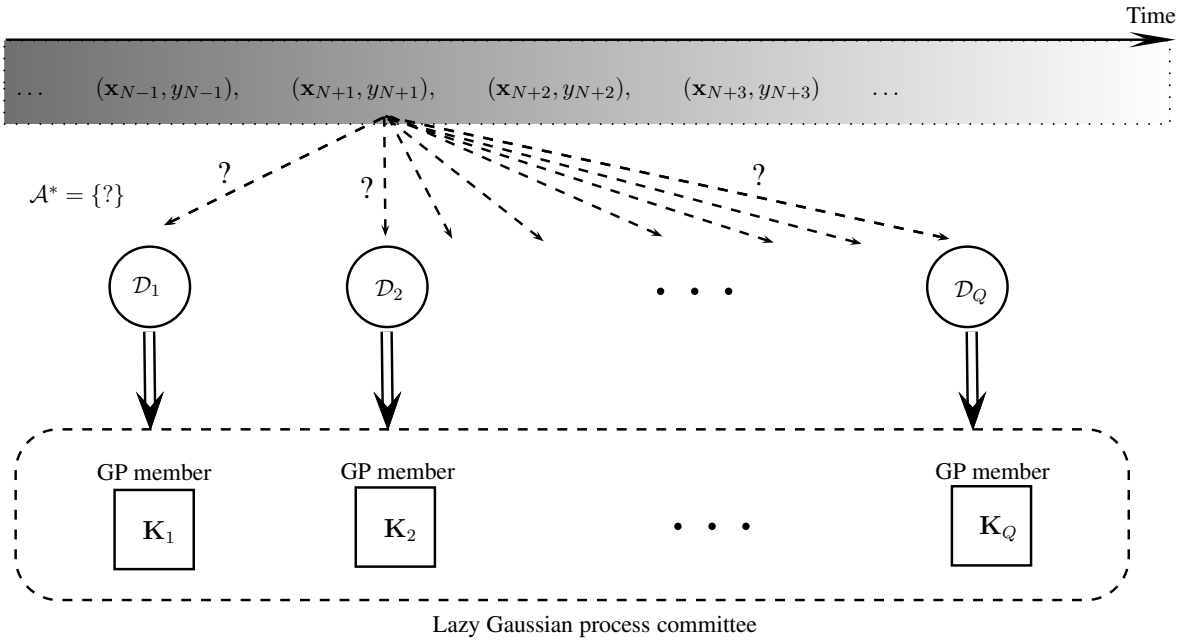


Figure 9.3.: Allocating new data point to the GP members. The selection problem tries to answer which GP should be selected in order to maximize the committee's performance in the long-run.

Clearly it can make sense to select which GPs are taken into $\mathcal{A}$ by optimizing some criterion. The idea here is to maximize the likelihood of LGPC on a small subset of training examples. To see this I introduce a *reference set* $\mathcal{R}$, in which both inputs $\mathbf{X}_\mathcal{R}$ and outputs $\mathbf{y}_\mathcal{R}$ are observed. The reference set can be constructed, for instance, by subsampling all previous training data $\{\mathcal{D}_q^{\langle N \rangle}\}_{q=1}^{Q}$ with the addition of the current training example $(\mathbf{x}_{N+1}, y_{N+1})$. Note that the terms in the numerator and the denomination of (9.3) are all Gaussian distributions over y_*. Thus, the predictive distribution for $\mathbf{y}_\mathcal{R}$ at time N can be approximated by a Gaussian distribution with mean and

covariance as follows

$$\mathbb{E}_{\widehat{p}}^{\langle N\rangle}(\mathbf{y}_{\mathcal{R}}) = \mathbb{C}_{\widehat{p}}^{\langle N\rangle}(\mathbf{y}_{\mathcal{R}}) \sum_{q\in\mathcal{Q}} \Big(\mathbb{C}(\mathbf{y}_{\mathcal{R}} \,|\, \mathcal{D}_q^{\langle N\rangle}, \mathbf{X}_{\mathcal{R}})^{-1} \mathbb{E}(\mathbf{y}_{\mathcal{R}} \,|\, \mathcal{D}_q^{\langle N\rangle}, \mathbf{X}_{\mathcal{R}})\Big), \tag{9.4}$$

$$\mathbb{C}_{\widehat{p}}^{\langle N\rangle}(\mathbf{y}_{\mathcal{R}}) = \Big(-(Q-1)\boldsymbol{\Sigma}_{\mathcal{R}\mathcal{R}}^{-1} + \sum_{q\in\mathcal{Q}} \mathbb{C}(\mathbf{y}_{\mathcal{R}} \,|\, \mathcal{D}_q^{\langle N\rangle}, \mathbf{X}_{\mathcal{R}})^{-1}\Big)^{-1}, \tag{9.5}$$

where $\boldsymbol{\Sigma}_{\mathcal{R}\mathcal{R}}$ is the covariance matrix evaluated at $\mathbf{X}_{\mathcal{R}}$. The predictive mean and covariance of each GP, i.e. $\mathbb{E}(\mathbf{y}_{\mathcal{R}} \,|\, \mathcal{D}_q^{\langle N\rangle}, \mathbf{X}_{\mathcal{R}})$ and $\mathbb{C}(\mathbf{y}_{\mathcal{R}} \,|\, \mathcal{D}_q^{\langle N\rangle}, \mathbf{X}_{\mathcal{R}})$, can be obtained from (9.2). Note that as $\mathbf{y}_{\mathcal{R}}$ is known, the log probability of $\mathbf{y}_{\mathcal{R}}$ under the current model can be evaluated by substituting (9.4) and (9.5) into the following

$$\begin{aligned} L_{\mathcal{R}}^{\langle N\rangle} \coloneqq & -\frac{|\mathcal{R}|}{2}\log(2\pi) + \frac{1}{2}\log\left|\mathbb{C}_{\widehat{p}}^{\langle N\rangle}(\mathbf{y}_{\mathcal{R}})^{-1}\right| \\ & -\frac{1}{2}\left(\mathbf{y}_{\mathcal{R}} - \mathbb{E}_{\widehat{p}}^{\langle N\rangle}(\mathbf{y}_{\mathcal{R}})\right)^{\top} \mathbb{C}_{\widehat{p}}^{\langle N\rangle}(\mathbf{y}_{\mathcal{R}})^{-1} \left(\mathbf{y}_{\mathcal{R}} - \mathbb{E}_{\widehat{p}}^{\langle N\rangle}(\mathbf{y}_{\mathcal{R}})\right), \end{aligned}$$

where $|\mathcal{R}|$ represents the number of references points in $\mathcal{R}$. When $\mathcal{R}$ is arbitrarily selected and sufficiently large, one can consider $L_{\mathcal{R}}^{\langle N\rangle}$ as a proxy for the likelihood of training examples for LGPC. Hence, I hereinafter call $L_{\mathcal{R}}$ the log *pseudo*-likelihood.

As a consequence, the problem of the optimal selection $\mathcal{A}^*$ at time $N+1$ can be formulated as

$$\mathcal{A}^* \coloneqq \arg\max_{\mathcal{A}\subseteq\mathcal{Q}} L_{\mathcal{R}}^{\langle N+1\,|\,\mathcal{A}\rangle} - L_{\mathcal{R}}^{\langle N\rangle}, \text{ subject to } |\mathcal{A}| \leq S,$$

which is unfortunately a combinatorial problem and cannot be solved efficiently. However, it is worth to highlight that the increment of pseudo-likelihood $L_{\mathcal{R}}^{\langle N+1\,|\,\mathcal{A}\rangle} - L_{\mathcal{R}}^{\langle N\rangle}$ satisfies the *diminishing returns* behavior. That is, adding a new GP to the selection $\mathcal{A}$ increases the pseudo-likelihood more, if one has selected few GPs; and less, if one has already selected many GPs. This formalism can be formalized using the combinatorial concept of submodularity [118]. Specifically, let $F(\mathcal{A}) \coloneqq L_{\mathcal{R}}^{\langle N+1\,|\,\mathcal{A}\rangle} - L_{\mathcal{R}}^{\langle N\rangle}$, the submodular characteristic of F indicates that for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{Q}$ and $q \in \mathcal{Q} \setminus \mathcal{B}$ it holds that $F(\mathcal{A}\cup\{q\}) - F(\mathcal{A}) \geq F(\mathcal{B}\cup\{q\}) - F(\mathcal{B})$.

Interest of optimizing a submodular function has grown in the machine learning community recently [99, 101, 100]. In practice, heuristics such as *greedy selection* are often used. The greedy algorithm starts with the empty set, and iteratively adds the element $q^* \coloneqq \arg\max_{q\in\mathcal{Q}\setminus\mathcal{A}} F(\mathcal{A}\cup\{q\})$, until S elements have been selected. A fundamental result by [118] stated that for submodular functions, the greedy algorithm achieves at least a constant fraction $(1-1/e)$ of the objective value obtained by the optimal solution. Moreover, no polynomial time algorithm can provide a better approximation guarantee unless $\mathsf{P} = \mathsf{NP}$ [60].

Note that evaluating $F(\mathcal{A}\cup\{q\})$ can be expensive as it is required to re-compute (9.4) and (9.5) for all $q \in \mathcal{Q}$ in each iteration. In fact, such computation is unnecessary due to the submodularity of F [112]. Define the *marginal benefit* of q as $\Delta_q \coloneqq F(\mathcal{A}\cup\{q\}) - F(\mathcal{A})$, the submodularity indicates that Δ_q never increases over iterations. Based on this observation, the greedy selection scheme is given in Fig. 9.4. The algorithm starts with the set $\mathcal{A}$ being empty, and $\mathcal{J}$ containing the indices of all GPs. In the first iteration, the GP with maximal Δ is selected from $\mathcal{J}$ and added to $\mathcal{A}$. This is achieved by evaluating the marginal benefit for all GPs in the committee. An ordered list of $\{\Delta_j\}_{j\in\mathcal{J}}$ is maintained. From the second iteration, only the top GP in this ordered list will

Input : desired size of selection S (≥ 2)
Output: greedy selection $\mathcal{A}$

Initialization $\mathcal{A} \leftarrow \emptyset$, $\mathcal{J} \leftarrow \{1, \ldots, Q\}$;
$\forall j \in \mathcal{J} : \Delta_j \leftarrow F(\mathcal{A} \cup \{j\}) - F(\mathcal{A})$;
$j^* \leftarrow \arg\max_{j \in \mathcal{J}} \Delta_j$;
$\mathcal{A} \leftarrow \mathcal{A} \cup \{j^*\}$, $\mathcal{J} \leftarrow \mathcal{J} \setminus \{j^*\}$;
for $s \leftarrow 2$ **to** S **do**
    repeat
        $j^* \leftarrow \arg\max_{j \in \mathcal{J}} \Delta_j$;
        $\Delta_{j^*} \leftarrow F(\mathcal{A} \cup \{j^*\}) - F(\mathcal{A})$;
        if $\forall j \in \mathcal{J} \setminus \{j^*\} : \Delta_{j^*} > \Delta_j$ **then**
            $\mathcal{A} \leftarrow \mathcal{A} \cup \{j^*\}$, $\mathcal{J} \leftarrow \mathcal{J} \setminus \{j^*\}$;
    until $|\mathcal{A}| = s$;

Figure 9.4.: Greedy subset selection for LGPC.

be evaluated. If the new marginal benefit of that GP stays on top, then it is added to $\mathcal{A}$. Otherwise, the list of marginal benefits is re-sorted and, subsequently, the new top GP is evaluated.

In addition to Algorithm 1, it is also possible to solve this problem using hybrid genetic algorithms and, in particular, random permutation generators [52]. These methods may be configured to search for a single subset $\mathcal{A}$, or an ensemble of subsets $\mathcal{A}_i$ which all keep the value of $F(\mathcal{A} \cup \{q\}) - F(\mathcal{A})$ at a reasonable level, and on other hand, they are minimal in size and have minimal pairwise intersections. Although the mathematical background of those methods seems to be different, it would be interesting to look at them for more general comparison as a future work.

9.2.2. Incremental Update of LGPC

Once a set of GPs is selected by Fig. 9.4, the problem turns to updating these GPs for the data inclusion in an incremental fashion. Although the update of inputs $\mathbf{X}$ and outputs $\mathbf{y}$ can be done straightforwardly, the update of a covariance matrix $\mathbf{K}$ and its inverse $\mathbf{J} \coloneqq \mathbf{K}^{-1}$ is more complicated. Specifically, the effect of a new point $\mathbf{x}_{N+1}$ on $\mathbf{K}$ and $\mathbf{J}$ can be expressed as

$$\mathbf{K}^{\langle N+1 \rangle} \coloneqq \begin{bmatrix} \mathbf{K}^{\langle N \rangle} & \mathbf{u}^\top \\ \mathbf{u} & v \end{bmatrix}, \qquad \mathbf{J}^{\langle N+1 \rangle} \coloneqq \begin{bmatrix} \mathbf{J}^{\langle N \rangle} + \frac{1}{\mu}\mathbf{g}\mathbf{g}^\top & \mathbf{g} \\ \mathbf{g}^\top & \mu \end{bmatrix},$$

with $\mathbf{u} \coloneqq [k(\mathbf{x}_{N+1}, \mathbf{x}_1), \ldots, k(\mathbf{x}_{N+1}, \mathbf{x}_N)]^\top$ and $v \coloneqq k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1})$. Following the partitioned inversion matrix equations, it yields

$$\mathbf{g} \coloneqq -\mu \mathbf{J}^{\langle N \rangle}\mathbf{u}, \qquad \mu \coloneqq \left(v - \mathbf{u}^\top \mathbf{J}^{\langle N \rangle}\mathbf{u}\right)^{-1}.$$

In practice, Cholesky factorization is often used so that $\mathbf{L}^\top\mathbf{L} \coloneqq \mathbf{K}$, which leads to a more efficient and accurate solution for frequently occuring terms such as $\mathbf{x}^\top\mathbf{K}^{-1}\mathbf{x} = \|\mathbf{L}^{-1}\mathbf{x}\|^2$ and $\log|\mathbf{K}| = 2\left(\mathbf{1}^\top \log(\mathrm{diag}\,(\mathbf{L}))\right)$. The lower triangular matrix $\mathbf{L}$ can be also updated incrementally

such that

$$\mathbf{L}^{\langle N+1\rangle} \coloneqq \begin{bmatrix} \mathbf{L}^{\langle N\rangle} & \mathbf{0} \\ \boldsymbol{l}^\top & \eta \end{bmatrix},$$

where $\boldsymbol{l}$ can be solved by $\mathbf{L}^{\langle N\rangle}\boldsymbol{l} = \mathbf{u}$, and subuently, $\eta \coloneqq \sqrt{v - \|\boldsymbol{l}\|^2}$.

Further notice that the model may deal with an endless stream of data during online learning. Thus, it is necessary to limit the number of training examples maintained by each GP and delete old training examples when necessary. Let m be the index of training example being deleted. Construct $\mathbf{P} \coloneqq \mathbf{I} - (\boldsymbol{\delta}_m - \boldsymbol{\delta}_{T+1})(\boldsymbol{\delta}_m - \boldsymbol{\delta}_{T+1})^\top$, as a $(T+1)$-dimensional permutation matrix, where $\boldsymbol{\delta}_m$ is a $(T+1)$-dimensional zero vector with one on the m^{th} dimension. When a new point comes in, it is first appended to $\mathbf{K}^{\langle N+1\rangle}$. Then, the deletion of the m^{th} example can be performed efficiently using $\left[\mathbf{P}\mathbf{K}^{\langle N+1\rangle}\mathbf{P}\right]_{\Uparrow}$, where $[\cdot]_{\Uparrow}$ represents shrinking a $(T+1)$-dimensional matrix or vector to a T-dimensional one by removing the last row and column of it. Thus, the non-increasing update equation of $\mathbf{J}$ is given by

$$\mathbf{J}^{\langle N+1\rangle} \coloneqq \left[\mathbf{P}\mathbf{J}^{\langle N+1\rangle}\mathbf{P}\right]_{\Uparrow} - \frac{1}{r}\mathbf{s}^\top\mathbf{s},$$

where $\mathbf{s} \coloneqq [[k(\mathbf{x}_m, \mathbf{x}_1), \ldots, k(\mathbf{x}_m, \mathbf{x}_{N+1})]\mathbf{P}]_{\Uparrow}$ and $r \coloneqq k(\mathbf{x}_m, \mathbf{x}_m)$.

So far I have not said which training example should be deleted. One simple method is to choose it randomly or to remove the oldest training example over time. Alternatively, one can remove the point that yields maximal mutual information with the new point. In this work, I follow the score measure introduced by [40]. Specifically, for each point i in set $\mathcal{D}_q$ the score is given by $\xi_t \coloneqq \frac{\alpha_t}{J_{tt}^{\langle N+1\rangle}}$, where α_t is the t^{th} element of $\mathbf{J}^{\langle N+1\rangle}\mathbf{y}$. If a deletion is needed for the q^{th} GP, then the training example with minimal ξ will be removed from $\mathcal{D}_q$. The scores are computationally cheap as they can be calculated in linear time. Although there may exist more sophisticated selection schemes, they usually consume more computational time and hence are not considered in this work.

9.2.3. Predictions of Query Points

Given a query point $\mathbf{x}_*$, the predictive mean and variance can be calculated by evaluating (9.4) and (9.5) straightforwardly. One can observe that the way of combining predictions in (9.4) automatically assigns less weight (through the inverse predictive variance) to those GPs that are uncertain about their predictions. The time complexity is $\mathcal{O}(QT)$ for predicting the mean and variance of a test point.

For the sake of efficiency and accuracy, it is reasonable to only invoke nearby GPs in a neighborhood of $\mathbf{x}_*$ for the prediction. The key observation is that $k(\mathbf{x}_*, \mathbf{x})$ depends merely on the constant σ if $\mathbf{x}_*$ is far away from $\mathbf{x}$. As σ is randomly initialized in LGPC, a poor prediction could be given by the q^{th} GP when $\mathbf{x}_*$ is far away from all points in the set $\mathcal{D}_q$. The search of neighboring GPs can be performed by evaluating $\frac{1}{|\mathcal{D}_q|}\sum_{\mathbf{x}\in\mathcal{D}_q} k(\mathbf{x}_*, \mathbf{x})$ for all $\{\mathcal{D}_q\}_{q=1}^{Q}$ and, subsequently, selecting those having largest values.

9.3. Experiments

Two sets of experiments were carried out to validate LGPC. First, we compared the accuracy and the efficiency of LGPC with the standard GPR and state-of-the-art online regression methods. Second, I investigated several factors that affect the performance of LGPC in order to gain more insights of it. An application to the mouse-trajectory prediction is demonstrated at the end.

The experiments were conducted on six large data sets downloaded from the Internet [1]. On each data set, we linearly rescaled into the range of $[0, 1]$ and randomly held out half of the data for training; the remaining was used as a test set. Each experiment was repeated 10 times.

Five baseline methods were employed for comparison. They were standard GP regression (GPR), sparse GP using pseudo-inputs (SGPP) [146][2], local GP regression (LoGP) [121][3], Bayesian committee machine (BCM) [157] and sparse online GP regression (SOGP) [40][4]. Note that GPR and SGPP are offline algorithms, they are presented here to show the performance when the whole training data is given beforehand. A Gaussian kernel function with white noise was used as the covariance function for all methods, which was obtained by setting $\mathbf{W}$ in (9.1) as the identity matrix. The maximum number of inducing inputs in SGPP and SOGP was 50. The threshold for creating a new local model in LoGP was 0.5. The size of the committee was 20 for BCM and LGPC, in which each GP maintained at most 100 training examples. For BCM and LGPC, the first 20 training examples were sequentially assigned to each member for initialization. The reference set of LGPC had a size of 3, which consisted of the current training example and two examples randomly sampled from the aforetime data. The number of selected GPs in LGPC for data inclusion was 5 (i.e. $S := 5$ in Fig. 9.4). Moreover, three variations of BCM were employed in the experiment. BCM_o denotes that each time only one GP is randomly selected for data inclusion; BCM_s represents randomly selecting a subset with a size of 5, which corresponds to a randomized counterpart of LGPC; and BCM_a represents selecting all GPs. For predicting test inputs, all GPs in the committee of BCM and LGPC were invoked. The hyperparameters were randomly initialized for all methods. The conjugate gradient method was employed to optimize the hyperparameters for GPR, SGPP, SOGP and LoGP. For SOGP, an EM algorithm with 10 cycles was built for jointly optimizing the posterior process and the hyperparameters.

9.3.1. Comparison of Predictive Accuracy

The comparison of predictive accuracy between different GP methods is summarized in Table 9.1, where the root mean square error was used as the evaluation metric. It is evident from the results that, LGPC gave a considerably lower test error than LoGP, BCM and SGPP. In particular, LGPC showed a significant improvement over all BCM variants on the majority data sets, which indicates the effectiveness of the proposed subset selection strategy. Empirically, I found that good performance could have been achieved after learning from first few thousands examples. After

[1] `delta`: $7,129 \times 6$, `http://www.dcc.fc.up.pt/~ltorgo/Regression/`;
`bank`: $8,192 \times 8$, `http://www.cs.toronto.edu/~delve/`;
`cpuact`: $8,192 \times 12$, `http://www.cs.toronto.edu/~delve/`;
`elevator`: $8,752 \times 17$, `http://www.dcc.fc.up.pt/~ltorgo/Regression/`;
`houses`: $20,640 \times 8$, `http://lib.stat.cmu.edu/datasets/`;
`sarcos`: $44,484 \times 21$, `http://www.gaussianprocess.org/gpml/data/`.

[2] `http://www.cs.man.ac.uk/~neill/gp/`

[3] `http://www.ias.informatik.tu-darmstadt.de/Member/DuyNguyen-Tuong`

[4] `http://www.cs.ubbcluj.ro/~csatol/`

that, the accuracy of LGPC did not change significantly. On `delta`, `cpuact` and `houses`, the performance of LGPC was comparable to SOGP. In fact, the performance of LGPC can be further improved by allowing each GP to maintain more training examples, as it is shown in the third experiment.

Table 9.1.: LGPC versus baseline methods. The root mean square error on different test sets were measured. Results were averaged over ten runs. Smaller value indicates better performance.

Model	`delt`	`bank`	`cpua`	`elev`	`hous`	`sarc`
LGPC	0.041	0.084	0.072	0.053	0.157	0.032
LoGP	0.065	0.188	0.219	0.107	0.232	0.089
BCM_o	0.044	0.113	0.115	0.066	0.164	0.070
BCM_s	0.043	0.108	0.114	0.069	0.180	0.073
BCM_a	0.045	0.122	0.119	0.083	0.203	0.077
SOGP	0.040	0.047	0.074	0.038	0.143	0.023
GPR	0.039	0.041	0.030	0.031	0.115	0.016
SGPP	0.045	0.061	0.079	0.065	0.161	0.095

9.3.2. Comparison of Computation Speed

The comparison of training and prediction speed is shown in Fig. 9.5. Different online methods were trained (tested) on `houses` data set with increasing training (testing) examples, i.e. 500, $1,000$, $2,000$, $4,000$ and $8,000$ data points, respectively. The setup of each method was same as in the last experiment. For the sake of fair comparison, the update of kernel matrix for LGPC, LoGP and BCM was implemented in the same manner as described in Section 9.2.2.

As can be seen in Fig. 9.5, in both training and prediction phrases, LGPC showed a substantial reduction of time comparing to GPR and LoGP. In particular, LGPC took less time for learning $8,000$ points (210s) than SGPP (240s) and SOGP (340s). On the other hand, SGPP and SOGP were found to be extremely efficient in the prediction, as their time cost only increased at a very low pace with respect to the number of test points. This is due to the fact that the prediction of SGPP and SOGP involves only a small covariance matrix based on 50 inducing inputs, whereas LGPC invokes all GPs in the committee for prediction. Nonetheless, it is possible to speed up LGPC by invoking only the nearest GP for the prediction as aforementioned. In short, LGPC is a more appropriate choice than SOGP in a real-time application which requires fast training.

One may notice that LGPC took more training time than BCM_s and its speed advantages over BCM_a was not dramatic, which is slightly disappointing. This is attributed to the fact that, although LGPC saves the computational resources by limiting the number of GPs for data inclusion, it spends extra computational time on selecting a near-optimal subset with Fig. 9.4. The prediction complexity of LGPC, BCM_o, BCM_s and BCM_a was virtually same. Nonetheless, it should be noted that LGPC outperformed all BCM variants significantly on `houses` in terms of predictive accuracy as detailed in Table 9.1, which makes LGPC overall more preferable than BCM.

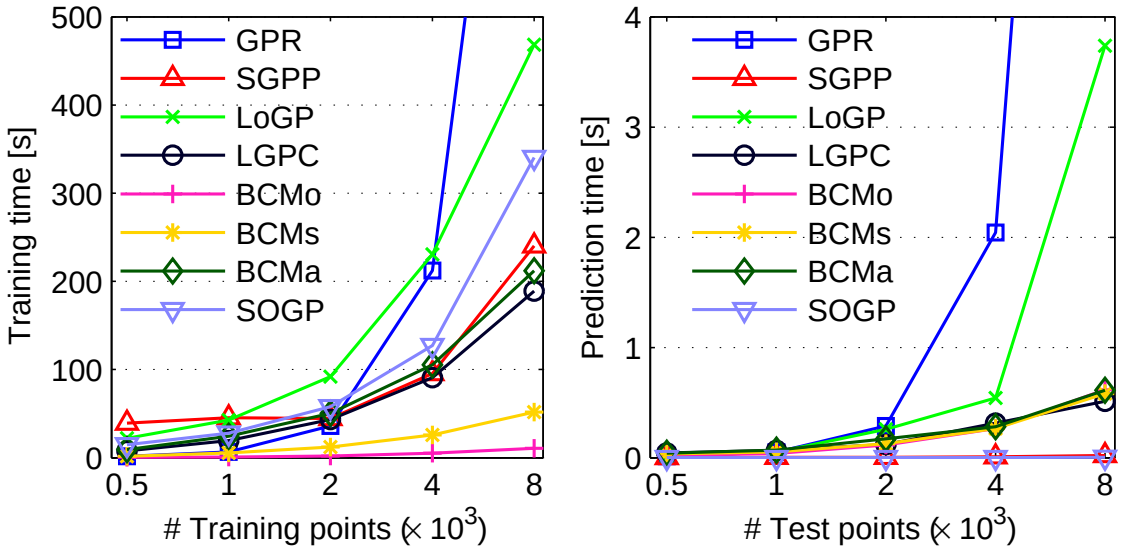


Figure 9.5.: Time cost in second (averaged over 10 runs) required for training and predicting, respectively. In each run, a training set and a test set were randomly sampled from houses data set and the time cost was measured respectively. The training and prediction time of $8,000$ data points required for GPR was 1100s and 15s, respectively. Note that prediction time of LGPC can be reduced to 0.02s if only the nearest GP is invoked for predicting a test point.

Table 9.2.: The root mean square error of LGPC on different test sets. Results were averaged over ten runs. Smaller value indicates better performance.

(a) Predictive error w.r.t. the size of the committee of LGPC.

Q	delt	bank	cpua	elev	hous	sarc
5	0.043	0.093	0.087	0.065	0.171	0.034
10	0.042	0.102	0.079	0.056	0.167	0.033
15	**0.041**	0.085	0.075	0.057	0.161	0.034
20	0.041	0.084	**0.072**	0.053	0.157	0.033
25	0.041	**0.076**	0.080	0.053	0.156	**0.032**
30	0.041	0.085	0.095	**0.052**	**0.155**	0.032

(b) Predictive error w.r.t. maximum number of examples maintained by each GP member.

T	delt	bank	cpua	elev	hous	sarc
50	0.042	0.114	0.098	0.072	0.172	0.040
100	0.041	0.084	0.072	0.053	0.157	0.032
150	**0.040**	0.068	0.057	0.040	0.152	0.028
200	0.040	**0.054**	**0.050**	**0.039**	**0.146**	**0.022**

9.3.3. Exploration of Model Parameters

In order for LGPC to be a practical tool in real-world applications, it is necessary to make decisions about the details of its specification. Fortunately, there are not many free parameters in LGPC, as all hyperparameters are randomly initialized and are fixed during the learning process[5]. The exploration focused on three parameters that mainly govern the performance of LGPC. Namely, the committee size Q, the maximum number of maintained training points T and the size S of the selected subset for data inclusion.

The performance of LGPC with respect to different sizes of the committee is summarized in Table 9.2(a), where $S := 5$ and $T := 100$. On `delta`, `bank` and `cpuact`, the predictive accuracy reached its peak when the size of the committee was around 20. After that, the performance dropped slightly. On larger data sets such as `houses` and `sarcos` the test error had a steady decline as Q increasing. This indicates that it suffices to employ a small committee for learning a small amount of data. For a large data set increasing the size of the committee provides more capability to account for the complex pattern, which generally leads to higher predictive accuracy.

Table 9.2(b) summarizes the test results of LGPC with respect to different settings of T, where $S := 5$ and $Q := 20$. As expected, one can improve the predictive accuracy significantly by allowing each GP to maintain more training examples. Moreover, when $T := 200$ it was observed that the high accuracy was achieved much earlier than $T := 50$; and the performance was often more robust afterwards.

Finally, to study the performance with respect to different sizes of the selected subset, I fixed $T := 100, Q := 20$ and trained LGPC with $S := 2, 4, 6, 8$ and 10, respectively. It was found that on `delta` and `sarcos` the predictive accuracy was not sensitive to the size of selected subset. On `cpuact`, `elevator` and `houses`, selecting more than two GPs slightly degraded the performance. The optimal size for `bank` was 6. In short, it seems that the optimal value of S varies with data sets.

9.3.4. Mouse-Trajectory Prediction

The mouse-trajectories of continuous motor movements provide a way of measuring the ongoing cognitive processes that lead to the participant's final choice. A main advantage of modeling trajectories with Gaussian process regression over the various summary statistics used previously [65] (e.g. maximum deviation, area under curve) is that less information is thrown away. The posterior density of a GP shows a normatively correct summary of the data.

I applied LGPC for learning mouse-trajectory of different users in an Internet banking scenario. To collect the data, I developed a website for simulating an environment, in which participants were asked to transfer funds to a dummy account. A complete procedure was composed of five interfaces, i.e. login, account overview, transaction details, TAN authentication and confirmation. Ten participants were involved and each with three trials; the input information was *same* for all trials. A Javascript code was developed for tracking mouse coordinates on every `onmousemove` event. The trajectories of the first two trials (ca. 2700 points/user) were used for training models. The goal was to predict the trajectory of the last trial (ca. 1000 points/user).

The predicted trajectories of three users using LGPC, SOGP and offline GPR are visualized in

[5] I did try few heuristics for initializing the hyperparameters, such as initializing 20 Gaussian kernels with different widths (e.g. $2^{-9}, \ldots, 2^{9}, 2^{10}$) and the same noise level; or setting $\mathbf{W}$ in (9.1) for each GP as a random block matrix. However, such attempts did not yield better predictive accuracy than the random initialization.

Fig. 9.6 and Fig. 9.7. It was observed that users behaved differently even when they were performing the same task. For instance, the first user used the tab key moves the cursor and entered the TAN code with the numpad, resulting a short and simple mouse-trajectory. On the other hand, the third user entered the TAN code using a virtual keyboard on the web page, which made the trajectory sway horizontally. By learning from the first two trials, reasonable predictions of the third trial were obtained from all methods. However, when the interface contained many elements and the trajectory became more complicated, the offline GPR gave noisy predictions as depicted in Fig. 9.6. It can be seen that LGPC performed as good as the state-of-the-art SOGP in learning users' trajectories.

With a new training point arriving about every 10ms (less than one minute of running time will result in thousands of data points), LGPC is a more preferable method due to its fast learning speed. Efficient trajectory prediction would be beneficial in security applications, such as distinguishing between individuals and early warning of identity theft.

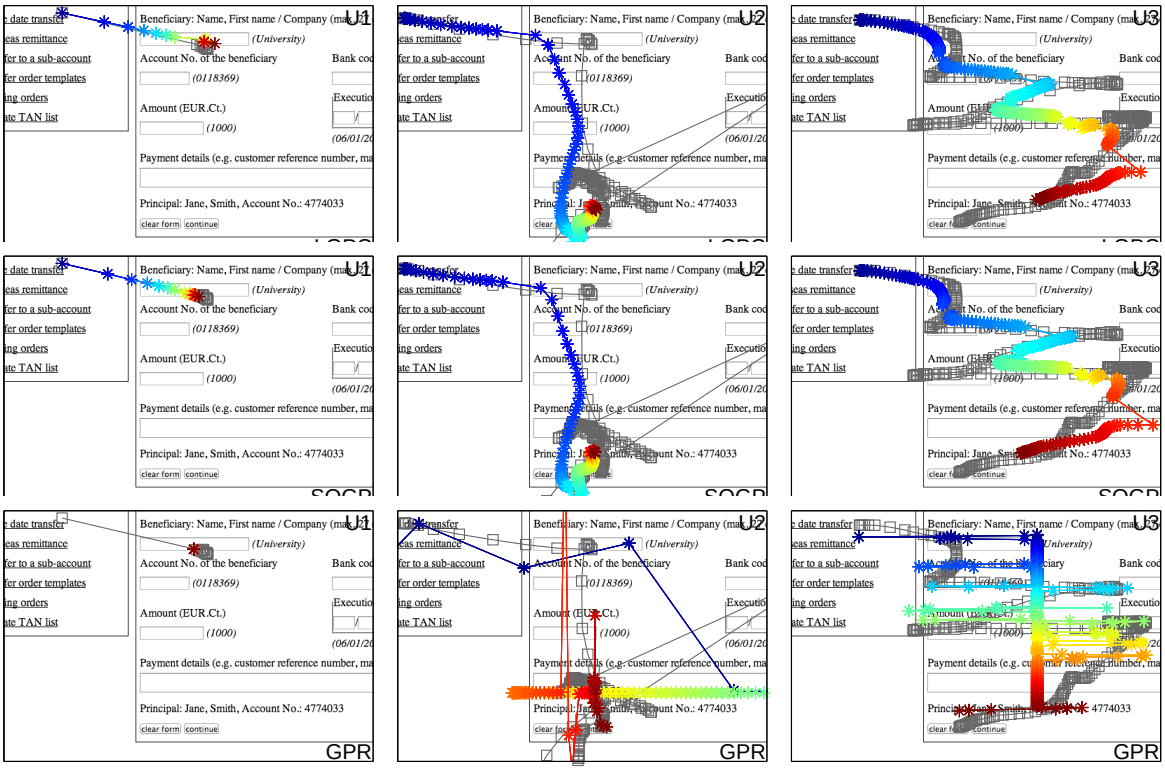


Figure 9.6.: The predictions of mouse-trajectories when users are inputting the transaction information. Each column represents a user. The gray curve with □ denotes a user's trajectory in the third trial. The model's prediction is illustrated by the color curve with ⋆, whose head is blue and tail is red.

9.4. Conclusion

GP faces a low-efficiency problem when it is applied to real-time online applications. This work has proposed a novel method for reducing the computational demand of GP regression. It consists of multiple GPs where each maintains only a small set of training examples. Each time a subset of GPs

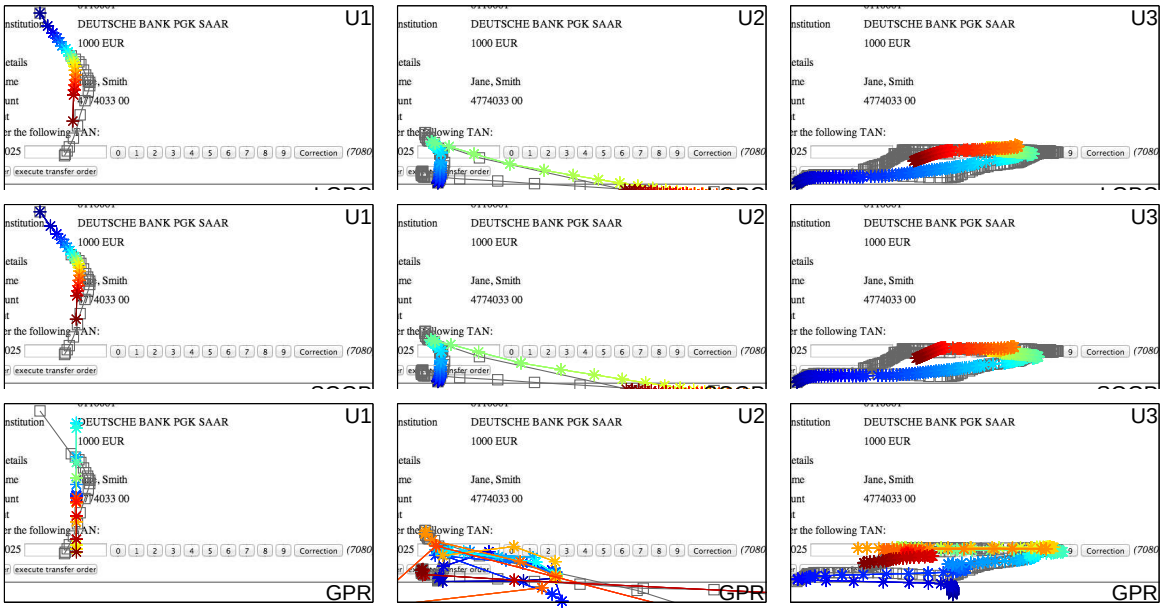


Figure 9.7.: The predictions of mouse-trajectories when users are inputting the security code. Each column represents a user. The gray curve with □ denotes a user's trajectory in the third trial. The model's prediction is illustrated by the color curve with ⋆, whose head is blue and tail is red.

is selected for including newly arrived training examples. The selection is performed by optimizing a submodular function. Unlike previous work, LGPC removes the need for parameter-fitting and requires little tuning efforts. An improvement of accuracy and efficiency over existing online GP methods has been demonstrated in the experiment. In particular, as a modified version of Bayesian committee machine, I have showed that updating a chosen subset of GPs is more effective than updating the whole committee, which leads to better predictive accuracy. As demonstrated in the task of mouse-trajectory prediction, LGPC can be applied to a wide range of real-time applications, such as learning motor dynamics and inferring temporal dynamics of mental phenomena.

An important question for future studies is to determine the optimal size of the committee. One possible way is to expand or shrink the committee size during the online learning. In addition, more effective strategies for initializing the hyperparameters remain to be determined. Furthermore, it should be interesting to infringe the independence between GP members for a further improvement on the accuracy.

Chapter 10

Online Prediction of System Call Sequence with Side Information

In this chapter, I will focus attention on the *sequence prediction* problem and introduce a method for efficiently online learning and predicting the system call sequence. Sequence prediction is a key task in machine learning and data mining. It involves observing a sequence of symbols one at a time and predicting the next symbol before it is revealed. This technique has been successfully applied in a large variety of disciplines, such as stock market analysis, natural language processing and DNA sequencing. The problem of sequence prediction has received considerable attention throughout the years in information theory, machine learning and data mining. Typically, the *Markov property* is assumed when modeling a sequence. That is, a finite history of the past, i.e. the *context*, can be useful in predicting the future. The length of the context is called the *order* of Markov models. Previous work shows ample evidence of the fact that making such an assumption is often reasonable in a practical sense [183, 18]. For instance in natural language processing, it is often well-enough to describe text by a fixed order Markov models (e.g. bigram, trigram), though the next word is not necessarily related to its previous words.

From the information security perspective, a related application is modeling the execution path of a process on a desktop/mobile system in real-time. Each process produces an ordered sequence of system calls which request different services from the operating system. In this case, each symbol in the sequence represents a system call accompanied with arguments and a return value. An illustrative example is depicted in Fig. 10.1.

Three remarks are in order. First, some system calls have a long range dependency. For instance, after creating a file the process may produce hundreds of system calls before it finally closes the file. In this case, the dependency between `creat` and `close` can not be observed from a short context of `close`. Although one can increase the order of Markov models to capture information from a long distant context, it is often difficult in practice due to the requirement of vast amounts of training data and more sophisticated smoothing algorithms [27]. In general, the length of context needed to make an accurate prediction is not constant, but rather depends on the recently executed system calls. Second, the information from the arguments and return values (e.g. file descriptor, memory address and signal) may be also indicative in predicting the next system call. Considering a process repetitively reads data from the file 1 and writes data to the file 2. A resulting system call sequence may look like

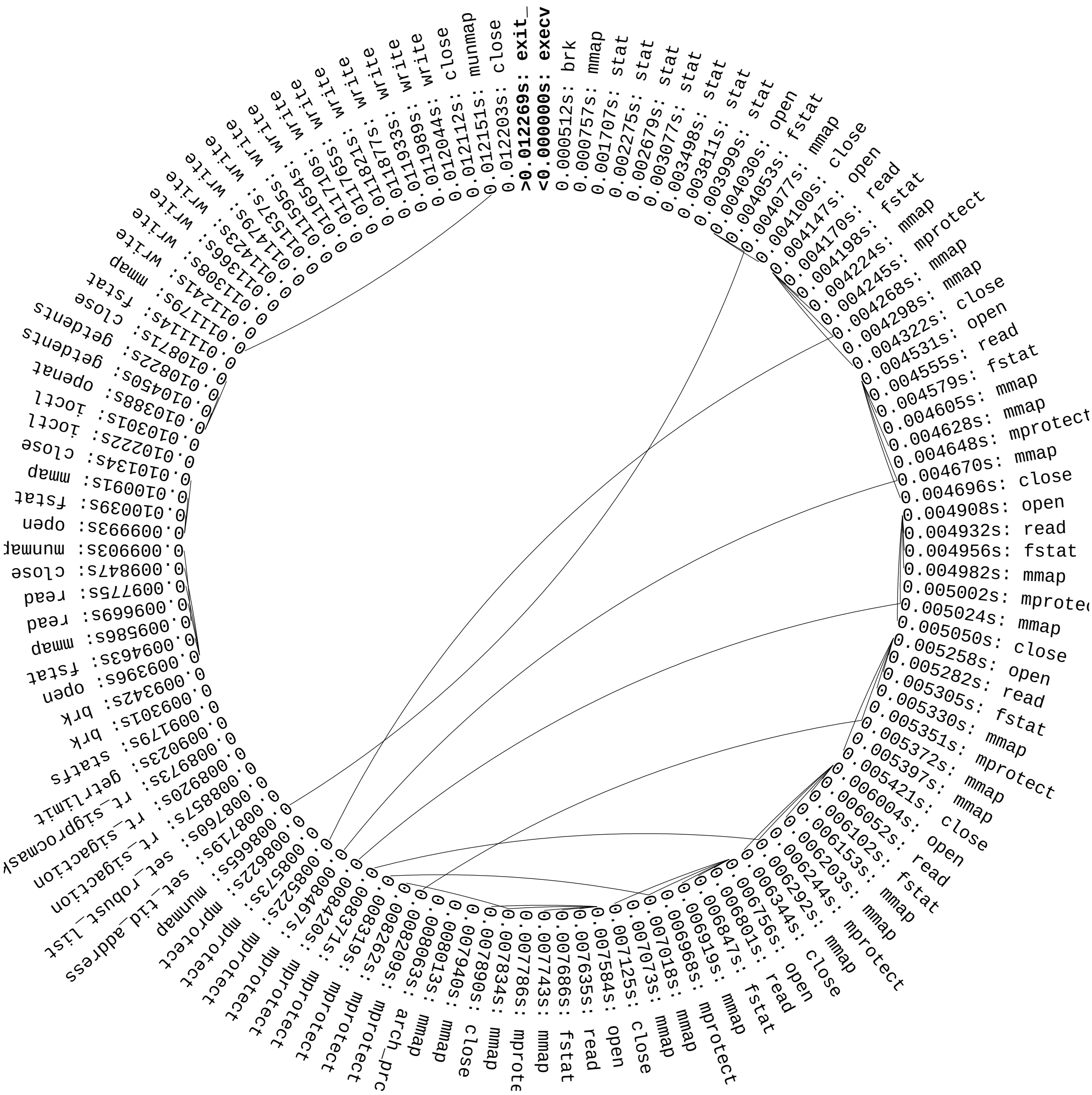


Figure 10.1.: A circular plot of a system call trace when running `ls` on Linux, which was collected using `strace`. System calls are plotted clockwise, starting with `execve` and ending with `exit` on top. A time stamp is labeled in front of each system call. A curve connects two system calls if the return value of the former was used as an argument of the latter.

Call	Argument	Return
open	(”/lib/librt.so”, ORDONLY)	= 3
read	(3, ”“177ELF“2“1“1”)	= 832
fstat	(3, -stmode=SIFREG, stsize=317)	= 0
mmap	(NULL, 4096, PROTREAD—PROTWRITE)	= 0x7f2f
mmap	(NULL, 2129016, PROTREAD)	= 0x7f2f
mprotect	(0x7f2f7, 2093056, PROTNONE)	= 0
mmap	(0x7f2fa, 8192, PROTREAD)	= 0x7f2f
close	(3)	= 0

Table 10.1.: A sample segment of this sequence is detailed, with argument defined within the parentheses. For the sake of clarity, some long arguments (e.g. string) are omitted. The dependencies between the return value and argument are highlighted with arrow lines.

```
open(1)
read(1)
close(1)
open(2)
write(2)
close(2)
open(1)
...
```

Assume that one has observed the above sequence with seven system calls; the goal is to predict the next system call. Without using the knowledge of the arguments, a bigram model based merely on the name of adjacent system call will predict `read` and `write` with even chance. However, as the file 2 has been closed, the correct prediction should be `read`. Although one can solve this problem by extending Markov models with more sophisticated graphical models, incorporating side information is in general not straightforward for probabilistic Markov models. Third, a process may exhibit different behaviors at various points during its lifetime, depending on user's input and the status of the system. In other word, the sequence is usually not stationary and no prior assumption on its distribution should be made. This suggests the necessity of an online model that can be continuously updated, preserving information from a long distant context while giving more emphasis to recent data, so that the stationarity is not required.

I focus on the problem of predicting the next system call given an observed sequence. The solution of this problem can be extremely useful in a wide range of applications, such as anomaly detection [166, 57], buffer cache management in operating system [66], power management in smartphones [124] and sandbox systems [123]. I leverage both context and side information of each system call and model a sequence in an online fashion. The proposed algorithm performs prediction in real-time and can quickly update the model when a prediction error is made.

The rest of the chapter is organized as follows. Section 10.1 briefly reviews previous work on sequence prediction. Subsequently, my novel contribution is highlighted. Section 10.2 describes the problem formulation. I next cast sequence prediction as a linear separation problem in Section 10.3. The proposed method is presented in Section 10.4. Experimental results are demonstrated in Section 10.5. Section 10.6 concludes the chapter and points out some future directions.

10.1. Related Work

The problem of sequence prediction has a fairly long history and has received much attention from the field of game theory [132, 14, 81], information theory [32, 33, 59, 168], and machine learning [82, 125, 51, 170, 171, 56]. One of the most useful tools is context trees, which store informative histories and the probability of the next symbol given these [168, 8]. Context trees use only a few recently observed symbols for prediction. The number of symbols that are used depends on the specific context in which the prediction is made. The motivation for exploring context tree strategies stems from their simplicity and their success in lossless data compression applications [110]. Another family of approach based on Bayesian nonparametric models has generated considerable recent research interest [154, 170, 171]. It is assumed that the distribution of the current symbol is determined by some random process (e.g. Dirichlet process, Pitman-Yor process) governed by its context. The hierarchy is defined recursively to the first symbol in the sequence, on which a global base distribution is defined. These models give state-of-the-art performance in language modeling, however, inference in such models is not straightforward. It often relies on repeated random sampling (e.g. Markov chain Monte Carlo), which can be time-consuming in practice.

Another related line of work is online learning, which takes place in a sequence of consecutive rounds. On each round, the learner is given a question and is required to provide an answer to this question. The performance of an online learning algorithm is measured by the cumulative loss suffered by the prediction along the run on a sequence of question-answer pairs. The Perceptron algorithm [114, 134, 122] is perhaps the first and simplest online learning algorithm designed for answering yes/no questions. Adaptations of the Perceptron for multiclass categorization tasks include [54, 39]. As the Perceptron algorithm is essentially a gradient descent (first-order) method, recent years have seen a surge of studies on the second-order online learning [20, 17, 53]. For example, the confidence-weighted algorithm [53] maintains a Gaussian distribtuion over some linear classifier hypotheses and employs it to control the online update of parameters. Several work has followed this idea and showed that parameters' confidence information can significantly improve online learning performance [53, 37, 38, 163].

The system call sequence was mainly studied by computer security researchers in the early days [62, 103, 166]. They used patterns in the sequence to identify misuses and intrusions in systems. To contain the attack preemptively, plan recognition was developed, aiming at recognizing and predicting goals based on observed system call sequences [70, 61]. Recently, the problem of system call prediction attracted much attention due to its importance in many applications. For example, in a sandbox the amount of time that a process must suspend for a security check can be eliminated when the current system call is correctly predicted, yielding a more efficient sandbox implementation [123]. On mobile devices it has been demonstrated that system calls prediction can be used to design user-oriented prefetching techniques [66] and reduce power consumption [124]. However, most of these studies are over-simplistic in the sense that they focused only on the names of system calls and overlooked the arguments and return values. One possible reason is the difficulty in representing this side information, which requires a different modeling technique, such as rule learning [25, 152]. Hence, it can not be incorporated into a sequence prediction model in a straightforward manner.

In this chapter I introduce a novel online algorithm for predicting system calls in a sequence. My algorithm combines the ideas from both context trees [51, 90] and second-order online learning algorithms [53, 37, 38, 163]. Unlike previous work on system call prediction that only uses context

information, I also consider side information such as arguments, return values and structures into learning and prediction. The side information can be straightforwardly incorporated into my model, giving a further boost to the accuracy of prediction. Furthermore, I propose several techniques to improve the efficiency (in terms of both time and memory) of my algorithm on long sequences, yielding a good scalabilty on big data.

10.2. Problem Formulation

Denote the alphabet of the observed symbols as $\Sigma \coloneqq \{1, \ldots, K\}$. Let Σ^* be the set of all finite length sequences over the alphabet Σ. Specifically, the empty sequence ϵ is included in Σ^*. I focus on the online learning framework, where learning is performed in rounds. Let $x^{\langle t \rangle} \in \Sigma$ be the t^{th} symbol in a sequence. Denote $\mathbf{x}^{\langle 1:t-1 \rangle} \in \Sigma^*$ be the *context* of $x^{\langle t \rangle}$, i.e. $\mathbf{x}^{\langle 1:t-1 \rangle} \coloneqq x^{\langle 1 \rangle}, \ldots, x^{\langle t-1 \rangle}$. For completeness, let $\mathbf{x}^{\langle t:t-1 \rangle} \coloneqq \epsilon$. On round t, the algorithm first predicts $\widehat{x}^{\langle t \rangle} \in \Sigma$ according to its current prediction rule and the context $\mathbf{x}^{\langle t:t-1 \rangle}$. After that, the true symbol $x^{\langle t \rangle}$ is revealed and the algorithm suffers a loss which reflects the degree to which its prediction was wrong. The algorithm then has the option to modify its prediction rule, with the explicit goal of improving the accuracy of its predictions for the rounds to come.

Assume that any symbol in the sequence is determined by its context, the problem of sequence prediction can be formulated as finding a function $f : \Sigma^* \to \Sigma$. To predict the t^{th} symbol one can simply set $\widehat{x}^{\langle t \rangle} \coloneqq f(\mathbf{x}^{\langle 1:t-1 \rangle})$. I generalize this definition and allow the algorithm to output predictions from a real-valued set Y. Specifically, let $Y \coloneqq \mathbb{R}^K$ and $f : \Sigma^* \to Y$, where a prediction $\mathbf{y} \in Y$ is interpreted as a degree of confidence for each of the symbols in Σ. Consequently, the mapping from a score vector $\mathbf{y}$ to an actual symbol in Σ is via $\widehat{x} \coloneqq \arg\max_{k \in \Sigma} y_k$. On round t, the loss of f is measured by a zero-one loss function $\ell_{\mathbb{1}} \left(f; (\mathbf{x}^{\langle 1:t-1 \rangle}, x^{\langle t \rangle}) \right)$. That is, $\ell_{\mathbb{1}}$ is zero if $\widehat{x}^{\langle t \rangle} = x^{\langle t \rangle}$. Therefore, my ultimate goal is to incrementally learn a function f which minimizes

$$\frac{1}{T} \sum_{t=1}^{T} \ell_{\mathbb{1}} \left(f; (\mathbf{x}^{\langle 1:t-1 \rangle}, x^{\langle t \rangle}) \right),$$

where T is the length of the sequence.

10.3. Sequence Prediction as Linear Separation

Having described a general scheme for sequence prediction, I now focus on determining the form of f to obtain a concrete algorithm. In what follows I cast the sequence prediction problem as the problem of linear separation in a Hilbert space, which is a popular topic in machine learning. The reader will see that by doing so one can harness powerful machine learning tools such as the Perceptron algorithm [134, 122] and online convex programming [139] to my purpose.

As it was suggested in the beginning of this chapter, the number of previous symbols needed to make an accurate prediction is usually not constant, but rather depends on the identity of those symbols. With this consideration in mind, define a *suffix-closed* set $V \subset \Sigma^*$ such that for every $\mathbf{s} \in V$, every suffix of $\mathbf{s}$ (including ϵ) is also contained in V. To allow the algorithm to look as far back as needed, the set V can be set large enough. Specifically, let $\mathcal{H}$ be the Hilbert space of

square integrable functions $\psi : V \to \mathbb{R}$ endowed with the inner product

$$\langle \zeta, \psi \rangle = \sum_{\mathbf{s} \in V} \zeta(\mathbf{s})\psi(\mathbf{s}),$$

and the induced norm $\|\zeta\| = \sqrt{\langle \zeta, \zeta \rangle}$. Note that if one can bound $|V|$ by a constant, then the Hilbert space $\mathcal{H}$ is isomorphic to the $|V|$-dimensional vector space, i.e. $\mathbb{R}^{|V|}$.

On round t the context $\mathbf{x}^{\langle 1:t-1 \rangle}$ is observed, this sequence is mapped to the function $\psi \in \mathcal{H}$ as follows

$$\psi(\mathbf{s}^{\langle 1:i \rangle}) \coloneqq \begin{cases} 1 & \text{if } \mathbf{s}^{\langle 1:i \rangle} = \epsilon \\ e^{-\rho i} & \text{if } \mathbf{s}^{\langle 1:i \rangle} \in \mathrm{suf}(\mathbf{x}^{\langle 1:t-1 \rangle}) \\ 0 & \text{otherwise} \end{cases}, \tag{10.1}$$

where $\mathrm{suf}(\mathbf{x}^{\langle 1:t-1 \rangle})$ denotes the set of all suffixes of $\mathbf{x}^{\langle 1:t-1 \rangle}$. The decay factor $\rho > 0$ is a predefined hyperparameter and mitigates the effect of long contexts on the function ψ. It is noticed from Eq. (10.1) that all suffixes of $\mathbf{x}^{\langle 1:t-1 \rangle}$ are mapped to non-zero values; the value tends to decrease as the length of suffix increases. This idea expresses the assumption that symbols appearing earlier in a sequence have the least importance in modeling the current symbol. As the reader will see in Section 10.4.3, this assumption can be infringed to some extent by incorporating side information into my model.

Having mapped sequences to functions in $\mathcal{H}$, the next step is to create separating hyperplanes in $\mathcal{H}$ for prediction. I employ a *multi-class context tree*. A multi-class context tree is a K-ary tree, each node of which represents one of the sequences in V. Specifically, the root of the tree represents the empty sequence ϵ. The node that represents the sequence $\mathbf{x}^{\langle i:j \rangle}$ is the child of the node representing the sequence $\mathbf{x}^{\langle i+1:j \rangle}$. An observed sequence thus defines a path from the root of the tree to one of its nodes. Note that this path can either terminate at an inner node or at a leaf. I associate each node with a K-dimensional vector. In other words, a multiclass context tree can be represented as a function $\tau : V \to \mathbb{R}^K$. An illustrative example is given in Fig. 10.2. In particular, if one only looks at the k^{th} element of the vector on every node and denote the corresponding context tree as $\tau_k : V \to \mathbb{R}$, then it is easy to verify that τ_k is embedded in $\mathcal{H}$.

To construct the context tree on rounds, I initialize $\tau^{\langle 1 \rangle}$ to be a tree of a single (the root) node which assigns a weight of zero to the empty sequence, i.e. $V^{\langle 1 \rangle} \coloneqq \{\epsilon\}$. After receiving $\mathbf{x}^{\langle 1:t-1 \rangle}$, a trivial solution is adding all sequences in the set $\mathrm{suf}(\mathbf{x}^{\langle 1:t-1 \rangle})$ to $V^{\langle t \rangle}$ and associate each of which with an undetermined vector in $\mathbb{R}^K$. The method for determining the value of these vectors will be presented in Section 10.4.1. For a long sequence adding all suffixes to the tree can impose serious computational problems, as the required memory for storing the tree grows quadratically with t. This issue will be resolved in Section 10.4.2.

Returning to the sequence prediction problem, let $\mathbf{x}^{\langle 1:t-1 \rangle}$ be the sequence of observed symbols on round t, and let $\psi^{\langle t \rangle}$ be its corresponding function in $\mathcal{H}$ defined by Eq. (10.1). Denote $\tau_k^{\langle t \rangle}$ as the current context tree subject to the class k. By following the description in Section 10.2, the prediction problem can be formulated as

$$\widehat{x}^{\langle t \rangle} \coloneqq \arg\max_{k \in \Sigma} \underbrace{\left\langle \psi^{\langle t \rangle}, \tau_k^{\langle t \rangle} \right\rangle}_{f(\mathbf{x}^{\langle 1:t-1 \rangle})}. \tag{10.2}$$

Geometrically, one can consider $\mathcal{H}$ as a space partitioned by K hyperplanes, whose normal are

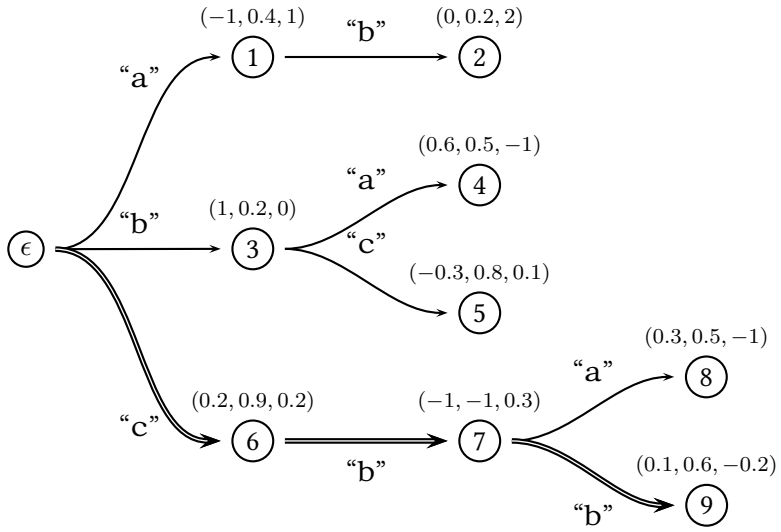


Figure 10.2.: An example of a multi-class context tree, where $K = 3$ and $V = \{\epsilon, \mathtt{a}, \mathtt{ba}, \mathtt{b}, \mathtt{ab}, \mathtt{cb}, \mathtt{c}, \mathtt{bc}, \mathtt{abc}, \mathtt{bbc}\}$. The label on each node represents the index. Notice how the index matches the element in V. The context associated with each node is indicated on the edges of the tree along the path from the root ϵ to that node. The vector associated with each node is provided above each node. This context tree can be parameterized as a 10×3 matrix, with the first column $(0, 0, 0)^\top$ corresponds to the empty sequence ϵ. Considering the context tree as a function, given an input sequence "$\mathtt{aabbc}$", the output from this context tree is $(0.1, 0.6, -0.2)$, whose path is plotted with double lines.

given by $\tau_1, \ldots, \tau_k$, respectively. The t^{th} symbol is then predicted by picking the hyperplane that gives the maximum (signed) distance to the vector $\psi^{\langle t \rangle}$.

10.4. Online Learning Algorithm

It can be seen in Section 10.3, the predictor is fully specified by a multi-class context tree τ, which can be represented by $\tau_1, \ldots, \tau_K$. Given a fixed V, τ_k can be parameterized by a vector $\mathbf{w}_k \in \mathbb{R}^{|V|}$. Denote $\mathbf{W} \coloneqq [\mathbf{w}_1, \ldots, \mathbf{w}_K]$, in which rows correspond to vectors associated with each node as depicted in Fig. 10.2. The size of $\mathbf{W}$ is thus $|V| \times K$. Note that a context tree is now fully specified by its weight vector $\mathbf{W}$ and structure V. That is, every $\{\mathbf{W}, V\}$ represents a unique τ and vice versa. Therefore, the problem of learning an accurate predictor can be reduced to the problem of determining $\mathbf{W}$ and V. Denote $\boldsymbol{\psi}^{\langle t \rangle} \in \mathbb{R}^{|V|}$ a vector corresponding to the sequence $\mathbf{x}^{\langle 1:t-1 \rangle}$. To construct $\boldsymbol{\psi}^{\langle t \rangle}$ I simply follow Eq. (10.1) and only assign values to the sequences in $\mathbf{x}^{\langle 1:t-1 \rangle} \cap V$. Elements of $\boldsymbol{\psi}^{\langle t \rangle}$ are indexed in the same order as $\mathbf{w}_k^{\langle t \rangle}$. Thus, the score vector $\mathbf{y}$ described in Section 10.2 amounts to $(\mathbf{W}^{\langle t \rangle})^\top \boldsymbol{\psi}^{\langle t \rangle}$.

This section describes the proposed online learning algorithm in four parts. I first describe the method to learn $\mathbf{W}$, subsequently, I present an approach for constructing V in a memory-efficient way. Extension for incorporating side information is described towards the end. Finally, several implementation issues are highlighted.

10.4.1. Learning Weight Vectors

I first show how the update of $\mathbf{W}$ can be performed in rounds. My method is closely related to the family of confidence-weighted linear classifiers [53, 37, 38, 163]. Following the idea of previous work, I maintain a Gaussian distribution for every column of $\mathbf{W}$ with a mean vector $\boldsymbol{\mu}_k \in \mathbb{R}^{|V|}$ and a diagonal covariance matrix $\boldsymbol{\Lambda}_k \in \mathbb{R}^{|V|\times|V|}$, i.e. $\mathbf{w}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$. Notice that by restricting $\boldsymbol{\Lambda}_k$ to a diagonal matrix, the weights become independent [1]. This is not true in real-world, yet it is necessary due to the large value of $|V|$. For the sake of efficiency, the predicted symbol is simply approximated by $\arg\max_{k\in\Sigma} \boldsymbol{\mu}_k \cdot \boldsymbol{\psi}^{\langle t\rangle}$ instead of using weight vectors sampled from $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$. In other words, the information captured by $\boldsymbol{\Lambda}_k$ does not influence the decision. This is analogous to Bayes point machines [83].

On each round, I update the model by minimizing the Kullback-Leibler divergence between new distribution and the old one while ensuring that the probability of correct prediction on t^{th} symbol is not smaller than the confidence hyperparameter $\eta \in [0, 1]$. After revealing the true symbol $r \coloneqq x^{\langle t\rangle}$, I need to update $(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$ to the solution of the following optimization problem

$$\left(\boldsymbol{\mu}_k^{\langle t+1\rangle}, \boldsymbol{\Lambda}_k^{\langle t+1\rangle}\right) = \arg\min_{\boldsymbol{\mu},\boldsymbol{\Lambda}} \quad \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \,\|\, \mathcal{N}\left(\boldsymbol{\mu}_k^{\langle t\rangle}, \boldsymbol{\Lambda}_k^{\langle t\rangle}\right)\right) \tag{10.3}$$

$$\text{s.t.} \qquad \Pr_{\mathbf{w}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Lambda})}\left[\mathbf{w}_r \cdot \boldsymbol{\psi}^{\langle t\rangle} \geq \mathbf{w} \cdot \boldsymbol{\psi}^{\langle t\rangle}\right] \geq \eta. \tag{10.4}$$

Notice that the optimization problem in Eq. (10.3) needs to be solved $K - 1$ times for every $k \in \Sigma \setminus r$ on each round, which can be computationally expensive. I therefore provide a simplified algorithm, where only two updates is required on each round. The intuition was to ensure that the true symbol is more likely to be predicted than the symbol that is its closest competitor. Specifically, let s be the highest ranked wrong symbol on round t. That is,

$$s \coloneqq \arg\max_{k\in\Sigma\setminus r} \boldsymbol{\mu}_k^{\langle t\rangle} \cdot \boldsymbol{\psi}^{\langle t\rangle}. \tag{10.5}$$

In each round only $(\boldsymbol{\mu}_r, \boldsymbol{\Lambda}_r)$ and $(\boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s)$ are updated as follows

$$\left(\boldsymbol{\mu}_r^{\langle t+1\rangle}, \boldsymbol{\Lambda}_r^{\langle t+1\rangle}\right) = \arg\min_{\boldsymbol{\mu},\boldsymbol{\Lambda}} \quad \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \,\|\, \mathcal{N}\left(\boldsymbol{\mu}_r^{\langle t\rangle}, \boldsymbol{\Lambda}_r^{\langle t\rangle}\right)\right)$$

$$\text{s.t.} \qquad \Pr_{\mathbf{w}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Lambda})}\left[\mathbf{w} \cdot \boldsymbol{\psi}^{\langle t\rangle} \geq \mathbf{w}_s \cdot \boldsymbol{\psi}^{\langle t\rangle}\right] \geq \eta. \tag{10.6}$$

$$\left(\boldsymbol{\mu}_s^{\langle t+1\rangle}, \boldsymbol{\Lambda}_s^{\langle t+1\rangle}\right) = \arg\min_{\boldsymbol{\mu},\boldsymbol{\Lambda}} \quad \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \,\|\, \mathcal{N}\left(\boldsymbol{\mu}_s^{\langle t\rangle}, \boldsymbol{\Lambda}_s^{\langle t\rangle}\right)\right)$$

$$\text{s.t.} \qquad \Pr_{\mathbf{w}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Lambda})}\left[\mathbf{w}_r \cdot \boldsymbol{\psi}^{\langle t\rangle} \geq \mathbf{w} \cdot \boldsymbol{\psi}^{\langle t\rangle}\right] \geq \eta. \tag{10.7}$$

Notice how the constraint of Eq. (10.6) and Eq. (10.7) differs from each other. I follow the

[1] One may consider $\mathbf{W}$ as a random variable from a matrix normal distribution, i.e. $\mathbf{W} \sim \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V})$, where $\mathbf{U}$ and $\mathbf{V}$ represents the correlation among-row and among-column, respectively. However, under the assumption of independent weights and independent symbols, $\mathbf{U}$ and $\mathbf{V}$ are simply diagonal matrices. This results an equivalent formulation to my results.

derivation in [37] and obtain the closed-form update as

$$\boldsymbol{\mu}_r^{\langle t+1\rangle} = \boldsymbol{\mu}_r^{\langle t\rangle} + \alpha \boldsymbol{\Lambda}_r^{\langle t\rangle} \boldsymbol{\psi}^{\langle t\rangle} \tag{10.8}$$

$$\boldsymbol{\mu}_s^{\langle t+1\rangle} = \boldsymbol{\mu}_s^{\langle t\rangle} - \alpha \boldsymbol{\Lambda}_s^{\langle t\rangle} \boldsymbol{\psi}^{\langle t\rangle} \tag{10.9}$$

$$\boldsymbol{\Lambda}_r^{\langle t+1\rangle} = \left(\left(\boldsymbol{\Lambda}_r^{\langle t\rangle} \right)^{-1} + 2\alpha\phi \operatorname{diag}^2 \left(\boldsymbol{\psi}^{\langle t\rangle} \right) \right)^{-1} \tag{10.10}$$

$$\boldsymbol{\Lambda}_s^{\langle t+1\rangle} = \left(\left(\boldsymbol{\Lambda}_s^{\langle t\rangle} \right)^{-1} + 2\alpha\phi \operatorname{diag}^2 \left(\boldsymbol{\psi}^{\langle t\rangle} \right) \right)^{-1}, \tag{10.11}$$

where $\operatorname{diag}^2 \left(\boldsymbol{\psi}^{\langle t\rangle} \right)$ is a diagonal matrix made from the squares of the elements of $\boldsymbol{\psi}^{\langle t\rangle}$ on the diagonal. The inverse of diagonal matrix can be computed element-wise. The coefficient α is calculated as follows

$$\alpha = \frac{-(1 + 2\phi m) + \sqrt{(1 + 2\phi m)^2 - 8\phi(m - \phi v)}}{4\phi v},$$

where

$$m = \left(\boldsymbol{\mu}_r^{\langle t\rangle} - \boldsymbol{\mu}_s^{\langle t\rangle} \right) \cdot \boldsymbol{\psi}^{\langle t\rangle} \tag{10.12}$$

$$v = \left(\boldsymbol{\mu}_r^{\langle t\rangle} \right)^\top \boldsymbol{\Lambda}_r \boldsymbol{\mu}_r^{\langle t\rangle} - \left(\boldsymbol{\mu}_s^{\langle t\rangle} \right)^\top \boldsymbol{\Lambda}_s \boldsymbol{\mu}_s^{\langle t\rangle} \tag{10.13}$$

$$\phi = \Phi^{-1}(\eta), \tag{10.14}$$

and $\Phi^{-1}(\cdot)$ is the inverse of the normal cumulative distribution function.

For initialization I set $\boldsymbol{\mu}_k^{\langle 1\rangle} := \mathbf{0}$ and $\boldsymbol{\Lambda}_k^{\langle 1\rangle} := \mathbf{I}$ for all k, where $\mathbf{I}$ is the identity matrix. It is noticed from Eq. (10.8) and Eq. (10.9) that during online learning the mean weight vector is updated in a similar fashion as in the Perceptron. The confidence of all observed suffixes is increased by shrinking the corresponding value on the diagonal of $\boldsymbol{\Lambda}_k$ (see Eq. (10.10) and Eq. (10.11)), which leads to the update of weight vector in the next round more focusing on low confidence features.

10.4.2. Memory-Efficient Update of Suffix Set

Having described the method for learning the weight vectors of the context tree, I now focus on determining its structure, i.e. V. Instead of adding all suffixes of the context to V on each round, I introduce three strategies for constructing V in a memory-efficient way.

First of all, I only update V if the probability constraint

$$\Pr_{\substack{\mathbf{w}_r \sim \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\Lambda}_r) \\ \mathbf{w}_s \sim \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s)}} \left[\mathbf{w}_r \cdot \boldsymbol{\psi}^{\langle t\rangle} \geq \mathbf{w}_s \cdot \boldsymbol{\psi}^{\langle t\rangle} \right] \geq \eta \tag{10.15}$$

is violated. Note that Eq. (10.15) can be rewritten as

$$(\boldsymbol{\mu}_r - \boldsymbol{\mu}_s) \cdot \boldsymbol{\psi}^{\langle t\rangle} \geq \phi \sqrt{\left(\boldsymbol{\psi}^{\langle t\rangle} \right)^\top (\boldsymbol{\Lambda}_r + \boldsymbol{\Lambda}_s) \boldsymbol{\psi}^{\langle t\rangle}},$$

where $\phi = \Phi^{-1}(\eta)$. Further, I introduce a loss function as

$$\ell_\phi\left(\{(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)\}_{k=1}^K; (\mathbf{x}^{\langle 1:t-1\rangle}, x^{\langle t\rangle})\right) := \max\left(0, \phi\sqrt{\left(\boldsymbol{\psi}^{\langle t\rangle}\right)^\top (\boldsymbol{\Lambda}_r + \boldsymbol{\Lambda}_s)\boldsymbol{\psi}^{\langle t\rangle}} - (\boldsymbol{\mu}_r - \boldsymbol{\mu}_s)\cdot\boldsymbol{\psi}^{\langle t\rangle}\right). \quad (10.16)$$

It is easy to verify that satisfying the probability constraint Eq. (10.15) is equivalent to satisfying $\ell_\phi = 0$. In this case, I simply set $V^{\langle t+1\rangle}$ to be equal to $V^{\langle t\rangle}$. Otherwise I add all sequences in $\mathrm{suf}(\mathbf{x}^{\langle 1:t-1\rangle})$ to $V^{\langle t\rangle}$. Note that ρ and ϕ can be tuned as a trade-off between the passiveness and aggressiveness of the update.

Second, when a sequence is extremely long, adding all suffixes of a long context can impose serious memory growth problem. Hence, it is not a practical solution. To limit the maximum depth of the context tree, I prune the context $\mathbf{x}^{\langle 1:t-1\rangle}$ to a certain length $\kappa^{\langle t\rangle}$ before adding its suffixes to V, where $\kappa^{\langle t\rangle}$ is given by

$$\kappa^{\langle t\rangle} = \min\left(\left\lfloor \frac{1}{\rho}\log \ell_{\mathbb{1}}(t)\right\rfloor, t-1\right),$$

with $\ell_{\mathbb{1}}(t)$ denoting the number of prediction mistakes made by the algorithm so far. The intuition behind is to limit the depth of the context tree by the number of prediction mistakes, which is inspired by [51]. As a consequence, one can straightforwardly translate the mistake bound of confidence weighted classifier (Theorem 4 in [37]) into a bound on the growth-rate of the resulting context tree [51, 90].

Finally, I limit the size of V by removing the elements giving smallest $\sum_k \mu_{k,i}^2$ when $|V|$ exceeds the maximum allowed size Q, where $\mu_{k,i}$ is the i^{th} element of $\boldsymbol{\mu}_k$ and $i \in [1, Q]$. This criterion has been shown effective in recursive feature elimination [79] and has a good theoretical support [29, 23]. Alternatively, one can also use the $\sum_k 1/\lambda_{k,i}$ or $\sum_k |\mu_{k,i}|/\lambda_{k,i}$ as the criterion, where $\lambda_{k,v}$ is the v^{th} element on the diagonal of $\boldsymbol{\Lambda}_k$. By employing the above three strategies the context tree grows at a much slower pace and the algorithm can utilize memory more conservatively. Finally, the pseudo-code of my algorithm is summarized in Fig. 10.3, which is called EOSP in the sequel for short.

10.4.3. Incorporation of Side Information

Thus far I augment only context information from the sequence. As I highlighted in the beginning of this chapter, side information of system calls can support the prediction of the next symbol. Apart from that, in language modeling grammars (e.g. part-of-speech tags), topics, styles are helpful to predict the next word [75, 164]. Comparing to the n-gram models and Bayesian nonparametrics models [154, 170], a key advantage of the proposed approach is its simplicity of leveraging side information. Specifically, if side information on round t can be given in the form of a vector $\mathbf{b}^{\langle t\rangle} \in \mathbb{R}^B$, one can directly incorporate it into the prediction via a linear combination as follows

$$\widehat{x}^{\langle t\rangle} := \arg\max_{k\in\Sigma} \boldsymbol{\mu}_k^{\langle t\rangle}\cdot\boldsymbol{\psi}^{\langle t\rangle} + \boldsymbol{\gamma}_k^{\langle t\rangle}\cdot\mathbf{b}^{\langle t\rangle}.$$

This corresponds to replacing $\boldsymbol{\psi}^{\langle t\rangle}$ in Fig. 10.3 as a $(Q+B)$-dimensional vector $[\boldsymbol{\psi}^{\langle t\rangle}, \mathbf{b}^{\langle t\rangle}]$. The dimension of the mean vector and confidence matrix associated with each symbol is extended

Input : Damping factor: $\rho > 0$; confidence parameter: $\eta \in [0, 1]$; maximum allowed size of V: $Q > 0$.
Output : Mean vectors and confidences matrices: $\{(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)\}_{k=1}^{K}$; set: V.
Initialize: $\forall k \in \Sigma$, $(\boldsymbol{\mu}_k^{\langle 1 \rangle}, \boldsymbol{\Lambda}_k^{\langle 1 \rangle}) = (\mathbf{0}, \mathbf{I})$, $\phi = \Phi^{-1}(\eta)$, $V^{\langle 1 \rangle} = \{\epsilon\}$;

for $t = 1, 2, \ldots$ **do**
  Construct $\boldsymbol{\psi}^{\langle t \rangle}$ from $\mathbf{x}^{\langle 1:t-1 \rangle}$; /* Eq. (10.1) */
  Rank all symbols by $\boldsymbol{\mu}_k^{\langle t \rangle} \cdot \boldsymbol{\psi}^{\langle t \rangle}$;
  Receive the true symbol r;
  Compute s; /* Eq. (10.5) */
  Suffer loss ℓ_ϕ; /* Eq. (10.16) */
  if $\ell_\phi > 0$ **then**
    while $|V^{\langle t \rangle}| > Q - \kappa^{\langle t \rangle}$ **do**
      $i = \arg \min_{j=1,\ldots,Q} \sum_{k \in \Sigma} \mu_{k,j}^2$;
      $\forall k \in \Sigma$, $\mu_{k,i} = 0$;
      Remove the i^{th} sequence from $V^{\langle t \rangle}$;
    $V^{\langle t+1 \rangle} = V^{\langle t \rangle} \cup \{\mathbf{x}^{\langle t-i:t-1 \rangle} \mid 1 \leq i \leq \kappa^{\langle t \rangle}\}$;
    Set $(\boldsymbol{\mu}_r^{\langle t+1 \rangle}, \boldsymbol{\Lambda}_r^{\langle t+1 \rangle})$ and $(\boldsymbol{\mu}_s^{\langle t+1 \rangle}, \boldsymbol{\Lambda}_s^{\langle t+1 \rangle})$; /* Eqs. (10.8) to (10.11) */

Figure 10.3.: Efficient online sequence prediction (EOSP).

accordingly. Note that an ineffective representation of side information can adversely affect the prediction performance as well, hence there has to be some mechanism for selecting features that really contribute to prediction. In the proposed algorithm, this can be done by constantly setting $\mu_{k,i+b}$ to zero if $\sum_k |\mu_{k,i+b}|/\lambda_{k,i+b}$ is too small. In addition, one can also initialize $\boldsymbol{\Lambda}_k \coloneqq \begin{pmatrix} \mathbf{I}_{|V| \times |V|} & \\ & \gamma \mathbf{I}_{B \times B} \end{pmatrix}$ with $0 < \gamma < 1$ to balance the learning rate of the weights on the context and side information. Specifically, when $\gamma = 1$ the side information shares the same learning rate with context information; when $\gamma = 0$ the side information does not contribute the learning and prediction at all.

The side information used in this work is summarized in Table 10.2. The idea of using these attributes is mainly based on experiences and observations. For instance, I observed that system calls with similar functionality tend to occur together, which could be due to some sub-task of the process. Thus, if a particular group of system calls is frequently observed in the recent context, then the next system call is very likely from the same group. In present work, system calls are grouped manually by their documentation, which is partially based on [142]. It is also possible to automatically group system calls by using topic models [174]. Another observation is that a block of system calls repeats themselves especially when some of them return an error. This was probably attributed to the exception handling (e.g. restart mechanism) of a process. Thus, a simple statistic of the error codes is maintained and considered as one of the evidences for predicting the next system call. In practice, the side information listed in Table 10.2 can be easily extracted from the context with negligible computational cost.

Table 10.2.: Side information used in our algorithm for system call prediction.

Feature set	Size	Description
File descriptor	2	The number of opened files and the number of closed files, respectively.
File type	9	Each element represents the number of opened files of a particular type, such as RDONLY, WRONLY, APPEND, etc.
Functional group	9	Each element represents the number of occurrences of system calls associated with a group given a context. The groups were built in advance by categorizing similar system calls together, resulting 9 groups in total. For instance, the "file" group includes `creat`, `open`, `close`, `read`, etc. The "process" group includes `fork`, `wait`, `exec`, etc. The "signal" group includes `signal`, `kill`, `alarm`, etc.
Access location	12	Each element represents the number of accesses to a particular directory, such as `/usr/bin`, `/usr/lib`, `/usr/tmp`, etc.
Error code	124	Each element represents the number of caught errors of each code, such as ENOENT, EAGAIN, EBGDF, etc.
POSIX signal	28	Each element represents the number of sent signals of each type, such as `SIGSEGV`, `SIGABRT`, `SIGBUS` etc.
String character	256	Each element represents the frequency value of a string character. A `char` is considered as an 8-bit value, resulting 256 possible characters. We only count characters in the string that is not file path.

10.4.4. Efficient Implementation

It can be observed from Eqs. (10.8) to (10.11) that most of the entries of $\boldsymbol{\psi}$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Lambda}_k$ are zero, which implies a possibility to improve the efficiency by storing them in a compact way. In the implementation, I store $\boldsymbol{\mu}_k$, $\boldsymbol{\Lambda}_k$ and $\boldsymbol{\psi}$ in sparse vectors. The algorithms of addition and dot product for sparse vectors can be found in [46]. Moreover, as the updates of $(\boldsymbol{\mu}_r, \boldsymbol{\Lambda}_r)$ and $(\boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s)$ are independent to each other, line 13 Fig. 10.3 can be implemented in a parallel manner. Furthermore, the operations on V can be implemented efficiently using a data structure called *suffix trie*. Finally, removing one element at a time (line 8 to 11 Fig. 10.3) is time consuming and in practice I remove as much as half of Q when $|V^{\langle t \rangle}| > Q - \kappa^{\langle t \rangle}$.

10.5. Experiments

Two sets of experiments were carried out to validate my algorithm. First, I compared the accuracy and efficiency of EOSP with state-of-the-art sequence prediction methods. Second, I investigated several factors that affect the performance of EOSP in order to gain more insights of it.

The experiments were conducted on three groups of data. The first set of data is from BSM

(Basic Security Module) data portion of 1998 DARPA intrusion detection evaluation data set created by MIT Lincoln Labs[2]. I used a subset of training data, which contained four-hour BSM audit data of all processes running on a Solaris machine. Each system call was recorded with its corresponding arguments and return value. The second group of data was obtained from University of New Mexico [166], in which system call traces of several process were generated in either "synthetic" or "live" manner[3]. The experiment was conducted on their "live" normal data, where traces of programs were collected during normal usage of real users. Unlike DARPA data set, a trace in UNM data set is just a list of system call names; no arguments and return values are available. Therefore, for UNM data set only the functional group in Table 10.2 was available as side information. The third data set was collected by me. By using `strace` and a prepared script, I collected system call sequences with their corresponding arguments and return values from all executable programs on an Ubuntu system. The program options were chosen solely for the purpose of exercising the program, and not to meet any real user's requests. From these three sources I selected a total of 8 data sets, and their characteristics are summarized in Table 10.3.

Table 10.3.: Characteristics of data sets used in the experiment.

Data set	# calls	# seq.	Min. len.	Max. len.	Avg. len.
darpa	243	200	2	3,074	57
lpr1	182	2,766	82	59,565	1,080
lpr2	182	1,232	74	39,306	449
sendmail1	190	8,000	8	173,664	669
sendmail2	190	8,000	8	149,616	648
stide1	164	8,000	225	146,695	1,055
stide2	164	8,000	108	174,401	1,255
ubuntu	458	1,218	2	53,247	952

Four sequence prediction methods were employed in the experiment. They were interpolated Kneser-Ney (IKN) [27], online prediction suffix tree (PST) [51], sequence memoizer (SM) [171], and learning experts (LEX) [56]. I restricted the maximum length of context to 50 for all algorithms except for SM, which was designed for modeling context with infinite length. Specifically, I used a 50-gram IKN in the experiment. For LEX the number of experts was set to one and $d := 50$, resulting an individual sequence predictor trained with the log loss. The maximum depth of the context tree for PST was set to 50. These four methods were compared with the proposed EOSP, and the algorithm with side information denoting as EOSP_s . The confidence parameter η was 0.8; the damping factor ρ was 0.1; the maximum length of the context was 50 and the maximum size of V was 20,000.

10.5.1. Comparison of Predictive Performance

The comparison of predictive performance between different methods is summarized in Table 10.4, where the online error rate and perplexity were used as evaluation metrics. The online error rate of an algorithm on a given input sequence is defined to be the number of prediction mistakes

[2] `http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/`

[3] `http://www.cs.unm.edu/~immsec/systemcalls`

the algorithm makes on that sequence normalized by the length of the sequence. The perplexity reflects an algorithm's performance when taking its probabilistic output into account. For EOSP I just normalized the score vector to obtain the prediction probability $\Pr[\widehat{x}^{\langle t \rangle} \mid \mathbf{x}^{1:t-1}]$. The reported results were averaged over all sequences in each data set respectively.

It is evident from the results that, EOSP and EOSP_s gave a considerably better prediction than other baseline algorithms. In particular, EOSP_s achieved the best performance on the majority data sets (seven out of eight in terms of perplexity), which indicates the effectiveness of incorporating side information into the model. On five out of six UNM data sets, I observed an improvement by just incorporating the functional group information. On `darpa` and `ubuntu` data sets where side information are fully available, a striking improvement of EOSP_s over EOSP was observed. In general, I found SM is a strong competitor in terms of online error rate. However, EOSP and EOSP_s still outperformed SM with appreciable lower perplexity on all data sets. This suggests a potentially valuable property for my method, e.g. for combining it with other probabilistic model in a big system. Moreover, SM is much slower than EOSP on long sequence, as the reader will see in the next experiment.

Table 10.4.: Experimental results on different data sets. Smaller value indicates better performance.

(a) Online error rate (%) of different algorithms.

Data set	EOSP	EOSP_s	IKN	PST	SM	LEX
`darpa`	50.11	**48.17**	52.14	49.25	49.75	51.11
`lpr1`	41.63	41.53	41.09	46.24	**40.88**	42.27
`lpr2`	47.44	**47.03**	47.61	48.52	47.24	51.15
`sendmail1`	33.47	34.26	35.62	33.65	**33.06**	36.81
`sendmail2`	33.11	33.91	33.52	34.17	**32.19**	38.96
`stide1`	8.34	**8.29**	8.54	8.59	8.41	9.06
`stide2`	**7.75**	7.75	8.09	7.95	7.78	8.51
`ubuntu`	40.90	**36.13**	38.90	39.23	75.26	52.72

(b) Online perplexity of different algorithms.

Data set	EOSP	EOSP_s	IKN	PST	SM	LEX
`darpa`	48.98	**40.23**	78.34	98.97	63.07	82.36
`lpr1`	9.82	**9.23**	16.14	17.05	14.75	11.71
`lpr2`	12.94	**11.08**	21.43	16.34	19.94	22.19
`sendmail1`	8.31	**8.17**	11.48	30.34	9.23	11.90
`sendmail2`	8.33	**7.96**	11.50	30.38	9.17	12.46
`stide1`	1.42	**1.41**	2.08	3.42	1.92	4.06
`stide2`	**1.39**	1.41	1.98	3.23	1.67	4.51
`ubuntu`	33.13	**31.65**	42.81	35.35	68.25	35.62

10.5.2. Comparison of Efficiency

The comparison of computation speed and memory consumption for all algorithms is shown in Fig. 10.4 and Fig. 10.5, respectively. I concatenated all traces in `sendmail` to obtain a long sequence, and tested different methods on this sequence with increasing length. The setup of each

method was same as in the last experiment. For the sake of fair comparison, all algorithms were implemented in C/C++. I only plotted the curve for EOSP as EOSP_s took almost same amount of time and memory in the experiment. As can be seen in Fig. 10.4, EOSP showed a substantial reduction of time comparing to other baseline algorithms. Moreover, the time cost of EOSP only increased at a very low pace with respect to the length of the sequence. As I expected, LEX and SM were extremely slow especially on long sequences, since on each round they require gradient descent and Gibbs sampling, respectively. On contrary, in EOSP one only need to compute dot products of sparse vectors on each round, which can be done efficiently. On the other hand, though the memory consumption of EOSP is higher than other baselines at the beginning, it remained almost constant with increasing length of the sequence. Methods such as IKN and LEX, however, consume more and more memory as the sequence becomes longer. This demonstrates the effectiveness of the update strategies described in Section 10.4.2.

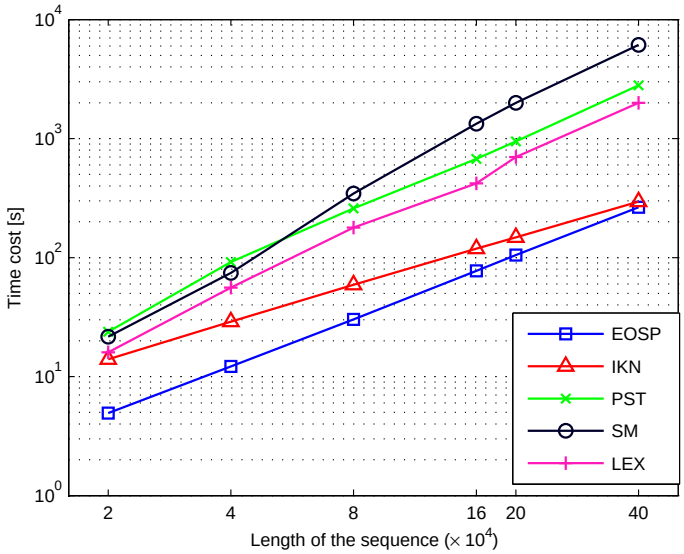


Figure 10.4.: Time cost in second (averaged over 10 runs) of different algorithms. Both axes are in logarithmic scale.

10.5.3. Exploration of Model Parameters

In order for EOSP to be a practical tool in real-world applications, it is necessary to make decisions about the details of its specification. The exploration focused on three parameters that mainly govern the performance of EOSP. Namely, the confidence parameter η, the maximum length of the context, and the maximum size of V. I focused only on EOSP and ignored all side information in this set of experiments.

The performance of EOSP with respect to different settings of confidence parameter η is summarized in Table 10.5. I fixed the maximum length of the context to 50 and maximum size of V to $20,000$. On the majority of data sets, the online error rate hit the bottom when η is around 0.9. However, the lowest perplexity was often observed when $\eta := 0.8$; the perplexity slightly increased after that. In general, bigger value of η allows the algorithm to perform a more confident update on each round, which generally leads to higher predictive accuracy when the data is

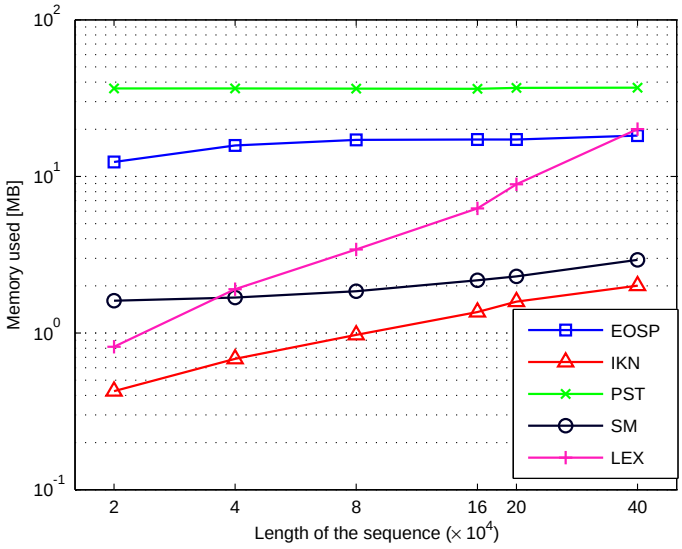


Figure 10.5.: Memory consumption (averaged over 10 runs) of different algorithms. Both axes are in logarithmic scale.

noise-less.

Table 10.5.: Performance of EOSP w.r.t. different settings of confidence parameter η. Smaller value indicates better performance.

(a) Online error rate (%) of EOSP

Data set	0.6	0.7	0.8	0.9
darpa	50.41	50.52	**50.11**	50.39
lpr1	41.61	41.55	41.63	**41.43**
lpr2	47.21	47.02	47.44	**46.56**
sendmail1	34.25	34.12	33.47	**33.12**
sendmail2	33.32	**33.10**	33.11	33.74
stide1	8.21	8.25	8.34	**8.13**
stide2	7.79	7.79	7.75	**7.64**
ubuntu	44.77	44.76	**40.90**	44.60

(b) Perplexity of EOSP

Data set	0.6	0.7	0.8	0.9
darpa	**48.82**	49.05	48.98	49.03
lpr1	9.74	9.73	9.82	**9.72**
lpr2	12.62	12.57	12.94	**12.33**
sendmail1	8.67	8.64	**8.31**	8.44
sendmail2	8.69	8.67	**8.33**	8.47
stide1	1.43	1.44	**1.42**	1.42
stide2	1.40	1.41	**1.39**	1.39
ubuntu	32.93	33.02	32.93	**32.29**

Table 10.6 summarizes the results of EOSP with respect to different maximum length of the context, where $\eta := 0.8$ and $Q := 20,000$. Although one may expect an improvement of the predictive accuracy by allowing the algorithm to look back long distant context, I found that the optimal length of the context varies with data sets. On `darpa`, `lpr1` and `stide1`, for example, the context length of 40 was sufficient for a good prediction; increasing this length did not improve the prediction. On `ubuntu`, the online error rate decreased with increasing context length up to 100. In general, I found that the prediction of EOSP is not adversely affected by the overlength context, though its efficiency can be degraded due to more memory consumption.

Table 10.6.: Performance of EOSP w.r.t. different maximum length of context.

(a) Online error rate (%) of EOSP

Data set	20	40	60	80	100
darpa	50.21	**50.10**	50.10	50.10	50.10
lpr1	41.66	**41.63**	41.63	41.65	41.65
lpr2	**47.45**	47.45	47.45	47.45	47.45
sendmail1	35.70	33.48	**33.47**	33.47	33.47
sendmail2	33.67	33.60	**33.11**	33.11	33.11
stide1	8.45	**8.27**	8.27	8.27	8.27
stide2	8.02	**7.75**	7.75	7.75	7.75
ubuntu	41.84	41.23	40.90	40.75	**40.69**

(b) Perplexity of EOSP

Data set	20	40	60	80	100
darpa	**48.97**	48.97	48.98	48.98	48.98
lpr1	**9.78**	9.82	9.82	9.82	9.82
lpr2	12.94	**12.93**	12.94	12.94	12.94
sendmail1	8.36	**8.31**	8.31	8.31	8.31
sendmail2	8.38	**8.32**	8.32	8.33	8.33
stide1	1.43	**1.42**	1.42	1.42	1.42
stide2	1.40	**1.39**	1.40	1.40	1.40
ubuntu	33.15	33.13	33.12	33.11	**33.10**

Finally, to study the performance with respect to different sizes of V, I fixed $\eta := 0.8$ and the maximum length of context to 50. Results are summarized in Table 10.7. It was found that on the majority of data sets predictive performance can be improved by allowing V to contain more suffixes, which can be expected. However, on `darpa` data set having at most $4,000$ suffixes in V was sufficient for obtaining a good result; increasing the upper bound of $|V|$ did not improve the performance but raised the memory consumption. This is probably due to that most sequences in `darpa` are short (with average length of 57) and hence there are not many combinations for frequently occurred subsequences. In general, if the patterns in a sequence are simple (especially with some periodicity), then one can set a small size for V.

Table 10.7.: Performance of EOSP w.r.t. different maximum size ($\times 10^3$) of V.

(a) Online error rate (%) of EOSP.

Data set	1	2	4	8	16
darpa	50.39	50.21	**50.13**	50.13	50.13
lpr1	42.45	42.18	41.83	41.68	**41.65**
lpr2	48.21	48.00	47.56	**47.45**	47.45
sendmail1	38.01	36.57	35.92	35.24	**34.63**
sendmail2	34.98	33.55	32.88	32.60	**32.43**
stide1	9.23	8.99	8.62	8.32	**8.27**
stide2	8.74	8.57	8.28	7.90	**7.85**
ubuntu	42.73	42.20	41.97	41.51	**41.21**

(b) Perplexity of EOSP.

Data set	1	2	4	8	16
darpa	**48.97**	48.98	48.98	48.98	48.98
lpr1	10.37	10.15	9.93	**9.82**	9.82
lpr2	13.54	13.44	13.01	**12.94**	12.94
sendmail1	9.46	8.84	8.47	**8.31**	8.31
sendmail2	9.49	8.88	8.49	**8.33**	8.33
stide1	1.50	1.47	1.45	**1.43**	1.43
stide2	1.46	1.45	1.42	**1.40**	1.40
ubuntu	33.53	33.30	33.19	33.15	**33.11**

10.6. Conclusion

Motivated by the problem of system call prediction, this chapter has proposed a novel method for predicting the next symbol in a sequence. The sequence prediction problem can be seen as a discrete counterpart of online regression Chapter 9. Unlike previous methods in this field, my algorithm does not rely on a fixed length context during learning and can be easily incorporated with side information. The algorithm maintains a set of distributions over parameters. On each round, the distributions are updated to satisfy a probabilistic constraint. The update can be computed in closed-form and implemented using sparse vectors. Moreover, I proposed several strategies to reduce the memory consumption, allowing a good scalability on long sequences. Experiments on real-world data sets showed that my method outperforms state-of-the-art online sequence prediction methods in both accuracy and efficiency, and incorporation of side information does significantly improve the predictive accuracy. An important question for future studies is to explore theoretical properties of the proposed algorithm, such as the convergence rate under different noise settings. In particular, it would be interesting to develop a robust algorithm for predicting sequence with adversarial noise.

The proposed method can serve as a backbone in a wide range of real-time applications, such as intrusion detection and power consumption modeling on mobile devices. Comparing to previous methods in this area, my algorithm allows one to incorporate the domain knowledge as side information to improve prediction. Besides, the proposed framework can be also adapted to perform other tasks, such as language modeling and structure prediction. In Chapter 11, the reader

will see how to extend this model to handle partially labeled data stream.

Chapter 11

Communication-Efficient Online Semi-Supervised Learning in Client-Server Settings

The challenge of handling large-scale data can be interpreted in different ways. In Chapter 9 and Chapter 10, I have concentrated on the time-cost and aim to speed-up the learning algorithms. Though the time-cost is an important aspect, it is not the only one. In this chapter, I describe a completely novel learning problem, where the goal is to reduce the communication-cost over the network. Solving this problem is a significant step towards learning large-scale data in the distributed settings.

Distributed data acquisition is at the heart of the big data explosion. Smartphones, surveillance videos, wearable sensors, and a variety of smart devices (Internet of Things) generate data at geographically distributed points, and the goal is to learn valuable insights from these massive data streams. This chapter considers such a setting where a set of distributed clients each generate an ongoing stream of data and a server seeks to learn a model of the data. We impose two practical limitations on the setting. First, because of the costs of having humans label large quantities of data, it is assumed that only a small fraction of the data are labeled. In particular, we focus on a setting where only the first, e.g., 2% of the training data are labeled. Second, because communication bandwidth is often expensive and battery-draining (e.g., a mobile device on a cellular network), I seek communication-efficient solutions such that each client is limited to sending to the server only a small fraction of the unlabeled data it generates, and limited in how much information it receives from the server.

As a motivating example, consider an intelligent traffic management system comprised of a set of surveillance cameras and a server. The server analyzes images from cameras and provides applications such as helmet violation, high-occupancy vehicle detection, and wrong-way vehicle alarms. To develop such a system, the model on the server needs to be configured by teaching it baseline images. Traditionally, it requires each camera to constantly upload images, and human annotators to manually label those uploads on the server. In practice, however, the network bandwidth is restricted and the labeling effort is limited. Therefore, a workable solution would be training an initial model with limited labels on the server, and selectively transmitting only the most informative images from each camera.

As another example, consider wearable devices (e.g., smartwatches) that measure sensory data, which is increasing dramatically both temporally and in fidelity. However, the device does not offer heavy computing power and may just serve as a front end for a remote system. To utilize the sensory data for intelligent tasks (e.g., recognizing human activities), the collected data on the device need to be transmitted wirelessly to a more powerful device (e.g., a smartphone or laptop). However, due to the bandwidth and battery limitations, it is unrealistic for the device to transmit every measurement. Often, the connection is only established at set intervals or manually by users, at which time only a selective subset of the measurements may be transmitted.

An elegant solution to these problems will face many challenges. First, the amount of data generated by clients can be huge, and even potentially unlimited. As a result, the vast majority of data on the server are unlabeled. Typically, it is not sufficient to train a model with a good generalization ability based merely on limited labeled data. Second, when the volume and velocity of data is high, it is very costly and impossible to store all data either on clients or the server. Thus, traditional approaches that first store data and then train on a static collection are not appropriate in this case. Third, transmitting massive data on the network is discouraged in practice, especially when the network bandwidth is restricted or the communication cost is expensive (e.g., on a cellular network). It may also be mis-classified as a *denial-of-service attack*, and dropped/blocked.

At first sight, this learning problem seems to share some characteristics of online, semi-supervised, and active learning, which have been extensively studied in the machine learning community. However, it should be noted that my setting differs from these traditional learning settings and may require evolutionary changes to existing algorithms. Unlike online learning problems where all training data are assumed to be labeled, there is only a limited amount of labeled data in my setting. It also differs from typical semi-supervised learning where all labeled and unlabeled data is available ahead of time. Moreover, it differs from standard active learning in that there is no oracle available for providing feedback. Although both settings involve selective sampling, their intentions are different: active learning aims to save labeling efforts, whereas I attempt to reduce the bandwidth consumption between the server and clients (while also keeping the labeling effort to only a small fraction of the data). By considering online, semi-supervised, and active learning jointly, my goal is to develop a modular framework for learning from a remote partially labeled data stream while reducing the bandwidth consumption.

In this chapter, I present a novel framework for solving this learning problem in an effective and communication-efficient manner (see Figure 11.1). On the server side, my solution combines two diverse learners working collaboratively, yet in distinct roles, on the partially labeled data stream. A compact, online graph-based semi-supervised learner is used to predict labels for the unlabeled data arriving from the clients. Specifically, I adapt the Harmonic Solution learner to online use via an incremental k-center clustering approach that maintains the graph structure solely on a set of k centroid nodes. Random samples are then repeatedly drawn from the model according to the confidence of its prediction, and used to train a second learner on the server, a linear classifier (specifically, a soft confidence-weighted classifier). The second learner updates its hypothesis based on these samples and their predicted labels. I show how these two learners can be combined in an optimization problem. On the client side, my solution prioritizes data based on an active-learning metric that favors instances that are more uncertain (i.e., close to the classifier's decision hyperplane) and yet far from each other (as measured by covariance). To reduce communication, the server sends the classifier's weight-vector to the client only periodically. At any point in time, the classifier can be used as a standalone model for predicting labels for new test data.

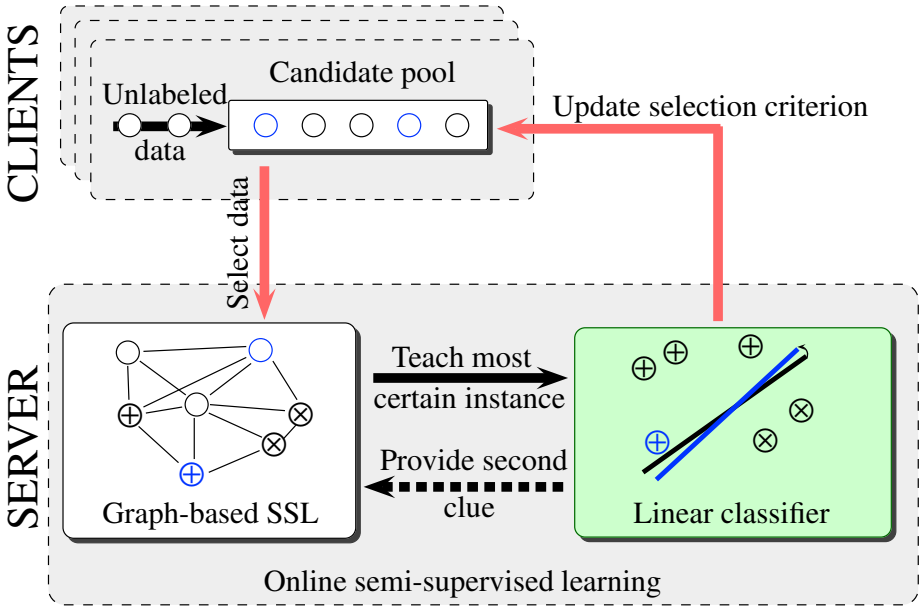


Figure 11.1.: An illustration of the proposed framework. The server contains two learners: a graph-based semi-supervised model and a linear classifier. They collaborate together to learn from a partially labeled data stream. At any point in time, the linear classifier can be used as a standalone component for predicting labels for new test data. The communication flow between each client and the server is represented by red arrows.

The main contributions of this chapter are:

- I introduce a novel learning setting motivated by many big data applications, and present a general framework that surmounts the challenges inherent in this setting. The proposed framework is modular in design, flexible, and can be practically incorporated into a variety of useful systems.
- I present a novel techniques at the clients and the server that are well-suited to providing high classification accuracy with reduced communication and labeling costs.
- The experiment results on real-world data sets show that this particular combination of techniques outperforms other approaches, and in particular, often outperforms (communication expensive) approaches that send *all* the data to the server.

11.1. Related Work

Online learning, semi-supervised learning and active learning are three different problem settings, which have been studied both separately and jointly. Perhaps the earliest exploration in combining semi-supervised learning with active learning is by McCallum et al. [111], where they combined an expectation-maximization algorithm with an active learning algorithm. Recently, many extensions of semi-supervised methods (e.g., S3VM [9], harmonic solution [180] and co-training [15]) to the active learning setting have been proposed (e.g., [115, 181, 165, 178]). In practice, active semi-supervised learning has a wide range of applications, from spoken language understanding [159]

to document clustering [84] to content-based image retrieval [86, 177, 179]. Unfortunately, these methods do not meet the requirements of my problem setting, in which data items arrive in an online fashion, not in batch.

Another line of research is combining online learning with semi-supervised learning, which is extremely useful for adaptive systems with partially labeled input. Most of the algorithms in this line rely on indirect forms of feedback, such as a model's own prediction and the structure of data, to incrementally improve itself. Grabner et al. [74] used a heuristic method to greedily label unlabeled examples in an object tracking application. Goldberg et al. [72] extended the online SVM [19] to the semi-supervised setting by adding a regularization term to the objective function of SVM. Valko et al. [160] extended the graph-based semi-supervised learning method [180] to the online setting, by computing the harmonic solution on an approximate similarity graph in an incremental fashion. In my setting, this family of methods can be adapted for the server's use. However, it does not reduce communication costs because no selection is performed prior to transmission to the server.

The intention of online active learning was to extend the traditional active learning from the *pool-based* setting to the *stream-based* setting [31]. Zhu et al. [182] introduced a minimal variance principle to guide instance selection from a data stream. Bifet et al. [11] presented a weighted ensemble classifier and cluster model to handle large data stream volumes. Chu et al. [30] designed optimal instrumental distributions for allowing unbiased sampling in data streams. However, such methods are not applicable to my setting, as they cannot learn from unlabeled instances.

Finally, the idea of integrating online learning, semi-supervised learning, and active learning into one framework can be traced back to Shen et al. [140]. They extended the self-organizing incremental neural network [69] with semi-supervised learning and active learning. On each round, the algorithm selects some "teacher" nodes from each cluster and uses them to label all unlabeled nodes in the corresponding cluster. Later, Goldberg et al. [73] provided a Bayesian model for this learning setting. The model maintains a posterior distribution of weights through particle filtering and sequential Monte Carlo techniques. Instances that are highly disagreed according to the current particles are queried for labeling.

Unlike these prior works [140, 73], which intended to reduce the labeling effort for adaptive systems, my goal is to reduce the communication and labeling costs in a distributed client-server system. Moreover, the following three obstacles limit the possibility of adapting previous methods to my problem setting. First, the prior methods are not applicable to the client-server model, where the concerns of client and server must be well-separated. Most previous methods are developed in a bottom-up manner, by gradually extending the availability of original supervised learning methods to give rise to more complex settings. Thus, they are not *modular* in design. For example, Shen et al. [140] used the self-organizing incremental neural network [69] as the "seed" model. Their active learning and semi-supervised learning extensions work exclusively with the seed model, making it difficult to isolate each component. In a distributed setting, it is important to elucidate each subsystem for addressing a separate concern, as they may be deployed in different physical locations with different configurations.

Second, prior methods are not communication-efficient. More precisely, there is no efficient way to transmit the selection criterion to the clients. For example, Zhu et al. [181] selected instances based on their estimated risk on a graph, which would require each client to maintain a graph locally. Similar difficulty can be found in Goldberg et al. [73], where the uncertainty score is computed based on a set of particles (parameterized by a set of vectors), thereby requiring each client to maintain a set of vectors. High communication costs are incurred in keeping a cient's set

up-to-date.

Third, prior methods are computationally demanding. For example, Goldberg et al. [73] used a sequential Monte Carlo technique to update the model, requiring a number of iterations for learning a new datum. In the distributed setting, algorithms on both client and server should be lightweight and avoid time-consuming computations. This is because clients usually have few resources other than essential input and output functions. The server, though, offers more resources, and must respond agilely so that the new selection criterion can be quickly generated and distributed without forcing clients to wait.

11.2. Notations

Denote by $\mathcal{X}$ an instance domain and by $\mathcal{Y}$ a set of labels. Let $\mathcal{H}$ be a hypothesis class, where each $h \in \mathcal{H}$ is a mapping from $\mathcal{X}$ to $\mathcal{Y}$. In this chapter, I concentrate on the confidence-rated binary classification problem, where $\mathcal{H}$ is the class of linear separators. In this case, $\mathcal{X}$ is a subset of the Euclidean space $\mathbb{R}^d$, $\mathcal{Y} = \{+1, -1\}$, and each hypothesis in $\mathcal{H}$ is a linear function parametrized by a weight vector $\mathbf{w} \in \mathbb{R}^d$. For each $\mathbf{x} \in \mathcal{X}$, define $h(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$. In practice, one can handle a bias term by adding a dummy feature to all $\mathbf{x}$ and set $d = d+1$. The reader can interpret $\mathrm{sign}(h(\mathbf{x}))$ as the actual binary label predicted by h, and $|h(\mathbf{x})|$ is a degree of confidence in this prediction. The quality of a prediction is measured by a loss function $\ell(h; (\mathbf{x}, y))$, which represents the penalty of predicting $\mathrm{sign}(h(\mathbf{x}))$ when the correct label is $y \in \mathcal{Y}$. Two common choices of loss function are *zero-one loss* and *hinge loss*.

For the sake of simplicity, I will present the techniques in this chapter assuming there is only a single client. The framework can be readily generalized to multiple clients, as discussed in Section 11.7. Denote the set of unlabeled instances on the client by $V = \{\mathbf{x}_t\}_{t=1}^{v}$, where each $\mathbf{x}_t \in \mathcal{X}$. The client selects instances for uploading. On the server side, only a small set of labeled data $L = \{(\mathbf{x}_t, y_t)\}_{t=1}^{l}$ is available at the beginning, followed with a set of unlabeled data $U = \{\mathbf{x}_t\}_{t=l+1}^{n}$ uploaded from the client. I assume the server receives incoming data one-by-one. The total number of instances received by the server is n, and in the setting $l \ll n$, and $n \ll v$. Starting from an initial hypothesis h_0, the server incrementally constructs a sequence of hypotheses $h_1, h_2 \ldots, h_n$ according to L and U. Ultimately, the goal of the server is to find a hypothesis that will exhibit high classification accuracy (e.g., under zero-one loss) on some unseen test set.

11.3. General Framework

I present a general framework for communication-efficient online semi-supervised learning in the client-server setting. The framework will be described in a way that the modules can be easily understood in isolation, and changes or extensions to functionality would be easily localized. Specifically, I start in this section with a big picture by describing the philosophy behind the system design, and a high-level overview of the framework. Later, Section 11.4 and Section 11.5 will drill down on the techniques on the server and client, respectively.

11.3.1. Design Philosophy

When it comes to a practical framework, several pressing concerns have to be kept in mind. First, it requires careful coordination and control of data being passed between the server and clients.

In particular, the server sends a criterion to guide the client to select instances. The client sends selected instances to the server, which may affect the selection criterion of next rounds. In both directions, the transmission must be efficient. I use a windowed pool-based method wherein each client maintains a small bounded-size buffer of its most recent data. When the buffer fills, a subset of the data is chosen for uploading to the server. After that, the buffer is emptied so that new data can be accommodated. The selection criterion is only updated every time the buffer is emptied. This enables a fine-grained control over the communication bandwidth by simply changing the buffer size and the size of the uploaded subset.

On the server side, the employed learning algorithm should be efficient enough to perform (near) real-time online learning, and be flexible enough to be a standalone module for predicting labels on a new set of test data. Moreover, the selection criterion should be represented in a way that the server can transmit it to the client with a low communication cost. Fortunately, existing state-of-art machine learning algorithms already have many lightweight and flexible aspects that can serve as a good start.

In the context of online semi-supervised learning, it is natural to train a model using labels obtained by the model's own predictions [179, 80]. However, this approach may suffer significantly from the accumulation of wrongly predicted labels over many rounds, resulting in an inaccurate hypothesis. For this reason, it is preferable to update a hypothesis conservatively, thereby alleviating the fluctuations in the performance of the hypothesis.

11.3.2. Proposed Framework

The framework (Figure 11.1) is designed based on the above considerations. It can be decomposed into several components that drive different functionalities. On the client side, I perform data triage by selecting instances from a candidate pool, where the selection criterion is controlled by the server. On the server side, an online semi-supervised learning algorithm is employed to handle unlabeled submissions. The key is to maintain two learners—a graph-based semi-supervised model and a linear classifier—and let them collaborate to exploit unlabeled data. Specifically, incoming instances are added to the training set of the first learner, which is represented by a graph. The nodes of the graph are instances, and the edges between nodes reflect the similarity between the corresponding instances. Then, the first learner predicts labels for all unlabeled instances in the graph, and randomly samples an instance according to the confidence of its predictions in order to teach the second learner. The second learner updates its hypothesis, and delivers a new selection criterion to the client. At any time, the second learner can be used as a standalone model for predicting new test data.

While different machine learning algorithms can be used as a part of this framework, some techniques lend themselves to my problem setting better than others. In this work, I employ the harmonic solution (HS) [180] as the first learner and the soft confidence-weighted classifier (SCW) [163] as the second leaner. My choice offers several advantages. First, SCW is simple, fast and enjoys state-of-the-art performance on classification. Second, SCW performs a conservative update especially with noisy labels. Third, SCW can be parameterized by a weight vector and a covariance matrix, allowing the server to deliver the selection criterion to the client with a low communication cost. In this work, I simply transmit the weight vector of SCW to the client. On the other hand, HS nicely complements SCW by providing feedback using the data manifold. It can leverage the similarities between instances, which is something that SCW overlooks, to determine labels of unlabeled data. By peering these two models together, my method enjoy the best of

both worlds, efficient learning and simple parameterization due to SCW, and the ability to exploit manifold information disclosed by unlabeled examples due to HS. Moreover, SCW and HS can be incorporated into a single optimization problem.

One may find it is debatable whether a two-learner structure is really a preferable choice comparing to a single learner. For example, one of the alternatives is to train a linear classifier using its own predicted labels without leveraging data manifold information [80]. Unfortunately, such an idea is not effective according to my experiments. Sometimes, the results are even worse than not using any unlabeled data. The reason is twofold. First, a single unlabeled instance can hardly provide any useful information. Second, most of the online linear classifiers only return a single hypothesis on each round, precluding any other possible hypotheses. Hence, some previous work employed Bayesian methods to update a (posterior) distribution over the hypothesis [87, 73]. Unfortunately, the posterior is often complicated. It is not known how to perform the update analytically. Therefore, the learning process can be easily misled and stuck in a wrong direction. Another alternative is to use a graph-based method solely. However, due to the nonparametric nature of graph-based methods, it is not straightforward to deliver the server's model to clients with a low communication cost (for the same reason, nonparametric methods are not favorable in my problem setting). Moreover, graph-based methods are also less efficient for predicting new data, as they usually involves matrix inversion. A two-learner structure, in contrast, surmounts the above problems by complementing each other's drawbacks. The choice of two learners with different underlying mechanisms is a key for good performance.

If the communication cost is defined as the total number of vectors in $\mathbb{R}^d$ transmitted over the network, then a straightforward implementation of the proposed framework incurs a cost of at most

$$\underbrace{l + \lfloor \frac{v-l}{q} \rfloor \omega + \min\left((v-l) \bmod q, \omega\right)}_{\text{client to server}} + \underbrace{\lfloor \frac{v-l}{q} \rfloor}_{\text{server to client}} , \tag{11.1}$$

where l is the number of labeled instances on the server; v is the total length of the unlabeled sequence on the client; q is the size of the pool on the client; and ω is the number of uploaded instances every time the pool gets full. By ignoring rounding issues, this can be approximated as $l + \frac{v-l}{q}(\omega + 1)$.

In the rest of the chapter, I shall elaborate each component of the proposed framework, presenting their technical details and describing how they cooperate with each other to meet the global goal.

11.4. Online Semi-Supervised Learning on the Server

In this section, I describe the server's two learners. First, I review the two standard learners adapted to the proposed setting, and then I show how to adapt and combine them to handle a partially labeled data stream.

11.4.1. Soft Confidence-Weighted Classifier

I first describe the soft confidence-weighted (SCW) classifier for constructing hypotheses $h_1, h_2, \ldots, h_l$ in an incremental fashion. In a nutshell, the SCW algorithm maintains a Gaussian distribution parameterized by a mean $\mathbf{w} \in \mathbb{R}^d$ and a full covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$. The mean

$\mathbf{w}$ corresponds to the current linear function as described in Section 11.2. The covariance matrix $\mathbf{\Sigma}$ captures the uncertainty and correlation of each feature in $\mathbf{w}$. Given a new labeled instance $(\mathbf{x}_t, y_t) \in L$, SCW sets the new distribution to be the solution of the following optimization problem,

$$\begin{aligned}(\mathbf{w}_t, \mathbf{\Sigma}_t) :=& \arg\min_{\mathbf{w},\mathbf{\Sigma}} \{\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{w}, \mathbf{\Sigma}) \,\|\, \mathcal{N}(\mathbf{w}_{t-1}, \mathbf{\Sigma}_{t-1})\right) \\ &+ C \max(0, \phi\sqrt{\mathbf{x}_t^\top \mathbf{\Sigma} \mathbf{x}_t} - y_t \mathbf{x}_t^\top \mathbf{w})\},\end{aligned} \tag{11.2}$$

where the hyperparameter ϕ controls the confidence of each update, and C balances between passiveness and aggressiveness. Intuitively, the optimization problem trades off between two requirements. The first term forces the Kullback-Leibler divergence D_{KL} between the new weight distribution and the old one to be small, so that the parameters do not change dramatically per instance. The second term requires that the new vector $\mathbf{w}_t$ should perform well on $(\mathbf{x}_t, y_t)$.

This optimization problem has a closed-form solution:

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \alpha_t y_t \mathbf{\Sigma}_{t-1} \mathbf{x}_t, \quad \mathbf{\Sigma}_t = \mathbf{\Sigma}_{t-1} - \beta \mathbf{\Sigma}_{t-1} \mathbf{x}_t \mathbf{x}_t^\top \mathbf{\Sigma}_{t-1}. \tag{11.3}$$

The updating coefficients are calculated as follows:

$$\alpha = \min\{C, \max\{0, \frac{1}{\upsilon\zeta}(-m\psi + \sqrt{\frac{1}{4}m^2\phi^4 + \upsilon\phi^2\zeta})\}\}, \tag{11.4}$$

$$\beta = \frac{\alpha\phi}{\sqrt{u} + \upsilon\alpha\phi}, \tag{11.5}$$

where $u = \frac{1}{4}(-\alpha\upsilon\phi + \sqrt{\alpha^2\upsilon^2\phi^2 + 4\upsilon})^2$, $\upsilon = \mathbf{x}_t^\top \mathbf{\Sigma}_{t-1} \mathbf{x}_t$, $m = y_t \mathbf{x}_t^\top \mathbf{w}_{t-1}$, $\psi = 1 + \frac{\phi^2}{2}$ and $\zeta = 1 + \phi^2$.

Compared to other online linear algorithms such as passive-aggressive [36], confidence-weighted [37] and adaptive regularization of weights [38], SCW enjoys the adaptive margin property and reduces the total number of updates over rounds. Most importantly, SCW performs a more conservative update when dealing with a mislabeled instance [163]. In fact, in my experiments SCW outperformed other alternatives on many real-world data sets with noise. For this reason, we later use SCW to learn the instances with "noisy" labels predicted by the first learner.

11.4.2. Harmonic Solution

Harmonic solution (HS) is a graph-based semi-supervised learning method, which assumes that labeled data and unlabeled data are available in advance. Specifically, let $\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_L \\ \tilde{\mathbf{y}}_U \end{bmatrix}$ where $\mathbf{y}_L = [y_1, y_2, \ldots, y_l]^\top$ and $\tilde{\mathbf{y}}_U$ denote the estimated values on unlabeled data instances. The goal of HS is to minimize the quadratic objective function,

$$\tilde{\mathbf{y}}^* = \min_{\tilde{\mathbf{y}}} \tilde{\mathbf{y}}^\top \mathbf{\Delta} \tilde{\mathbf{y}}, \tag{11.6}$$

where $\mathbf{\Delta} = \mathbf{D} - \mathbf{S}$ is the graph Laplacian of the similarity graph, which is represented by a matrix $\mathbf{S}$ of weights $s_{i,j}$ that encode pairwise similarities, and $\mathbf{D}$ is a diagonal matrix whose entries are

given by $\sum_j s_{i,j}$. HS can be computed in a closed form, which has three representations as follows:

$$\tilde{\mathbf{y}}_U^* = (\mathbf{D}_{UU} - \mathbf{S}_{UU})^{-1}\mathbf{S}_{UL}\mathbf{y}_L \tag{11.7}$$

$$= -\boldsymbol{\Delta}_{UU}^{-1}\boldsymbol{\Delta}_{UL}\mathbf{y}_L \tag{11.8}$$

$$= (\mathbf{I} - \mathbf{P}_{UU})^{-1}\mathbf{P}_{UL}\mathbf{y}_L, \tag{11.9}$$

where $\mathbf{P} = \mathbf{D}^{-1}\mathbf{S}$ is the transition matrix on the graph.

In HS, the confidence of using labeled instances to predict unlabeled instances can be achieved in two ways. One way is to regularize $\boldsymbol{\Delta}$ in Eq. (11.8) as $\boldsymbol{\Delta} + \lambda\mathbf{I}$ where λ is a scalar and $\mathbf{I}$ is the identity matrix. When $\lambda = 0$, the solution turns into the ordinary harmonic solution. When $\lambda = \infty$, the confidence of labeling unlabeled instances decreases to zero. Alternatively, one can incorporate the knowledge given by $h_l = \{\mathbf{w}_l, \boldsymbol{\Sigma}_l\}$, i.e. the hypothesis constructed by SCW on labeled instances alone, back into HS. This is illustrated by the dashed line in Fig. 11.1. Specifically, denote by $\mathbf{g}_U$ the soft labels in $[0, 1]$ on unlabeled data produced by h_l, each element of which is computed by $\Phi(\frac{|\mathbf{x}^\top \mathbf{w}_l|}{\sqrt{\mathbf{x}^\top \boldsymbol{\Sigma}_l \mathbf{x}+1}})$, where Φ is the cumulative function of the normal distribution. Similar to Eq. (11.9), the harmonic solution after incorporated h_l is given by

$$\tilde{\mathbf{y}}_U^* = (\mathbf{I} - (1-\eta)\mathbf{P}_{UU})^{-1}((1-\eta)\mathbf{P}_{UL}\mathbf{y}_L + \eta\mathbf{g}_U), \tag{11.10}$$

where η is a scalar in $[0, 1]$. Setting $\eta = 0$ would reduce the solution to the ordinary harmonic solution. At another extreme, setting $\eta = 1$ would ignore the data manifold and completely rely on the predictions of h_l to train SCW.

11.4.3. Efficient Online Adaptation of HS

Note that one need an efficient online version of HS to fit it into the framework. I assume the server receives instances one-by-one. An obvious method is taking each new unlabeled instance, connecting it to its neighbors, and recomputing the harmonic solution. However, the matrix inversion involved has the computational complexity $\mathcal{O}(n^3)$ when the graph contains n nodes. Consequently, this naive solution quickly becomes impractical as more and more instances are added to the graph.

To address this problem, I restrict the size of the graph by substituting the vertices with a smaller set of k distinct centroids. Specifically, I make use of a *doubling algorithm* for incremental k-center clustering, which assigns points to centroids in a near optimal way [24]. The original algorithm maintains a set of centroids such that the distance between any two centroids in is at least R.

In my framework, the algorithm is adapted as follows. For initialization, we set R to a small positive number, k to be larger than l, and $V_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$. On round t, a new instance $\mathbf{x}_t$ is directly added to the set of centroids if $|V_{t-1}| < k$. If $|V_{t-1}| = k$, then the algorithm first tries to greedily remove a centroid from $V_{t-1} \setminus \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ such that every two centroids in the remained set are no close than R. If such attempt is not successful, then the algorithm doubles R and does the removal again. Finally, V_t is obtained by adding $\mathbf{x}_t$ to the modified V_{t-1}. Figure 11.2 illustrates this procedure. Note that on each round t, $V_t \subseteq \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ and $\mathbf{x}_t \in V_t$. Moreover, unlike the original version, the modified algorithm only guarantees that every two centroids in $V_t \setminus \{\mathbf{x}_1, \ldots, \mathbf{x}_l, \mathbf{x}_t\}$ are no closer than R.

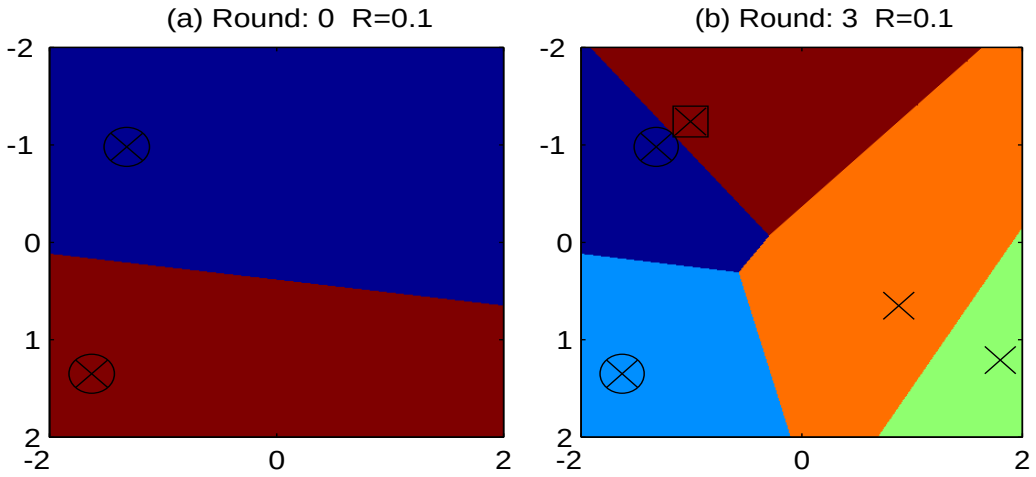


Figure 11.2.: Adapted doubling algorithm in our framework. ◯ is labeled point, × is centroid and □ is the current point on the t^{th} round. Color indicates the partition of the space according to the centroids. For this example, we set $l = 2$, $k = 5$ and $R = 0.1$. (a) Initially, the centroid set V_0 contains only two labeled points. (b) In the first three rounds, each new point is directly added to the centroid set. (c) On the 4^{th} round, as V_{t-1} is already full, we have to remove a centroid from it. we We double R to 0.2, remove the centroid corresponding to the red region from the 3^{rd} round, and add the current point to the centroid set. (d) We double R again, remove the centroid of the green region from the 4^{th} round, and add the new point. (e) The centroid set after 20 rounds. centroid set.

After restricting the size of the graph, the remaining bottleneck of HS includes updating $\mathbf{S}$ and inverting a $k \times k$ matrix. While the incremental update of $\mathbf{S}$ can be easily done with a block matrix, speeding up the matrix inversion is less straightforward [169]. In this work, I use conjugate gradient descent to solve an equivalent linear system and therefore avoid the expensive inversion. Note that the solution of $\tilde{\mathbf{y}}_U$ in Eq. (11.8) is equivalent to the solution of the following linear system,

$$\mathbf{\Delta}_{UU}\tilde{\mathbf{y}}_U = -\mathbf{\Delta}_{UL}\mathbf{y}_L. \tag{11.11}$$

The hope is that each iteration is $\mathcal{O}(k)$ and convergence can be reached in relatively few iterations, in contrast to the naive inversion that costs $\mathcal{O}(k^3)$. To ensure fast convergence, I use the Jacobi preconditioner, which is simply the diagonal of $\mathbf{\Delta}_{UU}$; and set the initial guess of $\tilde{\mathbf{y}}_U$ to be the solution of the last round.

11.4.4. Combining HS with SCW

I now show the construction of hypotheses $h_{l+1}, \ldots, h_n$ from unlabeled data U by combining HS with SCW. Intuitively, let HS teach its most confident predictions to SCW. To see that, I first

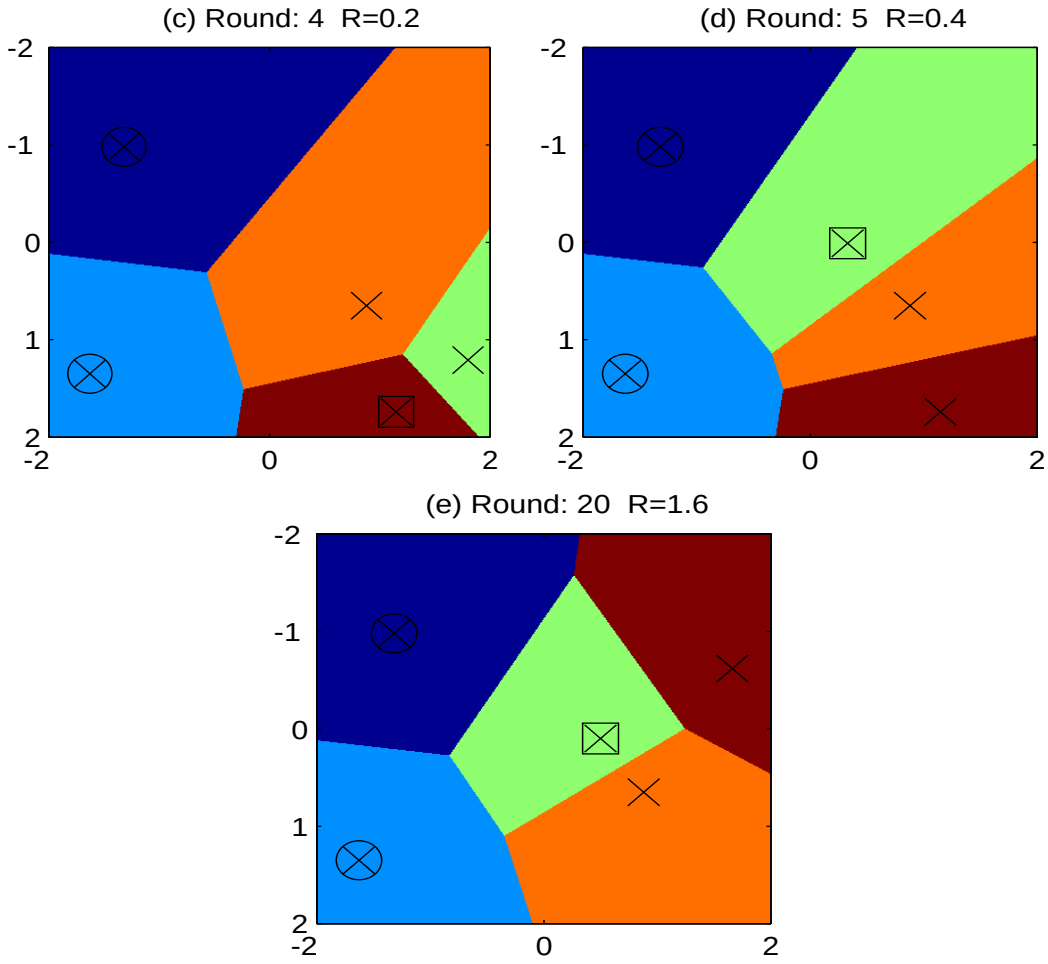


Figure 11.2. (cont.)

rewrite Eq. (11.9) so that each element of $\tilde{\mathbf{y}}_U$ is given by

$$\begin{aligned}\tilde{y}_i &= \sum_{j:y_j=1} (\mathbf{I} - \mathbf{P}_{Ui})^{-1}\mathbf{P}_{Uj} - \sum_{j:y_j=-1} (\mathbf{I} - \mathbf{P}_{Ui})^{-1}\mathbf{P}_{Uj} \\ &= p_i^1 - p_i^{-1}, \end{aligned} \tag{11.12}$$

where p_i^1 and p_i^{-1} can be interpreted as the probability of instance $\mathbf{x}_i$ belongs to the positive and negative class, respectively. Therefore, one can use $|\tilde{y}_i| \in [0, 1]$ to represent the confidence of predicting the label $\text{sign}(\tilde{y}_i)$ to the instance $\mathbf{x}_i$.

Though SCW and HS are conceptually separated, they can now be combined in an optimiza-

tion problem as follows,

$$
\begin{aligned}
(\mathbf{w}_t, \mathbf{\Sigma}_t) \coloneqq \arg\min_{\mathbf{w},\mathbf{\Sigma}} \{ & \mathrm{D}_{\mathrm{KL}} \left(\mathcal{N}(\mathbf{w}, \mathbf{\Sigma}) \,\|\, \mathcal{N}(\mathbf{w}_{t-1}, \mathbf{\Sigma}_{t-1}) \right) \\
& + C \max(0, \phi \sqrt{\mathbf{x}_j^\top \mathbf{\Sigma} \mathbf{x}_j} - \tilde{y}_j \mathbf{x}_j^\top \mathbf{w}) \} \\
\text{s.t.} \quad & j \sim \text{categorical}(\bar{p}_{l+1}, \bar{p}_{l+2}, \ldots, \bar{p}_k) \\
& \bar{p}_i = \frac{|\tilde{y}_i|}{\sum_i |\tilde{y}_i|}, \quad \text{where } \tilde{y}_i \text{ is given by Eq. (11.12).}
\end{aligned}
\tag{11.13}
$$

The combined algorithm works as follows. On round t, the new unlabeled instance $\mathbf{x}_t$ is added to V_{t-1} to construct V_t. The training instances fed to SCW are sampled according to HS confidence into labels of V_t. The highly uncertain predictions are likely to be excluded from learning.

Note that on round t the current instance $\mathbf{x}_t$ is always learned by HS (because $\mathbf{x}_t$ is added to V_{t-1} for constructing V_t), but it is not necessarily learned by SCW. Depending on the confidence of HS, SCW may be fed with any instance in V_t. More precisely, there are three outcomes of an unlabeled instance: (i) it is taught to SCW by random sampling; (ii) it is retained in the centroids set V; or (iii) it is removed from V by the clustering algorithm in Section 11.4.3.

One can observe some similarity between Eq. (11.13) and the objective function of online manifold regularization [72]. The latter used the manifold constraint as a regularization term in the objective function. While both methods attempt to learn a large margin separator using manifold information, the major difference is in the search space. In particular, online manifold regularization searches on a class of hypotheses to find one that is smooth on the graph. But when the hypothesis space is severely restricted, such as linear functions, the manifold regularization term simply turns into a penalty on the weight-vector, preventing the algorithm from harnessing any useful information about the manifold. My method, in contrast, learns a linear function conditioned on labels induced by the manifold, providing better performance and flexibility.

11.4.5. Predicting New Data

At any point in time, the learned SCW on the server can be used as a standalone component for predicting the labels for new (test) data. An obvious way is to use the last hypothesis directly returned by SCW as the output classifier. However, the training set could happen to be such that it ends up with a bad last hypothesis. To promote robustness and stability, I employ the *cutoff averaging* technique to build an ensemble as the output classifier [50], rather than committing to a single online hypothesis.

In cutoff averaging, each distinct online hypothesis is associated with a *survival time*, which is defined as the number of consecutive rounds the corresponding hypothesis survives before SCW replaces it with a new hypothesis. On the last round n, one has observed a sequence of online hypothesis $\{h_t\}_{t=0}^{n-1}$. Let $\Theta_\nu \subseteq \{h_t\}_{t=0}^{n-1}$ be the set of distinct hypotheses whose survival time is greater than ν. The cutoff averaging technique defines the output hypothesis h^* as a weighted average over the hypothesis in Θ_ν, where the weight of a hypothesis with survival time r is proportional to $r - \nu$. The cutoff parameter ν sets the bar for acceptance into the ensemble. Define the sequence of binary variables $\{B_t\}_{t=0}^{n-1}$ as follows

$$
B_t = \begin{cases} 1 & \text{if } t = 0 \text{ or if } t \geq \nu \text{ and } h_{t-\nu} = \ldots = h_t \\ 0 & \text{otherwise} \end{cases}
\tag{11.14}
$$

The optimal ν^* can be determined by solving the following optimization problem:

$$\nu^* = \arg \min_{\nu:\Theta_\nu \neq \emptyset} \left\{ \bar{\ell} + \sqrt{\frac{\gamma \bar{\ell}}{\sum B_t}} + \frac{7\gamma}{2 \sum B_t} \right\} \tag{11.15}$$
$$\text{s.t.} \qquad \bar{\ell} = (\sum_{t=0}^{n} B_t)^{-1} \sum_{t=0}^{n} B_{t-1} \ell(h_{t-1}; \mathbf{x}_t, \tilde{y}_t),$$

where γ is a constant with respect to ν. This solution ensures a large ν and a sparse ensemble if a few online hypotheses stand out with significantly long survival times. If most of the hypotheses have short survival times, then a small ν is preferred and the output ensemble is dense. Note that the maximal number of distinct survival times in a sequence of n hypotheses is $\mathcal{O}(\sqrt{n})$. Thus, the search space of ν^* is small enough for efficient computation.

11.5. Selective Sampling on Clients

Given a communication budget, the client needs to select instances from an unlabeled candidate pool such that the model on the server might be improved by learning these instances. Random selection is a simple approach, but a better selection criterion should meet the current demands of the server's model. To design such a criterion, the client needs (full/partial) information about the model currently on the server. Although there are two learners on the server, I transmit only the weight-vector of SCW from the server to the client because SCW directly determines the performance interested in (while HS serves to reduce the uncertainty of SCW) and transmitting only the weight-vector is communication-efficient.

Two important aspects of a good criterion are the utility and redundancy. The utility measures the potential improvement of SCW associated with each instance. The redundancy measures the degree of information sharing by the selected instances. For a candidate pool $Q = \{\mathbf{x}_1, \ldots, \mathbf{x}_q\}$, let the *utility score* be the sum of their individual utilities, i.e., $f_u(Q) = \sum_{i=1}^{q} f_u(\mathbf{x}_q)$. The redundancy is denoted by $f_r(Q)$. The desired selections should be optimal in terms of both utility and redundancy. Formally, given a communication budget ω for processing the pool Q, the goal is to select a subset T from Q such that

$$T = \arg \max_{T \subseteq Q : |T| = \omega} f_u(T) - f_r(T). \tag{11.16}$$

Previous research on active learning has proposed several choices for f_u and f_r [138, 67]. I use function value based scores, namely $\frac{1}{1+|\mathbf{x}^\top \mathbf{w}|}$, as f_u [21], and the sample covariance of instances as f_r [78]. As f_u is linear and $-f_r$ is *submodular*, Eq. (11.16) turns into a submodular function, which satisfies a *diminishing returns* property. A near-optimal solution of Eq. (11.16) can be found efficiently using a greedy algorithm [118]. Intuitively, this favors the instances that are close to the decision hyperplane of the current SCW and far away from each other. Note that the submitted instances are of low confidence according to SCW, and by querying HS for their labels, they may offer some supervision to SCW.

11.6. Experiments

I conducted a series of experiments to verify the effectiveness of the proposed framework in the context of communication-efficient online semi-supervised distributed learning. The first experiment focuses on the server's model and compares the proposed two-learner method against several baselines including its one-learner counterparts. The second experiment focuses on the client's selection criterion. Finally, a sensitivity analysis for the framework is presented to gain more insights into its performance.

11.6.1. Experimental Setup

Experiments were conducted on seven data sets downloaded from either the UCI ML repository (wearable, skin) or the LIBSVM website (mushroom, mnist, webspam, gisette, ijcnn1). The motion recognition data set wearable and digit recognition data set mnist were converted into a set of binary problems, respectively, where each class is discriminated against every other class. Totally, I produced 20 problems from wearable and 45 from mnist. For each data set, I balanced the number of instances of each class and linearly rescaled the feature values into the range $[-1, 1]$.

I evaluated the algorithms using a set of trials with different partitions of the training and test data. In each trial, I randomly held out half of the data for testing; all instances in the test set were labeled by the algorithms. The remaining data was used for training, of which only a small amount was labeled. Both training and test sets were class-balanced. Next, I randomly permuted the training data and kept labeled data always at the beginning. All algorithms were then incrementally trained with the same permutation in each trial. For evaluation, I paused the training at regular intervals, computed the output hypothesis so far, and calculated its test accuracy. I used the first trial to tune the hyperparameters (e.g., C, ϕ in Eq. (11.13)), and the best choice for its hyperparameter is then fixed in the remaining trials. I used $\eta = 0$ in Eq. (11.10). The reported results were averaged over 100 trials.

In all experiments, I used a 5-nearest neighbor graph as the similarity graph of HS on the server. The edges were weighted as $s_{i,j} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2d\sigma^2})$, where d is the number of features and σ denotes the mean of their standard deviations. The maximum number of centroids in the graph was 300. In Sections 11.6.2 and 11.6.3, the initial 2% of the training instances are labeled. The size of the candidate pool on the client was 50, from which 10 instances were submitted to the server (a 20% sampling rate). Section 11.6.4 presents a sensitivity analysis to the labeling and sampling rates. My implementations and experiments code are public available[1].

11.6.2. Comparison of Server's Model

I first compare different models on the server and show the effectiveness of the proposed method. To focus on the server side, I let the client randomly select instances to upload. In particular, the following methods were evaluated in this experiment.

none No unlabeled instances are uploaded to the server. The server stops learning right after labeled instances. Assuming that unlabeled instances can provide useful information, then this approach should give the worst performance.

[1] http://home.in.tum.de/~xiaoh/kdd2014/kdd2014code

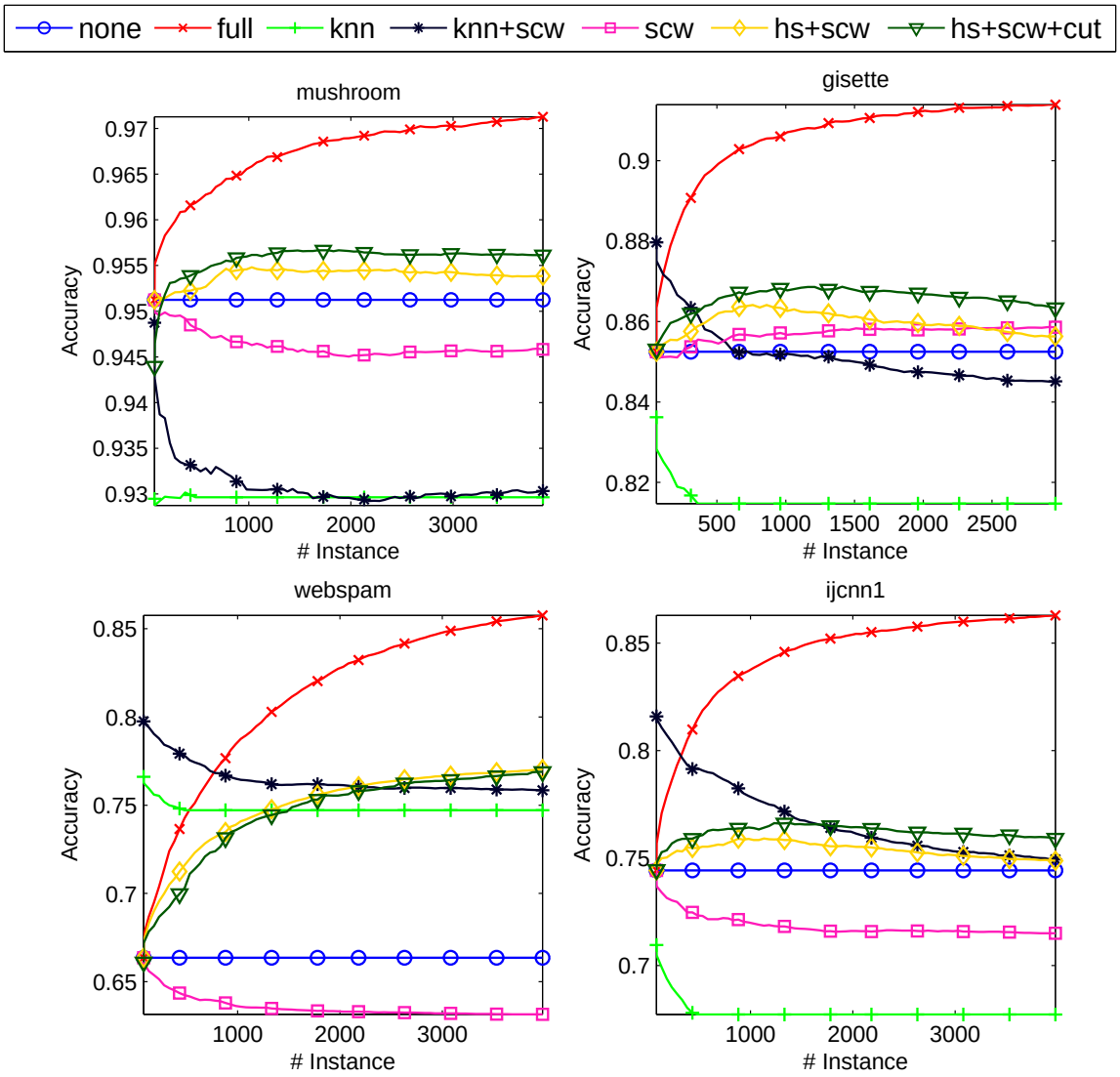


Figure 11.3.: Test accuracy of different models on the server. The x-axis represents the number of unlabeled instances on the client. The origin corresponds to the point where the initial 2% of the data has been labeled and learned and the first unlabeled instance comes in. The client randomly selected 10 instances from every 50 instances. `full` is an idealized approach in which an oracle labels all selected instances. `none` does not upload any unlabeled instance to the server, so the corresponding test accuracy is constant.

`full` All uploaded instances are labeled by an oracle. Intuitively, this approach should give the best result due to the availability of full information. This is an idealized case with 10x (20% vs. 2%) more labeled data.

`knn` The server employs k-nearest neighbors algorithm, where $k = 5$. The training set is built by first including all labeled instances, and then adding unlabeled instances with its corre-

sponding predicted labels. The maximum number of allowed training examples is 300.

knn+scw The server consists of a two-learner model: knn followed by scw. The prediction of knn is used for training scw.

scw The server consists of an SCW model only, which "learns" each unlabeled instance using its own prediction.

hs+scw Proposed two-learner model on the server.

hs+scw+cut Proposed hs+scw model with cutoff averaging for predicting test data.

Note that, none and full are essentially standard online learning, in which models are trained with labeled data. I also implemented a baseline containing HS only on the server. However, I had to terminate it due to its poor efficiency. A comparison of the above methods is shown in Fig. 11.3.

It can be observed that proposed hs+scw and hs+scw+cut enjoy superior performance on 8 out of 10 problems comparing to other partial label competitors. On 45 mnist problems, hs+scw and hs+scw+cut yielded on average 0.966 and 0.971 accuracy, respectively. On 20 wearable problems, hs+scw and hs+scw+cut gave 0.699 and 0.714 accuracy, respectively. They are consistently better than the single-learner counterpart scw on all data sets. This indicates the effectiveness of leveraging manifold information of the graph. In fact, on webspam, ijcnn1 and wearable, scw is even worse than none. On webspam, its test accuracy starts with 0.658, decreasing over time and finally yielded 0.637. This is due to the fact that scw completely relies on its own prediction for learning. When the labeling rate is small, the initial hypothesis constructed by labeled data may not be accurate enough. As a consequence, the prediction of scw on the new instance is likely to be wrong, which in turn might mislead the learning procedure. The knn-based approaches, which employ majority voting based on local information, did not show consistent performance. On gisette, webspam, and ijcnn1, the test accuracy of knn decreases until the maximum number of training instances is reached, whereas on mnist it increases. This indicates that a simple bootstrapping for knn is not robust. Also note that, it is not straightforward to formulate a communication-efficient selection policy for knn due to its nonparametric nature. The idea of using the prediction of knn to teach scw is not effective, often resulting in degraded performance of scw over time. One may note that knn enjoys superior performance on skin. This is probably due to the characteristics of this data set. Each instance in skin has only three features, representing red, green, and blue color, respectively. The task of distinguishing skin from non-skin on such data is particularly suitable for knn.

11.6.3. Comparison of Selection Strategy

Fixing the model on the server as hs+scw, I study the following strategies on the client side.

all All unlabeled instances are uploaded without selection. This incurs 5x the communication costs versus other approaches.

rand Randomly selects instances for uploading.

certain The most certain instances according to the current server model $\mathbf{w}$ are uploaded. The score is defined as $|\mathbf{x}^\top \mathbf{w}|$. This method is similar in spirit to [80].

uncertain The most uncertain instances are uploaded. The score is defined as $\frac{1}{1+|\mathbf{x}^\top \mathbf{w}|}$.

submod Selection is done by optimizing the submodular function described in Section 11.5. It simultaneously considers the uncertainty and redundancy.

Note that there are many ways to wrap $|\mathbf{x}^\top \mathbf{w}|$ into a selection criterion, such as transforming it into a probability value [73, 30]. However, despite introducing extra hyperparameters into the model, they are not significantly different in essence. For the sake of clarity, I concentrate on the above five strategies. The result is shown in Fig. 11.4, in which `none` and `full` are as defined in Section 11.6.2.

It is interesting to see that `all`, which transmits all unlabeled data, does not lead to better performance. In fact, on mnist, mushroom, and gisette, `all` yields worse test accuracy compared to selective transmission. This confirms the intuition that not all unlabeled instances are useful. It also suggests the necessity of using a selective sampling strategy on the client. Not only the communication costs can be saved, but also a better model might be learned. Moreover, it can be observed that `uncertain` and `submod` show significant improvements over `rand`. They often converge faster than `rand` and lead to better optimal hypotheses. On the contrary, selecting most certain instances is not beneficial. On ijcnn1 and skin, the accuracy decreases over time (the accuracy of `certain` on skin drops to under 80% at 4000 instances, and is not shown to better see the other results), showing that a bad client selection strategy can have negative impact on the performance of the server's model. On mnist and mushroom, `submod` further improves over `uncertain`, while `uncertain` is better for gisette and ijcnn1.

11.6.4. Sensitivity Analysis

The goal of this experiment was to investigate how the labeling and sampling rates impact the test accuracy of the server's model. I used `hs+scw+cut` on the server and `submod` on the client because this combination enjoyed the best results according to the previous experiments. Figure 11.5 shows the result.

It can be easily identified that the brightest and darkest areas of each matrix are often located in the bottom-left and top-right corners, respectively. This suggests that better test accuracy can be achieved by increasing the sampling rate or labeling rate, which is consistent with the intuition. Unlike the sampling rate, changing the labeling rate often does not significantly affect the performance. For example, on mushroom the accuracy is quite insensitive to the labeling rate but improves significantly with increasing sampling rate. Across all data sets, a more promising way to improve the performance on the server's model is to incur increased communication costs by sending more data, rather than increased human costs by manually labeling more instances.

11.7. Conclusion

This chapter poses a new learning problem on the client-server design, which is motivated by real-world applications such as intelligent traffic systems and wearable devices. To solve this problem, I have presented a framework that provides communication-efficient online semi-supervised learning in the client-server setting. The framework consists of two parts. On the server side, two learners work collaboratively to learn from a partially labeled data stream. The two-learner structure can effectively exploit the data manifold to determine labels for unlabeled data. It is also

efficient in the sense that it does not require storing all the data. The proposed method enjoys superior and stable performance on several real-world data sets. On the client side, I investigated several selection criteria and showed how the server communicates with the client. I showed that a selection criterion based on uncertainty and redundancy is effective. It is worth highlighting that intelligent sampling on the client not only saves communication costs, but, perhaps surprisingly, also may result in a better model on the server compared to uploading all instances.

The promising results in this chapter raise a few important questions. First, I have not studied how to adapt the sampling rate over time. Intuitively, the model on the server requires more data to learn in the beginning. As the learning procedure goes on, the potential hypothesis space on the server shrinks and thus requires less data to learn. Hence, it makes sense to compute the optimal number of selected instances on each round for further reducing the communication cost. Second, I have assumed the incoming data on the server is stationary. In many real-world applications, the true hypothesis is not fixed but slowly changing over time. Assuming a small number of additional labels are available during the drift (e.g., with the help of human annotators), it would be straightforward for my framework to handle such drifts. Specifically, I first adapt HS to include newly labeled data in the set of centroids V. Then, we reset the learning rate of SCW, which is controlled by $\mathbf{\Sigma}$. Third, it is interesting to employ other combinations of learners in the proposed framework. Because my framework is modular in design, it offers flexibility for incorporating additional algorithms.

While for simplicity I considered the case of a single client, my framework can be readily extended to learn a model across multiple clients. Because a client's candidate pool is discarded once its selection has been made (i.e., there is no per-client history), processing a full pool of data is the same regardless of which client processes it (assuming iid data streams). Thus, as long as the server sends the current weight-vector to the client who is about to send data next, the processing and hence the accuracy is effectively the same as in the single client case, with the same overall sampling rate. If the data generation rate (data instances per second) increases linearly with the number of clients, though, two issues arise. First, the aggregate weight-vectors per second that the server sends increases linearly. This can be mitigated by having the server operate in rounds such that instances from multiple clients are processed in each round, and an updated weight-vector is sent only at the end of the round. This implies that clients select instances based on a less-frequently-updated uncertainty measure. Note also that in this case, one would like to apply the redundancy measure *across* the clients, which would require additional communication and coordination. Alternatively, one can use the simpler `uncertain` selection criterion, which performed nearly as well as `submod` (i.e., as uncertainty + redundancy). Second, the aggregate data sent by clients increases linearly. This can be mitigated by sticking to a fixed aggregate data instances sent per second, using some combination of subselecting which clients send instances at each round and reducing the sampling rate of those clients that do.

Finally, another interesting area for future work is to adapt the framework and integrate the concepts of *Internet of Things*, where sensors and actuators embedded in physical objects are linked through wired and wireless networks. In this setting, the widely deployed smart things can be seen as clients in my framework. The proposed framework enables (near) instantaneous responses from a server for planning and decision making, and saves communication cost through the selection procedure on the clients.

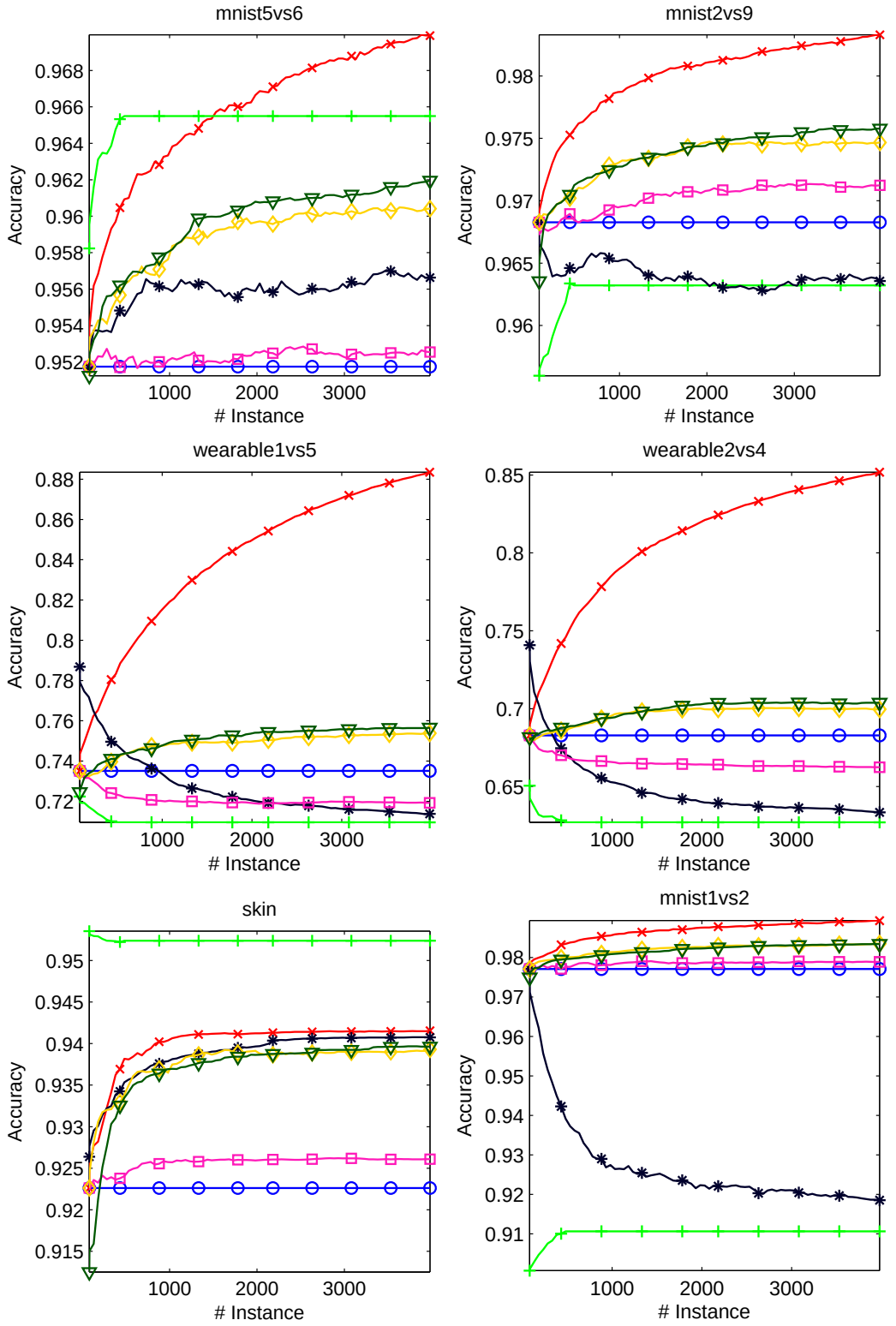


Figure 11.3. (cont.)

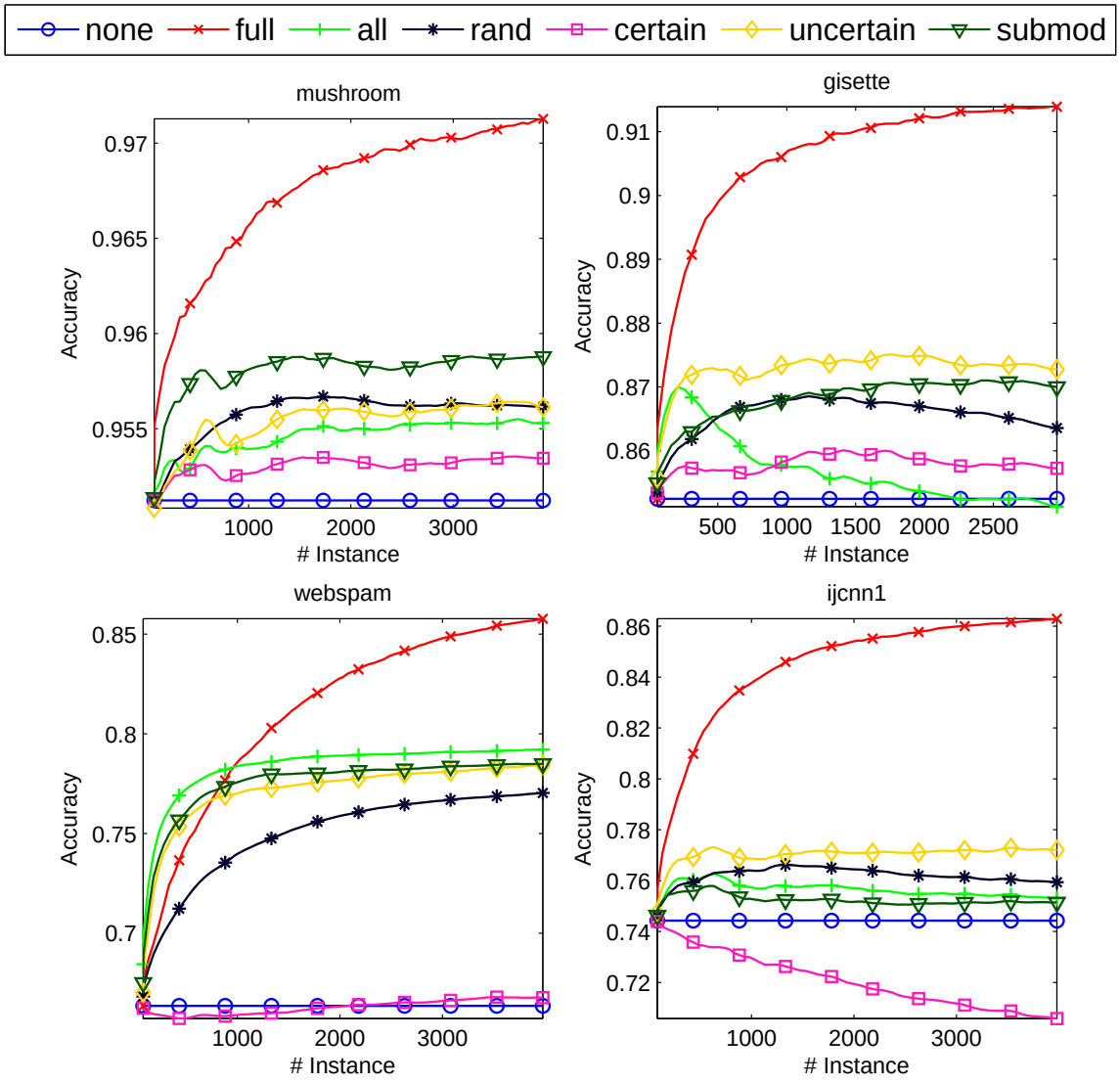


Figure 11.4.: Test accuracy of different selection strategies for a fixed communication budget. The client selected 10 instances from every 50 instances, except for `all`, which selected all instances and hence incurs 5x the communication costs. The server used `hs+scw+cut`. The labeling rate was 2%, except for `full`, which labeled all selected instances using an oracle.

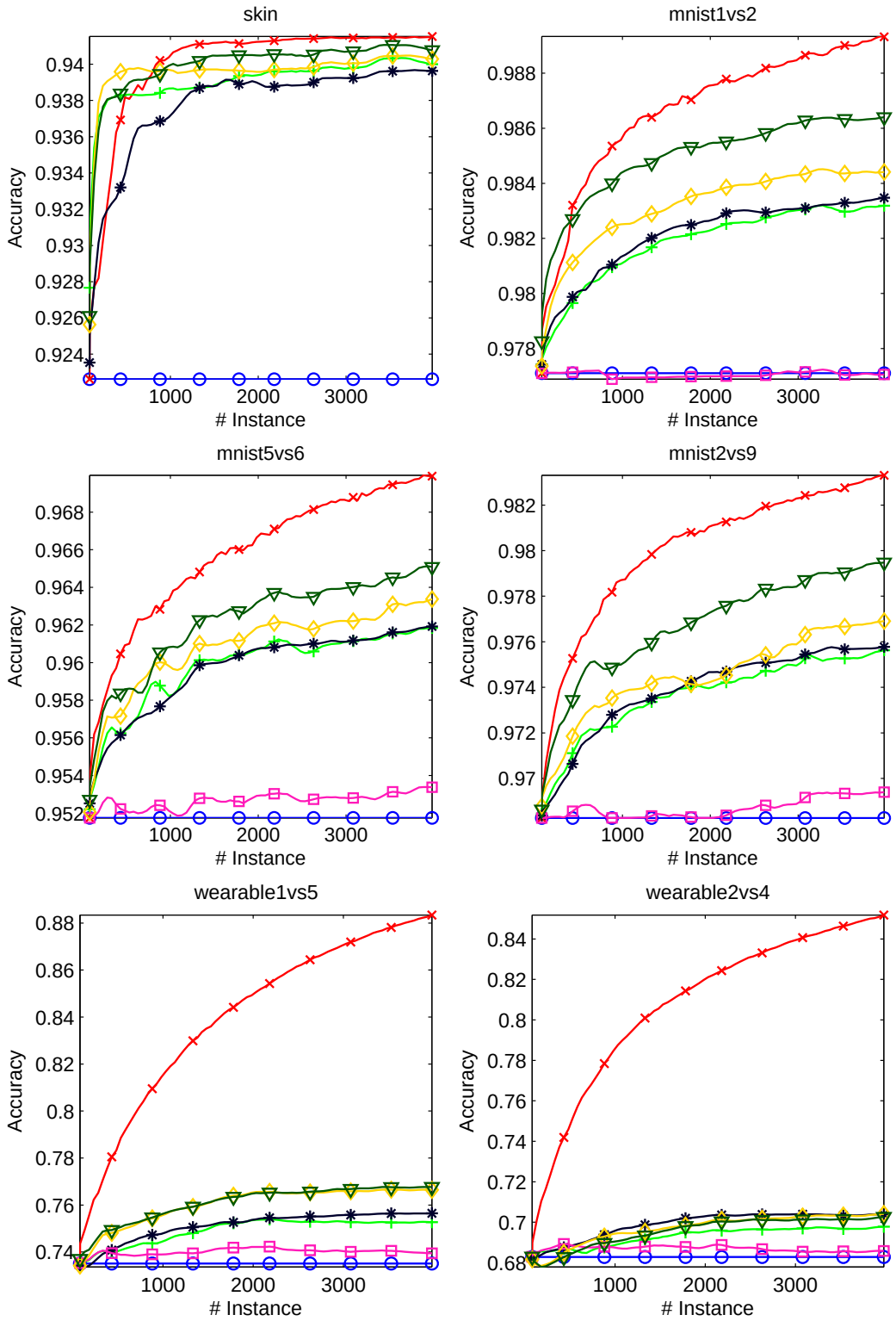


Figure 11.4. (cont.)

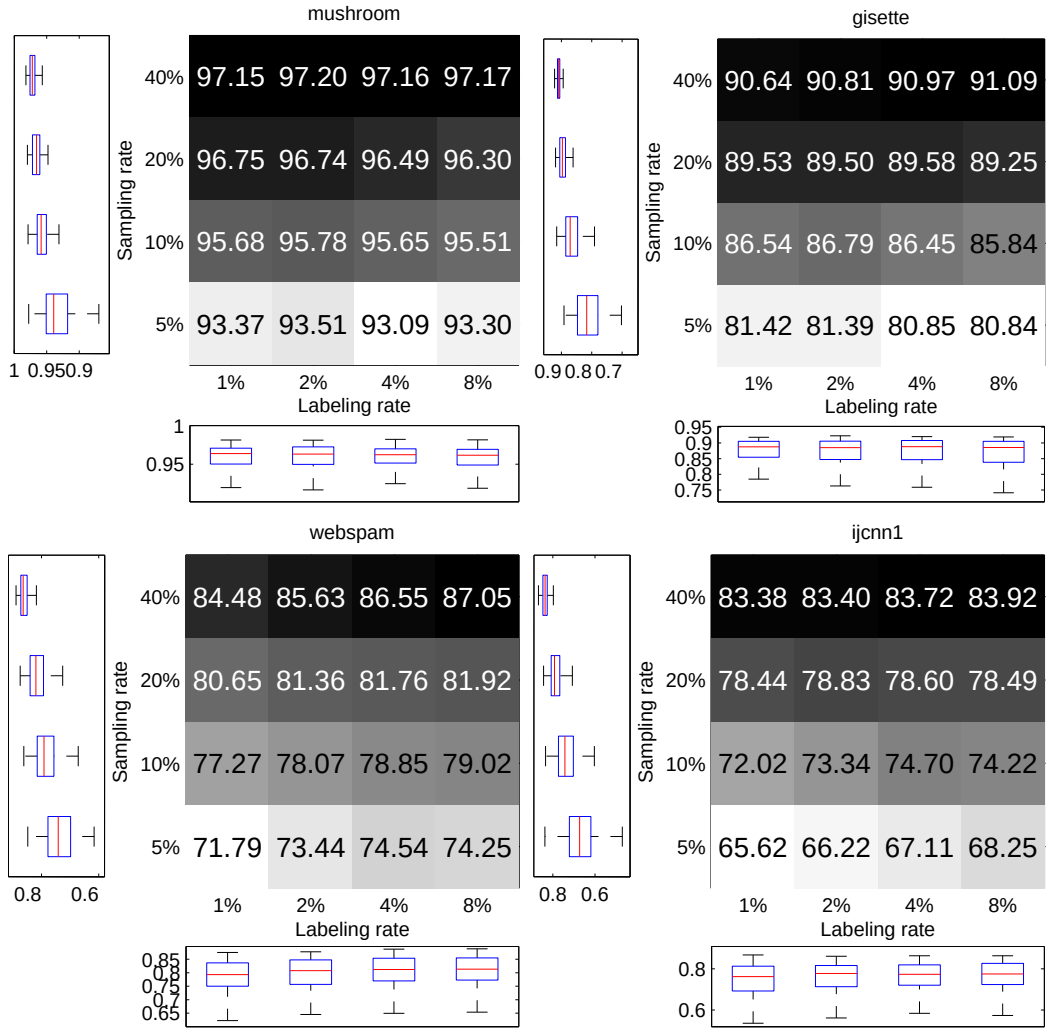


Figure 11.5.: Sensitivity analysis of the labeling rate (amount of human effort) and sampling rate (amount of communication) on different data sets. The value of the matrix represents the mean test accuracy of the last hypothesis constructed by `hs+scw+cut`. Darker color represents higher value. The column represents the labeling rate, varying from 1%, 2%, 4% to 8%. The row represents the sampling rate on the client, varying from 5%, 10%, 20% to 40%. The size of the candidate pool is 100. The selection strategy is `submod`. The result is averaged over 100 trials. The marginal boxplots are depicted along the corresponding axes. The values outside the range of $[Q_3 - 1.5(Q_3 - Q_1), Q_3 + 1.5(Q_3 - Q_1)]$ are considered as outliers, where Q_1 and Q_3 are the 25^{th} and 75^{th} percentiles, respectively. All outliers are removed from the boxplot for the sake of clarity.

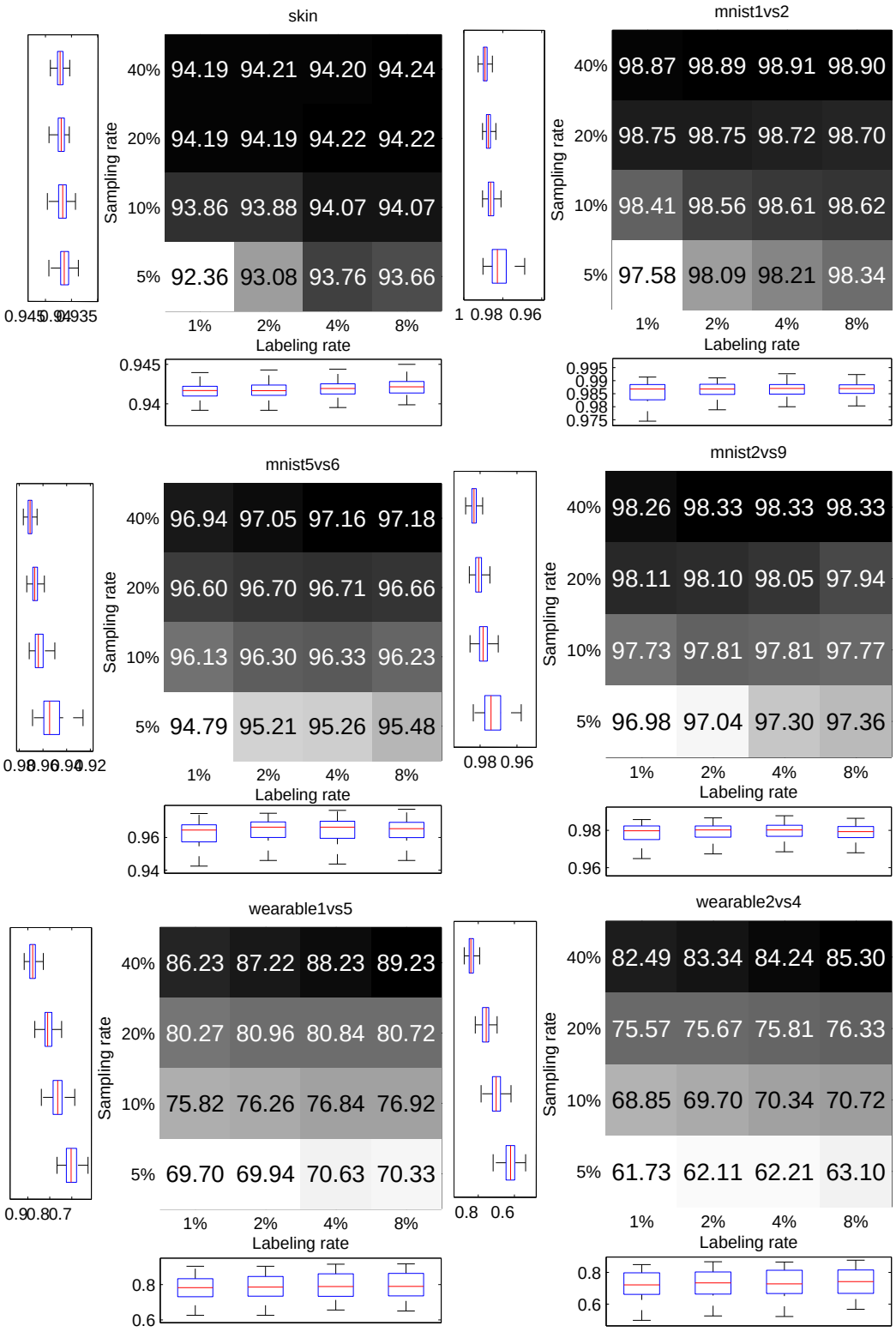


Figure 11.5. (cont.)

Chapter 12

Conclusion

Due to the ability to quickly adapt and to find patterns in large diverse data sources, machine learning algorithms become a great potential asset to application developers in security domains. They have become a valuable tool for detecting and preventing malicious activity. In many of these systems, human participants may exploit the adaptive component to gain some advantage. Therefore, it is important for practitioners quantify the vulnerabilities presented in existing learning techniques, and design robust and scalable learning mechanisms for security-sensitive applications. The work I have presented in this dissertation significantly advanced the state-of-the-art in this field of study with four primary contributions: a vulnerability analysis of linear classifier and convex-inducing classifier, a reliable algorithm for learning from multiple observers, scalable online algorithms for security-sensitive applications, and finally, a novel framework for communication-efficient distributed learning. However, research in this field has many challenges remain. These challenges suggest several new directions for research within both fields of machine learning and computer security.

The remainder of the chapter summarizes the main contributions of this dissertation in greater detail, and lists several open problems in the intersection of machine learning and security and my outlook on the future of this field.

12.1. Summary of Contributions

The contributions of this dissertation span the spectrum of practical attacks on learning algorithms, new robust and scalable learning algorithms, and thorough experimental evaluation. We detail these contributions below.

12.1.1. Identifying Vulnerabilities of Algorithms and Adversarial Capabilities

The first major contribution of this dissertation is the theoretical analysis for the vulnerabilities of machine learning algorithms. Traditional machine learning research were originally conceived under the assumption of faithful data and did not explicitly account for potential data manipulation by adversaries. In Chapter 4 and Chapter 3, I studied the exploratory attack where an adversary disguises the malicious instance as benign by querying the classifier. In Chapter 4, I presented the definition of the vulnerability of a multiclass classifier to linear probing. I showed that the

effective exploratory attack based on the linear probing is feasible under some assumption on the adversarial cost. The theoretical establishment in Theorem 4.3 not only reveals the vulnerability of the classifier, but also implies a way to construct a robust classifier that resists exploratory algorithm based on linear probing, e.g. by jointly minimizing Eq. (4.9) and the error function in the training procedure. Based on theoretical results, I presented an algorithm for deceiving the multi-class linear classifier by disguising the adversarial instance as other classes with approximately minimal cost, while issuing polynomially many queries in: the number of features, the range of feature value, the number of classes and the number of iterations. Chapter 5 continued the study of the exploratory attack problem and generalized it to the family of classifier with convex-inducing decision boundaries. I formalized the exploratory attack on convex-inducing classifiers as a ℓ_p-norm minimization problem. To solve this problem, I developed an algorithm based on random walks in a sequence of progressively smaller convex bodies. Underlying the algorithm is a sophisticated analysis, which clearly elucidates the convergence rate, the upper bound of expected iterations and the required number of samples per iteration. In the experiment, I showed that the proposed algorithm yields a robust convergence rate by bounding the feasible region with norm balls. In Chapter 6 I focused a different type of attack called causative label flips attack. The problem studies the best a combination of label flips under a given budget so that a classifier trained on such data will have maximal classification error. I presented an optimization framework based on Tikhonov regularization for solving this problem. I then proposed ALFA algorithm for attacking support vector machines, which can be efficiently solved by alternating between a LP and a QP solver. A lesson learned from Chapter 6 is that previous robust learning algorithms based on the assumption of random label noise may be too optimistic as they underestimate the adversary's impact on the classifier's performance. I also demonstrated the devastating impact of the attack algorithms on a newsletter classifier and a face recognition system. Hopefully, I have raised enough attention about the vulnerabilities of the machine learning algorithms, and underlined the importance of reliable learning for the machine learning researchers and security analysts.

12.1.2. Presenting Reliable Algorithms Resilient to Adversaries

The second principal contribution I make in this dissertation is developing a novel reliable learning algorithm that is less prone to adversarial data. In Chapter 7, I considered the problem of supervised learning from multiple observers, where each instance in $\mathcal{X}$ is associated with multiple but unreliable labels in $\mathcal{Y}$. Given the training data $\mathbf{X}$ and $\mathbf{Y}$, the goals were threefold: (a) estimate the unknown ground truth $\mathbf{Z}$; (b) learn a regression function $f : \mathcal{X} \to \mathcal{Z}$ generalized well on unseen instances; (c) for each observer model its expertise as a function of the input instance and the ground truth. Unlike previous work that used probabilistic parametric model to address the classification problem [130, 131], I focused on the regression problem and designed a nonparametric model using Gaussian process. The underlying assumption is that, if two instances are close to each other in the input space $\mathcal{X}$, then their corresponding groundtruth should be close in the latent space $\mathcal{Z}$, which results in the similar position in $\mathcal{Y}$. The parameters of the model can be estimated by maximizing the posterior of the latent variables with L-BFGS solver. From a practical perspective, I highlighted a wide range of domains that my method can be applied on, from crowdsourcing platform, photo rating website to sensor network. In Chapter 8, I demonstrated an application of aesthetics score assessment for the online photo sharing service. In this example, for each photo only the subjective ratings from multiple users are observed, whereas the objective

aesthetics score is a latent variable. Comparing to the simple heuristics such as "take the average", "majority vote", and "filter out anomalies", my method suggests a new way to leverage information from multiple sources. From the technical perspective, I found the underlying connection between the proposed model and other hierarchical models such as, hierarchical GP-LVM [102] and the hierarchical probabilistic PCA [155]. A similar training approach could be taken with the proposed model. Moreover, I highlighted the potential risk of overfitting when the number of latent variables is large or when the hierarchy becomes too deep. My work suggests that the overfitting problem can be alleviated when using proper initialization on each layer. Other methods such as adding prior distribution or regularization on each node may also be effective.

12.1.3. Presenting Online Algorithms for Large-Scale Data Stream

Robust learning algorithms are otherwise useless if they are not efficient enough in real-world problems. Thus, my third contribution is the design and analysis of efficient online learning algorithms for handling large-scale data. In Chapter 9, I focused on speeding up Gaussian process regression for online learning, which can be used to improve the training efficiency of the model in Chapter 7. In particular, I pointed out the the heavy computation of the likelihood and the derivatives involved in the ordinary Gaussian process model. To solve this problem, the proposed LGPC algorithm partitions the data and allocates it to a committee consisted of a number of independent GPs with different hyperparameters. Thus the main problem turned into the allocation of new training examples, which was solved using submodular optimization. Experiments showed that LGPC is a more appropriate choice than SOGP in real-time applications which require fast training. Moreover, its predictive accuracy makes it overall more preferable than BCM. The size of the committee determines the capability to account for the complex pattern. Larger size generally leads to higher predictive accuracy. I applied LGPC to mouse-trajectory prediction in an Internet banking scenario and used it to predict user's hand movements in real-time.

In Chapter 10 I studied the sequence prediction problem. I formulated it as an online multi-class classification problem, which can be seen as a complement to the online regression problem in Chapter 9. The problem is motivated by the system call prediction. Previous work that rely on high-order Markov models [27] were often difficult in practice due to the requirement of vast amounts of training data and more sophisticated smoothing algorithms. My method was based on the multi-class confidence-weighted algorithm [53]. I described the method to learn the weight matrix $\mathbf{W}$ by minimizing the Kullback-Leibler divergence. For storing previously observed sequence, I used a memory-efficient context tree, which grows the context tree at a much slower pace so that the algorithm can utilize memory more conservatively. A distinguished factor of my work is incorporating the side information into the model. Its effectiveness had been shown in the experiments. The solution of this problem can be extremely useful in computer security applications, such as anomaly detection [166, 57], sandbox systems [123], buffer cache management in operating system [66] and power management in smartphones [124].

12.1.4. Establishing Distributed Learning Framework for Client-Server Settings

To meet the challenge of big data, my fourth contribution is posing a novel learning problem and establishing a flexible framework for the client-server design. In particular, I considered a scenario where a distributed system consists of clients, a server, and a communication network. The clients submit partially labeled training data to the server. The server learns a model from incoming data.

This new setting is called communication-efficient learning, which aims to keep the performance of the model while reducing the communication cost over the network. This problem abstracts a common scenario where training data is too big to be stored locally and labeled completely. In my proposed framework, the server performs online semi-supervised learning. I employed the harmonic solution (HS) [180] as the first learner and the soft confidence-weighted classifier (SCW) [163], which is an extension of the algorithm in Chapter 9, as the second leaner. My choice offered several advantages. First, SCW is simple, fast and enjoys state-of-the-art performance on classification. Second, SCW performs a conservative update especially with noisy labels. Third, SCW can be parameterized by a weight vector and a covariance matrix, allowing the server to deliver the selection criterion to the client with a low communication cost. To determine the labels for unlabeled data, I proposed an efficient online version of HS, which can be integrated into SCW in an optimization problem. Given a communication budget, the client selects instances from an unlabeled candidate pool such that the model on the server might be improved by learning these instances. This was done by using the similar technique as in Chapter 9. The proposed method enjoys superior and stable performance on several real-world data sets. On the client side, I investigated several selection criteria and showed how the server communicates with the client. I showed that a selection criterion based on uncertainty and redundancy is effective. It is worth highlighting that intelligent sampling on the client not only saves communication costs, but, perhaps surprisingly, also may result in a better model on the server compared to uploading all instances.

12.2. Discussion and Open Problems

To develop a reliable and scalable system in the real world, one has to take relevant security threats into the consideration. While previous chapters marked several contributions to the field of machine learning in security-sensitive domains, a number of open issues and future directions arise. In the remainder of this section, I identify promising directions for designing reliable and scalable machine learning algorithms.

12.2.1. Faithful Evaluation with Scarce Groundtruth

Unlike many other machine learning applications, groundtruth data is hard to obtain in adversarial environments. Moreover, defining groundtruth for adversarial learning is difficult since the concept of malicious behavior is vague and often depends on the context. Furthermore, it is also difficult to leverage crowdsourcing platforms (e.g. Amazon Mechanical Turk) to label adversarial data, as attackers may hide their activities so that even humans are not able to identify them. It potentially requires domain-expert knowledge to correctly identify adversarial events. This challenge poses several open problems.

In addition, current evaluation techniques for performance measurement of machine learning algorithms do not take into account adversarial noise. Thus, such techniques can not provide information about the security level of a classification system under attacks. As experiments in Chapter 7 showed, they are likely to provide over-optimistic estimates of their performance. It is necessary to develop evaluation methods to measure the security level of classifiers on a given set of data.

Open Problem 1. *How to collect labeled data sets with adversarial events?*

Open Problem 2. *How to numerically evaluate the security level of a learning algorithm?*

12.2.2. Detecting Malicious Training As Pre-Processing

A straightforward way to reduce the adversarial noise is to remove the malicious instances before the training phase starts. In other word, one simply filters the data before training any learner to allow the learner to be more secure. The primary challenge is how to accurately identify malicious data that should affect the learner in adverse ways.

Traditional anomaly detection methods look appropriate at the first glance. However, they generally require a clean data set for the initial training of the detector itself. In practice, it could be difficult to obtain a clean data set. Even if there is such data set, the size is usually limited. Furthermore, as there will be less data to train on after removing suspect data, the learner may require more input to learn the target function. Thus, the problems here are summarized as follows.

Open Problem 3. *How to efficiently detect malicious instances from training data?*

Open Problem 4. *If training data is not given in batch but given in a sequence form, how to detect malicious instances in an online fashion?*

Open Problem 5. *How does the removal of malicious instances affect the learning rate of the algorithm?*

Open Problem 6. *Is the detector itself vulnerable to adversaries?*

12.2.3. Ensemble Methods for Secure Learning

In Chapter 9, the proposed LGPC model consisted several learners. Intuitively, even if a single learner may be individually vulnerable, it is more difficult for adversaries to attack all learners simultaneously. This suggests an ensemble may have several advantages in a security-sensitive environment. First, an ensemble can be shown to have more flexibility in the functions they can represent. Second, it allows one to combine learners designed to capture different aspects of the task. For instance, one can build an ensemble based on different feature sets to reduce common vulnerabilities under feature noise. Third, as the hypothesis produced by an ensemble is not necessarily contained within the hypothesis space of the learner from which it is built, it is more difficult for the adversary to reverse engineer the system.

To properly develop an ensemble method for secure learning, one must first assess the vulnerability of several candidate learners. Then, one should choose a base set of models and sets of features for them to learn on. If an ensemble is flexible enough, one can also improve the existing ensemble by adding a new learner according to the security threats. Remaining open problems in this line of research include:

Open Problem 7. *Do more learners always suggest more security?*

Open Problem 8. *How to assess the vulnerability of a group of learners?*

Open Problem 9. *When a new security threat is identified, how to patch an ensemble without training all learners from the scratch?*

Open Problem 10. *In the online learning scenario, how to train multiple learners asynchronously?*

12.2.4. Privacy-Preserving Learning in Distributed Settings

Chapter 11 introduced a new learning problem for the distributed setting. In practice, different clients may require different selection policies in order to reduce the redundancy between their submissions. To personalize the selection strategy, each client needs to be aware of what others submit. Therefore, the goal turns into designing an aggregate statistics on a data set without disclosing local information about individual elements of the data. It is often the case that the goals of *utility* and *privacy* are inherently discordant. For a mechanism to be useful, its responses must closely resemble some target statistic of the data entries. However to protect privacy, it is often necessary for the mechanism be "smoothed" to reduce the individual entries' influence on this distribution. This trade-off poses some interesting research problems.

Open Problem 11. *If one of the client is controlled by the adversary, how does it affect the learning algorithm on the server? Is this equivalent to the problem in Chapter 6?*

Open Problem 12. *Can we design a communication protocol for clients, so that the goal of utility and privacy can be achieved simultaneously?*

Open Problem 13. *Can we design a communication protocol for both server and clients, so that the goal of maximizing utility and privacy and reducing the network overall bandwidth can be achieved simultaneously?*

12.3. Final Words

This work began as a keen interest in machine learning and its application in security sensitive domains. It has raised a broad variety of research questions. Some of these questions stem from machine learning methodologies, others stem from the real-world security applications. Although traditions and practices (e.g. model analysis, experimental work) of machine learning and computer security diverge in many aspects, this dissertation has identified several interesting and important problems in this emerging discipline. These problems often require a thorough recapitulation of its theoretical foundations. Understanding these issues represents a significant step toward real-world secure learning. I expect that the demand for secure machine learning is not limited to the traditional computer security domain, but will grow and expand to other application domains such as online advertisement, social networks, and recommendation system.

Bibliography

[1] R.A. Abrams and D.A. Balota. Mental chronometry: Beyond reaction time. *Psychological Science*, 2(3):153–157, 1991.

[2] H. Alzer. Some beta-function inequalities. *Proc. of Royal Society of Edinburgh: Section A Mathematics*, 133(04):731–745, 2003.

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

[4] D. Angluin. Queries revisited. *Theoretical Computer Science*, 313(2):175–194, 2004.

[5] Keith Ball. Cube slicing in $\mathbb{R}^n$. *Proc. of American Mathematical Society*, 97(3):pp. 465–473, 1986.

[6] D. Barbara and S. Jajodia. *Applications of data mining in computer security*. Springer, 2002.

[7] M. Barreno, B. Nelson, A.D. Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[8] Ron Begleiter, Ran El-Yaniv, and Golan Yona. On prediction using variable order markov models. *J. Artif. Intell. Res. (JAIR)*, 22:385–421, 2004.

[9] Kristin Bennett, Ayhan Demiriz, et al. Semi-supervised support vector machines. In *Proceedings of NIPS*, 1999.

[10] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *JACM*, 51(4):540–556, 2004.

[11] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of SIGKDD*. ACM, 2009.

[12] B. Biggio, B. Nelson, and B. Laskov. Support vector machines under adversarial label noise. In *Proc. of 3rd ACML*, pages 97–112, 2011.

[13] C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[14] David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

[15] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*. ACM, 1998.

[16] A. Bratko, B. Filipič, G.V. Cormack, T.R. Lynam, and B. Zupan. Spam filtering using statistical data compression models. *JMLR*, 7:2673–2698, 2006.

[17] Cristian Brotto, Claudio Gentile, and Fabio Vitale. On higher-order perceptron algorithms. *Advances in Neural Information Processing Systems*, 19, 2007.

[18] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[19] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. *Proceedings of NIPS*, 2001.

[20] Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005.

[21] Nicolo Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective sampling for linear classification. *JMLR*, 7, 2006.

[22] O. Chapelle, B. Schölkopf, A. Zien, et al. *Semi-supervised learning*. MIT Press, 2006.

[23] Olivier Chapelle and S Sathiya Keerthi. Multi-class feature selection with support vector machines. In *Proceedings of the American statistical association*, 2008.

[24] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *STOC*. ACM, 1997.

[25] Abhishek Chaturvedi, Sandeep Bhatkar, and R Sekar. Improving attack detection in host-based ids by learning properties of system call arguments. In *In Proceedings of the IEEE Symposium on Security and Privacy*. Citeseer, 2005.

[26] S. Chen, J. Zhang, G. Chen, and C. Zhang. What if the irresponsible teachers are dominating? In *Proc. 24th AAAI*, 2010.

[27] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996.

[28] T. Chen and J. Ren. Bagging for gaussian process regression. *Neurocomputing*, 72(7):1605–1610, 2009.

[29] Xue-wen Chen, Xiangyan Zeng, and Deborah van Alphen. Multi-class feature selection for texture classification. *Pattern Recognition Letters*, 27(14):1685–1691, 2006.

[30] Wei Chu, Martin Zinkevich, Lihong Li, Achint Thomas, and Belle Tseng. Unbiased online active learning in data streams. In *Proceedings of SIGKDD*. ACM, 2011.

[31] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2), 1994.

[32] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[33] Thomas M Cover and Aaron Shenhar. Compound bayes predictors for sequences with apparent markov structure. *Systems, Man and Cybernetics, IEEE Transactions on*, 7(6):421–424, 1977.

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

[35] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, 2002.

[36] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

[37] Koby Crammer, Mark Dredze Fern, and O Pereira. Exact convex confidence-weighted learning. In *In Advances in Neural Information Processing Systems 22*. Citeseer, 2008.

[38] Koby Crammer, Alex Kulesza, Mark Dredze, et al. Adaptive regularization of weight vectors. *Advances in Neural Information Processing Systems*, 22:414–422, 2009.

[39] Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991, 2003.

[40] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[41] F. Dabbene, PS Shcherbakov, and BT Polyak. A randomized cutting plane method with probabilistic geometric convergence. *SIAM Journal on Optimization*, 20:3185, 2010.

[42] R. Dale, C. Kehoe, and M.J. Spivey. Graded motor responses in the time course of categorizing atypical exemplars. *Memory & Cognition*, 35(1):15–28, 2007.

[43] N. Dalvi, P. Domingos, et al. Adversarial classification. In *Proc. of 10th SIGKDD*, pages 99–108. ACM, 2004.

[44] R. Datta, D. Joshi, J. Li, and J. Wang. Studying aesthetics in photographic images using a computational approach. *Proc. ECCV*, pages 288–301, 2006.

[45] R. Datta, J. Li, and J.Z. Wang. Algorithmic inferencing of aesthetics and emotion in natural images: An exposition. In *Proc. 15th ICIP*, pages 105–108. IEEE, 2008.

[46] Timothy A Davis. *Direct methods for sparse linear systems*, volume 2. Society for Industrial and Applied Mathematics, 2006.

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

[48] O. Dekel and O. Shamir. Learning to classify with missing and corrupted features. In *Proc. of 25th ICML*, pages 216–223, 2008.

[49] O. Dekel and O. Shamir. Good learners for evil teachers. In *Proc. of 26th ICML*, pages 233–240. ACM, 2009.

[50] Ofer Dekel. From online to batch learning with cutoff-averaging. In *Proceedings of NIPS*, 2008.

[51] Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. Individual sequence prediction using memory-efficient context trees. *Information Theory, IEEE Transactions on*, 55(11):5251–5262, 2009.

[52] Ślezak Dominik. Rough sets and functional dependencies in data: Foundations of association reducts. In *Transactions on Computational Science V*, pages 182–205. Springer, 2009.

[53] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM, 2008.

[54] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[55] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *JACM*, 38(1):1–17, 1991.

[56] Elad Eban, Aharon Birnbaum, Shai Shalev-Shwartz, and Amir Globerson. Learning the experts for online sequence prediction. In *ICML*, 2012.

[57] Eleazar Eskin, Wenke Lee, and Salvatore J Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 1, pages 165–175. IEEE, 2001.

[58] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[59] Meir Feder, Neri Merhav, and Michael Gutman. Universal prediction of individual sequences. *Information Theory, IEEE Transactions on*, 38(4):1258–1270, 1992.

[60] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[61] Li Feng, Xiaohong Guan, Sangang Guo, Yan Gao, and Peini Liu. Predicting the intrusion intentions by observing system call sequences. *Computers & Security*, 23(3):241–252, 2004.

[62] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for unix processes. In *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128. IEEE Computer Society Press, 1996.

[63] J.B. Freeman and N. Ambady. Motions of the hand expose the partial and parallel activation of stereotypes. *Psychological Science*, 20(10):1183–1188, 2009.

[64] J.B. Freeman, K. Pauker, E.P. Apfelbaum, and N. Ambady. Continuous dynamics in the real-time perception of race. *Journal of Experimental Social Psychology*, 46(1):179–185, 2010.

[65] Jonathan B Freeman and Nalini Ambady. Mousetracker: Software for studying real-time mental processing using a computer mouse-tracking method. *Behavior Research Methods*, 42(1):226–241, 2010.

[66] Peter Fricke, Felix Jungermann, Katharina Morik, Nico Piatkowski, Olaf Spinczyk, Marco Stolpe, and Jochen Streicher. Towards adjusting mobile devices to user's behaviour. In *Analysis of Social Media and Ubiquitous Data*, pages 99–118. Springer, 2011.

[67] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and information systems*, 2013.

[68] G. Fumera, I. Pillai, and F. Roli. Spam filtering based on the analysis of text information embedded into images. *JMLR*, 7:2699–2720, 2006.

[69] Shen Furao and Osamu Hasegawa. An incremental network for on-line unsupervised classification and topology learning. *Neural Networks*, 19(1), 2006.

[70] Christopher W Geib and Robert P Goldman. Plan recognition in intrusion detection systems. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 1, pages 46–55. IEEE, 2001.

[71] A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *Proc. of 23rd ICML*, pages 353–360. ACM, 2006.

[72] Andrew B Goldberg, Ming Li, and Xiaojin Zhu. Online manifold regularization: A new learning setting and empirical study. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2008.

[73] Andrew B Goldberg, Xiaojin Zhu, Alex Furger, and Jun-Ming Xu. Oasis: Online active semi-supervised learning. In *Proceedings of AAAI*, 2011.

[74] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*. Springer, 2008.

[75] Thomas L Griffiths, Mark Steyvers, David M Blei, and Joshua B Tenenbaum. Integrating topics and syntax. *Advances in neural information processing systems*, 17:537–544, 2005.

[76] B. Grünbaum. *Convex polytopes*, volume 221. Springer, 2003.

[77] Branko Grünbaum. Partitions of mass-distributions and of convex bodies by hyperplanes. *Pac. J. Math.*, 10:1257–1261, 1960.

[78] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of ICML*. ACM, 2005.

[79] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[80] Yoav Haimovitch, Koby Crammer, and Shie Mannor. More is better: Large scale partially-supervised sentiment classification. In *Proceedings of ACML*, 2012.

[81] James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

[82] David P Helmbold and Robert E Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68, 1997.

[83] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines. *The Journal of Machine Learning Research*, 1:245–279, 2001.

[84] Ruizhang Huang and Wai Lam. An active learning framework for semi-supervised document clustering with language modeling. *Data & Knowledge Engineering*, 68(1), 2009.

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

[86] Vijay S Iyengar, Chidanand Apte, and Tong Zhang. Active learning using adaptive resampling. In *Proceedings of SIGKDD*. ACM, 2000.

[87] T Jaakkola and M Jordan. A variational approach to bayesian logistic regression models and their extensions. In *AISTATS*. Citeseer, 1997.

[88] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *JACM*, 51(4):671–697, 2004.

[89] R. Kannan, L. Lovász, and M. Simonovits. Random walks and an o*(n5) volume algorithm for convex bodies. *Random structures and algorithms*, 11(1):1–50, 1997.

[90] Nikos Karampatziakis and Dexter Kozen. Learning prediction suffix trees with winnow. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 489–496. ACM, 2009.

[91] Y. Ke, X. Tang, and F. Jing. The design of high-level features for photo quality assessment. In *Proc. CVPR*, volume 1, pages 419–426. IEEE, 2006.

[92] M. Kearns and M. Li. Learning in the presence of malicious errors. In *Proc. of 20th STOC*, pages 267–280. ACM, 1988.

[93] S. Keerthi and W. Chu. A matching pursuit approach to sparse gaussian process regression. In *Advances in Neural Information Processing Systems*, volume 18, page 643. MIT Press, 2006.

[94] S.S. Keerthi, S. Sundararajan, K.W. Chang, C.J. Hsieh, and C.J. Lin. A sequential dual method for large scale multi-class linear svms. In *Proc. of 14th SIGKDD*, pages 408–416. ACM, 2008.

[95] J.E. Kelley. The cutting-plane method for solving convex programs. *SIAM Journal on Applied Mathematics*, 8(4):703–712, 1960.

[96] G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

[97] A.R. Klivans, P.M. Long, and R.A. Servedio. Learning halfspaces with malicious noise. *JMLR*, 10:2715–2740, 2009.

[98] M.K. Kozlov, S.P. Tarasov, and L.G. Khachiyan. The polynomial solvability of convex quadratic programming. *USSR Comput. Math. and Math. Phys.*, 20(5):223–228, 1980.

[99] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1650. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[100] A. Krause, H.B. Mcmahan, C. Guestrin, and A. Gupta. Selecting observations against adversarial objectives. In *Advances in Neural Information Processing Systems*. MIT Press, 2007.

[101] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[102] Neil D Lawrence and Andrew J Moore. Hierarchical gaussian process latent variable models. In *Proceedings of the 24th international conference on Machine learning*, pages 481–488. ACM, 2007.

[103] Wenke Lee, Salvatore J Stolfo, and Philip K Chan. Learning patterns from unix process execution traces for intrusion detection. In *AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 50–56, 1997.

[104] A.Y. Levin. On an algorithm for the minimization of convex functions. *Soviet Mathematics Doklady*, 160:1244–1247, 1965.

[105] C. Li and T. Chen. Aesthetic visual quality assessment of paintings. *Selected Topics in Signal Processing, IEEE Journal of*, 3(2):236–252, 2009.

[106] C. Li, A. Gallagher, A.C. Loui, and T. Chen. Aesthetic quality assessment of consumer photos with faces. In *Proc. ICIP*, pages 3221–3224, 2010.

[107] D. Lowd and C. Meek. Adversarial learning. In *Proc. of 11th SIGKDD*, pages 641–647. ACM, 2005.

[108] D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *Proc. of 2nd Conference on Email and Anti-Spam*, pages 125–132, 2005.

[109] M.A. Maloof. *Machine learning and data mining for computer security: methods and applications*. Springer, 2006.

[110] Alvaro Martin, Gadiel Seroussi, and Marcelo J. Weinberger. Linear time universal coding and time reversal of tree sources via fsm closure. *Information Theory, IEEE Transactions on*, 50(7):1442–1468, 2004.

[111] Andrew McCallum, Kamal Nigam, et al. Employing EM and pool-based active learning for text classification. In *Proceedings of ICML*, 1998.

[112] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

[113] A. Moorthy, P. Obrador, and N. Oliver. Towards computational models of the visual aesthetic appeal of consumer videos. *Proc. ECCV*, pages 1–14, 2010.

[114] TS Motzkin and IJ Schoenberg. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6(3):393–404, 1954.

[115] Ion Muslea, Steven Minton, and Craig A Knoblock. Active+ semi-supervised learning= robust multi-view learning. In *Proceedings of ICML*, 2002.

[116] B. Nelson, M. Barreno, F.J. Chi, A.D. Joseph, B.I.P. Rubinstein, U. Saini, C. Sutton, JD Tygar, and K. Xia. Exploiting machine learning to subvert your spam filter. In *Proc. of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, page 7, 2008.

[117] Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, Shing hon Lau, Steven Lee, Satish Rao, Anthony Tran, and J. D. Tygar. Near-optimal evasion of convex-inducing classifiers. In *Proc. of 13th AISTATS*, 2010.

[118] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[119] D.J. Newman. Location of the maximum on unimodal surfaces. *JACM*, 12(3):395–398, 1965.

[120] J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *Recent Advances in Intrusion Detection*, pages 81–105. Springer, 2006.

[121] Duy Nguyen-Tuong, Jan R Peters, and Matthias Seeger. Local gaussian process regression for real time online model learning. In *Advances in Neural Information Processing Systems*, pages 1193–1200, 2008.

[122] Albert BJ Novikoff. On convergence proofs for perceptrons. Technical report, DTIC Document, 1963.

[123] Yoshihiro Oyama, Koichi Onoue, and Akinori Yonezawa. Speculative security checks in sandboxing systems. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 8–pp. IEEE, 2005.

[124] Abhinav Pathak, Y Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, pages 153–168. ACM, 2011.

[125] Fernando C Pereira and Yoram Singer. An efficient extension to mixture techniques for prediction and decision trees. *Machine Learning*, 36(3):183–199, 1999.

[126] BT Polyak and PS Shcherbakov. A randomized method for solving semidefinite programs. In *Adaptation and Learning in Control and Signal Processing*, volume 9, pages 227–231, 2007.

[127] J. Quiñonero-Candela and C.E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[128] L. Rademacher and N. Goyal. Learning convex bodies is hard. In *Proc. of 22nd COLT*, pages 303–308, 2009.

[129] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA, 2006.

[130] V.C. Raykar, S. Yu, L.H. Zhao, A. Jerebko, C. Florin, G.H. 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.

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

[132] Herbert Robbins. Asymptotically subminimax solutions of compound statistical decision problems. In *Herbert Robbins Selected Papers*, pages 7–24. Springer, 1985.

[133] R.T. Rockafellar. *Convex analysis*, volume 28. Princeton Univ Pr, 1997.

[134] F Ronsenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65:386–408, 1958.

[135] L.A. Santaló. *Integral geometry and geometric probability*. Cambridge Univ Pr, 2004.

[136] R. Schneider. *Convex bodies: the Brunn-Minkowski theory*. Cambridge Univ Pr, 1993.

[137] M. Seeger, C.K.I. Williams, and N.D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Workshop on AI and Statistics*, volume 9, page 2003, 2003.

[138] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[139] Shai Shalev-shwartz and Yoram Singer. Convex repeated games and fenchel duality. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2006.

[140] Furao Shen, Hui Yu, Keisuke Sakurai, and Osamu Hasegawa. An incremental online semi-supervised active learning algorithm based on self-organizing incremental neural network. *Neural Computing and Applications*, 20(7), 2011.

[141] N.Z. Shor. Cut-off method with space extension in convex programming problems. *Cybernetics and Systems Analysis*, 13(1):94–96, 1977.

[142] Abraham Silberschatz, Peter B Galvin, and Greg Gagne. *Operating system concepts*. J. Wiley & Sons, 2009.

[143] R.L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, pages 1296–1308, 1984.

[144] Alex J. Smola and Peter Bartlett. Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems*, pages 619–625. MIT Press, 2001.

[145] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of venus images. In *Proc. 9th NIPS*, pages 1085–1092, 1995.

[146] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264. MIT press, 2006.

[147] J.H. Song and K. Nakayama. Target selection in visual search as revealed by movement trajectories. *Vision research*, 48(7):853–861, 2008.

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

[149] M.J. Spivey, M. Grosjean, and G. Knoblich. Continuous attraction toward phonological competitors. *Proceedings of the National Academy of Sciences of the United States of America*, 102(29):10393–10398, 2005.

[150] K. Tan, K.S. Killourhy, and R.A. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *Proc. 5th RAID*, pages 54–73. Springer-Verlag, 2002.

[151] K. Tan, J. McHugh, and K. Killourhy. Hiding intrusions: From the abnormal to the normal and beyond. In *Information Hiding*, pages 1–17. Springer, 2003.

[152] Gaurav Tandon and Philip Chan. Learning rules from system call arguments and sequences for anomaly detection. In *ICDM Workshop on Data Mining for Computer Security (DMSEC)*, pages 20–29, 2003.

[153] H. Tang, N. Joshi, and A. Kapoor. Learning a blind measure of perceptual image quality. In *Proc. CVPR*, pages 305–312. IEEE, 2011.

[154] Yee Whye Teh. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics, 2006.

[155] Michael E Tipping and Christopher M Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999.

[156] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2:45–66, 2002.

[157] V. Tresp. A bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

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

[159] Gokhan Tur, Dilek Hakkani-Tür, and Robert E Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2), 2005.

[160] Michal Valko, Branislav Kveton, Huang Ling, Ting Daniel, et al. Online semi-supervised learning on quantized graphs. In *Proceedings of UAI*, 2010.

[161] L. Vicente, G. Savard, and J. Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2):379–399, 1994.

[162] D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *Proc. 9th CCS*, pages 255–264. ACM, 2002.

[163] Jialei Wang, Peilin Zhao, and Steven CH Hoi. Exact soft confidence-weighted learning. *arXiv preprint arXiv:1206.4612*, 2012.

[164] Xuerui Wang, Andrew McCallum, and Xing Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 697–702. IEEE, 2007.

[165] Zheng Wang, Yangqiu Song, and Changshui Zha. Efficient active learning with boosting. In *SDM*, 2009.

[166] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 133–145. IEEE, 1999.

[167] 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.

[168] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context-tree weighting method: Basic properties. *Information Theory, IEEE Transactions on*, 41(3):653–664, 1995.

[169] Virginia Vassilevska Williams. Breaking the Coppersmith-Winograd barrier, 2011.

[170] Frank Wood, Cédric Archambeau, Jan Gasthaus, Lancelot James, and Yee Whye Teh. A stochastic memoizer for sequence data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1129–1136. ACM, 2009.

[171] Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. The sequence memoizer. *Communications of the ACM*, 54(2):91–98, 2011.

[172] O. Wu, W. Hu, and J. Gao. Learning to rank under multiple annotators. In *Proc. 22nd IJCAI*, 2011.

[173] Han Xiao, T. Stibor, and C. Eckert. Evasion attack of multi-class linear classifiers. In *Proc. of 16th PAKDD*, pages 207–218, 2012.

[174] Han Xiao and Thomas Stibor. A supervised topic transition model for detecting malicious system call sequences. In *Proceedings of the 2011 workshop on Knowledge discovery, modeling and simulation*, pages 23–30. ACM, 2011.

[175] Y. Yan, R. Rosales, G. Fung, and J. Dy. Active learning from crowds. In *Proc. 28th ICML*, 2011.

[176] Y. Yan, R. Rosales, G. Fung, M. Schmidt, G. Hermosillo, L. Bogoni, L. Moy, J. Dy, and PA Malvern. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proc. AISTATS*, 2010.

[177] Zhi-Hua Zhou, Ke-Jia Chen, and Yuan Jiang. Exploiting unlabeled data in content-based image retrieval. In *Proceedings of ECML*. Springer, 2004.

[178] Zhi-Hua Zhou and Ming Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3), 2010.

[179] Zhi-Hua Zhou, De-Chuan Zhan, and Qiang Yang. Semi-supervised learning with very few labeled training examples. In *Proceedings of AAAI*, 2007.

[180] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML*, 2003.

[181] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, 2003.

[182] Xingquan Zhu, Peng Zhang, Xiaodong Lin, and Yong Shi. Active learning from data streams. In *Proceedings of ICDM*. IEEE, 2007.

[183] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on*, 24(5):530–536, 1978.