Learning via Query Synthesis

Dissertation by
Ibrahim Mansour Alabdulmohsin

In Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy

King Abdullah University of Science and Technology
Thuwal, Kingdom of Saudi Arabia

April, 2017
The dissertation of Ibrahim Mansour Alabdulmohsin is approved by the examination committee.

Committee Chairperson: Prof. Xiangliang Zhang
External Committee Member: Prof. Wei Wang (University of California, Los Angeles)
Committee Members: Prof. Xin Gao, Prof. David Keyes
Learning via Query Synthesis

Ibrahim Alabdulmohsin

Active learning is a subfield of machine learning that has been successfully used in many applications. One of the main branches of active learning is query synthesis, where the learning agent constructs artificial queries from scratch in order to reveal sensitive information about the underlying decision boundary. It has found applications in areas, such as adversarial reverse engineering, automated science, and computational chemistry. Nevertheless, the existing literature on membership query synthesis has, generally, focused on finite concept classes or toy problems, with a limited extension to real-world applications.

In this thesis, I develop two spectral algorithms for learning halfspaces via query synthesis. The first algorithm is a maximum-determinant convex optimization method while the second algorithm is a Markovian method that relies on Khachiyan’s classical update formulas for solving linear programs. The general theme of these methods is to construct an ellipsoidal approximation of the version space and to synthesize queries, afterward, via spectral decomposition. Moreover, I also describe how these algorithms can be extended to other settings as well, such as pool-based active learning.

Having demonstrated that halfspaces can be learned quite efficiently via query synthesis, the second part of this thesis proposes strategies for mitigating the risk of reverse engineering in adversarial environments. One approach that can be used to render query synthesis algorithms ineffective is to implement a randomized response. In this thesis, I propose a semidefinite program (SDP) for learning a distribution of classifiers, subject to the constraint that any individual classifier picked at random
from this distributions provides reliable predictions with a high probability. This algorithm is, then, justified both theoretically and empirically. A second approach is to use a non-parametric classification method, such as similarity-based classification. In this thesis, I argue that learning via the empirical kernel maps, also commonly referred to as 1-norm Support Vector Machine (SVM) or Linear Programming (LP) SVM, is the best method for handling indefinite similarities. The advantages of this method are established both theoretically and empirically.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examination Committee Page</td>
<td>2</td>
</tr>
<tr>
<td>Copyright</td>
<td>3</td>
</tr>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>List of Abbreviations</td>
<td>10</td>
</tr>
<tr>
<td>List of Figures</td>
<td>12</td>
</tr>
<tr>
<td>List of Tables</td>
<td>15</td>
</tr>
<tr>
<td><strong>I Introduction</strong></td>
<td>16</td>
</tr>
<tr>
<td>1. Overview of the Mathematical Theory of Learning</td>
<td>17</td>
</tr>
<tr>
<td>1.1 The Philosophical Roots of Learning</td>
<td>17</td>
</tr>
<tr>
<td>1.2 Why Learning?</td>
<td>21</td>
</tr>
<tr>
<td>1.3 The Mathematics of Learning</td>
<td>23</td>
</tr>
<tr>
<td>1.3.1 Information Theory</td>
<td>25</td>
</tr>
<tr>
<td>1.3.2 Statistical Learning Theory</td>
<td>29</td>
</tr>
<tr>
<td>1.3.3 A Unifying Approach</td>
<td>36</td>
</tr>
<tr>
<td>1.3.4 Final Remarks</td>
<td>39</td>
</tr>
<tr>
<td>1.4 A Taxonomy of Learning</td>
<td>39</td>
</tr>
<tr>
<td>1.4.1 Supervised and Unsupervised Learning</td>
<td>39</td>
</tr>
<tr>
<td>1.4.2 Passive and Active Learning</td>
<td>41</td>
</tr>
<tr>
<td>1.4.3 Linear and Non-Linear Methods</td>
<td>42</td>
</tr>
<tr>
<td>1.4.4 Adversarial and Non-Adversarial Environments</td>
<td>44</td>
</tr>
<tr>
<td>1.5 Summary</td>
<td>45</td>
</tr>
<tr>
<td><strong>2 Thesis Overview</strong></td>
<td>46</td>
</tr>
<tr>
<td>2.1 Statement of Contributions</td>
<td>46</td>
</tr>
<tr>
<td>2.2 Outline of Thesis</td>
<td>47</td>
</tr>
</tbody>
</table>
## II Query Synthesis Algorithms for Halfspaces

### 3 Efficient Active Learning of Halfspaces via Query Synthesis

#### 3.1 Introduction

#### 3.2 Why Learning Halfspaces using Query Synthesis?

#### 3.3 The Proposed Algorithm

- 3.3.1 Version Space Approximation
- 3.3.2 Efficient Implementation
- 3.3.3 Query Synthesis

#### 3.4 Evaluation

- 3.4.1 Methodology and Results
- 3.4.2 Analytical Argument

#### 3.5 Conclusions

### 4 Trading the Sample Complexity for Space and Time

#### 4.1 Introduction

#### 4.2 The Khachiyan Update Formulas

#### 4.3 The Proposed Algorithm

- 4.3.1 Description

#### 4.4 Experiments

#### 4.5 Analytical Justification

#### 4.6 Conclusions

### 5 Extensions of the Query Synthesis Algorithms

#### 5.1 Introduction

#### 5.2 Non-Homogenous Halfspaces

- 5.2.1 Derivation
- 5.2.2 Experiments

#### 5.3 Pool-Based Active Learning: An Aggressive Approach

- 5.3.1 Description
- 5.3.2 Experiments

#### 5.4 Conclusions
III Mitigating the Adversarial Risk of Reverse Engineering 108

6 Learning a Distribution of Linear Classifiers 109
6.1 Introduction ........................................... 109
6.2 Related Literature .................................. 113
   6.2.1 Causative Attack Strategies ............... 113
   6.2.2 Causative Defense Strategies ............. 114
   6.2.3 Exploratory Attack Strategies ............. 115
   6.2.4 Exploratory Defense Strategies ........... 116
6.3 The Randomization Method ....................... 118
   6.3.1 Derivation ..................................... 119
   6.3.2 Efficient Implementation ................... 123
   6.3.3 Analysis ....................................... 123
6.4 Experiments ......................................... 128
   6.4.1 Evaluation Methodology ..................... 128
   6.4.2 Accuracy-Variance Tradeoff ............... 131
   6.4.3 Reverse Engineering Attacks ............... 132
6.5 Conclusions ......................................... 134

7 Learning via the Empirical Kernel Maps 136
7.1 Introduction ........................................... 136
7.2 Previous Work ....................................... 140
   7.2.1 Kernel Approximation ......................... 140
   7.2.2 Non-Convex Optimization ..................... 141
   7.2.3 Nearest Neighbor Classification ............ 142
   7.2.4 Learning in Krein Spaces ................... 142
7.3 The 1-Norm Support Vector Machine ............ 143
   7.3.1 Description .................................... 144
   7.3.2 Computational Complexity ................... 147
7.4 Analysis ............................................. 148
   7.4.1 Large-Margin Classification ................. 148
   7.4.2 Empirical Kernel Maps ....................... 151
   7.4.3 Nearest Neighbor Classification ............ 153
   7.4.4 Neural Networks ................................ 155
   7.4.5 Structural Risk Minimization ............... 158
7.5 Experiments and Results ......................... 160
7.5.1 Synthetic Datasets ........................................ 160
7.5.2 Real Datasets ............................................. 160
7.5.3 Discussion .................................................. 165
7.6 Conclusions .................................................. 169

IV Conclusion .................................................. 171

8 Conclusion ................................................... 172

References ..................................................... 177

Appendices ...................................................... 193

A The Class of $f$ Divergences .................................. 194
A.1 Fundamental Properties of $f$ Divergences ............... 195
A.2 Axiomatic Treatment ....................................... 197

B Optimization over an Ellipsoid ................................ 201

C Query Synthesis for Halfspaces: MATLAB Code .......... 203

D Query Synthesis for Non-Homogenous Halfspaces: MATLAB Code 205

E Learning a Distribution of Classifiers: MATLAB Code .... 207

F The 1-Norm Support Vector Machine: MATLAB Code .... 209

G Publication List ............................................... 211
G.1 Journal and Conference Papers ............................. 211
G.2 Preprints ..................................................... 212
### LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT</td>
<td>Algorithmic Learning Theory</td>
</tr>
<tr>
<td>CCP</td>
<td>Concave Convex Procedure</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Density Function</td>
</tr>
<tr>
<td>COLT</td>
<td>Computational Learning Theory</td>
</tr>
<tr>
<td>CVX</td>
<td>Matlab Software for Disciplined Convex Programming</td>
</tr>
<tr>
<td>DAP</td>
<td>Direct Attribute Prediction</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time Warping</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation-Maximization</td>
</tr>
<tr>
<td>ERM</td>
<td>Empirical Risk Minimization</td>
</tr>
<tr>
<td>ESVM</td>
<td>Eigen-decomposition Support Vector Machine</td>
</tr>
<tr>
<td>FBMP</td>
<td>Fast Bayesian Matching Pursuit</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>k-NN</td>
<td>$k$ Nearest Neighbor</td>
</tr>
<tr>
<td>KKT</td>
<td>Karush-Kuhn-Tucker</td>
</tr>
<tr>
<td>LCS</td>
<td>Longest Common Subsequence</td>
</tr>
<tr>
<td>LOO</td>
<td>Leave-One-Out</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Programming</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>OCAS</td>
<td>Optimized Cutting Plane Algorithm</td>
</tr>
<tr>
<td>OMP</td>
<td>Orthogonal Matching Pursuit</td>
</tr>
<tr>
<td>PAC</td>
<td>Probably Approximately Correct</td>
</tr>
<tr>
<td>PSD</td>
<td>Positive Semidefinite</td>
</tr>
<tr>
<td>QBC</td>
<td>Query-by-Committee</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectifier Linear Unit</td>
</tr>
<tr>
<td>RHKS</td>
<td>Reproducing Hilbert Kernel Space</td>
</tr>
<tr>
<td>RHS</td>
<td>Right-Hand Side</td>
</tr>
<tr>
<td>RONI</td>
<td>Reject On Negative Impact</td>
</tr>
<tr>
<td>SABMP</td>
<td>Support Agnostic Bayesian Matching Pursuit</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>SDP</td>
<td>Semidefinite Programming</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SMO</td>
<td>Sequential Minimal Optimization</td>
</tr>
<tr>
<td>SOCP</td>
<td>Second-Order Cone Programming</td>
</tr>
<tr>
<td>SRM</td>
<td>Structural Risk Minimization</td>
</tr>
<tr>
<td>SV</td>
<td>Support Vector</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>VC</td>
<td>Vapnik-Chervonenkis</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1.1 The moon illusion refers to the fact that the moon is perceived to be larger than usual when it is near the horizon even though its proper angular size remains unchanged. 19

1.2 An example of a regression problem (left) with two possible fitting curves (middle and right). 29

1.3 Balancing between the risks of under-fitting and over-fitting is an important goal in any learning task. In this figure, we have two classes that are marked in RED and BLUE, and the goal of learning is to find a decision boundary that separates the two classes correctly with a high probability. 30

3.1 Using the bisection search, query learning of thresholds in \( \mathbb{R} \) can yield an exponential improvement in sample complexity over random sampling. 66

3.2 By blindly eliminating “half” of the hypotheses in \( \varepsilon^* \), we may push \( \varepsilon^* \) into a lower-dimensional subspace as depicted at the top, where arrows indicate the chosen direction of the cut. Here, the fact that volume(\( \varepsilon^* \)) \( \rightarrow 0 \) does not imply that \( \varepsilon^* \rightarrow 0 \). Instead, it is imperative to shrink both volume and axis length as depicted at the bottom. 76

3.3 In this figure, the estimation error is plotted in a log-scale against the number of queries for various query synthesis algorithms. The left and right columns correspond to \( k = 1 \) (single-query) and \( k = 5 \) (batch-mode) respectively. The rows correspond to \( d = 25, 50, 75 \) from top to bottom respectively. The † curves are for random sampling. The †† curves are for uncertainty sampling (\( k = 1 \), left) and orthogonal sampling (\( k = 5 \), right). The ◦ curves are for the query-by-bagging method (\( k = 1 \), left) and the representative sampling method (\( k = 5 \), right). Finally, the □ curves are for the new proposed method. 79

4.1 The number of rounds is plotted against estimation error when the problem dimension is \( d = 100 \). In each round, a batch of 50 queries are synthesized. 89
4.2 The total cumulative time is plotted against the number of queries, when a batch of 50 queries are synthesized at each round. Here, the problem dimension is $d = 100$. The new proposed method uses less than 50s to complete 3,000 rounds of the query synthesis algorithm, which yields an estimation error of 1E-8. By contrast, the other two methods take more than 3 hours to reach the same level of accuracy.

4.3 In this figure, the vertical axis is the number of rounds used by the proposed algorithm to achieve an estimation error of 1E-3, where the $x$-axis is the problem dimension $d$. The top plot corresponds to a batch size of 10, whereas the bottom plot corresponds to a batch size of 20.

5.1 In this figure, the estimation error is plotted in a log-scale against the number of queries for the query synthesis algorithm in Algorithm 3. Here, a batch of $k = 5$ queries are synthesized at each round. The ‘◦’ curve is for $d = 25$. The ‘□’ curve is for $d = 50$. Finally, the ‘♦’ curve is for $d = 75$.

5.2 In this figure, the estimation error is plotted in a log-scale against the number of queries for different pool-based active learning algorithms. Here, the problem dimension is $d = 25$ and the pool size at round $t$ is $100t$. The ‘◦’ curve is for random sampling (baseline method). The ‘□’ curve is for margin-based sampling. Finally, the ‘♦’ curve is for covariance-based sampling criteria proposed in Section 5.3.1.

6.1 The existing literature on adversarial learning can be classified into four quadrants.

6.2 In this figure, the blue line marks the classifier’s true decision boundary whereas the red line is the adversary’s estimate using the query synthesis algorithm of Chapter 3. The highlighted points mark the queries used by the algorithm.

6.3 In the top figures, the 80% and 90% confidence regions for the decision boundary are highlighted when applying the randomized SVM formulation in (6.4) to two synthetic datasets. In the bottom figures, the tradeoff curves between accuracy and variance are plotted where a classifier can operate anywhere within the blue-shaded region. The red circles mark the Pareto optimal points that achieve the maximum accuracy.
6.4 The tradeoff curves between spread $\chi$ and predictive accuracy for the 15 UCI datasets listed in Table 6.1, presented from left to right, top to bottom. The boundary of the blue-shaded region forms the set of Pareto optimal points.

6.5 Results of conducting the reverse engineering experiment in Section 6.4.3. In this figure, the red curve is estimation error when using a fixed classifier, the black curve is estimation error when using confidence-based randomization CONF-RAND, while the blue curve is estimation error when using a distribution of classifiers CLASS-RAND. The $x$-axis is the number of probes, while the $y$-axis is estimation error in a log-scale. The figures correspond to the datasets in Table 6.1 ordered left-to-right and top-to-bottom.

7.1 In this figure, two classes are shown in red and blue. A solid marker is a support vector, whose size is proportional to its weight $\lambda_i$. Classification regions for the red and blue classes are shown in yellow and green, respectively.

7.2 The 1-norm SVM method can be interpreted as a method of function approximation using two-layer neural networks. In this figure, $S_i = S(\cdot, x_i) : \mathcal{X} \to \mathbb{R}$ is the similarity function with the training instance $x_i$. The activation threshold at the output node is the bias term $\lambda_0$.

7.3 The six synthetic datasets that are used in evaluating 1-norm SVM. The Bayes risk of each dataset is indicated in the legend bar in Fig. 7.4.

7.4 Performance of 1-norm SVM on the six synthetic datasets. Each error rate, plotted in a log-scale, is an average of five i.i.d training/test sets. $P_1, \ldots, P_6$ stands for Problem 1, \ldots, Problem 6 shown in Fig. 7.3. The optimal Bayes risks are indicated in the legends bar. In the $y$-axis, each grid line between $10^z$ and $10^{z+1}$ is to be read as $1 \times 10^z$, $2 \times 10^z$, $\ldots$, $9 \times 10^z$. For example, the grid lines from $10^1$ to $10^2$ correspond to the values 10, 20, \ldots, 80, 90.
### LIST OF TABLES

6.1 Results of applying linear SVM (FIXED), CONF-RAND [1], and the proposed CLASS-RAND methods to 15 datasets. Here, \( n \) is the number of attributes while \( m \) is the number of examples in each dataset. Columns 2-4 list the results of the classification error rate of each method, while column 5 lists the spread of the distribution \( \chi \) that is used in the experiment. ............................................................... 133

7.1 Average test error rate results on 16 datasets using the positive semidefinite (PSD) similarity functions described in Section 7.5.2. Here, the number of training examples \( m \) for each dataset is shown in parentheses. .................................................... 166

7.2 Average test error rate results on 16 datasets using the indefinite (non-PSD) similarity functions described in Section 7.5.2. In this table, \( 0 \leq \beta \leq 1 \) is a measure of how indefinite the similarity function is. In particular, a value of \( \beta = 0 \) corresponds to PSD similarity functions while a value of \( \beta = 1 \) corresponds to negative semidefinite similarities. The rows in the table are ordered by the value of \( \beta \). The acronym NCO stands for non-convex optimization, DEN for the denoise method, FLIP for the flip method, ISVM for indefinite SVM, while ESVM stands for the eigen-decomposition SVM method. .................................................... 167

7.3 The number of support vectors SV used by 1-norm SVM and Eigen-decomposition Support Vector Machine (ESVM) for the 16 classification problems with indefinite similarity functions. ............................... 169

7.4 In this table, the second column lists the \( p \) values in increasing order of the two-tailed Wilcoxon’s signed rank test. The last column shows the critical values when Holm’s step-down procedure is used at the 95% confidence level. ................................................................. 170
Part I

Introduction
Chapter 1

Overview of the Mathematical Theory of Learning

1.1 The Philosophical Roots of Learning

For millennia, the term “natural philosophy” was the common term used to describe the practice of studying nature. This includes the study of space, time, motion, causation, heavenly objects, animals, and the study of us humans. Natural philosophy can be dated to more than 2,500 years ago, with various contributions made by influential scholars from all over the world. It had eventually led to the development of modern science.

Not surprisingly, many scientific achievements of the modern age trace their roots to ancient philosophy. The Greek “laughing philosopher” Democritus, for instance, developed a materialistic account that is based on the concept of the indivisibles. His philosophical work is rightfully considered a precursor to the modern theory of the atoms. Also, Pythagoreanism, which was an active Greek school of philosophy in the fourth century BCE, advocated a mathematical interpretation of nature. Their prominent scholar Archytas (ca. 420-ca. 350 BCE) firmly believed that the world could be described in mathematical proportions, a thought that was later revived through the ground-breaking works of Galileo, Copernicus, and Newton, among others [2, 3]. Similarly, Boolean logic, the powerhouse behind modern computers, is traced to Aristotle’s syllogism [4].

In addition to the laws of nature, philosophers also investigated the human mind. They pondered over issues such as: How do humans learn? What are the sources of
knowledge? And can beliefs ever be justified objectively? These questions and others formed a major branch of philosophy called epistemology. So, epistemology is, in concise terms, the philosophical theory of knowledge and learning [5].

In the eleventh century AC, a significant contribution to epistemology and, in turn, to our understanding of knowledge and learning was put forward by the great Arab scholar Ibn Al-Haytham (Alhazen) in his celebrated book *Optics*. In *Optics*, Ibn Al-Haytham argued that perception was, in fact, inferential. That is, instead of perceiving nature as it really is, we perceive nature by interpreting its stimuli [6]. Hence, the human mind could lead us astray through its erroneous interpretations. He, then, proceeded to show that many puzzling phenomena in nature were not quite mysterious after all because they were not ascribable to nature in the first place. Rather, they were phenomena of the human mind itself.

To illustrate his claims, Ibn Al-Haytham used the moon illusion as an example. The moon illusion, depicted in Fig. 1.1 is a famous paradox that has been recorded in many cultures across the globe. It refers to the fact that the moon appears near the horizon to be larger than its ordinary size, even though the angular size of its apparent disc, when properly measured, does not actually change [7]. Ibn Al-Haytham argued that the mind interpreted the visual stimuli of the moon near the horizon differently. He also argued that such an intermediary step of interpretation occurred in our minds too rapidly for us to be consciously aware of it. Several arguments were later proposed to explain how this process works. For instance, an old explanation stated that visual cues near the horizon may lead the brain to overestimate the distance to the moon, which, in turn, impacted how the brain interpreted the size of the moon [8].

Ibn Al-Haytham’s insight was quite profound at the time and, later, played a key role in the rise of medieval skepticism [6]. However, such a profound concept is taken for granted nowadays after countless other examples of optical illusions have been discovered and popularized. The list includes, but is not limited to, the “Hermann
grid” illusion discovered in 1870 by the physiologist L Hermann and the “parallel line” illusion discovered in 1860 by the astrophysicist J Zöllner [9]. More recently, a picture of a dress, associated with the hashtag #thedress, became a viral meme that revealed dramatic differences in human color perception [10]. Indeed, instead of perceiving the world as it really is, we perceive how our minds interpret the world around us!

We can describe Ibn Al-Haytham’s model of how the brain works in precise mathematical terms. Remarkably, this will turn out to be quite relevant to the general setting of learning, popularized by V Vapnik and colleagues, which forms the foundation of modern statistical learning theory [11, 12]. According to Ibn Al-Haytham, our (human) brain receives data $z \in \mathcal{Z}$ through our sensory systems where, for the purpose of generality, we only assume that $z$ resides in some observation space $\mathcal{Z}$. This data is, then, summarized or mapped into a higher-level representation $f(z) \in \mathcal{F}$ via some mapping $f : \mathcal{Z} \rightarrow \mathcal{F}$. After that, and rather than acting upon the raw data $z$ itself, we act according to our interpretation $f(z) \in \mathcal{F}$. Hence, our perception is conditionally independent of the our sensory stimuli given the brain’s interpretation. Such an intermediary step of interpretation is advantageous because it relieves our consciousness of the burden of processing our entire sensory stimuli.

To describe this model in more concrete terms, suppose $z$ is a retinal image that contains an object whereas $f(z)$ is the brain’s mechanism of estimating the distance
to the object. Note that \( f(z) \) summarizes the entire observation \( z \) into a scalar \( f(z) \in \mathbb{R}^+ \). Then, instead of perceiving the actual distance to the object, we perceive the distance suggested by \( f(z) \). Because we perceive the distance given by \( f(z) \), which can be inaccurate, phenomena such as the moon illusion may arise [8].

This particular model raises the following fundamental question: if the higher-level abstractions \( f(\cdot) \) can be inaccurate, how did the human brain obtain them in the first place? Are they genetically coded or are they learned from experience? While evidence suggests that many core functions of the brain are genetically coded, many experiments in psychology indicate that humans and animals alike also learn from observations. In particular, they learn through trial-and-error. For example, the visual cliff experiment suggests that infants learn to perceive depth with experience [13]. Similarly, a famous experiment by the American psychology Edward Thorndike showed that cats learn to escape puzzle boxes by experimenting rather than by thinking or reasoning. In addition, the cat experiment showed that animals also generalize by extending what they have already learned to new settings [14]. This view is also supported by independent works in neuroscience, which revealed that dopamine levels in the brain act as teaching signals; they increase when the outcome of an action is better than expected, and they decrease when the outcome is worse [15].

Consequently, learning from observations is not merely of a philosophical interest, but it also forms a core component of our cognition. Today, it plays an additional key role in science and technology. This raises the following important questions: Are there any universal laws that hold for all learning tasks? Are there fundamental limits to learning, which are analogous to the fundamental limits imposed by physical laws, such as the constancy of the speed of light? Can learning agents, such as us humans, improve their learning by designing their experience? Are there any implications of this in adversarial environments? Some of these questions and others will form the main subject of this thesis.
1.2 Why Learning?

Learning is a crucial component of what makes us intelligent \[16, 17\]. It allows us to arrive at complex conclusions about our environment using only our sensory stimuli, without having to be consciously aware of the underlying physical processes that make this happen. Through learning, children develop impressive cognitive skills at a very young age. For instance, there exists ample evidence indicating that children do understand the concept of causality, form prototypes, know of object permanence, and infer depth perception during infancy. By the age of five, children can take different perspectives, form strategies, and develop a theory of the mind \[18\]. Hence, in order to appreciate our cognition and what makes us intelligent, it is imperative that we understand what “learning” fundamentally entails.

Learning, for example, is not always successful. There are known accounts of animals that fail to learn. This includes rats that fail to associate specific types of food with electric shocks, and pigeons that “learn” to associate food with places even though the actual relationship between them is purely random \[19\]. Needless to mention, humans also fail to learn on several occasions. They develop superstitions such as \textit{Paraskevidekatriaphobia}, i.e. fear of “Friday the 13th”, which is considered to be the most widespread superstition in the United States \[20\]. It is estimated that the latter fear leads to about $1$ billion of lost revenues because many people refuse to conduct their normal daily operations during this day \[21\]. Superstition is, undoubtedly, a failure of learning. In modern terms, we say that it is an example of over-fitting. At the other end of the spectrum lies dogmatism, where new evidence is entirely neglected in favor of prior beliefs. A classical example in this regard was the war against heliocentrism during the Enlightenment despite the overwhelming empirical evidence for a heliocentric model of the universe. Dogmatism is also a failure of learning. In modern terms, we say that it is an example of under-fitting.

As the previous examples illustrate, it is imperative that we understand when
learning is likely to succeed and when it will inevitably fail. This question is not only of a philosophical essence, although it traces its roots to philosophy \[22, 17\], but it also has many practical applications. Today, many successful computer tools are built that learn from data including inverse geosounding \[23\], texture classification \[24\], image retrieval \[25\], medical prognosis \[26\], high-energy physics \[27\], spam filtering \[28, 29, 30, 31\], fraud detection \[32\], intrusion detection \[33, 34, 35\], text categorization \[36\], recommendation systems \[37\], and object detection \[38\]. These machine learning algorithms have been used to generate revenues, mitigate risks, and to combat crimes \[37, 39, 40\].

To summarize, learning is an old topic that has its roots in philosophy. It has recently outgrown its original questions and is currently of great practical value for nearly every conceivable field of research. As a result, it is imperative that we understand when and how learning succeeds. Most importantly, rigorous mathematical methods and treatments are needed for investigating these questions in order to draw unbiased conclusions. But can mathematics shed some insight into a phenomenon that is as complex as learning?

\[1\] Indeed, machine learning is enormously successful. However, to see why it has been wholeheartedly adopted in the industry, it is insightful to digress and examine how the notion of “solutions” had evolved over time. For hundreds of years, the conventional practice was to consider a problem “solvable” only if an analytical solution existed; that is, a provable solution was available in a closed-form. In regression and fitting, for instance, the least-squares problem was considered solvable but not the similar (more robust) 1-norm minimization problem. Also, finding the roots of a fourth-degree polynomial was considered solvable but not the roots of a fifth-degree polynomial. All of this, however, was forever changed in the second half of the twentieth century when computers came into the picture and algorithms became a substitute for closed-form expressions. Instead of thinking of solutions in terms of formulas that can be examined and manipulated by mathematicians, we started to think of solutions in terms of step-by-step execution with time and memory constraints! Moreover, and almost simultaneously, the notion of “solution” was even extended much further to include problems for which an algorithm could not even be written down explicitly but could be learned using sufficient data. This machine learning approach has allowed us to solve difficult tasks such as face and voice recognition, intrusion detection, and forecasting. Hence, machine learning can be thought of as the most recent development in a long history of the evolution of our tools for solving problems.
1.3 The Mathematics of Learning

For more than two thousand years, the question of learning, information, and intelligence belonged to the realm of philosophy. It was believed that these complex phenomena could not be investigated scientifically without losing their core essence. Naturally, it seemed reasonable to question whether humans could ever learn to study the phenomenon of “learning” objectively. Nevertheless, three major developments took place in the 1940s that challenged this view and, ultimately, brought learning and intelligence into the realm of science and mathematics. All of these events were hugely influential at the time and have continued to inspire new areas of research to this day.

The first significant event took place in 1943 when the neuroscientist W McCulloch and the logician W Pitts proposed the perceptron as a mathematical model for the biological neuron. According to this model, neurons in our brains were elementary computational devices that could carry out any logical function [41]. More specifically, a perceptron received signals through its inputs and fired a signal through its output if and only if a certain threshold $b \in \mathbb{R}$ was met. Formally, if $x_1, \ldots, x_n \in \mathbb{R}$ were the input signals to the perceptron, then its output $y$ was given by:

$$y = f\left(\sum_{i=1}^{n} x_i - b\right),$$

for some monotone non-decreasing function $f : \mathbb{R} \to \mathbb{R}$, such as the sigmoid function, the hyperbolic tangent function, or the Rectifier Linear Unit (ReLU). It is simple to show that this model can implement the AND and OR logic gates, when $f(u) = \mathbb{1}\{u \geq 0\}$. Adding inhibitory neurons, which are excited by default and inhibited by input, will also implement the NOT logic gate. Hence, the perceptron can indeed implement any Boolean function. This model was hugely influential and has led to the development of artificial neural networks, which is one of the most successful
machine learning algorithms in practice today [42, 43, 19, 44].

The second significant event took place in 1946 when the physicist R Cox published his celebrated theorem [45]. According to Cox’s theorem, probability theory is the only numerical system of quantifying and updating one’s degrees of belief that is mathematically consistent and agrees with common sense. In particular, rational deductive reasoning had to follow the laws of probability. This important theorem was later used by prominent probability theorists such as E Jaynes as a cornerstone for the Bayesian interpretation of probability theory and its many branches including Bayesian epistemology, Bayesian statistics, Bayesian decision theory, Bayesian learning theory, and Bayesian confirmation theory [4, 46]. So, whereas the work of McCulloch and Pitts suggested that human intelligence could be analyzed mathematically, Cox’s theorem suggested that probability theory was the right tool to achieve this.

The third significant event affirmed the role of mathematics, in general, and probability theory, in particular, in analyzing the phenomena of learning and intelligence. It took place in 1948 when C Shannon published his famous paper that introduced single-handedly the fruitful field of information theory [47, 48]. In his work, Shannon showed that information could be quantified, and that fundamental limits in information existed that were analogous to physical limits. This included, most famously, the compression limit theorem and the channel capacity theorem. Information theory was later broadened to include fertile fields such as coding theory, cryptography, portfolio theory, and pattern recognition [47].

All of the aforementioned developments inspired researchers to mathematically formalize the complex phenomena of learning and intelligence. Because learning from

\[ P(A = \text{true} | B = \text{false}) = P(A = \text{true}) \cdot \frac{P(B = \text{false} | A = \text{true})}{P(B = \text{false})}, \]

where the right-hand side is zero when B is false because \[ P(B = \text{false} | A = \text{true}) = 0 \] by definition of \( A \rightarrow B. \)
observations was identified early to be a pre-requisite for intelligence \cite{16}, learning from observations received a more rigorous mathematical investigation and had developed tremendously over the past 50 years. New active areas of research were developed such as Algorithmic Learning Theory (ALT), which uses the theory of computation as a building block for learning without assuming any statistical structure, and Computational Learning Theory (COLT), which was initiated by L Valiant to investigate the computational complexity of learning algorithms \cite{49}. Of a particular interest to us in this thesis is the field of statistical learning theory, which was initiated by the Russian mathematicians V Vapnik and A Chervonenkis in a series of papers published in the 1960s and the 1970s \cite{11}.

Both learning and information theory are intimately connected. The relationship between the two fields can be described in very brief terms by saying that information theory is the appropriate tool for analyzing active learning, where the agent is permitted to ask questions, while statistical learning theory is the appropriate tool for analyzing passive learning, where the agent receives and learns from observations passively. The differences between the two learning paradigms will be explored in a greater depth later in Section 1.4. However, because information theory and statistical learning theory form the foundational tools for analyzing those learning paradigms, we review the central results of information theory and statistical learning theory first.

1.3.1 Information Theory

As mentioned earlier, the field of information theory was introduced single-handedly by C Shannon in his celebrated 1948 paper \cite{48}. It originally addressed the limits of data compression, transmission, and signal processing. However, due to the fundamental importance of its results, it soon grew to be an ubiquitous field with applications covering a wide spectrum, including coding theory, cryptography, portfolio
theory, gambling, large deviation theory, statistics, and pattern recognition [47].

Compression Limit

The first fundamental question addressed in information theory is the question of the compression limit. Given a discrete hypothesis space $\mathcal{H}$ that is endowed with a probability mass function $P(h)$, the compression limit question is the following:

How many bits are needed, on average, to describe (encode) a hypothesis $h^* \in \mathcal{H}$ that is picked at random with probability $P(h)$ such that it can always be identified (decoded) correctly?

In the language of learning, this question can be re-phrased as follows. Suppose that a learning agent $L$ faces many learning tasks $T_i$ for $i \in \{1, 2, 3, \ldots \}$. In a learning task $T_i$, there exists an optimal answer that the learning agent $L$ needs to find, which we denote by $h_i \in \mathcal{H}$. This optimal answer resides in some space $\mathcal{H}$ that the agent $L$ knows in advance. Also, suppose that the learning agent knows a priori that $h^* = h_i$ with probability $P(h_i)$, where the probability is evaluated over the random choice of the task $T_i$. Then, the compression limit defined above is equivalent to the minimum number of Yes/No questions that the learning agent needs to ask, on average, before the agent can unambiguously find the optimal answers $h^*$.

The compression limit was proved by C Shannon in his original paper [48]. In his paper, Shannon showed that the compression limit of a discrete random variable $h \sim P(h)$ is given by the entropy $\mathbb{H}(h)$, where:

$$\mathbb{H}(h) = - \sum_{h \in \mathcal{H}} P(h) \log P(h) \quad (1.2)$$

For example, if $P(h) = \frac{1}{|\mathcal{H}|}$ is uniform over the hypothesis space $\mathcal{H}$, then the minimum number of Yes/No questions needed is $\log |\mathcal{H}|$. By contrast, if $P(h)$ is a Kronecker delta distribution, where the optimal hypothesis $h_i$ is always the same and the agent knows the correct answer in advance, then $\mathbb{H}(h) = 0$ because the agent $L$ does not
need to ask any questions to find the optimal answer in the latter case.

Using the law of large numbers, it is easy to see in hindsight why the Shannon entropy \( H \) is indeed equal to the compression limit. Suppose we have \( m \) observations of a discrete random variable \( x \in \mathcal{X} \) that are drawn i.i.d. from some probability mass function \( P(x) \). Then, by the law of large numbers, the fraction of times an element \( x \in \mathcal{X} \) appears in the sequence will converge almost surely to its true probability \( P(x) \) as \( m \to \infty \). In other words, the probability of the entire sequence of \( m \) observations satisfies\(^3\)

\[
\mathbb{P}(x_1, x_2, \ldots, x_m) \frac{1}{m} \to \prod_{x \in \mathcal{X}} P(x)^{P(x)}
\]

However, this implies that there exists a collection of typical sequences whose total probability is overwhelmingly close to one, where all sequences in this typical set are equally probable. Hence, the compression limit must be equal to the logarithm of the volume of the typical set, which converges to the Shannon entropy as \( m \to \infty \) \[^{47}]\). This line of reasoning shows that the Shannon entropy is not only a lower bound to the compression limit, but, also, that one can always devise a coding scheme whose average description length is made arbitrarily close to the compression limit using, for example, the notion of typicality mentioned above. We will return to the Shannon compression limit later, when we analyze the optimality of a query synthesis algorithm for learning halfspaces in \( \mathbb{R}^d \).

**Mutual Information**

The second fundamental concept in information theory is the notion of mutual information. Given two random variables \( x \sim P(x) \) and \( y \sim P(y) \), the mutual

\(^3\)Here, by convergence, we mean a convergence in probability. A random variable \( y \) converges in probability to a value \( b \) as \( m \to \infty \) if \( \mathbb{P}(|y - b| > \epsilon) \to 0 \) as \( m \to \infty \) for any \( \epsilon > 0 \) \[^{47}][^{50}]\). This type of convergence is often referred to as almost-sure convergence, abbreviated a.s. In this particular example, the random variable is the \( m \)-th root of the probability of the entire sequence.
information $I(x; y)$ is defined by:

$$I(x; y) = \mathbb{H}(x) - \mathbb{H}(x | y) = \mathbb{H}(y) - \mathbb{H}(y | x),$$

where $\mathbb{H}(x|y)$ is the conditional entropy of $x$ given an instance of $y$, averaged over all possible instances of $y$. Evidently, mutual information corresponds to the average reduction in the description length of $x$, i.e. the number of Yes/No questions needed, when the value of $y$ is known beforehand. Not surprisingly, $I(x; y) = 0$ if and only if $x$ and $y$ are independent of each other.

Mutual information provides the answer to a second fundamental question in information theory, namely the channel capacity limit. While the question of channel capacity does not, at first sight, bear any relation to learning and inference, it turns out that mutual information is also intimately related to learning. It has found important applications in machine learning, such as in feature selection and clustering [51]. Perhaps more importantly, it is an instance of the broad class of $f$-divergences, which can be motivated axiomatically [52, 53]. In Appendix A for instance, I give my own version of an axiomatic treatment of the class of $f$ divergences, which is closely similar to the axioms proposed by I Csiszár, but with simpler proofs. One member of the class of $f$ divergences is the total variation distance between the joint distribution $P(x, y)$ and the product of marginals $P(x) \cdot P(y)$, which, I proved in a recent work, to be a tight upper bound on the generalization risk of learning algorithms [54]. Through this connection, the Shannon mutual information $I(x; y)$ can be used to analyze the performance of machine learning algorithms. We will return to this statement shortly.
1.3.2 Statistical Learning Theory

The second key field of study, when analyzing machine learning algorithms, is statistical learning theory. Before I describe the central results of statistical learning theory, let us first examine what we would expect from any useful formal treatment of learning. To do this, we will consider the following two examples.

**Example 1 (Occam razor).** Suppose that a learning algorithm (learning agent) $\mathcal{L}$ receives i.i.d. observations $z_i \in \mathcal{Z}$, where each observation is an instance-label pair $z_i = (x_i, y_i) \in \mathbb{R} \times \mathbb{R}$. For simplicity, we will restrict attention to the one-dimensional case. Suppose also that the task of $\mathcal{L}$ is to predict the value of $y$ given the instance $x$. This, for example, is the case in a regression problem, which is illustrated in Fig. 1.2.

In Fig. 1.2 (left), the observations (training set) are plotted. These observations can be perfectly fitted with the two curves shown on the middle and the right of the figure. Having two options to choose from, we would prefer to use the figure on the middle since it is significantly simpler. Informally speaking, we know that the curve on the right contains more information than is available in the observations themselves. This principle of picking the simplest explanation is commonly known as the Occam razor principle. Because this principle is quite important, we expect any useful mathematical theory of learning to agree with it and to place it on a rigorous footing.

**Example 2 (Risk Decomposition).** The second example is illustrated in Fig. 1.3. In this figure, our observations are also instance-label pairs $z_i = (x_i, y_i) \in \mathbb{R}^2 \times \{-1, +1\}$.
but the labels $y_i$ are binary, which implies that this is a classification problem. Unlike the previous example, the options (hypotheses) we have in this problem do not necessarily fit all of the observations perfectly. In this case, we face the risk of under-fitting, as illustrated in the figure (left). However, trying to fit all of the observations perfectly is not always a good strategy because it can lead to over-fitting. This is depicted in the figure (right), where the decision boundary is much more complex than the decision boundary on the left just to be consistent with a single outlier. The two risks of under-fitting and over-fitting are universal in all learning tasks so we expect any useful mathematical theory of learning to formalize this decomposition and to allow us to design learning algorithms that can balance between the two risks.

One way to capture the above goals is to look into the generalization risk of learning algorithms. Before I describe what this means, we first formalize the general setting of learning. The following definition is due to Vapnik [11].

**Definition 1** (General Setting of Learning). In the general setting of learning, we have an observation space $\mathcal{Z}$ and a hypothesis space $\mathcal{H}$. A learning algorithm $\mathcal{L} : \cup_{m=1}^{\infty} \mathcal{Z}^m \rightarrow \mathcal{H}$, uses a finite set of observations to infer a hypothesis $h \in \mathcal{H}$. The risk (error rate) of the hypothesis $h \in \mathcal{H}$ is evaluated by a non-negative loss function $L(\cdot; h) : \mathcal{Z} \rightarrow \mathbb{R}^+$.\footnote{The loss function is, in fact, a function of both $z$ and $h$. Hence, its domain is the product space $\mathcal{Z} \times \mathcal{H}$. However, because the hypothesis $h$ is fixed once it is learned, it is more convenient to think of the loss function $L$ as a function of observations $z$ that is parameterized (indexed) by $h$. Note that $L$ may be randomized, in general.}
The general setting of learning encompasses many traditional learning tasks. We illustrate it with the following example.

**Example 3 (Mean Estimation).** Suppose that the observations \( z \) belong to a field \( \mathcal{Z} \), which is endowed with a probability distribution \( \mathbb{P}(z) \). Also, suppose that a learning algorithm uses a sample of \( m \) i.i.d. observations \( S_m = \{z_1, \ldots, z_m\} \in \mathcal{Z}^m \) to estimate the mean of the distribution \( \mathbb{P}(z) \). In this case, the hypothesis space \( \mathcal{H} \) is itself \( \mathcal{Z} \), and a good learning algorithm \( \mathcal{L} \) may simply use the sample average \( h = (1/m) \sum_{i=1}^{m} z_i \). If \( |z_i| \) is bounded almost surely, concentration of measure inequalities, such as the Hoeffding bound, guarantee that \( h \) will converge to the true mean very quickly (in probability) as \( m \to \infty \). In this setting, a suitable parametric loss function \( L(\cdot; h) : \mathcal{Z} \to \mathbb{R}^+ \) would be any appropriate distance \( L(z; h) = |z - h| \).

In an analogous manner, traditional machine learning tasks such as classification, regression, density estimation, and clustering can all be readily shown to fall under the general setting of learning.

Next, we define the two important concepts of the true and the empirical risks of the learning algorithm \( \mathcal{L} \).

**Definition 2 (True Risk).** The true risk of a learning algorithm \( \mathcal{L} \) is the expected true risk of its inferred hypothesis. More precisely, let \( S_m \in \mathcal{Z}^m \) be a collection of i.i.d training (empirical) observations, which are used by \( \mathcal{L} \) to infer a hypothesis \( h \in \mathcal{H} \). Then, the true risk of a hypothesis \( h \) with respect to the loss function \( L(\cdot; h) : \mathcal{Z} \to \mathbb{R}^+ \) is:

\[
R_{\text{true}}(h) = \mathbb{E}_{z \sim \mathbb{P}(z)} L(z; h)
\]

\[\text{(1.3)}\]

---

\(^5\)Strictly speaking, we assume that the space of observations is measurable, which means that it is equipped with a \( \sigma \)-algebra and a probability measure \( \mathbb{P}(z) \). Also, we assume that all random variables are measurable functions of the sample space \( \mathcal{Z} \). However, to simplify discussion and because these assumptions are standard, we omit them in the discussion.
In addition, the true risk of the learning algorithm \( \mathcal{L} \) is given by:

\[
\hat{R}_{\text{true}}(\mathcal{L}) = \mathbb{E}_{S_m} \mathbb{E}_{h|S_m} R_{\text{true}}(h)
\]

**Definition 3** (Empirical Risk). The empirical risk of a learning algorithm \( \mathcal{L} \) is the expected empirical risk of its inferred hypothesis. More precisely, let \( S_m \in \mathcal{Z}^m \) be a collection of i.i.d training (empirical) observations, which are used by \( \mathcal{L} \) to infer a hypothesis \( h \in \mathcal{H} \). Then, the empirical risk of \( h \) with respect to the loss function \( L(\cdot; h) : \mathcal{Z} \rightarrow \mathbb{R}^+ \) is:

\[
R_{\text{emp}}(h; S_m) = \mathbb{E}_{z \sim S_m} L(z; h) = \frac{1}{m} \sum_{z_i \in S_m} L(z_i; h) \quad (1.4)
\]

In addition, the empirical risk of the learning algorithm \( \mathcal{L} \) is given by:

\[
\hat{R}_{\text{emp}}(\mathcal{L}) = \mathbb{E}_{S_m} \mathbb{E}_{h|S_m} R_{\text{emp}}(h; S_m)
\]

The key problem that we observe is that the performance of a learning algorithm \( \mathcal{L} \) is given by its true risk, which is difficult to estimate. On the other hand, the empirical risk is easy to estimate but it is a biased estimator to the true risk that we care about. Therefore, we need to examine the generalization risk, which is defined by the difference:

\[
\hat{R}_{\text{gen}}(\mathcal{L}) = \left| \hat{R}_{\text{true}}(\mathcal{L}) - \hat{R}_{\text{emp}}(\mathcal{L}) \right|
\]

In particular, we would like to derive analytical upper bounds on the generalization risk of a learning algorithm given that we know how it operates. This is one approach of formalizing the phenomenon of learning.

In order to see why it is indeed fruitful to formalize a complex phenomenon such as learning by looking into the question of how to bound the generalization risk, we make note of the following points:
1. We know by definition of $\hat{R}_{\text{gen}}(\mathcal{L})$ that the true risk is bounded from above by:

$$\hat{R}_{\text{true}}(\mathcal{L}) \leq \hat{R}_{\text{emp}}(\mathcal{L}) + \hat{R}_{\text{gen}}(\mathcal{L})$$

The first term on the right-hand side (RHS) is the empirical risk, which is small only if the learning algorithm can fit its training set (empirical observations) sufficiently well. In other words, $\hat{R}_{\text{emp}}(\mathcal{L})$ quantifies the risk of under-fitting. On the other hand, the generalization risk $\hat{R}_{\text{gen}}(\mathcal{L})$ is small only if the performance of $\mathcal{L}$ on its finite training set is a faithful approximation to its true performance. Hence, $\hat{R}_{\text{gen}}(\mathcal{L})$ quantifies the risk of over-fitting. Therefore, the above inequality states in qualitative terms that:

$$\text{true risk} \leq \text{risk of under-fitting} + \text{risk of over-fitting},$$

which corresponds well with the risk-decomposition phenomenon that is universal in all learning tasks, as mentioned earlier in Example 2.

2. Given two learning algorithms $\mathcal{L}_1$ and $\mathcal{L}_2$, the risk decomposition given by $\hat{R}_{\text{emp}}(\mathcal{L}) + \hat{R}_{\text{gen}}(\mathcal{L})$ provides us with a rigorous basis for preferring one learning algorithm over the other even if the two learning algorithms perform equally well empirically. In particular, if two learning algorithms perform equally well empirically, the learning algorithm with the smallest generalization risk is preferred. This is one formal statement of the Occam razor principle that was described earlier in Example 1. In general, simpler learning algorithms tend to have a smaller generalization risk.

3. The true risk of a learning algorithm $\hat{R}_{\text{true}}(\mathcal{L})$ is often quite difficult to estimate. By contrast, the empirical risk $\hat{R}_{\text{emp}}(\mathcal{L})$ has an unbiased estimator, namely the true performance of the inferred hypothesis $\mathbf{h}$ on its training set, while the
generalization risk $\hat{R}_{\text{gen}}(\mathcal{L})$ can often be bounded analytically. Hence, via the risk decomposition, one obtains an upper bound to the true risk $\hat{R}_{\text{true}}(\mathcal{L})$.

4. By bounding the generalization risk $\hat{R}_{\text{gen}}(\mathcal{L})$ analytically, we can determine the conditions when learning is guaranteed to work well in practice. Hence, it will allow us to design more effective machine learning algorithms. In fact, some of the most successful machine learning algorithms in practice today, such as Support Vector Machine (SVM), were originally inspired by generalization bounds [11, 56, 57]. The support vector machine algorithm works by finding a large-margin classifier, which mitigates both the empirical risk and the generalization risk simultaneously. This connection between large-margin classification and performance is a key result that we will return to in Chapter 7 when discussing the empirical kernel maps and the performance of the 1-norm SVM classification algorithm.

So, we have reduced a phenomenon that is quite complex into a well-defined mathematical question: how does the generalization risk behave for a given learning algorithm $\mathcal{L}$ and a given sample size $m$? At a first sight, this may appear to be a question of convergence of sample averages to their mean. After all, if the empirical observations (training examples) were generated i.i.d. $z \sim \mathbb{P}(z)$ and all future observations are generated i.i.d. from the same distribution $\mathbb{P}(z)$, shouldn’t we conclude that $\hat{R}_{\text{emp}}(\mathcal{L})$ will converge to $\hat{R}_{\text{true}}(\mathcal{L})$ quickly as dictated by concentration of measure inequalities, such as the Hoeffding bound?

The answer to the previous question is: No. There are different ways of seeing why concentration of measure inequalities cannot be used to conclude that the empirical risk will always converge to the true risk. One approach is to note that the hypothesis $h$ that is inferred by $\mathcal{L}$ is selected based on the empirical observations (training set) $S_m$. In particular, $h$ may be chosen so that the empirical risk is as far away from the true risk as possible. In fact, this is not a far-fetched scenario; it is easy to design
such a learning algorithm. For example, if the observations $z$ were drawn from an uncountable measurable space $\mathcal{Z}$ with no atoms, let $h = S_m$ and define the parametric loss function by:

$$L(z; h) = I\{z \in h\} \in \{0, 1\}$$

Then, we always have $\hat{R}_{emp}(\mathcal{L}) = 0$ by construction. However, if $\mathcal{Z}$ is uncountable and has no atoms, the probability of observing the same sample twice is zero (i.e. $S_m \subset \mathcal{Z}$ has a zero probability measure). Hence, $\hat{R}_{true}(\mathcal{L}) = 1$. Consequently, there exists a learning algorithm $\mathcal{L}$ whose generalization risk is always $\hat{R}_{gen}(\mathcal{L}) = 1$ even if $m \to \infty$. This remark can be summarized as follows:

Concentration of measure inequalities cannot be used to conclude that the generalization risk of a learning algorithm will always vanish as $m \to \infty$ because the inferred hypothesis $h$ is chosen according to the empirical observations.

Therefore, in order to use concentration of measure inequalities, one valid approach is to guarantee that the convergence to the mean holds for the hypothesis $h$ before it is selected by the learning algorithm. In other words, uniform convergence across the entire hypothesis space $\mathcal{H}$, if it holds, will be sufficient to guarantee that the generalization risk will vanish as $m \to \infty$. This was the key insight made by V Vapnik and A Chervonenkis, which led them to introduce a combinatorial measure called the Vapnik-Chervonenkis (VC) dimension [11, 19, 58, 59]. The VC dimension is a property of the hypothesis space $\mathcal{H}$ that is fundamental in statistical learning theory.

One of the crowning achievements of the Vapnik-Chervonenkis theory is to show that having a finite VC dimension in the hypothesis space $\mathcal{H}$ implies that $|R_{emp}(h; S_m) - R_{true}(h)| \to 0$ in probability as $m \to \infty$ [11, 19, 59]. One version of this result is given by the following theorem [19]:
Theorem 1. Let \( \mathcal{L} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \rightarrow \mathcal{H} \) be a learning algorithm and let \( h \in \mathcal{H} \) be its inferred hypothesis. Let \( L(\cdot; h) \in \{0, 1\} \) be a binary parametric loss function that is used to measure the risk. Then, if the hypothesis space \( \mathcal{H} \) has a finite VC dimension \( v < \infty \), the following bound holds with a probability of at least \( 1 - \delta \) over the choice of the sample \( S_m \in \mathcal{Z}^m \):

\[
\left| R_{\text{true}}(h) - R_{\text{emp}}(h; S_m) \right| \leq \tilde{O}(\frac{1}{\delta} \sqrt{\frac{v}{m}})
\]

This theorem shows that the number of observations that are needed to learn successfully is \( \tilde{O}(v) \), where \( v \) is the VC dimension of the hypothesis space \( \mathcal{H} \). This number of observations that are needed to guarantee successful learning is often called the sample complexity.

A simple interpretation of the bound in Theorem 1 relates the sample complexity to the number of parameters that are used by the learning algorithm \( \mathcal{L} \). Suppose \( \mathcal{L} \) picks values for \( K \) different parameters during the learning process. For example, in linear classification methods, where \( x \in \mathbb{R}^d \) and the decision rule is given by \( \text{sign}(\langle w, x \rangle + b) \), the number of parameters is \( d + 1 \). By discretizing the possible values of those parameters into (say) \( N \) possible values each, the size of the discretized hypothesis space is \( |\mathcal{H}| = N^K \). It can be shown that a finite hypothesis space has a finite VC dimension \( v \), with \( v \leq \log |\mathcal{H}| \). Therefore, the VC dimension of the discretized space with \( K \) parameters is \( O(K) \). Hence, the number of observations that are needed to learn successfully (sample complexity) is proportional to the number of the parameters that are used by the learning algorithm \[11, 19\].

1.3.3 A Unifying Approach

Earlier, I stated that both information theory and statistical learning theory are fundamental tools in analyzing machine learning algorithms. In particular, information
theory provides bounds on the sample complexity of active learning, where the agent
\( \mathcal{L} \) is permitted to ask questions/queries. On the other hand, statistical learning theory
provides bounds on the sample complexity of passive learning, where \( \mathcal{L} \) observes and
learns passively. Despite this important connection between the two fields, they are,
however, quite distinct. They employ different tools and they bear no resemblance
with each other.

In a recent line of work, nevertheless, I have shown that an intimate connection
between the two fields exists by introducing the new notion of uniform generalization
[54].

**Definition 4** (Uniform Generalization). A learning algorithm \( \mathcal{L} : \cup_{m=1}^{\infty} \mathcal{Z}^m \rightarrow \mathcal{H} \)
generalizes uniformly if for any \( \epsilon > 0 \), there exists \( m_0(\epsilon) > 0 \) such that for all distrib-
utions \( \mathbb{P}(z) \) on \( \mathcal{Z} \), all parametric loss functions, and all sample sizes \( m > m_0(\epsilon) \), we
have \( |\hat{R}_{emp}(\mathcal{L}) - \hat{R}_{true}(\mathcal{L})| \leq \epsilon \).

Uniform generalization turned to be equivalent to an information-theoretic al-
gorithmic stability constraint on learning algorithms. This is summarized in the
following theorem [54].

**Definition 5** (Mutual Stability). Let \( x \in \mathcal{X} \) and \( y \in \mathcal{Y} \) be two random variables.
Then, the mutual stability between \( x \) and \( y \) is defined by:

\[
S(x; y) = \langle \mathbb{P}(x) \mathbb{P}(y), \mathbb{P}(x, y) \rangle,
\]

where \( \langle P, Q \rangle = 1 - ||P, Q||_\tau \), and \( ||P, Q||_\tau \) is the total variation distance.

**Definition 6** (Variational Information). The variational information \( J(x, y) \) be-
tween the two random variables \( x \) and \( y \) is defined by \( J(x, y) = 1 - S(x, y) \).

**Definition 7** (Algorithmic Stability). Let \( \mathcal{L} \) be a learning algorithm that receives
a finite sample of observations \( S_m = \{z_i\}_{i=1,...,m} \in \mathcal{Z}^m \) drawn i.i.d. from a fixed
distribution $P(z)$. Let $h \sim P_L(h|S_m)$ be the hypothesis produced by $L$, and let $z_{trn} \sim S_m$ be a random variable whose value is drawn uniformly at random from the sample $S_m$. Then, the algorithmic stability of $L$ is defined by: $S(L) = \inf_{P(z)} S(h; z_{trn})$, where the infimum is taken over all possible distributions of observations $P(z)$. A learning algorithm is called stable if $\lim_{m \to \infty} S(L) = 1$.

**Theorem 2.** For any learning algorithm $L : \bigcup_{m=1}^{\infty} Z^n \to H$, algorithmic stability (Definition 7) is both necessary and sufficient for uniform generalization (Definition 4). In addition, $|R_{gen}(L)| \leq 1 - S(h; z_{trn}) \leq 1 - S(L)$, where $R_{gen}(L)$ is the generalization risk of $L$.

Theorem 2 reveals that uniform generalization has, at least, three equivalent characterizations. We describe them next:

1. **Risk:** A learning algorithm $L$ generalizes uniformly if and only if its generalization risk $R_{gen}(L)$ in expectation goes to zero as $m \to \infty$ uniformly across all bounded parametric loss functions. This follows from Definition 4.

2. **Information:** A learning algorithm $L$ generalizes uniformly if and only if the amount of information contained in its inferred hypothesis $h$ about any individual observation in the sample $S_m$ goes to zero as $m \to \infty$. This, for example, is satisfied if $h$ has a finite description length (entropy), if $h$ is sufficiently randomized to anonymize records as in differential privacy, or if the induced concept class of $L$ has a finite VC dimension [54, 60].

3. **Stability:** A learning algorithm $L$ generalizes uniformly if and only if the contribution of any single observation on the hypothesis $h$ vanishes as $m \to \infty$. That is, a learning algorithm generalizes uniformly if and only if it is algorithmically stable.

Moreover, uniform generalization provides an information-theoretic route from generalization in expectation to generalization in probability. This connection be-
tween information theory with large deviation is explored in [60], where tightness results are also established. Other connections between learning and information theory can be established using the class of $f$ divergences (see Appendix A) and inequalities such as the Pinsker inequality [61, 47]. Using this approach, for instance, the connection between over-fitting and the complexity of the hypothesis space can be formalized [54].

1.3.4 Final Remarks

Going back to the initial remarks, we finally note that mathematics, in general, and probability theory, in particular, indeed allow us to formalize a complex phenomenon such as learning quite successfully. This formalism can be used to answer questions rigorously and to develop new effective machine learning algorithms as will be demonstrated throughout this thesis.

1.4 A Taxonomy of Learning

Learning is a very broad discipline with far-reaching implications. Not surprisingly, various learning strategies have been developed to address different tasks. These different paradigms of learning include, but are not limited to, supervised and unsupervised learning, active and passive learning, as well as learning using linear and non-linear methods. In addition, new techniques have been developed for learning in adversarial environments. Because the distinction between these different types of learning methods is critical for the remainder of this thesis, we briefly review these types of learning next.

1.4.1 Supervised and Unsupervised Learning

The first important distinction to make in machine learning is the difference between supervised and unsupervised learning. Loosely speaking, supervised learning refers
to learning environments whose observations come in question-answer pairs (a.k.a. instance-label pairs). In unsupervised learning, by contrast, observations comprise of pure instances. For example, learning to classify between spam and non-spam emails is a supervised classification task because emails are labeled. Here, the goal of learning is to be able to answer the question of whether a given email is a spam or not. On other hand, learning to detect communities in a social network is an unsupervised learning task. There is no “correct” answer to be made in the latter setting.

Formally, the observations \( z \in Z \) in a supervised learning task comprise of instance-label pairs \( z = (x, y) \in X \times Y \). The goal of learning is to be able to predict the label \( y \) given an instance \( x \). Ideally, if the target set \( Y \) (i.e. the space of labels) is countable and the loss function measures the mis-classification error rate, then the optimal approach is to predict using the Bayes rule:

\[
f(x) = \hat{y} = \arg \max_{y \in Y} \{ P(y = y | x = x) \}
\]

However, because the distribution of observations is unknown, the Bayes rule cannot be implemented exactly. Instead, an alternative goal of a learning algorithm would be to approximate it arbitrarily well using a sufficiently large sample size \( m \gg 1 \). This latter requirement is often called Bayes-risk consistency. It can be shown that algorithms such as the \( k \) Nearest Neighbor (k-NN) classifiers are consistent with respect to the Bayes risk when \( m \to \infty, k \to \infty, \) and \( k/m \to 0 \) under the assumption of smooth class conditional distributions \([62, 63]\). Informally speaking, this holds because at the specified limit, the distribution of the empirical observations in the sample \( S_m \) becomes arbitrarily close to the true distribution following the law of large numbers.

However, Bayes risk consistency is not always guaranteed. In fact, it is not always the “right” goal because Bayes risk consistency is achievable for arbitrary distributions
of observations only if non-parametric learning methods are used, which, in turn, require a large sample size in order to avoid over-fitting. Instead, the common goal when designing machine learning algorithms is consistency. A learning algorithm $\mathcal{L}$ is called consistent if the true risk of its inferred hypothesis $\mathbf{h}$ converges to the optimal true risk within the hypothesis space $\mathcal{H}$ as $m \to \infty$ \cite{11, 64, 12}. More formally,

**Definition 8** (Consistency). A learning algorithm $\mathcal{L} : \bigcup_{m=1}^{\infty} \mathbb{Z}^m \to \mathcal{H}$ is called consistent if its inferred hypothesis $\mathbf{h} \in \mathcal{H}$ satisfies:

$$R_{\text{true}}(\mathbf{h}) \to \inf_{\mathbf{h} \in \mathcal{H}} \{ R_{\text{true}}(\mathbf{h}) \}$$

almost surely as $m \to \infty$.

Consistency is a better criteria for evaluating learning algorithms than the Bayes risk consistency because the hypothesis space $\mathcal{H}$ is often restricted to avoid over-fitting. The most prototypical example is linear methods in classification, to be briefly described in Section \ref{1.4.3} where the Bayes risk is not often achievable by design. Consistency, on the other hand, measures how well a learning algorithm performs for the hypothesis space $\mathcal{H}$ it is designed to learn with.

### 1.4.2 Passive and Active Learning

The second important distinction to make in machine learning is the difference between passive and active learning. In passive learning, the learning agent receives observations from its environment passively. Formally, the observations $\mathbf{z}$ are assumed to be generated i.i.d. from some unknown distribution $\mathbb{P}(\mathbf{z})$. In active learning, by contrast, the learning agent interacts with its environment to design its own experience\footnote{In statistics, active learning is often referred to by the name optimal experimental design \cite{65}.} It specifically requests observations that are expected to reduce the uncertainty. This is quite analogous to the bisection method in search algorithms, a.k.a.
binary search, where the algorithm only inspects a value that reduces the region of uncertainty by a constant factor at each iteration. Active learning has been successfully used in many applications including character recognition, text classification, computational chemistry, spam filtering, and bioinformatics \[66, 67, 65\]. In addition, many software companies are increasingly reliant on active learning techniques, such as Google, IBM, and CiteSeer \[65, 68\].

While active learning is advantageous, it is, nevertheless, quite challenging. In order to implement it correctly, one needs a method of defining the region of uncertainty, a method of selecting the observation that is guaranteed (with a high probability) to decrease the region of uncertainty exponentially fast, and a method of updating this region of uncertainty. Even for the concept class of halfspaces, which is arguably the simplest and the most important concept class in machine learning, active learning turns out to be quite non-trivial as will be shown in details later. I will show, for instance, that generalizing the bisection method from \(\mathbb{R}\) to \(\mathbb{R}^d\) for an arbitrary \(d \geq 1\) will involve a combination of maximum-determinant convex optimization and eigenvalue decomposition, when the observations are assumed to be perfectly separable and noise-free. When observations are noisy or are non-separable, complications arise and no efficient algorithm can be designed for this task, in general, that is agnostic to the underlying probability distribution.

1.4.3 Linear and Non-Linear Methods

The third key distinction to make in machine learning is the difference between linear and non-linear methods. As discussed earlier, the risk in any learning task can be decomposed into the risk of under-fitting and the risk of over-fitting. One effective approach of balancing between the two risks is to use a linear method in machine learning, which includes many popular algorithms such as linear SVM, the perceptron, logistic regression, ridge regression, the lasso, and many others. Linear methods
play a key role in various areas including classification, regression, and compressed sensing. They can often achieve comparable performance to non-linear methods in many applications \cite{69,70,19}. Indeed, they are often considered as the most important class of machine learning algorithms in practice today \cite{71,69,72,19,70}. Due to their wide popularity, many open-source implementations currently exist for linear methods including SVMperf, Optimized Cutting Plane Algorithm (OCAS), Pegasos, Stochastic Gradient Descent (SGD), and LIBLINEAR \cite{69}.

Linear methods are usually restricted to the supervised learning setting. To recall, we have in a supervised learning setting an instance space $\mathcal{X}$ and an output space $\mathcal{Y}$. Observations come in pairs $z = (x, y)$ and the goal is to be able to predict the label $y \in \mathcal{Y}$ given the instance $x \in \mathcal{X}$. In linear methods, the instance space is the Euclidean plane $\mathbb{R}^d$ and the output space is a subset of $\mathbb{R}$. Prediction is performed using the rule:

$$\hat{y} = f(\langle w, x \rangle + b),$$

(1.5)

for some function $f : \mathbb{R} \rightarrow \mathbb{R}$ \cite{19}. For example, the function $f(\cdot)$ in a classification setting is often the sign function:

$$f(u) = \begin{cases} 
    +1 & u \geq 0 \\
    -1 & u < 0 
\end{cases}$$

In a regression setting, by contrast, $f(\cdot)$ can be the identity mapping $f(u) = u$. In a probabilistic classification algorithm, whose output is an estimate of the probability of the positive class, $f(\cdot)$ can be the sigmoid function, and so on.

While linear methods are often effective in many applications, they remain parametric methods whose capacity is limited. In particular, they work reasonably well only if the label $y$ was indeed approximately given by the form (1.5) for some function $f(\cdot)$. This, however, is not always the case. Hence, non-linear non-parametric
methods, which increase their complexity according to the data, are also important. This includes similarity-based classification, kernel density estimation, and regression via local weighted averaging. However, the latter class of algorithms are greatly susceptible to the risk of over-fitting, and, they can be computationally demanding.

A middle approach between linear and non-linear methods can be achieved, which is often quite effective in practice. This approach, sometimes called the empirical kernel map or the empirical similarity map, works by applying a linear learning algorithm in the space of embedded similarities \[73, 74\]. Empirical kernel maps are surprisingly effective. They retain the computational advantage of linear classification methods while expanding their capacity according to the data. They are shown theoretically to be capable of mitigating the risks of under-fitting and over-fitting simultaneously \[73\]. I will return to this approach and describe it in details later, where I show that it yields the best method to-date for classification with indefinite similarities.

1.4.4 Adversarial and Non-Adversarial Environments

The final distinction we emphasize in this thesis is the difference between adversarial and non-adversarial environments. Many machine learning algorithms have been successfully deployed in security-sensitive applications. These include spam and malicious email behavior detection \[28, 30, 29\], fraud detection \[32\], as well as intrusion detection \[33, 34, 35, 36, 75, 76, 77\]. The machine learning techniques employed cover a wide spectrum ranging from supervised learning algorithms such as neural networks, SVM, decision trees, the naïve Bayes classifier, and k-NN to unsupervised outlier detection algorithms such as density estimation-based methods and one-class SVM.

Unfortunately, however, security-sensitive applications are characterized by the presence of adversaries, and most prominent machine learning algorithms, such as SVM, were not originally designed for such adversarial environments \[78, 79, 80, 81\].
In the general setting of learning, discussed earlier, it is generally assumed that the past is similar to the future. That is, it is assumed that the empirical observations (training examples) are generated i.i.d. from the same distribution \( P(z) \) that generates future observations. Hence, by building a learning algorithm that works well for a given training set and by restricting its capacity to avoid over-fitting, the learning algorithm is expected to work well with unforeseen future examples as well. In security-sensitive applications, where an adversary can interfere either during or post training, this assumption is no longer valid. Hence, new techniques need to be developed to handle adversarial risks. These techniques include robust statistics, game theory, disinformation, and randomization. The implications of learning in adversarial environments will be a recurring theme throughout this thesis.

1.5 Summary

To summarize, learning is a crucial component of what makes us intelligent. It allows us to achieve difficult tasks that cannot be described algorithmically. Through learning, many successful applications have emerged including image retrieval, document categorization, spam filtering, intrusion detection, and recommendation systems. These machine learning applications have been used to generate revenues, mitigate risks, and to combat crimes.

In the last century, various mathematical tools have been introduced that formalize the complex phenomenon of learning. These include, most notably, information theory and statistical learning theory, which are used to analyze active and passive learning respectively. In addition to the distinction between active and passive learning, learning algorithms can be classified along other important dimensions as well. This includes the distinction between supervised and unsupervised learning, linear and non-linear methods, and learning in adversarial and non-adversarial environments.
Chapter 2

Thesis Overview

2.1 Statement of Contributions

The contributions of this thesis fall under the general subject of “learning via query synthesis”. In this section, I will briefly summarize my research contributions. They are described in slightly more details in the next section, and in full depth in the following chapters.

My research contributions can be summarized as follows. First, I have developed an active learning algorithm for homogenous noise-free halfspaces and demonstrated that its sample complexity is optimal in practice. The query synthesis algorithm can operate in both the single-query setting and the batch-mode setting, and it significantly outperforms all previous algorithms in the literature. However, its space and time complexity grow with time. To alleviate this computational burden in exchange for an increase in the sample complexity, I developed a second query synthesis method for halfspaces. The second method is a Markovian query synthesis algorithm. It maintains a constant memory and time complexity at each round, while still offering an exponential reduction in the sample complexity over passive learning. These algorithms are also extended to non-homogenous halfspaces and to pool-based active learning.

Second, I have analyzed the implications of these algorithms in adversarial environments. Specifically, because adversaries can use the active learning algorithms to reverse-engineer linear classifiers in adversarial environments, a new Semidefinite
Programming (SDP) is developed in this thesis, which mitigates such an adversarial risk. This algorithm is provably robust against the risk of reverse engineering, and it retains the generalization properties of linear classifiers.

Another approach of mitigating the risk of reverse engineering is to use non-linear methods of classification, where similarity-based classification forms a prototypical approach. In this thesis, I show that classification via the empirical kernel maps, also commonly known as 1-norm Support Vector Machine (SVM) or Linear Programming (LP) SVM, is the best classification algorithm for handling arbitrary (indefinite) similarity functions. My contribution in this regard is to establish this overlooked fact both theoretically and empirically. Empirical kernel maps are a middle approach between linear and non-linear classification. They enjoy many advantages in both theory and practice.

2.2 Outline of Thesis

Chapter 3 – Active learning of halfspaces via query synthesis

First, I will look into the problem of active learning of halfspaces via query synthesis. There is an emerging class of new applications for which active learning via query synthesis is a useful tool. These include automated science, such as the “robot scientist” experiment described in [83], and adversarial reverse engineering [84].

For applications in which we desire to learn a halfspace with arbitrary accuracy, the traditional pool-based query learning approach is inefficient because the required pool size grows exponentially fast with each iteration. Query synthesis, by contrast, builds new queries de novo and, hence, offers an attractive alternative approach. In the literature, however, query synthesis strategies have focused on either finite concept classes or on toy problems, such as geometrical shapes in two dimensions [85, 65].

In this thesis, I derive a new query synthesis algorithm for learning halfspaces in \( \mathbb{R}^d \). The proposed algorithm involves a maximum-determinant convex optimization
step and a spectral decomposition step, and it significantly outperforms all previous methods that have been proposed in the literature. It can operate in both the single-query setting and the batch-mode setting. In this thesis, I also present an analytical explanation for the success of the proposed algorithm.

A portion of this work has appeared in the following paper:


**Chapter 4 – Trading the Sample Complexity for Space and Time**

While the previous query synthesis algorithm is optimal in terms of its sample complexity, it has two main drawbacks. First, it relies on solving a maximum-determinant convex optimization problem, whose space and time complexity grow with time as more queries are added. This follows from the fact that the number of constraints in the optimization problem is equal to the number of queries that have been accumulated so far. Second, the algorithm repeatedly solves the entire optimization problem from scratch at each round without incorporating the knowledge gained in previous rounds. The objective of the second query synthesis algorithm is to resolve these drawbacks.

In this thesis, I develop a Markovian query synthesis algorithm, which encodes all of the relevant information about the past in its present state. That is, a query is used only once, and is discarded afterward. Consequently, the new algorithm enjoys a fixed space and time complexity at all rounds. In fact, the proposed algorithm utilizes a variant of the classical Khachiyan’s update formulas for solving linear programs, which can be computed very efficiently. This significant saving in space and time comes at the expense of an increase in the sample complexity; whereas the previous method
has a sample complexity of $O(d \log \frac{1}{\epsilon})$, the new algorithm has a sample complexity of $O(d^2 \log \frac{1}{\epsilon})$.

**Chapter 5 – Extensions of the Query Synthesis Algorithms**

After that, I describe how the query synthesis algorithms in Chapter 3 and Chapter 4 can be extended to new settings as well. First, I derive a new algorithm that extends membership query synthesis to non-homogenous halfspaces, and demonstrate experimentally that the new algorithm achieves an exponential saving in its sample complexity. Second, I describe how the ellipsoidal approximation approach can be extended to pool-based active learning by proposing a sampling criteria, which incorporates information from both the mean and the covariance of the uncertainty. This criteria is interpreted as an aggressive form of uncertainty sampling. I demonstrate experimentally how it achieves a higher accuracy than margin-based sampling by several orders of magnitude.

**Chapter 6 – Learning a Distribution of Linear Classifiers**

Next, because halfspaces can be learned quite efficiently via query synthesis, fixed linear classifiers in adversarial environments are vulnerable to the risk of reverse engineering. This poses a security risk, which has received little attention in the literature before.

In this thesis, I will introduce the risk of reverse engineering, demonstrate it on synthetic and real-world datasets, and investigate randomization as a suitable strategy to mitigate this risk. In particular, I have derived a SDP formulation for learning a large-variance distribution of classifiers subject to the constraint that any single classifier that is picked at random from this distribution provides reliable predictions with a high probability. The rationale behind this algorithm rests on recent seminal results on approximation hardness [86], which are used, in this thesis, to prove that
the proposed algorithm indeed protects against the risk of reverse engineering quite significantly.

Besides, I also analyze the algorithm theoretically. I will prove a generalization bound, which shows that the proposed algorithm is not susceptible to over-fitting. In addition, the generalization bound formalizes the tradeoff between the variance of the distribution and its predictive accuracy. One key contribution of this work is to establish that one can almost always incorporate randomization with a large variance without having to incur a loss in accuracy. In other words, the conventional approach of using a fixed classifier in adversarial environments is Pareto suboptimal. These conclusions are validated on both synthetic and real-world classification problems.

A portion of this work has appeared in the following paper:


**Chapter 7 – Learning via the Empirical Kernel Maps**

Another approach for mitigating the adversarial risk of reverse engineering is to use a non-linear method of classification. The rationale behind this approach also rests on recent results on approximation hardness. In particular, it has be shown that learning convex bodies is hard, in general. There are many non-parametric algorithms proposed in the literature for learning new concepts using similarities. These algorithms, however, either require that the similarity function be positive semidefinite or they handle indefinite similarities unsatisfactorily.

In this thesis, my contribution is to establish how an old method, namely the 1-norm support vector machine, provides the best solution for classification with indefinite similarities. I will present several new theoretical arguments in favor of the
1-norm SVM and demonstrate experimentally that the evidence in its favor is statistically significant. In the literature, 1-norm SVM is often used as an embedded feature selection method, where learning and feature selection are performed simultaneously. Despite its advantages in handling indefinite similarities, it remained a relatively less known method than more-involved less-accurate approaches such as kernel approximation and non-convex optimization. In particular, 1-norm SVM is rarely used as a standard benchmark for this task.

Portions of this work have appeared in the following papers:


2.3 Other Research Projects

Besides this thesis, I have also worked on additional research projects during my Ph.D. course of study. These are described briefly next.

Summability calculus

This is a monograph on Summability Calculus, which is a comprehensive theory of fractional finite sums. In this monograph, I develop the theory of fractional sums and show how it unifies many diverse results in the literature and that it has many important applications. For instance, it is deeply connected to the theory of summability of divergent series, which has found applications in Fourier analysis, quantum field theory, and dynamical systems, among others. Summability Calculus is also connected to the finite difference methods, which is widely applied in numerical analysis and in
computational science and engineering. It also contributes to, and benefits from, the methods of accelerating series convergence. Furthermore, it is intimately related to approximation methods, asymptotic analysis, and numerical integration. I have also connected it to the recent fruitful field of information theory, and use it to prove a stronger version of the celebrated Shannon-Nyquist sampling theorem. Throughout this monograph, many celebrated results are derived, strengthened, and generalized. The list includes, but is not limited to, the Bohr-Mollerup theorem, Stirling’s approximation, Glaisher’s approximation, the Shannon-Nyquist sampling theorem, the Euler-Maclaurin summation formula, and the Boole summation formula. Finally, I show that many identities that have been proved throughout the past 300 years by different mathematicians using different approaches can be derived in an elementary manner using the rules of Summability Calculus. Other new identities are derived for the first time here as well.

This work appears in the following manuscript:


Fractional parts and their relations to the Riemann zeta function

A well-known result, due to Dirichlet and later generalized by de la Vallée-Poussin, expresses a relationship between the sum of fractional parts and the Euler-Mascheroni constant. In this work, I prove an asymptotic relationship between the summation of the products of fractional parts with powers of integers on one hand, and the values of the Riemann zeta function, on the other hand. Dirichlet’s classical result falls as a particular case of this more general theorem.

This work appears in the following paper:

- I. Alabdulmohsin. “Fractional Parts and their Relations to the Values of the

A new analytic summability method for divergent series

The theory of summability of divergent series is a major branch of mathematical analysis that has found important applications in engineering and science. It addresses methods of assigning natural values to divergent sums, whose prototypical examples include the Abel summation method, the Cesáro means, and Borel summability method. In this work, I introduce a new summability method for divergent series and derive an asymptotic expression to its error term. I show that the summability method is both regular and linear, and that it arises quite naturally in the study of local polynomial approximations of analytic functions. Because the proposed summability method is conceptually simple and can be implemented in a few lines of code, it can be quite useful in practice for numerically determining the values, a.k.a. anit-limits, of divergent sums.

This work appears in the following paper:


From generalization in expectation to generalization in probability

One of the fundamental goals in any learning algorithm is to minimize its risk for over-fitting. Mathematically, this implies that the learning algorithm enjoys a small generalization risk, which is defined either in expectation or in probability. Both types of generalization are commonly used in the literature. For instance, generalization in expectation has been used to analyze algorithms, such as ridge regression and Stochastic Gradient Descent (SGD), whereas generalization in probability is used in the Vapnik-Chervonenkis (VC) theory and the Probably Approximately Correct (PAC) Bayesian framework, among others. In this work, I show that, while
a generalization in expectation does not imply a generalization in probability, a uniform generalization in expectation does imply concentration. I establish a chain rule for uniform generalization and use it to derive a tight deviation bound. Hence, an information-theoretic route exists from generalization in expectation to generalization in probability.

This work appears in the following paper:


Content-agnostic malware detection

Malware detection has been widely studied by analyzing either file dropping relationships or characteristics of the file distribution network. This work studies a global heterogeneous malware delivery graph fusing file dropping relationship and the topology of the file distribution network. The integration offers a unique ability of structuring the end-to-end distribution relationship. However, it brings large heterogeneous graphs to analysis. In our study, an average daily generated graph has more than 4 million edges and 2.7 million nodes that differ in type, such as Internet Protocol (IP), Uniform Resource Locator (URL), and files. We propose a novel Bayesian label propagation model to unify the multi-source information, including content-agnostic features of different node types and topological information of the heterogeneous network. Our approach does not need to examine the source codes nor inspect the dynamic behaviors of a binary. Instead, it estimates the maliciousness of a given file through a semi-supervised label propagation procedure, which has a linear time complexity w.r.t. the number of nodes and edges. The evaluation on 567 million real-world download events validates that our proposed approach efficiently detects malware with a high accuracy.

This work appears in the following paper:
Attribute-based zero-shot learning

One transfer learning approach that has gained a wide popularity lately is attribute-based zero-shot learning. Its goal is to learn novel classes that were never seen during the training stage. The classical route towards realizing this goal is to incorporate a prior knowledge, in the form of a semantic embedding of classes, and to learn to predict classes indirectly via their semantic attributes. Despite the amount of research devoted to this subject lately, no known algorithm has yet reported a predictive accuracy that could exceed the accuracy of supervised learning with very few training examples. For instance, the Direct Attribute Prediction (DAP) algorithm, which forms a standard baseline for the task, is known to be as accurate as supervised learning when as few as two examples from each hidden class are used for training on some popular benchmark datasets! In this work, we argue that this lack of significant results in the literature is not a coincidence; attribute-based zero-shot learning is fundamentally an ill-posed strategy. The key insight is the observation that the mechanical task of predicting an attribute is, in fact, quite different from the epistemological task of learning the “correct meaning” of the attribute itself. This renders attribute-based zero-shot learning fundamentally ill-posed. In more precise mathematical terms, attribute-based zero-shot learning is equivalent to the mirage goal of learning with respect to one distribution of instances, with the hope of being able to predict with respect to any arbitrary distribution. We demonstrate this overlooked fact on some synthetic and real datasets.

This work appears in the following paper:

- I. Alabdulmohsin, M. Cisse, X. Zhang, “Is Attribute-Based Zero-Shot Learn-
ing an Ill-Posed Strategy?”, in the *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD)*, Riva del Garda, Italy, September, 2016.

**Algorithmic stability and uniform generalization**

One of the central questions in statistical learning theory is to determine the conditions under which agents can learn from experience. This includes the necessary and sufficient conditions for generalization from a given finite training set to new observations. In this work, I prove that stability in the learning algorithm is equivalent to a uniform generalization in expectation across all parametric loss functions, and I provide various interpretations of this result. For instance, a relationship is proved between stability and data processing, which reveals that algorithmic stability can be improved by post-processing the inferred hypothesis or by augmenting training examples with artificial noise prior to learning. In addition, I establish a relationship between algorithmic stability and the size of the observation space, which provides a formal justification for dimensionality reduction methods. Finally, I connect algorithmic stability to the size of the hypothesis space, which recovers the classical PAC result that the size (complexity) of the hypothesis space should be controlled in order to improve algorithmic stability and improve generalization.

This work is a step towards unifying information theory with statistical learning theory. It has appeared in the following paper:


**Interference in wireless ad hoc networks with smart antennas**

One often cited advantage of smart antennas is their potential for reducing interference. This argument, however, ignores the implications of using smart antennas at the Medium Access Control (MAC) layer. In this work, I show that the use of
directional antennas in wireless ad hoc networks can actually increase interference due to limitations of virtual carrier sensing. I derive a simple mathematical expression for interference in both physical and virtual carrier sense networks, which reveals counter-intuitively that receivers in large dense networks with directional antennas can experience greater interference than in omnidirectional networks unless the beamwidth is sufficiently small. The validity of the mathematical analysis is confirmed using simulations.

This work has appeared in the following paper:


End-to-end delay analysis in wireless sensor networks

Wireless Sensor Networks (WSNs) have been a revolutionary emerging technology for many applications. This is because of their ability to form a vast network of small sensing devices, called motes, distributed in a sparse area to observe the surrounding environment. In such networks, not only are nodes capable of sensing and forwarding data, but they are also capable of carrying out simple computations and filtering out transmitted data. However, sensor nodes in a Wireless Sensor Network (WSN) are battery-equipped, low-power devices with delay constraints. To minimize power consumption, sleep cycles (a.k.a. service vacations) are often introduced. These sleep cycles prolong the battery lifetime but they also increase the end-to-end packet delay. In this work, we investigate the packet delay in a multi-hop wireless sensor network, employing an $M/G/1$ with service vacations. We derive an analytical model and validate it using numerical simulations. Specifically, we look into a relay network that carries sensor measurements to a data sink node, and analyze how its maximum hop-count is determined by its maximum end-to-end delay constraint. In addition, we look into the probability of packet dropping, which arises if sensor packets do not
adhere to the desired delay threshold constraint.

This work has appeared in the following paper:


2.4 Negative Results

In this section, I will briefly describe some of the projects that I worked on, which have led to negative results. Because negative results can be quite useful to the academic community, these investigations are briefly mentioned here.

Agnostic active learning

Agnostic active learning is a very challenging topic. The main goal is to design an algorithm that selects queries such that the true risk of the hypothesis decreases exponentially, a.k.a. linearly, with the number of iterations when observations are corrupted with noise. One key difficulty in the agnostic setting is the fact that a single query, in the presence of noise, no longer provides reliable evidence to prune the version space. Hence, in order to circumvent such a limitation, agnostic active learning algorithms have deviated from the single query paradigm in the realizable setting into the batch-mode paradigm, where multiple queries are selected at each round. This is required in order to be able to refine the set of candidate hypotheses with a statistically significant evidence. One prominent example of such an approach is the Agnostic Active (A²) learning algorithm proposed in [94]. It is similar in spirit to the selective sampling scheme of [95], and has been analyzed quite rigorously by [94, 96].

The overall approach that I pursued was to use generalization bounds to guide the query selection process for learning halfspaces in the low-noise setting, which is a concrete realization of the generic (A²) active learning algorithm. Unfortunately,
standard generalization bounds, such as the VC bounds, turned out to be too loose to be of any practical value in this task. For instance, to guarantee an approximation error of less than 0.1 for a halfspace in only five dimensions with a confidence exceeding 95%, the number of queries required by the VC bound is in the order of 300,000 queries! This, however, remains a project that I would like to return to in the future.

Compressed sensing via Monte Carlo estimation

In this project, I investigated the application of Monte Carlo estimation methods for sparse signal recovery. Given a sparse vector $x \in \mathbb{R}^d$, the fundamental theoretical result that initiated the field of Compressed Sensing is that we can recover $x$ efficiently using a few measurements $m \ll d$. This is achieved by using the Basis Pursuit algorithm in which we minimize the convex loss $||x||_1$ instead of minimizing the combinatorial function $||x||_0$ [97, 98]. Whereas the former method can be implemented in polynomial time, minimizing $||x||_0$ subject to the constraint $y = Ax$ is known to be NP-hard in general. In addition to Basis Pursuit, many algorithms have been proposed including Orthogonal Matching Pursuit (OMP), iterative thresholding, deflation search, message passing, as well as Bayesian methods such as the Fast Bayesian Matching Pursuit (FBMP) and the Support Agnostic Bayesian Matching Pursuit (SABMP) algorithms.

The goal of this project is to investigate whether or not Monte Carlo estimation methods can improve the recovery rate of the Bayesian sparse signal recovery method proposed in [99]. To do this, I examined the use of Monte Carlo estimation methods such as importance sampling and the Metropolis-Hastings algorithm [100]. I have also experimented with variance-reduction techniques that encode more information in the transition probabilities of the Markov chain, such as the sparsity rate if known, to improve the mixing time. However, while those methods can be used to recover sparse signals, their advantage over simpler greedy algorithms, such as OMP [101], remained questionable.
Domain adaptation with gene expression datasets

In this project, we looked into the problem of learning a common representation for gene expression datasets that are generated from multiple environments. Because different environments lead to different systematic errors, we can view this problem as a domain adaptation problem that aims at learning from instances that are generated from multiple distributions (a.k.a. domains) [102].

There are many possible domain adaptation approaches that can be used for this task. For instance, we can design a single optimization problem that implements transfer learning between domains by penalizing large differences between the learned models. Alternatively, we can jointly learn a common representation for all datasets and learn to classify according to the common representation, both at the same time. Here, non-convex optimization problems are approximately solved in my experiments via the Concave Convex Procedure (CCP) [103]. After experimenting with these different approaches, I concluded that the simplest (baseline) approach works best. That is, accuracy is improved best if we implement z-score normalization on each dataset separately before combining all them into one bigger dataset.
Part II

Query Synthesis Algorithms for Halfspaces
Chapter 3

Efficient Active Learning of Halfspaces via Query Synthesis

3.1 Introduction

As briefly described in Chapter 1, active learning is a subfield of machine learning that has received a growing interest over the past 20 years. It has been successfully used in many applications including character recognition, text classification, computational chemistry, spam filtering, and bioinformatics [66, 67, 65]. In addition, many software companies are increasingly reliant on active learning techniques, such as Google, IBM, and CiteSeer [65, 68]. Active learning is commonly referred to by the name of “optimal experimental design” in the statistical literature, and is occasionally referred to by the name of “query learning” in the machine learning community.

Given a supervised classification task, introduced earlier in Section 1.4.1, where observations comprise of instance-label pairs \((x, y) \in \mathcal{X} \times \mathcal{Y}\) and the goal is to predict the label \(y \in \mathcal{Y}\) given an instance \(x \in \mathcal{X}\), the central goal of active learning is to be able to learn well using only a few observations. There are various scenarios in which active learning arises quite naturally. For instance, in speech recognition and text classification, there is often an abundance of unlabeled instances \(\{x_1, x_2, \ldots\}\) that can be used to improve the classification accuracy. However, the labeling cost is not negligible. Hence, the goal of active learning is to nominate a small subset of such unlabeled examples for manual human annotation. Similarly, active learning can be used in dataset compression, in which the entire training set is reduced in size in order to improve the training time.
More precisely, suppose throughout this chapter that our instance space is the Euclidean plane \( \mathbb{R}^d \), and suppose that we would like to estimate a halfspace \( c^* = \{ x \in \mathbb{R}^d \mid \langle w^*, x \rangle \geq 0 \} \) for some unknown normal vector \( w^* \in \mathbb{R}^d \). In other words, we assume that \( w^* \in \mathbb{R}^d \) is fixed and our goal is to estimate it using as few queries as possible. Here, a query \((x_i, y_i)\) comprises of an instance \( x_i \in \mathbb{R}^d \) (the question) and a membership response (the answer):

\[
y_i = c^*(x_i) = \mathbb{I}\{x_i \in c^*\} = \text{sign}(\langle x_i, w^* \rangle) \in \{+1, -1\},
\]

where the query response \( y_i \) is provided by an oracle that knows \( w^* \). In essence, this amounts to the assumptions of operating in the realizable, deterministic, and noise-free setting; assumptions that are widely used in the literature (see for instance \cite{63, 72, 104, 105, 106, 107, 108, 109}). They are satisfied for many important classification problems such as document categorization \cite{110}. In addition, the assumed setting is equivalent to the task of reverse-engineering linear classifiers. We will return to the latter observation and its implications later in Chapter \cite{6}.

Halfspaces, as a concept class, have been studied extensively in the literature. For a passive learner (see for reference Section \cite{1.4.2}), a basic result in statistical learning theory can be used to provide sample complexity bounds. Specifically, suppose that \( \mathbb{P}(x) \) is a probability distribution of instances \( x \in \mathbb{R}^d \) and let \( \hat{w} \) be the hypothesis estimated based on the sample (training set) \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \), whose instances \( x_i \) are drawn i.i.d. according to \( \mathbb{P}(x) \) while \( y_i = \text{sign} \langle w^*, x_i \rangle \in \{+1, -1\} \). Define the estimation error by:

\[
e(\hat{w}) = \mathbb{P}_{x \sim \mathbb{P}(x)} \left\{ \text{sign} \langle w^*, x \rangle \neq \text{sign} \langle \hat{w}, x \rangle \right\}
\]

Then, a classical result using the Vapnik-Chervonenkis (VC) dimension (see for reference Theorem \cite{1}), states that \( \tilde{O}(\frac{d}{\epsilon}) \) training examples are needed in order to achieve
an estimation error of $\epsilon > 0$. For an active learner, on the other hand, a sample complexity of $\Omega(d \log \frac{1}{\epsilon})$ can be shown to be optimal using a sphere counting argument and information-theoretic lower bounds. Therefore, active learning may provide a significant saving in the sample complexity. We formally prove these two claims next.

**Theorem 3** (Sample Complexity for Passive Learning of Halfspaces). Let $\mathcal{L}$ be a learning algorithm that receives $m$ observations of the form $(x, y) \in \mathbb{R}^d \times \{-1, +1\}$, where $x \sim \mathbb{P}(x)$ are drawn i.i.d. and $y = \text{sign} \langle w^*, x \rangle$ for some fixed $w^* \in \mathbb{R}^d$. If $\mathcal{L}$ returns a hypothesis $\hat{w} \in \mathbb{R}^d$ that is consistent with all of the observations, then the distribution-free minimum number of observations needed to guarantee an estimation error of $\epsilon(\hat{w}) > 0$ is $\tilde{\Omega}(\frac{d}{\epsilon(\hat{w})})$, where $\epsilon(\hat{w})$ is defined in Eq. (3.1).

**Proof.** This follows from the Fundamental Theorem of Statistical Learning. Specifically, the learning algorithm $\mathcal{L}$ that returns a consistent hypothesis is, by definition, an empirical risk minimizer (ERM) with respect to the misclassification error rate. However, by the Fundamental Theorem of Statistical Learning, an ERM generalizes if and only if the hypothesis space $\mathcal{H}$ has a finite VC dimension $\nu < \infty$ with a distribution-free generalization risk of $\tilde{\Omega}(\frac{\nu}{m})$ in the realizable setting. Because the empirical risk is zero, since the hypothesis $\hat{w}$ is always consistent with the empirical observations, the estimation error $\epsilon$ is equal to the generalization risk. However, $\nu = d$ for homogenous halfspaces. Therefore, $\tilde{\Omega}(\frac{d}{\epsilon})$ observations are needed.

**Theorem 4** (Sample Complexity for Active Learning of Halfspaces). Let $w^* \in \mathbb{R}^d$ be some fixed vector and let the halfspace be defined by $c^* = \{x \in \mathbb{R}^d : \langle w^*, x \rangle \geq 0\}$. Then, the minimum amount of information in bits that is needed to learn a hypothesis $\hat{w}$ whose estimation error is $\epsilon(\hat{w})$ is $\Omega((d - 1) \log \frac{1}{\epsilon(\hat{w})})$.

**Proof.** The statement of the theorem trivially holds when $d = 1$ because the lower
bound on the sample complexity given by the theorem is zero, which is vacuous. So, we focus on the more interesting case when $d > 1$. With no loss of generality, we can assume that the distribution of observations $\mathbb{P}(x)$ is uniform on the surface of the unit sphere. This is because the estimation error in the latter case is $\frac{\theta(\hat{w}, w^*)}{\pi}$, where $\theta(a, b) \in [0, \pi]$ is the angle between the two vectors $a$ and $b$. Hence, for any other distribution $\mathbb{P}(x)$, the estimation error is, at worst, bounded by a constant multiple of the estimation error with respect to the uniform distribution.

Because the estimation error with respect to the uniform distribution is proportional to the angle between $\hat{w}$ and $w^*$, the minimum number of bits required to learn a hypothesis $\hat{w}$ whose estimation error is less than $\epsilon(\hat{w})$ is proportional to the logarithm of the number of non-overlapping regions at the surface of the unit sphere with an angular diameter of $\Theta(\epsilon)$. To count these regions, we note that the surface of the unit ball $\{w \in \mathbb{R}^d : \|w\|_2 = 1\}$ is a low-dimensional manifold with an intrinsic dimension of $d - 1$. The volume of this manifold is independent of $\epsilon$. However, the volume over the manifold of each of the non-overlapping regions that we need to count is $\Theta(\epsilon^{d-1})$. It follows that their count must be $\Theta(\frac{1}{\epsilon^{d-1}})$. Taking the logarithm, yields the desired result.

Comparing the sample complexity bounds for passive and active learning reveals a significant difference. This difference is, in fact, achievable, which is easily seen when learning thresholds in $\mathbb{R}$ [107]. Consider the example depicted in Fig. 3.1, where our target concept is $c^* = \{x \in \mathbb{R} \mid x \geq \theta\}$ for some unknown $\theta \in [0, 1]$, and our instances $x$ are uniformly distributed in the unit interval $[0, 1]$. Using the bisection method, one can estimate $\theta$ very quickly. If we start with the two examples $(x_1 = a, y_1 = -1)$ and $(x_2 = b, y_2 = +1)$, then the next synthetic queries would be $(x_3 = c, y_3 = -1)$.

---

1The reason the bound is vacuous when $d = 1$ is because we are dealing with homogenous halfspaces. When $d = 1$, there are only two homogenous halfspaces in $\mathbb{R}$, which are the two rays (half-lines) $(-\infty, 0]$ and $[0, \infty)$. To determine which halfspace corresponds to the positive class, we only need one observation. Hence, the sample complexity is independent of $\epsilon$. 

and \((x_4 = d, y_4 = +1)\) in tandem. Clearly, each query cuts the region of uncertainty by one half regardless of its label. Hence, we can achieve an error rate of \(\epsilon\) using only \(\tilde{O}(\log \frac{1}{\epsilon})\) queries \([107]\).

Extending the bisection method into \(\mathbb{R}^d\) for all \(d \geq 1\) is possible, at least in principle, using what is commonly referred to as the “halving algorithm” \([105]\). Unfortunately, the halving algorithm is an idealized method that is often prohibitively complex to implement in practice \([95, 104, 65]\). It operates by first computing the set of all hypotheses that are consistent with all past queries. This set of possible hypotheses is referred to as the version space, a term that was first introduced by Tom Mitchell \([112]\). After that, a query is synthesized that shrinks the “volume” of the version space by a factor of two. Then, the new version space is computed, and the entire process is repeated multiple times. The halving algorithm can be justified rigorously \([85, 104, 111, 107]\).

To approximate the ideal halving algorithm, two active learning paradigms have been studied in the literature. These are called pool-based sampling and query synthesis. We introduce them informally next.

**Definition 9 (Pool-Based Sampling).** In the pool-based sampling method, the learner \(\mathcal{L}\) observes a pool of unlabeled instances \(\mathcal{P}^t = \{x_i\}_{i=1}^t\) at round \(t\) that are drawn i.i.d. from their marginal distribution \(\mathbb{P}(x)\). Then, \(\mathcal{L}\) requests the labels \((y_{i_1}, y_{i_2}, \ldots, y_{i_k})\) to \(k\) instances \((x_{i_1}, x_{i_2}, \ldots, x_{i_k}) \subseteq \mathcal{P}^t\). The fundamental goal is to select the \(k\) instances whose labels are maximally informative to the value of \(w^*\).

**Definition 10 (Query Synthesis).** In the query-synthesis setting, the learner \(\mathcal{L}\) syn-
thesizes \( k \) instances (queries) from scratch and request their labels. Similar to the pool-based sampling method, the fundamental goal is to synthesize the \( k \) instances such that their responses are maximally informative.

Pool-based sampling methods, such as uncertainty sampling and the Query-by-Committee (QBC) algorithm \[108, 104, 105\], are quite popular. Unfortunately, these methods are only guaranteed to work well if the number of unlabeled examples (i.e. the pool size \( |P_t| \)) grows exponentially fast with each iteration. This is due to the fact that the required pool size is always proportional to \( \tilde{O}(\frac{1}{\epsilon}) \) \[108, 106, 109\]. Otherwise, pool-based sampling methods can perform quite poorly \[104, 113\]. Because the required pool size grows exponentially fast before we can guarantee an exponential reduction in sample complexity, query synthesis offers an attractive alternative approach. In query synthesis, new queries are constructed from scratch, and hence, the pool size limitation is completely eliminated.

In this chapter, we provide an efficient query synthesis algorithm for halfspaces. The new algorithm can be interpreted as an approximation to the ideal halving algorithm. In particular, we retain its key advantage; namely that its estimation error enjoys an exponential improvement over random sampling. At each iteration, the new algorithm consists of two steps. First, a convex optimization problem is solved that provides an approximate characterization of the version space. In the second step, a principal component is extracted, which yields the optimal synthetic query that shrinks the version space exponentially fast. Both steps can be implemented quite efficiently.

In addition to the classical active learning setting in which a single query is synthesized at each iteration, the proposed algorithm can also be readily extended into the batch setting, where \( 1 \leq k \leq d \) queries are synthesized per iteration. Batch-mode active learning has been recently described as one of the top challenges in active learning \[68\]. It is important in distributed parallel labeling environments, such as...
in biology [65]. Ideally, batch-mode active learning should produce queries that are both informative and diverse [105]. In our case, the new proposed algorithm operates in the batch setting by solving for the top $k$ eigenvectors in the second step of each iteration, thus achieving both objectives (details are provided in the sequel).

### 3.2 Why Learning Halfspaces using Query Synthesis?

Halfspaces are generally considered to be one of the most important concept classes in practice today [71, 69, 72]. The class of linear separators is broad and includes the Support Vector Machine (SVM) algorithm, the perceptron, as well as logistic regression, and they can achieve comparable accuracy to non-linear classifiers in many applications [69]. In fact, due to their wide popularity, many solvers currently exist for linear classification including SVMperf, Optimized Cutting Plane Algorithm (OCAS), Pegasos, Stochastic Gradient Descent (SGD), and LIBLINEAR [69]. Therefore, it is quite important to develop efficient algorithms for learning halfspaces, including active learning algorithms via query synthesis.

As mentioned earlier, there are two fundamental branches of active learning proposed in the literature: (1) pool-based sampling and (2) query synthesis [65]. Pool-based sampling has been successfully applied for a broad range of problems including classification and regression [113, 65]. In this setting, it is assumed that a large pool of unlabeled examples is available, hence the name, and the goal is to select a small subset of those examples to serve as new queries. Perhaps, the two most dominant approaches in pool-based sampling are uncertainty sampling and the QBC algorithm [113, 68]. Both can be interpreted as approximations to the ideal halving algorithm that was discussed earlier [104, 108]. For the concept class of homogenous (i.e. through the origin) halfspaces, [106] showed that uncertainty sampling could achieve an exponential reduction in sample complexity, while [109] proved similar performance guarantees for a perceptron-like uncertainty sampling algorithm [109].
Similarly, [108] showed that the QBC algorithm could achieve similar performance for the same concept class of homogenous halfspaces. However, all the pool-based sampling methods above require a pool size that grows exponentially fast.

Moreover, numerous other algorithms have been proposed for pool-based sampling. These include algorithms that are based on the expected error reduction [114], bagging and boosting [115], and Expectation-Maximization (EM) [116]. Nevertheless, uncertainty sampling and the QBC algorithm remain dominant and they perform quite competitively [113, 68].

Because pool-based sampling methods are computationally prohibitive, query synthesis offers an attractive alternative approach. In query synthesis, the learning agent constructs new queries de novo in order to reveal sensitive information about the true decision boundary. There is an emerging class of new applications for which this is a useful approach. These include automated science, such as the “robot scientist” experiment described in [83], and adversarial reverse engineering [84]. For instance, an autonomous robot scientist might infer hypotheses based on observations, and design new queries in order to refine those hypotheses [83]. Query synthesis has traditionally focused on regression tasks, under the name of optimal experimental design, because the optimal query can often be determined analytically in such cases [108, 65]. However, it has seldom been extended into the classification setting except for finite concept classes or artificial toy problems [85, 65]. In this chapter, we develop a new query synthesis algorithm for learning halfspaces in $\mathbb{R}^d$ for all $d \geq 1$. We require that the complexity of the query synthesis algorithm grows only polynomially with each iteration, while still offering an exponential reduction in estimation error. In the literature, little work, if any, has been done to achieve these goals. We believe that this algorithm fills an important gap in the active learning literature.

In batch-mode active learning, on the other hand, very few algorithms have been previously proposed [68]. In principle, the goal is to introduce diversity among queries
as a whole, while also ensuring that each query is informative by itself. In order to achieve both objectives, two general approaches have been proposed. The first approach is called representative sampling, which works by first clustering instances \( \{x_i\}_{i=1,2,...} \), which are believed to be informative, into \( k \) groups, and picking the \( k \) representative samples for those groups as queries [110]. The second approach is optimization-based, which treats diversity and uncertainty as two separate terms in a single objective function. For the concept class of homogenous halfspaces, diversity is measured by the pairwise angles between queries [105]. That is, diversity is defined by orthogonality.

3.3 The Proposed Algorithm

3.3.1 Version Space Approximation

Before we derive the query synthesis algorithm for halfspaces, let us first recapitulate how the ideal halving algorithm works. The ideal halving algorithm works in the following manner:

- **Step I:** Compute the version space \( \mathcal{V} \), which is the set of all hypotheses that are consistent with all past queries \((x_i, y_i)\). Because we only need to recover the normal vector \( w^* \) up to a constant factor when learning the halfspace \( c^* = \{x \in \mathbb{R}^d : \langle x, w^* \rangle \geq 0\} \), the version space can be characterized by the intersection of the surface of the unit sphere with a polyhedral set:

\[
\mathcal{V} = \{w \in \mathbb{R}^d \mid ||w||_2 = 1 \land \forall i, y_i \langle x_i, w \rangle \geq 0\} \tag{3.2}
\]

Here, \((x_i, y_i)\) are the previous observations. Eq. \((3.2)\) states, in other words, that the version space is the set of all unit-norm hypotheses \( w \in \mathbb{R}^d \) that are consistent with all of the observations that have been seen so far.
- **Step II:** Measure the “volume” of $\mathcal{V}$. One natural measure of volume is the surface area of the unit sphere that lies inside the polyhedral region as defined in Eq. [3.2]. Stop if the volume of $\mathcal{V}$ is below some predefined tolerance.

- **Step III:** Synthesize a new query that cuts the volume of $\mathcal{V}$ by a factor of two. That is, find an instance $x_{t+1}$ such that the volume of the new version space after observing the pair $(x_{t+1}, y_{t+1})$ is reduced by one half regardless of the value of $y_{t+1} \in \{+1, -1\}$. Then, increment $t$ and go back to **Step I**.

The key difficulty in query synthesis that makes the above halving algorithm prohibitively complex lies in the nature of the version space. Because we can always assume that $||w||_2 = 1$, an ideal definition of the version space is given by Eq. [3.2] whose natural measure of volume is the surface area. This definition, however, is difficult to deal with analytically. For instance, it is known that the related-problem of computing the volume of a polytope is $\#P$-hard [117].

In order to circumvent such a limitation, we propose to approximate the version space $\mathcal{V}$ by the largest ellipsoid $\varepsilon^* = (\mu^*, \Sigma^*)$, with mean $\mu^*$ and covariance $\Sigma^*$, which is consistent with all of the previous queries and whose mean $\mu^*$ lies at the surface of the unit sphere, i.e. $||\mu^*||_2 = 1$. This approximation can be interpreted as a smoothening of the original ideal definition of $\mathcal{V}$ so that query synthesis can be carried out quite efficiently.

In order to determine $\varepsilon^*$, we need to determine its mean $\mu^*$ and covariance matrix $\Sigma^*$. By definition of $\varepsilon^*$, we have:

$$ w \in \varepsilon^* \iff ||w - \mu^*||_{\Sigma^*} \leq 1, \quad (3.3) $$

where we write $||z||_A$ to denote the induced norm of $z$ by the positive definite matrix $A$. That is, $||z||_A = \sqrt{z^T A z}$.

To guarantee that the entire ellipsoid $\varepsilon^*$ is consistent with a query $(x_i, y_i)$, it can
be shown using Lagrange duality (see Appendix B) that Eq. (3.3) implies that the following condition is both necessary and sufficient:

\[ y_i \langle x_i, \mu^* \rangle \geq \|x_i\|_\Sigma^* \]  

(3.4)

The volume of an ellipsoid \( \varepsilon = (\mu, \Sigma) \) with mean \( \mu \) and covariance matrix \( \Sigma \) is proportional to \( \det \Sigma \). Because the logarithmic function is monotone and increasing, maximizing \( \det \Sigma \) is equivalent to maximizing the concave function \( \log \det \Sigma \). Consequently, in order to solve for \( \varepsilon^* \), we need to solve the following optimization problem:

\[
\begin{align*}
\text{maximize} & \quad \log \det \Sigma \\
\text{subject to} & \quad y_i \cdot (\mu^T x_i) \geq \sqrt{x_i^T \Sigma x_i}, \quad \text{for all } i = 1, 2, \ldots, t \\
& \quad \|\mu\|_2 = 1,
\end{align*}
\]

(3.5)

where we write \( \mathbb{S}_+^d \) to denote the positive semidefinite cone:

\[ \mathbb{S}_+^d = \{ A \in \mathbb{R}^{d \times d} : A \succeq 0 \} \]

The optimization problem, however, is not convex. To cast it as a convex optimization problem so that it can be solved quite efficiently, we introduce the new variable \( S = \Sigma^{\frac{1}{2}} \). Furthermore, we replace the equality constraint with an inequality constraint as follows:

\[
\begin{align*}
\text{maximize} & \quad \log \det S \\
\text{subject to} & \quad y_i \cdot (\mu^T x_i) \geq \|S x_i\|_2, \quad \text{for all } i = 1, 2, \ldots, t \\
& \quad \|\mu\|_2 \leq 1
\end{align*}
\]

(3.6)
It is clear that the constraint $||\mu||_2 \leq 1$ holds with equality at optimality. Otherwise, we can multiply both $\mu$ and $S$ by $||\mu||_2^{-1}$, which increases the objective function without violating the constraints. Therefore, the optimal solution to the optimization problem (3.6) is precisely the desired $\varepsilon^*$. However, the optimization problem in (3.6) is now convex. Upon making the substitution $\Sigma = S^2$, we recover the optimal solution to the original optimization problem (3.5). Thus, $\varepsilon^*$ can be determined efficiently, which provides us with a smooth approximation to the version space.

In applications where $d \gg 1$, estimating the entire $d \times d$ covariance matrix in (3.6) is computationally expensive. One method to remedy such a problem is to assume that $S$ is diagonal, which reduces the number of variables from $\Theta(d^2)$ down to $\Theta(d)$. This has the additional advantage of replacing the log-determinant function with a geometric-mean that can be handled more readily by Second-Order Cone Programming (SOCP) solvers [118]. The new optimization problem becomes:

$$\max_{\mu, s \geq 0} \left( \prod_{j=1}^{d} s_j \right)^{\frac{1}{d}}$$

subject to

$$y_i \cdot (\mu^T x_i) \geq ||s \circ x_i||_2, \quad \text{for all } i = 1, 2, \ldots, t$$

$$||\mu||_2 \leq 1$$

(3.7)

Throughout our experiments, this is the optimization problem we are going to use.

### 3.3.2 Efficient Implementation

The optimization problem in (3.6) falls under the general class of maximum-determinant (max-det) convex optimization problems. This is a class of optimization problems that have been studied extensively in the literature. They can be solved quite efficiently, both in the worst-case complexity theory and in practice [119].

On the other hand, the optimization problem in (3.7) can be cast as a SOCP problem by introducing new variables [118]. Second-Order Cone Programming solvers
are generally very efficient. Using CVX \cite{120}, for instance, the optimization problem in (3.7) takes 2s, 4s, and 14s to solve when \( d = 100 \) and the number of queries is 1000, 2000, and 5000 respectively. Similarly, it takes 4s, 9s, and 35s to solve when \( d = 200 \) and with 1000, 2000, and 5000 queries respectively\footnote{These figures were obtained on a workstation with 2.5 GHz Intel Core i5 processor.} The time complexity is almost linear with respect to the problem dimension \( d \) and to the number of queries.

### 3.3.3 Query Synthesis

Having obtained a smooth approximation to the version space \( \mathcal{V} \), the next step is to use our knowledge of \( \varepsilon^* \) to synthesize new queries. If our goal is to cut the volume of \( \varepsilon^* \) by a factor of two, then any query that resides in the orthogonal complement of \( \mu^* \) will achieve such an objective as proved next.

**Lemma 1** (Uncertainty Sampling). Suppose \( \varepsilon^* \) at iteration \( t \) has mean \( \mu^* \) and covariance matrix \( \Sigma^* \). Let \( x_{t+1} \) be any instance such that \( \langle x_{t+1}, \mu^* \rangle = 0 \). Then, regardless of the label \( y_{t+1} \in \{+1, -1\} \), exactly half of the points in \( \varepsilon^* \) are inconsistent with \((x_{t+1}, y_{t+1})\).

**Proof.** In order to show that exactly half of the points in \( \varepsilon^* \) are inconsistent with \((x_{t+1}, y_{t+1})\), we need to show that a one-to-one correspondence exists between the set of consistent points \( \varepsilon^+ = \{w \in \varepsilon^* | y_{t+1} \langle x_{t+1}, w \rangle > 0\} \) and the set of inconsistent points \( \varepsilon^- = \{w \in \varepsilon^* | y_{t+1} \langle x_{t+1}, w \rangle < 0\} \). We ignore the case \( \langle x_{t+1}, w \rangle = 0 \) because the volume of such a set is zero.

First, for any \( w \in \mathbb{R}^d \), write \( \bar{w} = \mu^* - (w - \mu^*) \). Note that \( \bar{w} = w \). Our first claim is that the condition \( w \in \varepsilon^* \) is equivalent to the condition \( \bar{w} \in \varepsilon^* \). This can be easily seen by using Eq. (3.3). If \( w \in \varepsilon^* \), then we have by definition:

\[
(w - \mu^*) \Sigma^{*^{-1}}(w - \mu^*) \leq 1,
\]
which is also satisfied for $\bar{w}$. The converse also holds. Hence, the statements $w \in \varepsilon^*$ and $\bar{w} \in \varepsilon^*$ are equivalent.

Next, we show that for any query $(x_{t+1}, y_{t+1})$ that satisfies the orthogonality condition $\langle x_{t+1}, \mu^* \rangle = 0$ and the condition $\langle x_{t+1}, w \rangle \neq 0$, exactly one of $w$ or $\bar{w}$ will be consistent with $(x_{t+1}, y_{t+1})$ for any $w \in \varepsilon^*$. This can be seen by writing:

$$y_{t+1} \langle x_{t+1}, w \rangle = y_{t+1} \langle x_{t+1}, \mu^* + (w - \mu^*) \rangle = y_{t+1} \langle x_{t+1}, w - \mu^* \rangle$$

In the second line, we used the orthogonality condition $\langle x_{t+1}, \mu^* \rangle = 0$. However, for $\bar{w} \in \varepsilon^*$, we also have by definition of $\bar{w}$:

$$y_{t+1} \langle x_{t+1}, \bar{w} \rangle = -y_{t+1} \langle x_{t+1}, w - \mu^* \rangle$$

Therefore, if $w \in \varepsilon^+$, then $\bar{w} \in \varepsilon^-$ and vice versa. Since $w, \bar{w} \in \varepsilon^*$, we deduce the statement of the lemma.

Lemma [1] reveals that uncertainty sampling, i.e. picking a random query that exactly lies in the orthogonal complement of $\mu^*$, will always eliminate half of the points in $\varepsilon^*$. Of course, this does not imply that the volume of $\varepsilon^*$ at the next iteration will reduce by one half since an entirely different ellipsoid in an entirely different region might be picked next. However, by synthesizing a query that cuts the largest ellipsoid by one half, we are effectively approximating the ideal halving algorithm in a greedy fashion.

Fortunately, we can do even much better than uncertainty sampling. To see how uncertainty sampling might fail, we note that even if the volume of $\varepsilon^*$ converges to zero, the ellipsoid itself may not converge to zero. For example, $\varepsilon^*$ might be pushed towards a lower dimensional subspace. What we desire, instead, is to guarantee that
Figure 3.2: By blindly eliminating “half” of the hypotheses in $\varepsilon^*$, we may push $\varepsilon^*$ into a lower-dimensional subspace as depicted at the top, where arrows indicate the chosen direction of the cut. Here, the fact that $\text{volume}(\varepsilon^*) \to 0$ does not imply that $\varepsilon^* \to 0$. Instead, it is imperative to shrink both volume and axis length as depicted at the bottom.

all axes of the ellipsoid are shrunk exponentially fast. As depicted in Fig. 3.2, one method to accomplish this is to cut $\varepsilon^*$ along its largest axes, where the weight of each axis is determined by its length.

Formally speaking, we would like to maximize the projection $\langle x_{t+1}, v_j \rangle$, where $v_j$ are the eigenvectors of $\Sigma^*$. However, we do not care about the sign of $\langle x_{t+1}, v_j \rangle$, i.e. we do not care if the light and dark regions in Fig. 3.2 are swapped. In addition, each $v_j$ is weighted by its length $\lambda_j$, where $\lambda_j$ is the corresponding eigenvalue. These requirements lead to the following optimization problem:

$$\begin{align*}
\text{maximize} & \quad \sum_{j=1}^{d} \lambda_j \langle x, v_j \rangle^2 \\
\text{subject to} & \quad \langle \mu^*, x \rangle = 0
\end{align*}$$

(3.8)

Because the norm of $x_{t+1}$ does not matter in our queries, we fix $\|x\|_2 = 1$ in (3.8). If we let $N$ be the orthonormal basis of the orthogonal complement of $\mu^*$ and write
\( x = N \alpha \), then the optimization problem (3.8) can be rewritten as:

\[
\maximize_{\alpha : \|\alpha\|_2 = 1} \quad \alpha^T (SN)^T (SN) \alpha
\]

(3.9)

Recall here that \( S = \Sigma^{\frac{1}{2}} \), which is the optimal solution to (3.6) or (3.7), depending on which formulation is used. Because \( N \) is orthonormal, the condition \( \| x \|_2 = 1 \) is equivalent to the condition \( \| \alpha \|_2 = 1 \).

It is an elementary result in linear algebra that the solution to the optimization problem (3.9), denoted as \( \alpha^* \), is the top eigenvector of the positive semidefinite matrix \( N^T S^T S N = N^T \Sigma^* N \), which can be computed efficiently. Therefore, we choose as our next query the unique instance:

\[
x_{t+1} = N \alpha^*
\]

(3.10)

Next, we extend the algorithm to the batch-mode setting in a straightforward manner. As suggested in the literature, we would like our \( k \) queries to be both informative and diverse. In our case, this amounts to the requirements that all \( k \) queries belong to the orthogonal complement of \( \mu^* \), and are optimal in the sense given by (3.8). In addition, diversity is enforced by requiring that the queries themselves be orthogonal to each other [105]. Similar to the single-query setting, the optimal solution is to pick the top \( k \) eigenvectors of the matrix \( N^T \Sigma^* N \), where \( N \) is the orthonormal basis to the orthogonal complement of \( \mu^* \). After that, we map those \( k \) eigenvectors into \( k \) synthetic queries using Eq. (3.10). The entire algorithm is summarized in Algorithm 1.
Algorithm 1: The proposed query synthesis algorithm for halfspaces.

Data: Observations \( \{(x_i, y_i)\}_{i=1}^{t} \)

Result: \( k \) synthetic queries \( \{x_{t+1}, x_{t+2}, \ldots, x_{t+k}\} \)

Begin:

1. Solve the optimization problem in (3.6) or (3.7). Let \( \mu^* \) and \( \Sigma^* \) be the optimal solutions.

2. Compute \( N \), which is the orthonormal basis to the orthogonal complement of \( \mu^* \) (the null-space of \( \mu^{*T} \)).

3. Compute \( \alpha_1, \alpha_2, \ldots, \alpha_k \), which are the top \( k \) eigenvectors of the matrix \( N^T \Sigma^* N \).

4. Return \( x_{t+1} = N \alpha_1, \ldots, x_{t+k} = N \alpha_k \).

3.4 Evaluation

3.4.1 Methodology and Results

In our evaluations, we used the SOCP formulation in (3.7) because it is more computationally efficient. The optimization problem was solved using CVX [120], where the MATLAB implementation code is provided in Appendix C. For the alternative methods, training was done using linear SVM [71]. Because the proposed algorithm targets the realizable noise-free setting, the experiments were carried out for various choices of problem dimension \( d \) and random choices of coefficient vectors \( w^* \in \mathbb{R}^d \).

Single-Query Synthesis

In order to validate the proposed method in the single-query setting, we compare it with the following methods:

1. Random Sampling: Here, queries are chosen uniformly at random from the unit sphere. This is the baseline method.

2. Uncertainty Sampling: Here, queries are picked uniformly at random in the orthogonal complement of \( w \), where \( w \) is the coefficient vector learned using linear SVM. Because instances have zero margin, this method can be interpreted
Figure 3.3: In this figure, the estimation error is plotted in a log-scale against the number of queries for various query synthesis algorithms. The left and right columns correspond to \( k = 1 \) (single-query) and \( k = 5 \) (batch-mode) respectively. The rows correspond to \( d = 25, 50, 75 \) from top to bottom respectively. The ‘+’ curves are for random sampling. The ‘\( \lozenge \)’ curves are for uncertainty sampling (\( k = 1 \), left) and orthogonal sampling (\( k = 5 \), right). The ‘\( \circ \)’ curves are for the query-by-bagging method (\( k = 1 \), left) and the representative sampling method (\( k = 5 \), right). Finally, the ‘\( \Box \)’ curves are for the new proposed method.

as the query synthesis analog to the pool-based uncertainty sampling method.

3. Query-by-Bagging: This method takes a bagging approach to implement the QBC algorithm \([115]\). Following \([115]\), we used a bag size of 20. At each iteration, a pool of 1,000 instances is randomly generated, and the query with the largest disagreement among the 20 hypotheses is picked.

Batch-Mode Query Synthesis

In the batch setting, where \( k \) queries are synthesized at each iteration, we compare the proposed algorithm with the following three methods:

1. Random Sampling: Here, the \( k \) queries are chosen uniformly at random from the unit sphere. Again, this is the baseline method.
2. Orthogonal Sampling: In this method, $k$ orthogonal queries are chosen uniformly at random from the orthogonal complement of $w$, where $w$ is the coefficient vector learned using linear SVM. This can be interpreted as the query synthesis analog to the optimization-based method proposed in [105].

3. Representative Sampling: This is a clustering-based method proposed in [110].

In our implementation, we used a pool size of 1,000 instances.

The exact experiment ran as follows. For a fixed choice of $k$ and $d$, we began with a random choice of a unit-norm $w^* \in \mathbb{R}^d$, a single positive example, and a single negative example. After that, we ran the different query synthesis algorithms in parallel for up to a total of 1,000 queries.

At each iteration, we used past queries to estimate the true coefficient vector $w^*$ using either the support vector machine (SVM) [56] for the competing methods or the mean of the maximum inscribed ellipsoid in our proposed method. SVM was implemented using the LIBLINEAR package [71], while the proposed method was implemented using CVX [120] (implementation code is available in Appendix C). If we let $\hat{w}$ be the estimated coefficient vector, then estimation error is defined by $||w^* - \hat{w}||_2$.

Note that $||w^*||_2 = ||\hat{w}||_2 = 1$ always holds, which implies that the estimation error is a proper measure of disagreement between the two halfspaces. In the batch setting, we used $k = 5$. Also, all experiments were repeated for $d \in \{25, 50, 75\}$. Figure 3.3 shows the evaluation results.

As shown in the figure, the proposed spectral algorithm significantly outperforms all other methods in both the single-query setting and the batch-mode setting. In fact, unlike the other methods whose estimation error is subject to diminishing returns, the new proposed method always maintains an exponential reduction in estimation error as indicated by the linear curve in this log-scale plot. In fact, the estimation error is approximately given by $\exp\{-\frac{m}{2d}\}$. This implies that in order to achieve an estimation error of $||w^* - \hat{w}||_2 = \epsilon$, we only need $O(d \log \frac{1}{\epsilon})$ synthetic queries.
As proved earlier, a sample complexity of $\Omega(d \log \frac{1}{\epsilon})$ is optimal. Hence, the sample complexity of Algorithm 1 is optimal in practice (up to a leading constant).

### 3.4.2 Analytical Argument

The optimality of the proposed query synthesis algorithm is only demonstrated experimentally. However, we can justify it heuristically as follows. First, we know from the Fundamental Theorem of Statistical Learning that the estimation error of a passive learner that produces a consistent hypothesis is $\tilde{O}(\frac{d}{m})$. This is also an upper bound on the estimation error in the passive learning setting under the realizability assumption. Next, consider the following thought experiment. In this thought experiment, we have i.i.d. examples $(x_i, y_i)$ that are generated in tandem. At round $t$, the learner $\mathcal{L}$ maintains the version space $\mathcal{V}$, which is the set of all hypotheses that are consistent with all previous examples. Then, the pair $(x_{t+1}, y_{t+1})$ is revealed to the learner $\mathcal{L}$ only if:

$$\exists w_1, w_2 \in \mathcal{V} : \text{sign}(w_1, x_{t+1}) \neq \text{sign}(w_2, x_{t+1})$$

In other words, an example $(x_{t+1}, y_{t+1})$ is hidden from $\mathcal{L}$ if it does not convey any new information to $\mathcal{L}$. Now, if we write the region of uncertainty $U_t$ by:

$$U_t = \{x \in \mathbb{R}^d : \exists w_1, w_2 \in \mathcal{V} : \text{sign}(w_1, x) \neq \text{sign}(w_2, x)\}$$

Then:

$$\epsilon = \mathbb{P}_{x \sim \mathcal{P}(x)}\{x \in U_t\}$$

provides an upper bound to the estimation error of any hypothesis $w \in \mathcal{V}$ in the version space. The probability the pair $(x_{t+1}, y_{t+1})$ will be revealed to $\mathcal{L}$ is, by definition, equal to $\epsilon$. This means that every pair that is revealed to $\mathcal{L}$ is equivalent to $\frac{1}{\epsilon}$ of i.i.d. observations, which is the mean of this geometric random variable. Hence, if, at each round $t$, $\mathcal{L}$ draws at random an instance $x_{t+1}$ from the restriction of the marginal
distribution \( \mathbb{P}(z) \) in the region of uncertainty \( U_t \), the estimation error is guaranteed to drop exponentially fast because this is equivalent to the thought experiment just described. The sample complexity becomes \( O(\log \frac{1}{\epsilon}) \) ignoring the dependence on \( d \). Therefore, we can achieve an optimal sample complexity if we can draw the queries from the true region of uncertainty \( U_t \) at each round. But, how do we achieve this task efficiently?

To see why the spectral algorithm proposed in this chapter can be interpreted as an approximate implementation of the above thought experiment, we can re-write the region of uncertainty into the following equivalent form:

\[
U_t = \{ x \in \mathbb{R}^d : (\inf_{w_1, w_2 \in \mathcal{V}} x^T (w_1 w_2^T) x) \leq 0 \}
\]

In the above equation, if \( x \in U_t \) and \( w_1, w_2 \in \mathcal{V} \) are the two vectors that achieve the infimum in the last equation, then either \( x^T w_1 \leq 0 \) or \( x^T w_2 \leq 0 \) holds. Therefore, we have:

\[
U_t = \{ x \in \mathbb{R}^d : (\inf_{w \in \mathcal{V}} w^T x)(\sup_{w \in \mathcal{V}} w^T x) \leq 0 \}
\]

In our proposed method, we used an ellipsoidal approximation to the version space \( \mathcal{V} \). If, in the above expression, we replace \( \mathcal{V} \) by \( \varepsilon^*(\mu^*, \Sigma^*) \), we can solve each optimization problem separately. Using Lagrange duality (see Appendix B), it can be shown that:

\[
\begin{align*}
\arg \inf_{w \in \varepsilon(\mu, \Sigma)} w^T x &= \mu - \frac{1}{\sqrt{x^T \Sigma x}} \Sigma x \\
\arg \sup_{w \in \varepsilon(\mu, \Sigma)} w^T x &= \mu + \frac{1}{\sqrt{x^T \Sigma x}} \Sigma x
\end{align*}
\]

Plugging these equations into the definition of \( U_t \) gives us the following simple char-
characterization:

$$\hat{U}_t = \{ x \in \mathbb{R}^d \mid x^T (\mu^* \mu^{*T} - \Sigma^*) x \leq 0 \}$$

(3.12)

Here, we write $\hat{U}_t$ instead of $U_t$ to emphasize the fact that this is an approximate characterization to the true region of uncertainty $U_t$. Therefore, the task of the active learner reduces to the task of picking an instance that lies in $\hat{U}_t$ at round $t$.

The proposed spectral algorithm achieves precisely this task because it selects an instance $x_{t+1}$ such that $\langle x_{t+1}, \mu^* \rangle = 0$ while also maximizing $x_{t+1}^T \Sigma^* x_{t+1}$. Therefore, we expect the proposed algorithm to enjoy an optimal sample complexity in practice, which is indeed the case as demonstrated in the previous experiments.

3.5 Conclusions

In this chapter, I proposed a new query synthesis algorithm for learning homogenous halfspaces. Unlike pool-based sampling methods whose complexity grows exponentially fast with each iteration, the complexity of the proposed algorithm grows only polynomially while still offering an exponential reduction in estimation error. Experimentally, the new algorithm significantly outperforms popular active learning strategies such as uncertainty sampling and representative sampling. It enjoys a sample complexity of $O(d \log \frac{1}{\epsilon})$ in practice, which is proved to be optimal using a sphere counting argument.

---

This argument does not constitute a proof because the queries are not drawn at random from the restriction of $P(x)$ on the uncertainty region $U_t$. Instead, the query $x_{t+1}$ is chosen deterministically from the approximation $\hat{U}_t$. 
Chapter 4

Trading the Sample Complexity for Space and Time

4.1 Introduction

Earlier in Chapter 3, I presented an algorithm for learning a noise-free halfspace using query synthesis. The sample complexity of the algorithm was shown to be optimal in practice, and I presented an analytical justification for its performance. However, there are two main drawbacks of that algorithm. First, it relies on solving a maximum-determinant convex optimization problem, whose space and time complexities grow with time as more queries are added. This follows from the fact that the number of constraints in the optimization problem is equal to the number of queries that have been accumulated so far. Second, the algorithm repeatedly solves the entire optimization problem from scratch at each round without incorporating the knowledge gained in previous rounds.

In this chapter, these issues are resolved. I will present a Markovian query synthesis algorithm, which encodes all of the relevant information about the past in its present state. That is, a query is used only once, and is immediately discarded afterward. Consequently, the new algorithm enjoys a fixed space and time complexity at all rounds. In fact, the proposed algorithm utilizes a variant of the classical Khachiyan’s update formulas for solving linear programs, which can be computed very efficiently at each round.

The significant savings in space and time come at a cost, however. The price paid for using the new algorithm is an increase in the sample complexity. In particular,
whereas the earlier query synthesis algorithm enjoys an optimal sample complexity of $O(d \log \frac{1}{\epsilon})$ in practice, the new algorithm has a sample complexity of $O(d^2 \log \frac{1}{\epsilon})$. Moreover, whereas the previous algorithm could handle a batch size of up to $k \leq d$ queries at each round, the new algorithm can handle a batch size of up to $k < (d+1)/2$ queries. Therefore, the new algorithm is advantageous in settings where the savings in space and time complexity outweigh the increase in sample complexity.

To motivate the proposed method in more concrete terms, the algorithm proposed in Chapter 3 has been implemented using CVX with its default settings in large dimensions. When the problem dimension is $d = 1000$, for instance, which is smaller than the dimensionality of common classification tasks such as document categorization and gene expression datasets, we find that the maximum-inscribed ellipsoidal method takes more than 24 hours of CPU time and close to 4 GB of memory to find an estimate $\hat{w}$ with a relative error $\frac{||\hat{w} - w^*||_2}{||w^*||_2}$ of less than 20%, where $w^*$ is the correct solution. In fact, the optimization problem no longer fits into memory after making 2,000 queries and the algorithm essentially halts.

By contrast, the algorithm developed in this chapter takes less than 3 minutes of CPU time in the same workstation and less than 1 MB of memory to reach the same level of accuracy! The price for such a significant gain in space and time is an increase in sample complexity, as mentioned earlier. In particular, whereas the maximum-inscribed ellipsoidal method requires 2,000 queries to achieve a relative estimation error of less than 20% in $\mathbb{R}^{1000}$, the new proposed algorithm uses around 10,000 queries to achieve the same level of accuracy. Thus, the sample complexity is traded for space and time.

In this chapter, I will give an overview of the proposed query synthesis algorithm and how it relates to classical algorithms for solving linear programs. Then, I will present experimental results that validate the claimed advantages of the newly pro-

---

1The experiment was conducted on a 2.5 GHz Intel Core i5, 4GB 1333 MHz DDR3 workstation.
posed algorithm. Finally, I will present an analytical justification to the experimental findings, in which I argue that the sample complexity of this algorithm is \(O(d^2 \log \frac{1}{\epsilon})\).

### 4.2 The Khachiyan Update Formulas

Before I describe the proposed query synthesis algorithm for learning homogenous halfspaces, I will briefly describe the Khachiyan update formulas for solving a system of linear inequalities \(a_i^T w \geq b_i\), where \(i = 1, \ldots, m\), \(a_i \in \mathbb{R}^d\), and \(b_i \in \mathbb{R}\) [121]. These update formulas will be a starting point for the proposed query synthesis method. In particular, we will extend the Khachiyan update formulas to the batch-mode setting, develop a query synthesis formula, and analyze the properties of the overall method afterward. Note that throughout the sequel, we assume that \(d > 1\). When learning thresholds in \(\mathbb{R}\), on the other hand, the bisection method of Fig. 3.1 suffices.

To begin with, suppose we have a system of linear inequalities \(a_i^T w \geq b_i\) and that we would like to find a solution \(w^* \in \mathbb{R}^d\) that satisfies all of the inequalities. Then, the Khachiyan method proceeds as follows. First, we start with \(w_0 = 0\) and \(\Sigma_0 = R^2 I\), where \(R\) is chosen to be large enough so that \(\|w^*\|_2 < R\). After that, the algorithm proceeds iteratively. Given the current estimate \(w_t \in \mathbb{R}^d\), the algorithm terminates if \(w_t\) satisfies all of the inequality constraints. Otherwise, it picks a violating constraint \(a_i^T w_t < b_i\) and performs the updates:

\[
\begin{align*}
  w_t &= w_{t-1} + \frac{1}{d+1} \cdot \frac{\Sigma_{t-1} a_i}{\|a_i\|_{\Sigma_{t-1}}} \\
  \Sigma_t &= \frac{d^2}{d^2 - 1} \left[ \Sigma_{t-1} - \frac{2}{d+1} \cdot \frac{\Sigma_{t-1} a_i a_i^T \Sigma_{t-1}}{\|a_i\|^2_{\Sigma_{t-1}}} \right]
\end{align*}
\]  

(4.1)

It can be shown that the sequence of estimates \(w_t\) will converge to a solution to the system of linear inequalities, if it exists, in \(O(d^2 L)\), where \(L\) is the description length of the problem [121]. The sequence of ellipsoids \(\varepsilon(w_t, \Sigma_t)\) has a precise geometrical
interpretation \cite{121}. However, it suffices for our purposes to view them as small enclosing ellipsoids that contain a subset of the solutions of the system of linear inequalities.

The Khachiyan method played a pivotal historical role. It was the first algorithm for solving Linear Programming (LP), which was guaranteed to find a correct solution in polynomial time. This was a significant event because establishing the complexity class of linear programs was considered to be one of the most important open problems in theoretical computer science at the time \cite{122}. However, the Khachiyan method is rarely used in solving linear programs because the simplex method and the more-recent interior point methods often perform much better in practice \cite{123}.

4.3 The Proposed Algorithm

In this section, I will write $\mathcal{V}_t$ to denote the version space at round $t$. To recall, the version space is the set of hypotheses that are consistent with all queries. I will first describe the query synthesis algorithm and analyze it theoretically after that.

4.3.1 Description

The method proposed in this chapter for learning a homogenous halfspace via query synthesis is a generalization of the Khachiyan method to the batch-mode setting as well as a specific query synthesis rule. The algorithm is outlined in Algorithm 2. It maintains an ellipsoidal approximation to the version space $\mathcal{V}_t$, henceforth denoted $\varepsilon(\mu_t, \Sigma_t)$, such that $\mathcal{V}_t$ is entirely contained inside $\varepsilon(\mu_t, \Sigma_t)$. At round $t$, the algorithm synthesizes a batch of $k_t$ orthonormal queries at random from the orthogonal complement of $\mu_t$, which are guaranteed to reduce the volume of the sequence of ellipsoids $\varepsilon(\mu_t, \Sigma_t)$ exponentially fast. Here, the value of $k_t$ can be any number that satisfies $1 \leq k_t < (d+1)/2$, which is ideally determined by the application at hand. The queries’ response is, then, used to update the ellipsoidal approximation
using the batch-update formulas in Algorithm 2. Finally, the process terminates once convergence is reached.

It is worth noting that Algorithm 2 is a Markovian algorithm that encodes all of the relevant information about the past in its state variables \((\mu_t, \Sigma_t)\). In particular, queries are used only once and are discarded afterward. Moreover, it has a low computational overhead and enjoys a fixed space and time complexity at all rounds.

4.4 Experiments

As stated earlier, the proposed algorithm yields a significant saving in terms of space and time at the expense of an increase in the sample complexity. To recall, when the proposed algorithm was compared against the maximum-inscribed ellipsoid method described in Chapter 3, it was found that the new algorithm used less than 0.1% of the space and time resources used by the previous algorithm to achieve the same relative estimation error. The price for such a significant saving in space and time was a 5-fold increase in the sample complexity.

In this section, I present more experimental results. In each experiment reported here, the three methods of uncertainty sampling (baseline), the maximum-inscribed ellipsoidal method of Chapter 3 and the newly proposed algorithm in this chapter are compared. In all methods, a coefficient vector \(w^* \in \mathbb{R}^d\) was first generated uniformly at random from the surface of the unit sphere. Then, the different query synthesis methods were run in parallel to estimate the unknown \(w^*\). In the new proposed method, \(\mu_0\) is initialized to zero while \(\Sigma_0\) is initialized to \(I_{d \times d}\). The experiment was conducted for \(d = 100\) with a batch size of \(k = d/2\). Experiments were repeated multiple times and averages are reported.

The detailed performance results of this experiment are displayed in Fig. 4.1 and Fig. 4.2, which correspond to the sample complexity and time complexity respectively. In Fig. 4.1, the total number of rounds (vertical axis) is plotted against the estimation
Figure 4.1: The number of rounds is plotted against estimation error when the problem dimension is $d = 100$. In each round, a batch of 50 queries are synthesized.

error $||\hat{w} - w^*||_2$ (horizontal axis). In Fig. 4.2, the total cumulative time (vertical axis) is plotted against the total number of rounds (horizontal axis) that is made by each query synthesis algorithm.

As shown in Fig. 4.1, both the maximum-inscribed ellipsoid method and the algorithm proposed in this chapter yield an exponential gain in sample complexity, and they both outperform uncertainty sampling by several orders of magnitude. However, the algorithm proposed in this chapter is significantly superior to the two other methods in terms of its space and time complexity, as shown in Fig. 4.2. In particular, it enjoys a low computational overhead that is fixed at all rounds due to its simple update formulas. Hence, the new proposed method offers a middle compromise between the sample complexity on one hand, and the space and time complexity, on the other hand.

The linear curve in the log-scale plot in Fig. 4.1 shows that the sample complexity of the new proposed method is indeed $O(\log \frac{1}{\epsilon})$. To quantify the dependence on $d$, Fig. 4.2...
Figure 4.2: The total cumulative time is plotted against the number of queries, when a batch of 50 queries are synthesized at each round. Here, the problem dimension is $d = 100$. The new proposed method uses less than 50s to complete 3,000 rounds of the query synthesis algorithm, which yields an estimation error of $1E^{-8}$. By contrast, the other two methods take more than 3 hours to reach the same level of accuracy.

Figure 4.3: In this figure, the vertical axis is the number of rounds used by the proposed algorithm to achieve an estimation error of $1E^{-3}$, where the $x$-axis is the problem dimension $d$. The top plot corresponds to a batch size of 10, whereas the bottom plot corresponds to a batch size of 20.
4.3 plots the total number of queries made by the algorithm to achieve an estimation error of 1E-3 against the problem dimension \( d \). The top plot in Fig. 4.3 uses a batch size of 10, while the bottom plot uses a batch size of 20. Both plots suggest that the sample complexity is quadratic on \( d \), and that batch-mode query synthesis is effective when the batch size is not too big (i.e. if it satisfies \( 1 \leq k < (d + 1)/2 \)). More specifically, increasing the batch size by factor of two reduced the required number of rounds by a factor of two. In general, the number of rounds required to achieve an estimation error of less than \( \epsilon \) is \( O(\frac{d^2}{k} \log \frac{1}{\epsilon}) \) in practice, where \( k \) is the batch size. These empirical results are explained analytically in the following section.

### 4.5 Analytical Justification

We analyze the algorithm next. First, we show that \( \varepsilon(\mu_t, \Sigma_t) \) is indeed a valid ellipsoid in \( \mathbb{R}^d \). That is, we will prove that the matrices \( \Sigma_t \) satisfy \( \Sigma_t \succ 0 \) for all \( t \geq 0 \). Second, we show that the version space \( V_t \) satisfies \( V_t \subseteq \varepsilon(\mu_t, \Sigma_t) \). In other words, all the hypotheses \( w \in \mathbb{R}^d \) that are consistent with all of the queries that have been seen by round \( t \) will remain in our search space at round \( t + 1 \). Finally, we show that the sequence of enclosing ellipsoids \( \varepsilon(\mu_t, \Sigma_t) \), which contain \( V_t \), decrease in volume exponentially fast.

To achieve this, we begin with the following proposition.

**Proposition 1.** The sequence of matrices \( \Sigma_t \) in Algorithm 2 satisfy \( \Sigma_t \succ 0 \) for all \( t \geq 0 \).

**Proof.** The proof is by induction. First, we note that \( \Sigma_0 \succ 0 \) by definition of \( S_{++}^d \) so the base case trivially holds. Assume \( \Sigma_{t-1} \succ 0 \) and let \( v \neq 0 \) be any non-zero vector in \( \mathbb{R}^d \). We have:

\[
v^T \Sigma_t v = \frac{d^2 + k_t - 1}{d^2 - 1} \left[ v^T \Sigma_{t-1} v - \frac{2}{d+1} \sum_{i=1}^{k_t} \frac{(v^T \Sigma_{t-1} x_i)^2}{x_i^T \Sigma_{t-1} x_i} \right]
\]
However, because $\Sigma_{t-1} \succ 0$ by the inductive hypothesis, it induces an inner product. So, we have $(v^T \Sigma_{t-1} x)^2 \leq (v^T \Sigma_{t-1} v) \cdot (x^T \Sigma_{t-1} x)$ by the Cauchy-Schwarz inequality. Plugging this into the last equation yields:

$$v^T \Sigma_{t} v \geq \frac{d^2 + k_t - 1}{(d + 1)^2} \frac{d + 1 - 2k_t}{d + 1} v^T \Sigma_{t-1} v$$

Because $1 \leq k_t < (d + 1)/2$ and $v^T \Sigma_{t-1} v > 0$ for any $v \neq 0$ by the inductive hypothesis $\Sigma_{t-1} \succ 0$, we obtain $v^T \Sigma_{t} v > 0$. Since this holds for any $v \neq 0$, it proves that $\Sigma_{t} \succ 0$. 

The previous proposition shows that $\varepsilon(\mu_t, \Sigma_t)$ is indeed a sequence of ellipsoids in $\mathbb{R}^d$. Next, we show that the version space $\mathcal{V}_t$ is always contained inside the ellipsoid $\varepsilon(\mu_t, \Sigma_t)$.

**Theorem 5.** Let $\mathcal{V}_t$ be the set of all hypotheses $w \in \mathbb{R}^d$ that are consistent with all queries by the end of round $t$ and are contained in $\varepsilon(\mu_0, \Sigma_0)$. Then, we have $\mathcal{V}_t \subseteq \varepsilon(\mu_t, \Sigma_t)$ for all $t \geq 0$.

**Proof.** The proof is inspired by a similar proof to the Khachiyan update formulas [121]. It is also a proof by induction. We know that the statement of the theorem holds if we can guarantee that for all $w \in \varepsilon(\mu_{t-1}, \Sigma_{t-1})$ that are consistent with all new queries $\{x_1, \ldots, x_{k_t}\}$ at round $t$, we have $w \in \varepsilon(\mu_t, \Sigma_t)$. In other words, consistent hypotheses always remain in our search space. Because the statement trivially holds at $t = 0$, the theorem follows by induction afterward.

The proof relies on two facts. First, we note that the batch of queries $\{x_1, \ldots, x_{k_t}\}$ at round $t$ are always orthonormal, by construction, so they satisfy:

$$x_i^T x_j = \delta_{ij}, \quad (4.2)$$
where $\delta_{ij}$ is the Kronecker delta function. In addition, they also satisfy:

$$x^T \mu_{t-1} = 0, \quad \forall x \in \{x_1, \ldots, x_{k_t}\} \quad (4.3)$$

In other words, all queries have a zero-margin with respect to the current hypothesis $\mu_{t-1}$, which is inline with the spirit of uncertainty sampling.

Second, because convex bodies are mapped to convex bodies under affine transformations, we can restrict attention to the case when $\Sigma_{t-1} = I_{d \times d}$ with no loss of generality. An affine transformation always exists such that $\Sigma_{t-1} \to I_{d \times d}$ because an ellipsoid is, by definition, the image of a unit ball under some affine transformation. However, if $\Sigma_{t-1} = I_{d \times d}$, then we have by the matrix inversion lemma [124]:

$$\Sigma_t^{-1} = \frac{d^2 - 1}{d^2 + k_t - 1} \left[ I + \frac{2}{d - 1} \sum_{i=1}^{k_t} x_i x_i^T \right]$$

Here, again, we used the fact that queries satisfy Eq. (4.2). Because $w \in \epsilon_{t-1}$ is equivalent to $\|w - \mu_{t-1}\|_2^2 \leq 1$ when $\Sigma_{t-1} = I_{d \times d}$, the statement of the theorem follows if we can show that the following two conditions on $w \in \mathbb{R}^d$:

$$\|w - \mu_{t-1}\|_2^2 \leq 1 \land \forall x_j \in \{x_1, \ldots, x_{k_t}\} : y_j \langle w, x_j \rangle \geq 0 \quad (4.4)$$

imply necessarily that:

$$(w - \mu_t)^T \Sigma_t^{-1} (w - \mu_t) \leq 1 \quad (4.5)$$

The first condition in Eq. (4.4) states that $w \in \epsilon_{t-1}$, i.e. $w$ is a candidate hypothesis at round $t - 1$. The second condition in Eq. (4.4) states that $w$ is consistent with all of the new queries at round $t$. The consequent in Eq. (4.5) states that $w$ remains in our search space at the next round.

Using the update formulas in Algorithm 2 it can be shown with some algebraic
manipulations that:

\((w - \mu_t)^T \Sigma_t^{-1} (w - \mu_t) = \frac{d^2 - 1}{d^2 + k_t - 1} \cdot \left[ \|w - \mu_{t-1}\|_2^2 - 1 \right] + \frac{2}{d - 1} \sum_{i=1}^{k_t} M_i (M_i - 1) + 1,\)

where \(M_i = y_i x_i^T (w - \mu_{t-1})\). Because \(x_i^T \mu_{t-1} = 0\), we have \(M_i \geq 0\) for all hypotheses \(w\) that are consistent with the query-response \((x_i, y_i)\). Since \(\|w - \mu_{t-1}\|_2 \leq 1\) and \(\|x_i\|_2 = 1\), we apply the Cauchy-Schwarz inequality to deduce that \(M_i \leq 1\). Therefore, we deduce that \(M_i (M_i - 1) \leq 0\) for all \(i \in \{1, 2, \ldots, k_t\}\) if \(w\) is consistent with all new queries. The latter inequality along with the condition \(\|w - \mu_{t-1}\|_2 \leq 1\) imply that:

\((w - \mu_t)^T \Sigma_t^{-1} (w - \mu_t) \leq 1,\)

which is the desired result.

Theorem 5 shows that if \(w^*\) is contained inside the original ellipsoid \(\varepsilon(\mu_0, \Sigma_0)\), then it remains in \(\varepsilon(\mu_t, \Sigma_t)\) for all \(t \geq 0\). The following proposition reveals that the sequence of ellipsoids \(\varepsilon(\mu_t, \Sigma_t)\) shrinks in volume exponentially fast with each round.

**Theorem 6.** Let \(\varepsilon(\mu_t, \Sigma_t)\) be the sequence of ellipsoids generated by the algorithm in Algorithm 2. Then:

\[ \det \Sigma_T \leq (\det \Sigma_0) \cdot \exp \left\{ -\frac{1}{4(d + 1)} \sum_{t=1}^{T} k_t \right\}, \]

where \(1 \leq k_t < (d + 1)/2\) is the batch size at round \(t\).

**Proof.** Because the ratio of determinants is invariant to affine transformations [121], we can assume that \(\Sigma_{t-1} = I_{d \times d}\) with no loss of generality. Again, an affine transformation always exists such that \(\Sigma_{t-1}\) is mapped to \(I_{d \times d}\). Using the update rules in
Algorithm 2

\[
\frac{\det \Sigma_t}{\det \Sigma_{t-1}} = \left(\frac{d^2 + k_t - 1}{d^2 - 1}\right)^d \det \left( I - \frac{2}{d+1} \sum_{i=1}^{k_t} x_i x_i^T \right)
\]

Next, because \(x_i\) are orthonormal, they are a subset of a complete basis \(\{x_1, \ldots, x_d\}\) in \(\mathbb{R}^d\). Hence, we write the above as:

\[
\frac{\det \Sigma_t}{\det \Sigma_{t-1}} = \left(\frac{d^2 + k_t - 1}{d^2 - 1}\right)^d \det \left( \sum_{i=1}^{k_t} \frac{d-1}{d+1} x_i x_i^T + \sum_{i=k_t+1}^{d} x_i x_i^T \right)
\]

The determinant is the product of eigenvalues. So, we have:

\[
\frac{\det \Sigma_t}{\det \Sigma_{t-1}} = \left(\frac{d^2 + k_t - 1}{d^2 - 1}\right)^d \left(\frac{d-1}{d+1}\right)^{k_t}
= \left(1 + \frac{k_t}{d^2 - 1}\right)^d \left(1 - \frac{2}{d+1}\right)^{k_t}
\]

(4.6)

Finally, we use the identity \(\forall \gamma \in \mathbb{R} : 1 + \gamma \leq e^\gamma\):

\[
\frac{\det \Sigma_t}{\det \Sigma_{t-1}} \leq \exp \left\{ \frac{k_t d}{d^2 - 1} - \frac{2k_t}{d+1} \right\} = \exp \left\{ - \frac{k_t (d-2)}{d^2 - 1} \right\}
\leq \exp \left\{ - \frac{1}{4} \frac{k_t (d-1)}{d^2 - 1} \right\} = \exp \left\{ - \frac{1}{4} \frac{k_t}{d+1} \right\}
\]

In the last line, we assumed that \(d > 2\) so \(d - 2 \geq (d - 1)/4\). Thus, the statement of the proposition immediately follows when \(d > 2\). To handle the case when \(d = 2\), we use the exact expression in Eq. (4.6), which gives:

\[
\frac{\det \Sigma_t}{\det \Sigma_{t-1}} = \frac{(1 + \frac{k_t}{d})^{3k_t}}{3^{k_t}} \leq \left(\frac{e}{3}\right)^k_t = \exp \left\{ - (\log 3 - 1) k_t \right\}
\leq \exp \left\{ - \frac{1}{4} \frac{k_t}{d+1} \right\}, \quad d = 2
\]

Therefore, the proposition holds for all \(d > 1\). \(\square\)
Theorem 6 explains the sample complexity of $O(d^2 \log \frac{1}{\epsilon})$ that has been observed in practice, as mentioned in the previous section. First, we know that the set of $\epsilon$-accurate hypotheses has the volume $O(\epsilon^d)$ whereas the volume of the ellipsoidal approximation decreases exponential fast with the rate given by Theorem 6. Assuming, heuristically speaking, that the set of $\epsilon$-accurate hypotheses largely remain in our search space, the number of queries required for the ellipsoidal approximation to be entirely contained in the set of $\epsilon$-accurate hypotheses is, then, $O(d^2 \log \frac{1}{\epsilon})$.

4.6 Conclusions

In this chapter, a second query synthesis algorithm for learning halfspaces is developed. It is a Markovian algorithm, which encodes all of the relevant information about the past in its present state. That is, a query is used only once, and is immediately discarded afterward. Consequently, the new algorithm enjoys a fixed space and time complexity at all rounds. In fact, the proposed algorithm utilizes a variant of the classical Khachiyan’s update formulas for solving linear programs, which can be computed very efficiently. However, unlike the previous algorithm, whose sample complexity is $O(d \log \frac{1}{\epsilon})$, the sample complexity of this algorithm is shown to be $\tilde{O}(d^2 \log \frac{1}{\epsilon})$. 
Algorithm 2: The proposed Markovian query synthesis algorithm for learning homogeneous halfspaces.

Data:
1. Initial values $\mu_0 \in \mathbb{R}^d$ and $\Sigma_0 \in \mathbb{S}^d_{++}$ such that $w^* \in \varepsilon(\mu_0, \Sigma_0)$.
2. Sequence of positive integers $(k_t)_{t=1,2,\ldots}$, where $1 \leq k_t < (d + 1)/2$ is the batch size at round $t$.
3. Tolerance $\epsilon > 0$.

Result: An estimate $\hat{w} \in \mathbb{R}$ to the true coefficient vector $w^* \in \mathbb{R}^d$.

Begin:
1. Set $t = 0$.
2. Repeat until convergence:
   - $t \rightarrow t + 1$
   - Compute $N_{t-1} \in \mathbb{R}^{d \times (d-1)}$, which is an orthonormal basis to the orthogonal complement of $\mu_{t-1}$.
   - Draw $k_t$ orthogonal vectors at random from the surface of the unit sphere in $\mathbb{R}^{d-1}$. Let $\{\alpha_1, \ldots, \alpha_{k_t}\}$ be the chosen vectors.
   - Let $x_i = N_{t-1} \alpha_i$ for all $i = 1, 2, \ldots, k_t$ be the batch of synthetic queries at round $t$.
   - Query for the labels of $\{x_1, x_2, \ldots, x_{k_t}\}$. Let $y_i = \text{sign}\langle x_i, w^* \rangle$ be the responses.
   - Perform the batch updates:
     \[
     \begin{align*}
     \mu_t &= \mu_{t-1} + \frac{1}{d + 1} \sum_{j=1}^{k_t} y_j \frac{\Sigma_{t-1} x_j}{\|x_j\| \Sigma_{t-1}} \\
     \Sigma_t &= \frac{d^2 + k_t - 1}{d^2 - 1} \left[ \Sigma_{t-1} - \frac{2}{d + 1} \sum_{j=1}^{k_t} \frac{\Sigma_{t-1} x_j x_j^T \Sigma_{t-1}}{\|x_j\|^2 \Sigma_{t-1}} \right]
     \end{align*}
     \]
3. Return: $\hat{w} = \mu_t/\|\mu_t\|_2$
Chapter 5

Extensions of the Query Synthesis Algorithms

5.1 Introduction

The query synthesis algorithms proposed in Chapter 3 and Chapter 4 are designed for noise-free homogenous halfspaces. In this chapter, we briefly describe how such algorithms can be extended to other settings as well. For instance, we will present an extension of the query synthesis algorithms to non-homogenous halfspaces, and describe how the ellipsoidal approximation approach can be extended to pool-based active learning.

5.2 Non-Homogenous Halfspaces

5.2.1 Derivation

We begin by describing how the query synthesis algorithms proposed in the previous chapters can be extended to non-homogenous halfspaces of the form:

\[ c^* = \{ x : \langle w^*, x \rangle + b^* \geq 0 \} \]  \hspace{1cm} (5.1)

Let \( \varepsilon = (\mu, \Sigma) \) be an ellipsoidal approximation to the version space at some round \( t \). Then, similar to what was done before, we would like to synthesize queries that eliminate half of the ellipsoid \( \varepsilon \) while being aligned with the principal axes of \( \Sigma \) as much as possible. Unlike the previous case, however, we now interpret a query \( \tilde{x} \) as a \( (d+1) \)-dimensional vector whose last component is fixed to \( \tilde{x}_{d+1} = 1 \). In this
case, a non-homogenous halfspace of the form given in Eq. 5.2 can be written as a homogenous halfspace of the form:

$$c^* = \{ \bar{x} : \langle \bar{w}, \bar{x} \rangle \geq 0 \}$$

(5.2)

where $\bar{w} \in \mathbb{R}^{d+1}$ and $\bar{w}_{d+1} = b$ is the bias term of the non-homogenous halfspace.

In light of Lemma 1 and the subsequent discussion in Chapter 3 we will synthesize queries that maximize $x^T \Sigma x$ subject to the two constraints:

$$\mu^T x = 0, \quad x_{d+1} = 1$$

Both constraints can be written in the compact matrix notation form $Bx = c$, where:

$$c = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad B = \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_{d+1} \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

(5.3)

However, because $c \neq 0$, a straightforward extension of the previous algorithms would lead to optimization problems that are now intractable.

To make the query synthesis algorithm computationally tractable, we decompose the query $x$ into two components $x = s + u$, which are orthogonal with respect to the inner product induced by the positive definite matrix $\Sigma$. In other words, $s^T \Sigma u = 0$. We will select $s$ in a way that handles the bias term of the non-homogenous halfspace and select $u$ to align the query $x$ with the principal axes of $\Sigma$. The overall algorithm is shown in Algorithm 3.

The first part of Algorithm 3 performs two steps. It computes an ellipsoidal approximation to the version space using the maximum-determinant convex optimization formulation derived earlier in Chapter 3. We write $\varepsilon = (\mu, \Sigma)$ to denote the

---

1 In Algorithm 3, it is assumed that $B$ is full-rank, otherwise the constraints are not satisfiable. In particular, this is equivalent to the condition that $\exists j \in \{1, \ldots, d\} : \mu_j \neq 0.$
Algorithm 3: The proposed query synthesis algorithm for non-homogenous halfspaces.

Data: Observations \( \{(x_i, y_i)\}_{i=1,2,...,t} \)
Result: \( k \) synthetic queries \( \{x_{t+1}, x_{t+2}, \ldots, x_{t+k}\} \)

Begin:

1. Solve the optimization problem in (3.6) or (3.7). Let \( \mu \) and \( \Sigma \) be the optimal solutions.
2. Set \( s = B^T(BB^T)^{-1}c \), where \( B \) and \( c \) are given by Eq. 5.3.
3. Let:
   \[
   \hat{B} = \begin{bmatrix}
   \mu_1 & \mu_2 & \cdots & \mu_{d+1} \\
   0 & 0 & \cdots & 1 \\
   s_1 & s_2 & \cdots & s_{d+1}
   \end{bmatrix}
   \]
4. Compute \( N \), which is the orthonormal basis to the null-space of \( \hat{B} \).
5. Compute \( \alpha_1, \alpha_2, \ldots, \alpha_k \), which are the top \( k \) eigenvectors of the matrix \( N^T \Sigma^* N \).
6. Compute \( u_{t+1} = N\alpha_1, \ldots, u_{t+k} = N\alpha_k \).
7. Return \( x_{t+1} = s + u_{t+1}, \ldots, x_{t+k} = s + u_{t+k} \).

solution. Also, it computes \( s \), which is the first component of our query \( x = s + u \). Here, \( s \) is the minimum norm solution that satisfies the constraints \( Bs = c \). It has the closed-form expression:

\[
\begin{align*}
   s &= B^T(BB^T)^{-1}c = \begin{bmatrix} \mu & e_{d+1} \end{bmatrix} \begin{bmatrix} \mu^T \mu & \mu_{d+1} \\ \mu_{d+1} & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
   &= \begin{bmatrix} \mu & e_{d+1} \end{bmatrix} \frac{1}{\mu^T \mu - \mu_{d+1}^2} \begin{bmatrix} 1 & - \mu_{d+1} \\ - \mu_{d+1} & \mu^T \mu \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
   &= \frac{1}{\mu^T \mu - \mu_{d+1}^2} \left( - \mu_{d+1} \mu + (\mu^T \mu) e_{d+1} \right),
\end{align*}
\]

where \( e_j \) is the standard unit basis, which is zero everywhere except at position \( j \). In fact, any solution \( s \in \mathbb{R}^d \) that satisfies \( Bs = c \) here suffices. Throughout the sequel,
however, we assume that $s$ is the minimum norm solution to the system of equations $Bs = C$.

Since $s^T \Sigma u = 0$, finding a query $x$ that maximizes $x^T \Sigma x$ is equivalent to finding $u$ that maximizes $u^T \Sigma u$ subject to the orthogonality constraint $s^T \Sigma u = 0$. This can be cast as an eigenvalue problem. We have:

$$\begin{align*}
\text{maximize} & \quad u^T \Sigma u \\
\text{subject to} & \quad \mu^T u = 0, \ u_d+1 = 0, \ s^T \Sigma u = 0 \\
& \quad \|u\|_2 = 1
\end{align*}$$

We can place all of the three linear equality constraints into a matrix $\tilde{B}$, as described in Algorithm 3:

$$\begin{align*}
\text{maximize} & \quad u^T \Sigma u \\
\text{subject to} & \quad \tilde{B} u = 0 \\
& \quad \|u\|_2 = 1
\end{align*}$$

Writing $N$ for the orthonormal basis to the null space of $\tilde{B}$, we can solve the optimization problem above by solving the eigenvalue problem:

$$\begin{align*}
\text{maximize} & \quad \alpha^T N^T \Sigma N \alpha \\
\text{subject to} & \quad \|\alpha\|_2 = 1
\end{align*}$$

If $\alpha_1, \ldots, \alpha_k$ are the top eigenvectors of the matrix $N^T \Sigma N$, then the $k$ queries are $s + N\alpha_1, s + N\alpha_2, \ldots, s + N\alpha_k$.

This provides an extension of the query synthesis algorithms in previous chapters to non-homogenous halfspaces.
5.2.2 Experiments

Next, we demonstrate that the proposed query synthesis algorithm for non-homogenous halfspaces achieves an optimal sample complexity in practice. In our experiments, we used the Second-Order Cone Programming (SOCP) formulation in (3.7) because it is more efficient computationally. The optimization problem was solved using CVX [120], where the MATLAB implementation code is provided in Appendix D.

In the experiment, a vector $w \in \mathbb{R}^d$ and a bias $b \in \mathbb{R}$ were generated at random from the standard multivariate normal distribution. One positive and one negative examples were computed afterward. Then, the query synthesis algorithm in Algorithm 3 was executed for up to a maximum of 1,000 queries. At each round, the algorithm synthesized a batch of five queries, and the estimation error was computed. This experiment was repeated multiple times, and averages are reported.

The results are shown in Fig. 5.1, which plots the estimation error vs. the number of queries. As shown in the figure, Algorithm 3 indeed achieves an exponential reduction in the sample complexity. In fact, the sample complexity is $O(d \log \frac{1}{\epsilon})$, which is optimal as proved earlier in Theorem 4.

5.3 Pool-Based Active Learning: An Aggressive Approach

Second, we extend the query synthesis algorithms into the pool-based active learning setting. To recall, whereas query synthesis has a significant computational advantage, since it can achieve an exponential reduction in the sample complexity in a computationally efficient manner, pool-based active learning has an advantage over query synthesis in that its queries are interpretable by humans. Hence, pool-based active learning is useful when labeling is carried out manually by human oracles.

The canonical approach for pool-based active learning is uncertainty sampling. For halfspaces, this is often translated into margin-based sampling, where the instances with the least margin are chosen as queries. Margin-based active learning
Figure 5.1: In this figure, the estimation error is plotted in a log-scale against the number of queries for the query synthesis algorithm in Algorithm 3. Here, a batch of \( k = 5 \) queries are synthesized at each round. The ‘◦’ curve is for \( d = 25 \). The ‘□’ curve is for \( d = 50 \). Finally, the ‘♦’ curve is for \( d = 75 \).

can be shown to yield an exponential saving in the sample complexity if the pool size grows exponentially large with each round [106]. In this section, we show how the earlier query synthesis algorithms can be interpreted as aggressive forms of uncertainty sampling, which goes well beyond the margin in computing uncertainty. After that, we use the ellipsoidal approximation method to devise a pool-based active learning algorithm that is more efficient in its sample complexity than margin-based active learning.

5.3.1 Description

We assume we start again with an ellipsoidal approximation to the version space \( \varepsilon = (\mu, \Sigma) \) in the realizable, noise-free homogenous halfspace setting. Suppose that the optimal classifier is uniformly distributed over the ellipsoid \( \varepsilon \). Given an instance \( x \in \mathbb{R}^d \), the probability it is classified as +1 by the optimal classifier (i.e. the probability
its true label is +1 in the realizable setting) can be calculated as follows. We have:

\[
p(y = +1 | x, \mu, \Sigma) = \int p(w | \mu, \Sigma) \cdot p(y = +1 | x, w) \, dw \\
= \int p(w | \mu, \Sigma) \cdot I\{\langle x, w \rangle > 0\} \, dw \\
= \int p(w | \mu, \Sigma) \cdot I\{\langle x, w - \mu \rangle > -\mu^T x\} \, dw \\
= \int p(z | 0, \Sigma) \cdot I\{\langle x, z \rangle > -\mu^T x\} \, dw
\]

Now, we note that for any fixed \( x \in \mathbb{R}^d \) and \( z \sim \mathcal{N}(0, \Sigma) \), the inner product \( \langle x, z \rangle \) is itself a Gaussian random variable with zero mean and covariance \( x^T \Sigma x \). Therefore, the solution is:

\[
p(y = +1 | x, \mu, \Sigma) = 1 - \Phi(-\frac{\mu^T x}{\sqrt{x^T \Sigma x}}) = \Phi(\frac{\mu^T x}{\sqrt{x^T \Sigma x}})
\]

Hence, given a pool of queries, the query with the most uncertainty is the query that minimizes \( |\mu^T x|/(x^T \Sigma x) \). Without the covariance matrix, i.e. when it is isotropic, this sampling scheme reduces to margin-based sampling.

Looking back into the earlier algorithms in the previous chapters, we note that they can be interpreted as aggressive forms of uncertainty sampling. More precisely, since the quantity \( |\mu^T x|/(x^T \Sigma x) \) is non-negative, it is minimized when \( \mu^T x = 0 \). If we view the margin \( \mu^T x \) as infinitesimally small, then the greatest uncertainty is achieved by queries that maximize \( x^T \Sigma x \) subject to the constraint \( \mu^T x = 0 \). However, this is precisely what Algorithm [1] in Chapter [3] does.

Besides, this suggests a pool-based active learning algorithm, which iterates between two steps:

1. Ellipsoidal Approximation: It computes an ellipsoidal approximation to the ver-
sion space given the previous queries. This is implemented using the maximum-determinant convex optimization problem, described earlier in Chapter 3.

2. Query Selection: Given a pool of instances, it selects the query that minimizes $|\mu^T x|/(x^T \Sigma x)$. When multiple instances satisfy $\mu^T x = 0$, the instance that maximizes $x^T \Sigma x$ is selected.

We examine the performance of this pool-based active learning algorithm experimentally in the following section.

5.3.2 Experiments

Next, we demonstrate how the pool-based active learning algorithm described in the previous section outperforms margin-based active learning. The exact experiment runs as follows. For a fixed $d$ (e.g. $d = 25$ in our experiment), we generate a coefficient vector $w^* \in \mathbb{R}^d$ from the unit sphere. Then, one positive and one negative examples are computed. After that, the pool-based active learning algorithms run sequentially in rounds. At round $t$, a pool of size $O(t)$ (e.g. $100t$ in our experiment) is generated at random from $\mathcal{N}(0, I_d)$. Each pool-based active learning algorithm selects one query $x_t$ from the pool and receive its response $y_t = \text{sign}(\langle w^*, x_t \rangle)$. Based on these responses, the algorithm computes an estimate $\hat{w}$ to the true unknown coefficient vector $w^*$.

Finally, the estimation error $||\hat{w} - w^*||_2$ is recorded. The same experiment is repeated multiple times and averages are reported.

The three pool-based active learning algorithms used in our experiment are the following:

1. Random Sampling: This is a baseline method. At each round, the query is selected uniformly at random from the given pool. Then, Support Vector Machine (SVM) is implemented on all the queries to produce an estimate $\hat{w}$. After that, the error $||\hat{w} - w^*||_2$ is recorded.
Figure 5.2: In this figure, the estimation error is plotted in a log-scale against the number of queries for different pool-based active learning algorithms. Here, the problem dimension is $d = 25$ and the pool size at round $t$ is $100t$. The ‘$\circ$’ curve is for random sampling (baseline method). The ‘$\square$’ curve is for margin-based sampling. Finally, the ‘$\Diamond$’ curve is for covariance-based sampling criteria proposed in Section 5.3.1.

2. Margin-based Sampling: In this method, SVM is implemented on the previous queries, which provides a solution $w \in \mathbb{R}^d$. After that, the query with the least absolute margin $|w^T x|$ is selected. Once the query is selected, SVM is implemented on all the queries to produce an estimate $\hat{w}$. The error $||\hat{w} - w^*||_2$ is, then, recorded.

3. Covariance-Based Sampling: This is the sampling scheme described in the previous section. At each round, an ellipsoidal approximation is computed using the maximum-determinant convex optimization problem in (3.7). Let $(\mu, \Sigma)$ be the solution. After that, the query in the pool that minimizes $|\mu^T x|/(x^T \Sigma x)$ is selected. Once the query is selected, the maximum inscribed ellipsoid method is implemented on all the queries to produce an estimate $\hat{w}$. The error $||\hat{w} - w^*||_2$ is, then, recorded.

The results are shown in Fig. 5.2. As shown in the figure, the new pool-based...
active learning criteria, which is the analog of the query synthesis algorithms described in the previous chapters, indeed outperforms margin-based sampling by several orders of magnitude.

5.4 Conclusions

In this chapter, I described how the query synthesis algorithms in Chapter 3 and Chapter 4 could be extended to new settings as well. For instance, I derived a new algorithm that extended membership query synthesis to non-homogenous halfspaces, and demonstrated experimentally that the new algorithm also achieved an exponential saving in its sample complexity. Similarly, I described how the ellipsoidal approximation approach could be extended to pool-based active learning by proposing a sampling criteria, which incorporated both the margin and the covariance of uncertainty. Experimentally, it achieves a higher accuracy than margin-based sampling by several orders of magnitude.
Part III

Mitigating the Adversarial Risk of Reverse Engineering
Chapter 6

Learning a Distribution of Linear Classifiers

6.1 Introduction

In the previous chapters, I presented several algorithms for learning linear classifiers using query synthesis. Because the algorithms are quite efficient in their computational and sample complexities, they can be quite beneficial in active learning environments, such as automated science. However, these advantages also imply that the proposed algorithms can be quite beneficial in reverse-engineering linear classifiers in adversarial environments, which poses a severe security threat.

In practice, many machine learning algorithms have been successfully implemented for security-sensitive applications. These include spam and malicious email behavior detection \[28, 30, 29\], fraud detection \[32\], as well as intrusion detection \[33, 34, 35, 36, 75, 76, 77\]. The machine learning techniques employed cover a broad spectrum ranging from supervised learning algorithms, such as neural networks and \(k\)-Nearest Neighbor (k-NN), to unsupervised outlier detection algorithms, such as density estimation-based methods and one-class Support Vector Machine (SVM). Linear classifiers, including SVM and logistic regression, are also commonly used in practice today \[69, 125, 30\].

Unfortunately, however, security-sensitive applications are characterized by the presence of adversaries, and most prominent machine learning algorithms were not originally designed for such adversarial environments \[78, 79, 80, 81, 82\]. In the standard general setting of learning, described in Chapter 1, machine learning algorithms
assume that training data as well as unforeseen future data are generated i.i.d. from
the same underlying distribution. Hence, by building a learning algorithm that works
well for a given training dataset and by restricting its capacity to avoid over-fitting,
the learning algorithm is expected to work well with unforeseen future examples as
well. In security-sensitive applications, where an adversary can interfere either dur-
ing or post training, such assumptions are no longer valid. Indeed, not only can
adversaries interfere during the training stage by injecting “poisonous” training ex-
amples \cite{78, 126, 127}, but they can also interfere after the training phase concludes by
reverse-engineering the classifier, and using this newly-gained knowledge about the
decision rule in order to change tactics and avoid detection. We will return to these
two different types of attacks shortly.

The economic incentives for adversaries to mislead learning algorithms cannot be
overstated. For instance, it is not unusual for spammers to generate revenues that
exceed 20% of the value of the products sold through spam emails \cite{31}. Carpinter
and Hunt interpret this by noting that a response rate of only 0.001% to a single
spam email advertising a $50-product could generate more than $25,000 of returns
to spammers due to the sheer number of Internet users \cite{31}. Furthermore, it has
been estimated that e-commerce fraud activities may generate over $10 billion dol-
lars worldwide per annum in addition to the $1 billion dollars made annually from
telecommunications fraud \cite{32}.

Driven by ample economic incentives, an adversary can generate attacks against a
learning system, such as spam filters, either during the training phase or post training
when the system is operational. These two different realms of attacks have been
coined causative and exploratory attacks respectively \cite{128}. One common example of
a causative attack that is carried out during the training phase is to inject carefully-
crafted “poisonous” training examples in order to mislead the learning system \cite{78,
126, 127, 129}. By contrast, a typical example of an exploratory attack is evasion such
as the “good-word attack”, where words indicative of a non-spam email are added to spam emails in order to evade the spam detection system [128, 130, 84]. Ideally, machine learning algorithms ought to be robust against both types of attacks.

Whereas defense strategies against causative attacks have recently received a growing interest in the machine learning community (see for reference Section 6.2), less effort has been devoted towards protecting a classifier against exploratory attacks. The exploratory attack that is of particular interest in this chapter is reverse engineering, whose risk has been demonstrated for linear classifiers in the previous chapters. In fact, the query synthesis algorithms proposed in those chapters are not necessary to establish how reverse engineering poses a security threat. To see this, consider classification tasks such as spam detection. On one hand, given that observations $\mathbf{z} = (\mathbf{x}, \mathbf{y})_{i=1,2,...}$ are generated i.i.d. from a fixed underlying distribution $\mathbb{P}(\mathbf{z})$, the learning task of the defender is to predict the label $\mathbf{y}$ once we know the instance $\mathbf{x}$. Of course, the defender can employ any of the state-of-the-art methods, such as SVM or neural networks, to achieve this goal. Let $\hat{\mathbf{y}}$ be the predicted label of the resultant classifier. On the other hand, for an adversary that can probe the classifier with queries $\mathbf{x}_q$ and observe $\hat{\mathbf{y}}_q$, the new pair $\mathbf{z}_q = (\mathbf{x}_q, \hat{\mathbf{y}}_q)$ forms a new learning problem. Similar to the defender, the adversary can employ any of the state-of-the-art methods to learn $\hat{\mathbf{y}}_q$. Once the adversary learns to predict $\hat{\mathbf{y}}_q$ with sufficient accuracy, exploratory attacks such as evasion become easier to carry out.

The dilemma with reverse engineering is two-fold. First, development of new powerful machine learning algorithms not only facilitate the task of detecting intrusive behavior, but they also facilitate the task of reverse engineering the classifier by adversaries. To reiterate, this is because reverse engineering is another learning problem for which any learning algorithm can be employed\(^1\). Second, once the classifier is up

\(^{1}\text{Note that the adversary does not need to know the exact feature space or the learning algorithm used by the classifier. Instead, the adversary can go through the proper process of feature selection and model selection in order to estimate } \hat{\mathbf{y}}_q.\)
and running, it has been designed at the outset to perform well according to some pre-defined metrics such as precision, recall, or accuracy. Hence, while altering its behavior during operation can indeed make the reverse engineering task more difficult, it will likely lead to a loss of accuracy as a result, which indirectly translates into an adversarial gain.

Consequently, the goals of defending against reverse engineering can be summed up as follows. First, we would like to build at the outset a classification system that can always make reliable predictions while revealing as little information about its decision boundary as possible. Second, we would like to achieve this goal even if the adversary knows the system. In this paper, we show that a suitable method of randomization can meet these objectives. Specifically, instead of learning a fixed classifier, the defender can use the training data to infer a distribution of classifiers. During prediction, a single classifier is drawn at random from the learned distribution. In other words, the proposed algorithm belongs to the class of randomized predictors, which are often called the Gibbs classifiers [64, 131]. My contribution is to show that the problem of learning this distribution of classifiers, subject to the two conflicting constraints of enjoying a high accuracy and a large variance, can, in fact, be formulated as a Semidefinite Programming (SDP) problem, which can be solved quite efficiently.

The rest of the chapter is outlined as follows. First, we review the existing literature on adversarial learning and discuss where the reverse engineering problem fits into the literature. After that, we describe the reverse engineering problem in more details and formalize notation. In Section 6.3, I will show that a suitable method of randomization can indeed achieve the desired objectives. Finally, we conclude with a comprehensive evaluation that validates the proposed method.
6.2 Related Literature

The existing literature on adversarial learning can be broadly classified along two dimensions as depicted in Fig. 6.1. The first dimension is the type of the attack, whether it is causative or exploratory. The second dimension is the type of the proposed solution, which can be either an attack or a defense strategy. We review these four quadrants next.

6.2.1 Causative Attack Strategies

Causative machine learning attacks (see Section 6.1) have enjoyed a growing interest over the past 20 years. On the theoretical side, various frameworks have been used to analyze such attacks including statistical learning theory, game theory, and mathematical optimization. For example, [132] used the Probably Approximately Correct (PAC) framework to prove that in order to achieve a prediction accuracy...
that exceeds $1 - \epsilon$ in an adversarial environment, the adversary’s influence over the training data must be limited to, at most, $\epsilon/(1 + \epsilon)$ of the training examples only [132, 1]. In addition, causative attacks can be formalized as a game between the classifier and the adversary, in which each player seeks an optimal strategy knowing that its opponent would also seek an optimal strategy. Several lines of research have used such a game-theoretic setting [133, 134, 79, 135]. Moreover, [126] provided a gradient ascent method for poisoning attacks, in which the adversary could inject malicious data into the training dataset to mislead the SVM classifier [126]. Similarly, [136] proposed a method for adversarial label flipping attacks, in which they formulated an optimization framework for finding the label flips that minimize the classifier’s predictive accuracy.

On the practical side, [129] implemented attack strategies against the popular SpamBayes classifier and concluded that the performance of SpamBayes could severely degrade if the adversary gained access to less than 1% of the training dataset. Some of these attacks could be as simple as contaminating legitimate emails with a few spam terms before they were used for training. Also, [137] simulated many causative attacks against automatic polymorphic worm signature generation algorithms and concluded that even if training examples were always labeled correctly, the adversary could still obstruct learning by manipulating training examples. One example is red-herring, in which irrelevant patterns are added into worms during training so that the classifier focuses on such irrelevant patterns and ignores the real signatures.

6.2.2 Causative Defense Strategies

In general, both theoretical and practical work indicate that causative attacks against machine learning algorithms can be quite effective. In order to defend against such attacks, several methods have been proposed such as adversarial collective classification, ensemble methods, kernel matrix correction, robustness/regularization,
multiple-instance learning, and game-theoretic strategies, among others [1, 127, 138, 78, 134, 133, 79, 139, 140, 141, 135, 142, 143, 82]. For the sake of brevity, we only describe two simple, yet effective, defense strategies against many causative attacks.

The first strategy is called Reject On Negative Impact (RONI). In RONI, a preliminary screening of all training examples is conducted first, and those training examples that were found to be quite influential to the decision boundary are subsequently rejected [144]. Intuitively, the learning system should not be extremely dependent on any single training example, especially in adversarial environments. While simple, such a strategy was found to be effective at preventing some causative attacks [129, 144].

The second common defense strategy against causative attacks is robust statistics [145, 146]. In robust statistics, the hypothesis $h \in H$, which is inferred by a learning algorithm $L$, is designed to be insensitive to any slight changes in the training data. A prototypical example is to use the multivariate median as an estimate of central tendency, instead of using the ordinary mean [147]. Similarly, minimizing the $\ell_1$ loss in regression using Linear Programming (LP) is more robust to outliers than minimizing the $\ell_2$ loss using least squares regression [123]. Other approaches have also been popularized including trimming and Winsorizing, which are close in spirit to the RONI method described earlier [145].

6.2.3 Exploratory Attack Strategies

Unlike causative attacks, exploratory attacks are carried out while the system is operational. It is assumed that no learning takes place during such attacks. For example, an attacker might probe the classifier with queries in order to reveal some confidential information about the training dataset that was used by the system [1]. Perhaps the two most prominent examples of exploratory attacks are evasion and reverse engineering [148].
In an evasion attack, the adversary has an attack message $x$ that is likely to be detected by the classifier. Instead of sending $x$, the adversary creates a new message $x'$, which is similar to $x$ but can evade the detection system. Despite the fact that adversaries are constrained in how much they could camouflage their original messages, e.g. a spammer would still need to deliver the contents of the spam, evasion attacks have been successful. For example, [149] concluded through experiments that good-word attacks could be quite effective against statistical spam filters, such as naïve Bayes and maximum entropy filters [149]. In a different study, they also showed that evasion could be carried out quite efficiently, i.e. in polynomial time, for linear classifiers, while [130] extended this result to the general class of convex-inducing classifiers [150, 151, 130, 144]. A simple method relies on the sign witness test, whereby an adversary can infer with relative ease whether a feature contributes positively or negatively to the classifier’s final decision and uses this knowledge to evade the classifier [150, 148]. Linear classifiers were shown to be especially susceptible to this type of attacks.

Reverse engineering is a second important exploratory attack, in which the adversary attempts to model the classifier’s decision boundary. It might be used as a preliminary step towards more sophisticated exploratory attacks, or it can be used as a goal by itself if, for example, the decision boundary can reveal sensitive information [1]. In Chapter 3 I have shown that linear classifiers could, in general, be reverse-engineered quite efficiently via query synthesis. An illustration of this fact is depicted in Fig. 6.2 in two dimensions.

6.2.4 Exploratory Defense Strategies

The fourth and final quadrant in the literature of adversarial learning is the exploratory defense strategies. In a recent work, [128] analyzed the security of machine learning algorithms and described the problem of defending a classifier against ex-
The second proposed strategy is to increase the complexity of the hypothesis space so that the decision boundary becomes harder to reverse-engineer by the adversary. However, this approach, which we will explore later in Chapter 7, increases the risk for over-fitting. As mentioned earlier in Chapter 1, increasing the complexity of the hypothesis space will increase its Vapnik-Chervonenkis (VC) dimension. Hence, it will increase the generalization risk of the learning algorithm. The approach proposed in
this chapter, by contrast, is provably unsusceptible to the risk of over-fitting; it retains
the generalization ability of linear classifiers.

The third strategy is randomization. Instead of deterministically predicting a
binary label, the classifier could be trained to predict a probability distribution $P(y|x)$
and use such a probability estimate to pick a label $\hat{y} \in \{-1, 1\}$ through a randomized
mechanism. In a different line of work, [1] also listed randomization as a valid defense
strategy although little, if any, work has been conducted to validate this approach
[1]. In fact, [1] concluded that:

Randomization increases the adversary’s work, but it also will increase the
learner’s base error rate. Determining the right amount of randomization
is an open problem.

One objective of this chapter is to formalize the trade-off between accuracy and
randomization. A noteworthy result is that it is often possible to incorporate ran-
domization in order to increase the adversary’s reverse engineering effort, but without
increasing the learner’s classification error rate.

6.3 The Randomization Method

Because fixed linear classifiers can be reverse-engineered quite efficiently, we can mit-
igate the risk of reverse engineering by introducing noise. The rationale behind this
approach stems from recent seminal results in approximation hardness. In particular,
it has been shown that whereas a fixed noise-free halfspace can be reverse-engineered
quite easily, finding a halfspace that guarantees a prediction accuracy of, at least,
$\frac{1}{2} + \delta$ is NP-hard when the best possible halfspace only agrees with $1 - \epsilon$ of the queries
for any fixed $\epsilon, \delta > 0$ [86]. In other words, the slightest amount of noise alters the
computational complexity of the learning problem quite dramatically. Hence, noisy
halvespaces are significantly harder to reverse engineer than fixed noise-free halfspaces.
However, introducing noise to the classifier’s predicted label may degrade its performance. Thus, to mitigate the risk of reverse engineering without impacting the predictive accuracy of the classifier, the amount of noise introduced should be as large as possible, but not more! Because randomizing the predictor is equivalent to introducing noise, our goal is to maximize the variance of the distribution of classifiers subject to the constraint that any single classifier selected at random from such a distribution provides reliable predictions with a high probability. Once learning is concluded, the defender can mitigate the risk of reverse engineering attacks by picking a classifier at random for every query observed. We will illustrate how the method works using synthetic datasets and discuss how to interpret the accuracy-variance tradeoff curves afterward.

6.3.1 Derivation

Our starting point is the following problem. Suppose we have a system of linear inequalities $A w \leq b$, where $w \sim \mathcal{N}(\mu, \Sigma)$ is a multi-variate random vector whose covariance $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_n^2)$ is a diagonal matrix. Assume first that $\Sigma$ is fixed but $\mu$ can vary. Because $w$ is randomized, we seek a distribution $\mathcal{N}(\mu, \Sigma)$ such that the system of linear inequalities is satisfied with a probability that exceeds $\nu$, where $\frac{1}{2} < \nu < 1$, if $w$ is drawn at random from $\mathcal{N}(\mu, \Sigma)$. That is, we would like to find $\mu$ given $A, b, \Sigma$, and $\nu$. As mentioned earlier, this problem is computationally hard so we derive a convex relaxation instead.

First, if we let $\hat{a}_i^T$ be the $i$-th row of $A$, then the constraint:

$$\mathbb{P}_{w \sim \mathcal{N}(\mu, \Sigma)} (\hat{a}_i^T w \leq b_i) \geq \nu,$$
can be re-written as:

\[ P_{w \sim \mathcal{N}(\mu, \Sigma)} \left( \frac{\hat{a}_i^T (w - \mu)}{\sqrt{\hat{a}_i^T \Sigma \hat{a}_i}} \leq \frac{b_i - \hat{a}_i^T \mu}{\sqrt{\hat{a}_i^T \Sigma \hat{a}_i}} \right) \geq \nu \]

However, the random variable \( \frac{\hat{a}_i^T (w - \mu)}{\sqrt{\hat{a}_i^T \Sigma \hat{a}_i}} \) is distributed according to the standard Gaussian density. Therefore, the above constraint is equivalent to:

\[ \Phi \left( \frac{b_i - \hat{a}_i^T \mu}{\sqrt{\hat{a}_i^T \Sigma \hat{a}_i}} \right) \geq \nu, \]

where \( \Phi \) is the Cumulative Density Function (CDF) of the standard Gaussian density. Consequently, we have:

\[ \hat{a}_i^T \mu + \Phi^{-1}(\nu) \sqrt{\hat{a}_i^T \Sigma \hat{a}_i} \leq b_i \quad (6.1) \]

Since the only variable is \( \mu \), the entire problem can be reduced to a system of linear inequalities. Note that because we would like to satisfy a system of linear inequalities for an entire distribution, we ended up with a new system of linear inequalities that is stricter than the one we started with assuming \( \nu > \frac{1}{2} \).

Second, suppose that \( \Sigma \) is not fixed beforehand and that we would like to find a distribution \( \mathcal{N}(\mu, \Sigma) \) that both satisfies the constraints with a high probability and has a large variance. Assuming that \( w \in \mathbb{R}^d \), one convenient criterion that can be used to measure the spread of a multivariate Gaussian density is given by the following definition:

**Definition 11.** Let \( \mathcal{N}(\mu, \Sigma) \) be a multivariate Gaussian density with mean \( \mu \in \mathbb{R}^d \) and covariance \( \Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_d^2) \). We define its spread \( 0 \leq \chi \leq 1 \) using:

\[ \chi = \sqrt{\frac{\sum_{j=1}^{d} \sigma_j^2}{||\mu||_2^2 + \sum_{j=1}^{d} \sigma_j^2}} \quad (6.2) \]

The motivation behind choosing \( \chi \) as a measure of spread can be understood by
noting that if \((w_1, w_2, \ldots, w_t)\) is a sequence of random draws from \(\mathcal{N}(\mu, \Sigma)\), then:

\[
\chi = \sqrt{\frac{1}{2} \cdot \frac{\mathbb{E} ||w_{t+1} - w_t||^2}{\mathbb{E} ||w_t||^2}}
\]

In other words, \(\chi\) can be interpreted as a measure of the expected distance between two distinct draws \(w_t\) and \(w_{t+1}\) compared to their expected norm \(\mathbb{E} ||w_t||_2\). In adversarial environments, having a large value of \(\chi\) implies that the decision boundaries used for two different queries will be quite different from each other, which invariably makes the reverse engineering task more difficult to carry out.

It is straightforward to observe that maximizing \(\chi\) can be achieved by minimizing \(||\mu||_2^2/\sum_{j=1}^n \sigma_j^2\). However, an objective function of the latter form is not convex. One method to convexify it is to introduce new variables \(s_j = \sigma_j^2\), which would turn the non-convex objective function into a quadratic-over-linear term \(\mu^T \mu / s^T s\), which is jointly convex on \(\mu\) and \(s\) in the domain \(1^T s > 0\) \cite{123}. Next, the constraint in Eq. (6.1) is not convex. In order to retain convexity of the feasibility region, we linearize by omitting the square-root sign:

\[
\hat{a}_i^T \mu + \Phi^{-1}(\nu) \sum_{i=1}^n \hat{a}_i^2 s_i \leq b_i, \text{ for } i = 1, \ldots, m
\]

We are now left with one final ingredient, which is to handle the case when \(Aw \leq b\) is not feasible by incorporating slack variables \(\xi\) in a manner that is quite similar to soft-margin SVM. Intuitively speaking, this corresponds to the case where most of the constraints can be satisfied but not all. The final optimization problem, thus, becomes:
minimize $\mu, s, \xi$

\[
\frac{1}{2} \mu^T \mu + C \sum_{i=1}^{m} \xi_i
\]

subject to

\[
\hat{a}_i^T \mu + \Phi^{-1}(\nu) \sum_{j=1}^{n} \hat{a}_{ij}^2 s_j \leq b_i + \xi_i
\]

\[
s_j \geq 0, \quad \text{for} \ j = 1, 2, \ldots, n
\]

\[
\xi_i \geq 0, \quad \text{for} \ i = 1, 2, \ldots, m
\]  \(6.3\)

The above randomization method can be immediately applied to linear classification of the form $\hat{y}_i = \text{sign}(w^T x_i)$, where $x_i \in \mathbb{R}^d$ is a random instance. Because the original constraints in binary classification are of the form $y_i \cdot (w^T x_i) \geq 1$, we make the substitutions $\hat{a}_i = -y_i \cdot x_i$, $\hat{a}_{ij} = -y_i \cdot x_{ij}$ and $b_i = -1$ in (6.3). This yields the following optimization problem:

minimize $\mu, s, \xi$

\[
\frac{1}{2} \mu^T \mu + C \sum_{i=1}^{m} \xi_i
\]

subject to

\[
y_i \cdot (\mu^T x_i) \geq 1 + \Phi^{-1}(\nu) \sum_{j=1}^{n} x_{ij}^2 s_j - \xi_i
\]

\[
s_j \geq 0, \quad \text{for} \ j = 1, 2, \ldots, n
\]

\[
\xi_i \geq 0, \quad \text{for} \ i = 1, 2, \ldots, m
\]  \(6.4\)

The optimization problem in (6.4) is convex and can be used to learn an entire distribution of classifiers $\mathcal{N}(\mu, \Sigma)$. Its objective function ensures that the learned distribution of classifiers has a high variance, while the constraints ensure that the learned distribution has a high predictive accuracy.
6.3.2 Efficient Implementation

The optimization problem in (6.4) can be efficiently solved using SDP solvers such as MOSEK. Except for the quadratic-over-linear term $\mu^T \mu / 1^T s$, all other terms in (6.4) are linear in the optimization variables. The quadratic-over-linear term, however, can be minimized by minimizing a scalar $t \in \mathbb{R}$ subject to the convex constraint $\mu^T \mu / (1^T s) \leq t$. It can be shown using Schur’s complement that the latter constraint is equivalent to the two constraints [153]:

\[
\begin{align*}
(1) \quad & \begin{bmatrix} t & \mu^T \\ \mu & rI \end{bmatrix} \succeq 0, \\
(2) \quad & r = 1^T s,
\end{align*}
\]

Here, the notation $A \succeq 0$ implies that $A$ belongs to the positive semidefinite cone. Both constraints can be immediately handled by SDP solvers.

Experimentally, it often takes less time to solve the optimization problem in (6.4) than to solve SVM using a popular solver such as LIBSVM. For example, QSAR and THEOREM UCI datasets contain 1,000 and 3,000 training examples respectively. It takes MOSEK around 0.7s and 2s to solve (6.4) for the two datasets respectively, while LIBSVM takes 2.2s and 4s to find a solution for the same datasets. The MATLAB implementation code is provided in Appendix F.

6.3.3 Analysis

Next, we analyze the proposed algorithm. In this section, we will prove that the proposed randomization method does not increase the risk of over-fitting significantly compared to the use of fixed linear classifiers, and establish theoretically that there exists a tradeoff between accuracy, on one hand, and the variance of the learned distribution, on the other hand. We will describe the set of Pareto optimal points afterwards and demonstrate that having a fixed classifier in adversarial environments
is Pareto sub-optimal.

**Theorem 7.** Let $S = \{(x_i, y_i)\}_{i=1,...,m}$ be a sample that consists of $m$ instance-label pairs $(x_i, y_i)$ drawn i.i.d. from some unknown joint distribution $D$. Let $H = \mathbb{R}^d$ be the hypothesis space of homogenous linear classifiers in $\mathbb{R}^d$, and let $Q = \mathcal{N}(\mu, \Sigma)$ be a multivariate Gaussian density over $H$ that is learned based on the sample $S$. If for every instance $x$, a label $\hat{y} \in \{-1, +1\}$ is predicted by first drawing $w \sim Q$ and setting $\hat{y} = \text{sign}(w^T x)$, then the following bound on the true risk of the algorithm holds for all $M > 0$:

$$
\mathbb{P}_{(x,y) \sim D, w \sim Q}\{ y \neq \hat{y} \} \leq \mathbb{P}_{(x,y) \sim S}\{ y \cdot \mu^T x \leq M \} + \mathbb{P}_{(x,y) \sim D}\{ \Phi\left(-\frac{M}{\sqrt{\mu^T \Sigma \mu}}\right) \} + \tilde{O}\left(\sqrt{\frac{d}{m}}\right),
$$

where $\Phi(\cdot)$ is the CDF of the standard Gaussian density and the probability is taken over the random choice of $S$ and the internal randomness of the prediction rule.

**Proof.** First, let $w \in H$ and $\mu \in H$ be some arbitrary fixed hypotheses and write:

$$
\mathbb{P}_{(x,y) \sim D}\{ y \cdot w^T x \leq 0 \} \leq \mathbb{P}_{(x,y) \sim D}\{ y \cdot \mu^T x \leq M \}
$$

$$
+ \mathbb{P}_{(x,y) \sim D}\{ y \cdot w^T x \leq 0 \mid y \cdot \mu^T x \geq M \}
$$

Because this always holds for any $w$ and any $\mu$, it holds when $w$ is drawn from $H$ according to $Q$ and $\mu = \mathbb{E}_{w \sim Q}[w]$. So:

$$
\mathbb{P}_{(x,y) \sim D, w \sim Q}\{ y \cdot w^T x \leq 0 \} \leq \mathbb{P}_{(x,y) \sim D}\{ y \cdot \mu^T x \leq M \}
$$

$$
+ \mathbb{P}_{(x,y) \sim D, w \sim Q}\{ y \cdot w^T x \leq 0 \mid y \cdot \mu^T x \geq M \}
$$

Next, we bound each term on the Right-Hand Side (RHS) separately. First, we have
by the standard VC result:

\[ P_{(x,y) \sim D} \{ y \cdot \mu^T x \leq M \} \leq P_{(x,y) \sim S} \{ y \cdot \mu^T x \leq M \} + \tilde{O}(\sqrt{\frac{d}{m}}) \]

In addition, we can strengthen the above bound so that it holds uniformly for all values \( M > 0 \) with a high probability by re-writing the above inequality into:

\[ P_{(x,y) \sim D} \{ y \cdot \mu^T x - M \leq 0 \} \leq P_{(x,y) \sim S} \{ y \cdot \mu^T x - M \leq 0 \} + \tilde{O}(\sqrt{\frac{d+1}{m}}), \]

That is, we view \( M \) as if it were the bias of the class of non-homogenous linear classifiers. Because we introduce a biasing term by letting \( M \) be a free parameter determined by the sample \( S \), the VC dimension becomes \( d+1 \).

To bound the second term, we note that when \( w \sim Q \), we have:

\[ \mathbb{E}_{w \sim Q}[y \cdot w^T x] = y \cdot \mu^T x \]

However, \( w \) is a Gaussian random variable with mean \( \mu \) and covariance \( \Sigma \). Therefore:

\[ y \cdot w^T x - y \cdot \mu^T x \sim \mathcal{N}(0, x^T \Sigma x) \]

Hence, our goal reduces to bounding the tail of the CDF of the standard Gaussian density function. That is:

\[ P_{w \sim Q} \{ y \cdot w^T x \leq 0 \mid y \cdot \mu^T x \geq M \} = P_{Z \sim \mathcal{N}(0, x^T \Sigma x)}(Z \leq M) \]

\[ = \Phi\left(- \frac{M}{\sqrt{x^T \Sigma x}}\right) \]

Combining all bounds together yields the desired result. \( \square \)

Theorem 7 shows that the true risk of the proposed learning algorithm can be
decomposed into three components:

1. Accuracy: First, we have the margin-based empirical risk, which is the fraction of training examples whose prediction margin is less than $M$ with respect to (abbreviated w.r.t.) the mean $\mu$. This quantifies the accuracy of the mean of the distribution $\mu$ on the sample $S$. In the proposed algorithm, this risk is minimized because the margin w.r.t $\mu$ is maximized on all training examples by design.

2. Variance: Second, we have a variance-related component, which reveals two observations. If the variance of the distribution is small, i.e. $||\Sigma||_2 \approx 0$, then the contribution of the second term on the overall risk is small. This agrees with intuition since a smaller variance is expected to improve the prediction accuracy. However, Theorem 7 also reveals that one can tolerate a large variance in the distribution $\mathcal{N}(\mu, \Sigma)$ without severely impacting the accuracy of the classifier if the prediction margin $M$ w.r.t. the mean $\mu$ of the distribution is itself large. We will later show that this, in fact, is the key result that allows us to randomize without impacting accuracy for most classification problems of interest.

3. Generalization: The final term is the standard VC generalization bound. This generalization term holds for linear classifiers regardless of whether or not randomization is introduced.

An illustration to the tradeoff between accuracy and variance is provided in Fig. 6.3. In this figure, the two classification problems in $\mathbb{R}^2$ are depicted at the top along with the 80% and 90% confidence regions of the decision boundary. Recall that the decision boundary $\hat{y} = \text{sign}(w^T x)$ is randomized because $w$ is picked at random from $\mathcal{N}(\mu, \Sigma)$. Clearly, the confidence regions follow extremely well the actual shape of the two classes. Because the two classes on the left are well-separated from each other, the size of the 80% and 90% confidence regions is much bigger than in the
classification problem on the right. This is because the margin $M$ is large for the classification problem on the left.

Aside from the confidence regions, the plots at the bottom of Fig. 6.3 reveal the tradeoff curves between the predictive accuracy $\mathbb{E}_{(x,y) \sim D}(\hat{y} = y)$ of the distribution vs. its spread $\chi$ for each of the two classification problems. On the left, we note that $\chi$ can be increased without incurring any notable reduction in accuracy because the two classes themselves are well-separated. However, it is more difficult to incorporate variance in the classification problem on the right without suffering a notable loss in accuracy because the two classes are close to each other in the original space. This agrees with the statement of Theorem 7.

The accuracy-variance tradeoff curves convey a wealth of information. Before this is discussed, we recall the following definition:

**Definition 12** (Pareto Optimality). A Pareto optimal point is a pair $(\alpha, \chi)$, where $\alpha$ is predictive accuracy and $\chi$ is the spread of the distribution, given by Definition 11, such that no gain in $\alpha$ can be made without causing a loss in $\chi$ and vice versa. The set of Pareto optimal points form the tradeoff curve.

Any Pareto optimal point is a sound strategy. Keeping Pareto optimality in mind, the following remarks are useful when interpreting the tradeoff curves:

1. Every Pareto optimal point along the tradeoff curve is globally optimal with respect to some objective function of the form $\chi + \gamma \alpha$. The exact value of $\gamma$ that makes a Pareto point globally optimal is given by the slope of the line tangential to the curve at that particular point.

2. If we let $\alpha_{\text{max}}$ be the maximum attainable accuracy in the classification problem, then choosing $(\alpha, \chi) = (\alpha_{\text{max}}, 0)$ is not always Pareto optimal. This is because it is possible to introduce randomization in many classification problems without incurring a loss in accuracy as illustrated in Fig. 6.3(a).
3. The point of intersection between the tradeoff curve and $\chi = 1$ corresponds to blind prediction, which is a measure of class imbalance. The closer the intersection is to $(\frac{1}{2}, 1)$, the more balanced the two classes are. The impact of variance is more notable when the tradeoff curve intersects with $\chi = 0$ and $\chi = 1$ at two distant points.

The optimal tradeoff between accuracy and variance depends obviously on the application at hand, and every Pareto optimal point along the tradeoff curve is a sound strategy as discussed earlier. In our experiments, however, we will always choose the Pareto optimal point that maximizes the predictive accuracy of the distribution. To reiterate, such a choice of Pareto optimality is not equivalent to choosing $\chi = 0$ as highlighted in red in Fig. 6.3.

6.4 Experiments

To recall, I showed in Chapter 3 that fixed linear classifiers can be reverse-engineered quite efficiently. By contrast, learning the mean of the distribution $\mathcal{N}(\mu, \Sigma)$ is computationally hard because the amount of noise increases near the decision boundary. Hence, randomization is guaranteed to mitigate the adversarial risk of reverse engineering quite significantly. The main claim of this chapter is that randomization can be introduced with a large variance, without impacting the predictive accuracy of the classifier. In addition, I proposed a SDP method to implement this.

In this section, we will validate these claims experimentally. We will show that the proposed randomization method not only mitigates the risk of reverse engineering, but that it also performs significantly better than baseline randomization methods.

6.4.1 Evaluation Methodology

As illustrated earlier in Fig. 6.1, the only two strategies suggested in the literature for mitigating exploratory attacks, such as reverse engineering, are randomization
and disinformation. To reiterate, “disinformation” refers to the act of keeping the learning process, including datasets and training algorithms, as confidential as possible. Clearly, disinformation does not compete with randomization because both supplement each other.

For randomization, the only method suggested previously in the literature is to randomize predicted labels according to the confidence estimates associated with them \[1\]. That is, if the learning algorithm outputs confidence estimates, such as probabilities or functional margins, randomization may be introduced for uncertain
labels. In SVM, one method to implement this approach is to predict using:

\[
\hat{y}_i = \text{sign}(w^T x_i + \eta),
\]

(6.5)

where \(\eta\) is a number drawn uniformly at random from the set \([-1, +1]\). This approach is valid because SVM is constructed at the outset to separate the positive and negative classes by a hyperplane with a margin of 1. Obviously, the prediction rule in Eq. (6.5) does not impact those points whose functional margin exceeds 1.

Alternatively, one may compute probability estimates and introduce randomization according to \(P(y \mid x)\). However, because probability estimates are typically defined using some monotone function of margins, see for instance [154], probability-based randomization and margin-based randomization are equivalent. Therefore, we will restrict our experiments to confidence-based randomization of the form given in Eq. (6.5). Throughout the sequel, we will refer to this method as \text{CONF-RAND} whereas our proposed method of randomizing classifiers will be referred to as \text{CLASS-RAND}.

Throughout the experiments, we compare \text{CLASS-RAND} against the two baseline methods: \text{CONF-RAND} and using a fixed classifier. We will show that \text{CLASS-RAND} introduces more variance for the same level of accuracy, and that this advantage translates into having a greater robustness against the risk of reverse engineering. In order to validate our claims, we used 15 datasets in our experiment selected from the UCI machine learning repository [155]. The datasets are listed in Table 6.1. Linear SVM was implemented using the LIBSVM library [156], whereas \text{CLASS-RAND} was modeled using CVX and solved using MOSEK solver [120, 153]. We used a training-to-test split ratio of 9:1, and the same training-to-test split was used in all methods. The same experiment was repeated 10 times and averages are reported.
Figure 6.4: The tradeoff curves between spread $\chi$ and predictive accuracy for the 15 UCI datasets listed in Table 6.1, presented from left to right, top to bottom. The boundary of the blue-shaded region forms the set of Pareto optimal points.

### 6.4.2 Accuracy-Variance Tradeoff

In each dataset, the first exercise is to plot the tradeoff curve. This can be implemented by recording the predictive accuracy and the spread $\chi$ for various choices of $C$ and $\nu$. Once a model $(C, \nu)$ is selected and the optimization problem in (6.4) is solved, we obtain the desired distribution $\mathcal{N}(\mu, \Sigma)$. The tradeoff curves for the 15 datasets are shown in Fig. 6.4.

Having obtained a distribution $\mathcal{N}(\mu, \Sigma)$ for linear classifiers, prediction is performed using the Gibbs predictor [64, 131], which is summarized into three steps:

1. **Step 1**: Receive a new query $x_i$.

2. **Step 2**: Pick a new $w$ at random from $\mathcal{N}(\mu, \Sigma)$.

3. **Step 3**: Predict using $\hat{y}_i = \text{sign}(w^T x_i)$.

The same process is to be repeated for every new test example. For **conf-rand** and fixed classifier methods, on the other hand, the value of $C$ is also selected using 10-fold
cross-validation, which is implemented for each method separately. In **fixed svm**, prediction is performed using \( \hat{y}_i = \text{sign}(w^T x_i) \) whereas prediction in **conf-rand** is performed using Eq. (6.5).

Results of applying the above approach are provided in Table 6.1. Note that the classification accuracy is not impacted in all datasets when randomization is introduced using either **conf-rand** or **class-rand** methods. However, **class-rand** introduces significantly more variance for the same level of accuracy as will be demonstrated next. This is due to the fact that confidence-based randomization such, as by using Eq. (6.5), can be interpreted as a method of introducing randomization along one dimension only. On the other hand, our proposed method of randomizing classifiers learns an entire diagonal covariance matrix \( \Sigma \) and introduces randomization across all dimensions.

### 6.4.3 Reverse Engineering Attacks

Reverse engineering attacks are harder to carry out when the decision boundary is selected at random from a distribution with a large variance. As mentioned earlier, this is due to fundamental approximation hardness results, and it is demonstrated experimentally in Fig. 6.5. The figure plots the estimation error against the number of queries on a log-scale, when uncertainty sampling is employed by the adversary for randomized classifiers while the query synthesis algorithm proposed in Chapter 3 is used for fixed classifiers. To re-iterate, the query synthesis algorithm of Chapter 3 only operates in the noise-free setting, which is equivalent to the use of a fixed linear classifier in adversarial environments. The advantage of randomizing classifiers is quite significant across all figures. In addition, the proposed SDP algorithm is significantly more robust against the risk of reverse engineering than the baseline.

---

2 Throughout our discussion, we use homogenous halfspaces, where the prediction rule is \( \hat{y}_i = \text{sign}(w^T x_i) \). This is merely for notational convenience since non-homogenous halfspaces, whose prediction rule is \( \hat{y}_i = \text{sign}(w^T x_i + b) \) for some biasing term \( b \), can be equivalently thought of as homogenous halfspaces when a constant feature is added to the instances \( x \rightarrow [1, x] \in \mathbb{R}^{d+1} \).
Table 6.1: Results of applying linear SVM (fixed), CONF-RAND [1], and the proposed CLASS-RAND methods to 15 datasets. Here, \( n \) is the number of attributes while \( m \) is the number of examples in each dataset. Columns 2-4 list the results of the classification error rate of each method, while column 5 lists the spread of the distribution \( \chi \) that is used in the experiment.

<table>
<thead>
<tr>
<th>Dataset ((n, m))</th>
<th>Accuracy vs. Randomization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FIXED</td>
</tr>
<tr>
<td>1 - Iris ((4, 100))</td>
<td>(0.00 \pm 0.0%)</td>
</tr>
<tr>
<td>2 - Vertebral ((6, 310))</td>
<td>(13.9 \pm 5.1%)</td>
</tr>
<tr>
<td>3 - Seeds ((7, 140))</td>
<td>(2.14 \pm 4.8%)</td>
</tr>
<tr>
<td>4 - Diabetes ((8, 768))</td>
<td>(20.1 \pm 4.0%)</td>
</tr>
<tr>
<td>5 - Breast Cancer ((9, 699))</td>
<td>(3.91 \pm 2.9%)</td>
</tr>
<tr>
<td>6 - Digits- 0 vs. 8 ((17, 1499))</td>
<td>(2.28 \pm 0.8%)</td>
</tr>
<tr>
<td>7 - Digits- 1 vs. 7 ((17, 1557))</td>
<td>(0.32 \pm 0.5%)</td>
</tr>
<tr>
<td>8 - Climate ((18, 540))</td>
<td>(5.56 \pm 3.6%)</td>
</tr>
<tr>
<td>9 - Parkinisons ((22, 195))</td>
<td>(13.2 \pm 7.6%)</td>
</tr>
<tr>
<td>10 - Ionosphere ((33, 351))</td>
<td>(11.4 \pm 3.8%)</td>
</tr>
<tr>
<td>11 - QSAR ((41, 1055))</td>
<td>(11.7 \pm 3.3%)</td>
</tr>
<tr>
<td>12 - Theorem ((52, 3059))</td>
<td>(27.0 \pm 2.5%)</td>
</tr>
<tr>
<td>13 - Spambase ((57, 4601))</td>
<td>(6.76 \pm 1.1%)</td>
</tr>
<tr>
<td>14 - Sonar ((60, 208))</td>
<td>(31.0 \pm 14.0%)</td>
</tr>
<tr>
<td>15 - MUSK v.1 ((166, 476))</td>
<td>(14.3 \pm 3.6%)</td>
</tr>
</tbody>
</table>
randomization method. In those figures, the estimation error is defined by:

\[
\text{estimation error} = \left\| \frac{w}{\|w\|_2} - \frac{\hat{w}}{\|\hat{w}\|_2} \right\|_2^2,
\]

where \( w \) is the true classifier used by the defender while \( \hat{w} \) is the adversary’s estimate of \( w \). In order to appropriately compare hyperplanes, both \( w \) and \( \hat{w} \) are normalized to unit norm.

### 6.5 Conclusions

Machine learning algorithms have been immensely successful in numerous applications including environmental sciences, bioinformatics, recommendations systems, and text classification, to name just a few. They are also attractive solutions for security-sensitive applications such as spam filtering, intrusion detection, and fraud detection. Nevertheless, such adversarial environments pose new challenges to the learning systems, and part of the challenge is to be able to make the learning system robust against exploratory attacks, such as evasion and reverse engineering.

In this chapter, we show that the risk of reverse engineering can be mitigated by learning a distribution of classifiers, as opposed to the conventional practice of learning a single fixed classifier. Such a distribution of classifiers can, in fact, be learned quite efficiently using Semidefinite Programming solvers such as MOSEK and SDPT3. We provide a convex formulation for the learning task and apply it to real-world classification problems. Using the accuracy-variance tradeoff curves, we show that having a fixed linear classifier is, generally, a Pareto-suboptimal strategy in adversarial environments. By drawing classifiers at random from a distribution with a large variance, the adversary’s effort to carry out a reverse engineering attack will increase significantly at a little cost to the learning system.
Figure 6.5: Results of conducting the reverse engineering experiment in Section 6.4.3. In this figure, the red curve is estimation error when using a fixed classifier, the black curve is estimation error when using confidence-based randomization CONF-rand, while the blue curve is estimation error when using a distribution of classifiers class-rand. The x-axis is the number of probes, while the y-axis is estimation error in a log-scale. The figures correspond to the datasets in Table 6.1 ordered left-to-right and top-to-bottom.
Chapter 7

Learning via the Empirical Kernel Maps

7.1 Introduction

Linear classification methods are advantageous because they balance between the risks of under-fitting and over-fitting. However, as shown in Chapter 3, fixed linear classifiers in adversarial environments are quite vulnerable to the risk of reverse engineering. In Chapter 6, one method of mitigating the reverse engineering risk is introduced, whereby a distribution of classifiers is learned by the defender. I proved theoretically that such a randomization algorithm also balanced between the risks of under-fitting and over-fitting, while also being robust against the risk of reverse engineering.

Nevertheless, a different approach for mitigating the risk of reverse engineering exists, which is to use a non-linear method of classification. This is briefly alluded to in previous chapters. The most prototypical approach for non-linear classification is to use a non-parametric similarity-based classifier, such as the kernelized Support Vector Machine (SVM). However, similarity-based algorithms often require that the similarity function be Positive Semidefinite (PSD) or they handle indefinite similarities unsatisfactorily. In other words, popular off-the-shelf non-linear classifiers are often effective only if the similarity function is positive semidefinite.

Definition 13 (Positive Semidefinite Function). A function $S : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ is called positive semidefinite if and only if for any collection of instances $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}^m$, the corresponding similarity (a.k.a. kernel) matrix $K \in \mathbb{R}^{m \times m}$ defined by $K_{ij} = S(x_i, x_j)$ is positive semidefinite.
Example 4. The inner product $S(x_i, x_j) = x_i^T x_j$ is a positive semidefinite function. Its corresponding similarity matrix is often referred to as the Gram matrix.

Too see why the condition of positive semi-definiteness is often required, we can look into the kernelized SVM, which is considered as one of the top ten machine learning algorithms in practice today [157]. The SVM algorithm was developed in the 1990s by V Vapnik and colleagues [158, 56, 11]. It works by constructing a separating hyperplane in a high- (possibly infinite) dimensional feature space that separates the positive from the negative instances. Most crucially, it seeks a separating hyperplane, which ensures that most training instances are correctly classified with a large margin. The SVM algorithm was inspired by deep theoretical foundations that make use of the Vapnik-Chervonenkis (VC) dimension to establish the generalization ability of this family of classifiers [57, 11]. In informal terms, by seeking a large margin classifier, SVM tends to reduce its own risk of over-fitting. By employing the kernel trick, it reduces its own risk of under-fitting as well. Hence, both risks are minimized.

The need for a positive semidefinite similarity function in SVM can be seen if we look into the dual-form representation. Specifically, the classifier in SVM is often computed by solving the following dual optimization problem:

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \alpha^T Q \alpha - 1^T \alpha \\
\text{subject to} & \quad 0 \leq \alpha \leq C 1, \quad \sum_{i=1}^{m} \alpha_i y_i = 0
\end{align*}$$

Here, $Q_{ij} = y_i y_j K_{ij}$, $C$ is a fixed tradeoff constant, and $1 = (1, 1, \ldots, 1)^T \in \mathbb{R}^m$. In the dual-form in (7.1), the similarity function has to be positive semidefinite in order to guarantee the convexity of the optimization problem and the existence of a Reproducing Hilbert Kernel Space (RHKS). When it is positive semidefinite, the optimization problem in (7.1) can be solved quite efficiently and its optimal solution
can be used to construct a large margin separating hyperplane in some implicit feature space. Such advantages are no longer guaranteed when the similarity function is indefinite.

In real-life applications, however, many indefinite similarity functions are commonly used. In bioinformatics, for example, the Longest Common Subsequence (LCS) is often used to define the similarity between genetic sequences. In association rule mining, set operations such as the union and/or intersection can be used to define the similarity between transactions. In text mining, furthermore, one can use human-judged similarities between concepts and words or use the symmetrized Kullback-Leibler divergence when treating documents as probability distributions of individual words. In the study of trajectories or time series in general, one can use the Dynamic Time Warping (DTW) and its many variants. In computer vision, the tangent distance or the shape matching distance are quite popular \[159, 160, 161, 162\]. Other indefinite similarities are also frequently encountered in psychology, neuroscience, and economics \[163\]. Extending large-margin classification to indefinite similarities will have many important applications.

In this chapter, we propose a method for classification with arbitrary similarity functions. The performance of the algorithm does not depend on whether or not the similarity function is positive semidefinite. Specifically, we prove that implementing a linear classifier over the empirical kernel maps, also known as the 1-norm SVM method, is the best method to-date for classification with indefinite similarities\[1\]. This conclusion is supported by both theoretical and experimental arguments. In particular, I will prove that the 1-norm SVM algorithm also balances between the risks for under-fitting and over-fitting for both definite and indefinite similarities, and that its advantage over other popular methods is statistically significant.

There are several reasons why the 1-norm SVM method is competitive with, if

---

1The empirical kernel maps were briefly mentioned in Chapter 1. They will be described in more details later in Section 7.4.2.
not superior to, more recent approaches in its predictive accuracy. Unlike many alternative methods that have been proposed in the literature, 1-norm SVM retains the convexity of its optimization problem and treats both training and test examples consistently. It is closely connected to many well-established learning algorithms, such as artificial neural networks, nearest neighbor classifiers, and SVM. As will be discussed in more details in the sequel, these connections between 1-norm SVM and those learning algorithms provide a formal justification to the use of 1-norm SVM when learning with indefinite similarities.

In the literature, 1-norm SVM is often used as an embedded feature selection method, where learning and feature selection are performed simultaneously [88, 89, 90, 91, 92, 93]. It was studied in [89], where it was argued that 1-norm SVM has an advantage over the standard form of SVM in (7.1) when there are noisy redundant features. Despite the fact that it was suggested to be a viable method for classification with indefinite similarities more than 15 years ago [163], it remained a relatively less known method than more-involved less-accurate approaches such as kernel approximation and non-convex optimization. In particular, 1-norm SVM is rarely used as a standard benchmark for the task (see for instance [164, 165, 166, 167]).

The rest of the chapter is outlined as follows. First, we review the existing literature on learning with indefinite similarities. Second, we describe the 1-norm SVM method and show how it can be adapted to handle binary classification with indefinite similarities. We provide various motivations behind its formulation by relating 1-norm SVM to artificial neural networks, nearest neighbor classifiers, and SVM. We also show that 1-norm SVM can be interpreted as a method of minimizing an upper bound on the expected true risk (prediction error rate). After that, we present experimental results using synthetic and real datasets, which validate the advantage of 1-norm SVM in handling indefinite similarities over all other methods.
7.2 Previous Work

Several methods have been proposed in the literature for learning with indefinite similarities. Some of these methods are old, such as non-convex optimization \[167\], while others are more recent such as Eigen-decomposition Support Vector Machine (ESVM) that was proposed recently in 2013 \[168\]. Generally speaking, however, the most dominant methods can be grouped into four broad approaches: (1) kernel approximation, (2) non-convex optimization, (3) nearest neighbor classification, and (4) learning in Krein spaces. We review each approach next.

7.2.1 Kernel Approximation

The first approach for learning with indefinite similarities is the kernel approximation method. In this approach, not only does the learning algorithm look for a hypothesis that can correctly classify training instances with a large margin, but it also approximates the indefinite similarity matrix with a PSD matrix so that the resulting optimization problem can be solved quite efficiently. That is, it is implicitly assumed that the advantage that would be reaped from convexifying a non-convex optimization problem outweigh the information loss we incur by artificially altering (distorting) the similarity matrix.

Two of the earliest kernel approximation methods are the denoise and the flip methods. Both methods adjust the eigenvalues of the similarity matrix of training examples so that it becomes positive semidefinite. The two methods differ, however, in how they alter those eigenvalues. On one hand, the denoise method sets all negative eigenvalues to zero. The motivation behind this approach is to assume that negative eigenvalues are caused by noise \[169\]. On the other hand, the flip method flips the sign of the negative eigenvalues, hence the name. This method aims at retaining some of the information coded in those negative eigenvalues \[169, 163\]. A third more involved kernel approximation method is to formulate a max-min optimization
problem that both seeks support vectors as well as a PSD kernel that approximates
the indefinite similarity matrix. The latter approach was introduced by Luss and
d’Aspremont in 2007 with improvements in training time reported in the following
years \cite{166, 165, 164}.

All of the kernel approximation methods above guarantee that the optimization
problem remains convex during training. During prediction, however, the original
indefinite similarity function is used. Hence, past and future examples are treated
inconsistently. In addition, such methods are only useful when the similarity matrix
is approximable by a PSD matrix. For other similarity functions, such as the sigmoid
kernel that can occasionally yield a negative semidefinite matrix for certain values of
its hyperparameters, the kernel approximation approach cannot be utilized. In fact,
and as will be shown later in the evaluations in Section \ref{evaluations} the accuracy of some
of these methods can even become worse than random guessing, especially when the
similarity matrix is close to being negative semidefinite.

\subsection{Non-Convex Optimization}

The second approach for learning with indefinite similarities is non-convex optimiza-
tion. In contrast to the previous kernel approximation approach, non-convex op-
timization implicitly assumes that treating training and test examples consistently
by keeping the similarity function intact is more important than convexifying the
problem. Of course, because the optimization problem is non-convex, however, this
approach can terminate at a local minimum.

In the literature, non-convex optimization with indefinite similarities for SVMs has
received a fair attention. In the theoretical side, Haasdonk interprets this approach
as a method of minimizing the distance between reduced convex hulls in a pseudo-
Euclidean space \cite{162}. In the practical side, Sequential Minimal Optimization (SMO)
type decomposition methods, which seek a stationary point, have been proposed
for indefinite similarity functions such as the sigmoid kernel \([167]\). Nevertheless, because non-convex optimization can terminate at a stationary point that can be quite distant from the globally optimal solution, non-convex optimization does not guarantee learning \([159]\). Besides, this approach only works reasonably well if the similarity matrix is approximately PSD.

### 7.2.3 Nearest Neighbor Classification

The third approach for handling indefinite similarities is nearest neighbor classification. In the \(k\) Nearest Neighbor (k-NN) rule, the \(k\) most similar instances are identified and the most frequent label in those \(k\) instances is predicted. This can be applied regardless of whether the similarity function is positive semidefinite or not. Nearest neighbor classification is indeed a very popular algorithm \([157]\). However, it suffers from two main drawbacks. First, it is not as accurate as other approaches such as learning in Krein spaces and the 1-norm SVM method. Second, the entire training set is used for prediction, which makes the task of predicting a label computationally demanding.

### 7.2.4 Learning in Krein Spaces

The last major approach that has been proposed in the literature resolves many of the issues that are inherent in the kernel approximation and non-convex optimization methods. It proposes efficient learning algorithms that treat training and test examples consistently, and achieves a very high accuracy in practice. This relatively recent approach is based on learning in Krein spaces, in which the similarity function is decomposed into the sum of one positive semidefinite kernel and one negative semidefinite kernel \([170][168]\). By taking such a decomposition of similarity functions, learning in Krein spaces embraces the idea that the negative part of a similarity function contains viable information \([168]\).
Whereas learning in a Hilbert space can be formulated as a minimization problem, learning in a Krein space is formulated as a stabilization problem, where a saddle point to the objective function is found. One fairly recent algorithm that has been proposed to solve the stabilization problem is called ESVM [168]. While this algorithm has been shown to outperform all previous methods, its primary drawback is that it does not produce sparse solutions, hence the entire list of training examples are often needed during prediction.

As will be shown later in the evaluations in Section 7.5, 1-norm SVM and ESVM both outperform all other methods significantly, and their performance is strikingly similar despite the different approaches employed by the two algorithms. Nevertheless, 1-norm SVM has the added advantage of producing a solution that is often sparser than ESVM by many orders of magnitude. Hence, 1-norm SVM will prove to be the most effective method for classification with indefinite similarities.

7.3 The 1-Norm Support Vector Machine

In this section, the 1-norm SVM method is described in details. Theoretical arguments as to why it is a viable method for classification with indefinite similarities will be presented in the next section.

The 1-norm SVM method was proposed more than 15 years ago [163]. It was rediscovered under many guises later, such as for being a special case of the generalized SVM [171] and for being a method of embedding similarities into features [159, 169]. However, it is widely used in the literature for embedded feature selection [88, 89, 90, 91, 92, 93].

The form of the 1-norm SVM method used here has been proposed as a method for embedded feature selection in previous works, such as [89, 92]. As shown later in (7.3), this form uses a dictionary of basis functions. In general, however, the term “1-norm SVM” is often used to mean the traditional SVM, but with an \( \ell_1 \) regularization term. This latter formulation can be shown to be a special case of the general form that is used in this paper. In particular, \( \lambda_i \) is the \( i \)-th entry of the coefficient vector \( w \), while the basis function \( h_i(x) \) is the projection of \( x \) on the \( i \)-th dimension. This restricted form is the one often used for feature selection.
7.3.1 Description

Before we describe the 1-norm SVM method, we recall the binary classification setting. In this setting, we have an instance space $\mathcal{X}$ and a target set $\mathcal{Y} = \{+1, -1\}$. Every observation is a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ drawn from some fixed unknown distribution $D$. A classifier $h : \mathcal{X} \to \mathcal{Y}$ is a rule that maps each instance $x \in \mathcal{X}$ to either the positive class or the negative class. Throughout this paper, such a classifier is assumed to be of the form $h(x) = \text{sign}(f(x))$ for some function $f : \mathcal{X} \to \mathbb{R}$, where $f$ is learned on the basis of a set of $m$ training examples $\{(x_i, y_i)\}_{i=1}^{m}$ drawn i.i.d. from $D$.

One natural method of predicting labels is similarity-based classification, which is a generalization to nearest neighbor classifiers. Given a similarity function $S : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, we can predict whether $y = +1$ or $y = -1$ for an instance $x$ based on how similar $x$ is to the fixed set of $m$ training examples. A general approach of similarity-based classification is to use a decision rule of the form:

$$f(x) = \lambda_0 + \sum_{i=1}^{m} \lambda_i y_i S(x, x_i) \quad (7.2)$$

To reiterate, $x$ here is the instance whose label we would like to predict, whereas $\{(x_i, y_i)\}_{i=1}^{m}$ is a training set of $m$ observations drawn i.i.d. from some fixed unknown distribution $D$.

To interpret the decision rule in $(7.2)$, we note that $\lambda_0$ is a biasing term that is similar to the activation threshold in neural networks or the prior in Bayesian methods, while $\lambda_i$ for $i \geq 1$ quantifies how important the training example $(x_i, y_i)$ is to the classification rule. According to $(7.2)$, if an instance $x$ is “more” similar to the weighted set of negative training examples, then $f(x)$ would be negative; otherwise, $f(x)$ is positive. Different methods of learning the weights $\lambda$ yield different learning algorithms.

One of the most successful algorithms for similarity-based classification is SVM
itself. In SVM, the similarity function $S$ is always chosen to be positive semidefinite, which implies that there exists a feature mapping $\phi : \mathcal{X} \to \mathbb{F}$ for some Hilbert space $\mathbb{F}$ endowed with an inner product $\langle \cdot, \cdot \rangle$ such that:

$$\forall x_i, x_j \in \mathcal{X} : S(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

In the literature, $\mathbb{F}$ is often referred to as the feature space \[172\]. The existence of a feature space for a similarity function $S$ is equivalent to the statement that $S$ is positive semidefinite \[172\]. In this case, $S$ is often referred to as a kernel.

If the similarity function $S$ is PSD, then similarity in the instance space $\mathcal{X}$ can be interpreted differently in the feature space $\mathbb{F}$. Specifically, instead of performing a similarity-based classification in the instance space $\mathcal{X}$, one can seek a separating hyperplane with a large functional margin in the feature space $\mathbb{F}$. This is precisely the approach employed by SVM. The Representer Theorem states that such an approach yields a decision rule that is identical to the similarity-based classification rule in \[7.2\] \[74, 172\]. Hence, SVM is, indeed, a similarity-based classification algorithm.

Support Vector Machine has been quite successful in practice. However, SVM can only be utilized if the similarity function is PSD as mentioned earlier. When the similarity function is indefinite, a different learning algorithm is required such as the 1-norm SVM. To see how 1-norm SVM can be adapted to handle indefinite similarities, we begin with the formulation proposed in \[89\]. In this formulation, we have a dictionary of basis functions $\mathbb{D} = \{h_1(\cdot), h_2(\cdot), \ldots\}$, where $h_j : \mathcal{X} \to \mathbb{R}$, and consider classification using:

$$f(x) = \lambda_0 + \sum_{j \geq 1} \lambda_j \cdot h_j(x) \quad (7.3)$$

In the above expression for $f$, the basis functions $h_j : \mathcal{X} \to \mathbb{R}$ are fixed and the only variables to be optimized are $\lambda_j$ for $j \geq 0$. 
In [89], the following optimization problem is proposed for finding the weights $\lambda$ (Eq. (5) in [89]):

\[
\begin{align*}
\text{minimize} & \quad ||\lambda||_1 + C ||\xi||_1 \\
\text{subject to} & \quad y_i \cdot \left( \lambda_0 + \sum_j \lambda_j \cdot h_j(x_i) \right) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0, \quad \text{for all } i = 1, 2, \ldots, m
\end{align*}
\]

Here, $C$ is a tradeoff parameter between regularization and fitting. We will provide several motivations behind this formulation shortly.

To utilize the above method in handling indefinite similarities, we set $h_j(\cdot) = y_j S(x_j, \cdot)$, where $S(x_j, \cdot) : \mathcal{X} \to \mathbb{R}$ is a function that measures the similarity to the instance $x_j$. In addition, we impose the non-negativity constraint $\lambda_j \geq 0$ for all $j \geq 1$ to ensure that any instance $x_j$ can be representative to its own class $y_j$ only. This gives us the following Linear Programming (LP):

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=0}^m \lambda_i + C \sum_{i=1}^m \xi_i \\
\text{subject to} & \quad \begin{bmatrix} y_i & Q \end{bmatrix} \lambda \geq 1 - \xi \\
& \quad \lambda_i, \xi_i \geq 0, \quad \text{for all } i = 1, 2, \ldots, m
\end{align*}
\]

Here, $y \in \{-1, +1\}^m$ is a vector of class labels for all $m$ training examples and $Q \in \mathbb{R}^{m \times m}$ is, again, given by:

\[
Q_{i,j} = y_i y_j S(x_i, x_j)
\]

The above formulation is a simple LP that can be solved quite efficiently using, for example, the Gurobi solver [173]. Note that unlike the standard formulation of

---

3In [89] Eq. (5), the hinge loss is used explicitly in the objective function, which is equivalent to the use of slack variables in our formulation.
Figure 7.1: In this figure, two classes are shown in red and blue. A solid marker is a support vector, whose size is proportional to its weight $\lambda_i$. Classification regions for the red and blue classes are shown in yellow and green, respectively.

SVM in (7.1), the LP formulation in (7.4) remains convex even when the matrix $Q$ is not PSD because both the objective function and inequality constraints are linear in the optimization variables ($\lambda, \xi$). Once the LP is solved, we predict the label of a new instance $x$ using the earlier classification rule in (7.2).

Training examples $x_i$ with $\lambda_i > 0$ are analogous to the support vectors in SVM, and we will refer to them as support vectors here as well. As depicted in Fig. 7.1, each support vector is ‘carefully’ placed in the plane to guard a region dominated by its respective class. In practice, because the regularization term in the objective function minimizes the $\ell_1$ norm of $\lambda$, the vector $\lambda$ tends to be sparse and the number of support vectors tends to be small.

### 7.3.2 Computational Complexity

Before we analyze the 1-norm SVM method, we make a final remark on its computational complexity. The 1-norm SVM method is a LP, for which many efficient solvers currently exist. These include, for instance, the Gurobi solver [173], the CPLEX solver [174], and MATLAB’s built-in `linprog` command. When using interior-point methods, rigorous bounds have been established for the number of computations required to solve a linear program [123]. In general, it can be shown that the 1-norm SVM method requires, at most, $O(m^3)$ computations in the worst case.
7.4 Analysis

In this section, we provide several motivations for using the LP in (7.4) to classify with indefinite similarities. First, we show that 1-norm SVM can be interpreted as a method of producing a decision boundary with a large similarity margin in the instance space $\mathcal{X}$. Using theoretical bounds expressed in terms of the margin, large-margin classification is proved to be less susceptible to over-fitting. On a related note, we show that 1-norm SVM is a large-margin classifier because it can also be interpreted as an $\ell_1$-regularized linear SVM applied to the empirical kernel maps.

Next, we relate 1-norm SVM to nearest neighbor classifiers, which reveals that 1-NN classification is a special case of the 1-norm SVM. Nearest neighbor classifiers are some of the most important classification algorithms in practice today with provable performance bounds. For example, it has long been established that the true risk (test error rate) of 1-NN is asymptotically bounded from above by twice the Bayes risk [175].

After that, we show how the 1-norm SVM can be interpreted as an approximate implementation of the Structural Risk Minimization (SRM) induction principle [11]. This can be deduced by establishing the connection between 1-norm SVM and two-layer neural networks, which, in turn, justifies the use of $\ell_1$ regularization. Similarly, we show that the objective function in (7.4) can be interpreted as a method of minimizing an upper bound on expected prediction error rate using the Leave-One-Out (LOO) error estimation method.

7.4.1 Large-Margin Classification

We begin by interpreting 1-norm SVM as a large-margin classifier. Given an instance space $\mathcal{X}$, a target set $\mathcal{Y} = \{+1, -1\}$, and a suitable measure of similarity $S : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, we can define the similarity between an instance $x \in \mathcal{X}$ and a label $y \in \mathcal{Y}$ using a weighted sum of similarities with all of the training instances that belong to the label
y. In other words, we may write:

\[
S(x, Y; \lambda) = \sum_{i=1}^{m} \lambda_i S(x, x_i) \cdot I\{y_i = y\}
\]

to denote the class similarity between \(x\) and a class \(y \in \{+1, -1\}\). Here, the weight \(\lambda_i \geq 0\) represents the importance of the training instance \(x_i\) to its own class \(y_i\). In addition, we can introduce an offset \(\lambda_0\) that quantifies prior preference. This offset plays a role that is similar to the prior in Bayesian methods, the activation threshold in neural networks, and the offset in SVM. Thus, we consider classification using the rule:

\[
\hat{y} = \text{sign}\{S(x, +1; \lambda) - S(x, -1; \lambda) + \lambda_0\},
\]

which is identical to the classification rule of 1-norm SVM given in (7.2). Moreover, we define the similarity margin \(M_i\) for the training example \((x_i, y_i)\) in the usual sense:

\[
M_i = S(x_i, y_i; \lambda) - S(x_i, -y_i; \lambda) + y_i \lambda_0
\]

This notion of similarity margin reduces to the notion of functional margin when the similarity function \(S\) is PSD. In general, the \(i\)-th training example \((x_i, y_i)\) is classified correctly if and only if its margin is positive.

Maximizing the minimum similarity margin can be formulated as a LP. First, we write:

\[
\begin{align*}
\text{maximize} & \quad M \\
\text{subject to} & \quad S(x_i, y_i; \lambda) - S(x_i, -y_i; \lambda) + y_i \lambda_0 \geq M, \quad (\text{for all } 1 \leq i \leq m) \\
& \quad \lambda \geq 0
\end{align*}
\]

However, the decision rule given by (7.5) does not change when we multiply the
weights $\lambda$ by any fixed positive constant including constants that are arbitrarily large. This is because the decision rule only looks into the sign of its argument. In particular, we can always rescale the weights $\lambda$ to be arbitrarily large, for which $M \to \infty$. This degree of freedom implies that we need to maximize the ratio $M/||\lambda||$ instead of maximizing $M$ in absolute terms. Here, any norm $|| \cdot ||$ suffices but the 1-norm is preferred because it produces sparse solutions and because it gives a better accuracy in practice.

Since our objective is to maximize the ratio $M/||\lambda||_1$, we can fix $M = 1$ and minimize $||\lambda||_1$. This results in the following optimization problem:

$$\min_{\lambda} ||\lambda||_1$$
$$\text{subject to } S(x_i, y_i; \lambda) - S(x_i, -y_i; \lambda) + y_i \lambda_0 \geq 1, \text{ for all } i = 1, 2, \ldots, m$$
$$\lambda_i \geq 0, \text{ for all } i = 1, 2, \ldots, m$$

In addition, to avoid over-fitting outliers or noisy observations and to be able to handle the case of non-separable classes, slack variables $\xi$ can be introduced as well. This results in the LP formulation of the 1-norm SVM given earlier in (7.4). Hence, 1-norm SVM can be interpreted as a method of finding a decision boundary with a large similarity margin in the instance space $\mathcal{X}$. Such an interpretation holds regardless of whether or not the similarity function is PSD. Thus, we expect 1-norm SVM to perform well even for indefinite similarity functions.

There are various results established in the literature for large-margin classification, which reveal that maximizing the margin can help mitigate the risk of

---

4Sparse solutions are important for at least two reasons. First, only a small subset of the training set is needed during prediction, and hence prediction can be carried out quite efficiently. Second, minimizing the number of support vectors can be interpreted as a method of minimizing an upper bound on the expected test error rate (see for example Eq. (93) in [57] in the case of SVM and the discussion in Section 7.4.5 in the case of 1-norm SVM).

5An alternative derivation is as follows. We can introduce the coefficient vector $w = \lambda/M$ and fix $||\lambda|| = 1$ to avoid the issue of rescaling. Then, maximizing the margin $M$ becomes equivalent to minimizing an appropriate norm of $w$ such as the $\ell_1$ norm. This leads to the same linear program that is used in 1-norm SVM.
over-fitting. If we define $R_D(h)$ to be the true risk of an inferred classifier $f$, i.e. $R_D(f) = \mathbb{P}_{(x,y) \sim \mathcal{D}} [y f(x) < 0]$, then several bounds of the following form can be established [176, 172]:

$$R_D(f) \leq \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}\{y_i f(x_i) < M\} + O\left(\frac{C(\mathcal{H})}{M \sqrt{m}}\right)$$  \hspace{1cm} (7.6)

Here, the first term measures the margin-based error, which is the fraction of training examples whose margin is below a fixed number $M$, while the second term is a generalization risk that depends on the margin $M$, the number of training examples $m$, and on some appropriate measure of complexity of the hypothesis class $C(\mathcal{H})$. Most importantly, such results hold uniformly for all $M \in (0,1)$. Because maximizing the margin on the training set for a fixed $m$ and $\mathcal{H}$ reduces both terms in the right-hand side simultaneously, large-margin classification, such as by using the 1-norm SVM, tend to perform well in practice.

### 7.4.2 Empirical Kernel Maps

In addition to being interpreted as a method of producing a decision boundary with a large similarity margin, the 1-norm SVM can also be interpreted as an $\ell_1$-regularized SVM applied to the empirical kernel map. As mentioned earlier in Section 1.4.3, learning on the empirical kernel map can be interpreted as a middle approach between linear and non-linear classification methods.

The empirical kernel map is defined as follows. Given a training set $\{(x_i, y_i)\}_{i=1,\ldots,m}$, one can introduce the new mapping:

$$\phi(\cdot) = (y_1 S(x_1, \cdot), \ldots, y_m S(x_m, \cdot)) : \mathcal{X} \to \mathbb{R}^m,$$

Essentially, this mapping turns similarities into features [74, 159, 169].

Given the mapping $\phi : \mathcal{X} \to \mathbb{R}^m$, we can apply any classification algorithm
on the new features $\phi(x)$. In particular, we can use linear SVM, which yields the following optimization problem:

$$
\begin{align*}
\text{minimize} & \quad \|w\| + C \sum_{i=1}^{m} \xi_i \\
\text{subject to} & \quad y_i \left( \phi(x_i)^T w + b \right) \geq 1 - \xi_i, \quad \text{for all } i = 1, 2, \ldots, m \\
& \quad \xi_i \geq 0, \quad \text{for all } i = 1, 2, \ldots, m
\end{align*}
$$

(7.7)

Knowing that the decision boundary is given by:

$$
y = \text{sign}(\phi(x)^T w + b) = \text{sign}(b + \sum_{i=1}^{m} w_i y_i S(x, x_i)).
$$

we note that $b$ corresponds in our earlier notation to the bias term $\lambda_0$ while $w_i = \lambda_i$ for all $i \geq 1$.

With $\ell_1$ regularization, the optimization problem in (7.7) becomes nearly identical to that of 1-norm SVM in (7.4). The only (minor) difference is the non-negativity constraint $\lambda_i \geq 0$ for $i \geq 1$, which we imposed in 1-norm SVM so that it would also behave like a nearest neighbor classification algorithm as will be discussed later. Consequently, 1-norm SVM can be interpreted as a method of producing a separating hyperplane with a large functional margin applied to the empirical kernel map $\phi$.

Again, although it is possible to use any norm in the regularization term of the objective function in (7.7), such as $\ell_2$ regularization, $\ell_1$ regularization is preferred because it produces a sparse solution so that only a small subset of the training set is needed during prediction.

Applying linear classification to the empirical kernel map can be justified rigorously. In [73], it is shown that linear classification via the empirical kernel map, which is identical to our interpretation of the 1-norm SVM method discussed above, is guaranteed to have a small true risk (prediction error rate) as long as the similarity function is reasonable. Here, “reasonable” means that a high similarity exists
between objects of the same class and a low similarity exists between objects of different classes. This guarantee on performance holds regardless of whether or not the similarity function is PSD \cite{73}. Therefore, we expect 1-norm SVM to perform quite well for indefinite similarities.

7.4.3 Nearest Neighbor Classification

Nearest neighbor classifiers are some of the most important data mining algorithms in practice today \cite{157}, and theoretical results for such algorithms have long been established. For instance, one well-known result shows that the true risk of 1-NN is asymptotically bounded from above by twice the Bayes risk \cite{175}. Here, we show that 1-NN classification is a special case of the 1-norm SVM method.

**Lemma 2.** Given a semi-metric \( d : \mathcal{X} \times \mathcal{X} \to [0, \infty) \), let \( S(x_i, x_j) = \psi(\gamma \cdot d(x_i, x_j)) \) : \( \mathcal{X} \times \mathcal{X} \to (0, 1] \) for some bandwidth \( \gamma > 0 \) be a radial monotone decreasing function of distance that satisfies \( \psi(0) = 1 \) and the property: \( z_1 > z_2 \Rightarrow \lim_{\gamma \to \infty} \frac{\psi(\gamma z_1)}{\psi(\gamma z_2)} = 0 \). Also, let \( C = 1 + \epsilon \) for some \( \epsilon > 0 \) and fix \( \lambda_0 = 0 \) (i.e. with no bias term). Then, the behavior of 1-norm SVM can be made arbitrarily close to 1-NN using a sufficiently large bandwidth \( \gamma \to \infty \).

**Proof.** First, because \( d \) is a semi-metric on the instance space \( \mathcal{X} \), we have \( d(x_i, x_i) = 0 \) and \( S(x_i, x_i) = 1 \). By setting \( \lambda_0 = 0 \), the margin \( M_i \) for the training example \((x_i, y_i)\) reduces to:

\[
M_i = y_i \sum_{j} \lambda_j y_j S(x_i, x_j) = \lambda_i + y_i \sum_{j \neq i} \lambda_j y_j S(x_i, x_j)
\]

Since \( S(x_i, x_j) > 0 \) and \( y_j \in \{+1, -1\} \), we deduce the sandwich inequality:

\[
- \sum_{j \neq i} \lambda_j S(x_i, x_j) \leq M_i - \lambda_i \leq \sum_{j \neq i} \lambda_j S(x_i, x_j) \quad (7.8)
\]
However, if \( x_i \neq x_j \), then \( S(x_i, x_j) \to 0 \) at the limit \( \gamma \to \infty \) by assumption. Because the size of the training set \( m \) is assumed to be finite, \( M_i \to \lambda_i \) as \( \gamma \to \infty \).

Thus, the 1-norm SVM optimization problem can be made arbitrarily close to the following LP:

\[
\text{minimize} \quad \lambda, \xi \quad \sum_{i=1}^{m} \lambda_i \to (1 + \epsilon) \sum_{i=1}^{m} \xi_i \\
\text{subject to} \\
\lambda_i + \xi_i \geq 1 \quad (\text{for all } i) \\
\lambda, \xi \geq 0
\]

Because \( \epsilon > 0 \), the optimal solution is given by \( \lambda_i = 1 \) for all \( i \).

Next, suppose we have a new observation \( x \), and let \( x_j \) be its nearest neighbor in the training set with respect to the semi-metric \( d \). Then:

\[
\lim_{\gamma \to \infty} \frac{\sum_{i=1}^{m} S(x, x_i)}{S(x, x_j)} = 1 + \lim_{\gamma \to \infty} \sum_{i \neq j} S(x, x_i) S(x, x_j) = 1 + \sum_{i \neq j} \lim_{\gamma \to \infty} \frac{S(x, x_i)}{S(x, x_j)} = 1
\]

Here, we interchanged the limit and the summation because the sum is finite. The above equation shows that as \( \gamma \to \infty \), class similarity is dominated by the nearest neighbor. Hence, 1-norm SVM using the prediction rule in Eq. (7.2) can be made arbitrarily close to the 1-NN rule using a sufficiently large bandwidth \( \gamma \).

Aside from the extreme case in which 1-norm SVM reduces to 1-NN classification, the 1-norm SVM method can, in general, be interpreted as a weighted nearest neighbor classification algorithm as depicted in Fig. 7.1. In this figure, every support vector in 1-norm SVM exerts some influence in its vicinity in the instance space \( X \), where the “amount” of influence is determined by its weight \( \lambda_i \). For a new instance \( x \), the prediction rule in (7.2) becomes a weighted nearest neighbor rule, where the
weight of a neighbor \( x_i \) is determined by the product of its influence \( \lambda_i \) and its similarity \( S(x, x_i) \) to the new instance \( x \). Such an interpretation of 1-norm SVM holds because \( \lambda_i \geq 0 \) for all \( i \geq 1 \). Later in Section 7.5 we will show that 1-norm SVM is superior in its predictive accuracy over the \( k \)-NN classifier, despite the apparent similarity between the two methods.

### 7.4.4 Neural Networks

In addition to SVM and nearest neighbor classifiers, the 1-norm SVM method is closely connected to neural networks as well. This can be observed in the decision rule in (7.2) or (7.5), which can be interpreted as a neural network with one hidden layer and one output node as depicted in Fig. 7.2. The similarity functions \( S_j = S(\cdot, x_j) \) form the activation functions in the hidden nodes, whereas the bias term \( \lambda_0 \) is the activation threshold at the output node. When similarity functions are radial, i.e. functions of distance, such neural networks are commonly referred to as Radial Basis Function (RBF) networks.

This connection between 1-norm SVM and two-layer neural networks leads to two important observations that are related to the risk of under-fitting and over-fitting respectively. The first observation shows that 1-norm SVM is not vulnerable to the risk of under-fitting because its decision rule has the capacity to approximate any function arbitrarily well if the similarity function is radial. The second observation shows that 1-norm SVM is not vulnerable to the risk of over-fitting because its regularization term minimizes the “size” of the weights at the output layer of the neural network.

**Universal Function Approximation**

The first key observation is related to function approximation. Ideally, we would like the decision rule in the neural network in Fig. 7.2 to be as close as possible to the optimal Bayes rule. In principle, therefore, classification is reduced to function
Figure 7.2: The 1-norm SVM method can be interpreted as a method of function approximation using two-layer neural networks. In this figure, \( S_i = S(\cdot, x_i) : \mathcal{X} \to \mathbb{R} \) is the similarity function with the training instance \( x_i \). The activation threshold at the output node is the bias term \( \lambda_0 \).

approximation. However, it has been established that if the instance space is the Euclidean plane \( \mathbb{R}^d \) and the similarity function is radial, then the RBF neural network of Fig. 7.2 with a fixed bandwidth is capable of universal function approximation under mild conditions [177]. Consequently, if the training set is sufficiently large, such that 1-norm SVM can freely choose its support vectors in the plane, then 1-norm SVM has the capacity to approximate the optimal decision rule arbitrarily well. Hence, its risk for under-fitting is limited.

Size of the Weights

The second key observation is related to the size of the weights at the output layer. Given an instance space \( \mathcal{X} \) and a two-layer neural network with fixed activation functions at the hidden nodes, let \( \mathcal{H} \) denotes the set of all hypotheses that can be produced by the two-layer neural network. Suppose \( (x, y) \sim \mathcal{D} \) are drawn i.i.d. Then, [178] has shown that the following bound on true risk (prediction error rate) holds uniformly for all \( f \in \mathcal{H} \) with a probability of at least \( 1 - \delta \) over the random choice of
$m$ training examples (Theorem 1 in \[178\]):

$$R_D(f) \leq \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}\{y_i f(x_i) < M\} + \epsilon(M, m, \delta),$$

where $R_D(f) = \mathbb{P}_{(x,y) \sim \mathcal{D}}(y f(x) < 0)$ is the true risk. Using the notions of the fat-shattering dimension and Lipschitz continuity, it can be shown that $\epsilon(M, m, \delta) = \tilde{O}(\|\lambda\|_1)$ for the decision rule of the 1-norm SVM method \[178\]. As a result, the true risk is bounded uniformly across the hypothesis space $\mathcal{H}$ by:

$$R_D(f) \leq \|\lambda\|_1 + \frac{C(M)}{m} \sum_{i=1}^{m} \mathbb{I}\{y_i f(x_i) < M\},$$

(7.9)

where $C(M)$ is a function of $M$. A similar bound can be obtained that holds uniformly for all $0 < M < 1$ \[178\]. Contrasting the latter bound with the earlier margin-based bound in (7.6) suggests that minimizing $\|\lambda\|_1$ in the 1-norm SVM method plays a role that is similar to minimizing the complexity of the hypothesis space. A similar conclusion can be inferred more directly using Lagrange duality\[^6\]. In particular, the bound in (7.6) reveals that the regularization term $\|\lambda\|_1$ is preferable to other norms.

Because the activation functions at the hidden nodes in 1-norm SVM are not fixed, since they do depend on the random choice of training examples, the bound in (7.9) does not hold for 1-norm SVM. Nevertheless, it provides an informal justification to the use of 1-norm SVM with indefinite similarities since it compares favorably well with the objective function given in (7.4). In particular, the first term is the $\ell_1$ regularization term, which is identical in both expressions. Moreover, the second term in (7.9) is related to the hinge loss, which is the second term of the objective function in 1-norm SVM. In 1-norm SVM, both terms are minimized.

\[^6\]Using Lagrange duality, it is well-known that adding $\ell_p$ regularization on some optimization variable $\lambda$ in the objective function is equivalent to setting some upper bound on $\|\lambda\|_p$ (see for instance \[70\]). This shows that minimizing $\|\lambda\|_1$ in the 1-norm SVM method indeed plays the role of minimizing the complexity of the hypothesis space.
7.4.5 Structural Risk Minimization

The connection between 1-norm SVM and neural networks reveals that 1-norm SVM can be interpreted as a method of striking a balance between under-fitting and over-fitting. A similar conclusion can be established using the LOO error estimation method. To do this, we begin with the following lemma.

Lemma 3. Let \( S_m = \{(x_1, y_1), \ldots, (x_m, y_m)\} \) be a fixed set of \( m \) training examples, which is used to train the 1-norm SVM. Let \( \lambda^* \) and \( \xi^* \) be the optimal solutions of the LP in (7.4). Also, let \( e_{LOO} \) be the expected leave-one-out validation error rate on the same training set. Then:

\[
e_{LOO} \leq \frac{||\lambda^*||_0 + ||\xi^*||_0}{m} \tag{7.10}
\]

Here, \( ||z||_0 \) denotes the number of non-zero entries in \( z \).

Proof. Let \( \lambda^* \) and \( \xi^* \) be the optimal solutions to the 1-norm SVM in (7.4). If \( \xi^*_i = \lambda^*_i = 0 \), then the \( i \)-th training example was classified correctly and it will continue to be classified correctly if it is the only example removed from the training set. The latter statement holds because removing the \( i \)-th example from the training set is equivalent to adding the new constraint \( \lambda_i = \xi_i = 0 \) to the LP formulation (7.4), which is the original optimal value of \( \lambda^*_i \) and \( \xi^*_i \). Because the new feasibility region is a subset of the original feasibility region and it contains the original optimal solution, the optimal solution remains unchanged. Hence:

\[
e_{LOO} \leq \frac{||\lambda^* + \xi^*||_0}{m} \leq \frac{||\lambda^*||_0 + ||\xi^*||_0}{m},
\]

Theorem 8. Let \( f_S \) be a random variable that stands for the hypothesis produced by 1-norm SVM when trained on a randomly selected training set \( S \). Let \( R_D(f_S) \) be the
true risk (prediction error rate) of the hypothesis \( f_S \). Then:

\[
\mathbb{E}_{S_{m-1}}[R_D(f_{S_{m-1}})] \leq \frac{\mathbb{E}_{S_m}[\|\lambda\|_0]}{m} + \frac{\mathbb{E}_{S_m}[\|\xi\|_0]}{m}
\]  

(7.11)

Here, expectation of the true risk is taken over all possible training sets of size \( m - 1 \) whereas remaining expectations are taken over all possible training sets of size \( m \).

**Proof.** By the Luntz-Brailovsky theorem \[179, 180\], we have:

\[
\mathbb{E}_{S_{m-1}}[R_D(f_{S_{m-1}})] = \mathbb{E}_{S_m}[e_{\text{LOO}}],
\]  

(7.12)

where \( e_{\text{LOO}} \) is the leave-one-out validation error. Using Eq. (7.12) and Lemma 3 yields the desired result. 

The tradeoff in Eq. (7.11) is analogous to the classical tradeoff in estimation between bias and variance \[181\]. On one hand, one can fit the training set perfectly, e.g. by using a radial similarity function with a sufficiently large bandwidth that effectively turns 1-norm SVM into a 1-NN classifier, but the number of support vectors becomes at its worst, hence high variance. On the other hand, one can choose a tiny number of support vectors but this tends to increase the empirical risk (training error rate), hence high bias. In the 1-norm SVM formulation in (7.4), the cost function penalizes both training error (bias) and the number of support vectors (variance) simultaneously by penalizing the \( \| \cdot \|_1 \) of slack variables \( \xi \) and weights \( \lambda \). Because minimizing \( \| \cdot \|_1 \) promotes sparsity \[123\], Corollary \[8\] states that 1-norm SVM can be interpreted as a method of minimizing the expected true risk. Hence, it is an approximate implementation of the structural risk minimization (SRM) induction principle \[11\].
7.5 Experiments and Results

In the previous section, we described theoretically why 1-norm SVM was a viable tool for classification with indefinite similarities. In this section, we present experimental results of applying the 1-norm SVM algorithm to synthetic and real-world classification problems, which demonstrate its effectiveness in handling indefinite similarity functions.

7.5.1 Synthetic Datasets

First, 1-norm SVM was tested on six synthetic datasets depicted in Fig. 7.3. In these datasets, the Radial Basis Function $S(x_i, x_j) = \exp\{-\gamma \|x_i - x_j\|^2\}$ was used, where the bandwidth parameter $\gamma$ was selected using a grid search on a separate validation set. Figure 7.4 plots the test error rate as a function of training set size $m$, with the Bayes risk for each classification problem indicated in the legend bar. As shown in Fig. 7.4, the test error rate approaches the optimal Bayes risk for sufficiently large training sets in all six classification problems. This test verifies that 1-norm SVM is capable of producing accurate decision boundaries for various complex mixtures of classes, which is consistent with the universal approximation property of the RBF similarity function discussed earlier in Section 7.4.4.

7.5.2 Real Datasets

For real datasets, we compared the performance of 1-norm SVM against popular classification algorithms for both PSD and non-PSD similarity functions. We will first describe the datasets and test methodology, and discuss the test results afterwards.

Datasets

The following datasets and similarity functions were used:

(A) IMDB: This is a graph-based dataset that contains movies released between...
Figure 7.3: The six synthetic datasets that are used in evaluating 1-norm SVM. The Bayes risk of each dataset is indicated in the legend bar in Fig. 7.4.

1996 and 2001 [182]. The binary class label identifies whether the opening weekend box-office receipts exceeded $2 million or not. An edge weight between two movies is the number of common production companies, actors, producers, or directors. In our implementation, all edge weights were normalized to fall in the range [0, 1], and the following similarity functions were used:

(a) PSD: The Jaccard index $S_{i,j} = \frac{\sum_k \min\{w_{i,k}, w_{j,k}\}}{\sum_k \max\{w_{i,k}, w_{j,k}\}}$, where $w_{i,k}$ is the edge weight.
(b) Non-PSD: Edge weight $S_{i,j} = w_{i,j}$ and $S_{i,i} = 1$.

(B) Word-Sim-353: This dataset contains human-judged similarities between English words [183]. All similarities are again normalized to fall in the range [0, 1] and self-similarity is set to unity. We grouped words into two categories: ‘living’ vs. ‘non-living’, and used the two similarity functions specified earlier for the IMDB dataset. Examples of the ‘living’ class include children, Maradona, brother, carnivore, and mammal.
Figure 7.4: Performance of 1-norm SVM on the six synthetic datasets. Each error rate, plotted in a log-scale, is an average of five i.i.d training/test sets. P1, . . . , P6 stands for Problem 1, . . . , Problem 6 shown in Fig. 7.3. The optimal Bayes risks are indicated in the legends bar. In the y-axis, each grid line between $10^z$ and $10^{z+1}$ is to be read as $1 \times 10^z$, $2 \times 10^z$, . . . , $9 \times 10^z$. For example, the grid lines from $10^1$ to $10^2$ correspond to the values 10, 20, . . . , 80, 90.

(C) Caltech-101: This dataset contains images of various objects [184]. We grouped images of ‘Big Cats’, ‘Winged Insects’, and ‘Flowers’ into three classes and trained three separate binary classifiers between every pair of classes. Each image was converted into a histogram using the two MATLAB commands `rgb2gray` and `imhist`, and Laplace normalization was used. This effectively represents the $i$-th image by a probability distribution $p_i$. We, then, used the following two similarity functions:

(a) PSD: The intersection (a.k.a. overlapping coefficient), which is given by

$$S_{i,j} = \sum_k \min\{p_{i,k}, p_{j,k}\}. $$

(b) Non-PSD: We used $S_{i,j} = \max\{0, 1 - 0.1 \times D(p_i \| \| p_j)\}$, where $D(p_i \| \| p_j)$ is the symmetrized Kullback-Leibler divergence.

(D) Splice: This is a biological sequence classification dataset [185] that was down-

---

7The reason behind choosing 0.1 is because 95% of pairwise distances are less than 10.
loaded from the UCI repository [155]. Each example is a 60-letter DNA sequence. We performed classification between the two classes EI and IE. The similarity functions used were:

(a) PSD: We used the implementation of string subsequence kernels given in [180]. Because string kernels can grow quite rapidly, we normalized using the cosine similarity: 

\[ S_{i,j} = \frac{K_{i,j}}{\sqrt{K_{i,i} K_{j,j}}}. \]

(b) Non-PSD: We used the longest-common-subsequence (LCS) between two strings. Because each string is 60 letters in length, we set 

\[ S_{i,j} = \frac{\text{LCS}(x_i, x_j)}{60}. \]

(E) **CNAE-9**: This is a text classification dataset available at the UCI repository, where each text is represented using a bag of words. The dataset contains nine classes and we randomly selected five binary classification problems: 1-vs-5, 5-vs-4, 6-vs-8, 3-vs-9, and 2-vs-7. These are represented by P1 through P5 in Table 7.2 respectively. The two similarity functions are:

(a) PSD: The cosine similarity 

\[ S_{i,j} = \frac{x_i^T x_j}{||x_i|| ||x_j||}, \]

which is commonly used for text classification tasks [159].

(b) Non-PSD: The second similarity function used is a variant to the first. Specifically, we have 

\[ S_{i,j} = \frac{v^T v}{||x_i|| ||x_j||}, \]

where \( v_k = \min\{x_{i,k}, x_{j,k}\} \).

(F) **Ionosphere, Australian, Breast Cancer, Haberman, and Diabetes**: These are five binary classification problems with numeric features available at the UCI repository. We used the following similarity functions:

(a) PSD: The RBF kernel 

\[ S_{i,j} = e^{-\gamma ||x_i - x_j||_2^2}, \]

which is considered the default similarity function for numeric attributes in popular SVM packages such as LIBSVM [156].

---

8We performed a random permutation of the set of integers \( \{1, 2, \ldots, 9\} \). Each pair of adjacent labels was used as a binary classification problem, where the 9th label is trained vs. the 1st.
(b) Non-PSD: The sigmoid kernel \( S_{i,j} = \tanh(\gamma \cdot x_i^T x_j + r) \), which is popular due to its origins in neural networks. To ensure that the kernel matrix is not PSD, we fixed \( r = -\frac{1}{9} \).

Test Methodology and Results

When the similarity function is PSD, we compared the performance of 1-norm SVM vs. the standard form of SVM in (7.1) and k-NN. For each dataset, the values of the tradeoff constant \( C \) and \( k \) were selected using 5-fold cross validation for \( C \in \{2, 4, 8, 16, 32\} \) and \( k \in \{2, 4, 6, 8, 10, 12, 14\} \). When the RBF kernel is used, the bandwidth \( \gamma \) is also selected using 5-fold cross validation in the grid \( \gamma \in \{2^{-15}, 2^{-14}, \ldots, 1\} \). SVM was implemented using the LIBSVM library [156], whereas 1-norm SVM was implemented using the Gurobi solver [173]. In all classification problems, we reported the average test error rate of five random training-to-test splits, with a training-to-split ratio of 4:1. The same split is always used in both SVM and 1-norm SVM.

When the similarity function is indefinite (non-PSD), we compared the performance of 1-norm SVM against the four dominant methods used in the literature:

1. Non-convex optimization: This was implemented using the LIBSVM library with its \( -t 4 \) option. When the similarity matrix is non-PSD, the LIBSVM package seeks a stationary point using non-convex optimization [167].

2. Kernel approximation: PSD kernel approximation was tested using the three methods discussed earlier in Section 7.2: (1) the denoise method, (2) the flip method, and (3) the indefinite SVM proposed by Luss and d’Aspremont [165]. The denoise and flip methods were implemented by supplying the modified (PSD) kernel matrix to LIBSVM using the \( -t 4 \) option. The indefinite SVM

---

\( ^9 \)It has been shown that the sigmoid kernel is PSD only if \( r \geq 0 \) [187]. However, using \( r < 0 \) tends to perform better [167].
method was tested using the implementation available for download at the authors' website.

3. Nearest Neighbor Classification: Here, k-NN was implemented on the original indefinite similarity function.

4. SVM in Krien Spaces: SVM in Krien spaces was implemented using the ESVM algorithm described in [168]. ESVM comprises of two main steps: (1) eigen-decomposition, and (2) SVM training. LIBSVM was used for the SVM training step.

In all methods, hyper-parameters were selected using cross validation and grid search, implemented separately for each individual method. Test results for PSD and non-PSD similarity functions are shown in Table 7.1 and 7.2 respectively. Because the datasets are balanced, we used classification error rate as a measure of performance. All results reported here are based on the best selected hyper-parameters of these methods.

7.5.3 Discussion

In this section, we review the test results when using PSD and non-PSD similarity functions for the 16 real datasets described earlier.

Positive Semidefinite Similarities

We begin our discussion by looking into the test results for positive semidefinite (PSD) similarities. As shown in columns 2 and 3 of Table 7.1 when the similarity function is PSD, the performance of 1-norm SVM is comparable to that of SVM for all 16 datasets. When running statistical significance tests, we find no statistically significant evidence that one method outperforms the other at the 95% confidence level. For example, the two-tailed Wilcoxon’s signed rank test [188] gives a value of $p = 0.155$. By contrast, both algorithms tend to outperform $k$-NN in classification
Table 7.1: Average test error rate results on 16 datasets using the positive semidefinite (PSD) similarity functions described in Section 7.5.2. Here, the number of training examples $m$ for each dataset is shown in parentheses.

<table>
<thead>
<tr>
<th>Dataset ($m$)</th>
<th>1-norm svm</th>
<th>SVM</th>
<th>k-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB (1441)</td>
<td>16.0%</td>
<td>15.6%</td>
<td>21.2%</td>
</tr>
<tr>
<td>Word-Sim-353 (437)</td>
<td>12.9%</td>
<td>13.7%</td>
<td>13.3%</td>
</tr>
<tr>
<td>Caltech-101-P2 (368)</td>
<td>26.0%</td>
<td>24.0%</td>
<td>40.0%</td>
</tr>
<tr>
<td>Caltech-101-P3 (379)</td>
<td>19.8%</td>
<td>19.2%</td>
<td>33.1%</td>
</tr>
<tr>
<td>Caltech-101-P1 (387)</td>
<td>31.7%</td>
<td>31.7%</td>
<td>38.7%</td>
</tr>
<tr>
<td>Splice (1527)</td>
<td>6.56%</td>
<td>5.79%</td>
<td>10.3%</td>
</tr>
<tr>
<td>CNAE-9-P1 (240)</td>
<td>0.56%</td>
<td>0%</td>
<td>0.42%</td>
</tr>
<tr>
<td>CNAE-9-P5 (240)</td>
<td>2.05%</td>
<td>1.15%</td>
<td>1.67%</td>
</tr>
<tr>
<td>CNAE-9-P3 (240)</td>
<td>1.67%</td>
<td>0.94%</td>
<td>2.50%</td>
</tr>
<tr>
<td>CNAE-9-P2 (240)</td>
<td>0.44%</td>
<td>1.22%</td>
<td>1.25%</td>
</tr>
<tr>
<td>CNAE-9-P4 (240)</td>
<td>1.89%</td>
<td>0.83%</td>
<td>1.67%</td>
</tr>
<tr>
<td>Ionosphere (351)</td>
<td>7.14%</td>
<td>6.57%</td>
<td>14.6%</td>
</tr>
<tr>
<td>Australian (690)</td>
<td>16.6%</td>
<td>16.9%</td>
<td>14.4%</td>
</tr>
<tr>
<td>Breast Cancer (699)</td>
<td>3.15%</td>
<td>3.51%</td>
<td>4.75%</td>
</tr>
<tr>
<td>Haberman (398)</td>
<td>30.2%</td>
<td>31.2%</td>
<td>22.5%</td>
</tr>
<tr>
<td>Diabetes (768)</td>
<td>28.9%</td>
<td>27.9%</td>
<td>26.3%</td>
</tr>
</tbody>
</table>
Table 7.2: Average test error rate results on 16 datasets using the indefinite (non-PSD) similarity functions described in Section 7.5.2. In this table, $0 \leq \beta \leq 1$ is a measure of how indefinite the similarity function is. In particular, a value of $\beta = 0$ corresponds to PSD similarity functions while a value of $\beta = 1$ corresponds to negative semidefinite similarities. The rows in the table are ordered by the value of $\beta$. The acronym NCO stands for non-convex optimization, DEN for the denoise method, FLIP for the flip method, ISVM for indefinite SVM, while ESVM stands for the eigen-decomposition SVM method.

<table>
<thead>
<tr>
<th>Dataset ($m$)</th>
<th>$\beta^{(1)}$</th>
<th>1-NORM svm</th>
<th>k-NN</th>
<th>NCO Approximation</th>
<th>Kernel space</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SVM</td>
<td></td>
<td>SVM</td>
<td>DEN</td>
</tr>
<tr>
<td>IMDB (1441)</td>
<td>0.01</td>
<td>18.8%</td>
<td>26.1%</td>
<td>17.7%</td>
<td>17.6%</td>
</tr>
<tr>
<td>Word-Sim-353 (437)</td>
<td>0.03</td>
<td>14.7%</td>
<td>14.9%</td>
<td>15.6%</td>
<td>15.2%</td>
</tr>
<tr>
<td>Caltech-P2 (368)</td>
<td>0.05</td>
<td>22.1%</td>
<td>27.7%</td>
<td>44.0%</td>
<td>62.0%</td>
</tr>
<tr>
<td>Caltech-P3 (379)</td>
<td>0.08</td>
<td>22.8%</td>
<td>26.7%</td>
<td>40.7%</td>
<td>54.5%</td>
</tr>
<tr>
<td>Caltech-P1 (387)</td>
<td>0.09</td>
<td>30.1%</td>
<td>35.6%</td>
<td>40.9%</td>
<td>50.8%</td>
</tr>
<tr>
<td>Splice (1527)</td>
<td>0.10</td>
<td>5.70%</td>
<td>7.49%</td>
<td>5.74%</td>
<td>5.97%</td>
</tr>
<tr>
<td>CNAE-9-P1 (240)</td>
<td>0.32</td>
<td>0%</td>
<td>8.30%</td>
<td>13.2%</td>
<td>6.17%</td>
</tr>
<tr>
<td>CNAE-9-P5 (240)</td>
<td>0.33</td>
<td>3.72%</td>
<td>6.25%</td>
<td>10.9%</td>
<td>6.61%</td>
</tr>
<tr>
<td>CNAE-9-P3 (240)</td>
<td>0.34</td>
<td>2.50%</td>
<td>24.6%</td>
<td>25.3%</td>
<td>25.4%</td>
</tr>
<tr>
<td>CNAE-9-P2 (240)</td>
<td>0.34</td>
<td>0.33%</td>
<td>22.5%</td>
<td>21.6%</td>
<td>20.7%</td>
</tr>
<tr>
<td>CNAE-9-P4 (240)</td>
<td>0.34</td>
<td>2.56%</td>
<td>4.17%</td>
<td>16.3%</td>
<td>9.67%</td>
</tr>
<tr>
<td>Ionosphere (351)</td>
<td>1.0</td>
<td>10.3%</td>
<td>18.6%</td>
<td>37.7%</td>
<td>70.3%</td>
</tr>
<tr>
<td>Australian (690)</td>
<td>1.0</td>
<td>12.2%</td>
<td>18.4%</td>
<td>40.7%</td>
<td>40.7%</td>
</tr>
<tr>
<td>Breast Cancer (699)</td>
<td>1.0</td>
<td>4.32%</td>
<td>2.88%</td>
<td>32.9%</td>
<td>32.9%</td>
</tr>
<tr>
<td>Haberman (398)</td>
<td>1.0</td>
<td>26.6%</td>
<td>24.8%</td>
<td>26.6%</td>
<td>40.1%</td>
</tr>
<tr>
<td>Diabetes (768)</td>
<td>1.0</td>
<td>22.7%</td>
<td>33.2%</td>
<td>33.3%</td>
<td>70.3%</td>
</tr>
</tbody>
</table>

\* $\beta = \sum_{i} \frac{1}{|\lambda_i|} \cdot \frac{1}{\|\lambda_i < 0\|}$, where $\lambda_i$ are eigenvalues of the similarity matrix.

(2) The algorithm failed to terminate.
accuracy. This validation verifies that 1-norm SVM is a viable algorithm for binary classification even when the similarity function is positive semidefinite (PSD). This conclusion agrees with previous results in the literature [89].

**Indefinite Similarities**

In contrast to the previous case, the use of indefinite similarity functions presents an entirely different picture. When comparing the test error rate of 1-norm SVM (shown in column 3 of Table 7.2) with the other methods (in columns 4-9), we find that 1-norm SVM and ESVM (i.e. learning in Krein spaces) outperform all other methods significantly in nearly all the datasets. The performance of ESVM, however, is very similar to that of 1-norm SVM, which is quite intriguing given the very different approaches employed by the two algorithms.

Nevertheless, unlike ESVM whose solution is quite dense, the 1-norm SVM method yields very sparse solutions so that prediction time is faster. In fact, 1-norm SVM yields solutions that are often 10-20 times, sometimes even 100 times, sparser than ESVM. Table 7.3 lists the number of support vectors used by both methods.

In order to verify statistical significance at the 95% confidence level, we used Holm’s step-down procedure for multiple comparisons applied to the two-tailed Wilcoxon’s signed rank test [188, 189]. More specifically, each null hypothesis $H_i$ asserts that 1-norm SVM and the $i$-th alternative classifier have similar performance. When $H_i$ is tested using the two-tailed Wilcoxon’s signed rank test, the resulting $p$ values are shown in Table 7.4. Using a confidence level of 95% in Holm’s step-down procedure, we find that the null hypothesis is rejected for non-convex optimization, k-NN, and all kernel approximation methods. This confirms that 1-norm SVM outperforms non-convex optimization and kernel approximation with a statistically significant evidence.

However, there is no statistically significant evidence at the 95% confidence level that 1-norm SVM outperforms ESVM in terms of predictive accuracy. Here, it is
Table 7.3: The number of support vectors SV used by 1-norm SVM and ESVM for the 16 classification problems with indefinite similarity functions.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>No. of training examples</th>
<th>No. of SVs in 1-norm SVM</th>
<th>No. of SVs in ESVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB</td>
<td>1441</td>
<td>630</td>
<td>1438</td>
</tr>
<tr>
<td>Word-Sim-353</td>
<td>437</td>
<td>26</td>
<td>436</td>
</tr>
<tr>
<td>Caltech-101-P2</td>
<td>368</td>
<td>39</td>
<td>386</td>
</tr>
<tr>
<td>Caltech-101-P3</td>
<td>379</td>
<td>47</td>
<td>379</td>
</tr>
<tr>
<td>Caltech-101-P1</td>
<td>387</td>
<td>35</td>
<td>386</td>
</tr>
<tr>
<td>Splice</td>
<td>1527</td>
<td>224</td>
<td>1527</td>
</tr>
<tr>
<td>CNAE-9-P1</td>
<td>240</td>
<td>2</td>
<td>233</td>
</tr>
<tr>
<td>CNAE-9-P5</td>
<td>240</td>
<td>21</td>
<td>240</td>
</tr>
<tr>
<td>CNAE-9-P3</td>
<td>240</td>
<td>23</td>
<td>238</td>
</tr>
<tr>
<td>CNAE-9-P2</td>
<td>240</td>
<td>2</td>
<td>235</td>
</tr>
<tr>
<td>CNAE-9-P4</td>
<td>240</td>
<td>23</td>
<td>240</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>39</td>
<td>351</td>
</tr>
<tr>
<td>Australian</td>
<td>690</td>
<td>7</td>
<td>690</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>699</td>
<td>20</td>
<td>699</td>
</tr>
<tr>
<td>Haberman</td>
<td>398</td>
<td>28</td>
<td>398</td>
</tr>
<tr>
<td>Diabetes</td>
<td>768</td>
<td>13</td>
<td>768</td>
</tr>
</tbody>
</table>

perhaps worth reiterating that the 1-norm SVM significantly outperforms ESVM in terms of sparsity of solutions as shown in Table 7.3. Therefore, the 1-norm SVM method achieves the highest predictive accuracy among all methods that learn with indefinite similarities, while also retaining sparsity of the support vector set.

Finally, it is worth pointing out that indefinite similarity functions in our evaluation led to lower test error rates than PSD similarity functions in roughly 50% of the datasets. This includes, most notably, the datasets: caltech-101-p2, australian, haberman, and diabetes. Therefore, even for classification problems where PSD similarity functions are readily available, learning with non-PSD kernels remains important because it can result in a better classification accuracy.

7.6 Conclusions

One technique of mitigating the adversarial risk of reverse engineering is to use a similarity-based non-linear classifier. However, popular off-the-shelf non-linear classi-
Table 7.4: In this table, the second column lists the $p$ values in increasing order of the two-tailed Wilcoxon’s signed rank test. The last column shows the critical values when Holm’s step-down procedure is used at the 95% confidence level.

<table>
<thead>
<tr>
<th>Null Hypothesis</th>
<th>$p$ value</th>
<th>Adjusted Critical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-norm SVM vs. SVM with</td>
<td>0.0003</td>
<td>0.0083</td>
</tr>
<tr>
<td>non-convex optimization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-norm SVM vs. Denoise</td>
<td>0.0008</td>
<td>0.0100</td>
</tr>
<tr>
<td>1-norm SVM vs. $k$-NN</td>
<td>0.0016</td>
<td>0.0125</td>
</tr>
<tr>
<td>1-norm SVM vs. Flip</td>
<td>0.0052</td>
<td>0.0167</td>
</tr>
<tr>
<td>1-norm SVM vs. Indefinite SVM</td>
<td>0.0107</td>
<td>0.0250</td>
</tr>
<tr>
<td>1-norm SVM vs. ESVM</td>
<td>0.0771</td>
<td>0.0500</td>
</tr>
</tbody>
</table>

Classifiers are often effective only if the similarity function is positive semidefinite. In this chapter, we show theoretically and experimentally how the 1-norm support vector machine is a better method for handling indefinite similarities. The 1-norm SVM method formulates large-margin separation as a convex linear programming (LP) problem without requiring that the similarity function be positive semidefinite (PSD). It uses the indefinite similarity function directly without any transformation, and, hence, it always treats both training and test examples consistently. Furthermore, by relating 1-norm SVM with neural networks and error bounds of the leave-one-out estimation method, the 1-norm SVM method can be interpreted as an approximate implementation of the structural risk minimization (SRM) induction principle. Hence, it is robust against the risks of under-fitting and over-fitting. Finally, 1-norm SVM indeed achieves the highest accuracy among all previous methods for classification with indefinite similarities with a statistically significant evidence, while also retaining sparsity of the support vector set.
Part IV

Conclusion
Chapter 8

Conclusion

Active learning is a subfield of machine learning that has received a growing interest over the past 20 years. It has found many important applications, including text classification, speech recognition, image retrieval, character recognition, span filtering, and bioinformatics. In addition, many software companies are increasingly reliant on active learning techniques, such as Google, IBM, and CiteSeer.

One of the main branches of active learning is query synthesis, where the learning agent constructs artificial queries from scratch in order to reveal sensitive information about the underlying decision boundary. It has an advantage over pool-based sampling methods because it can achieve an exponential saving in the sample complexity without having to maintain an exponentially increasing pool of examples. It has found applications in areas, such as adversarial reverse engineering, automated science, and computational chemistry. Nevertheless, the existing literature on membership query synthesis has, generally, focused on finite concept classes or toy problems, with a limited extension to real-world applications.

In this thesis, I developed an efficient query synthesis algorithm for halfspaces. The new algorithm can be interpreted as an approximation to the ideal halving algorithm, whose prototypical example is the bisection method in \( \mathbb{R} \). At each iteration, the new algorithm consists of two steps. First, a maximum-determinant convex optimization problem is solved that provides an approximate characterization of the version space. This optimization problem can be cast as a Second-Order Cone Programming (SOCP) problem, for which many efficient solvers currently exist. In the
second step, a principal component is extracted via spectral decomposition, which yields the optimal synthetic query that shrinks the version space exponentially fast.

In addition to the classical active learning setting in which a single query is synthesized at each iteration, the proposed algorithm can also be readily extended into the batch setting, where $1 \leq k \leq d$ queries are synthesized per iteration. Batch-mode active learning has been recently described as one of the top challenges in active learning. It is important in distributed parallel labeling environments, such as in biology. Ideally, batch-mode active learning should produce queries that are both informative and diverse. In our case, the new proposed algorithm operates in the batch setting by solving for the top $k$ eigenvectors in the second step of each iteration, thus achieving both objectives.

Experimentally, the proposed spectral algorithm significantly outperforms all other methods in both the single-query setting and the batch-mode setting. In fact, unlike the other methods whose estimation error is subject to diminishing returns, the new proposed method always maintains an exponential reduction in estimation error. Moreover, it achieves the optimal sample complexity of $O(d \log \frac{1}{\epsilon})$ in practice. This performance has been justified analytically as well.

However, there are two main drawbacks of the first query synthesis algorithm. First, it relies on solving a maximum-determinant convex optimization problem, whose space and time complexity grows with time as more queries are added. This follows from the fact that the number of constraints in the optimization problem is equal to the number of queries that have been accumulated so far. Second, the algorithm repeatedly solves the entire optimization problem from scratch at each round without incorporating the knowledge gained in previous rounds.

In the second chapter of this thesis, these issues are resolved by developing a second Markovian query synthesis algorithm. The new algorithm encodes all of the relevant information about the past in its present state. That is, a query is used
only once, and is immediately discarded afterward. Consequently, the new algorithm enjoys a fixed space and time complexity at all rounds. In fact, it utilizes a variant of the classical Khachiyan’s update formulas for solving linear programs, which can be computed very efficiently. Experimentally, the savings in space and time are quite significant, compared to the first algorithm.

However, the advantages of the second algorithm come at a cost. The price paid for using the new algorithm is an increase in the sample complexity. In particular, whereas the earlier query synthesis algorithm enjoys an optimal sample complexity of $O(d \log \frac{1}{\epsilon})$ in practice, the new algorithm has a sample complexity of $O(d^2 \log \frac{1}{\epsilon})$. Moreover, whereas the previous algorithm could handle a batch size of up to $k \leq d$ queries at each round, the new algorithm can handle a batch size of up to $k < \frac{(d+1)}{2}$ queries. Therefore, the new algorithm is advantageous in settings where the savings in space and time complexity outweigh the increase in sample complexity.

Having demonstrated that halfspaces can be learned quite efficiently via query synthesis, the second part of this thesis proposes strategies for mitigating the risk of reverse engineering in adversarial environments. In practice, many machine learning algorithms have been successfully deployed in security-sensitive applications. These include spam and malicious email behavior detection, fraud detection, as well as intrusion detection. The machine learning techniques employed cover a broad spectrum ranging from supervised learning algorithms, such as neural networks and $k$ Nearest Neighbor (k-NN) to unsupervised outlier detection algorithms, such as density estimation-based methods and one-class Support Vector Machine (SVM). Linear classifiers, such as SVM and logistic regression, are also commonly used in practice today, because they can often achieve comparable accuracy to non-linear classifiers in many applications. In fact, due to their wide popularity, many solvers currently exist for linear classification including SVMperf, Optimized Cutting Plane Algorithm (OCAS), Pegasos, Stochastic Gradient Descent (SGD), and LIBLINEAR. Since they
are easily reverse-engineered using the query synthesis algorithms developed in this thesis, it is quite important to protect these classifiers in adversarial environments.

In this thesis, I investigate randomization as a suitable strategy for mitigating the adversarial risk of reverse engineering. The rationale behind this approach stems from recent seminal results in approximation hardness. In particular, I develop a convex optimization formulation for learning a distribution of classifiers, subject to the constraint that any individual classifier picked at random from this distribution provides reliable predictions with a high probability. This convex optimization formulation is then cast as a Semidefinite Programming (SDP), which is readily handled by SDP solvers. Experimentally, the proposed SDP algorithm is more robust against the risk of reverse engineering than other baseline randomization methods.

Perhaps more importantly, I show, by experimenting on real datasets, that one can almost always introduce randomization with a large variance without impacting the predictive accuracy of the overall procedure. Hence, the use of fixed linear classifiers in adversarial environments is Pareto suboptimal. This observation is explained theoretically by proving a generalization bound that connects the classification accuracy, on one hand, with the variance of the distribution and its margin, on the other hand. The derived bound reveals that randomization can be introduced with a large variance without impacting accuracy as long as the majority of training examples are classified correctly with a large margin. Hence, by drawing classifiers at random from a distribution with a large variance, the adversary’s effort to carry out a reverse engineering attack will increase significantly at a little cost to the learning system.

A second approach for mitigating the adversarial risk of reverse engineering is to use a non-parametric classification method, such as similarity-based classification. The rationale behind this approach also stems from recent seminal results in approximation hardness. Similarly-based classification has been widely studied in the past. The contribution of this thesis is to show that learning via the empirical ker-
nel maps, also commonly referred to as 1-norm SVM or Linear Programming (LP) SVM, is the best method for handling indefinite similarities. In particular, the 1-norm SVM method is conceptually simple, which makes it easy to implement and maintain. It is competitive, if not superior to, all other methods in terms of its predictive accuracy. By contrast, common alternative approaches either introduced sources of inconsistency using kernel approximation, lacked learning guarantees using non-convex optimization, or produced dense solutions by learning in Krein spaces.

Indefinite similarity functions naturally arise in many settings. In bioinformatics, for example, the the Longest Common Subsequence (LCS) is often used to define the similarity between genetic sequences. In text mining, furthermore, one can use human-judged similarities between concepts and words or use the Kullback-Leibler divergence when treating documents as probability distributions of individual words. In the study of trajectories or time series in general, one can use the Dynamic Time Warping (DTW) and its many variants. In computer vision, the tangent distance or the shape matching distance are quite popular. Other indefinite similarities are also frequently encountered in psychology, neuroscience, and economics. Extending large-margin classification to indefinite similarities will have many important applications.

The advantages of 1-norm SVM over other methods in handling indefinite similarities is established in this thesis both theoretically and experimentally. On the theoretical side, we relate it to neural networks and large-margin classifiers and derive error bounds using the leave-one-out estimation method. These results show that the 1-norm SVM method can be interpreted as an approximate implementation of the Structural Risk Minimization (SRM) induction principle. Hence, it is robust against the risks of under-fitting and over-fitting. Experimentally, we carry out a thorough evaluation on synthetic and real datasets, which reveal that the evidence in favor of 1-norm SVM is found to be statistically significant at the 95% confidence level.
REFERENCES


[178] P. L. Bartlett, “For valid generalization, the size of the weights is more important than the size,” vol. 9, p. 134, 1997.


APPENDICES
Appendix A

The Class of $f$ Divergences

The class of $f$ divergences for measuring the amount of mutual information between random variables is becoming increasingly important in machine learning. The work of [190] has shown that $f$ divergences are related to surrogate loss functions, while my work in [54] has shown that a special member of the class of $f$ divergences, namely the total variation distance, can be used to provide a uniform generalization bound in expectation for machine learning algorithms. By exploiting inequalities between $f$ divergences such as the Pinsker inequality [61, 47], an elementary proof to the generalization risk in the finite hypothesis case can be derived. Other $f$ divergences can also be quite useful for analyzing machine learning algorithms due to the proliferation of inequalities in the literature that relate various members of the class of $f$ divergences to each other [61].

However, $f$ divergence are not the only possible methods of measuring the distance between probability distributions. Additional methods include, for instance, the Mahalanobis distance and the $\ell_p$ norms [191]. This raises the following important question: why do we restrict attention to the class of $f$ divergences? Could it be that a more profound measure of information between random variables exists that does not belong to the class of $f$ divergences? The purpose of this appendix is to answer this question. I will show that the only valid measure of information between random variables is the class of $f$ divergences. Specifically, I will show that $f$ divergences arise from three elementary axioms. Before I do that, I will briefly describe the class of $f$ divergences and discuss their fundamental properties.
The contents of this appendix are my own. However, similar (but not identical) axioms with different proofs have been proposed many years ago by I Csiszár [52, 53]. The purpose of this appendix is to give an overview of this important class of divergences and to simplify the proofs of its axiomatic basis.

A.1 Fundamental Properties of \( f \) Divergences

**Definition 14** (\( f \) Divergence). Let \( P \) and \( Q \) be two probability measures on a measurable space \( \Omega \). Then, the \( f \) divergence between \( P \) and \( Q \) with respect to some convex function \( f \) is defined by:

\[
D_f(P||Q) = \int_{\Omega} f\left(\frac{dP}{dQ}\right) dQ
\]

In Definition 14, it is immediately deduced that \( D_f(P||Q) = D_g(P||Q) \) for any probability measures \( P \) and \( Q \) if \( f(u) = g(u) + c(u - 1) \). Hence, it is customary to normalize \( f \) such that \( f(1) = 0 \). This convention has a useful interpretation. If \( P \) and \( Q \) are equal almost everywhere, then \( D_f(P||Q) = 0 \) if \( f \) is normalized. However, by Jensen’s inequality, we have:

\[
D_f(P||Q) = \mathbb{E}_Q f\left(\frac{dP}{dQ}\right) \geq f\left(\mathbb{E}_Q \frac{dP}{dQ}\right) = f(1) = 0
\]

This brings us to the following remark:

**Remark 1.** When \( f \) is normalized, i.e. \( f(1) = 0 \), the \( f \) divergence \( D_f(P||Q) \) is non-negative for any probability measures \( P \) and \( Q \). Furthermore, if \( f(u) \) is strictly convex at \( u = 1 \), then \( D_f(P||Q) = 0 \) if and only if \( P = Q \) almost everywhere.

The second important property is the data processing inequality. The data processing inequality has important implications in both information theory and in statistical learning theory. I have used it to provide a formal justification to some common
practices in machine learning such as post-processing the hypotheses, e.g. sparsification and pruning, as well as regularization by contaminating training examples with noise [54]. We prove the data processing inequality next.

Definition 15 (Generalized Mutual Informaiton). The generalized mutual information, a.k.a. informativity, between two random variables $X$ and $Y$ is defined by:

$$I_f(X;Y) = D_f(\mathbb{P}(X) \cdot \mathbb{P}(Y) \mid \mid \mathbb{P}(X,Y))$$

Proposition 2. If for three random variables $A, B, \text{ and } C$, we have the Markov chain $A \rightarrow B \rightarrow C$. Then: $I_f(A; B) \geq I_f(A; C)$.

Proof. To simplify analysis, we consider the case of having countable spaces and probability mass functions. All functions are assumed to be measurable. The general case over arbitrary measurable spaces hold by standard limiting arguments [50]. We write:

$$D_f(\mathbb{P}(A) \mid \mid \mathbb{P}(A|C)) = \sum_{a \in A} \mathbb{P}(A = a|C) \cdot f\left(\frac{\mathbb{P}(A = a)}{\mathbb{P}(A = a|C)}\right)$$

$$= \sum_{a \in A} \left(\mathbb{E}_{B|C=c} \mathbb{P}(A = a|B)\right) \cdot f\left(\frac{\mathbb{P}(A = a)}{\mathbb{E}_{B|C=c} \mathbb{P}(A = a|B)}\right)$$

However, when $f(u)$ is convex, then its conjugate $f^*(u) = uf(1/u)$ is also convex. So, we have by Jensen’s inequality:

$$D_f(\mathbb{P}(A) \mid \mid \mathbb{P}(A|C)) \leq \sum_{a \in A} \mathbb{E}_{B|C} \left[\mathbb{P}(A = a|B) \cdot f\left(\frac{\mathbb{P}(A = a)}{\mathbb{P}(A = a|B)}\right)\right]$$

$$= \mathbb{E}_{B|C} \left[D_f(\mathbb{P}(A|B) \mid \mid \mathbb{P}(A))\right]$$
Therefore:

\[
I_f(A;C) = \mathbb{E}_C \left[ D_f(P(A) \mid \mid P(A|C)) \right] \leq \mathbb{E}_{B,C} \left[ D_f(P(A) \mid \mid P(A|B)) \right] \\
= \mathbb{E}_B \left[ D_f(P(A) \mid \mid P(A|B)) \right] = I_f(A;B)
\]

\[\square\]

### A.2 Axiomatic Treatment

We introduce the following three axioms. These are similar to, but not exactly identical to, the original axioms used by I. Csiszár [52]. The proofs presented here are also simpler. Again, to simplify analysis, we consider the case of having countable sets endowed with probability mass functions. All functions are assumed to be measurable. We let \( \Omega \) denote the sample space and we consider the information gain that results from updating a prior belief \( P \) into a posterior belief \( Q \), where both \( P \) and \( Q \) are probability mass functions on \( \Omega \).

**Definition 16** (Permutation Axiom). The permutation axiom states that the divergence is represented by a sum of contributions of divergences of each elementary event:

\[
D_f(P||Q) = \sum_{\omega \in \Omega} g(P(\omega), Q(\omega)),
\]

for some function \( g : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R} \). We assume that \( g \) is second-order differentiable in its domain.

The permutation axiom states that the distance is a functional of the shape of the probability distributions \( P \) and \( Q \) alone, and that it is independent of the labels assumed by the sample space \( \Omega \). However, the distinction between elementary events \( \omega \in \Omega \) and arbitrary events, i.e. subsets of \( \Omega \), should not matter because such distinction is up to the observer. This brings us to the second axiom.
Definition 17 (Projection Axiom). The projection axiom states that if the posterior distribution $Q$ only says that an event has occurred in some subset $U \subset \Omega$, then the amount of information we gain is equivalent to the amount of information we would gain by treating $U$ as an elementary event.

More formally, let $c > 0$ be some constant, $U$ be some subset of the sample space $\Omega$, and suppose that $Q$ is a posterior distribution that satisfies:

$$Q(\omega') = cP(\omega'), \quad \forall \omega' \in U \subset \Omega$$

Then, we require that:

$$D_f(P||Q) = \sum_{\omega \in \Omega} g(P(\omega), Q(\omega)) = g(P(U), cP(U)) + \sum_{\omega \in \Omega \setminus U} g(P(\omega), Q(\omega))$$

Finally, we require that information be monotone. In other words, an observer cannot gain more information by making less observations. More formally:

Definition 18 (Monotonicity Axiom). If the prior belief is a mixture of two distributions $P = \alpha P_1 + (1 - \alpha) P_2$ and the posterior belief is also a mixture of two distributions $Q = \beta Q_1 + (1 - \beta) Q_2$, then we require that:

$$D_f(\alpha P_1 + (1 - \alpha) P_2 || \beta Q_1 + (1 - \beta) Q_2)$$

$$\leq \alpha \beta D(P_1||Q_1) + \alpha (1 - \beta) D(P_1||Q_2)$$

$$+ (1 - \alpha) \beta D(P_2||Q_1) + (1 - \alpha)(1 - \beta) D(P_2||Q_2)$$

Informally, the Monotonicity Axiom states that an observer who has more precise beliefs (both prior and posterior) will, on average, gain more information. Otherwise, an observer can gain information by manipulating its beliefs, superficially so to speak.
Theorem 9. The three axioms imply that:

\[ D_f(P \parallel Q) = \sum_{\omega \in \Omega} Q(\omega) f\left(\frac{P(\omega)}{Q(\omega)}\right), \]

for some convex function \( f : [0, \infty) \to [0, \infty) \).

Proof. From the Permutation Axiom, we write:

\[ D_f(P \parallel Q) = \sum_{\omega \in \Omega} g(P(\omega), Q(\omega)), \]

for some function \( g \). In particular, if the posterior distribution \( Q \) is a Kronecker delta distribution, i.e. we observe an elementary event \( \omega^* \in \Omega \), then:

\[ D_f(P \parallel \delta(\omega^*)) = \sum_{\omega \neq \omega^*} g(P(\omega), 0) + g(P(\omega^*), 1) \]  \hspace{1cm} (A.1)

Next, suppose that we only know that the event is in some subset \( E \subseteq \Omega \). Then, by the Projection Axiom and the previous equation:

\[ \sum_{\omega \in U} g(P(\omega), 0) + \sum_{\omega \in U} g\left( P(\omega), \frac{P(\omega)}{P(U)} \right) = \sum_{\omega \in U} g(P(\omega), 0) + g(P(U), 1) \]  \hspace{1cm} (A.2)

This implies that the following condition must hold:

\[ g(P(U), 1) = \sum_{\omega \in U} g\left( P(\omega), \frac{P(\omega)}{P(U)} \right) \]  \hspace{1cm} (A.3)

The above condition can be re-written as follows. We require that for any \( x > 0 \), any \( \lambda_i \geq 0 \) and \( \sum_i \lambda_i = 1 \), the following condition must hold:

\[ \sum_i g\left( \lambda_i x, \lambda_i \right) = g(x, 1) \]  \hspace{1cm} (A.4)
In particular, if \( \lambda_i = 1/n \), then:

\[
 n \cdot g(x/n, 1/n) = g(x, 1) \quad \Rightarrow \quad g(ux, u) = u \cdot g(x, 1)
\]

So, if we write \( ux = w \), we obtain:

\[
g(w, u) = u \cdot g\left(\frac{w}{u}, 1\right)
\]  \hspace{1cm} (A.5)

This tells us that:

\[
g(w, u) = u \cdot f\left(\frac{w}{u}\right),
\]  \hspace{1cm} (A.6)

for some function \( f(x) = g(x, 1) \). The only thing left is to prove that \( f \) must be a convex function. However, this follows from the third axiom. The Monotonicity Axiom is equivalent to the two conditions that \( D(P || Q) \) be convex on \( P \) and be convex on \( Q \). It can be immediately shown that both conditions imply that \( f(u) \) and its conjugate \( f^*(u) \) are convex in the domain \((0, 1)\). However, since \( f^*(u) = uf(u) \), convexity of \( f^* \) in \((0, 1)\) implies that \( f \) is convex in \((1, \infty)\). By the first axiom, \( f \) is the sum of smooth continuous functions. Hence, the fact that \( f \) is convex in \( \mathbb{R}^+ \setminus \{1\} \) implies that it is convex in \( \mathbb{R}^+ \). \qed
Appendix B

Optimization over an Ellipsoid

In this appendix, we prove a result that was used multiple times in this thesis. We would like to prove the following two identities:

\[
\begin{align*}
\arg \inf_{w \in \varepsilon(\mu, \Sigma)} z^T w &= \mu - \frac{1}{\sqrt{z^T \Sigma z}} \Sigma z \\
\arg \sup_{w \in \varepsilon(\mu, \Sigma)} z^T w &= \mu + \frac{1}{\sqrt{z^T \Sigma z}} \Sigma z
\end{align*}
\]  

(B.1)

We will denote by \( \varepsilon(\mu, \Sigma) \) the ellipsoid whose mean is \( \mu \) and whose covariance matrix is \( \Sigma \). First, suppose we want to minimize the linear function:

\[ f(w) = z^T w : \mathbb{R}^d \to \mathbb{R}, \]

for some fixed \( z \in \mathbb{R}^d \). Our optimization variable is \( w \in \varepsilon(\mu, \Sigma) \), where both \( \mu \) and \( \Sigma \) are fixed.

The optimization problem can be written as:

\[
\begin{align*}
\text{minimize} & \quad z^T w \\
\text{subject to} & \quad (w - \mu) \Sigma^{-1}(w - \mu) \leq 1
\end{align*}
\]

The Lagrangian is:

\[ L(w, \lambda) = z^T w + \lambda \cdot [(w - \mu) \Sigma^{-1}(w - \mu) - 1], \]
for some $\lambda > 0$. Because the optimization problem is convex and is always strictly feasible (i.e. Slater’s condition holds), we can solve it using the Karush-Kuhn-Tucker (KKT) optimality conditions \[123\]. First, we take the gradient with respect to $w$ and set it to zero:

$$ z + 2\lambda \Sigma^{-1}(w - \mu) = 0 $$

This implies that the optimal solution $w^*$ satisfies $w^* = \mu - c\Sigma z$ for some constant $c > 0$. Since we only need to determine $c$, we rewrite the first optimization problem as:

$$ \begin{align*}
\text{minimize} & \quad z^T \mu - cz^T \Sigma z \\
\text{subject to} & \quad c^2 (z^T \Sigma z) \leq 1
\end{align*} $$

Here, we plugged in the expression for $w^*$. Because $\Sigma \succeq 0$, the quadratic form $z^T \Sigma z$ is non-negative. Therefore, the above optimization problem has the optimal solution:

$$ c^* = \frac{1}{\sqrt{z^T \Sigma z}} $$

Plugging this into the earlier expression for $w^*$ proves that:

$$ \arg \inf_{w \in \epsilon(\mu, \Sigma)} z^T w = \mu - \frac{1}{\sqrt{z^T \Sigma z}} \Sigma z $$

The second identity can be proved in a similar manner.
Appendix C

Query Synthesis for Halfspaces: MATLAB Code

In this appendix, I provide a MATLAB implementation code for the query synthesis algorithm proposed in Chapter 3.

```matlab
function [ Xnxt, w_est ] = halfspace_query_synthesis( Xold, Yold, k )

% Active Learning of Halfspaces via Query Synthesis (Paper published at AAAI 2015), direct any questions to ibrahim.alabdulmohsin@kaust.edu.sa

% DESCRIPTION:
% This routine synthesizes near-optimal k queries to learn a halfspace
given the old membership queries.

% INPUT:
% k is the batch size (i.e. the number of new queries).
% Xold are the instances of the old queries. Each row of Xold corresponds to
% one instance.

% >>>>>> You MUST include at least one positive and one negative
% instance <<<<<
% Yold are the binary labels {−1, +1} that correspond to Xold

% OUTPUT:
% Xnxt is a (k x d) matrix of next queries
% w_est is the estimated coefficient vector for the halfspace (see
% the full paper for details)

% EXAMPLE:
% If we know that x1=(1,1) is in the halfspace, x2=(−1,−1) is outside the
% halfspace,
% and x3=(1,2) is in the halfspace, and we want to select two queries next then :

% Xold= [ 1 , 1 ;
% −1,−1 ;
% 1,2 ]
% Yold= [ 1 ;−1 ;1 ]
% k = 2

% To show the exponential decay, suppose w is fixed, and X contains one
positive and one negative instance.

% Then, we can estimate w using the following loop:

ERR=[]; k=5;
Y=sign(X*w);
for i=1:100,
    [Xnxt, w_est ] = select_next_queries(X, Y, k);
    X=[X; Xnxt];
    Y=sign(X*w);
    err = norm(w-w_est)
    ERR = [ERR err];
```
% DEPENDENCIES: You need to install CVX (http://cvxr.com/cvx/download/)

d=size(Xold,2);
m=size(Xold,1);

% solve the maximum ellipsoid problem
% it is assumed that at least one positive and one negative instance exist
cvx_begin quiet
try
    cvx_solver mosek
catch
    cvx_solver SDPT3
end
variable s(d)
variable u(d)

maximize ( geo_mean(s) )
subject to
    diag(Yold)*(Xold+u)>= norms(Xold.*(ones(m,1)*s'),2,2);
    norm(u)<=1

w_est = u;
S=diag(s); % the square root of the covariance matrix
N=null(u');
B = S*N; B=B'*B;
[alph,~] = eigs(B,k);
Xnxt = (N*alph)';

end
Appendix D

Query Synthesis for Non-Homogenous Halfspaces: MATLAB Code

In this appendix, I provide a MATLAB implementation code for the query synthesis algorithm for non-homogenous that is described in Chapter [5].

1 function [ Xnxt, w_est ] = non_homogenous_halfspace_query_synthesis( Xold, Yold, k )
2 %This algorithm is described in the PhD thesis of Alabdulmohsin (2017)
3 % direct any questions to ibrahim.alabdulmohsin@kaust.edu.sa
4 %
5 % DESCRIPTION:
6 % This routine synthesizes near-optimal k queries to learn a non-homogenous halfspace
7 % given the old membership queries.
8 % INPUT:
9 % k is the batch size (i.e. the number of new queries).
10 % Xold are the instances of the old queries. Each row of Xold corresponds to one instance.
11 % >>>>>> You MUST include at least one positive and one negative instance <<<<<
12 % Yold are the binary labels \{-1, +1\} that correspond to Xold
13 % OUTPUT:
14 % Xnxt is a (k x d) matrix of next queries. The last entry of X is always 1 (for the bias term in the non-homogenous case)
15 % w_est is the estimated coefficient vector for the halfspace (the last entry of w_est is the bias b)
16 % EXAMPLE:
17 % If we know that x1=(1,1) is in the halfspace, x2=(-1,-1) is outside the halfspace,
18 % and x3=(1,2) is in the halfspace, and we want to select two queries next then:
19 % Xold= [1,1, 1;
20 % -1,-1, 1;
21 % 1,2, 1]
22 % Yold= [1;-1;1]
23 % k = 2
24 % Here, we added a column of one’s for the bias term
25 % To show the exponential decay, suppose w is fixed, and X contains one positive and one negative instance.
26 % Then, we can estimate w using the following loop:
27 % ERR=[]; k=5;
28 % Y=sign(X*w);
for i = 1:100,
    [Xnxt, w_est] = select_next_queries(X,Y,k);
    X = [X; Xnxt];
    Y = sign(X*w);
    err = norm(w-w_est)
    ERR = [ERR err];
end
plot(ERR)

DEPENDENCIES: You need to install CVX (http://cvxr.com/cvx/download/)

d=size(Xold,2);
m=size(Xold,1);

solve the maximum ellipsoid problem
it is assumed that at least one positive and one negative instance exist

cvx_begin quiet
try
cvx_solver mosek
catch
cvx_solver SDPT3
end
variable s(d)
variable u(d)

maximize (geo_mean(s))
subject to
diag(Yold)*(Xold+u)>= norms(Xold.*(ones(m,1)*s'),2,2);
norm(u)<=1

svd = svd(X);
s1 = (B*B')
    s = B'*s1;

Bhat = [B; s'*S' 2];
N = null(Bhat);
J = S*N;
J = J'*J;
[beta,'] = eigs(J,k);
Xnxt = (N*beta)';
for i = 1:k
    Xnxt(i,:) = s'+Xnxt(i,:);
down
end
Appendix E

Learning a Distribution of Classifiers: MATLAB Code

In this appendix, I provide a MATLAB implementation code for the Semidefinite Programming (SDP) proposed in Chapter [6].

```matlab
function [ mu, s ] = wide_distribution_learn( Xtrn, Ytrn, nu, C, solver )
%This function learns a Gaussian distribution of non-homogenous linear classifiers
% INPUT:
% - Ytrn is the training vector of labels in {−1, +1}
% - Xtrn is the training matrix of instances in R^(m x d) (where d is the
% dimension of the data)
% - Both nu and C are parameters that influence accuracy. They are described in the
% paper (CIKM’14).
% - Solver is 1 if SDPT3, 2 if SEDUMI, and 3 if MOSEK (3 is recommended)
% OUTPUT:
% - mu is the mean of the distribution
% - s is a vector of the diagonal entries of the covariance matrix.
% - C If instances are in R^d, then both mu and s are in R^(d+1) to account
% - for the biasing term.

nu_c = norminv(nu);

% first step, add a constant component to X
n=size(Xtrn,2) + 1;
m=size(Xtrn,1);
Xtrn = [ones(m,1) Xtrn];

% begin the CVX solver

variable s(n)
variable mu(n)
variable xi(m)

minimize ( 0.5*quad_over_lin(mu, ones(1,n)*s) + C*ones(1,m)*xi)
subject to
    diag(Ytrn)*Xtrn*mu >= ones(m,1) + nu_c*(Xtrn.^2)*s - xi
```
39       x_i >= 0
40       s >= 0
41       cvx_end
42
43       end
Appendix F

The 1-Norm Support Vector Machine: MATLAB Code

In this appendix, I provide a MATLAB implementation code for the 1-norm Support Vector Machine (SVM) method proposed in Chapter 7.

```matlab
function [U, b] = one_norm_svm_train(Strn, Ytrn, C)
% This routine implements the 1-norm SVM using CVX with the Gurobi solver.
% If the Gurobi solver is unavailable, please comment out the line:
% "cvx_solver Gurobi"
% INPUT:
% - Strn is the similarity matrix.
% - Ytrn are the labels in {-1, +1}
% - C>0 is the tradeoff constant
% OUTPUT:
% - U are the multipliers (learned weights), which are denoted by lambda in the paper (ACML'15).
% - b is the offset (lambda_0)

mtrn = size(Strn, 1);
if mtrn ~= size(Strn, 2),
    disp('ERROR: The similarity matrix must be square');
    U = [];
    b = 0;
    return
end

% Form the Q matrix
Q = zeros(size(Strn));
for i = 1:mtrn,
    for j = 1:mtrn,
        if Ytrn(i) == Ytrn(j),
            Q(i, j) = Strn(i, j);
        else
            Q(i, j) = -Strn(i, j);
        end
    end
end

% Solve using CVX
cvx_begin
    cvx_solver Gurobi; % If Gurobi is unavailable, comment out this line.
    variable U(mtrn);
    variable E(mtrn);
    variable b;
    cvx_quiet(true);
end

U = U';
```
minimize (sum(U) + C*sum(E))
Q*U - b*Ytrn >= ones(mtrn,1) - E
U >= 0
E >= 0
cvx_end
end
Appendix G

Publication List

G.1 Journal and Conference Papers

1. I. Alabdulmohsin
   “An Information-Theoretic Route from Generalization in Expectation to Generalization in Probability”
in the *the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, Florida, April, 2017.

2. I. Alabdulmohsin, Y. Han, Y. Shen, X. Zhang
   “Content-Agnostic Malware Detection in Heterogeneous Malicious Distribution Graph”
in the *the 25th ACM International Conference on Information and Knowledge Management (CIKM)*, Indianapolis, October, 2016.

3. I. Alabdulmohsin, M. Cisse, X. Zhang,
   “Is Attribute-Based Zero-Shot Learning an Ill-Posed Strategy?”
in the *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD)*, Riva del Garda, Italy, September, 2016.

4. I. Alabdulmohsin, M. Cisse, X. Gao, X. Zhang,
   “Large Margin Classification with Indefinite Similarities”

5. I. Alabdulmohsin,
   “Algorithmic Stability and Uniform Generalization”

6. I. Alabdulmohsin, X. Gao, X. Zhang,
   “Efficient Active Learning of Halfspaces via Query Synthesis”
in *Association for the Advancement of Artificial Intelligence (AAAI)*, Austin, Texas, January, 2015.
7. I. Alabdulmohsin, X. Gao, X. Zhang,
“Support Vector Machines with Indefinite Kernels”

8. I. Alabdulmohsin, X. Gao, X. Zhang,
“Adding Robustness to Support Vector Machines Against Adversarial Reverse Engineering”
in *ACM International Conference on Information and Knowledge Management (CIKM)*, Shanghai, November, 2014.

9. I. Alabdulmohsin,
“Interference in Wireless Ad Hoc Networks with Smart Antennas”

10. I. Alabdulmohsin, A. Hyadi, L. Afify, B. Shihada,
“End-to-End Delay Analysis in Wireless Sensor Networks with Service Vacation”
in *IEEE Wireless Communications and Networking Conference (WCNC)*, Turkey, May, 2014.

### G.2 Preprints

1. Alabdulmohsin, I. M.,
“*Summability Calculus: A Comprehensive Theory of Fractional Finite Sums*”
in *arXiv:1209.5739 [math.CA]*, 2012
(currently under review by Springer).

2. Alabdulmohsin, I. M.,
“A new summability method for divergent series.”

3. Alabdulmohsin, I. M.,
“*Fractional Parts and their Relations to the Values of the Riemann Zeta Function*”