ROBUST LARGE MARGIN APPROACHES FOR MACHINE LEARNING IN ADVERSARIAL SETTINGS

by

MOHAMADALI TORKAMANI

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

September 2016

DISSERTATION APPROVAL PAGE

Student: MohamadAli Torkamani

Title: Robust Large Margin Approaches for Machine Learning in Adversarial Settings

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science by:

Daniel Lowd	Chair
Dejing Dou	Core Member
Christopher Wilson	Core Member
Hal Sadofsky	Institutional Representative

and

Scott L. Pratt	Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded September 2016

DISSERTATION ABSTRACT

MohamadAli Torkamani

Doctor of Philosophy

Department of Computer and Information Science

September 2016

Title: Robust Large Margin Approaches for Machine Learning in Adversarial Settings

Many agencies are now using machine learning algorithms to make high-stake decisions. Determining the right decision strongly relies on the correctness of the input data. This fact provides tempting incentives for criminals to try to deceive machine learning algorithms by manipulating the data that is fed to the algorithms. And yet, traditional machine learning algorithms are not designed to be safe when confronting unexpected inputs.

In this dissertation, we address the problem of adversarial machine learning; i.e., our goal is to build safe machine learning algorithms that are robust in the presence of noisy or adversarially manipulated data.

Adversarial machine learning will be more challenging when the desired output has a complex structure. In this dissertation, a significant focus is on adversarial machine learning for predicting structured outputs. First, we develop a new algorithm that reliably performs collective classification, which is a structured prediction problem. Our learning method is efficient and is formulated as a convex quadratic program. This technique secures the prediction algorithm in both the presence and the absence of an adversary.

Next, we investigate the problem of parameter learning for robust, structured prediction models. This method constructs regularization functions based on the limitations of the adversary. In this dissertation, we prove that robustness to adversarial manipulation of data is equivalent to some regularization for large-margin structured prediction, and vice versa.

An ordinary adversary regularly either does not have enough computational power to design the ultimate optimal attack, or it does not have sufficient information about the learner's model to do so. Therefore, it often tries to apply many random changes to the input in a hope of making a breakthrough. This fact implies that if we minimize the expected loss function under adversarial noise, we will obtain robustness against mediocre adversaries. Dropout training resembles such a noise injection scenario. We derive a regularization method for large-margin parameter learning based on the dropout framework. We extend dropout regularization to non-linear kernels in several different directions.

Empirical evaluations show that our techniques consistently outperform the baselines on different datasets.

This dissertation includes previously published and unpublished coauthored material.

CURRICULUM VITAE

NAME OF AUTHOR: MohamadAli Torkamani

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA

Isfahan University of Technology, Isfahan, Iran

DEGREES AWARDED:

Doctor of Philosophy, Computer and Information Science, 2016, University of Oregon

Master of Science, Artificial Intelligence, 2006, Isfahan University of Technology

AREAS OF SPECIAL INTEREST:

Machine learning, Statistics, Convex Optimization, Robust Modeling

PROFESSIONAL EXPERIENCE:

Graduate Research & Teaching Assistant, Department of Computer and Information Science, University of Oregon, 2011 to present

Research Intern, Clari, Mountain View, California, 2015

Research Intern, Comcast Labs, Washington, D.C., 2012

Research Assistant, Department of Electrical Engineering and Computer Science, Oregon State University, 2009 to 2011

GRANTS, AWARDS AND HONORS:

Graduate Teaching & Research Fellowship, Computer and Information Science, 2011 to present

PUBLICATIONS:

Torkamani, M., Lowd, D. (2013). Convex Adversarial Collective Classification. In *Proceedings of the 31th International Conference on Machine Learning (ICML 2014)*, Pages 642-650.

Torkamani, M., Lowd, D. (2014). On Robustness and Regularization of Structural Support Vector Machines. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, Pages 577-585.

Torkamani, M., Lowd, D. (2016). Marginalized and Kernelized Dropout Training for Support Vector Machines. Under review in *Journal of Machine Learning Research (JMLR).*

ACKNOWLEDGEMENTS

First and foremost I want to thank my advisor Daniel Lowd, who has given me every opportunity to pursue my ideas, and whose mentorship has shaped my development as a scientist. It has been an honor to be one of his first Ph.D. Students. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating.

I would like to thank my dissertation committee members Dejing Dou, Christopher Wilson, and Hal Sadofsky. I also would like to thank Andrzej Proskurowski and Jun Li for their constructive comments in the past years.

I gratefully acknowledge the funding sources, ARO grant and NSF grant that made my Ph.D. work possible.

Lastly, I would like to thank my friends and family for all their wholehearted support and encouragement. I want to thank my parents who raised me with love and taught me to love science. And most of all for my loving, supportive, encouraging, and patient wife Fereshteh, whose support during my Ph.D. is so appreciated.

To my wife, Fereshteh

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

"Things will go wrong in any given situation if you give them a chance."
– Edward A. Murphy

Smith's Law: *"Murphy was an optimist."*

Machine learning is widely used for prediction and decision-making, often taking the place of human agents. Reliability of machine learning algorithms is a rising concern in many sensitive applications, where the input data can be noisy and uncertain. The *uncertainty* and *noise* in the data used to be random most of the time, but now, the criminals have incentives to *adversarially* change the data. As tasks pertaining to detecting malicious activities are increasingly assigned to machine learning algorithms, criminals become increasingly motivated to put extra effort into deceiving these algorithms.

The traditional prediction models are vulnerable when confronting unexpected or maliciously manipulated data. This vulnerability is a serious problem for the applicability of this modern technology. The criminals are learning to tactfully disguise their actions. They strive to elaborately design innocent-looking fraudulent samples when attacking machine learning systems. As a result, since the classical machine learning techniques are not constructed with a safety mindset in the first place, their susceptibility to data manipulation makes them untrustworthy in many of the high-stake applications.

In this thesis, we present machine learning methods that are resilient and reliable. Our methods utilize domain knowledge and problem structures to deliver reliable predictions. This work has advanced the state-of-the-art in adversarial machine learning by introducing efficient algorithms for learning robust models when the output space is exponential in the input size. We show that by taking advantage of the weaknesses of the adversaries, we will be able to learn models that are particularly reliable when being attacked by those parties.

1.1. Motivation and approach

Conventional statistical methods – including machine learning – suppose that training and test instances are independently and identically drawn from the same distribution (the IID assumption)[1], which is frequently not true. Due to this fact, the traditional machine learning algorithms do not offer a realistic solution to many of the existing and emerging real-world problems, where there are fundamental reasons for the data samples to be interdependent or to be drawn from different distributions.

In fact, there are two common situations, in each of which the IID assumption does not hold. First, the train and the test data might have been drawn from two non-identical distributions. The difference in the distributions, at train time and at test time, can derive from: constant changes in the underlying data generation sources; random noise; subjective conceptual drift, e.g., change of topic in a discussion forum; or, a pensive agent might have intentionally manipulated some parts of the data. Spiteful manipulation of the data practically serves some

[1]If the data samples are independently and identically drawn from a distribution, then we say the samples are *IID*. The mathematical modeling of the distribution of IID samples is simpler and cleaner. Although the IID assumption *rarely* holds in practice, many statistical approaches, including classical machine learning, still suppose that it is satisfied.

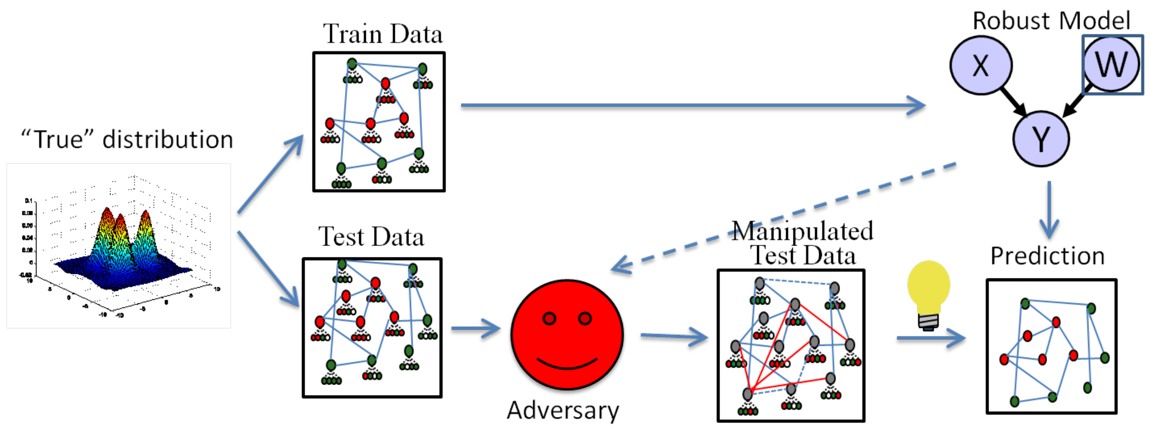


FIGURE 1.1. The adversary manipulates the unseen data as a response to the learner's strategy. A robust model decreases the harmful effect of the adversarial data alteration.

interests of the adversaries. To satisfy their interests, the adversaries design specific samples such that some *utility functions* are maximized.

The learner usually has little or no knowledge of the details of these utility functions. However, the adversary has either full information or a partial guess of the learner's strategies or the parameters of its decision-making algorithm. The adversary may increase its knowledge about the learner's underlying model by submitting query examples and studying the responses of the machine learning system. Potentially, the adversary will be able to acquire a near-perfect approximation of the internal functionality of the learner's prediction system.

Interdependent data samples are the second cause of violation of the IID assumption. The dependency of data samples can have different forms. For example, the sentences in a paragraph of an English text are not statistically independent. And in graphed data, wherein each vertex has a label, the label of each node may depend on the labels of the neighboring nodes. A particularly important type of dependency is when the desired output of the algorithm has some internal structure; examples of such outputs are the parse tree of a sentence, labeling of the nodes in a graph, and segments of an image.

To date, most of the modern methods in machine learning are designed to solve only one of these two challenges; i.e., *either* they approach the problem of inter-related data, *or* they develop robust algorithms against noise and natural or adversarial changes in the distribution of the data samples. In this thesis, we introduce novel methods in machine learning where *both* of the IID assumptions are violated: The samples are not independent, *and* they are not drawn from a static distribution at train and test time. In particular, we focus on the worst-case scenario, where the unseen data in the future will be intentionally manipulated by some adversary to deceive the machine learning algorithm (Figure 1.1).

We introduce a direct but efficient robust modeling approach for solving the problem of label prediction on graphs, where some opponent changes the properties of each node to misguide the labeling algorithm as much as possible. We will consider the conditions under which the efficiency of this algorithm is guaranteed. Then, we propose a regularization-based approach, which creates customized optimization programs to adopt the weaknesses of the adversaries and converts them to the points of strength of the machine learning algorithm. This is done by learning robust models that take the ultimate advantage of how the adversary budget allows joint changing of a set of values in the input data.

1.2. Learning and prediction under uncertainty

Learning classifiers in the presence of noisy and uncertain instances is a challenging and important task in modern machine learning. Noise in the data may refer to the observations that are added or multiplied by unknown random values, that have missing attributes, or that have inaccurate labels. Many of the real-world data, such as texts, gene expression data, or images and videos are naturally noisy. The noise can variously derive from, e.g., human error in data collection, data processing, and/or data tagging; measurement errors; and/or sub-optimal sampling resolutions. However, the existence of adversarial uncertainty in the data is a more severe issue. Given the prospect of cyber-crimes in this century, it is an important and more challenging task to learn models that are not only robust to random noise, but are also robust to the worst-case adversarial ones. Therefore, developing algorithms that are robust to the uncertainty caused by adversaries is of growing interest (Kloft and Laskov, 2007).

When the instances are noisy, in most of the cases, there exists little or no knowledge about the level of uncertainty in the data. In adversarial scenarios, the adversary usually aims for maximizing a utility function, while having some *budget* constraints for changing individual sets of features. Therefore, as the learner, we do not know whether the observed information is what it initially used to be, or if the adversary has changed it according to some underlying set of constraints and utilities.

The adversaries actively change their strategies: As the learner blocks them on one front, they seek to find another vulnerability of the machine learning system. This problem can be formulated as a game between the learner and the adversary: Each side will be rewarded when it chooses the right strategies.

One of the earliest works in adversarial machine learning was reverse engineering classifiers (Lowd and Meek, 2005b,a; Nelson et al., 2010). The idea is to find optimal attacks as a response to the specific model that the machine learning algorithm has learned. Then, the machine learning algorithm can adjust itself to be able to correctly classify the optimal attack. This ends up in a race between the two players of an antagonistic game: the learner and the adversary.

In general, finding the Nash equilibrium for this game is intractable. Dalvi et al. (2004) suggest that instead of finding a Nash equilibrium, we can select a strategy for the next move of the adversary. Brückner and Scheffer derive an optimization approach for finding the Nash equilibrium in static prediction games under certain convexity assumptions (Brückner and Scheffer, 2009; Brückner et al., 2012). They also propose a formulation for approximating the Stackelberg equilibria (Brückner and Scheffer, 2011; Sawade et al., 2013).

Assuming that an adversarial game is zero-sum leads to a min-max formulation: The learner tries to minimize a worst-case loss function under the adversarial manipulation of the input data. Globerson and Roweis (2006) modeled this data manipulation by feature deletion at test time. A generalized version of this method was later proposed by Teo et al. (2008).

Xu et al. (2009) show that penalizing the optimization program by the dual norm of the adversarial constraint is equivalent to optimizing against a worst-case adversary that can manipulate features within that constraining ball.

Developing secure algorithms that are not mistrained by poisoned data is a different view of adversarial machine learning. Data poisoning refers to engineering samples that are adversarially crafted to mislead a specific machine learning algorithm (Kloft and Laskov, 2007; Laskov and Kloft, 2009; Laskov and Lippmann, 2010; Biggio et al., 2012).

The dropout technique is another method that was originally introduced for stabilizing the behavior of deep neural networks in the presence of noise in the unseen data: During the training phase, some attributes of the data are randomly dropped out while learning the parameters (Srivastava et al., 2014). In shallow models, such as logistic regression (LR), dropout behaves as a regularizer that penalizes feature weights based on how much they influence the classifier's predictions (Wager et al., 2013). Since, in adversarial machine learning, robustness is often equivalent to regularization through the right penalty function, we expect to gain robustness by deriving regularization methods that emulate the effect of dropout training.

On the other hand, in many real-world scenarios, the machine learning algorithm does not need to be robust to the worst-case adversary. Instead, it suffices to learn the model such that it is reliable when encountering an average opponent that might change the input data frequently, yet randomly in order to deceive the algorithm. This fundamental idea, suggests that if we minimize the expected loss function under adversarial noise, we will gain some robustness against average adversaries. Dropout training simulates such an adversarial behavior. In this dissertation, we derive a closed-form formulation for the expected hinge loss. Our formulation is convex, and can be optimized efficiently.

In this thesis, we further expand some of the algorithms mentioned above to perform robust prediction of complex outputs. We will show how we can gain robustness by designing the appropriate regularization functions. We induce the regularization functions from a worst-case uncertainty set, or we derive them from the implicit marginalization effect of applying the dropout framework.

1.3. Predicting complex outputs

Structured learning is the problem of finding a predictive model for mapping the input data into complex outputs that have some internal structure. Structured output prediction is a challenging task by itself, but the problem becomes even more troublesome when the input data is adversarially manipulated to deceive the predictive model. The problem of adversarial structured output prediction is relatively new in the field of machine learning.

We can abstract many real-world applications as an adversarial structured output prediction problem. A motivating example of adversarial structured prediction is *collective classification* of interconnected and potentially dishonest nodes of a network. In a collective classification problem (Sen et al., 2008), the goal is to label a set of interconnected objects simultaneously, using both their attributes and their relationships. For example, linked web pages are likely to have related topics; friends in a social network are likely to have similar demographics; and proteins that interact with each other are likely to have similar locations and related functions. Probabilistic graphical models, such as Markov networks (Taskar et al., 2004a; Koller et al., 2003), and their relational extensions, such as Markov logic networks (Domingos and Lowd, 2009b), can handle both uncertainty and complex relationships in a single model, making them well-suited to collective classification problems (Torkamani and Lowd, 2013).

Many collective classification models are evaluated on test data that is drawn from a different distribution than the training data. This can be a matter of concept drift, such as varying topics in interconnected news web pages at different times, or the change in the distribution can be attributed to one or more adversaries who are actively modifying their behavior to avoid detection. For example, when the search engines began to use incoming links to rank web pages, spammers began posting comments on unrelated blogs or message boards, with links back to their websites. Since incoming links are used as an indication of the quality of the web page, manufacturing of the incoming links makes a spammy website appear more legitimate. Web spam (Abernethy et al., 2010; Drost and Scheffer, 2005) is one of many examples with explicitly adversarial domains; some other examples are counter-terrorism, online auction fraud (Chau et al., 2006), and spam in online social networks.

One important aspect of adversarial machine learning that is currently missing in the literature of adversarial structured prediction is a deep analysis of the vulnerability of structured output prediction methods to exploratory evasion attacks. In particular, in the existing studies, the assumption is that the adversary is completely aware of the classifier and the learned parameters of the classifier; but this assumption will not hold in practice, in general. In real problems, such as a web spam detector in a search engine, the parameters of the classifier are unknown for the spammers, and the spammers need to infer them by exploration techniques.

In this thesis, we address the problem of adversarial structured prediction and propose efficient algorithms for learning and prediction of structured outputs in adversarial settings.

1.4. Contributions

In this thesis, we propose novel methods for constructing large margin classifiers, which are robust to uncertainties and have a better generalization on the future data. Tractability of the robust learning algorithms is a central theme in this dissertation. We attack the hard problem of adversarial stuctured prediction. We prove that robustness can be achieved by penalizing the problem by a customized regularization function. Then, we show that the dropout framework also results in a regularization effect in the large margin classifiers, which leads to a better generalization of the predictive model. The following are the highlights of our contributions:

1. **Convex adversarial collective classification**

   We present a novel method for robustly performing collective classification in the presence of a malicious adversary that can modify up to a fixed number of binary-valued attributes. Our method is formulated as a convex quadratic program that guarantees optimal weights against a worst-case adversary in polynomial time. In addition to increased robustness against active adversaries, this kind of adversarial regularization can also lead to improved generalization, even when no adversary is present. In experiments on real and simulated data, our method consistently outperforms both non-adversarial and non-relational baselines.

2. **Equivalency of adversarial robustness and regularization**

   Previous analysis of binary SVMs has demonstrated a deep connection between robustness to perturbations over uncertainty sets and regularization of the weights. We explore the problem of learning robust models for structured prediction problems. We first formulate the problem of learning robust structural SVMs when there are perturbations in the feature space. We consider two different classes of uncertainty sets for the perturbations: ellipsoidal uncertainty sets and polyhedral uncertainty sets. In both cases, we show that the robust optimization problem is equivalent to the non-robust formulation with an additional regularizer. For the ellipsoidal uncertainty set, the additional regularizer is based on the dual norm of the norm that constrains the ellipsoidal uncertainty. For the polyhedral uncertainty set, we show that the robust optimization problem is equivalent to adding a linear regularizer in a transformed weight space related to the linear constraints of the polyhedron. We also show that the constraint sets can be combined, and we demonstrate some interesting special cases. This represents the first theoretical analysis of robust optimization of structural support vector machines. Our experimental results show that our method outperforms the non-robust structural SVMs on real-world data, when the test data distributions are drifted from the training data distribution.

3. **Robustness of large margin methods through dropout regularization**

   Dropout training is a regularization technique that consists of setting randomly selected input features or hidden units to zero for each training example. Dropout training was originally proposed for deep neural networks, but even shallow models, such as logistic regression, can benefit from training with this kind of noise. In this thesis, we analyze dropout training in support vector machines (SVMs). First, we derive a convex, closed-form objective for linear SVMs that marginalizes over all possible dropout noise. Our objective is simple, efficient to optimize, and closely approximates the exact marginalization. For SVMs with non-linear kernels, we define dropout over input space, feature space, and input dimensions. We introduce methods for approximate marginalization over feature space dropout, even when the feature space is infinite-dimensional, and Monte-Carlo methods for input space and dimension dropout. We introduce two methods for approximating dropout on the kernel feature map. The first uses a Fourier basis to approximate a high-dimensional kernel with a finite feature map and then applies our linear SVM dropout marginalization technique to the transformed representation. The second approximately marginalizes over dropout noise in the dual representation. In experiments on several text datasets, our marginalized objective is more accurate than standard linear SVM training. On several text datasets, our marginalized objective in the primal form is more accurate than standard linear SVM training. On MNIST and census data, both marginalized kernel dropout methods outperform the standard RBF kernel. We also introduce a novel dimension dropout method and show that it is more accurate than the standard RBF kernel on MNIST, especially when the training sizes are smaller.

1.5. Thesis outline

The following is the summary of the dissertation's chapters:

Chapter 2. Background: First, we review the basic concepts of statistical machine learning and structured prediction methods. Then, we focus on the high-level explanation of the adversarial machine learning algorithms. We introduce a general framework that abstracts most of the adversarial scenarios as a generic multi-agent game. The adversary's counteractive effects on the learning and prediction algorithms cause the learned model perform poorly in the future. To be robust to unpredictable effects, we should know the capabilities of the adversaries. We define a theoretical model for the adversary and categorize the properties of the adversary based on different criteria.

Chapter 3. Convex adversarial collective classification: In this chapter, we start by formulating the problem of adversarial collective classification as a bi-level minimax optimization program. We show that under certain interconnectivity conditions of the data graph, the solution of the lower-level optimization program is guaranteed to be integral after relaxation. Then, we introduce an equivalent quadratic optimization program that can be efficiently solved. We run experiments on the various datasets, and we show that our method always outperforms the baselines. This chapter is co-authored with my advisor Dr. Daniel Lowd and is published in the thirtyth proceedings of international conference on machine learning (Torkamani and Lowd, 2013).

Chapter 4. Equivalency of adversarial robustness and regularization: We focus on learning robust models for generic structured prediction problems. We discuss the different classes of uncertainty in the feature space: ellipsoidal and polyhedral. Then, we derive the robust optimization problem for each of these uncertainty sets. We show how the non-robust formulations become equivalent to the robust ones by adding a customized regularizer to their objective functions. We show how the customized regularization function should be derived from each specific uncertainty set, and we study several special cases of such sets. Finally, we derive a regularizer for combined ellipsoidal and polyhedral uncertainty sets. This chapter is co-authored with my advisor Dr. Daniel Lowd and is published in the thirty-first proceedings of international conference on machine learning (Torkamani and Lowd, 2014).

Chapter 5. Marginalization and kernelization of dropout for support vector machines: We study dropout training for support vector machines. We derive a closed-form objective function for linear SVMs. This objective is the result of marginalizing over the continuum of possible dropped out noisy samples. We also discuss the possibility of applying dropout to SVMs with non-linear kernels. We define the concept of applying dropout in input space, feature space, and input dimensions, and we introduce several methods for approximating the marginalization effect of dropout on kernel SVMs. The experimental results on several datasets, such as text and image classification, show that our methods are more accurate than the standard support vector machines. This chapter is co-authored with my advisor Dr. Daniel Lowd and is under review in the Journal of Machine Learning Research (JMLR).

Chapter 6. Conclusion and future directions: We summarize our contributions. We also discuss the future research directions and how the proposed methods in this thesis can be extended.

CHAPTER II

BACKGROUND

In this chapter, we review the basic concepts of adversarial machine learning. Our focus is the on methods that also apply to structured prediction problems. The chapter concludes with examples of real-world problems that are adversarial, and the output space is structured.

2.1. Statistical learning

In machine learning, output prediction is the procedure of observing the state $\mathbf{x}$ of some phenomenon (input) and using our understanding of the concept (learned model) to predict some hidden property $\mathbf{y}$ of the observed data (output). In this section, we briefly address the fundamentals of statistical machine learning.

2.1.1. Supervised learning

In supervised learning, the learner has access to samples that contain both the attributes' vectors and their corresponding labels. The training data samples $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\} \in (\mathcal{X} \times \mathcal{Y})^N$, are input-output pairs from the past. We assume that each sample $(\mathbf{x}_i, \mathbf{y}_i)$ is drawn from an underlying joint distribution over inputs and outputs: $P(\mathcal{X}, \mathcal{Y})$. Traditionally in machine learning, the researchers usually assume that $\mathbf{y}_i$ is the correct label for the input $\mathbf{x}_i$.

The goal is to find a mapping function (also known as a hypothesis function) $h \in \mathcal{H} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{H}$ is the space of relevant hypotheses, and $\mathcal{X}$ and $\mathcal{Y}$ are the set of possible inputs and outputs, respectively. Given $\mathbf{x} \in \mathcal{X}$, the predicted output is $\hat{\mathbf{y}} = h(\mathbf{x}) \in \mathcal{Y}$.

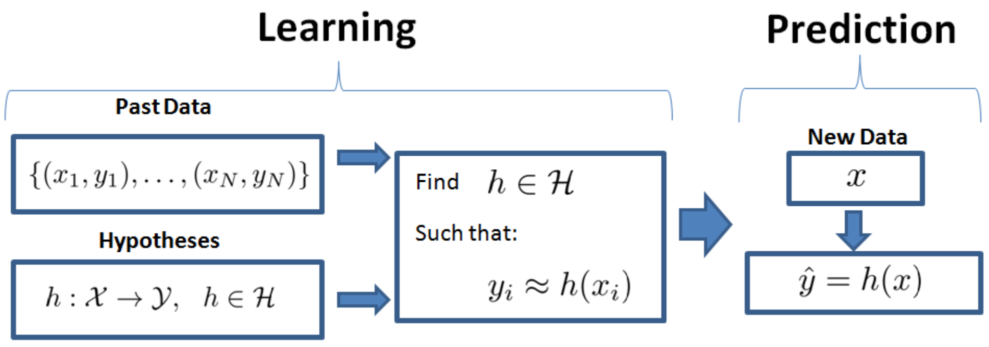


FIGURE 2.1. The supervised learning procedure

If $\mathcal{Y} = \mathrm{R}^m$ (m-constant), then the problem is called regression; if $|\mathcal{Y}| = 2$ (e.g. $\mathcal{Y} = \{0, 1\}$), then the prediction is called binary classification; if $\mathcal{Y}$ is a discrete set, and $|\mathcal{X}| \gg |\mathcal{Y}| > 2$, then the problem is called multi-class classification. If $|\mathcal{Y}|$ is extremely large and each member $\mathcal{Y}$ has some internal structure, then the problem is called "structured prediction".

The mapping function h should produce accurate predictions; i.e., for an input $\mathbf{x}_i$, the predicted output $\hat{\mathbf{y}}_i = h(\mathbf{x}_i)$ should be "close" to the true output $\mathbf{y}_i$. This closeness is usually defined by some non-negative loss function $l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathrm{R}$ that determines the distance of $\hat{\mathbf{y}}$ to $\mathbf{y}$. Sometimes the loss function $l(y', y)$ is not convex; and therefore, the optimization problem in Equation 2.5 is not tractable. Then, a convex surrogate function for $l(y', y)$ is used instead. We are interested in the hypothesis h that generalizes well to the unseen samples of the joint distribution over inputs and outputs. From a statistical point of view, we would like to find $h^* \in \mathcal{H}$, such that the expected loss is minimized:

$$h^* = \underset{h \in \mathcal{H}}{\arg\min} \; \mathrm{E}_{(\mathbf{x},\mathbf{y}) \sim P(\mathcal{X},\mathcal{Y})} \left[l \left(h(\mathbf{x}), \mathbf{y} \right) \right] \tag{Equation 2.1}$$

In real world problems, we don't have access to the whole population, or equivalently, we don't know $P(\mathcal{X}, \mathcal{Y})$; therefore, the empirical population (observed samples from the past) is used, instead:

$$\begin{aligned} h^* &= \underset{h \in \mathcal{H}}{\arg\min} \; \mathrm{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}} \left[l \left(h(\mathbf{x}), \mathbf{y} \right) \right] \\ &= \underset{h \in \mathcal{H}}{\arg\min} \; \frac{1}{N} \sum_{i=1}^{N} \; l \left(h(\mathbf{x}_i), \mathbf{y}_i \right) \end{aligned} \tag{Equation 2.2}$$

The term $\frac{1}{N} \sum_{i=1}^{N} \; l \left(h(\mathbf{x}_i), \mathbf{y}_i \right)$ is called the empirical risk. Figure 2.1 shows the procedure of supervised learning.

2.1.2. Generalized linear models

Flexibility of h mostly depends on the function space $\mathcal{H}$. We assume that h is parameterized by a parameter vector $\mathbf{w}$. In a general form, the hypothesis h can be a search pocedure that finds the best output. We can assume that the best output $\mathbf{y}$ maximizes some score function $s(\mathbf{x}, \mathbf{y}; \mathbf{w})$, then h can be formally defined as:

$$h(\mathbf{x}; \mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}}{\arg\max} \; s(\mathbf{x}, \mathbf{y}; \mathbf{w}) \tag{Equation 2.3}$$

In this thesis, we suppose that the scoring function is linear in the parameters $\mathbf{w}$:

$$s(\mathbf{x}, \mathbf{y}; \mathbf{w}) \;=\; \sum_{j=1}^{m} w_j f_j(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{f}(\mathbf{x}, \mathbf{y}) \qquad \text{(Equation 2.4)}$$

where $f_j(\mathbf{x}, \mathbf{y})$ is an arbitrary function of values from the input and the output space and is called a feature function. We refer to this parameterization of the hypothesis function as a generalized linear model (GLM).

For some problems, such as when $|\mathcal{Y}| = 2$, $\arg\max_{\mathbf{y} \in \mathcal{Y}} \; s(\mathbf{x}, \mathbf{y}; \mathbf{w})$ can be calculated in closed-form; then, we will have an explicit form for the hypothesis h.

2.1.3. Regularization

If the number of observed samples $|\mathcal{D}|$ is small, or if the number of possible hypotheses $|\mathcal{H}|$ is extremely large, then the learned hypothesis h^* in Equation 2.2 is likely to "overfit" the training data; i.e., we will achieve zero (or very small) empirical loss, but large errors on output prediction for unseen (test) data. We usually can not increase the number of training data, but we can control the "flexibility" of the hypothesis h to prevent it from overfitting to the training data. This task is performed by "regularizing" the hypothesis h. Regularization is done by minimizing a linear combination of the empirical risk and a penalty function $\Omega_{\mathcal{H}}(h)$ that controls the flexibility of h:

$$h^* \;=\; \arg\min_{h \in \mathcal{H}} \; \lambda \Omega_{\mathcal{H}}(h) + \frac{1}{N} \sum_{i=1}^{N} \; l\left(h(x_i), y_i\right) \qquad \text{(Equation 2.5)}$$

This approach is called regularized risk minimization. The coefficient of the regularization term λ is used to create a balance between the amount of penalization of the model parameters and the empirical risk minimization.

We can interpret the regularized risk minimization as an *a posteriori* probabilistic parameter learning method. The regularizer can be seen as the log of the prior distribution over the parameters, while its partition function does not depend on the parameters and can be removed from the objective of the optimization program (Bishop, 2006).

Choosing the right regularization function is crucial in gaining the desirable generalization effect. For example, in GLMs, if we have prior knowledge that the weights are IID and are drawn from a Gaussian distribution, then we set $\Omega_w(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$. This assumption is somewhat common because the squared L_2 norm is continuous, its derivative is simple, and it can be very efficiently optimized. If we expect the weight vector $\mathbf{w}$ to be sparse, then we can implicitly assume that it is drawn from a Laplacian distribution or equivalently set the regularization function to the L_1 norm: $\Omega_w(\mathbf{w}) = \sum_{j=1}^{m} |w_j|$

Clearly, such naïve assumptions are not necessarily optimal choices. Some of the main contributions of this thesis are centered around recipes for deriving effective regularization functions.

2.2. Structured learning

The traditional machine learning algorithms are designed to solve prediction problems whose outputs are a fixed number of binary or real-valued variables[1].

[1]In these prediction algorithms, the desired output must be representable as a K-dimensional vector, where K is a constant (e.g. $K = 1$ for scalars). For example, for a desired output $y \in \{c_1, \dots, c_K\}$, the common practice is to use a different representation for the output y. In

In contrast, there are problems with a strong interdependence among the output variables, often with sequential, graphical, or combinatorial structures. These problems involve prediction of complex outputs, where the output has some structure such as trees and graphs; these kinds of outputs are called structured outputs. Problems of this kind arise in security, computer vision, natural language processing, robotics, and computational biology, among many others.

Structured prediction (Bakir et al., 2007) provides a unified treatment for dealing with structured outputs. The structured prediction algorithms root back in a few seminal works: McCallum et al. (2000); Lafferty et al. (2001); Punyakanok and Roth (2001); Collins (2002); Koller et al. (2003); Altun et al. (2003); McAllester et al. (2004); Tsochantaridis et al. (2006), among others.

In this section, we explain the basics of structured prediction methods. We start with a brief explanation of the basics of supervised learning for structured prediction, and then we present some of the most practiced training algorithms for training structured predictors.

2.2.1. Motivation of using structured prediction

Before the emergence of the structured prediction algorithms, probabilistic graphical models (PGMs) (Pearl, 1988) were the most successful methods for solving problems with strongly interdependent outputs. By combining statistical learning and graph theory, PGMs provide a framework for making an inference about dependent variables and confounding factors. The basic idea behind PGMs is that the probability distribution function of the variables in the model can be factorized based on the graph of the direct dependencies among the variables.

this case, y will be represented as a K-dimensional binary vector $\mathbf{y}'$, where $\mathbf{y}'_i = 1$ if $y = c_i$, and is zero otherwise.

Although PGMs apply to many problems, they are overly general purpose, which is inevitably costly. Using the probability distribution function of variables in the model is desirable in theory, but estimating the parameters of the distribution functions – especially the normalization constants (a.k.a. partition functions), can be intractable. Structured prediction algorithms do not calculate the probability distribution of the variables explicitly, and mainly avoid the calculation of the normalization constants. Therefore, learning the parameters of structured prediction models is usually tractable, especially when tailored to specific problems.

The principal theme in all structured output prediction problems is the combinatorial nature of the labels. In particular, the number of possible outputs in such problems is exponential in the input size. This fact makes these problems distinctive from the classic problems that classical machine learning algorithms have been trying to solve. Therefore, new algorithms are needed for handling such problems.

2.2.2. Scoring function

A key concept in the state-of-the-art structured prediction algorithms is the notion of extended feature function in a GLM setting. The inputs of the feature functions are both the original input $\mathbf{x} \in \mathcal{X}$ and a hypothesized output $\tilde{\mathbf{y}} \in \mathcal{Y}$. We define $\mathbf{f}(\mathbf{x}, \mathbf{y})$ as the feature vector. The mathematical details of $\mathbf{f}(\mathbf{x}, \mathbf{y})$ are problem-specific. For example, in graphical models (Lauritzen, 1996), the feature function is the same as the vector of all potential functions (Bilmes et al., 2001; Torkamani and Lowd, 2013; Taskar et al., 2004a), and in maximum entropy (MaxEnt) models (Theil and Fiebig, 1984), or equivalently in log-linear models, the sufficient statistics are used as the feature functions.

In general, the choice of $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is a model selection problem. A specific example is collective classification of inter-connected documents (such as web pages) as "spam" and "non-spam". Let E be the set of the edges between the documents, where $e_{ik} = 1$ means that there is an edge from node i to node k and is zero otherwise. Also, let x_{ij} be the indicator variable that represents if the jth word is present in the ith document; for example if "v!agr@" has index 700 in the dictionary, then $x_{200,700} = 1$ means that the word "v!agr@" is present in the 200th document, and $x_{200,700} = 0$ means it is not present. Also let $y_i \in$ {"spam", "non-spam"} be encoded as the pair (y_{i1}, y_{i2}), where $(y_{i1}, y_{i2}) = (1, 0)$ means $y_i =$ "spam" , and $(y_{i1}, y_{i2}) = (0, 1)$ means $y_i =$ "non-spam". Now we can define a simple feature function:

$$f_{jk}(\mathbf{x}, \mathbf{y}) = \sum_i x_{ij} y_{ik} \qquad \text{(Equation 2.6)}$$

$$f_{ekk'}(\mathbf{x}, \mathbf{y}) = \sum_{i,j} e_{ij} y_{ik} y_{ik'} \qquad \text{(Equation 2.7)}$$

The feature function $\mathbf{f}(\mathbf{x}, \mathbf{y})$ now will be built by stacking all $f_{jk}(\mathbf{x}, \mathbf{y})$'s and $f_{ekk'}(\mathbf{x}, \mathbf{y})$'s in one vector. The feature function $\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}})$, with true values of $\mathbf{x}$ and a hypothetical output $\tilde{\mathbf{y}}$ is used as the higher level input to the mathematical model that describes the relevance of output structure $\tilde{\mathbf{y}}$. In particular, a linear combination of individual elements in $\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}})$ is used as the criterion for relevance of the hypothetical output $\tilde{\mathbf{y}}$ to the true $\mathbf{y}$, and is called the scoring function. Formally, the scoring function is defined in the following form:

$$score(\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{w}) = \mathbf{w}^T \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) \qquad \text{(Equation 2.8)}$$

$\mathbf{w}$ is called the model weight vector, and the goal of the machine learning algorithm is to learn such that the true labeling $\mathbf{y}$ gains the maximum score when plugged into the score function. Unfortunately, it is possible that in some cases an alternate labeling $\tilde{\mathbf{y}}$, which is very different than $\mathbf{y}$ also gains a high score. Therefore, the learning algorithm needs to select a $\mathbf{w}$ that penalizes such scenarios. We want to learn $\mathbf{w}$ such that the closer $\tilde{\mathbf{y}}$ is to $\mathbf{y}$, the higher the score of $\tilde{\mathbf{y}}$ is. Therefore $\Delta(\tilde{\mathbf{y}}, \mathbf{y})$ is defined as a measure of dissimilarity between $\tilde{\mathbf{y}}$ and $\mathbf{y}$. The Hamming distance between $\tilde{\mathbf{y}}$ and $\mathbf{y}$ is one of the popular choices. The difference function $\Delta(\tilde{\mathbf{y}}, \mathbf{y})$ plays an important role in many of the weight learning algorithms for structured output prediction.

In structured output prediction algorithms, a crucial problem is the hardness of searching different applicable $\tilde{y} \in \mathcal{Y}$ that maximizes the scoring function. In particular, after learning a weight vector $\mathbf{w}$, one will need to find the best output for a given input. This is the "argmax problem" defined in Equation 2.9 and referred to as maximum *a posteriori* (MAP) inference:

$$\begin{aligned} \hat{\mathbf{y}}_{prediction} &= h_{\mathbf{w}}(\mathbf{x}) \\ &= \arg\max_{\tilde{\mathbf{y}} \in \mathcal{Y}} \mathbf{w}^T \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) \end{aligned} \qquad \text{(Equation 2.9)}$$

This problem is not tractable in the general case. However, for specific $\mathcal{Y}$ and $\mathbf{f}(\mathbf{x}, \mathbf{y})$, one can use methods such as dynamic programming algorithms or integer programming algorithms to efficiently find solutions. In particular, if $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes over the vector representation of y, such that no feature depends on other features that have the same elements of y, then the problem is efficiently solvable.

2.2.3. Structured Prediction Methods

In this part, we briefly explain some of the primary methods for weight learning in structured prediction methods.

2.2.3.1. Structured Perceptron

The structured perceptron is an extension of the standard perceptron (Lippmann, 1987) to structured prediction (Collins, 2002; Collins and Duffy, 2002; McDonald et al., 2010). The algorithm of learning $\mathbf{w}$ is shown in Algorithm 1.

Algorithm 1 AveragedStructuredPerceptron($(x_1, y_1), \ldots, (x_N, y_N), maxIter$)

```
$\mathbf{w} \leftarrow [0, \ldots, 0]^T$
$c \leftarrow 1$
for $l = 1$ to $maxIter$ do
    for $i = 1$ to $N$ do
        $\hat{\mathbf{y}}_i = \arg\max_{\tilde{\mathbf{y}} \in \mathcal{Y}} \mathbf{w}^T \mathbf{f}(\mathbf{x_i}, \tilde{\mathbf{y}})$
        if $\hat{\mathbf{y}}_i \neq \mathbf{y}_i$ then
            $\mathbf{w} \leftarrow (1 - \theta_l)\mathbf{w} + \theta_l \alpha \left(\mathbf{f}(\mathbf{x_i}, \mathbf{y_i}) - \mathbf{f}(\mathbf{x_i}, \hat{\mathbf{y}}_\mathbf{i})\right)$
        end if
    end for
end for
return $\mathbf{w}$
```

In Algorithm 1, θ_l is a real number between 0 and 1 that determines the weight of the current update relative to previous weight in the lth iteration. In a simple averaging algorithm, we can set $\theta_l = \frac{1}{i}$. α as the learning rate. The algorithm applies an update to the weight whenever the output of $\arg\max_{\tilde{\mathbf{y}} \in \mathcal{Y}} \mathbf{w}^T \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}})$ is not equal to the true $\mathbf{y}$. Note that the algorithm is only applicable when the resulting output is either exactly equal to the true one, or it is completely different. In other words, the difference function $\Delta(\tilde{\mathbf{y}}, \mathbf{y}) \in \{0, 1\}$. As a consequence, this algorithm does not generalize well to unseen data.

2.2.3.2. Maximum entropy and log-linear models

The maximum entropy and log-linear models are duals of each other when seen as optimization programs. Therefore, both of them are essentially the same algorithm. In these algorithms, a parameterized distribution is discriminatively defined over an output $\tilde{\mathbf{y}}$ (or sometimes generatively over both the input $\mathbf{x}$ and the hypothetic label $\tilde{\mathbf{y}}$), the feature function $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is seen as the sufficient statistics of this distribution:

$$p(\tilde{\mathbf{y}}; \mathbf{x}, \mathbf{w}) = \frac{1}{z(\mathbf{x}, \mathbf{w})} e^{\mathbf{w}^T \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}})} \qquad \text{(Equation 2.10)}$$

The function $z(\mathbf{x}, \mathbf{w})$ is the normalization function, and is called the partition function. For $z(\mathbf{x}, \mathbf{w})$ we have:

$$z(\mathbf{x}, \mathbf{w}) = \sum_{\tilde{\mathbf{y}} \in \mathcal{Y}} e^{\mathbf{w}^T \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}})} \qquad \text{(Equation 2.11)}$$

The higher the value of $p(\tilde{\mathbf{y}}; \mathbf{x}, \mathbf{w})$ is for a specific $\tilde{\mathbf{y}}$, the more probable it is that $\tilde{\mathbf{y}}$ is "close" to the true labeling $\mathbf{y}$. Sometimes, $L(\tilde{\mathbf{y}}; \mathbf{x}, \mathbf{w}) = -\log p(\tilde{\mathbf{y}}; \mathbf{x}, \mathbf{w})$ is used as measure of unlikeliness of $\tilde{\mathbf{y}}$; smaller $L(\tilde{\mathbf{y}}; \mathbf{x}, \mathbf{w})$ means better $\tilde{\mathbf{y}}$:

$$\begin{aligned} L(\tilde{\mathbf{y}}; \mathbf{x}, \mathbf{w}) &= -\log p(\tilde{\mathbf{y}}; \mathbf{x}, \mathbf{w}) \\ &= -\mathbf{w}^T \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) + log(\sum_{\tilde{\mathbf{y}} \in \mathcal{Y}} e^{\mathbf{w}^T \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}})}) \qquad \text{(Equation 2.12)} \end{aligned}$$

The maximum entropy framework is one of the most successful methods for structured prediction. For example McCallum et al. applied this method to sequence labeling problems (McCallum et al., 2000), and a lot of follow-up work applied maximum entropy structured prediction in different disciplines (Califf and Mooney, 2003; McDonald and Pereira, 2005; Begleiter et al., 2004; Punyakanok and Roth, 2001; Chieu and Ng, 2002; Shen et al., 2007; Domke, 2013).

It is worth mentioning that conditional random fields (CRFs) can be seen as a more general framework where a probability distribution is fitted to the data, and the inference could be performed over structured outputs as well.

2.2.3.3. Re-ranking and search-based methods

Re-ranking is mostly applied to the natural language processing problems. Assume that we have access to the Oracle that solves some inference problem, but instead of generating "the best" output, it generates a list of "n best" outputs. Then, the learner's goal is to build a second model for choosing "one output" from the "n best" outputs. A second model then improves this initial ranking, using additional features as evidence. This approach allows a tree to be represented as an arbitrary set of features, without concerns about how these features interact or overlap, and without the need to define a derivation which takes these features into account (Collins and Duffy, 2002; Collins and Koo, 2005).

Re-ranking has been applied in a variety of NLP problems including parsing (Collins and Duffy, 2002; Collins and Koo, 2005; Charniak and Johnson, 2005), machine translation (Shen et al., 2004; Och et al., 2003), question answering (Ravichandran et al., 2003), semantic role labeling (Toutanova et al., 2005), and other tasks. A main feature of re-ranking is that different loss functions can be easily embedded into the algorithm and immediately tested. There are also some drawbacks. For example, in a re-ranking algorithm, one should have an Oracle for choosing n-best initial ranking, which may not be available, or n may be too large to be useful.

Search-based structured prediction can be seen as an improved and more advanced version of re-ranking. These algorithms are mostly developed by the re-enforcement learning community and have a flavor of solving the structured prediction problems from a planning perspective. Daumé et al. (Daumé Iii et al., 2009) introduced search-based structured prediction with the SEARN (SEarch And leaRN) algorithm. This algorithm integrates searching and learning to solve structured prediction problems. SEARN is a meta-algorithm that transforms structured prediction problems into simple classification problems, to which any binary classifier may be applied. SEARN is able to learn prediction functions for different loss functions and different features functions. There are several other related works that use similar techniques (Daumé III and Marcu, 2005; Daumé III, 2009b,a; Doppa et al., 2012).

2.2.3.4. Maximum-margin Markov networks

The max-margin Markov network (M^3N) class of structured prediction methods are a generalization of max-margin methods in traditional machine learning (also known as support vector machines (SVM)) to structured output prediction settings. The early work by Taskar et al. (Koller et al., 2003; Taskar et al., 2004a, 2005) was followed by a large quantity of additional progresses in development of max-margin methods (Tsochantaridis et al., 2006, 2004; Yu and Joachims, 2009; Sen et al., 2008; McDonald et al., 2007).

To date, the state-of-the-art structural SVM is the 1-slack formulation (Joachims et al., 2009), which solves the following optimization program:

$$\begin{aligned} &\underset{\mathbf{w},\zeta}{\text{minimize}}\ f(\mathbf{w}) + C\zeta \quad \text{subject to} \\ &\zeta \geq \max_{\tilde{\mathbf{y}}}\ \mathbf{w}^T(\phi(\mathbf{x},\tilde{\mathbf{y}}) - \mathbf{f}(\mathbf{x},\mathbf{y})) + \Delta(\mathbf{y},\tilde{\mathbf{y}}) \end{aligned} \qquad \text{(Equation 2.13)}$$

$f(\mathbf{w})$ is a regularization function, that penalizes "large" weights. Depending on the application, $f(\mathbf{w})$ can be any convex function in general. Semi-homogeneous functions, such as norms, or positive powers of norms are among the favorite choices[2]. $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is the most commonly used regularization function. For simplicity, I have expressed the input data as a single training example, but it can easily be expanded to set of N independent examples, each of which makes an independent contribution to the loss function. The variable ζ is the only slack variable, which should be minimized, along with the regularization function.

The large-margin Markov networks are developed as convex optimization programs. Therefore, it is mathematically convenient to derive robust formulations based on them. In this dissertation, we mainly focus on large-margin methods.

2.2.4. Optimization Algorithms

In most of the methods that we described above, the learning algorithm is embedded into the model, but for the max-margin methods, we usually come up with a mathematical optimization program. In the following, we briefly explain two of the state-of-the-art optimization algorithms that are used for structured learning.

– Cutting plane algorithm:

[2]A function $f(z)$ is semi-homogeneous if and only if $f(az) = a^\alpha f(z)$ for some positive α.

In parameter learning of the max-margin structured methods, the goal is to select the parameters for which the score of the true labels is ranked higher than the score of all alternate labels. Theoretically, this can be done via a convex optimization program, such as a quadratic program. The issue is that the number of alternate labels is usually exponential in the input size; therefore, listing all of them is intractable. The cutting plane algorithm at each iteration finds the alternate labeling that is most different from the true labeling and has the highest score, then adds appropriate constraints to make sure the score of the true labeling is relatively higher than this alternate labeling (Tsochantaridis et al., 2004, 2006; Koller et al., 2003; Taskar et al., 2005; Yu and Joachims, 2009; Joachims et al., 2009)

– Column generation:

  We can solve the convex program that is generated by the max margin approach in its dual form. The dual optimization program has a similar difficulty where the number of the dual variables is exponential in the input size. Similar to the cutting plane algorithm, the column generation method selects a dual variable at each iteration, and then adds it to the dual program. Solving the problem in its dual form is useful because then we can use the power of kernel functions. There are several works that use column generation for parameter learning (Taskar et al., 2005; Teo et al., 2008; Smola et al., 2007; McAuley et al., 2008).

– Exponentiated gradient:

  The exponentiated gradient algorithm also solves the optimization program in its dual form and uses a gradient ascent algorithm for each update in each iteration. The key point in the algorithm is that the gradient is exponentiated (i.e. e^g is used instead of the gradient g), and there are convergence theorems as well as experimental evaluations that prove the efficiency of this approach (Kivinen and Warmuth, 1997; Bartlett et al., 2004; Globerson et al., 2007; Collins et al., 2008).

2.3. Adversarial machine learning

In this section, we discuss the theoretical framework of adversarial machine learning in general, and at the same time address the main branches of the existing work that apply to structured prediction problems.

Adversarial machine learning studies machine learning techniques that are robust against adversarial components, which rule over the process of input data generation. As security challenges are increasing, the need for adversarial machine learning algorithms is becoming more apparent these days (Laskov and Lippmann, 2010). In analogy with security problems, adversarial machine learning can be seen as a game between two players, where one player wants to protect the normal functionality of a system, and the other player wants to pursue its malicious goals. In adversarial machine learning terminology, the first player is called the learner (or the defender), and the second player is called the adversary (or the attacker) (Dalvi et al., 2004). There has been a comprehensive body of work in recent years that examines the security of machine learning systems; this set involves different classes of possible attacks against machine learning systems (Lowd and Meek, 2005a; Globerson and Roweis, 2006; Teo et al., 2008; Lowd and Meek, 2005b; Blanzieri and Bryl, 2008; Brückner and Scheffer, 2009; Nelson, 2010; Brückner and Scheffer, 2011; Dreves et al., 2011; Brückner et al., 2012; Dritsoula et al., 2012; Sawade et al., 2013).

In the following subsection, we briefly address some of the most important aspects of the state-of-the-art methods, and we will discuss the common themes in adversarial machine learning algorithms.

We will also talk about regret minimization algorithms, which are somewhat complementary to the adversarial machine learning. In the regret minimization framework, Nature behaves like an adversary and sets the costs and rewards. The goal is to choose a sequence of actions that minimizes the future regret. Regret is defined as the sum of all incurred costs of chosen actions at all time steps, minus the sum of the costs when only one best-fixed action or policy had been taken at all the times. The best-fixed action would be the one that would have been selected if all of the costs were known in hindsight.

In this section, our perspective is mostly from the learner's point of view, and we categorize the adversarial attacks based on higher-level properties of an adversary. For an extensive collection of possible threats that make most of the classical machine learning algorithms vulnerable to adversarial attacks refer to Nelson (2010).

2.3.1. Adversary's theoretical model

We start this section with some definitions:

Antagonistic adversary and zero-sum games: The adversary's goals are explicitly against the duties of the learner; i.e. the adversary's win equally means the learner's loss and vice-versa. These games are called zero-sum, and such an opponent is known as an antagonistic adversary.

Non-antagonistic adversary and non-zero-sum games: If the opponent's goals are implicitly against the learner's goals, then the adversary is seeking its benefits, which may or may not be directly harmful to the learner. Whenever the amount of bilateral rewards and losses of each side of the game are not necessarily equal, then the game is non-zero-sum. If increasing the cost of the learner is not the primary aim of the adversary, then it is a non-antagonistic adversary.

Modeling the non-zero-sum game is relatively simple. Let $\mathbf{w} \in \mathcal{W}$ be the parameters of the learners model, and $\mathbf{a} \in \mathcal{A}$ be the parameters of the adversary's model, through which the adversary directly affects the performance of the machine learning algorithm. $\mathcal{W}$ and $\mathcal{A}$ are respectively the action space for the learner and the adversary. Also, let $r_a(\mathbf{w}, \mathbf{a})$ be the loss function that the learner wants to minimize by choosing the right $\mathbf{w}$[3]. An antagonistic adversary wants to maximize the loss of the learner by selecting an appropriate action $\mathbf{a}$. Therefore, the adversarial game can be formulated as:

$$\min_{\mathbf{w} \in \mathcal{W}} \max_{\mathbf{a} \in \mathcal{A}} \quad r_a(\mathbf{w}, \mathbf{a}) \qquad \text{(Equation 2.14)}$$

We present a general abstraction of adversarial games in Algorithm 2.

The machine learning algorithm (the learner) chooses an algorithm such as decision tree classification, Naïve Bayes, support vector machine, etc., and learns the parameters of the selected model based on its prior belief about the adversary and the previously observed data. On the other hand, the adversary also chooses an action from its plausible set of actions; this action is selected

[3]The function $r_a(\mathbf{w}, \mathbf{a})$ is the reward of the adversary. In a zero-sum game, the reward function for the learner is $r_l(\mathbf{w}, \mathbf{a}) = -r_a(\mathbf{w}, \mathbf{a})$; therefore, $r_a(\mathbf{w}, \mathbf{a})$ is the loss function from the learner's perspective.

Algorithm 2 Adversarial Game

```
Initialize:
  – Learner's prior belief:
      ∗ The learner chooses a model $\mathcal{M}$ as the machine learning algorithm.
      ∗ The learner initializes its belief about the adversary's set of strategies:
        $\hat{\mathcal{A}}$ based on the previous observations.
      ∗ The learner selects parameters $\mathbf{w}$ of the model $\mathcal{M}$ based on $\hat{\mathcal{A}}$ and the
        earlier observations.
  – Adversary's prior belief:
      ∗ The adversary chooses a set of strategies $\mathcal{A}$ based on its own prior
        knowledge and restrictions.
      ∗ The adversary initializes its initial belief on the learner's model $\hat{\mathcal{M}}$, and
        its belief on the model parameters $\hat{\mathbf{w}}$.
      ∗ The adversary chooses an action $a \in \mathcal{A}$.
  – Nature sets the laws:
      ∗ Nature chooses a set of incentives $\mathcal{R}$.
while Set of Incentives $\mathcal{R}$ exists do
    Defend:
  – The learner updates its approximation of the adversary's set of strategies $\hat{\mathcal{A}}$.
  – The learner updates parameters $\mathbf{w}$ based on $\hat{\mathcal{A}}$ and the observed adversary's
    action $\mathbf{a}$.
  – The learner gains reward $r_l(\mathbf{w}, \mathbf{a}) \in \mathcal{R}$
    Attack:
  – The adversary chooses an attack $a \in \mathcal{A}$
  – The adversary gains reward $r_a(\mathbf{w}, \mathbf{a}) \in \mathcal{R}$
  – The adversary updates $\hat{\mathcal{A}}$ based on the observed reward $r_a(\mathbf{w}, \mathbf{a})$ and its
    new understanding of $\mathcal{R}$.
    Nature:
  – Nature updates $\mathcal{R}$.
end while
```

based on the adversary's prior belief about the learner's choice of the model and its parameters. Note that each of the adversary's or learner's moves can be randomized or deterministic. In fact, each of the players may choose a mixed strategy, rather than a fixed move. It is Nature that decides on the amount of positive or negative payoffs of each combination of the strategies that are chosen by the players. For example, in email spam detection, there are three sides: the spam-filter, the spammer, and the user of the email service. Some emails are considered as spam by some users but are valuable information for some other users. Therefore, if the spam filter algorithm wants to use a fixed model for all users, then it should carefully update its belief about the pay-offs that are made by Nature.

The order of the itemized events in Algorithm 2 can be completely arbitrary. Each of the existing approaches to adversarial machine learning is designed based on some assumptions about Algorithm 2. In the following, we briefly categorize the main possibilities.

2.3.1.1. Type of adversarial problems

A key point of difference, among algorithmic approaches that are designed for adversarial machine learning, is the order in which the events of Algorithm 2 occur. In particular, the existing studies are mostly based on three general assumptions regarding the possible order in which events occur:

- **Based on Stackelberg competition scenario :** The Stackelberg competition model is a strategic game model where one of the players (called "the leader") plays first, and then the other player (called "the follower") plays sequentially. This model is the closest model to real-world challenges. The learner (the leader) updates its model parameters after observing the adversary's (the follower's) action, and possibly incurs some losses (Globerson and Roweis, 2006; Teo et al., 2008; Brückner and Scheffer, 2011; Sawade et al., 2013; Torkamani and Lowd, 2013).

- **Based on Nash Equilibria:** In these models, although the order of events is arbitrary, hypothetically, there exist optimum joint strategies of both players, where no player gains more rewards by deviating from its current policy. It is a known fact from Game Theory that such optima do not necessarily exist among pure strategies (Brückner and Scheffer, 2009; Dreves et al., 2011; Brückner et al., 2012; Dritsoula et al., 2012).

- **Based on Poisoning the Training Data:** The adversary generates several specially designed data points and injects them into the training data. The adversary's goal in these kind of attacks is to make the machine learning algorithm learn a wrong model in the first place. Such attacks can be designed to target individual machine learning algorithms (Biggio et al., 2012; Dekel et al., 2010; Biggio et al., 2013a,b, 2014).

- **Based on Regret Minimization:** In these models, the adversary and Nature are the same, and Nature chooses a new cost function for each action of the learner at each iteration of the game. The goal is to minimize the regret that the learner would suffer, in comparison with what they would have chosen at a time in which they knew all of the costs imposed by Nature, in hindsight, and had chosen a fixed strategy as the response. (Shalev-Shwartz, 2011; Ross et al., 2011, 2010).

In general, finding the Nash equilibrium becomes harder when the dimensionality of the players' actions is large and the utility functions are arbitrary. Brückner and Scheffer (2009) show that under certain convexity and separability conditions of the utility function, a Nash equilibrium exists; this equilibrium can be found by simulating the adversarial game. Therefore, Stackelberg competitions are more approachable techniques, because the learner should select the strategy that restricts the worst-case adversary in a minimax formulation. The learner attempts to minimize a loss function assuming a worst-case adversarial manipulation. An unrealistic assumption that many of the papers make to simplify the problem is the continuity of feature functions, which does not hold in many domains (Globerson and Roweis, 2006; Teo et al., 2008; Brückner and Scheffer, 2011; Sawade et al., 2013; Brückner and Scheffer, 2009; Dreves et al., 2011; Brückner et al., 2012; Dritsoula et al., 2012).

Globerson and Roweis (2006) formulate the problem of feature deletion at test time as a Stackelberg game. This method is only applicable to binary and multi-label classification and does not apply to the structured output prediction problems. Another weakness of this approach is that it is only robust to feature deletion; other possible adversarial manipulations of data, such as feature, are ignored. Teo et al. (2008) generalized the former method to all invariants of input data[4]. This method is not practical whenever the number of possible transformations is exponential in the input size (or sometimes infinite).

2.3.1.2. Knowledge about the opponent

From the knowledgeability perspective, there are two types of adversaries: passive or active. Passive adversaries do not have access to the learner's model, so they try to attack the system, and observe the outcomes, in order to infer the parameters of the algorithm working behind the scenes. Active adversaries have

[4]In machine learning and computer vision terminology, an "invariant" of a data point $\mathbf{x}$ with label $\mathbf{y}$ is a variation of $\mathbf{x}$, namely $\tilde{\mathbf{x}}$, that the classifier of interest still labels it as $\mathbf{y}$.

full access to the learner's model and the parameters that the learner has selected for the model (Lowd and Meek, 2005b,a; Blanzieri and Bryl, 2008). A passive adversary may converge to an active adversary in theory, especially if the learner does not update its model parameters. In real-world' problems, the adversaries are passive in general, but most of the existing studies focus on the active adversary assumption.

It is also important for the learner to know the adversary's limitations and incentives. If the model is non-antagonistic, then the adversary has its own incentives; knowing these incentives can be used in modeling the adversary. This knowledge can be used in generating robust model parameters for the learner. The effectiveness of our methods depends on how accurately we model the adversary, but the true costs and constraints of the adversary are rarely known in advance. There is not much work that models the incentives of the adversary, but there are a few methods that assume that adversary is rational (Nguyen et al., 2013).

One advantage of the learner is the adversary's limitations; most of the Stackelberg games use this fact to learn robust models by incorporating the restrictions of the adversary into the learning algorithm (Globerson and Roweis, 2006; Teo et al., 2008; Torkamani and Lowd, 2013; Livni and Globerson, 2012). Some other recent papers have considered the relationship between regularization and robustness to restricted adversaries in SVMs. Xu et al. (2009) demonstrate that using a norm as a regularizer is equivalent to optimizing against a worst-case adversary that can manipulate features within a ball defined by the dual norm. There are several related works that expand this idea in different directions (Xu et al., 2010; Xu and Mannor, 2012). For example in follow-up work Xu et al. expand their approach to the robust regression problem(Xu et al., 2010).

2.3.1.3. The role of Nature

In Algorithm 2, we have separated the adversary and Nature. In fact, the adversary follows the rules that Nature sets. For example, in stock markets, there are some traders (adversaries) who want to increase their pay-offs by choosing the right portfolio, but the demands of the market are the main criteria that affect the stock indices. Another example is the laws of physics that Nature sets. A robot controller algorithm should be robust to adversarial accidents that threaten the autonomous robot agents, but falling from 2-feet-tall piece of rock is clearly different than falling from a cliff which has the height of 500 feet. As a result, it is important for both learner and the adversary to learn the laws of Nature as well.

2.3.2. Adversarial learning techniques

In this subsection, we review some of the primary techniques in adversarial machine learning that are applicable to supervised learning methods, and we formulate the adversarial game as set of optimization programs.

2.3.2.1. Utility-based approaches

Utility-based approaches are of the early works in adversarial machine learning that are applicable to structured prediction as well. In these models, both the learner and the adversary have their specific utility functions. The utilities are some arbitrary reward functions. In a game-theoretic framework, each of the players tries to maximize its reward. Brückner and Scheffer take this approach in some of their papers. They show that for particular non-antagonistic utility functions, the prediction game has a unique Nash equilibrium, and they derive a simulation-based algorithm for finding the converging models (Brückner and

Scheffer, 2009; Brückner et al., 2012). In another work, they model the interaction between the learner and the adversary as a Stackelberg game, in which the learner plays the role of the leader and the adversary reacts to the learned model (Brückner and Scheffer, 2011). This framework is, in fact, a minimax scenario where the learner tries to minimize the maximum possible harmful damage that the adversary can cause. These methods are not designed for structured prediction problems, but their underlying framework is general purpose. Satisfying some of the assumptions may not be possible, especially for finding the Nash equilibrium. The main drawback of these works is that the formulations assume a relaxed action space; this assumption does not hold in many structured (and non-structured) output spaces.

Other works expand the analysis of the conditions for the existence of the Nash equilibrium; for example, Dreves et al. have analyzed the Karush-Kuhn-Tucker (KKT) conditions for which the generalized Nash equilibrium exists (Dreves et al., 2011).

2.3.2.2. Max-margin-based Adversarial Structured Learning

The max-margin based algorithms include the large class of SVM classifiers. Therefore, many adversarial methods are based on max-margin formulations. The following is a brief review of each of these methods.

– **Embedding the simulated adversary:**

The key idea for making max-margin learning approaches robust to adversarial data manipulation is to embed the adversarial uncertainty component into the optimization program of the max-margin method. (Schölkopf et al., 1997) were among the first authors who used the idea of using virtual (e.g. noise-polluted) samples for training the model. This approach was first used as an embedded part of the algorithm for binary SVMs by Globerson and Roweis (2006) when a limited number of the features could be set to zero by the adversary at test time. Later Teo et al. (2008) expanded this idea to include a wider class of possible adversaries. The main limitation of the latter work is that there should exist an efficient computational procedure for simulating the adversary. This is not always tractable because the number of possible adversarial manipulations of input data can be extremely large.

Other approaches with a similar nature. For example, Biggio et al. (2011) formulate the problem in the dual form and model the adversarial noise as the Hadamard product of a noise matrix and the kernel matrix. Some other authors assume that the adversarial noise is drawn from a distribution and try to ensure robustness to that kind of perturbation (Livni and Globerson, 2012; Maaten et al., 2013).

– **Robustness by regularization:**

In general, robust optimization addresses optimization problems in which some degree of uncertainty governs the known parameters of the model. Ben-Tal and Nemirovski (Ben-Tal and Nemirovski, 1998, 1999, 2000, 2001) showed that there exist a range of applications that could be formulated in a robust convex optimization framework. Robust linear programming is a central method in most of these formulations. Bertsimas and Sim (2004) show that for box-bounded disturbances, the parameters can take the worst-case value, and there is a trade-off between optimality and robustness. In Bertsimas et al. (2004), the authors focus on the case when the disturbance of the inputs is restricted to an ellipsoid around the actual values defined by some norm. They show that the robust linear programming problem can be reduced to a convex cone program, where the conic constraint is defined by the dual of the original norm. A number of other authors have explored the application of robust optimization to classification problems (e.g.,(Lanckriet et al., 2003; El Ghaoui et al., 2003; Bhattacharyya et al., 2004; Shivaswamy et al., 2006)). Recently, Xu et al. (2009) showed that regularization of support vector machines can be derived from a robust formulation, and they also argue that robustness in feature space entails robustness in sample space.

– **Robustness to poisoning attacks:**

  Poisoning attack is used to refer to a scenario, where the adversary injects some corrupted samples to the training data to make sure that the classifier will learn a wrong model, and as a result, the test error increases. To the best of my knowledge, there is no existing published work that attempts to guarantee robustness against these kind of attacks. Filling this gap is worthwhile, and it is specially relevant to applications in which the number of training samples is limited.

  Biggio et al. (2012) have studied this problem for non-structural prediction. They investigate a family of poisoning attacks against SVMs. Most of the learning algorithms assume that their training data comes from a natural distribution, and therefore they are vulnerable to these kind of attacks. An intelligent adversary can, to some extent, predict the change of the SVM's decision function due to malicious input and use this ability to construct malicious data. Dekel et al. (2010) solve a similar problem for binary SVMs, where they apply several relaxations to the integer program formulation of the problem, and use L_∞ as the regularizer. Because of this choice of regularizer, they end up with a linear program. In their paper, they state that with the choice of L_∞ regularization their method is more efficient and don't go into more arguments. There are some other works in the literature that attempt to train models that are robust to poisoning attacks (Biggio et al., 2013a,b, 2014).

2.3.2.3. Online learning and regret minimization:

Online learning is based on the idea of choosing the best strategy based on the data that is being received in a stream (Shalev-Shwartz, 2011). The amount of available data is usually huge. Therefore, we prefer to look at each data point only for a limited number of times – ideally only once. Regret minimization is an adversarial method for learning in online settings.

2.4. Applications of adversarial structured learning

Improving the performance of structured prediction algorithms is one of our main contributions in this thesis. In this section, we review the significance that this improvement will have on the real-world applications.

2.4.1. Collective Classification

Many real-world relational learning problems can be formulated as a collective classification problem. For example, webspam detection can be formulated as a joint classification problem where each webpage is either spam or non-spam, and the label of each webpage not only depends on its contents but also depends on the label of neighboring webpages that are linked to it (Sen et al., 2008; Abernethy et al., 2010).

Our paper "Collective Adversarial Collective Classification" (Torkamani and Lowd, 2013), is the first published work in the field of structured output prediction that is designed to be directly robust against adversarial manipulation of data at test time. We assumed that the adversary can change up to D attributes of all webpages, and by incorporating this limitation of the adversary[5] in a robust optimization program, we come up with an efficient method[6] for robustly solving the problem of collective classification in associative Markov networks (Taskar et al., 2004a).

Other researchers have solved this problem with an implicit effort to address the robustness issue. Sen et al. (2008) discuss that the "Iterative Classification Algorithm" (Jensen and Neville, 2002; Lu and Getoor, 2003) is relatively robust to the order that the nodes are visited, but their method is not robust to the manipulation of test data. Tian et al. (2006) introduce an additional heuristic weight on top of a dependency network (Neville and Jensen, 2007; Lowd and Shamaei, 2011) to model the strength of the dependencies. Although this additional weight makes the approach robust to random noise, the method is not robust to malicious noise. McDowell et al. (2009) introduce the cautious iterative classification algorithm, where at each local classification, the classifier also generates a confidence criterion about the performed classification. If this criterion is less than some threshold, the predicted label is ignored by the algorithm. This

[5]This is the main limitation of the adversary. Therefore, the adversary cannot manipulate "everything" in the network.

[6]For binary labels, such as spam detection, the efficiency is guaranteed. When there are more than two possible labels, the results are approximate, in theory but in practice, we get pretty accurate results.

method is also heuristic and does not rely on the related literature of robust machine learning.

Abernethy et al. (2010) introduce the “WITCH” algorithm, which uses a graph regularization approach to utilizing the link information for regularizing the model parameters. Their method gains implicit robustness due to regularization, but it is not robust to adversarial attacks against the collective classification algorithms.

2.4.2. Anomaly Detection

Anomaly detection is the problem of detecting unusual samples among some ordinary ones. For example, detecting network intrusions or instances of credit card fraud are acts of anomaly detection. An intrusion detection system is now an important part of any computer network. When a set of agents in the network collaborate in an attack, then the network protection system needs to perform structured prediction to determine the role of each agent in the network. There is a group of papers that use conditional random fields or hidden Markov models to perform this task (Gupta et al., 2007, 2010; Qiao et al., 2002). The main drawback of these methods is the issue of robustness of the algorithms. In other words, these methods use machine learning algorithms to improve the robustness issue of the system, but the used algorithms themselves are not robust to engineered attacks.

Song et al. (2013) introduce a one-class classification approach for detecting the sequential anomalies. Their method is robust to outliers in the training data. Although the method is elegant, what makes it less applicable to adversarial settings is that the adversarially manipulated samples are different than outliers.

In particular, the adversary manipulates the data as a response to the learned parameters of the classification method.

2.4.3. Practical applications

The following is a list of some of the real-world applications of adversarial structured prediction.

2.4.3.1. Security applications

Security issues are becoming more serious and critical these days, and naturally, machine learning tools are also being used to solve some of these problems. The security challenges can be formulated as a game between the defender (or learner) and the attacker (or the adversary). Not only the action space in security games is large, but also the limited resources of the defender is a challenge in most cases. In fact, in real-world security problems, there are not enough agents to patrol all the targets that the adversary could attack. Therefore, deciding the placement of the resources is highly important.

Pita et al. (2008); Jain et al. (2010b) have developed an algorithm called ARMOR, which is now deployed at the Los Angeles International Airport (LAX) to randomize the checkpoints on the roadways that enter the airport. By randomization, the strategies are drawn from some mixture of strategy distributions, rather than a taking a fixed pure strategy all the times. As a result, the criminal will not be able to precisely determine the next action. Some other related works are IRIS (Tsai et al., 2009), fast generation of the flight schedules (Jain et al., 2010a), PROTECT (Shieh et al., 2012; Fang et al., 2013), GUARDS

(Pita et al., 2011), among others (Yin et al., 2011, 2012; Jiang et al., 2013b,a; Basilico et al., 2009; An et al., 2012; Korzhyk et al., 2011).

Dickerson et al. (2010) look at security games from a graph theoretic approach and propose a greedy algorithm for protecting the moving targets from adversaries.

2.4.3.2. Computer vision

Both robustness and structured output prediction are highly needed in the computer vision applications.

Fua et al. (2013) propose a working set based approximate subgradient descent algorithm to solve the optimization program of the structured SVM. They solve an image segmentation problem, where exact inference is intractable, and the most violated constraints can only be approximated. They randomly sample new constraints, instead of computing them using the more expensive approximate inference techniques. This random sampling is not designed to explicitly block the adversaries, but it gains some robustness at the prediction time. From the theory point of view, we know that this method should not work well in general, because the randomly selected constraints may be insignificant, and this slows down the convergence of the algorithm. However, this method has been successful in their application.

Gong et al. (2012) propose a structured prediction method where the output space is a subset of two distinct manifolds, and their method tries to be robust to noise and to choose the output from the right manifold. This method is shown to be efficient in human motion-capturing from videos. Ranjbar et al. (2013) focuses on keeping robust features in advance to gain robustness in the structured prediction. Exploiting the domain knowledge is also a method that increases robustness in play-type recognition for a football game, which is recorded by noisy sensors (Chen et al., 2014b).

2.4.4. Speech recognition

As the applications of structured prediction grow in different subfields of signal processing, the robustness issue becomes more prominent. Speech recognition is an attractive example. Zhang et al. have parameterized a noise model, and they have embedded it into the optimization program. They optimize for the noise control parameter as well (Zhang et al., 2010, 2011). In their problem the noise in the speech signal is not adversarial, and adversarial speech recognition is also among the fields that have major applications in real-world problems.

In the next chapter, we introduce a novel method for efficient collective classification in adversarial settings.

CHAPTER III

CONVEX ADVERSARIAL COLLECTIVE CLASSIFICATION

This work was published in the proceedings of the thirtieth International Conference on Machine Learning (ICML 2013). I was the primary contributor to the methodology and writing, and designed and conducted the experiments. My Ph.D. advisor, Dr. Daniel Lowd contributed partly to the methodology and writing. Daniel Lowd was the principle investigator for this work.

In collective classification (Sen et al., 2008), we wish to jointly label a set of interconnected objects using both their attributes and their relationships. For example, linked web pages are likely to have related topics; friends in a social network are likely to have similar demographics; and proteins that interact with each other are likely to have similar locations and related functions. Probabilistic graphical models, such as Markov networks, and their relational extensions, such as Markov logic networks (Domingos and Lowd, 2009a), can handle both uncertainty and complex relationships in a single model, making them well-suited to collective classification problems.

However, many collective classification models must also cope with test data that is drawn from a different distribution than the training data. In some cases, this is simply a matter of concept drift. For example, when classifying blogs, tweets, or news articles, the topics being discussed will vary over time. In other cases, the change in distribution can be attributed to one or more adversaries actively modifying their behavior in order to avoid detection. For example, when search engines began using incoming links to help rank web pages, spammers began posting comments on unrelated blogs or message boards with links back to their websites. Since incoming links are used as an indication of quality, manufacturing incoming links makes a spammy web site appear more legitimate. In addition to web spam (Abernethy et al., 2010; Drost and Scheffer, 2005), other explicitly adversarial domains include counter-terrorism, online auction fraud (Chau et al., 2006), and spam in online social networks.

Rather than simply reacting to an adversary's actions, recent work in adversarial machine learning takes the proactive approach of modeling the learner and adversary as players in a game. The learner selects a function that assigns labels to instances, and the adversary selects a function that transforms malicious instances in order to avoid detection. The strategies chosen determine the outcome of the game, such as the success rate of the adversary and the error rate of the chosen classifier. By analyzing the dynamics of this game, we can search for an effective classifier that will be robust to adversarial manipulation. Even in non-adversarial domains such as blog classification, selecting a classifier that is robust to a hypothetical adversary may lead to better generalization in the presence of concept drift or other noise (Figure 3.1).

Early work in adversarial machine learning included methods for blocking the adversary by anticipating their next move (Dalvi et al., 2004), reverse engineering classifiers (Lowd and Meek, 2005b,a) (and later: (Nelson et al., 2010)), and building classifiers robust to feature deletion or other invariants (Globerson and Roweis, 2006; Teo et al., 2008). More recently, Brückner and Scheffer showed that, under modest assumptions, Nash equilibria can be found for domains such as spam (Brückner and Scheffer, 2009). However, current adversarial methods assume that instances are independent, ignoring the relational nature of many domains.

In this chapter, we present Convex Adversarial Collective Classification (CACC), which combines the ideas of associative Markov networks (Taskar et al., 2004a) (AMNs) and convex learning with invariants (Teo et al., 2008). Unlike previous work in learning graphical models, CACC selects the most effective weights *assuming a worst-case adversary* who can modify up to a fixed number of binary-valued attributes. Unlike previous work in adversarial machine learning, CACC allows for dependencies among the labels of different objects, as long as these dependencies are associative. Associativity means that related objects are more likely to have the same label, which is a reasonable assumption for many collective classification domains. Surprisingly, all of this can be done in polynomial time using a convex quadratic program.

In experiments on real and synthetic data, CACC finds much better strategies than both a naïve AMN that ignores the adversary and a non-relational adversarial baseline. In some cases, the adversarial regularization employed by CACC helps it generalize better than AMNs even when the test data is not modified by any adversary.

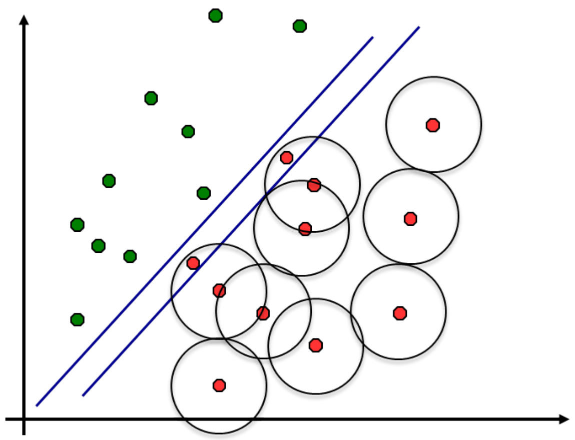


FIGURE 3.1. The adversary knows the parameters of our classifier and can maliciously modify data to attack. The learner should select the best classifier, assuming the worst adversarial manipulation.

3.1. Max-margin relational learning

We use uppercase bold letters ($\mathbf{X}$) to represent sets of random variables, lowercase bold letters ($\mathbf{x}$) to represent their values, and subscripts and superscripts (x_{ij}, y_i^k) to indicate individual elements in those sets.

Markov networks (MNs) represent the joint distribution over a set of random variables $\mathbf{X} = \{X_1, \ldots, X_N\}$ as a normalized product of factors:

$$P(\mathbf{X}) = \frac{1}{Z} \prod_i \phi_i(\mathbf{D}_i)$$

where Z is a normalization constant so that the distribution sums to one, ϕ_i is the ith factor, and $\mathbf{D}_i \subseteq \mathbf{X}$ is the scope of the ith factor. Factors are sometimes referred to as potential functions. For positive distributions, a Markov network can also be represented as a *log-linear model*:

$$P(\mathbf{X}) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(\mathbf{D}_i)\right)$$

where w_i is a real-valued weight and f_i a real-valued feature function. For the common case of indicator features, each feature equals 1 when some logical expression over the variables is satisfied and 0 otherwise.

A factor or potential function is *associative* if its value is at least as great when the variables in its scope take on identical values as when they take on different values. For example, consider a factor ϕ parameterized by a set of non-negative weights $\{w^k\}$, so that $\phi(y_i, y_j) = \exp(w^k)$ when $y_i = y_j = k$ and 1 otherwise. ϕ is clearly associative, since its value is higher when $y_i = y_j$. An *associative Markov network* (AMN) (Taskar et al., 2004a) is an MN where all factors are associative. Certain learning and inference problems that are intractable in general MNs have exact polynomial-time solutions in AMNs with binary-valued variables, as will be discussed later.

An MN can also represent a conditional distribution, $P(\mathbf{Y}|\mathbf{X})$, in which case the normalization constant becomes a function of the evidence, $Z(\mathbf{X})$.

In this chapter, we focus on collective classification, in which each object in a set is assigned one of K labels based on its attributes and the labels of related objects. We now give an example of a simple log-linear model for collective classification, which we will continue to use for the remainder of the chapter. Following Taskar et al. (2004a), let $y_i^k = 1$ if the ith object is assigned the kth label, and 0 otherwise. We use x_{ij} to represent the value of the jth attribute of the ith object. The relationships among the objects are given by E, a set of undirected edges of the form (i, j).

Our model includes features connecting each attribute x_{ij} to each label y_i^k, represented by the product $x_{ij}y_i^k$. To add the prior distribution over the labels, we simply define an additional feature $x_{i,0}$ that is 1 for every object, similar to a bias node in neural networks. For each pair of related objects $(i, j) \in E$, we also include a feature $y_i^k y_j^k$ which is 1 when both the ith and jth object are assigned label k. This leads to the following model:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{ijk} w_j^k x_{ij} y_i^k + \sum_{(i,j)\in E,k} w_e^k y_i^k y_j^k\right) \qquad \text{(Equation 3.1)}$$

Note that all objects share the same attribute weights, w_j^k, and all links share the same edge weights, w_e^k, in order to generalize to unseen objects and relationship graphs. This model can also be easily expressed as a Markov logic network

(MLN) (Domingos and Lowd, 2009a) in which formulas relate class labels to other attributes and the labels of linked objects.

MLNs make it easy to compactly describe very complex distributions. For example, a simple collective classification model can be defined using relatively simple formulas, as shown in Table **??**. The subscript j and superscript k indicate that different formulas are defined for each attribute $j \in \{1, \ldots, M\}$ and object label $k \in \{1, \ldots, K\}$. The formula from the first line defines features for the prior distribution over labels in the absence of any attributes or links. The next line relates each object's attributes to its label. The third line relates the labels of neighboring objects. Note that the first formula may be omitted as a special case of the second if we assume that a special bias attribute $\texttt{Attribute}_0(o)$ is true for every object o.

A common inference task is to find the most probable explanation (MPE), the most likely assignment of the non-evidence variables $\mathbf{y}$ given the evidence. This can be done by maximizing the unnormalized log probability, since log is a monotonic function and the normalization factor Z is constant over $\mathbf{y}$. For the simple collective classification model, the MPE task is to find the most likely labeling given the links and attributes:

$$\arg\max_{\mathbf{y}} \sum_{ijk} w_j^k x_{ij} y_i^k + \sum_{(i,j)\in E,k} w_e^k y_i^k y_j^k$$

In general, inference in graphical models is computationally intractable. However, for the special case of AMNs with binary-valued variables, MPE inference can be done in polynomial time by formulating it as a min-cut problem (Kolmogorov and Zabin, 2004). For $w_e^k \geq 0$, our working example of a collective classification model is an AMN over the labels $\mathbf{y}$ given the links E and attributes $\mathbf{x}$. In general, associative interactions are very common in collective classification problems since related objects tend to have similar properties, a phenomenon known as homophily. Markov networks and MLNs are often learned by maximizing the (conditional) log-likelihood of the training data (e.g., Lowd and Domingos (2007)). An alternative is to maximize the margin between the correct labeling and all alternative labelings, as done by max-margin Markov networks (M^3Ns) (Taskar et al., 2004b) and max-margin Markov logic networks (M^3LNs) (Huynh and Mooney, 2009). Both approaches are intractable in the general case. For the special case of AMNs, however, max-margin weight learning can be formulated as a quadratic program which gives optimal weights in polynomial time as long as the variables are binary-valued (Taskar et al., 2004a). We now briefly describe the solution of Taskar et al., which will later motivate our adversarial extension of AMNs. (We use slightly different notation from the original presentation in order to make the structure of $\mathbf{x}$ and $\mathbf{y}$ clearer.)

The goal of the AMN optimization problem is to maximize the margin between the log probability of the true labeling, $h(\mathbf{w}, \mathbf{x}, \hat{\mathbf{y}})$, and any alternative labeling, $h(\mathbf{w}, \mathbf{x}, \mathbf{y})$. For our problem, h follows from Equation 3.1: $h(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \sum_{i,j,k} w_j^k x_{ij} y_i^k + \sum_{(i,j)\in E,k} w_e^k y_i^k y_j^k$. We can omit the $\log Z(\mathbf{x})$ term because it cancels in the difference. Margin scaling is used to enforce a wider margin from labelings that are more different. We defined this difference as the Hamming distance: $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = N - \sum_{i,k} y_i^k \hat{y}_i^k$ where N is the total number of objects. We thus obtain the following minimization problem with an exponential number of constraints (one for each $\mathbf{y}$):

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & h(\mathbf{w},\mathbf{x},\hat{\mathbf{y}}) - h(\mathbf{w},\mathbf{x},\mathbf{y}) \geq \Delta(\mathbf{y},\hat{\mathbf{y}}) - \xi \quad \forall \mathbf{y} \in \mathcal{Y} \end{aligned} \qquad \text{(Equation 3.2)}$$

Minimizing the norm of the weight vector is equivalent to maximizing the margin. The slack variable ξ represents the magnitude of the margin violation, which is scaled by C and used to penalize the objective function. To transform this into a tractable quadratic program, Taskar et al. modify it in several ways. First, they replace each product $y_i^k y_j^k$ with a new variable y_{ij}^k and add constraints $y_{ij}^k \leq y_i^k$ and $y_{ij}^k \leq y_j^k$. In other words, $y_{ij}^k \leq \min(y_i^k, y_j^k)$, which is equivalent to $y_i^k y_j^k$ for $y_i^k, y_j^k \in \{0,1\}$. Second, they replace the exponential number of constraints with a continuum of constraints over a relaxed set of $y \in \mathcal{Y}'$, where $\mathcal{Y}' = \{\mathbf{y} : y_i^k \geq 0; \sum_k y_i^k = 1; y_{ij}^k \leq y_i^k; y_{ij}^k \leq y_j^k\}$. Since all constraints share the same slack variable, ξ, we can take the maximum to summarize the entire set by the most violated constraint. After applying these modifications, substituting in h and Δ, and simplifying, we obtain the following optimization problem for our collective classification task:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \mathbf{w} \geq 0; \\ & \xi - N \geq \max_{\mathbf{y}\in\mathcal{Y}'} \sum_{i,j,k} w_j^k x_{ij}(y_i^k - \hat{y}_i^k) \\ & \qquad + \sum_{(i,j)\in E,k} w_e^k (y_{ij}^k - \hat{y}_{ij}^k) - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \end{aligned} \qquad \text{(Equation 3.3)}$$

Finally, since the inner maximization is itself a linear program, we can replace it with the minimization of its dual to obtain a single quadratic program (not shown).

For the two-class setting, Taskar et al. prove that the inner program always has an integral solution, which guarantees that the weights found by the outer quadratic program are always optimal.

For simplicity and clarity of exposition, we have used a very simple collective classification model as our working example of an AMN. This model can easily be extended to allow multiple link types with different weights, link weights that are a function of the evidence, and higher-order links (hyper-edges), as described by Taskar et al. (2004a). Our adversarial variant of AMNs, which will be described in Section 4, supports most of these extensions as well.

3.2. Convex formulation

Collective classification problems are hard because the number of joint label assignments is exponential in the number of nodes. As discussed in Section 2, if neighboring nodes are more likely to have the same label, then the collective classification problem can be represented as an associative Markov network (AMN), in which max-margin learning and MPE inference are both efficient. To construct an adversarial collective classifier, we start with the AMN formulation (Equation 3.3) and incorporate an adversarial invariant, similar to the approach of Globerson and Roweis (2006). Specifically, we assume that the adversary may change up to D binary-valued features x_{ij}, for some positive integer D that we select in advance. We use $\hat{\mathbf{x}}$ to indicate the true features and $\mathbf{x}$ to indicate the adversarially modified features. The number of changes can be written as: $\Delta(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{i,j} x_{ij} + \hat{x}_{ij} - 2x_{ij}\hat{x}_{ij}$

We define the set of valid $\mathbf{x}$ as $\mathcal{X}' = \{\mathbf{x} : 0 \leq x_{ij} \leq 1; \Delta(\mathbf{x}, \hat{\mathbf{x}}) \leq D\}$. Note that $\mathcal{X}'$ is a relaxation that allows fractional values, much like the set $\mathcal{Y}'$ defined by

Taskar et al. We will later show that there is always an integral solution when both the features and labels are binary-valued.

In our adversarial formulation, we want the true labeling $\hat{\mathbf{y}}$ to be separated from any alternate labeling $\mathbf{y} \in \mathcal{Y}'$ by a margin of $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ given any $\mathbf{x} \in \mathcal{X}'$. Rather than including an exponential number of constraints (one for each $\mathbf{x}$ and $\mathbf{y}$), we use a maximization over $\mathbf{x}$ and $\mathbf{y}$ to find the most violated constraint:

$$\begin{aligned}
&\max_{\mathbf{y}\in\mathcal{Y}',\mathbf{x}\in\mathcal{X}'} h(\mathbf{w},\mathbf{x},\mathbf{y}) - h(\mathbf{w},\mathbf{x},\hat{\mathbf{y}}) + \Delta(\mathbf{y},\hat{\mathbf{y}}) \\
= &\max_{\mathbf{y}\in\mathcal{Y}',\mathbf{x}\in\mathcal{X}'} \sum_{i,j,k} w_j^k x_{ij} y_i^k + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k \\
&\qquad - \sum_{i,j,k} w_j^k x_{ij} \hat{y}_i^k - \sum_{(i,j)\in E,k} w_e^k \hat{y}_{ij}^k \\
&\qquad + N - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \qquad \text{(Equation 3.4)}
\end{aligned}$$

Next, we convert this to a linear program. Since $x_{ij}y_i^k$ is bilinear in $\mathbf{x}$ and $\mathbf{y}$, we replace it with the auxiliary variable z_{ij}^k, satisfying the constraints: $z_{ij}^k \geq 0$; $z_{ij}^k \leq x_{ij}$; and $z_{ij}^k \leq y_i^k$. The removes the bilinearity and is exactly equivalent as long as x_{ij} or y_i^k is integral.

Putting it all together and removing terms that are constant with respect to $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$, we obtain the following linear program:

$$\begin{aligned}
\max_{\mathbf{x},\mathbf{y},\mathbf{z}} \quad & \sum_{i,j,k} w_j^k (z_{ij}^k - \hat{y}_i^k x_{ij}) + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \\
\text{s.t.} \quad & 0 \leq x_{ij} \leq 1; \quad \sum_{i,j} x_{ij} + \hat{x}_{ij} - 2x_{ij}\hat{x}_{ij} \leq D \\
& 0 \leq y_i^k; \quad \sum_k y_i^k = 1; \quad y_{ij}^k \leq y_i^k; \quad y_{ij}^k \leq y_j^k \\
& z_{ij}^k \leq x_{ij}; \quad z_{ij}^k \leq y_i^k \quad \forall i,j,k \qquad \text{(Equation 3.5)}
\end{aligned}$$

Given the model's weights, this linear program allows the adversary to change up to D binary features. Recall that, in the AMN formulation, the exponential number of constraints separating the true labeling from all alternate labelings are replaced with a single non-linear constraint that separates the true labeling from the best alternate labeling (Eqs. Equation 3.2,Equation 3.3). This non-linear constraint contains a nested maximization. We have a similar scenario, but here the margin can also be altered by changing the binary features, affecting the probabilities of both the true and alternate labelings. By substituting this new MPE inference task (Equation 3.5) into the original AMN's formulation, the resulting program's optimal solution will be robust to the worst manipulation of the input feature vector:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \quad \text{s.t.} \;\; \mathbf{w} \geq 0; \\ & \xi - N \geq \max_{\mathbf{x},\mathbf{y},\mathbf{z}} \sum_{i,j,k} w_j^k (z_{ij}^k - \hat{y}_i^k x_{ij}) + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k \\ & \qquad - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \qquad \text{s.t.} \\ & \qquad 0 \leq y_i^k; \;\; \sum_k y_i^k = 1; \;\; y_{ij}^k \leq y_i^k; \;\; y_{ij}^k \leq y_j^k \\ & \qquad 0 \leq x_{ij} \leq 1; \;\; \sum_{i,j} x_{ij} + \hat{x}_{ij} - 2x_{ij}\hat{x}_{ij} \leq D \\ & \qquad z_{ij}^k \leq x_{ij}; \quad z_{ij}^k \leq y_i^k \end{aligned} \qquad \text{(Equation 3.6)}$$

The mathematical program in Equation 3.6 is not convex because of the bilinear terms and the nested maximization (similar to solving a bilevel Stackelberg game). Fortunately, we can use the strong duality property of linear programs to resolve both of these difficulties. The dual of the maximization linear program is a minimization linear program with the same optimal value as the primal problem. Therefore, we can replace the inner maximization with its dual minimization problem to obtain a single convex quadratic program that minimizes over $\mathbf{w}$, ξ, and the dual variables (not shown). A similar approach is used by Globerson and Roweis (2006). As long as this relaxed program has an integral optimum, it is equivalent to maximizing only over integral $\mathbf{x}$ and $\mathbf{y}$. Thus, the overall program will find optimal weights. Taskar et al. (2004a) prove that the inner maximization in a 2-class AMN always has an integral solution. We can prove a similar result for the adversarial AMN:

Theorem 1. *Equation Equation 3.5 has an integral optimum when* $\mathbf{w} \geq 0$ *and the number of classes is 2.*

Proof Sketch. The structure of our argument is to show that an integral optimum exists by taking an arbitrary adversarial AMN problem and constructing an equivalent AMN problem that has an integral solution. Since the two problems are equivalent, the original adversarial AMN must also have an integral solution. First, we use a Lagrange multiplier to incorporate the constraint $\Delta(\mathbf{x}, \hat{\mathbf{x}}) \leq D$ directly into the maximization. The extra term acts as a "per-change" penalty, which remains linear in $\mathbf{x}$. Minimizing over the Lagrange multiplier effectively adjusts this per-change penalty until there are at most D changes between $\mathbf{x}$ and $\hat{\mathbf{x}}$, but does not affect the integrality of the inner maximization. Next, we replace all $\mathbf{x}$ variables with equivalent variables $\mathbf{v}$. Assume that either $w_j^1 = 0$ or $w_j^2 = 0$, for all j. (If both are positive, then we can subtract the smaller value from both to obtain a new set of weights with the same optimum as before.) We define $\mathbf{v}$ as follows:

$$v_{ij}^1 = \begin{cases} x_{ij} & \text{if } w_j^1 > 0, \\ 1 - x_{ij} & \text{if } w_j^1 = 0. \end{cases}$$

$$v_{ij}^2 = 1 - v_{ij}^1$$

By construction:

$$\sum_{i,j,k} w_j^k x_{ij}(y_i^k - \hat{y}_i^k) = \sum_{i,j,k} w_j^k v_{ij}^k (y_i^k - \hat{y}_i^k)$$

Thus, we can replace the $\mathbf{x}$ variables with $\mathbf{v}$. Since the connections between the v_{ij}^k and corresponding y_i^k variables are all associative, this defines an AMN over variables $\{\mathbf{y}, \mathbf{v}\}$, which is guaranteed to have an integral solution when there are only two classes.

By translating $\mathbf{v}$ back into $\mathbf{x}$, we obtain a solution that is integral in both $\mathbf{x}$ and $\mathbf{y}$. □

A complete proof can be found in Appendix A.

Many extensions of our model are possible. One extension is to restrict the adversary to only changing certain features of certain objects. For example, in a web spam domain, we might assume that the adversary will only modify spam pages. We could also have different budgets for different types of changes, such as a separate budget for each web page, or even separate budgets for changing the title of a web page and changing its body. These are easily expressed by changing the definition of $\mathcal{X}'$ and adding the appropriate constraints to the quadratic program. Our model can also support higher-order cliques, as described by Taskar et al. (2004a), as long as they are associative. For simplicity, our exposition and experiments focus on the simpler case described above.

One important limitation of our model is that we do not allow edges to be added or removed by the adversary. While edges can be encoded as variables in the model, they result in non-associative potentials, since the presence of an edge is not associated with either class label. Instead, the presence of an edge increases the probability that the two linked nodes will have the same label. Handling the adversarial addition and removal of edges is an important area for future work, but will almost certainly be a non-convex problem.

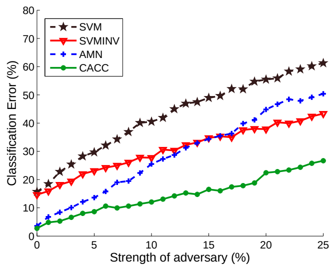
3.3. Experiments

In this section, we describe our experimental evaluation of CACC. Since CACC is both adversarial and relational, we compared it to four baselines: AMNs, which are relational but not adversarial; SVMInvar (Teo et al., 2008), which is adversarial but not relational; and SVMs with a linear kernel, which are neither. AMNs, SVMInvar, and SVMs can be seen as special cases of CACC: fixing the adversary's budget D to zero results in an AMN, fixing the edge weights w_e^k to zero results in SVMInvar, and doing both results in an SVM.

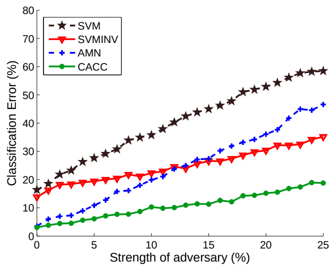
3.3.1. Datasets

We evaluated our method on three collective classification problems.

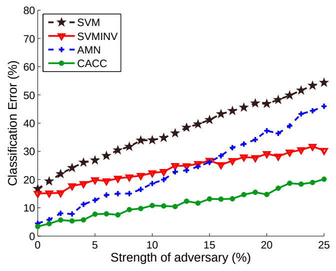
Synthetic. To evaluate the effectiveness of our method in a controlled setting where the distribution is known, we constructed a set of 10 random graphs, each with 100 nodes and 30 Boolean features. Of the 100 nodes, half had a positive label ('+') and half had a negative label ('$-$'). Nodes of the same class were more likely to be linked by an edge than nodes with different classes. The features were divided evenly into three types: positive, negative, and neutral. Half of the



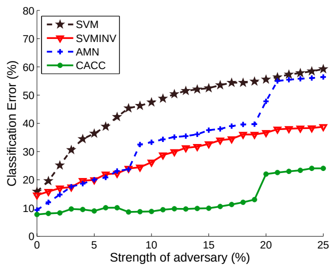
(a) Synthetic dataset: 0%



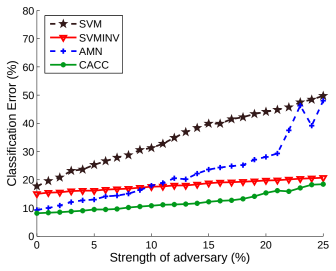
(b) Synthetic dataset: 10%



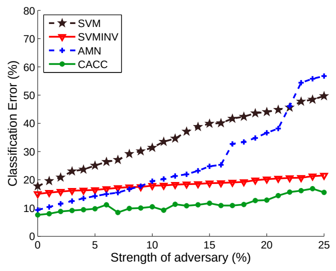
(c) Synthetic dataset: 20%



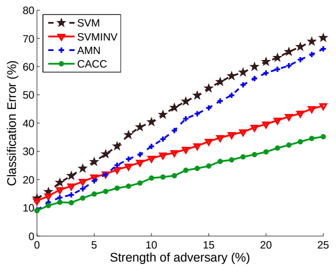
(d) Political blogs: 0%



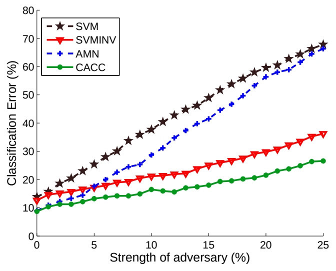
(e) Political blogs: 10%



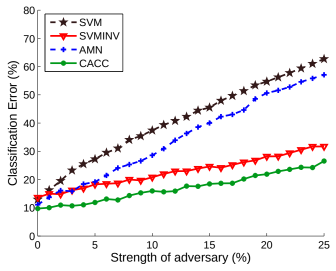
(f) Political blogs: 20%



(g) Reuters: 0%



(h) Reuters: 10%



(i) Reuters: 20%

FIGURE 3.2. Accuracy of different classifiers in presence of worst-case adversary. The number following the dataset name indicates the adversary's strength at the time of parameter tuning. The x-axis indicates the adversary's strength at test time. Smaller is better.

positive and negative nodes had different feature distributions based on their class; that is, the positive nodes had more positive attributes and the negative nodes had more negative attributes, on average. In such nodes, on average there are 6 words, one of which is of the opposite class's words, two words are consistent with the class label and three words are neutral. The other half of the nodes had an ambiguous distribution consisting mainly of the neutral words (on average one word is consistent with class label, one word is not consistent and 3 words are neutral). Therefore, an effective classifier for these graphs must rely on both the attributes and relations. On average, each node had 8 neighbors, 7 of which had the same class and 1 of which had a different class.

Political Blogs. Our second domain is based on the Political blogs dataset collected by Adamic and Glance (2005). The original dataset contains 1490 online blogs captured during the 2004 election cycle, their political affiliation (liberal or conservative), and their linking relationships to other blogs. We extended this dataset with word information from four different crawls at different dates in 2012: early February, late February, early May and late May. We used mutual information to select the 100 words that best predict the class label (Peng et al., 2005), only using blogs from February and half of the blogs in early May, in order to limit the influence of test labels on our training procedure. We found that some of the blogs in the original dataset were no longer active, and had been replaced by empty or spam web pages. We manually removed these from consideration. Finally, we partitioned the blogs into two disjoint subsets and removed all edges between nodes in the different subsets.

Reuters. As our third dataset, we prepared a Reuters dataset similar to the one used by Taskar et al. (2004a). We took the ModApte split of the Reuters-21578 corpus and selected articles from four classes: crude, grain, trade, and money-fx. We used the 200 words with highest mutual information as features. We linked each document to the two most similar documents based on TF-IDF weighted cosine distance. We split the data into 7 sets based on time, and performed the tuning and then the training phases based on this temporal order (as explained in 3.3.3.).

3.3.2. Simulating an adversary

In real world adversarial problems, the adversary does not usually have complete access to the model parameters. Researchers have widely studied the different ways that an adversary can acquire access to the model parameters actively or passively (Lowd and Meek, 2005b,a). In this section, we have examined two extreme cases. In the first, the adversary has complete access to the model parameters and manipulates the features to maximize the misclassification rate. Since exactly maximizing the error rate is typically NP-hard, our intelligent adversary instead maximizes the margin loss by solving the linear program in (Equation 3.5). In the second scenario, the random adversary randomly toggles D binary features, representing random noise or perhaps a very naïve adversary.

3.3.3. Methodology and metrics

In order to evaluate the robustness of these methods to malicious adversaries, we applied a simulated adversary to both the tuning data and the test data. We assumed the worst-case scenario, in which the adversary has perfect knowledge of the model parameters and only wants to maximize the error rate of the classifier. Since exactly maximizing the error rate is typically NP-hard, our intelligent adversary instead maximizes the margin loss by solving the linear program in Equation 3.5 for a fixed budget. Each model was attacked separately. On the validation data, we used adversarial budgets of 0% (no adversarial manipulation), 10%, and 20% of the total number of features present in the data. This allowed us to tune our models to "expect" adversaries of different strengths. Of course, we rarely know the exact strength of the adversary in advance. Thus, on the test data, we used budgets that ranged from 0% to 25%, in order to see how well different models did against adversaries that were weaker and stronger than expected.

We used the fraction of misclassified nodes as our primary evaluation criterion. For all methods, we tuned the regularization parameter C using held-out validation data. For the adversarial methods (CACC and SVMInvar), we tuned the adversarial training budget D as well. All parameters were selected to maximize performance on the tuning set with the given level of adversarial manipulation.

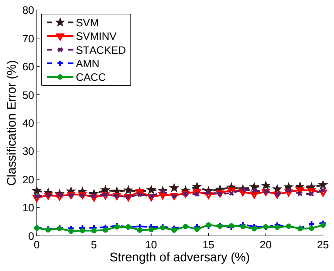
For political blogs, we tuned our parameters using the words from the February crawls, and then learned models on early May data and evaluated them on late May data. In this way, our tuning procedure could observe the concept drift within February and select parameters that would handle the concept drift during May well. For Synthetic data, we ran 10-fold cross validation. For Reuters, we split the data into 7 sets based on time. We tuned parameters using articles from time t and $t+1$ and then learned on articles at time $t+1$ and evaluated on articles from time $t+2$.

We used CPLEX to solve all quadratic and linear programming problems. Most problems were solved in less than 1 minute on a single core.

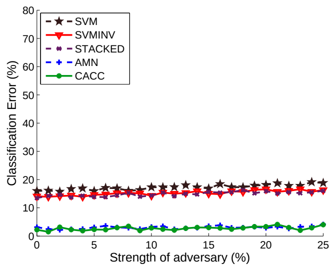
All of our code and datasets are available upon request.

3.3.4. Results and discussion

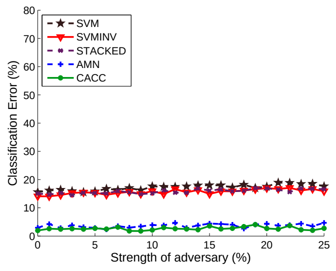
Figure 3.2 shows the performance of all four methods on test data manipulated by rational adversaries of varying strength (0%-25%), after being tuned against adversaries of different strengths (0%, 10%, and 20%). Lower is better. On the far left of each graph is performance without an adversary. To the right of each graph, the strength of the adversary increases.



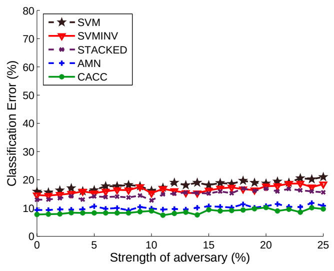
(a) Synthetic dataset: 0%



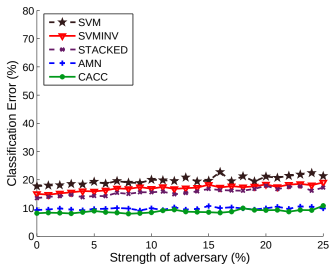
(b) Synthetic dataset: 10%



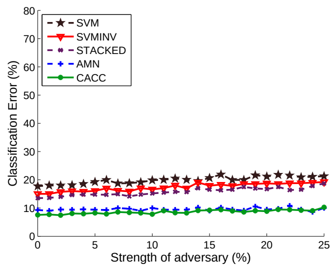
(c) Synthetic dataset: 20%



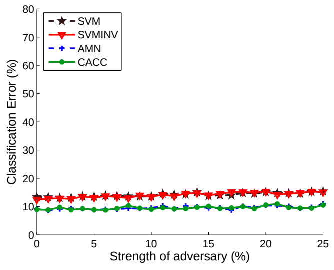
(d) Political blogs: 0%



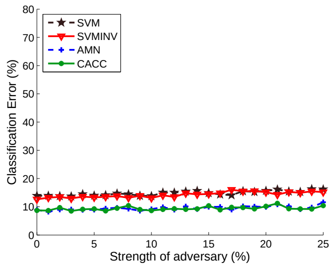
(e) Political blogs: 10%



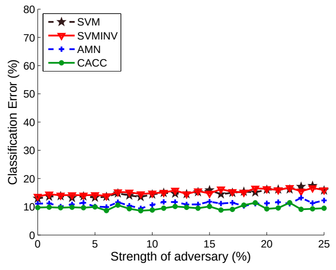
(f) Political blogs: 20%



(g) Reuters: 0%



(h) Reuters: 10%



(i) Reuters: 20%

FIGURE 3.3. Accuracy of different classifiers in presence of random adversary. We observe that even strong random attacks are not efficient in disguising the true class of the sample.

When a rational adversary is present, CACC clearly and consistently outperforms all other methods. When there is no adversary, its performance is similar to a regular AMN. On political blogs, it appears to be slightly better, which may be the result of the large amount of concept drift in that dataset.

As expected, tuning against stronger adversaries (10% and 20%) makes CACC more effective against stronger adversaries at test time. Surprisingly, tuning

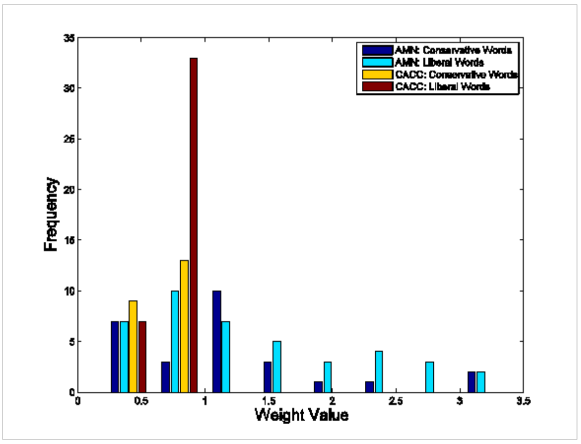


FIGURE 3.4. The distribution of the learned weight values for different models. The robust method tends to have a high density on the weights that are saturated.

against a stronger adversary does not significantly reduce performance against weaker adversaries: CACC remains nearly as effective against no adversary when tuned for a 20% adversary as when tuned for no adversary. Specifically, when there is no adversary at test time, the increase in error rate from training against a 20% adversary is less than 1% on Synthetic and Reuters, and on Political the error rate actually decreases slightly. Thus, this additional robustness comes at a very small cost.

In Figures 3.2d, 3.2e, and 3.2f, the AMN classification error jumps sharply as the adversary budget increases. This is the point when enough nodes are mis-classified that links are actively misleading in one or two of the eight cross-validation folds, leading to worse performance than the SVM for those folds. This demonstrates that relational classifiers are potentially more vulnerable to adversarial attacks than non-relational classifiers. A smoother version of this effect can also be observed on both the synthetic dataset and Reuters.

Another interesting result was that our solutions on Reuters were always integral, even though the number of classes is 4 and integrality is not guaranteed.

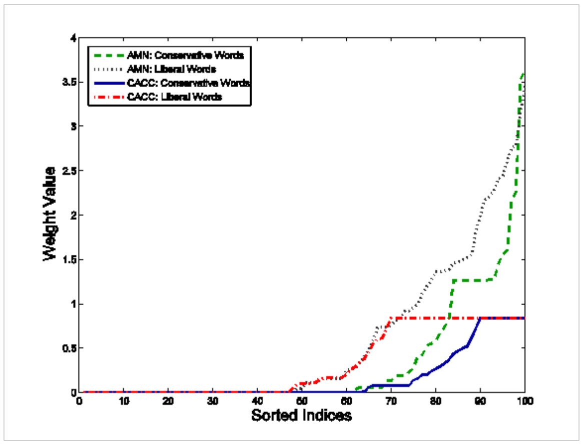


FIGURE 3.5. The sorted learned weights for each method. The robust method constrains the maximum value of the weights. This suggests that robustness could also be achieved through regularization with L_∞ norm.

An inspiring observation is about the distribution of learned weights in robust and non-robust models. The robust models have restricted the maximum value that the weight parameter can take Figure (3.5). Intuitively, this means that if the learner unconditionally trusts the importance of a certain feature, then it will become a point of weakness for itself. The adversarial budget in this experiment had been an L_1, therefore, from a technical point of view, this result suggests that we can achieve the same robustness by regularizing the weights by an L_∞ norm. This was the motivation of the work that we present in the next chapter.

We also performed additional experiments against irrational adversaries that modify attributes uniformly at random. These random attacks had little effect on the accuracy of any of the methods; all remained nearly as effective as against no adversary (Figure 3.3).

3.4. Conclusion

In this chapter, we provide a generalization of SVMInvar (Teo et al., 2008) and AMNs (Taskar et al., 2004a) that combines the robustness of SVMInvar with the ability to reason about interrelated objects. In experiments on real and synthetic data, CACC finds consistently effective and robust models, even when there are more than two labels.

In the next chapter, we extend robustness to adversarial manipulation of input data to generic structured prediction models. We show how robustness is equivalent to regularization for structured models, and we propose methods for developing customized regularization functions for particular adversarial uncertainty sets.

CHAPTER IV

EQUIVALENCY OF ADVERSARIAL ROBUSTNESS AND REGULARIZATION

This work was published in the proceedings of the thirty-first International Conference on Machine Learning (ICML 2014). I was the primary contributor to the methodology and writing, and designed and conducted the experiments. My Ph.D. advisor, Dr. Daniel Lowd contributed partly to the methodology and writing. Daniel Lowd was the principle investigator for this work.

Traditional machine learning methods assume that training and test data are drawn from the same distribution. However, in many real-world applications, the distribution is constantly changing. In some cases, such as spam filtering and fraud detection, an adversary may be actively manipulating it to defeat the learned model. In such cases, it is beneficial to optimize the model's performance on not just the training data but on the worst-case manipulation of the training data, where the manipulations are constrained to some domain-specific uncertainty set. For example, in an image classification problem, the uncertainty set could include minor translations, rotations, noise, or color shifts of the training data. This type of robust optimization leads to models that perform well on points that are "close" to those in the training data.

In general, robust optimization addresses optimization problems in which some degree of uncertainty governs the known parameters of the model Ben-Tal and Nemirovski (1998, 1999, 2000, 2001); Bertsimas and Sim (2004). In many of the existing robust formulations, robust linear programming is a central method. For example, Bertsimas et al. Bertsimas et al. (2004) show that when the disturbance of the inputs is restricted to an ellipsoid around the true values defined by some norm, then the robust linear programming problem can be reduced to a convex cone program. A number of other authors have explored the application of robust optimization to classification problems (e.g., Lanckriet et al. (2003); El Ghaoui et al. (2003); Bhattacharyya et al. (2004); Shivaswamy et al. (2006)). Recently, Xu et al. Xu et al. (2009) showed that regularization of support vector machines can be derived from a robust formulation. However, robustness for structured prediction models has remained largely unexplored.

In this chapter, we develop a general-purpose technique for learning robust structural support vector machines. Our basic approach is to consider the worst-case corruption of the input data within some uncertainty set and use this to define a robust formulation. This optimization problem is often much harder than standard training of structural SVMs when written directly; we overcome this obstacle by transforming the robust optimization problem into a standard structural support vector machine learning problem with an additional regularizer. This gives us both robustness and computational efficiency in the structured prediction setting, as well as establishing an elegant relationship between robustness and regularization for structural SVMs.

We demonstrate our approach on a new dataset consisting of snapshots of political blogs from 2003 through 2013, based on the political blogs dataset from Adamic and Glance (2005). Blogs are classified as liberal or conservative using both their words and link structure. To make this more challenging, we train on blogs from 2004 but evaluate on every year, from 2003 to 2013. In this domain, we define an uncertainty set, show to construct an appropriate regularizer, and show that this regularization can lead to substantially lower test error than a non-robust model.

4.1. Preliminaries

We begin by describing our notation and then provide a brief overview of structural support vector machines.

$\mathbf{x}$ and $\mathbf{y}$ denote the vectorized input and the representation of the structured output in the training data, respectively. For simplicity of notation, we assume a single training example, such as a single social network graph, but our results easily extend to a set of training examples.

The feature vector $\phi(\mathbf{x}, \mathbf{y})$ is a function of both inputs and labels (and also manipulated input or alternate labels, when used as the input argument). We use $\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}})$, to refer to the difference between two feature vectors with different labels $\mathbf{y}$ and $\tilde{\mathbf{y}}$; in particular: $\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) = \phi(\mathbf{x}, \tilde{\mathbf{y}}) - \phi(\mathbf{x}, \mathbf{y})$. The value of $\mathbf{w}^T\phi(\mathbf{x}, \tilde{\mathbf{y}})$ is called the *score* of labeling $\mathbf{x}$ as $\tilde{\mathbf{y}}$, for the given model weights $\mathbf{w}$.

$\Delta(\mathbf{y}, \tilde{\mathbf{y}})$ is a scalar distance function, such as Hamming distance, which is a measure of dissimilarity between the true and alternate labels.

We use $\|.\|$ to refer to a general norm function and $\|.\|^*$ for the dual norm of $\|.\|$, where $\|\mathbf{y}\|^* = \sup\{\mathbf{y}^T\mathbf{x} | \|\mathbf{x}\| \leq 1\}$.

In this chapter, we focus on the derivation of robust formulations for 1-slack structural SVM Joachims et al. (2009). (With minor changes, the results of this chapter can be applied to n-slack structural SVMs as well, but we skip them here.) The optimization program of a 1-slack structural SVM is:

$$\underset{\mathbf{w},\zeta}{\text{minimize}}\ f(\mathbf{w}) + C\zeta \quad \text{subject to} \qquad \text{(Equation 4.1)}$$
$$\zeta \geq \max_{\tilde{\mathbf{y}}}\ \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}})$$

where $\mathbf{x}$ is the vector of all input variables, $\mathbf{y}$ is the desired structured query variables, and $\mathbf{w}$ is the vector of the model parameters. The goal is to learn $\mathbf{w}$.

$f(\mathbf{w})$ is a regularization function that penalizes "large" weights. Depending on the application, $f(\mathbf{w})$ can be any convex function in general. Semi-homogeneous functions, such as norms or powers of norms with power value equal to or greater than 1, are among the favorite choices. (A function $f(z)$ is semi-homogeneous if and only if $f(az) = a^{\alpha} f(z)$ for some positive α.) $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is the most commonly used regularization function.

4.2. Robust structural SVMs

In this section, we motivate and define a robust formulation of structural SVMs. We begin by considering how an adversary might modify an input in order to maximize the prediction error, and use this to derive a definition of a robust structural SVM in sample space and feature space.

4.2.1. Worst-case/Adversarial data manipulation

Adversaries might have a wide range of goals, but in the worst case they will antagonistically try to reduce the accuracy of the predictive model. For structural SVMs, the predicted output is chosen by solving $\tilde{\mathbf{y}} = \arg\max_{\tilde{\mathbf{y}}} \mathbf{w}^T \phi(\mathbf{x}, \tilde{\mathbf{y}})$, where $\mathbf{w}^T \phi(\mathbf{x}, \tilde{\mathbf{y}})$ is the classification score. Thus, an adversary's antagonistic goal would be to replace the true input $\mathbf{x}$ with a manipulated version $\tilde{\mathbf{x}}$ that maximizes the classification loss $\Delta(\mathbf{y}, \tilde{\mathbf{y}})$. If the highest scoring label is not unique, we assume the adversary tries to maximize the minimum loss in the set:

$$\begin{aligned} &\underset{\tilde{\mathbf{x}}}{\text{maximize}}\ \min_{\tilde{\mathbf{y}}}\ \Delta(\mathbf{y}, \tilde{\mathbf{y}}), \quad \text{subject to} \\ &\tilde{\mathbf{y}} \in \arg\max_{\tilde{\mathbf{y}} \neq \mathbf{y}} \mathbf{w}^T \phi(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \\ &\tilde{\mathbf{x}} \in \mathcal{S}(\mathbf{x}, \mathbf{y}) \end{aligned} \qquad \text{(Equation 4.2)}$$

$\mathcal{S}(\mathbf{x}, \mathbf{y})$ is a domain-specific *uncertainty set*, which constrains the set of possible corrupt inputs $\tilde{\mathbf{x}}$. We always assume that $\mathbf{x} \in \mathcal{S}(\mathbf{x}, \mathbf{y})$, which means $\mathbf{x}$ can remain unchanged. The set $\mathcal{S}(\mathbf{x}, \mathbf{y})$ can contain a wide range of possible variations, such as the amount of affordable/possible change in an attribute, or the restrictions that are enforced on combinations of changes among several attributes.

The bi-level optimization program in (Equation 4.2) is not tractable in general, especially when $\mathbf{x}$ and $\mathbf{y}$ have integer components. A slightly more tractable solution is to relax the program and only require that $\tilde{\mathbf{y}}$ be scored higher than the true output $\mathbf{y}$:

$$\begin{aligned} &\underset{\tilde{\mathbf{x}}, \tilde{\mathbf{y}}}{\text{maximize}}\ \Delta(\mathbf{y}, \tilde{\mathbf{y}}), \qquad \text{subject to} \\ &\mathbf{w}^T \phi(\tilde{\mathbf{x}}, \mathbf{y}) \leq \mathbf{w}^T \phi(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \\ &\tilde{\mathbf{x}} \in \mathcal{S}(\mathbf{x}, \mathbf{y}) \end{aligned} \qquad \text{(Equation 4.3)}$$

The maximization in (Equation 4.3) might be infeasible, but its Lagrangian relaxation is always feasible:

$$\begin{aligned} &\underset{\tilde{\mathbf{x}}, \tilde{\mathbf{y}}}{\text{maximize}}\ \lambda \mathbf{w}^T \boldsymbol{\Delta}\phi(\tilde{\mathbf{x}}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \\ &\text{subject to} \qquad \tilde{\mathbf{x}} \in \mathcal{S}(\mathbf{x}, \mathbf{y}) \end{aligned} \qquad \text{(Equation 4.4)}$$

We want to attract the reader's attention to the similarity of (Equation 4.4), and the nested max operation in the constraint of (Equation 4.1). In fact, $\lambda\mathbf{w}^T\mathbf{\Delta}\phi(\tilde{\mathbf{x}},\mathbf{y},\tilde{\mathbf{y}}) + \Delta(\mathbf{y},\tilde{\mathbf{y}})$ is a component of the loss function that the learner wants to minimize. In the next subsection, we reformulate the standard 1-slack structural SVM so that the effect of adversarial manipulation of input data will be minimized.

4.2.2. Robust formulation in sample space

Our goal is to find a set of model parameters that perform well against the worst-case manipulated input $\tilde{\mathbf{x}}$ in the uncertainty set. We formulate this by replacing the loss-augmented margin in (Equation 4.1) with the worst-case adversarial loss obtained by (Equation 4.4):

$$\underset{\mathbf{w}}{\text{minimize}}\; Cf(\mathbf{w}) + \sup_{\tilde{\mathbf{x}}\in\mathcal{S}(\mathbf{x},\mathbf{y}),\tilde{\mathbf{y}}} \mathcal{L}_\lambda(\mathbf{w},\tilde{\mathbf{x}},\tilde{\mathbf{y}},\mathbf{y}) \qquad \text{(Equation 4.5)}$$

where $\mathcal{L}_\lambda(\mathbf{w},\tilde{\mathbf{x}},\tilde{\mathbf{y}},\mathbf{y}) = \lambda\mathbf{w}^T\mathbf{\Delta}\phi(\tilde{\mathbf{x}},\mathbf{y},\tilde{\mathbf{y}}) + \Delta(\mathbf{y},\tilde{\mathbf{y}})$. We replace the maximization with a sup operator to indicate that the maximum value might not be achieved. Both λ and C are tunable parameters that can be determined by cross-validation. In the following lemma we show that it is possible to tune only one of them by performing a re-parameterization.

Lemma 1. *For semi-homogeneous $f(.)$, the problem (Equation 4.5) can be equivalently re-written in the following form:*

$$\underset{\mathbf{w}}{\text{minimize}}\; Cf(\mathbf{w}) + \sup_{\tilde{\mathbf{x}}\in\mathcal{S}(\mathbf{x},\mathbf{y}),\tilde{\mathbf{y}}} \mathcal{L}(\mathbf{w},\tilde{\mathbf{x}},\tilde{\mathbf{y}},\mathbf{y}) \qquad \text{(Equation 4.6)}$$

where $\mathcal{L}(\mathbf{w},\tilde{\mathbf{x}},\tilde{\mathbf{y}},\mathbf{y}) = \mathbf{w}^T\mathbf{\Delta}\phi(\tilde{\mathbf{x}},\mathbf{y},\tilde{\mathbf{y}}) + \Delta(\mathbf{y},\tilde{\mathbf{y}})$

Proof. Let $\mathbf{w}' = \lambda\mathbf{w}$, and $C' = \frac{C}{\lambda^\alpha}$. Then, for a semi-homogeneous $f(.)$, where $f(a\mathbf{w}) = a^\alpha f(\mathbf{w})$, we have $Cf(\mathbf{w}) = \frac{C}{\lambda^\alpha} f(\lambda\mathbf{w})$. Therefore, for semi-homogeneous regularization functions $f(.)$ by re-parameterization of $\mathbf{w}'$ as $\mathbf{w}$, and C' as C, (Equation 4.5) can be rewritten as (Equation 4.6). □

Problem (Equation 4.6) is similar in form to a standard structural SVM, except that the inner maximization is done over both $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$. This is potentially much harder than simply maximizing over $\tilde{\mathbf{y}}$, since the input often has a much higher dimension than the output. For example, when labeling a set of 1000 web pages, there are only 1000 labels to predict but 1,000,000 possible hyperlinks that the adversary could add or remove. In the next subsection, we show that we can avoid the above-mentioned computational complexity by instead restricting the variations in the feature space.

4.2.3. Robustness in feature space

Let $\mathbf{\Delta x}$ be the disturbance in the sample space such that: $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{\Delta x}$. Then, by finite difference approximation[1]:

$$\begin{aligned} \phi(\tilde{\mathbf{x}}, \mathbf{y}) &= \phi(\mathbf{x} + \mathbf{\Delta x}, \mathbf{y}) = \phi(\mathbf{x}, \mathbf{y}) + \delta(\tilde{\mathbf{x}}, \mathbf{y}) \\ \phi(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) &= \phi(\mathbf{x} + \mathbf{\Delta x}, \tilde{\mathbf{y}}) = \phi(\mathbf{x}, \tilde{\mathbf{y}}) + \delta(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \end{aligned}$$

Note that we are not introducing any error; both functions $\delta(\tilde{\mathbf{x}}, \mathbf{y})$ and $\delta(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ contain as many high-order approximation terms as needed for achieving infinitesimal error introduction, although we never unpack these functions. In fact, the difference between $\delta(\tilde{\mathbf{x}}, \mathbf{y})$ and $\delta(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is particularly important; let

[1]For more on finite difference approximations, refer to Smith Smith (1985).

$\delta_{\tilde{\mathbf{y}}}(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}}) = \delta(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \delta(\tilde{\mathbf{x}}, \mathbf{y})$, then:

$$\begin{aligned} & \phi(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \phi(\tilde{\mathbf{x}}, \mathbf{y}) \\ = \;& \phi(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}, \tilde{\mathbf{y}}) - \phi(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}, \mathbf{y}) \\ = \;& \phi(\mathbf{x}, \tilde{\mathbf{y}}) - \phi(\mathbf{x}, \mathbf{y}) + \delta(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \delta(\tilde{\mathbf{x}}, \mathbf{y}) \\ = \;& \phi(\mathbf{x}, \tilde{\mathbf{y}}) - \phi(\mathbf{x}, \mathbf{y}) + \delta_{\tilde{\mathbf{y}}}(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}}) \qquad \text{(Equation 4.7)} \end{aligned}$$

Therefore, the manipulation of the input data affects the margin $\mathcal{L}(.)$ in (Equation 4.6) through $\delta_{\tilde{\mathbf{y}}}(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}})$. In the rest of the chapter, we will use δ^i to refer to the ith element of the vector $\delta_{\tilde{\mathbf{y}}}(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}})$.

Clearly, $\delta_{\tilde{\mathbf{y}}}$ depends on the specific choice of the alternate labeling $\tilde{\mathbf{y}}$, as well as $\tilde{\mathbf{x}}$, $\mathbf{x}$, and $\mathbf{y}$. Let:

$$\Delta^2\Phi(\mathbf{x}, \mathbf{y}) = \{\delta = \delta_{\tilde{\mathbf{y}}}(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}}) |\ \forall \tilde{\mathbf{x}} \in \mathcal{S}(\mathbf{x}, \mathbf{y}), \tilde{\mathbf{y}}\} \qquad \text{(Equation 4.8)}$$

be the set of all possible variations. Note that $\Delta^2\Phi(\mathbf{x}, \mathbf{y})$ is independent of $\tilde{\mathbf{y}}$. In the next section, we introduce some mechanical procedures for calculating $\Delta^2\Phi(\mathbf{x}, \mathbf{y})$ from $\mathcal{S}(\mathbf{x}, \mathbf{y})$, for certain choices of $\mathcal{S}(\mathbf{x}, \mathbf{y})$ and $\phi(\mathbf{x}, \mathbf{y})$.

Lemma 2. *Let* $\mathcal{L}_1(\mathbf{w}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \mathbf{y}) = \mathbf{w}^T(\phi(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \phi(\tilde{\mathbf{x}}, \mathbf{y})) + \Delta(\mathbf{y}, \tilde{\mathbf{y}})$*, and* $\mathcal{L}_2(\mathbf{w}, \delta, \tilde{\mathbf{y}}) = \mathbf{w}^T(\phi(\mathbf{x}, \tilde{\mathbf{y}}) - \phi(\mathbf{x}, \mathbf{y}) + \delta) + \Delta(\mathbf{y}, \tilde{\mathbf{y}})$. *Then we will have:*

$$\sup_{\delta \in \Delta^2\Phi(\mathbf{x}, \mathbf{y}), \tilde{\mathbf{y}}} \mathcal{L}_2(\mathbf{w}, \delta, \tilde{\mathbf{y}}) \geq \sup_{\tilde{\mathbf{x}} \in \mathcal{S}(\mathbf{x}, \mathbf{y}), \tilde{\mathbf{y}}} \mathcal{L}_1(\mathbf{w}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \mathbf{y})$$

Proof sketch. The left-hand side of the inequality is equal to the right-hand side except that the supremum is taken over a superset of function values. Thus, the left-hand side cannot be any less than the right-hand side. $\square$

Now, we can rewrite the robust formulation in (Equation 4.6) over variations in the feature space:

$$\underset{\mathbf{w}}{\text{minimize}}\ Cf(\mathbf{w}) + \sup_{\delta \in \Delta^2\Phi(\mathbf{x},\mathbf{y}),\tilde{\mathbf{y}}} \mathcal{L}(\mathbf{w},\delta,\tilde{\mathbf{y}}) \qquad \text{(Equation 4.9)}$$

where $\mathcal{L}(\mathbf{w},\delta,\tilde{\mathbf{y}}) = \mathbf{w}^T(\boldsymbol{\Delta}\phi(\mathbf{x},\mathbf{y},\tilde{\mathbf{y}}) + \delta) + \Delta(\mathbf{y},\tilde{\mathbf{y}})$.

By Lemma (2), the objective of (Equation 4.9) is an upper-bound for the objective of (Equation 4.6); therefore, the formulation of the problem in (Equation 4.9) is an approximate, but more tractable, solution for (Equation 4.6).

In the next section, we will show that for a wide class of $\Delta^2\Phi(\mathbf{x},\mathbf{y})$'s, problem (Equation 4.9) reduces to an optimization program which can be solved as efficiently as an ordinary 1-slack structural SVM.

4.3. Mapping the uncertainty sets

In many real world problems, there exists some expert knowledge about the uncertainty sets in the sample space. For example, for the webpage classification problem, a spammer can modify web pages by adding and removing words and links, but is constrained by the cost of compromising legitimate web pages, which takes time and effort, or obfuscating spam pages, which may make them less effective at gaining clicks. We can approximate this with a simple budget on the number of words and links the adversary can change over the entire dataset. Even when such information is not readily available, it may be possible to infer an uncertainty set from training data. For example, if our dataset contains outliers, we can pair each outlier ($\tilde{\mathbf{x}}$) with the most similar non-outlier ($\mathbf{x}$) and take the differences as possible directions of manipulation: $\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x}$. The convex hull of these difference vectors (or an approximation thereof) can be used to define an uncertainty set for any instance.

Lemma 2 states that the robust formulation in feature space is a reasonable approximation for the robust formulation in the sample space, but it does not suggest any mechanical procedure for calculating the uncertainty sets in feature space from the ones in the sample space. We now derive such procedures for certain types of uncertainty sets and feature functions.

Many features of interest, including logical conjunctions, can be represented as products of several variables. We define a *multinomial feature function* as a sum of many such products:

$$\phi_{\mathcal{C}}(\mathbf{x}, \mathbf{y}) = \sum_{(c_x, c_y) \in \mathcal{C}} \prod_{i \in c_x} \mathbf{x}_i \prod_{i \in c_y} \mathbf{y}_i \qquad \text{(Equation 4.10)}$$

where $\mathcal{C}$ is a set of cliques and (c_x, c_y) are the index sets of the attribute and output variables that contribute to the feature. The summation groups together many products that share the same pattern into a single, aggregate feature so that they may be considered collectively. For example, in web page classification, the multinomial feature $\sum_i x_{i,j} y_i$ could represent the number of web pages with label 1 that contain word j. This is equivalent to having many features with tied weights.

Lemma 3. *If the feature function $\phi_{\mathcal{C}}(\mathbf{x}, \mathbf{y})$ is multinomial with $\mathbf{0} \leq \mathbf{x}, \mathbf{y} \leq \mathbf{1}$; then, its disturbance $\delta^{\mathcal{C}}$ can be upper-bounded by a function of the variations in the* *sample space, and the following inequality relation holds:*

$$\frac{|\delta^{\mathcal{C}}|^p}{\alpha_{\mathcal{C}}|\mathcal{C}|^{\frac{p}{q}}} \leq \sum_{c_x \in \mathcal{C}} \sum_{i \in c_x} |\tilde{\mathbf{x}}_i - \mathbf{x}_i|^p \qquad \text{(Equation 4.11)}$$

where $p \geq 1$ is an arbitrary power value and $\frac{1}{p} + \frac{1}{q} = 1$; $\alpha = \max_{c_x \in \mathcal{C}} |c_x|^{(p-1)}$; $|c_x|$ is the number of evidence variables in c_x; and $|\mathcal{C}|$ is the number of different sets c_x in $\mathcal{C}$.

Now, the resulting inequality of Lemma 3 can be used as the core inequality for upper-bounding the variations in the feature space.

The proofs can be found in Appendix B.

The next theorem is the main result of this section.

Theorem 2. *For multinomial feature functions and spherical uncertainty sets in the sample space$\mathcal{S}(\mathbf{x}, \mathbf{y}) = \{\tilde{\mathbf{x}} \mid \|\tilde{\mathbf{x}} - \mathbf{x}\|_p \leq B, p \geq 1\}$, one can construct an ellipsoidal uncertainty set in the feature space:*

$$\Delta^2\Phi(\mathbf{x}, \mathbf{y}) = \{\delta | \quad \|\mathbf{M}\delta\|_p \leq 1\} \qquad \text{(Equation 4.12)}$$

where $\mathbf{M}$ is a diagonal matrix with $\frac{1}{B(d\alpha_i)^{\frac{1}{p}}|\mathcal{C}_i|^{\frac{1}{q}}}$ on the (i,i)th position. d, α_i, and $|\mathcal{C}_i|$ are appropriate constants.

Proof. Assume that $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_L\}$ is a set of cliques that covers all variable $\mathbf{x}_i$'s. Note that such a set should exist; otherwise, some variables are never used in the model. For each of the cliques, we form a corresponding difference in the feature function from Equation 4.7, and apply Lemma 3. By adding all of the resulting inequalities, we obtain:

$$\begin{aligned}
\sum_{\mathcal{C}_i \in \mathcal{P}} \frac{|\delta^{\mathcal{C}_i}|^p}{\alpha_i |\mathcal{C}_i|^{\frac{p}{q}}} &\leq d \sum_{i=1}^{\dim(\mathbf{x})} |\tilde{\mathbf{x}}_i - \mathbf{x}_i|^p \\
&= d\|\tilde{\mathbf{x}} - \mathbf{x}\|_p^p \leq dB^p \\
\Rightarrow \quad \sum_{\mathcal{C}_i \in \mathcal{P}} \frac{|\delta^{\mathcal{C}_i}|^p}{B^p d\alpha_i |\mathcal{C}_i|^{\frac{p}{q}}} &\leq 1 \\
\Rightarrow \quad \sum_{\mathcal{C}_i \in \mathcal{P}} \left(\frac{1}{B(d\alpha_i)^{\frac{1}{p}} |\mathcal{C}_i|^{\frac{1}{q}}} |\delta^{\mathcal{C}_i}| \right)^p &\leq 1
\end{aligned}$$

where $\alpha_i = \max_{c_x \in \mathcal{C}_i} |c_x|^{(p-1)}$, and $|c_x|$ is the number of variables in c_x. Since it is possible that cliques cover overlapping sets of variables, the coefficient $d \geq 1$ will be used to maintain the inequality.

Now let $\frac{1}{B(d\alpha_i)^{\frac{1}{p}}|\mathcal{C}_i|^{\frac{1}{q}}}$ be the diagonal entry in matrix $\mathbf{M}$ that corresponds to feature disturbance $\delta^{\mathcal{C}_i}$. For this choice of $\mathbf{M}$, $\|\mathbf{M}\delta\|_p \leq 1$. $\square$

We have an example of applying Theorem 2 in Section 6.2, which will show how this construction works in practice.

Corollary 1. *If* $\mathcal{S}(\mathbf{x}, \mathbf{y}) = \{\tilde{\mathbf{x}} \mid \|\tilde{\mathbf{x}} - \mathbf{x}\|_1 \leq B\}$*, then* $\mathbf{M}$ *can be constructed by setting* $\frac{1}{Bd}$ *as its* (i,i)*th element, which results in a tighter upper bound.*

The proof can be found in Appendix B.

4.4. Robust optimization programs

Our main contribution in this chapter is achieving robust formulations that can be efficiently solved. We do this by demonstrating a connection between robustness to certain perturbations in feature space and certain types of weight regularization. In this section we derive formulations for achieving robust weight learning in structural SVMs when $\Delta^2\Phi(\mathbf{x}, \mathbf{y})$ is an ellipsoid, a polyhedron, or the intersection of an ellipsoid and a polyhedron.

4.4.1. Ellipsoidal constrained uncertainty

We first consider the case when the uncertainty set $\Delta^2\Phi(\mathbf{x}, \mathbf{y})$ is ellipsoidal. Recall that any ellipsoid can be represented in the form of $\{\mathbf{t} \,|\, \|\mathbf{M}\mathbf{t}\| \leq 1\}$, where $\|.\|$ is the relevant norm.

Theorem 3. *For $\Delta^2\Phi(\mathbf{x}, \mathbf{y}) = \{\delta \,|\, \|\mathbf{M}\delta\| \leq 1\}$ where $\mathbf{M}$ is positive definite, the optimization program of the robust structural SVM in (Equation 4.9) reduces to the following regularized formulation of the ordinary 1-slack structural SVM:*

$$\underset{\mathbf{w},\zeta}{\text{minimize}} \; Cf(\mathbf{w}) + \|\mathbf{M}^{-1}\mathbf{w}\|^* + \zeta \qquad \text{(Equation 4.13)}$$

$$\text{subject to}$$

$$\zeta \geq \sup_{\tilde{\mathbf{y}}} \; \mathbf{w}^T \boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}})$$

where $\|.\|^*$ is the dual norm of $\|.\|$.

Proof. We begin with the robust formulation of a structural SVM from (Equation 4.9), where the uncertainty set of δ is defined by the ellipsoid $\|\mathbf{M}\delta\| \leq 1$:

$$\underset{\mathbf{w}}{\text{minimize}} \; Cf(\mathbf{w}) + \sup_{\|\mathbf{M}\delta\| \leq 1, \tilde{\mathbf{y}}} \mathcal{L}(\mathbf{w}, \delta, \tilde{\mathbf{y}})$$

Let $\nu = \mathbf{M}\delta$, so that $\delta = \mathbf{M}^{-1}\nu$. Then we will have:

$$\begin{aligned}
& \sup_{\|\mathbf{M}\delta\|\leq 1, \tilde{\mathbf{y}}} \mathcal{L}(\mathbf{w}, \delta, \tilde{\mathbf{y}}) \\
= & \sup_{\|\mathbf{M}\delta\|\leq 1, \tilde{\mathbf{y}}} \mathbf{w}^T(\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \delta) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \\
= & \sup_{\|\mathbf{M}\delta\|\leq 1} \mathbf{w}^T\delta + \sup_{\tilde{\mathbf{y}}} \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \\
= & \sup_{\|\nu\|\leq 1} \mathbf{w}^T\mathbf{M}^{-1}\nu + \sup_{\tilde{\mathbf{y}}} \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}})
\end{aligned}$$

By definition of the dual norm, $\sup_{\|\nu\|\leq 1}(\mathbf{w}^T\mathbf{M}^{-1})\nu = \|\mathbf{M}^{-T}\mathbf{w}\|^*$. Since $\mathbf{M}^{-1}$ is also a definite matrix, it is symmetric; therefore, $\|\mathbf{M}^{-T}\mathbf{w}\|^* = \|\mathbf{M}^{-1}\mathbf{w}\|^*$.

$$= \|\mathbf{M}^{-1}\mathbf{w}\|^* + \sup_{\tilde{\mathbf{y}}} \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}})$$

By substitution, the rest of the proof is straightforward. □

Note that Theorem 3 can still be applied when M is not positive definite by using the Moore-Penrose inverse of M instead of the regular inverse. The result in Theorem 3 uses the technique of robust linear programming with arbitrary norms that is introduced in Bertsimas et al. (2004). This theorem can also be seen as a generalization of Theorem 3 in Xu et al. (2009) to structural SVMs. Theorem 3 shows the direct connection between the robust formulation and regularization of the non-robust formulation for structural SVMs.

Corollary 2. *For disturbances of the form $\|\delta\| \leq B$ in the feature space, with B being a maximum budget for the applicable changes and $\|.\|$ being an arbitrary norm, robustness can be achieved by adding the regularization function $B\|\mathbf{w}\|^*$ to the objective.*

Proof. Since $\|\delta\|/B \leq 1 \Rightarrow \|\frac{1}{B}\delta\| = \|\frac{1}{B}\mathbf{I}\delta\| \leq 1$. Let $\mathbf{M} = \frac{1}{B}\mathbf{I}$, then $\mathbf{M^{-1}} = B\mathbf{I}$. Thus, $\|\mathbf{M}^{-1}\mathbf{w}\|^* = \|B\mathbf{I}\mathbf{w}\|^* = B\|\mathbf{w}\|^*$. By Theorem 3, $B\|\mathbf{w}\|^*$ is the appropriate regularization function. □

Note that $\mathbf{M}$ can also be seen as a tuning parameter. In particular, if there is a low-dimensional representation of $\mathbf{M}$, then tuning $\mathbf{M}$ might be an option.

The commonly used L_2 regularization can be in fact interpreted as a regularization function that enforces robustness to disturbances in the feature space that are restricted to a hypersphere.

Corollary 3. *If $f(\mathbf{w}) = 0$, then setting $\mathbf{M} = \frac{1}{C}I$ and $\|.\| = \|.\|_2$ will recover the commonly used L_2-regularized structural SVM.*

Proof. If $\mathbf{M} = \frac{1}{C}I$, then $\mathbf{M^{-1}} = CI$. Note that the L_2 norm is dual to itself. Therefore, $f(\mathbf{w}) + \|\mathbf{M}^{-1}\mathbf{w}\|_2^* = 0 + \|CI\mathbf{w}\|_2 = C\|\mathbf{w}\|_2$. □

Corollary 4. *Robustness to variations restricted by a Mahalanobis norm $\|\delta\|_\mathbf{S} = \sqrt{\delta^T\mathbf{S}\delta} \leq 1$, where $\mathbf{S}$ is positive definite, is equivalent to adding the regularization function $\|\mathbf{w}\|_{\mathbf{S}^{-1}} = \sqrt{\mathbf{w}^T\mathbf{S}^{-1}\mathbf{w}}$ to the objective.*

Proof. Let $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ be the spectral decomposition of $\mathbf{S}$. Set $\mathbf{M} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}^T$ and the norm $\|.\|$ to $\|.\|_2$. Then $\|\mathbf{M}\delta\|_2 = \sqrt{\delta^T\mathbf{M}^T\mathbf{M}\delta} = \sqrt{\delta^T\mathbf{M}^2\delta} = \sqrt{\delta^T\mathbf{S}\delta}$. Therefore the resulting regularization function will be $\|\mathbf{M}^{-1}\mathbf{w}\|_2^* = \|\mathbf{M}^{-1}\mathbf{w}\|_2 = \sqrt{\mathbf{w}^T\mathbf{M}^{-T}\mathbf{M}^{-1}\mathbf{w}} = \sqrt{\mathbf{w}^T\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{w}} = \sqrt{\mathbf{w}^T\mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^T\mathbf{w}} = \sqrt{\mathbf{w}^T\mathbf{S}^{-1}\mathbf{w}} = \|\mathbf{w}\|_{\mathbf{S}^{-1}}$, Note that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ because $\mathbf{U}$ is a unitary matrix. □

4.4.2. Polyhedral constrained uncertainty

For some problems, an ellipsoid may not be a good representation of the uncertainty set, but almost any convex uncertainty set can be approximated by a polyhedron. In this subsection we consider the situations in which we are aware of the shape of the polyhedral constraints on the variations in the feature space; i.e., $\Delta^2\Phi(\mathbf{x},\mathbf{y}) = \{\delta|\mathbf{A}\delta \leq \mathbf{b}\}$. The next theorem shows that polyhedral uncertainty sets are equivalent to linear regularization in a transformed feature space. We begin with a supporting lemma.

Lemma 4. *If $\mathbf{x} \in \mathcal{S}(\mathbf{x},\mathbf{y})$, then for the corresponding $\Delta^2\Phi(\mathbf{x},\mathbf{y}) = \{\delta|\mathbf{A}\delta \leq \mathbf{b}\}$, $\mathbf{b}$ is a non-negative vector.*

Proof. $\mathbf{x} \in \mathcal{S}(\mathbf{x},\mathbf{y})$, and $\phi(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) - \phi(\tilde{\mathbf{x}},\mathbf{y}) = \phi(\mathbf{x},\tilde{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}) + \delta$. Therefore, when $\tilde{\mathbf{x}} = \mathbf{x}$ then $\delta = \mathbf{0}$, so we should have $\mathbf{0} \in \Delta^2\Phi(\mathbf{x},\mathbf{y})$. Therefore, for $\delta = \mathbf{0}$, $\mathbf{A}\delta = \mathbf{A0} \leq \mathbf{b}$; i.e., $\mathbf{b} \geq \mathbf{0}$. □

Theorem 4. *For $\Delta^2\Phi(\mathbf{x},\mathbf{y}) = \{\delta|\mathbf{A}\delta \leq \mathbf{b}\}$, the optimization program of the robust structural SVM in (Equation 4.9) reduces to the following ordinary 1-slack structural SVM*

$$\begin{aligned} &\underset{\lambda\geq 0,\zeta}{\text{minimize}} \;\; Cf(\mathbf{A}^T\lambda) + \lambda^T\mathbf{b} + \zeta \\ &\text{subject to} \;\; \zeta \geq \sup_{\tilde{\mathbf{y}}} \lambda^T\mathbf{A}\boldsymbol{\Delta}\phi(\mathbf{x},\mathbf{y},\tilde{\mathbf{y}}) + \Delta(\mathbf{y},\tilde{\mathbf{y}}) \end{aligned} \qquad \text{(Equation 4.14)}$$

Proof. By substituting the uncertainty set $\Delta^2\Phi(\mathbf{x},\mathbf{y}) = \{\delta|\mathbf{A}\delta \leq \mathbf{b}\}$ into the optimization program (Equation 4.9), we obtain:

$$\underset{\mathbf{w}\geq 0}{\text{minimize}} \;\; Cf(\mathbf{w}) + \sup_{\mathbf{A}\delta\leq\mathbf{b},\tilde{\mathbf{y}}} \mathcal{L}(\mathbf{w},\delta,\tilde{\mathbf{y}}) \qquad \text{(Equation 4.15)}$$

We can rewrite $\sup_{\mathbf{A}\delta \leq \mathbf{b}, \tilde{\mathbf{y}}} \mathcal{L}(\mathbf{w}, \delta, \tilde{\mathbf{y}})$ as:

$$\begin{aligned} & \sup_{\mathbf{A}\delta \leq \mathbf{b}, \tilde{\mathbf{y}}} \mathbf{w}^T(\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \delta) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \\ = & \sup_{\mathbf{A}\delta \leq \mathbf{b}} \mathbf{w}^T\delta + \sup_{\tilde{\mathbf{y}}} \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \end{aligned}$$

We perform a Lagrangian relaxation on $\mathbf{A}\delta \leq \mathbf{b}$:

$$\begin{aligned} = & \inf_{\lambda \geq 0} \sup_{\delta} (\mathbf{w}^T\delta - \lambda^T\mathbf{A}\delta + \lambda^T\mathbf{b}) \\ & + \sup_{\tilde{\mathbf{y}}} \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \\ = & \inf_{\lambda \geq 0} \left(\lambda^T\mathbf{b} + \sup_{\delta} (\mathbf{w}^T - \lambda^T\mathbf{A})\delta \right) \\ & + \sup_{\tilde{\mathbf{y}}} \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \end{aligned}$$

Note that the value of the $\sup_{\delta} (\mathbf{w}^T - \lambda^T\mathbf{A})\delta$ will be $+\infty$, unless $\mathbf{w} = \mathbf{A}^T\lambda$, therefore:

$$= \begin{cases} \inf_{\lambda \geq 0} \lambda^T\mathbf{b} + \sup_{\tilde{\mathbf{y}}} [\mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}})] \\ \qquad\qquad\qquad\qquad \text{if } \mathbf{w} = \mathbf{A}^T\lambda \\ +\infty \qquad\qquad\qquad\quad \text{otherwise.} \end{cases}$$

Therefore (Equation 4.15) can be rewritten as:

$$\begin{aligned} & \underset{\mathbf{w} \geq 0}{\text{minimize}}\ Cf(\mathbf{w}) + \inf_{\lambda \geq 0} \lambda^T\mathbf{b} + \\ & \qquad \sup_{\tilde{\mathbf{y}}} \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}}) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \\ & \text{subject to} \quad \mathbf{w} = \mathbf{A}^T\lambda \qquad\qquad \text{(Equation 4.16)} \end{aligned}$$

By substituting $\mathbf{w}$ with $\mathbf{A}^T\lambda$, (Equation 4.16) can be equivalently written as (Equation 4.14). Note that by Lemma (4), the value of $\mathbf{b}$ is is always non-negative, so no value of λ can lead the value of the objective in the outer minimization to negative infinity. $\square$

It is a known fact that maximization (or minimization) of L_1 and L_∞ norms of affine functions can be converted to linear programs (Boyd and Vandenberghe, 2004). In the following proposition, we state that both Theorem 3 and Theorem 4 will lead to equivalent optimization programs in these cases.

Proposition 1. *If the disturbances in the feature space are restricted by some ellipsoid that is defined by L_1 or L_∞ norms, then optimization program that is generated by Theorem 3 can be equivalently transformed to one that is generated by Theorem 4*

The proof can be found in Appendix B.

4.4.3. Ellipsoidal/Polyhedral conjunction

In some cases, the uncertainty set in feature space may resemble an ellipsoid but with additional linear constraints. We can model this as the intersection of an ellipsoid and a polyhedron. The following theorem describes how such uncertainty sets can be transformed into regularizers.

Theorem 5. *For $\Delta^2\Phi(\mathbf{x}, \mathbf{y}) = \{\delta | \|\mathbf{M}\delta\| \leq 1, \mathbf{A}\delta \leq \mathbf{b}\}$, the optimization program of the robust structural SVM in (Equation 4.9) reduces to the following ordinary*

1-slack structural SVM:

$$\begin{aligned}
&\underset{\mathbf{w},\lambda\geq 0,\zeta}{\text{minimize}}\ Cf(\mathbf{w}) + \|\mathbf{M}^{-1}(\mathbf{w}-\mathbf{A}^T\lambda)\|^* + \mathbf{b}^T\lambda + \zeta \\
&\text{subject to} \\
&\qquad \zeta \geq \sup_{\tilde{\mathbf{y}}} \mathbf{w}^T\boldsymbol{\Delta}\phi(\mathbf{x},\mathbf{y},\tilde{\mathbf{y}}) + \Delta(\mathbf{y},\tilde{\mathbf{y}}) \qquad \text{(Equation 4.17)}
\end{aligned}$$

The proof of Theorem 5 is a combination of the proofs of Theorems 3 and 4. First, we perform the Lagrangian relaxation as in the proof of 4, and then we add the dual of $\mathbf{M}^{-1}(\mathbf{w}-\mathbf{A}^T\lambda)$ (the coefficient of δ) as the regularization term.

The results in Theorems 3, 4, and 5 apply to binary and multi-class SVMs as well simply by restricting the space of y to a small set of values. For Theorem 3, this reduces to results proved by Xu et al. (2009). For the later theorems, we are not aware of any analogous previous work in binary or multi-class SVMs.

Some limiting cases of Theorem 5 are also interesting. For example, for a (geometrically) infinitely large polyhedron $\mathbf{A}\delta \leq \mathbf{b}$ (e.g., elements of the vector $\mathbf{b}$ are infinitely large), λ must be $\mathbf{0}$, which recovers the regularization term $\|\mathbf{M}^{-1}\mathbf{w}\|^*$ introduced in Theorem 3.

Let $\lambda_1,\ldots,\lambda_m$ be the eigenvalues of $\mathbf{M}$. If $\min(\lambda_i) \to +\infty$ (for example, a diagonal matrix with very large numbers on the diagonal), then as a result $\delta \to \mathbf{0}$ in the robust formulation. Intuitively, this means that the uncertainty set only contains the unmodified input $\mathbf{x}$. In this case, $\mathbf{M}^{-1}$ approaches the zero matrix, and as a result the regularization term $\|\mathbf{M}^{-1}(\mathbf{w}-\mathbf{A}^T\lambda)\|^*$ fades as expected. On the other hand, if $\max(\lambda_i) \to 0$, then $\|\mathbf{M}^{-1}(\mathbf{w}-\mathbf{A}^T\lambda)\|^* \approx \|L_\mathbf{M}\mathbf{I}(\mathbf{w}-\mathbf{A}^T\lambda)\|^* = L_\mathbf{M}\|(\mathbf{w}-\mathbf{A}^T\lambda)\|^*$, where $L_\mathbf{M} \to +\infty$. Therefore, the constraint $\mathbf{w} = \mathbf{A}^T\lambda$ must be satisfied, leading to (Equation 4.14).

4.5. Experiments

We demonstrate the utility of our approach by applying it to a collective classification problem.

4.5.1. Dataset

We introduce a new dataset based on the political blogs dataset collected by Adamic and Glance (2005). The original dataset consists of 1490 blogs and their network structure from the 2004 presidential election period. Each blog is labeled as liberal or conservative. We expanded this dataset by crawling the actual blog texts in different years to obtain a vector of 250 word features for each blog in each yearly snapshot from 2003 to 2013. We used the internet archive website (`https://archive.org/web/`) to obtain snapshots of each blog in each year. We selected the snapshot closest to October 10th of each year and removed blogs that were inactive for an 8 month window (4 months before and after October 10th).

The political affiliation of a blog can thus be inferred from both the words on the blog and its hyperlink relationships to other blogs, which are likely to have similar political views. Since political topics evolve quickly over time, we expect a significant amount of concept drift over the years, especially over the word features. Since the test distribution is evolving significantly, we might expect a robust model to outperform a non-robust model when trained and tested on different years.[2]

[2]We plan to release both the expanded political blogs dataset and our robust SVM implementation after publication of this work.

4.5.2. Problem Formulation

In our experiments, we use both word features and link features. We construct one multinomial feature for each word i and label k, $\phi_{ik}(\mathbf{x}, \mathbf{y}) = \sum_j x^w_{ji} y_{jk}$, where $x^w_{ji} = 1$ if the jth blog contains the ith word, and $y_{jk} = 1$ if the jth blog has label k. We also construct a link feature for each label k: $\phi_k(\mathbf{x}, \mathbf{y}) = \sum_{ij} x^e_{ij} y_{ik} y_{jk}$, where $x^e_{ij} = 1$ if there is a link from the ith blog to the jth blog.

For our constraints, we assume that the number of words added or removed is bounded by some budget, B_w, and the number of edges by another budget, B_e. Thus, letting $\mathbf{x}^w$ be vector of all word-related variables, $\|\tilde{\mathbf{x}}^w - \mathbf{x}^w\|_1 \le B_w$. Similarly, $\|\tilde{\mathbf{x}}^e - \mathbf{x}^e\|_1 \le B_e$.

In order to construct the uncertainty set in the feature space, we follow the construction procedure in Theorem 2 and then apply Corollary 1. For the word features ϕ_{ik} and edge features ϕ_k we can construct separate uncertainty sets:

$$\begin{aligned}
|\delta_{ik}| &\le \sum_i |\tilde{x}^w_{ik} - x^w_{ik}| \\
\Rightarrow \sum_l |\delta_{ik}| &\le \sum_{i,l} |\tilde{x}^w_{ik} - x^w_{ik}| = \|\tilde{\mathbf{x}}^w - \mathbf{x}^w\| \le B_w \\
|\delta_{ek}| &\le \sum_{i,j} |\tilde{x}^e_{ij} - x^e_{ij}| = \|\tilde{\mathbf{x}}^e - \mathbf{x}^e\| \le B_e
\end{aligned}$$

In our domain there are two classes, liberal and conservative, so $k \in \{0, 1\}$. As a result: $\sum_{k=0}^1 \sum_l \frac{|\delta_{lk}|}{2B_w} \le 1$, and $\sum_{k=0}^1 \frac{|\delta_{ek}|}{2B_e} \le 1$. Summing the equalities and dividing by two:

$$\sum_{lk} \frac{|\delta_{lk}|}{4B_w} + \sum_k \frac{|\delta_{ek}|}{4B_e} \le 1$$

Finally, let $\delta = [\delta_{11}, \ldots, \delta_{nm}, \delta_{e0}, \delta_{e1}]^T$, where $m = 250$ is the number of words attribute that are chosen from training data, and n is the number of the nodes in the graph. Then, $\mathbf{M}$ is a diagonal matrix with entries $[\frac{1}{4B_w}, \ldots, \frac{1}{4B_w}, \frac{1}{4B_e}, \frac{1}{4B_e}]$, so we will have $\|\mathbf{M}\delta\|_1 \leq 1$. Note that, in this uncertainty translation, the base case of Lemma 3 holds in the first place, so the inequality is in its tightest form.

4.5.3. Methods and Results

We partitioned the blogs into three separate sub-networks and used three-way cross-validation, training on one sub-network, using the next as a validation set for tuning parameters, and evaluating on the third. We used mutual information to select the 250 most informative words separately for each training set. However, rather than training, tuning, and testing on the same year, we trained and tuned on the snapshot from 2004 and evaluated the models on every snapshot from 2003 to 2013.

Standard structural SVMs have one parameter C that needs to be tuned. The robust method has an additional regularization parameter $C' = 1/B_e = 1/B_w$ which scales the strength of the robust regularization.[3] We chose these parameters from the semi-logarithmic set $\{0, .001, .002, .005, .1, \ldots, 10, 20, 50\}$. We intentionally added 0 to this set to allow the algorithm remove one of the regularization terms. We learned parameters using a cutting plane method, implemented using the Gurobi optimization engine 5.60 (Gurobi Optimization, 2014) for running all integer and quadratic programs. We ran for 50 iterations and selected the weights from the iteration with the best performance on the tuning set.

[3]In general, B_e and B_w could be tuned separately, but we did not do this in our experiments.

Figure 4.1 shows the average error rate of the robust and non-robust formulations in each year. In 2004, both have very similar accuracy. This is not surprising, since they were tuned for this particular year. In years before and after 2004, the error rate increases for both models. However, the error rate of the robust model is often substantially lower than the non-robust model. We attribute this to the fact that the robust model has additional L_∞ regularization (since L_∞ is the dual of the L_1 uncertainty set used). This prevents the model from relying too much on a small set of features that may change, such as a particular political buzzword that might go out of fashion. These results demonstrate that robust methods for learning structural SVMs can lead to large improvements in accuracy, even when we do not have an explicit adversary or a perfect model of the perturbations.

4.6. Related work

In this chapter, the big picture of our formulation for robustness in the presented algorithms is based on a minimax formulation, where the learner minimizes a loss function and, at the same time, the antagonistic adversary tries to maximize the same quantity. Some related work has focused on designing classifiers that are robust to adversarial perturbation of the input data in a minimax formulation. For example, Globerson and Roweis (2006) introduce a classifier that is robust to feature deletion. Teo et al. (2008) extend this to any adversarial manipulation that can be efficiently simulated. Livni and Globerson (2012) show that a minimax formulation of robustness in the presence of stochastic adversaries results in L_2 (Frobenius for matrix weights) regularization, and for the multi-class case results in two-infinity regularization of the model weights. Torkamani and

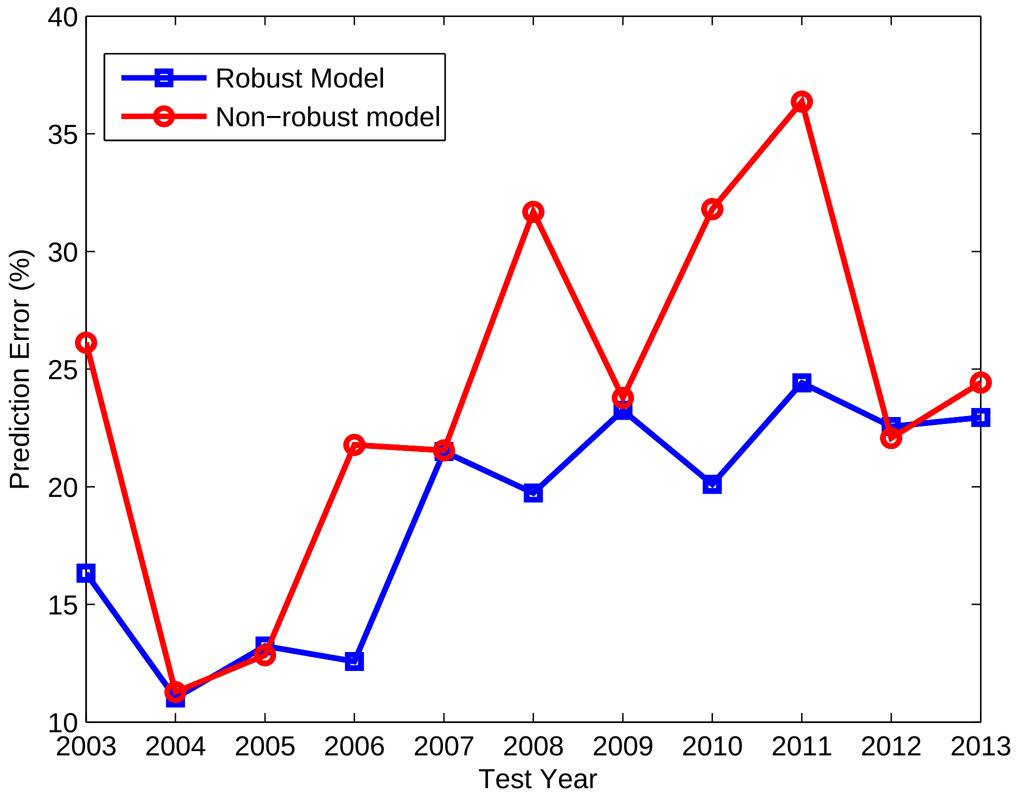


FIGURE 4.1. Average prediction error of robust and non-robust models, trained on year 2004 and evaluated on years 2003-2013.

Lowd (2013), show that for associative Markov networks, robust weight learning for collective classification can be efficiently done with a convex quadratic program.

Xu et al.'s work on robustness and regularization (Xu et al., 2009) is the most related previous work, which analyzes the connection between robustness and regularization in binary SVMs. Our work goes well beyond these results (and the ones mentioned in the introduction) by analyzing arbitrary structural SVMs and showing how they can be made robust without directly simulating the adversary, by choosing the appropriate regularization function.

4.7. Conclusion

In this chapter, we showed that the robust formulation of structural SVMs, which is intractable in general, can be reduced to tractable optimization programs for special uncertainty sets. We also showed that for multinomial feature functions, ellipsoidal uncertainty in sample space can be translated to one in feature space. We also showed that robustness to polyhedral uncertainties, can be achieved by linear regularization of the objective and linear transformation of the feature space. We introduced a new dataset that can be used for structured output prediction in the presence of distribution change over time. Experimental results showed that our method outperforms the standard non-robust approach in the presence of concept drift in the real word data.

So far our focus had been on worst-case adversarial changes in the input data. In the next chapter, we introduce a regularization method, which robustifies the machine learning in the presence of average adversaries. The proposed method optimizes a new loss function, which is the expected hinge loss function under dropout noise.

CHAPTER V

MARGINALIZATION AND KERNELIZATION OF DROPOUT FOR SUPPORT VECTOR MACHINES

This work is under review in the Journal of Machine Learning Research (JMLR). I was the primary contributor to the methodology and writing, and designed and conducted the experiments. My Ph.D. advisor, Dr. Daniel Lowd contributed partly to the methodology and writing. Daniel Lowd was the principle investigator for this work.

A central problem in machine learning is learning complex models that generalize to unseen data. One common solution is to use an ensemble of many models instead of a single model. Another strategy is to expand the dataset, either implicitly or explicitly, by exploiting invariances in the domain. Both strategies reduce the variance of the estimator, leading to more robust models. Dropout training can be viewed as an instance of either of these strategies. In dropout training, portions of the model or input data are randomly "dropped out" while learning the parameters (Srivastava et al., 2014). Thus, dropout can be viewed as optimizing a distribution of models, or optimizing a model on a distribution over datasets. In deep networks, this reduces co-adaptation of the weights and allows more complex models to be learned with less overfitting. In shallow models, such as logistic regression (LR), dropout acts as a regularizer that penalizes feature weights based on how much they influence the classifier's predictions (Wager et al., 2013).

Support vector machines (SVMs) are among the most popular and effective classification methods, obtaining state-of-the-art results in many domains. SVM training algorithms reduce generalization error by maximizing the (soft) margin between the classes. For linear classifiers, this amounts to minimizing the hinge loss plus a quadratic weight regularizer. To learn a non-linear classifier, SVMs can use a kernel function to compute dot products in a high-dimensional feature space without constructing the explicit feature representation. While the max-margin principle is helpful in improving generalization, overfitting remains a risk when learning complex functions from limited data. Kernelized SVMs are at the greatest risk, due to their increased expressivity.

Previous work on dropout has mostly focused on deep networks and logistic regression (Srivastava et al., 2014; Wager et al., 2013; Wang and Manning, 2013; Maaten et al., 2013). For logistic regression, there are methods to make training more efficient by approximating or marginalizing over the randomness introduced by dropout (Wager et al., 2013; Maaten et al., 2013). Other papers analyze the quantitative and qualitative effect of dropout in logistic regression (Wager et al., 2013, 2014). The only work on dropout in SVMs is limited to linear SVMs and consists of a relatively complicated method for optimizing the marginalized dropout objective (Chen et al., 2014a).

In this chapter, we analyze dropout in both linear and non-linear SVMs. Our goal is to develop methods that are simple, efficient, and effective at improving the generalization of SVMs on real-world datasets. For linear SVMs, we show that the expected hinge loss under dropout noise can be closely approximated as a smooth, closed-form function. This marginalized dropout objective is easy to optimize and leads to improved performance on a number of datasets.

For non-linear SVMs, we present two methods for efficiently performing dropout on the kernel feature map, even when this feature map is high- or infinite-dimensional. Our first method generates a linear representation of the input data by randomly sampling from the Fourier transformation bases of the kernel function as introduced by Rahimi and Recht (2007). It then learns a linear SVM with marginalized dropout noise on this transformed feature representation. The second method approximates the effect of dropout in feature space by adding a weighted L_2 regularizer to the dual variables in the SVM optimization problem. In experiments on digit classification and census datasets, both methods lead to improved performance compared to a standard SVM with a radial basis function (RBF) kernel, but the transformed feature representation method is more effective than dual regularization.

5.1. Related work

The connection between different types of noise and regularization has been explored by many authors. For example, Bishop (1995) shows that adding Gaussian noise to neural network inputs while training is equivalent to L_2 regularization of the weights. For the case of linear SVMs, Xu et al. (2009) demonstrate that worst-case additive noise with bounded norm is equivalent to regularizing the weights with the dual norm. Globerson and Roweis (2006) introduce the "nightmare at test time" scenario in which an adversary removes a certain number of features from the model, setting them to zero. They propose a modified SVM formulation to optimize performance against such an adversary.

Wager et al. (2013) analyze the regularization effect of dropout noise in generalized linear models (GLMs) by computing a second-order approximation to the expected loss of the dropout-corrupted data. This allows the dropout objective to be optimized explicitly rather than implicitly. Unfortunately, this second-order approximation cannot be applied to linear SVMs because the hinge loss is not differentiable.

Maaten et al. (2013) also introduce methods for learning linear models with corrupted features, marginalizing over the corruption by introducing a surrogate upper bound of the logistic loss. For certain loss functions and noise distributions, they can compute the marginalized objective directly; for logistic loss, they minimize an upper bound on the expected loss instead. They do not consider hinge loss. Chen et al. (2014a) extend these methods to analyze linear SVMs with dropout noise. Since exactly computing the marginalized objective is hard, the authors introduce a variational approximation. They optimize this approximate objective using expectation maximization and iterative least squares. The goals of Chen et al. are similar to ours, but our formulation is simpler and easier to optimize.

Wang and Manning (2013) introduce a fast way to approximate the expected dropout gradient. The key idea is to draw the noised activation of each unit from a normal distribution instead of directly sampling many Bernoulli variables. By using this approximation several times for each training example, the variance of the gradients is reduced without a significant increase in computation time. They also present a closed-form solution which relies on approximating the logistic function as a Gaussian cumulative distribution function.

In this chapter, we also use a Gaussian approximation to the noisy dot products. However, we focus on hinge loss rather than logistic loss, and we show how to compute compute the gradient analytically without sampling or introducing any additional approximations.

Dropout is significantly different from additive noise, since the expected perturbation of a feature depends on its value in the data. For example, features that are already zero will be perturbed by standard additive noise, but remain unchanged by dropout. Instead, dropout noise is best viewed as an instance of *multiplicative* noise, since each feature is multiplied by 0 with some probability δ and $1/(1-\delta)$ with probability $(1-\delta)$.

To date, there has been limited exploration of training with multiplicative noise other than dropout[1], and no study of training SVMs with multiplicative noise. In this chapter, we address both of these questions, leading to a better understanding of how noise relates to generalization in different types of models.

5.2. Dropout in linear SVMs

A standard formulation for learning linear SVMs is to minimize the hinge loss of the training data with a quadratic regularizer on the weights:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N}[1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)]_+ \qquad \text{(Equation 5.1)}$$

where $\mathbf{w}$ and b are the model parameters (weights and bias); the training data consists of instance and label pairs, $\mathbf{x}_i \in \mathcal{R}^n$ and $y_i \in \{+1, -1\}$; λ is the L_2 regularization coefficient; and $[z]_+ = \max(z, 0)$ is the hinge function. We focus on binary classification, where labels are $+1$ and -1; multiclass classification can be reduced to binary classification.

The idea of dropout training is to optimize performance over a *distribution* of model structures or datasets. For linear SVMs, this amounts to minimizing the

[1]Wang et al. (2013) also consider multiplicative Gaussian noise, and observe that it is equivalent to dropout under the quadratic approximation.

expected loss over noisy versions of the training data:

$$\text{minimize}_{\mathbf{w},b} \; \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} E_{\tilde{\mathbf{x}}_i}[1 - y_i(\mathbf{w}^T\tilde{\mathbf{x}}_i + b)]_+ \qquad \text{(Equation 5.2)}$$

For dropout noise, $\tilde{\mathbf{x}}_i$ is constructed by removing features from the original training example $\mathbf{x}_i$ with some dropout probability δ. More formally, $\tilde{\mathbf{x}}_i$ can be represented as $\mathbf{x}_i$ with multiplicative noise: $\tilde{x}_{ij} = \zeta_j x_{ij}$, where $\zeta_j = 0$ with probability δ and $\zeta_j = 1/(1-\delta)$ with probability $1-\delta$. Note that $E[\zeta_j] = 1$ and $E[\tilde{\mathbf{x}}_i] = \mathbf{x}_i$.

When the data is low-dimensional, or the data matrix is extremely sparse, it may be affordable to compute the expected loss or its gradient exactly. More formally, when there are few non-zeros in a data sample or the weight vector is expected to be sparse (e.g., because of an ℓ_1 regularization), then $E_{\tilde{\mathbf{x}}_i}[1 - y_i(\mathbf{w}^T\tilde{\mathbf{x}}_i + b)]_+$ can be expanded to $\sum_\xi p(\xi)[1 - y_i((\mathbf{w}\odot\mathbf{x}_i)^T\xi + b)]_+$, where ξ is the vector of the multiplicative noise in all dimensions, $\odot$ is the elementwise (Hadamard) product, and $p(\xi) = \delta^{(\#\text{zeros in }\xi)}(1-\delta)^{(\#\text{ones in }\xi)}$. Since the number of applicable dropout noise vectors is exponential in the number of the non-zeros in $\mathbf{w}\odot\mathbf{x}_i$ (i.e., $\|\mathbf{w}\odot\mathbf{x}_i\|_0$), for small values of $\|\mathbf{w}\odot\mathbf{x}_i\|_0$ the computation of the expected value of the loss function under dropout noise may be tractable. There can be cases where the data is not sparse, but the weight vector is expected to be sparse, due to a sparsity-inducing penalty. Even in such a scenario, if we start the optimization algorithm with a sparse initial weight vector, we may be able to calculate the exact dropout expectation during the optimization.

The difficulty comes when the data is high-dimensional and the expected weight vector is relatively dense. Then, neither the expected loss nor its gradient can be efficiently calculated.

The simplest alternative is to approximate the expected loss with sampling or Monte-Carlo methods. For online learning algorithms (such as Pegasos (Shalev-Shwartz et al., 2011)), noisy instances can be generated in each iteration. For batch learning algorithms, we can approximate this expectation using K noisy replications of the dataset:

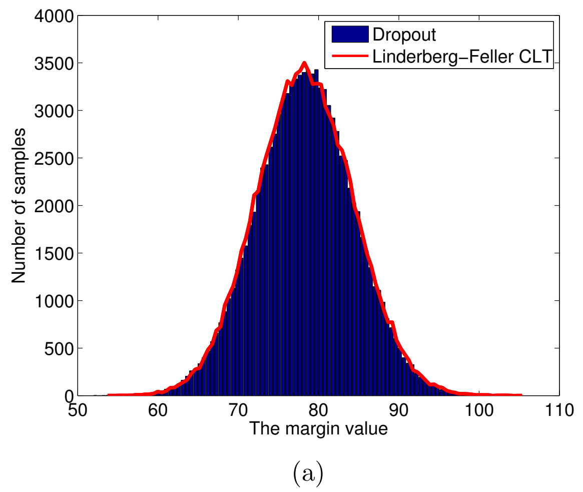
$$\text{minimize}_{\mathbf{w},b}\ \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{K}\sum_{k=1}^{K}\sum_{(\tilde{\mathbf{x}},y)\in\tilde{\mathcal{D}}^{(k)}}[1 - y(\mathbf{w}^T\tilde{\mathbf{x}} + b)]_+$$

where $\tilde{\mathcal{D}}^{(k)}$ is the kth noisy replication of $\mathcal{D}$, in which each instance $\mathbf{x}$ has been replaced by a noised instance $\tilde{\mathbf{x}}$.

The Monte-Carlo approach is simple, but it can be computationally expensive. Obtaining a good approximation of the expectation may require many iterations for online algorithms or many noisy replications of the data for batch algorithms. Thus, we propose to approximate the expectation analytically, rather than stochastically.

The advantages of an analytic approximation are faster training times and more accurate solutions. This idea has already been applied to dropout in logistic regression, either optimizing an approximation or an upper bound on the expected logistic loss (Wager et al., 2013; Maaten et al., 2013). For linear SVMs, the quadratic approximation cannot be applied, because hinge loss is non-differentiable.

In this section, we derive a smooth approximation of the expected hinge loss. The objective is easy to compute and can be optimized directly with standard gradient-based methods.



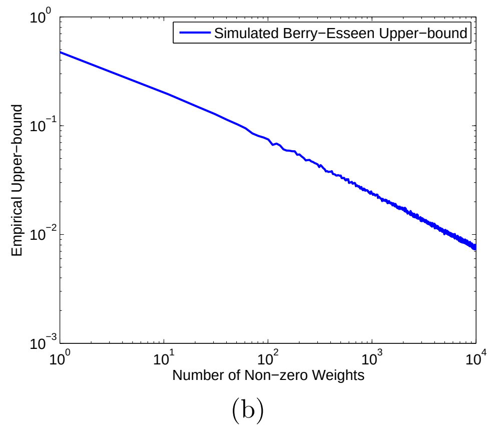


FIGURE 5.1. The results of running a Monte-Carlo simulation of calculating $1 - y(w^T\tilde{x} + b)$ for randomly drawn $\tilde{x}$'s and drawings from the approximated Gaussian distributions. The dimension of each sample $\tilde{x}$ is 50 in the histogram on the left. Right: simulation of the Berry-Esséen upper-bound for different number of non-zero weights

Let $\tilde{x}_i = x_i \odot \zeta$ ($\zeta = [\zeta_1, \ldots, \zeta_m]^T$ and m is the dimension of x_i) be the corrupted version of x and y be its label, such that ζ_j's are independently and identically drawn from a Bernoulli distribution with parameter δ. According to the Lindeberg-Levy central limit theorem, if we have minimum and maximum values for the features and the weights, by the increase of the dimension m and non-zero weights and features per example, the margins of the SVM for this sample converge-in-distribution as following: $1 - y(w^T\tilde{x}_i + b) \xrightarrow{.D.} \mathcal{N}(1 - y(w^T x_i + b), \frac{\delta}{1-\delta}\sum_{j=1}^m x_{ij}^2 w_j^2)$.

In practice, in an SVM training process with fixed regularization, the weights have bounded magnitude. This is similar to the approach of Wang and Manning (2013), where they propose a similar application of the central limit theorem to improve the speed of Monte-Carlo dropout in logistic regression. Figure 5.1a shows an example distribution over margin values according to sampled dropout noise and the approximated Gaussian distribution. Although the dimension of the sample vectors in this simulation is small (~ 50), we observe a close match between the two histograms.

Lemma 5. *The expected value of the hinge function over a normal distribution is:*

$$\mathbb{E}_{\xi\sim\mathcal{N}(\mu,\sigma^2)}[\xi]_+ = \mu\Phi(\frac{\mu}{\sigma}) + \sigma\phi(\frac{\mu}{\sigma}) \qquad \text{(Equation 5.3)}$$

where Φ and ϕ are respectively the cumulative and probability density functions of a normal distribution with zero mean and variance equal to one.

The proof is provided in Appendix C.

Therefore, by Lemma 5, the optimization program of the SVM with L_2 regularization in the primal form (Problem Equation 5.2) with dropout noise can be approximated by the following optimization program:

$$\text{minimize}_{w,b} \ \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{N} \ u_i\Phi(\frac{u_i}{\sigma_i}) + \sigma_i\phi(\frac{u_i}{\sigma_i}) \qquad \text{(Equation 5.4)}$$

where $u_i = 1 - y_i(w^T x_i + b)$, $\sigma_i = \sqrt{\frac{\delta}{1-\delta}\sum_{j=1}^{m} x_{ij}^2 w_j^2}$ (m is the number of features), Φ and ϕ are the cumulative and probability density functions of the standard normal distribution. A direct proof is given in Appendix C.

5.2.1. Convexity

The marginalized cost function (Equation 5.4) is nonlinear, but it is always convex. We use the following lemma for proving its convexity:

Lemma 6. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a multivariate function. Also let $g(t) = f(x_0 + t\Delta x)$ ($t \in \mathbb{R}$) for some arbitrary $x_0, \Delta x \in \mathbb{R}^m$. If $g(t)$ is convex in t for all $x_0, \Delta x \in \mathbb{R}^m$, then $f(x)$ is convex in x.*

Proof. By the definition of convexity, it suffices to show $(1-\lambda)f(A) + \lambda f(B) \geq f((1-\lambda)A + \lambda B)$ for any $\lambda \in [0,1]$ and any $A, B \in \mathbb{R}^m$. Let $x_0 := A$ and $\Delta x := B - A$, then the former inequality is equivalent to $(1-\lambda)g(0) + \lambda g(1) \geq g(\lambda)$, which holds by assumed convexity of g. □

In the following theorem, we prove that the proposed cost function is surprisingly convex. Therefore, it can be efficiently optimized by off-the-shelf optimization algorithms.

Theorem 6. *The marginalized loss $f(w, b; y_i, x_i) = u_i\Phi(\frac{u_i}{\sigma_i}) + \sigma_i\phi(\frac{u_i}{\sigma_i})$ is jointly convex in w and b for any given sample and label pair (x_i, y_i), where $u_i = 1 - y_i(w^T x_i + b)$, $\sigma_i = \sigma_\delta\sqrt{\sum_{j=1}^m x_{ij}^2 w_j^2}$.*

Proof. Consider a slice cut of the objective function in an arbitrary direction $(\Delta w, \Delta b)$ from an arbitrary point (w, b) in the parameter space.

Let:

$$\begin{aligned}
u_i(t) &= 1 - y_i\left((w + t\Delta w)^T x_i + b + t\Delta b\right) = 1 - y_i(w^T x_i + b) - t(y_i \Delta w^T x_i + y_i \Delta b) \\
&= U - \Delta U t \\
\sigma_i(t) &= \sigma_\delta\sqrt{\sum_k x_{ik}^2 (w_k + t\Delta w_k)^2} = \sigma_\delta\sqrt{\sum_k x_{ik}^2(w_k^2 + 2tw_k\Delta w_k + t^2\Delta w_k^2)} \\
&= \sigma_\delta\sqrt{\sum_k x_k^2 w_k^2 + t\sum_k 2x_k^2 w_k \Delta w_k + t^2\sum_k x_k^2 \Delta w_k^2)} \\
&= \sigma_\delta\sqrt{S + pt + qt^2} \qquad \text{(Equation 5.5)}
\end{aligned}$$

where $U = 1 - y_i(w^T x_i + b)$, $\Delta U = y_i(\Delta w^T x_i + \Delta b)$, $S = \sum_k x_k^2 w_k^2$, $p = \sum_k 2x_k^2 w_k \Delta w_k$ and $q = \sum_k x_k^2 \Delta w_k^2$. Also, let $f(t) = u_i(t)\Phi(u_i(t)/\sigma_i(t)) + \sigma_i(t)\phi(u_i(t)/\sigma_i(t))$. Based on Lemma 6, if $f(t)$ is convex in t for any (x_i, y_i), w, b, Δw and Δb, then $f(w, b; y_i, x_i)$ is jointly convex in its parameters. We have:

$$\begin{aligned}\frac{\partial^2 f(t)}{\partial^2 t} &= \frac{e^{-\frac{(U-\Delta U t)^2}{2(S+pt+qt^2)\sigma_\delta^2}}\left((2\Delta US + \Delta Upt + pU + 2qtU)^2 + (4qS - p^2)(S + tp + qt^2)\sigma_\delta^2\right)}{4\sqrt{2\pi}(S + tp + qt^2)^{5/2}\sigma_\delta} \\ &= \frac{e^{-\frac{(u_i)^2}{2\sigma_i^2}}\left((2\Delta US + \Delta Upt + pU + 2qtU)^2 + (4qS - p^2)\sigma_i^2\right)}{4\sqrt{2\pi}(\sigma)^5/\sigma_\delta^4} \qquad \text{(Equation 5.6)}\end{aligned}$$

Note that the denominator of the second derivative is non-negative $(4\sqrt{2\pi}(\sigma)^5/\sigma_\delta^4 \geq 0)$, and in the nominator, all terms are always non-negative, except $4qS - p^2$, which can be negative for some values of S, p and q (i.e. $e^{-\frac{(U-\Delta U t)^2}{2(S+pt+qt^2)\sigma_\delta^2}} \geq 0$, $(2\Delta US + \Delta Upt + pU + 2qtU)^2 \geq 0$ and $(S + tp + qt^2)\sigma_\delta^2 \geq 0$).

By definition, $\sigma_i(t)$ is always non-negative. Consider the hypothetical values of S, p and q, for which, there exist some t such that $\sigma_i(t) = \sigma_\delta\sqrt{S + pt + qt^2} = 0$. Then the roots of $\sigma_i(t)$, will be $t = \frac{-p \pm \sqrt{p^2 - 4qS}}{2q}$.

As long as $\sigma_i(t)$ has no real roots (i.e. $\sqrt{p^2 - 4qS}$ is imaginary), we will have $(4qS - p^2) > 0$, and as a result $\frac{\partial^2 f(t)}{\partial^2 t} > 0$.

The marginalized cost function is undefined for $\sigma_i = 0$, which appears in $\frac{u_i}{\sigma_i}$, however, it is continuous and convex in the limit as $\sigma_i(t) \to 0$ (or equivalently $t \to \frac{-p \pm \sqrt{p^2 - 4qS}}{2q}$, when $p^2 - 4qS \geq 0$). Let $\min_t \sigma_i(t) = 0$ (i.e. for some values of S, p and q, $p^2 - 4qS \geq 0$), then it is easy to show that:

$$\lim_{\sigma_i(t)\to 0^+} f(t) \quad = \quad \begin{cases} u_i(t) & u_i(t) > 0 \\ 0 & u_i(t) \le 0 \end{cases} = [u_i(t)]_+ = [1 - y_i((w + t\Delta w)^T x_i + b + t\Delta b)] +$$

which is the hinge loss of misclassifying the ith sample as t varies. As a result, if $\sigma_i(t) = 0$ for the ith training sample, then the contribution of that sample to the overall objective function will be exactly the same as adding a regular hinge-loss. Clearly, the overall objective remains convex: addition of several convex functions result in a convex function. Therefore, for any possible $\sigma_i(t)$ ($\sigma_i(t) \ \geq \ 0$), the function $f(t)$ is convex. Correspondingly, $f(w, b; y_i, x_i)$ will be convex (by Lemma 6).

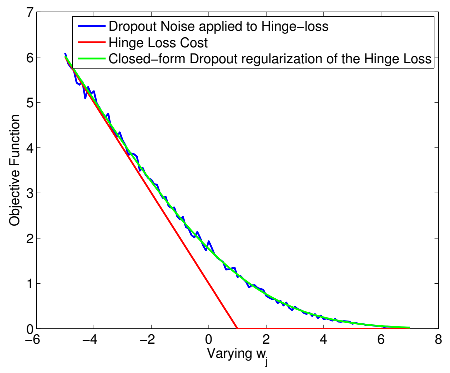
□

5.2.2. Regularization effect

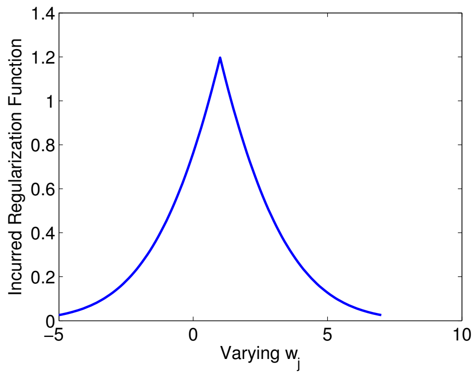
The resulting cost function (Equation 5.4) can be directly optimized, and it is not the same as the hinge loss any more. In order to understand the theoretical reasons of why dropout performs well in shallow model such as SVMs, we can compare the resulting cost function with ordinary hinge-loss. From a theoretical point of view, the generalization power of dropout-based methods comes from the regularization penalty $\mathcal{R}_{\text{dropout}}(w)$ that dropout incurs to the model weights:

$$\mathcal{R}_{\text{dropout}}(w) = \sum_{i=1}^{N} \ u_i\Phi(\frac{u_i}{\sigma_i}) + \sigma_i\phi(\frac{u_i}{\sigma_i}) - [1 - y_i(w^T x_i + b)]_+ \text{(Equation 5.7)}$$

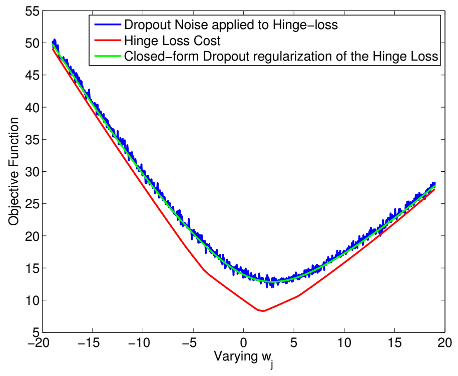
where $u_i = 1 - y_i(w^T x_i + b)$, $\sigma_i = \sqrt{\sum_{j=1}^{m} x_{ij}^2 w_j^2}$. Although, the incurred regularization function is highly non-convex, but as proved the previous section, the overall cost function remains convex (5.2).



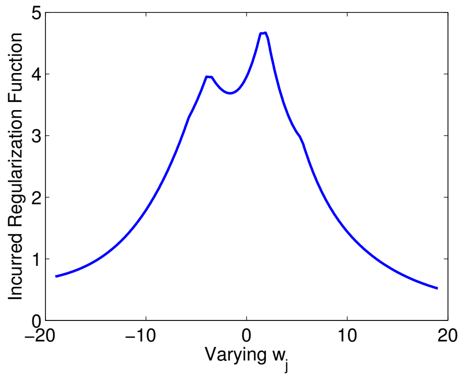
(a) Single sample's contribution to the loss function



(b) The regularization effect of one sample (i.e. the marginalized loss minus the hinge loss)varying the weight vector in one dimension.



(c) Aggregated loss of several samples



(d) The aggregated regularization effect of several samples from a one dimensional cut of the loss function

FIGURE 5.2. Losses and differences in losses as a function of a single model weight.

Note that the marginalized cost function is always an upper-bound on the hinge loss. Although the effective regularization function is non-convex, the marginalized objective function itself is convex.

5.2.3. Approximation quality

Since the approximation depends on the central limit theorem, (assuming that $z_i = 1 - y_i(w^T x_i + b) \sim \mathcal{N}(u_i, \sigma_i^2)$), this method should be used when the data is not extremely sparse (e.g., there are at least 10 non-zero features in the average sample), and the regularization penalty does not favor extremely sparse solutions.

More formally, let $u_i = 1 - y_i(\bar{w}^T \tilde{x}_i + b)$ be a random variable that represents the margin for some fixed weights $\bar{w}$ and some arbitrary dropped-out sample $\tilde{x}_i$ with the desired label y_i, and $m_{\bar{w}}$ be the number of non-zero elements in $\bar{w}$. Also let $F_{u_i}(z) = P_{u_i}(u_i \leq z)$ be the cumulative density function (CDF) of u_i. By the Berry-Esséen theorem, the supremum of the difference between the CDF of u_i and its Gaussian approximation is upper-bounded by:

$$\sup_z |F_{u_i}(z) - \Phi(\frac{z - \mu_i}{\sigma_i})| \leq \frac{C\rho_i}{\sigma_i^3 \sqrt{m_{\bar{w}}}} \qquad \text{(Equation 5.8)}$$

By the best estimate to date, $C \leq 0.4748$ (Korolev and Shevtsova, 2012). ρ_i is the third moment of u_i, and can be calculated in closed-form. In Figure 5.1b, we simulate this upper-bound for different numbers of non-zero weights on a toy dataset. In practice, we observe that the true and the approximated distributions of u_i closely match each other as in Figure 5.1a.

It is easy to prove that the optimization program in (Equation 5.4) is always an upper bound on the regular SVM's objective. Therefore, the dropout approximation is in fact an optimization transfer that intrinsically applies extra regularization effects on the learned weights. The objective is an smooth approximation of a convex function (the expected hinge-loss), and is easily differentiated and optimized with gradient descent, LBFGS, or other standard methods.

We provide visual intuition about our proposed approximation in Figure 5.2. In Figure 5.2a, We consider one single sample, and show the hinge loss (red), its closed form expectation from Equation 5.3 (green), and the Monte-Carlo function when the function is averaged over actual dropout noisy samples (blue). The noised hinge loss provides an upper bound that is tight at the extremes and smooth in between. Figure 5.2c shows how several samples with different margins form the aggregated loss function. As the dimensionality of model weights increases, the approximation tightly converges to the true expectation which is convex. For very low-dimensional inputs ($\sim$ 4-5), the method can still be applied but might perform poorly. This method is appropriate for real-world problems, where we deal with hundreds or thousands of dimensions.

5.3. Dropout in non-linear SVMs

By using the kernel trick, SVMs can learn a linear classifier in a higher dimensional feature space without explicitly constructing those features. The kernel trick relies on the dual SVM optimization program:

$$\begin{aligned} &\text{maximize}_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \; y_i y_j \alpha_i \alpha_j k(x_i, x_j) \\ &\text{subject to } \; \sum_i y_i \alpha_i = 0, \; 0 \le \alpha_i \le 1/\lambda \; \forall i \end{aligned} \qquad \text{(Equation 5.9)}$$

where y is the vector of labels, λ is the L_2 regularization weight of the primal optimization program, and $k(x_i, x_j) = f(x_i)^T f(x_j)$ is the dot-product (reproducing kernel) of a feature function vector $f(.)$ in a Hilbert space. For many feature functions, the kernel entry $k(x_i, x_j)$ can be calculated even if $f(\mathbf{x})$ has no explicit representation and is infinite dimensional. Instead of maintaining feature weights $\mathbf{w}$ (which could be infinite dimensional), the dual problem uses instance weights α. The predicted label for a new instance $\mathbf{x}'$ is given by: $\text{sign}(\sum_i y_i \alpha_i k(x_i, \mathbf{x}'))$. The instances $\mathbf{x}_i$ with $\alpha_i > 0$ are commonly referred to as *support vectors*.

5.3.1. Defining dropout in kernels

In deep networks, dropout can be applied to the input layer, any of the hidden layers, or some combination of them. In SVMs with non-linear kernels, we can analogously apply dropout noise to the either the *input space attributes* or the *implicit features*.

Given a kernel function $k(x_i, x_j)$ with corresponding feature function f, we define the *kernelized dropout function*, $\tilde{k}(x_i, x_j, \zeta, \xi)$, as a function of both the instances, x_i and x_j, and the dropout noise, ζ and ξ. The specific definition depends on the type of dropout:

– Input space dropout:

$$\tilde{k}(x_i, x_j, \zeta, \xi) = k(\zeta \odot x_i, \xi \odot x_j)$$

– Feature space dropout:

$$\tilde{k}(x_i, x_j, \zeta, \xi) = (\zeta \odot f(x_i))^T (\xi \odot f(x_j))$$

We can also drop the whole support vectors (i.e. dropping out α's). This turns out to be very similar to some variations of bagging, therefore we skip it in this chapter.

For a linear kernel, feature space and input space are identical, so dropout in both spaces is the same. Dimension dropout is also the same, since excluding dimensions from the kernel calculation is equivalent to multiplying those attributes by zero.

$$\begin{aligned} k_{\zeta,\xi}(\zeta \odot x_i, \xi \odot x_j) &= \sum_{l:\zeta_l \neq 0, \xi_l \neq 0} (\zeta_l x_{i,l})(\xi_l x_{j,l}) \\ &= \sum_l (\zeta_l x_{i,l})(\xi_l x_{j,l}) = k(\zeta \odot x_i, \xi \odot x_j) \end{aligned} \qquad \text{(Equation 5.10)}$$

More generally, dimension dropout is equivalent to input space dropout for any kernel function that only depends on the dot-products of the original vectors, and not the original vectors themselves. That is, if $k(x_i, x_j) = g(x_i^T x_j)$ for some function g, then dimension dropout is equivalent to input space dropout. This includes all polynomial kernels, which can be expressed as $k(x_i, x_j) = (x_i^T x_j + c)^d$.

One kernel where they differ is the radial basis function (RBF) kernel: $k(x_i, x_j) = \exp\left(\frac{-\gamma \|x_i - x_j\|^2}{2}\right)$. The RBF kernel is *translation invariant*, so that $k(x_i + \Delta, x_j + \Delta) = k(x_i, x_j)$. Standard input space dropout does not maintain this invariance, since the effect of zeroing out an attribute depends on its original magnitude.

Dropout can be applied both to training and testing data. In fact after learning the model, we can apply the dropout noise to the test data, and then perform the classification on the corrupted input (or make the final classification by the ensemble result of classifying several noisy versions of the same input data). We address this issue later. In Appendix E, we derive the marginalized (expected) prediction function for dimension dropout in RBF kernels.

Ideally, we would like to find the dual solution for the kernelized version of Equation 5.2. Instead of the one-to-one correspondence of α_i's and x_i's, we need to index each α_i by the noise value as well. If we let $\alpha_i(\zeta)$ be the corresponding dual variable for the noisy sample $\tilde{x}_i = \zeta \odot x_i$ (or equivalently, the noisy feature $\tilde{f}(x_i) = \zeta \odot f(x_i)$), then Equation 5.9 turns to the following calculus of variation optimization problem:

$$\begin{aligned}
\underset{\alpha}{\text{maximize}}\ & \mathbf{E}[\alpha_i(\zeta)] \\
& -\frac{1}{2}\sum_{i,j} y_i y_j \mathbf{E}[\alpha_i(\zeta)\alpha_j(\xi)\tilde{k}(x_i, x_j, \zeta, \xi)] \\
\text{subject to}\ & \sum_i y_i \mathbf{E}_\zeta[\alpha_i(\zeta)] = 0, \\
& 0 \le \alpha_i(\zeta) \le 1/\lambda \quad \forall i, \zeta
\end{aligned} \qquad \text{(Equation 5.11)}$$

where ζ and ξ are drawn from the dropout noise distribution.

Proposition 2. *After applying dropout in input or feature space to a valid kernel, the resulting matrix is a valid kernel.*

Proof. We first define an augmented instance space $\mathcal{X}'$ containing both the original attributes and the dropout noise, e.g., $x'_i = (x_i, \zeta)$ and $x'_j = (x_j, \xi)$. We then define a kernel k' over this space by constructing an appropriate feature function f'. For input space dropout, let $f'(x'_i) = f(\zeta \odot x_i)$, and for feature space dropout, let $f'(x'_i) = \zeta \odot f(x_i)$. In both cases, it follows that $\tilde{k}(x_i, x_j, \zeta, \xi) = f'(x'_i)^T f'(x'_j) = k'(x'_i, x'_j)$. □

The kernel from input dimension dropout is not guaranteed to be positive semidefinite (PSD). However in practice, we rarely observed non-PSD kernels;

even the rare cases of non-PSD kernels had very small (in magnitude) negative eigenvalues. We solved the optimization programs by implementing on-the-fly kernels in LibSVM, and solved the corresponding optimization programs by the SMO algorithm. In our experiments, the optimization programs always converged.

Example 5.1. We present an example in which dropout in dimension can result in a non-PSD kernel. For a dataset with only two samples $\{x_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, x_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\}$, suppose dimension-dropout generates the following noisy dataset: $\{\tilde{x}_1^1 = \begin{pmatrix} 1 \\ \times \end{pmatrix}, \tilde{x}_1^2 = \begin{pmatrix} \times \\ 0 \end{pmatrix}, \tilde{x}_2^1 = \begin{pmatrix} \times \\ 1 \end{pmatrix}, \tilde{x}_2^2 = \begin{pmatrix} 0 \\ \times \end{pmatrix}\}$, where '$\times$' means that the corresponding dimension is dropped. Then the RBF-dimension-dropout kernel will generate the following kernel matrix, which has negative eigenvalues for some values of γ:

$$\begin{pmatrix} 1 & 1 & 1 & e^{-\gamma} \\ 1 & 1 & e^{-\gamma} & 1 \\ 1 & e^{-\gamma} & 1 & 1 \\ e^{-\gamma} & 1 & 1 & 1 \end{pmatrix} \qquad \text{(Equation 5.12)}$$

In the rest of this section, we introduce several approximations and variations of Equation 5.11, which we will evaluate in the experiments section.

5.3.2. Marginalized dropout in feature space

For many kernels, the explicit feature representation is extremely high-dimensional or even infinite-dimensional. Therefore, direct application of dropout marginalization will not be practical. In this subsection, we introduce two methods for taking advantage of both the efficiency of kernel tricks and the accuracy improvement by dropout.

5.3.2.1. Kernel approximation

Although kernel methods have proven to be successful in predictive non-linear models, learning these models requires $\mathcal{O}(n^2)$ memory as well as a long training time, and computing the decision function can be costly when the number of support vectors is large. These two issues make kernel methods less practical, especially on large datasets. Randomized algorithms for approximating kernel matrices (Schölkopf, 2002; Blum, 2006) have inspired several methods for efficiently converting the training and evaluation of kernel machines into linear weight learning and score prediction (Rahimi and Recht, 2007; Le et al., 2013). The basic idea behind these methods is to find a relatively low-dimensional feature representation $z(x)$ such that $z(x_i)^T z(x_j)$ approximates the desired kernel function, $k(x_i, x_j)$.

Besides the practical efficiency of such feature representation methods, we can take advantage of more complicated linear methods to improve the prediction. For example, this will allow us to naturally apply the marginalized linear SVM method from Section 5.2. on $z(x)$ as the training features.

We have built on Rahimi and Recht's method (Rahimi and Recht, 2007) by focusing on the RBF kernel. However, it can be applied to any other translation-invariant kernel as well. Their method is based on Bochner's theorem (Rudin, 2011): "A continuous translation-invariant kernel $k(x_i, x_j) = k(x_i - x_j)$ on R^d is positive definite if and only if $k(\Delta)$ is the Fourier transform of a non-negative measure." By randomly sampling from the terms of this Fourier transformation, we can approximate the kernel with some convergence guarantees. As a result, for the RBF kernel, $k(x_i, x_j) = \exp\left(\frac{-\gamma\|x_i - x_j\|_2^2}{2}\right)$, we can randomly draw random frequencies $\{\beta_1, \ldots, \beta_D\}$ from a normal probability density function with mean zero and covariance $2\gamma dI$ (d is the dimension of input space and I is the identity matrix), and draw random rotation angles $\{\alpha_1, \ldots, \alpha_D\}$ uniformly from $[0, 2\pi]$. Then, $z(x) = \sqrt{\frac{2}{D}}[\cos(\beta_1^T x + \alpha_i), \ldots, \cos(\beta_D^T x + \alpha_D)]^T$ will be the linear feature representation, such that $z(x_i)^T z(x_j) \approx \exp\left(\frac{-\gamma\|x_i - x_j\|_2^2}{2}\right)$.

Our experimental results show that applying the marginalized linear SVM on top of this feature representation will outperform the accuracy of an exact kernel SVM on the MNIST and Adult datasets.

5.3.2.2. α-Regularization

Next, we consider dropout in the feature space of the original kernel feature mapping. Formally: $k(\tilde{x}_i, \tilde{x}_j) = \tilde{f}(x_i)^T \tilde{f}(x_j)$, where $\tilde{f}(x) = f(x) \odot \zeta$, $\mathbf{E}_\zeta[\tilde{f}(x)] = f(x)$ (i.e., $\mathbf{E}[\zeta] = \mathbf{1}$), and let $\mathrm{var}(\zeta) = \sigma_\zeta^2 \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

We would prefer to optimize the marginalized dropout objective directly, as done in the linear case. There are two key challenges. First, the dual formulation in Equation 5.11 has an exponential number of variables, one for each possible corruption of each training instance. Second, for infinite-dimensional feature functions, the dropout noise will also be infinite-dimensional. We can solve both problems by introducing a simple approximation: we constrain the value of $\alpha_i(\zeta)$ to a constant α_i for all ζ. In other words, all noisy copies of the same instance will share the same weight in the SVM.

The following theorem shows how this simplification results in a tractable approximation of Equation 5.11.

Theorem 7. *When each α_i is constant, Equation 5.11 is equivalent to standard SVM learning (Equation 5.9) with a modified kernel $Q = K + \sigma_\zeta^2 K \odot \mathbf{I}$, where K is the original kernel matrix and σ_ζ^2 is the variance of each dimension of the dropout noise, ζ_l.*

The proof can be found in Appendix D.

This optimization problem can be viewed as adding a weighted L_2 regularizer on the α_i weights:

$$R(\alpha) = \frac{\sigma_\zeta^2}{2} \sum_i k(x_i, x_i)\alpha_i^2 \qquad \text{(Equation 5.13)}$$

Therefore, we refer to this technique as α-regularization. In the dual program of L_2-SVMs (where the squared value of hinge-loss is minimized) (Chang et al., 2008), there is a similar regularization effect, but with constant coefficients. Our proposed α-regularization method puts a different weight on each of the dual variables in order to approximate the effect of dropout noise in the feature representation.

Note that all of the support vectors learned with α-regularization must be instances in the original training data. For a linear kernel, this means that the weight vector must lie within the span of the training data. Derezinski and Warmuth (2014) present hardness results for predictors that use linear combinations of instances, suggesting that this can be a serious disadvantage on some problems. In contrast, neither Monte-Carlo dropout, nor the marginalized linear SVM, nor the marginalized linear SVM on the approximated kernel has this restriction.

5.3.3. Monte-Carlo dropout in input space and dimension

We can create a Monte-Carlo approximation of Equation 5.11 by replacing the expectations over all dropout noise with K samples of dropout noise for each training instance. This is equivalent to learning from several noisy copies of the training data. For input space dropout, we can create several noisy replications of the training data and apply standard SVM learning algorithms. This works because input space dropout applies noise *before* computing the kernel.

For input dimension dropout, we need to keep track of the dropout noise explicitly and use it to modify the kernel computation. For example, for the RBF kernel, $k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$. When applying dimension dropout, we need to modify the distance computation so that it only considers non-dropped-out dimensions. Let $d(x_i, x_j; \zeta, \xi)^2 = \sum_{l:\zeta_l \neq 0, \xi_l \neq 0} (\zeta_l x_{i,l} - \xi_l x_{j,l})^2$. Then $\tilde{k}_{rbf}(x_i, x_j, \zeta, \xi) = e^{-\gamma d(x_i, x_j, \zeta, \xi)^2}$. To implement this efficiently, we represent x_i and x_j as sparse vectors where all unspecified dimensions are dropped out and all non-dropped-out zeros are encoded explicitly. In the kernel computation, we iterate only over dimensions where both x_i and x_j have a defined value (which could be zero).

The key advantage of input dimension dropout is that it maintains the translation-invariance property of RBF kernels. The key disadvantage is that the resulting kernel matrix may be non-PSD. In our experiments, we found that input dimension dropout outperforms the ordinary RBF kernel. Furthermore, the negative eigenvalues of this kernel were usually very small in magnitude and did not cause any practical problems for the sequential minimal optimization (SMO) algorithm. If necessary, techniques for stabilizing the optimization of non-PSD kernels could be applied here as well (Lin and Lin, 2003).

For RBF kernels, a model learned with dropout may work poorly on non-noisy instances. We apply two different approaches in our experiments. The first is to ignore this difference and apply the model directly. For small dropout probabilities (5%-10%), the additional bias should be small. The second approach is to compute the expected kernel function over all possible dropout noise. Since each dropout probability is independent, this can be done in linear time. (The proof is provided in Appendix E.) We refer to this latter approach as the "corrected" prediction. In both cases, dropout noise is applied by removing random features and not rescaling the remaining features; the rescaling correction $(1/(1-\delta))$ is designed for linear models and causes problems when support vectors and test instances are scaled differently.

5.4. Empirical results

Datasets. We ran our experiments on several text classification datasets, the MNIST digit classification dataset (LeCun et al., 1998), and the Adult dataset from the UCI repository. The text datasets were two sentiment analysis datasets (PolarityV2, Subj) introduced by Pang and Lee (2004), three datasets based on 20-newsgroups (AthR, BpCrypt, XGraph) previously used by Wang and Manning (2012), and four Amazon sentiment datasets (Books, Kitchen, DVD, Electronics). We also constructed an artificial dataset called M27 from MNIST. In M27, we have selected all 2 and 7 digits from MNIST. For each digit, we randomly selected two integer numbers $i \in [1, 390]$ and $j \in [391, 784]$, then set all pixels that correspond to the indices from i to j of the vectorized 784-dimensional digit image to zero. We repeat this for the training, the tuning, and the testing data.

Methods. On the text datasets, our main point is to show the comparative performance of different linear SVM-based methods: we compare the marginalized SVM (SVM-Marg), Monte-Carlo dropout (SVM-MC), and α-regularization (α-Reg), all with linear kernels (Table 5.1). For nonlinear kernels we focus on radial basis functions (RBF). We compare different linear methods that use the random Fourier bases as feature representation with the exact regular SVM and our proposed α-regularization method (Table 5.2).

Experimental Setup. For the text classification experiments, we used five fold cross validation. For the Monte-Carlo methods, we generate K copies of the training data and apply dropout noise to each sample independently. Learning with noisy replications of the training data is an approximation to minimizing the expected loss when the noise elements are randomly drawn from their respective distribution. All hyper-parameters are selected by cross-validation. For the approximated kernel experiments in Table 5.2, we set the dimension of Fourier bases $D = 4000$ for MNIST and M27 datasets, and $D = 1500$ for the Adult dataset. But, we tuned all other hyper-parameters using held-out data, then re-trained the final model by including both the training and tuning data samples.

Nonlinear models (without linear approximation) are more sensitive to hyper-parameters. Because of this fact, we have also tuned σ_ζ^2 for the nonlinear kernel α-regularization method, as well as the ℓ_2 regularization coefficient λ, and the RBF kernel parameter γ for all methods. On the other hand, the tuning procedure usually selected larger dropout probabilities for linear (both linear and linear approximation of RBF) models.

Results. Table 5.1 shows the error percentage of each linear classifier on each of the text datasets. The best-performing variant is shown in bold.

Marginalized dropout outperforms all other methods-except in one dataset, on which α-regularization outperforms SVM-Marg. Monte-Carlo dropout training led to improved results on all datasets. α-Reg led to slight improvements on seven of nine datasets but usually worked worse than SVM-Marg, suggesting that marginalization in the primal is more effective when applicable. We have also compared our methods with logistic regression, LR with Monte-Carlo dropout, and LR with (marginalized) deterministic dropout (Wang and Manning, 2013). The results have a solid basic trend, whenever SVM itself outperforms LR, the SVM-based dropout methods also outperform the LR-based dropout methods, and vice-versa.

TABLE 5.1. Classification error (%) of linear classifiers on text datasets. The last column is the decrease percentage of the prediction error for best method (mostly SVM-Marg) vs. SVM.

Dataset	SVM	SVM-MC	α-Reg	SVM-Marg	Err.Dec.(%)
AthR	7.16	8.98	**6.74**	6.88	5.87 (SVM-Marg:3.91)
BpCrypt	2.22	1.52	2.22	**1.21**	45.49
Polar2	19.70	18.90	18.70	**16.10**	18.27
Subj	12.96	11.76	13.00	**11.12**	14.20
XGraph	9.33	8.51	9.02	**7.48**	19.83
Books	17.43	16.95	17.35	**13.34**	23.46
Kitchen	12.31	11.61	11.91	**10.61**	13.74
DVD	17.55	16.84	17.61	**15.43**	12.08
Elect.	14.01	13.82	13.91	**11.88**	15.06

We compared the performance of several linear weight learning algorithms using Fourier basis features for approximating the RBF kernel with ordinary RBF kernel on three datasets in Table 5.2. We observe that the simple least-squares (LS+Fourier) method (Rahimi and Recht, 2007) outperforms the exact RBF kernel on two datasets just by itself. However, when it is combined with the marginalized SVM it outperforms all other methods.

We also observe that α-regularization always outperforms the regular kernel SVM on these datasets. However, it does not work nearly as well as the marginalized SVM on the Fourier basis. We attribute this difference to the fact that α-regularization is constrained to only using support vectors from the original dataset, unlike the marginalized SVM.

The highest gain is achieved on M27, where the training and testing is performed on samples with large missing portions. This fact suggests that dropout might be also useful for learning with missing data in non-linear models. (Dekel et al. (2010) have directly addressed this issue for linear models using a relaxation-based technique.)

TABLE 5.2. Classification error (%) of approximated RBF kernel.

Dataset	RBF-SVM Exact	LS (Fourier)	α-Reg	Lin.SVM (Fourier)	Marg.SVM (Fourier)	Err.Dec.(%)
MNIST	1.43	2.41	1.41	1.48	**1.37**	4.38
M27	6.31	5.97	6.05	5.66	**4.93**	27.99
Adult	15.1	14.9	14.97	14.93	**14.84**	1.75

Stacked jittered features are known to help improve the prediction accuracy of kernel SVMs on MNIST (Decoste and Schölkopf, 2002). The jittered pixels depend on the geometrical location of non-zero pixels. Unlike stacked jittered features, dropout works equally well with any fixed permutation of the pixels (regardless of the geometric shape of digits), therefore dropout training is different than learning with additional virtual features. Both methods can be applied simultaneously, but in this work, we would like to measure the amount of improvement that can be achieved only by applying dropout noise.

TABLE 5.3. Classification error(%) curve for different size subsets of MNIST. Comparing no-dropout standard RBF to Monte-Carlo dimension dropout and α-Reg. The last row is the decrease percentage of the prediction error for no dropout vs. the best dropout method.

Training size	**1000**	**2500**	**5000**	**10000**	**25000**	**50000**	**60000**
No-DO	6.84	4.70	3.54	2.85	2.10	1.53	1.43
α-Reg	6.85	4.69	3.57	3.05	2.07	1.49	1.41
DO	**6.44**	**4.26**	**3.34**	**2.65**	**1.95**	**1.50**	**1.40**
Err.Dec.(%)	5.85%	9.36%	5.65%	7.02%	7.14%	4.58%	2.80%

For dimension dropout, we can efficiently marginalize the dropout effect on the kernel at the prediction time. In *Appendix C*, we derive the marginalized prediction function. Table 5.3 shows results for variants of RBF SVMs on MNIST. We vary the training size from 1000 to 60,000 instances. On average, dropout shows small but consistent improvements over no dropout with training instances. On average, training with dropout noise leads to a 5.77% reduction in error. For smaller number of samples, prediction with expected kernel performs better than all other methods. α-Reg was slightly more accurate than no dropout for larger training sets. For the Monte-Carlo methods, we used 100 noisy replications of the training data for the linear methods. For the kernel methods: we used 10 noisy replications training subsets of sizes 1000, 2500, 5000, and 10000; 6 replications for sample size of 25000; and, 3 replications for the samples sizes 50000 and 60000. Larger numbers of replications become increasingly expensive, due to the increased number of support vectors and larger kernel matrices. In experiments with 3 replications of 60,000 examples, the accuracy for increased to 98.61% which suggests more replications could result in higher prediction accuracy.

5.5. Conclusion

While previous results (Hinton et al., 2012; Maaten et al., 2013; Wager et al., 2013; Wang et al., 2013; Wang and Manning, 2013) show that learning with dropout noise can improve the accuracy of neural networks and logistic regression, our work confirms that dropout training can improve the prediction accuracy of SVMs as well.

In this chapter, we introduced two new methods that take advantage of dropout learning without actually drawing samples from a noise distribution. These methods marginalize the effect of dropout in the primal (Marginalized SVM) and dual formulations (α-Regularization) of the SVM optimization program. Both of these methods are simple and easy to implement. The experimental results show that these methods often outperform ordinary SVMs.

This results are the first work to use dropout to improve the performance of SVMs with non-linear kernels. We presented two types of dropout with kernels and experimentally showed their effectiveness. We showed that randomized kernel approximation may be used along with marginalized dropout in primal to improve both the performance and efficiency of kernel machines.

CHAPTER VI

CONCLUSION AND FUTURE DIRECTIONS

This thesis presents novel convex optimization algorithms for learning robust large margin models. Our methods rely on formulating the machine learning problems as mathematical optimization programs that can be efficiently solved. In all of our contributions, we started from a conceptual formulation of the problem and converted it to a manageable and convex problem, which can be solved by off-the-shelf convex optimization methods.

6.1. Summary of contributions

– **Convex adversarial collective classification** Our method robustly performs collective classification in the presence adversary. The formulation is a convex quadratic program that can be effieciently solved. This solution improved the performance of collective classification, even if there was no adversarial component in the test data. Our method consistently outperforms both non-adversarial and non-relational baselines.

– **Equivalency of adversarial robustness and regularization** Our method takes advantage of the adversary's weakness, and converts their weakness to its strength. For each adversary that is capable of altering the feature space, we can derive specific regularization functions that immunes the machine learning algorithm to that type of adversary. Since the method only adds extra convex regularization functions to the objective of the original optimization program, little computation overhead is added. Therefore, the problem can be optimized in the same order as the non-robust optimization program.

- **Robustness of large margin methods through dropout regularization** Average adversaries do not have enough information about the underlying machine learning system, and they do not have ample computation resources to calculate an optimal attack. As a result, they resort to frequent random attacks. Their hope is that some of the random changes in the input data finally tricks the machine learning algorithm. In order to be robust against such adversaries, we can minimize the expected loss function, when data is randomly changing. Dropout training is a great match for such circumstances. We derive the regularization effect of marginalized dropout on linear and non-linear SVMs. Our derivation is simple and convex. Experimentally we show that our method is efficient, and that it almost always outperforms regular SVMs.

6.2. Future directions

The ideal goal is to design a global recipe for robustness that applies to most of the machine learning algorithms; however, only the vulnerability of a few of machine learning algorithms is studied in depth; many algorithms remain unexplored.

6.2.1. Improving Adversarial Machine Learning

The robustness of many of the machine learning algorithms is not studied in depth yet. As we suggest in Algorithm 2, a range of combinations of explicit adversarial and chance-based adverse situations can be studied altogether. Some other future directions in adversarial machine learning are:

- Scaling-up current methods

  Scaling up adversarial methods to large datasets remains an open issue. A promising direction is using online algorithms that are shown to be successful in other fields of machine learning.

- Learning utility functions

  If we can approximate the opponent's utility, then we will have a more realistic model of the adversarial game. In addition, we will be able to use decision theoretic approaches to model non-zero-sum games. Solving non-zero-sum games in adversarial settings is another important issue that needs to be addressed.

- Efficient use of knowledge about the opponent

  We have shown that by taking advantage of adversary's limitations, we can design more robust algorithms; yet, there are still many details about how to translate the raw knowledge about the adversary into useful parameters in the learning algorithm.

All of these items apply to both structured and non-structured output prediction.

6.2.2. Expansion of Existing Work to Structural Settings

There exist many methods in adversarial machine learning that are designed for specific problems. By right abstraction, these methods can be generalized to the wider class of structured output prediction. Good examples of such methods are regret minimization algorithms; these methods are based on elegant mathematical foundations, and they are designed to be robust against adversarial noise. There are only a couple of papers that use regret minimization algorithms for structured output prediction. An important feature of regret minimization algorithms is that they are mostly based on some scalable online algorithm, which is a great candidate for scaling up existing structured prediction algorithms.

On the other hand, regret minimization algorithms can also benefit from the work that is already done in the field of adversarial machine learning. The current regret minimization algorithms assume that the adversary is completely arbitrary[1]. A potential improvement to regret minimization algorithms can be gained by restricting the adversary in a more realistic and practical way.

In this thesis, we derived a formulation for robustness through dropout regularization in ordinary SVMs. This method can be expanded to be applied to structured prediction problems as well. Due to the hardness of the optimization problems of structured learning, this expansion needs more research and is not trivial. However, our promising result on the ordinary SVMs suggests that marginalized dropout should improve structured prediction as well.

[1]Although there are some simple versions of bounded adversaries, which are mostly from the reinforcement learning community, the possible restrictions of the adversary are not studied as comprehensively as it's done in adversarial machine learning.

APPENDIX A

INTEGRALITY OF THE ADVERSARIAL SOLUTION IN CONVEX ADVERSARIAL COLLECTIVE CLASSIFICATION

Lemma 7. *For K=2, any fixed j and $0 \leq x_{ij}, y_i^k \leq 1$, $\hat{y}_i^k \in \{0,1\}$,$\sum_k y_i^k = 1$ and $\sum_k \hat{y}_i^k = 1$, if $A_j^k = \sum_{i=1}^N \min(x_{ij}, y_i^k) - x_{ij}\hat{y}_i^k$, then $\sum_{k=1}^K A_j^K \geq 0$.*

Proof. $A_j^1 + A_j^2 = \sum_{i=1}^N \min(x_{ij}, y_i^1) - x_{ij}\hat{y}_i^1 + \min(x_{ij}, y_i^2) - x_{ij}\hat{y}_i^2$. Since $y_i^1 + y_i^2 = 1$ and $\hat{y}_i^1 + \hat{y}_i^2 = 1$, we can rewrite it as $\sum_{i=1}^N \min(x_{ij}, y_i^1) - x_{ij}(\hat{y}_i^1 + \hat{y}_i^2) + \min(x_{ij}, 1 - y_i^1)$ $= \sum_{i=1}^N \min(x_{ij}, y_i^1) + \min(x_{ij}, 1 - y_i^1) - x_{ij}$. Now three cases can happen:

(a) If $x_{ij} \geq \max(y_i^1, 1 - y_i^1)$, then $\min(x_{ij}, y_i^1) + \min(x_{ij}, 1 - y_i^1) - x_{ij} = y_i^1 + 1 - y_i^1 - x_{ij}$ $= 1 - x_{ij} \geq 0$.

(b) If $\min(y_i^1, 1 - y_i^1) \leq x_{ij} \leq \max(y_i^1, 1 - y_i^1)$, then $\min(x_{ij}, \min(y_i^1, 1 - y_i^1)) + \min(x_{ij}, \max(y_i^1, 1 - y_i^1)) - x_{ij} = \min(x_{ij}, \min(y_i^1, 1 - y_i^1)) + x_{ij} - x_{ij} = \min(x_{ij}, y_i^1, 1 - y_i^1) \geq 0$.

(c) If $x_{ij} \leq \min(y_i^1, 1 - y_i^1)$, then $\min(x_{ij}, y_i^1) + \min(x_{ij}, 1 - y_i^1) - x_{ij} = x_{ij} + x_{ij} - x_{ij}$ $= x_{ij} \geq 0$.

Therefore $\min(x_{ij}, y_i^1) + \min(x_{ij}, y_i^2) - x_{ij}$ is always nonnegative and consequently $A_j^1 + A_j^2 = \sum_{i=1}^N \min(x_{ij}, y_i^1) - x_{ij}\hat{y}_i^1 + \min(x_{ij}, y_i^2) - x_{ij}\hat{y}_i^2$ is always nonnegative. □

Lemma 8. *For $K = 2$, in the optimal solution of the final quadratic program, W^* satisfies the following property: $\min(w_j^1, w_j^2) = 0\ \forall j = 1 \ldots m$.*

Proof. Let $\theta_j = \min(w_j^1, w_j^2)$, we define $u_j^1 = w_j^1 - \theta_j$ and $u_j^2 = w_j^2 - \theta_j$, by substitution the objective of the constraint's linear program will be:

$$\sum_{i,j,k}(u_j^k+\theta_j)z_{ij}^k-(u_j^k+\theta_j)x_{ij}\hat{y}_i^k+\underbrace{\sum_{(i,j)\in E,k}w_e^k y_{ij}^k-\sum_{i,k}y_i^k\cdot\hat{y}_i^k+\sum_{i,j}\delta_{ij}(1-2\hat{x}_{ij})x_{ij}}_{B}$$

$$=\sum_j\sum_i u_j^1z_{ij}^1-u_j^1x_{ij}\hat{y}_i^1+u_j^2z_{ij}^2-u_j^2x_{ij}\hat{y}_i^2+\theta_j(z_{ij}^1-x_{ij}\hat{y}_i^1+z_{ij}^2-x_{ij}\hat{y}_i^2)+B$$

$$=\sum_j\left[\sum_i F_{ij}+\theta_j\underbrace{\sum_i H_ij}_{\geq 0}\right]+B$$

In which F_{ij} and H_{ij} are:

$$\begin{aligned}F_{ij}&=&u_j^1z_{ij}^1-u_j^1x_{ij}\hat{y}_i^1+u_j^2z_{ij}^2-u_j^2x_{ij}\hat{y}_i^2\\H_{ij}&=&z_{ij}^1-x_{ij}\hat{y}_i^1+z_{ij}^2-x_{ij}\hat{y}_i^2\end{aligned}$$

According to Lemma 7, $\sum_i(z_{ij}^1-x_{ij}\hat{y}_i^1+z_{ij}^2-x_{ij}\hat{y}_i^2)\geq 0$, therefore the coefficient of each θ_j is non-negative. Since $\theta_j=\min(w_j^1,w_j^1)\geq 0$, thus:

i. If the optimization algorithm chooses smaller value for θ_j, the relaxed inequality constraint will not be violated, and also smaller θ_j will not imply larger ξ.

ii. A smaller θ_j will directly reduce the objective value.

Therefore, the optimization algorithm chooses the smallest possible θ_j, which is $\theta_j=0\ \forall j$. So $\min(w_j^1,w_j^2)=0$ or equivalently $w_j^1w_j^2=0\ \forall j=1\ldots m$. □

Theorem 8. *Adversary's problem in Equation 3.4, has integral solution for both* $\mathbf{X}$ *and* Y.

Proof. According to Lemma 8, we know that $\min(w_j^1, w_j^2) = 0$ for all j. So we can rewrite Equation 3.4 as:

$$\max_{\mathbf{y}\in\mathcal{Y}', 0\le\mathbf{x}\le 1} \sum_{i,j} D_{ij} + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k - \sum_{i,k} y_i^k \cdot \hat{y}_i^k + \sum_{i,j} \delta_{ij}(1-2\hat{x}_{ij})x_{ij} \qquad \text{(Equation A.1)}$$

Where $D_{ij} = w_j^1 z_{ij}^1 - w_j^1 x_{ij}\hat{y}_i^1 + w_j^2 z_{ij}^2 - w_j^2 x_{ij}\hat{y}_i^2$. Here we assume that either w_j^1 or w_j^2 is not zero. Because this is the interesting case, otherwise the proof is trivial. Therefore, since either w_j^1 or w_j^2 is zero. We have:

$$\begin{aligned}
D_{ij} &= w_j^1 \min(x_{ij}, y_i^1) - w_j^1 x_{ij}\hat{y}_i^1 + w_j^2 \min(x_{ij}, y_i^2) - w_j^2 x_{ij}\hat{y}_i^2 \\
&= I(w_j^1 = 0)\left[w_j^1 \min(1-x_{ij}, y_i^1) - w_j^1(1-x_{ij})\hat{y}_i^1 + w_j^2 \min(x_{ij}, y_i^2) - w_j^2 x_{ij}\hat{y}_i^2\right] + \\
&\quad I(w_j^2 = 0)\left[w_j^1 \min(x_{ij}, y_i^1) - w_j^1 x_{ij}\hat{y}_i^1 + w_j^2 \min(1-x_{ij}, y_i^2) - w_j^2(1-x_{ij})\hat{y}_i^2\right]
\end{aligned}$$

Let $v_{ij}^k = x_{ij} I(w_j^k > 0) + (1-x_{ij}) I(w_j^k = 0)$, where $I(.)$ is the indicator function, then:

$$\begin{aligned}
D_{ij} &= I(w_j^1 = 0)\left[w_j^1 \min(v_{ij}^1, y_i^1) - w_j^1 v_{ij}^1\hat{y}_i^1 + w_j^2 \min(v_{ij}^2, y_i^2) - w_j^2 v_{ij}^2\hat{y}_i^2\right] + \\
&\quad I(w_j^2 = 0)\left[w_j^1 \min(v_{ij}^1, y_i^1) - w_j^1 v_{ij}^1\hat{y}_i^1 + w_j^2 \min(v_{ij}^2, y_i^2) - w_j^2 v_{ij}^2\hat{y}_i^2\right] \\
&= \big(I(w_j^1 = 0) + I(w_j^2 = 0)\big)\left[w_j^1 \min(v_{ij}^1, y_i^1) - w_j^1 v_{ij}^1\hat{y}_i^1 + w_j^2 \min(v_{ij}^2, y_i^2) - w_j^2 v_{ij}^2\hat{y}_i^2\right] \\
&= w_j^1 \min(v_{ij}^1, y_i^1) - w_j^1 v_{ij}^1\hat{y}_i^1 + w_j^2 \min(v_{ij}^2, y_i^2) - w_j^2 v_{ij}^2\hat{y}_i^2 \qquad \text{(Equation A.2)}
\end{aligned}$$

Clearly, we $v_{ij}^1 + v_{ij}^2 = 1$, because:

$$\begin{aligned}
v_{ij}^1 + v_{ij}^2 &= x_{ij} I(w_j^1 > 0) + (1 - x_{ij}) I(w_j^1 = 0) + x_{ij} I(w_j^2 > 0) + (1 - x_{ij}) I(w_j^2 = 0) \\
&= x_{ij} \underbrace{\left[I(w_j^1 > 0) + I(w_j^2 > 0) \right]}_{=1} + (1 - x_{ij}) \underbrace{\left[I(w_j^1 = 0) + I(w_j^2 = 0) \right]}_{=1} \\
&= x_{ij} + 1 - x_{ij} = 1.
\end{aligned}$$

Obviously, as a result we will have $z_{ij}^k = \min(v_{ij}^k, y_i^k)$, because otherwise increasing z_{ij}^k can increase the objective, so the solver program will choose the maximum possible value for z_{ij}^k. By Lemma 9, and reformulation of suggested D_{ij} in Equation A.2, we conclude that Equation A.1 has integral solution for y_i^k and v_{ij}^k for all i, j and $k = 1, 2$. Since inetgrality of v_{ij}^k implies integrality of x_{ij}, proof is complete. $\square$

Lemma 9. *If K=2, for any* $W = [W^1, W^2]$, $W^k = [w_1^k, \dots, w_m^k]^T$*, linear program in Equation A.1, has an integral solution.*

Proof. Here, our argument is similar to the proof of the Theorem 1 of Taskar et al. (2004a). We show that for any fractional solution $\mathbf{X}$ (and respectively $\mathbf{V}$) and Y of Equation A.1, we can construct a new feasible integral assignment $\mathbf{X}'$ and Y', that increases the objective or does not change it.

Since all w_e^k's and w_j^k's are positive, therefore, $y_{ij}^k = \min(y_i^k, y_j^k)$ and $z_{ij}^k = \min(y_i^k, x_{ij})$; this means that the slack variables corresponding to $z_{ij}^k \leq y_i^k$,$z_{ij}^k \leq x_{ij}$ and $y_{ij}^k \leq y_i^k$,$y_{ij}^k \leq y_j^k$ are zero, because otherwise by increasing y_{ij}^k or z_{ij}^k, the objective could be increased.

Let $\lambda^k = \min(\min_{i, y_i^k > 0} y_i^k, \min_{ij, v_{ij}^k > 0} v_{ij}^k)$ and $\lambda = \lambda^1$ or $\lambda = -\lambda^2$. We propose a new construction of solution, that either increases the objective or does not change it, and at the same time reduces the number of fractional values in the solution.

$$\begin{aligned}
v_{ij}^{'1} &= v_{ij}^1 - \lambda I(0 < v_{ij}^1 < 1),\ v_{ij}^{'2} = v_{ij}^2 + \lambda I(0 < v_{ij}^2 < 1) \\
z_{ij}^{'1} &= z_{ij}^1 - \lambda I(0 < z_{ij}^1 < 1),\ z_{ij}^{'2} = z_{ij}^2 + \lambda I(0 < z_{ij}^2 < 1) \\
y_i^{'1} &= y_i^1 - \lambda I(0 < y_i^1 < 1),\ \ y_i^{'2} = y_i^2 + \lambda I(0 < y_i^2 < 1) \\
y_{ij}^{'1} &= y_{ij}^1 - \lambda I(0 < y_{ij}^1 < 1),\ y_{ij}^{'2} = y_{ij}^2 + \lambda I(< 0 y_{ij}^2 < 1)
\end{aligned}$$

It is obvious that by this update, at least two of the fractional values become integral. First, we show that in this new construction, values remain feasible. So we need to show that $v_{ij}^{'1} + v_{ij}^{'2} = 1$,$y_i^{'1} + y_i^{'2} = 1$, $v_{ij}^{'k} \geq 0$, $y_i^{'k} \geq 0$, $y_{ij}^{'k} = \min(y_i^{'k}, y_j^{'k})$ and $z_{ij}^{'k} = \min(v_{ij}^{'k}, y_i^{'k})$. In the following we show that all of the feasibility requirements are satisfied.

$$v_{ij}^{'1} + v_{ij}^{'2} = v_{ij}^1 - \lambda I(0 < v_{ij}^1 < 1) + v_{ij}^2 + \lambda I(0 < v_{ij}^2 < 1 = v_{ij}^1 + v_{ij}^2 = 1.$$

$$y_i^{'1} + y_i^{'2} = y_i^1 - \lambda I(0 < y_i^1 < 1) + y_i^2 + \lambda I(0 < y_i^2 < 1) = y_i^1 + y_i^2 = 1.$$

Above we used the fact that if v_{ij}^1 is fractional, then v_{ij}^2 will also be fractional, and similarly if y_i^1 is fractional then y_i^2 will also be fractional, since $v_{ij}^1 + v_{ij}^2 = 1$ and $y_i^1 + y_i^2 = 1$. To show $v_{ij}^{'k} \geq 0$ and $y_i^{'k} \geq 0$, we prove that $\min_{ij}\ v_{ij}^{'k} \geq 0$ and $\min_i\ y_i^{'k} \geq 0$.

$$\begin{aligned}
\min_{ij}\ v_{ij}^{\prime k} &= \min_{ij}(v_{ij}^k - (\min(\min_{i,y_i^k>0} y_i^k, \min_{ij,v_{ij}^k>0} v_{ij}^k))I(0 < v_{ij}^k < 1)) \\
&= \min\left(\min_{ij}\ v_{ij}^k, \min_{ij}\left[v_{ij}^k - (\min(\min_{i,y_i^k>0} y_i^k, \min_{ij,v_{ij}^k>0} v_{ij}^k))\right]\right) \\
&\geq \min\left(\min_{ij}\ v_{ij}^k, \min_{ij}\left[v_{ij}^k - (\min_{ij,v_{ij}^k>0} v_{ij}^k)\right]\right) \\
&\geq \min_{ij}\left[v_{ij}^k - (\min_{ij,v_{ij}^k>0} v_{ij}^k)\right] = 0.
\end{aligned}$$

$$\begin{aligned}
\min_{i}\ y_i^{\prime k} &= \min_{i}(y_i^k - (\min(\min_{i,y_i^k>0} y_i^k, \min_{ij,v_{ij}^k>0} v_{ij}^k))I(0 < y_i^k < 1)) \\
&= \min\left(\min_{i}\ y_i^k, \min_{i}\left[y_i^k - (\min(\min_{i,y_i^k>0} y_i^k, \min_{ij,v_{ij}^k>0} v_{ij}^k))\right]\right) \\
&\geq \min\left(\min_{i}\ y_i^k, \min_{i}\left[y_i^k - (\min_{i,y_i^k>0} y_i^k)\right]\right) \\
&\geq \min_{i}\left[y_i^k - (\min_{i,y_i^k>0} y_i^k)\right] = 0.
\end{aligned}$$

The last step in showing that the proposed construction is feasible is showing that $y_{ij}^{\prime k} = \min(y_i^{\prime k}, y_j^{\prime k})$ and $z_{ij}^{\prime k} = \min(v_{ij}^{\prime k}, y_i^{\prime k})$.

$$\begin{aligned}
y_{ij}^{\prime 1} &= y_{ij}^1 - \lambda I(0 < y_{ij}^1 < 1) \\
&= \min(y_i^1, y_j^1) - \lambda I(0 < \min(y_i^1, y_j^1) < 1) \\
&= \min(y_i^1 - \lambda I(0 < y_i^1 < 1), y_j^1 - \lambda I(0 < y_j^1 < 1)) \\
&= \min(y_i^{\prime 1}, y_j^{\prime 1}).
\end{aligned}$$

$$\begin{aligned}
y_{ij}^{'2} &= y_{ij}^2 + \lambda I(0 < y_{ij}^2 < 1) \\
&= \min(y_i^2, y_j^2) + \lambda I(0 < \min(y_i^2, y_j^2) < 1) \\
&= \min(y_i^2 + \lambda I(0 < y_i^2 < 1), y_j^2 + \lambda I(0 < y_j^2 < 1)) \\
&= \min(y_i^{'2}, y_j^{'2}).
\end{aligned}$$

$$\begin{aligned}
z_{ij}^{'1} &= z_{ij}^1 - \lambda I(0 < z_{ij}^1 < 1) \\
&= \min(v_{ij}^1, y_i^1) - \lambda I(0 < \min(v_{ij}^1, y_i^1) < 1) \\
&= \min(v_{ij}^1 - \lambda I(0 < v_{ij}^1 < 1), y_i^1 - \lambda I(0 < y_i^1 < 1)) \\
&= \min(v_{ij}^{'1}, y_i^{'1}).
\end{aligned}$$

$$\begin{aligned}
z_{ij}^{'2} &= z_{ij}^2 + \lambda I(0 < z_{ij}^2 < 1) \\
&= \min(v_{ij}^2, y_i^2) + \lambda I(0 < \min(v_{ij}^2, y_i^2) < 1) \\
&= \min(v_{ij}^2 + \lambda I(0 < v_{ij}^2 < 1), y_i^2 + \lambda I(0 < y_i^2 < 1)) \\
&= \min(v_{ij}^{'2}, y_i^{'2}).
\end{aligned}$$

So far we have shown that the new variable construction is feasible, and it remains to show that we can increase the objective. We substitute the newly constructed feasible values in Equation A.1 and subtract the objective with unchanged values from it. Then we show that with proper choice of $\lambda = \lambda^1$ or of $\lambda = -\lambda^2$, we can improve the objective.

$$\begin{aligned}
V_{old} &= \sum_{i,j} D_{ij} + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k - \sum_{i,k} y_i^k \cdot \hat{y}_i^k + \sum_{i,j} \delta_{ij}(1-2\hat{x}_{ij})x_{ij} \\
&= \sum_{i,j} w_j^1 z_{ij}^1 - w_j^1 v_{ij}^1 \hat{y}_i^1 + w_j^2 z_{ij}^2 - w_j^2 v_{ij}^2 \hat{y}_i^2 \\
&\quad + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k - \sum_{i,k} y_i^k \cdot \hat{y}_i^k + \sum_{i,j} \delta_{ij}(1-2\hat{x}_{ij})x_{ij} \\
&= \sum_{i,j} w_j^1 z_{ij}^1 - w_j^1 v_{ij}^1 \hat{y}_i^1 + w_j^2 z_{ij}^2 - w_j^2 v_{ij}^2 \hat{y}_i^2 \\
&\quad + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \\
&\quad + \sum_{i,j} \delta_{ij}(1-2\hat{x}_{ij}) \left[\left(I(w_j^1>0) - I(w_j^1=0)\right) v_{ij}^1 + I(w_j^1=0)\right] \\
&= \sum_{i,j} w_j^1 z_{ij}^1 - w_j^1 v_{ij}^1 \hat{y}_i^1 + w_j^2 z_{ij}^2 - w_j^2 v_{ij}^2 \hat{y}_i^2 \\
&\quad + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \\
&\quad + \sum_{i,j} \left[\delta_{ij}(1-2\hat{x}_{ij}) \left(I(w_j^1>0) - I(w_j^1=0)\right)\right] v_{ij}^1 + C.
\end{aligned}$$

Above we have used the fact that $x_{ij} = I(w_j'^k > 0)v_{ij}'^k + I(w_j'^k = 0)(1 - v_{ij}'^k) = I(w_j'^1 > 0)v_{ij}'^1 + I(w_j'^1 = 0)(1 - v_{ij}'^1) = \left(I(w_j'^1 > 0) - I(w_j'^1 = 0)\right) v_{ij}'^1 + I(w_j'^1 = 0)$.

$$
\begin{aligned}
V_{new} &= \sum_{i,j} w_j^1 z_{ij}'^1 - w_j^1 v_{ij}'^1 \hat{y}_i^1 + w_j^2 z_{ij}'^2 - w_j^2 v_{ij}'^2 \hat{y}_i^2 \\
&\quad + \sum_{(i,j)\in E,k} w_e^k y_{ij}'^k - \sum_{i,k} y_i'^k \cdot \hat{y}_i^k \\
&\quad + \sum_{i,j} \left[\delta_{ij}(1-2\hat{x}_{ij})\left(I(w_j^1>0) - I(w_j^1=0)\right)\right] v_{ij}'^1 + C \\
&= \sum_{i,j} [w_j^1(z_{ij}^1 - \lambda I(0<z_{ij}^1<1)) - w_j^1 \hat{y}_i^1 (v_{ij}^1 - \lambda I(0<v_{ij}^1<1)) \\
&\qquad + w_j^2(z_{ij}^2 + \lambda I(0<z_{ij}^2<1)) - w_j^2 \hat{y}_i^2 (v_{ij}^2 + \lambda I(0<v_{ij}^2<1))] \\
&\quad + \sum_{(i,j)\in E} \left[w_e^1(y_{ij}^1 - \lambda I(0<y_{ij}^1<1)) + w_e^2(y_{ij}^2 + \lambda I(0<y_{ij}^2<1))\right] \\
&\quad - \sum_i \hat{y}_i^1 \cdot (y_i^1 - \lambda I(0<y_i^1<1)) + \hat{y}_i^2 \cdot (y_i^2 + \lambda I(0<y_i^2<1)) \\
&\quad + \sum_{i,j} \left[\delta_{ij}(1-2\hat{x}_{ij})\left(I(w_j^1>0) - I(w_j^1=0)\right)\right] (v_{ij}^1 - \lambda I(0<v_{ij}^1<1)) + C \\
&= V_{old} + \sum_{i,j} [w_j^1(-\lambda I(0<z_{ij}^1<1)) - w_j^1 \hat{y}_i^1(-\lambda I(0<v_{ij}^1<1)) \\
&\qquad + w_j^2(\lambda I(0<z_{ij}^2<1)) - w_j^2 \hat{y}_i^2(\lambda I(0<v_{ij}^2<1))] \\
&\quad + \sum_{(i,j)\in E} \left[w_e^1(-\lambda I(0<y_{ij}^1<1)) + w_e^2(\lambda I(0<y_{ij}^2<1))\right] \\
&\quad - \sum_i \hat{y}_i^1 \cdot (-\lambda I(0<y_i^1<1)) + \hat{y}_i^2 \cdot (+\lambda I(0<y_i^2<1)) \\
&\quad + \sum_{i,j} \left[\delta_{ij}(1-2\hat{x}_{ij})\left(I(w_j^1>0) - I(w_j^1=0)\right)\right] (-\lambda I(0<v_{ij}^1<1)).
\end{aligned}
$$

Therefore, we can write $V_{new} - V_{old}$ as:

$$\begin{aligned}
V_{new} - V_{old} &= \lambda[\sum_{i,j}[-w_j^1 I(0 < z_{ij}^1 < 1) + w_j^1 \hat{y}_i^1 I(0 < v_{ij}^1 < 1) \\
&\quad + w_j^2 I(0 < z_{ij}^2 < 1) - w_j^2 \hat{y}_i^2 I(0 < v_{ij}^2 < 1)] \\
&\quad + \sum_{(i,j)\in E} \left[-w_e^1 I(0 < y_{ij}^1 < 1) + w_e^2 I(0 < y_{ij}^2 < 1)\right] \\
&\quad - \sum_i \hat{y}_i^1 \cdot (-I(0 < y_i^1 < 1)) + \hat{y}_i^2 \cdot (+I(0 < y_i^2 < 1)) \\
&\quad + \sum_{i,j} -\delta_{ij}(1 - 2\hat{x}_{ij}) \left(I(w_j^1 > 0) - I(w_j^1 = 0)\right) I(0 < v_{ij}^1 < 1)] \\
&= \lambda D.
\end{aligned}$$

The change in objective is λD, and since D is constant with respect to λ, by choosing $\lambda = -\lambda^2$ for negative D, or $\lambda = \lambda^1$ for positive D, we can always have positive or zero λD. It means that the integral solution will increase the objective or will not change it, while leaving fewer fractional values.

□

APPENDIX B

PROOFS FOR EQUIVALENCE OF ROBUSTNESS AND REGULARIZATION IN LARGE MARGIN METHODS

Proof of Lemma 3:

Proof. We form $\delta^{\mathcal{C}}_{\tilde{\mathbf{y}}}(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}})$ from Equation 4.7:

$$\delta^{\mathcal{C}} = \delta^{\mathcal{C}}_{\tilde{\mathbf{y}}}(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}}) =$$

$$\phi_{\mathcal{C}}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \phi_{\mathcal{C}}(\tilde{\mathbf{x}}, \mathbf{y}) - \phi_{\mathcal{C}}(\mathbf{x}, \tilde{\mathbf{y}}) - \phi_{\mathcal{C}}(\mathbf{x}, \mathbf{y}) =$$

$$\sum_{(c_x, c_y) \in \mathcal{C}} (\prod_{i \in c_x} \tilde{\mathbf{x}}_i - \prod_{i \in c_x} \mathbf{x}_i)(\prod_{i \in c_y} \tilde{\mathbf{y}}_i - \prod_{i \in c_y} \mathbf{y}_i) \qquad \text{(Equation B.1)}$$

For an individual elements of the vector δ as expanded in (Equation B.1), we can apply Hölder's inequality to the right-hand side:

$$|\delta^{\mathcal{C}}| \leq$$

$$(\sum_{c_x \in \mathcal{C}} |\prod_{i \in c_x} \tilde{\mathbf{x}}_i - \prod_{i \in c_x} \mathbf{x}_i|^p)^{\frac{1}{p}} (\sum_{c_y \in \mathcal{C}} |\prod_{i \in c_y} \tilde{\mathbf{y}}_i - \prod_{i \in c_y} \mathbf{y}_i|^q)^{\frac{1}{q}}$$

where $\frac{1}{p} + \frac{1}{q} = 1$. Since $|\prod_{i \in c_y} \tilde{\mathbf{y}}_i - \prod_{i \in c_y} \mathbf{y}_i|^q \leq 1$, we will have: $\sum_{c_y \in \mathcal{C}} |\prod_{i \in c_y} \tilde{\mathbf{y}}_i - \prod_{i \in c_y} \mathbf{y}_i|^q \leq |\mathcal{C}|$, therefore:

$$|\delta^{\mathcal{C}}| \leq |\mathcal{C}|^{\frac{1}{q}} (\sum_{c_x \in \mathcal{C}} |\prod_{i \in c_x} \tilde{\mathbf{x}}_i - \prod_{i \in c_x} \mathbf{x}_i|^p)^{\frac{1}{p}}$$

After applying Lemma 10 and raising both sides of the inequality to the power of p, we will have:

$$\begin{aligned} |\delta^{\mathcal{C}}|^p \leq |\mathcal{C}|^{\frac{p}{q}}(\alpha \sum_{c_x \in \mathcal{C}} \sum_{i \in c_x} |\tilde{\mathbf{x}}_i - \mathbf{x}_i|^p) & \\ \Rightarrow \quad \frac{|\delta^{\mathcal{C}}|^p}{\alpha |\mathcal{C}|^{\frac{p}{q}}} \leq \sum_{c_x \in \mathcal{C}} \sum_{i \in c_x} |\tilde{\mathbf{x}}_i - \mathbf{x}_i|^p & \qquad \text{(Equation B.2)} \end{aligned}$$

where $\alpha = \max_{c_x \in \mathcal{C}} |c_x|^{(p-1)}$, and $|c_x|$ is the number of variables in c_x. $\square$

The proof of Lemma 3 depends on the following lemma:

Lemma 10. *For any sequence $a_1, \ldots, a_n, b_1, \ldots, b_n$, such that $0 \leq a_i, b_j \leq 1$, we have $|\prod_{i=1}^n a_i - \prod_{i=1}^n b_i|^p \leq n^{(p-1)} \sum_{i=1}^n |a_i - b_i|^p$.*

Proof. For $n = 1$, the inequality is trivial. Let $u_1 = \prod_{i=1}^{\lfloor n/2 \rfloor} a_i$, $u_2 = \prod_{i=1}^{\lfloor n/2 \rfloor} b_i$, $v_1 = \prod_{i=\lfloor n/2 \rfloor + 1}^{n} a_i$, and $v_2 = \prod_{i=\lfloor n/2 \rfloor + 1}^{n} a_i$. Also it is a known fact that $|f + g|^p \leq 2^{p-1}(|f|^p + |g|^p) \;\; g, f \in \mathbf{R}$. We have:

$$\begin{aligned} |\prod_{i=1}^n a_i - \prod_{i=1}^n b_i|^p &= |u_1 v_1 - u_2 v_2|^p \\ &= |u_1 v_1 - u_1 v_2 + u_1 v_2 - u_2 v_2|^p \\ &\leq 2^{p-1}(|u_1 v_1 - u_1 v_2|^p + |u_1 v_2 - u_2 v_2|^p) \\ &= 2^{p-1}(u_1^p |v_1 - v_2|^p + v_2^p |u_1 - u_2|^p) \\ &\leq 2^{p-1}(|v_1 - v_2|^p + |u_1 - u_2|^p) \end{aligned}$$

by recursive application of the above procedure, the products can be decomposed at most $\log_2 n$ times. Therefore,

$$\begin{aligned} |\prod_{i=1}^{n} a_i - \prod_{i=1}^{n} b_i|^p &\leq 2^{(p-1)\log_2 n} \sum_{i=1}^{n} |a_i - b_i|^p \\ &= n^{p-1} \sum_{i=1}^{n} |a_i - b_i|^p \end{aligned}$$

□

Proof of Corollary 1:

Proof. We begin with the result of Theorem 4.3, where $\frac{1}{B(d\alpha_i)^{\frac{1}{p}}|\mathcal{C}_i|^{\frac{1}{q}}}$ is the coeficient of variations in the feature corresponding to clique $\mathcal{C}_i$. Since $p = 1$ then $q = \infty$, and $\alpha_i = \max_{c_x \in \mathcal{C}_i} |c_x|^{(p-1)} = 1$:

$$\begin{aligned} \frac{1}{B(d\alpha_i)^{\frac{1}{p}}|\mathcal{C}_i|^{\frac{1}{q}}} &= \frac{1}{Bd|\mathcal{C}_i|^{\frac{1}{\infty}}} \\ &= \frac{1}{Bd} \end{aligned}$$

Also in (Equation B.2), set $p = 1$ and $q = \infty$.

$$\begin{aligned} &|\delta^{\mathcal{C}}| \leq \\ &(\sum_{c_x \in \mathcal{C}} |\prod_{i \in c_x} \tilde{\mathbf{x}}_i - \prod_{i \in c_x} \mathbf{x}_i|) \max_{c_y \in \mathcal{C}} |\prod_{i \in c_y} \tilde{\mathbf{y}}_i - \prod_{i \in c_y} \mathbf{y}_i| \end{aligned}$$

Since $\max_{c_y \in \mathcal{C}} |\prod_{i \in c_y} \tilde{\mathbf{y}}_i - \prod_{i \in c_y} \mathbf{y}_i| = 1$, we will be using a tighter upper-bound. □

Proof of Proposition 1:

Proof. We prove the case when regularization function is $\|\mathbf{w}\| = \|\mathbf{w}\|_\infty$ (the proofs for $\|\mathbf{M}^{-1}\mathbf{w}\|_\infty$ and $\|\mathbf{M}^{-1}\mathbf{w}\|_1$ are very similar, but for simplicity we chose this case). Recall that the optimization program of the robust structural SVM is:

$$\begin{aligned}&\underset{\mathbf{w},\xi}{\text{minimize}}\ c_1 f(\mathbf{w}) + c_2\|\mathbf{w}\|_\infty + \xi \quad \text{subject to} \qquad \text{(Equation B.3)}\\ &\xi \geq \max_{\tilde{\mathbf{y}}}\ \mathbf{w}^T(\phi(\mathbf{x},\tilde{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y})) + \Delta(\mathbf{y},\tilde{\mathbf{y}})\end{aligned}$$

It can be re-written as:

$$\begin{aligned}&\underset{\mathbf{w},\xi,t}{\text{minimize}}\ c_1 f(\mathbf{w}) + c_2 t + \xi \quad \text{subject to}\\ &\xi \geq \max_{\tilde{\mathbf{y}}}\ \mathbf{w}^T(\phi(\mathbf{x},\tilde{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y})) + \Delta(\mathbf{y},\tilde{\mathbf{y}})\\ &w_i \leq t,\ \ -w_i \leq t\ \ \forall w_i\end{aligned}$$

In vector form we can write these constraints as: $\mathbf{w} \leq \mathbf{1}t$ and $-\mathbf{w} \leq \mathbf{1}t$. Clearly, there are two vectors $\mathbf{s_1}$ and $\mathbf{s_2}$ for which:

$$\begin{aligned}\mathbf{w} + \mathbf{s_1} = \mathbf{1}t \quad &\Rightarrow \quad \mathbf{w} = \mathbf{1}t - \mathbf{s_1}\\ -\mathbf{w} + \mathbf{s_2} = \mathbf{1}t \quad &\Rightarrow \quad \mathbf{w} = \mathbf{s_2} - \mathbf{1}t\end{aligned}$$

Let $\gamma = [\mathbf{s_1}^T\ \mathbf{s_2}^T\ t]^T$, $m = \dim \mathbf{w}$, $\mathbf{I_{s_1}} = [\mathbf{I}_{m\times m}\ \underline{\mathbf{0}}_{m\times m}\ \mathbf{0}_{m\times 1}]$, $\mathbf{I_{s_2}} = [\underline{\mathbf{0}}_{m\times m}\ \mathbf{I}_{m\times m}\ \mathbf{0}_{m\times 1}]$, and $\mathbf{I_t} = [\underline{\mathbf{0}}_{1\times m}\ \underline{\mathbf{0}}_{1\times m}\ 1]$. (i.e. $\mathbf{s_1} = \mathbf{I_{s_1}}\gamma$, $\mathbf{s_2} = \mathbf{I_{s_2}}\gamma$, $t = \mathbf{I_t}\gamma$). By substitution:

$$\begin{aligned}\mathbf{w} &= \mathbf{1I_t}\gamma - \mathbf{I_{s_1}}\gamma = (\mathbf{1I_t} - \mathbf{I_{s_1}})\gamma\\ \mathbf{w} &= \mathbf{I_{s_2}}\gamma - \mathbf{1I_t}\gamma = (\mathbf{I_{s_2}} - \mathbf{1I_t})\gamma\end{aligned}$$

which implies $(\mathbf{1I_t} - \mathbf{I_{s_1}})\gamma = (\mathbf{I_{s_2}} - \mathbf{1I_t})\gamma$, therefore: $(2 * \mathbf{1I_t} - \mathbf{I_{s_1}} - \mathbf{I_{s_2}})\gamma = \mathbf{0}$, or equivalently $\gamma \in \mathcal{N}(2 * \mathbf{1I_t} - \mathbf{I_{s_1}} - \mathbf{I_{s_2}})$, where $\mathcal{N}(.)$ returns the null-space of the input matrix. Let columns of matrix $\mathbf{B}$ span $\mathcal{N}(2 * \mathbf{1I_t} - \mathbf{I_{s_1}} - \mathbf{I_{s_2}})$, also let $\gamma = \mathbf{B}\lambda$, we will have $\mathbf{w} = (\mathbf{1I_t} - \mathbf{I_{s_1}})\mathbf{B}\lambda$. Let $\mathbf{A} = \mathbf{B}^T(\mathbf{1I_t} - \mathbf{I_{s_1}})^T$ and $\mathbf{b} = \mathbf{I_t}^T$, then we can rewrite Problem (Equation B.3) as:

$$\begin{aligned} &\underset{\lambda \geq 0, \xi}{\text{minimize}}\ c_1 f(\mathbf{A}^T\lambda) + c_2\mathbf{b}^T\lambda + \xi \quad \text{subject to} \\ &\xi \geq \max_{\tilde{\mathbf{y}}}\ \lambda^T\mathbf{A}(\phi(\mathbf{x}, \tilde{\mathbf{y}}) - \phi(\mathbf{x}, \mathbf{y})) + \Delta(\mathbf{y}, \tilde{\mathbf{y}}) \end{aligned}$$

Note that since $(2 * \mathbf{1I_t} - \mathbf{I_{s_1}} - \mathbf{I_{s_2}})\mathbf{B} = \underline{\mathbf{0}}$, we will have $(\mathbf{1I_t} - \mathbf{I_{s_1}})\mathbf{B} = (\mathbf{I_{s_2}} - \mathbf{1I_t})\mathbf{B}$, and A can be the transpose of any of them. □

APPENDIX C

DIRECT PROOF FOR DERIVATION OF MARGINALIZED LINEAR SVM

In the following, we provide a direct proof for the asymptotic results of marginalizing the linear SVMs.

Let $f_e(w, b)$ be the exact dropout-marginalization of the regularized hinge loss:

$$\begin{aligned} f_e(w,b) &= \lambda\|w\| + \sum_i \mathbb{E}_\xi \max(0, 1 - y_i(w^T x_i \odot \xi/(1-\delta) + b)) \\ &= \lambda\|w\| + \sum_i \mathbb{E}_\xi (1 - y_i(w^T x_i \odot \xi/(1-\delta) + b))\mathbb{I}(1 - y_i(w^T x_i \odot \xi/(1-\delta) + b) \geq 0) \end{aligned}$$

where $\mathbb{I}$ is the step function (i.e. $\mathbb{I}(True) = 1$ and $\mathbb{I}(False) = 0$). Let $l(w, b) = \mathbb{E}_\xi(1 - y_i(w^T x_i \odot \xi/(1-\delta) + b))\mathbb{I}(1 - y_i(w^T x_i \odot \xi/(1-\delta) + b) \geq 0)$. If $z_i = 1 - y_i(w^T x_i \odot \xi/(1-\delta) + b) \sim \mathcal{N}(\mu_i, \sigma_i^2)$, then:

$$\begin{aligned} l(w,b) &= \mathbb{E}_\xi z_i \mathbb{I}(z_i \geq 0) = \int_{-\infty}^{\infty} z_i \mathbb{I}(z_i \geq 0)\phi_\xi(z_i) dz_i \\ &= \int_0^\infty z_i \phi_\xi(z_i) dz_i = \int_0^\infty z_i \frac{\phi(\frac{z_i - \mu_i}{\sigma_i})}{\sigma_i} dz_i \\ &= \int_0^\infty z_i \frac{\phi(\frac{z_i - \mu_i}{\sigma_i})}{\sigma_i} dz_i \qquad \text{(Equation C.1)} \end{aligned}$$

where $\phi_\xi(z_i)$ is the PDF of $\mathcal{N}(\mu_i, \sigma_i^2)$, and $\phi(z_i)$ is the PDF of the standard normal distribution (i.e. $\mathcal{N}(0, 1)$).

We can construct the expectation of the truncated normal in the following way:

$$\begin{aligned} l(w,b) &= \lim_{b \longrightarrow +\infty} \int_0^b z_i \frac{\phi(\frac{z_i-\mu_i}{\sigma_i})}{\sigma_i} \left(\frac{\Phi(\frac{b-\mu_i}{\sigma_i}) - \Phi(\frac{0-\mu_i}{\sigma_i})}{\Phi(\frac{b-\mu_i}{\sigma_i}) - \Phi(\frac{0-\mu_i}{\sigma_i})} \right) dz_i \\ &= \int_0^{+\infty} z_i \frac{\phi(\frac{z_i-\mu_i}{\sigma_i})}{\sigma_i} \left(\frac{1-\Phi(\frac{-\mu_i}{\sigma_i})}{1-\Phi(\frac{-\mu_i}{\sigma_i})} \right) dz_i \qquad \text{(Equation C.2)} \end{aligned}$$

where $\Phi(x)$ is the CDF of standard normal distribution. Since, $\Phi(-x) = 1 - \Phi(x)$, we have:

$$\begin{aligned} l(w,b) &= \int_0^{+\infty} z_i \frac{\phi(\frac{z_i-\mu_i}{\sigma_i})}{\sigma_i} \left(\frac{\Phi(\frac{\mu_i}{\sigma_i})}{1-\Phi(\frac{-\mu_i}{\sigma_i})} \right) dz_i \\ &= \Phi(\frac{\mu_i}{\sigma_i}) \int_0^{+\infty} z_i \frac{\phi(\frac{z_i-\mu_i}{\sigma_i})}{\sigma_i(1-\Phi(\frac{-\mu_i}{\sigma_i}))} dz_i \qquad \text{(Equation C.3)} \end{aligned}$$

Note that $\frac{\phi(\frac{z_i-\mu_i}{\sigma_i})}{\sigma_i(1-\Phi(\frac{-\mu_i}{\sigma_i}))}$ is the PDF of truncated normal distribution: $\mathcal{N}(\mu_i, \sigma_i^2 | 0 \leq z_i \leq +\infty)$, therefore:

$$\begin{aligned} l(w,b) &= \Phi(\frac{\mu_i}{\sigma_i}) \mathbb{E}_{z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)}(z_i) \\ &= \Phi(\frac{\mu_i}{\sigma_i})(\mu_i + \sigma_i \frac{\phi(-\frac{\mu_i}{\sigma_i})}{1-\Phi(-\frac{\mu_i}{\sigma_i})}) \qquad \text{(Equation C.4)} \end{aligned}$$

Because of the symmetry of the standard normal distribution, we have $\phi(-\frac{\mu_i}{\sigma_i}) = \phi(\frac{\mu_i}{\sigma_i})$ and $1 - \Phi(-\frac{\mu_i}{\sigma_i}) = \Phi(\frac{\mu_i}{\sigma_i})$, therefore:

$$l(w,b) = \mu_i \Phi(\frac{\mu_i}{\sigma_i}) + \sigma_i \phi(\frac{\mu_i}{\sigma_i}) \qquad \text{(Equation C.5)}$$

APPENDIX D

α-REG PROOF

In this appendix, we provide a proof for Theorem 7.

Proof. We start with Problem Equation 5.11. By expanding the expectation, we will have:

$$\begin{aligned}&\text{maximize}_\alpha \sum_i \int_{\zeta_i \in \mathcal{Z}} \alpha_i(\zeta_i) dP(\zeta_i) - \frac{1}{2}\sum_{i,j} y_i y_j \int_{\zeta_i,\zeta_j \in \mathcal{Z}} \alpha_i(\zeta_i)\alpha_j(\zeta_j)\tilde{k}(x_i, x_j, \zeta_i, \zeta_j) dP(\zeta_i) dP(\zeta_j)\\ &\text{subject to } \sum_i y_i \int_{\zeta_i \in \mathcal{Z}} \alpha_i(\zeta_i) dP(\zeta_i) = 0,\ 0 \le \alpha_i(\zeta_i) \le C\ \forall i, \zeta_i \qquad \text{(Equation D.1)}\end{aligned}$$

The theorem assumes $\alpha_i(\zeta_i)$'s are independent of ζ_i's and equal to scalars α_is, and $\tilde{f}(x) = f(x) \odot \zeta_i$ (i.e. dropout in feature space). So (Equation 5.11) can be written as:

$$\begin{aligned}&\text{maximize}_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \int_{\zeta_i,\zeta_j \in \mathcal{Z}} \alpha_i\alpha_j (f(x_i) \odot \zeta_i)^T (f(x_j) \odot \zeta_j) dP(\zeta_i) dP(\zeta_j)\\ &\text{subject to } \sum_i y_i \alpha_i = 0,\ 0 \le \alpha_i \le C\ \forall i, \qquad \text{(Equation D.2)}\end{aligned}$$

We have:

$$\begin{aligned}
& \frac{1}{2}\sum_{i,j} y_i y_j \int_{\zeta_i,\zeta_j\in\mathcal{Z}} \alpha_i\alpha_j (f(x_i)\odot\zeta_i)^T(f(x_j)\odot\zeta_j)dP(\zeta_i)dP(\zeta_j) \\
= \; & \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i\alpha_j \mathbb{E}_{\zeta_i,\zeta_j\sim p(\delta)}[\sum_k f_k(x_i)f_k(x_j)\zeta_{ik}\zeta_{jk}] \\
= \; & \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i\alpha_j \sum_k f_k(x_i)f_k(x_j)\mathbb{E}_{\zeta_{ik},\zeta_{jk}\sim p(\delta)}[\zeta_{ik}\zeta_{jk}] \qquad \text{(Equation D.3)}
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_{\zeta_{ik},\zeta_{jk}\sim p(\delta)}[\zeta_{ik}\zeta_{jk}] &= \mathbb{E}_{\zeta_{ik},\zeta_{jk}\sim p(\delta)}[(\zeta_{ik}-1)(\zeta_{jk}-1)+\zeta_{ik}+\zeta_{jk}-1] \\
&= \mathbb{E}_{\zeta_{ik},\zeta_{jk}\sim p(\delta)}[(\zeta_{ik}-1)(\zeta_{jk}-1)] + \mathbb{E}_{\zeta_{ik}\sim p(\delta)}[\zeta_{ik}] + \mathbb{E}_{\zeta_{jk}\sim p(\delta)}[\zeta_{jk}] - 1 \\
&= \mathbb{E}_{\zeta_{ik},\zeta_{jk}\sim p(\delta)}[(\zeta_{ik}-1)(\zeta_{jk}-1)] + 1 \\
&= I(i=j)\sigma_\zeta^2 + 1
\end{aligned}$$

where $I(i=j)=1$ if $i=j$ and zero otherwise, $\mathbb{E}_{\zeta_{ik}\sim p(\delta)}[\zeta_{ik}] = \mathbb{E}_{\zeta_{jk}\sim p(\delta)}[\zeta_{jk}] = 1$. Since ζ_{ik} and ζ_{jk} are identically and independently drawn from the noise distribution,

$$\mathbb{E}_{\zeta_{ik},\zeta_{jk}\sim p(\delta)}[(\zeta_{ik}-1)(\zeta_{jk}-1)] = \begin{cases} \mathrm{var}(\delta) = \sigma_\zeta^2 & \text{if } i=j \\ 0 & \text{if } i\neq j \end{cases}$$

Therefore,

$$\begin{aligned}
& \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \sum_k f_k(x_i) f_k(x_j) \mathbb{E}_{\zeta_{ik},\zeta_{jk}\sim p(\delta)}[\zeta_{ik}\zeta_{jk}] \\
= \; & \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j (1 + I(i=j)\sigma_\zeta^2) \sum_k f_k(x_i) f_k(x_j) \\
= \; & \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j (1 + I(i=j)\sigma_\zeta^2) k(x_i, x_j)
\end{aligned}$$

□

APPENDIX E

LINEAR TIME INFERENCE FOR DIMENSION DROPOUT IN RBF KERNEL

In the Chapter 5, we briefly mention the marginalized prediction for dropout-dimension, based on the expected kernel function. We now show how this is obtained.

Theorem 9. *For dimension dropout, the expected RBF kernel function between a support vector x with noise ξ and a test instance x' has the following closed-form solution:*

$$E_\zeta[\tilde{k}(x, x', \xi, \zeta)] = \prod_{l:\xi_l \neq 0} (\delta + (1-\delta)e^{-\gamma(x_l - x'_l)^2})$$

Proof. The dimension dropout RBF kernel is defined as follows:

$$\tilde{k}(x, x', \xi, \zeta) = e^{-\gamma d(x, x', \xi, \zeta)^2}$$

where $d(x, x', \xi, \zeta)^2 = \sum_{l:\xi_l \neq 0, \zeta_l \neq 0} (x_l - x'_l)^2$. We can also express this kernel as a product:

$$\tilde{k}(x, x', \xi, \zeta) = \prod_{l:\xi_l \neq 0, \zeta_l \neq 0} e^{-\gamma(x_l - x'_l)^2} = \prod_{l:\xi_l \neq 0} e^{-\gamma\zeta_l(x_l - x'_l)^2}$$

For dropout noise, ζ_l equals 0 with probability δ and 1 with probability $1-\delta$. Since each ζ_l is drawn independently, we can convert the expectation of a product to a product of expectations:

$$E_\zeta\left[\prod_{l:\xi_l \neq 0} e^{-\gamma\zeta_l(x_l - x'_l)^2}\right] = \prod_{l:\xi_l \neq 0} E_{\zeta_l}\left[e^{-\gamma\zeta_l(x_l - x'_l)^2}\right] = \prod_{l:\xi_l \neq 0} (\delta + (1-\delta)e^{-\gamma(x_l - x'_l)^2})$$

□

Since the SVM prediction is a linear function of kernel values, the expected prediction is a linear function of the expected kernel values:

$$E_\zeta\left[\sum_i y_i\alpha_i\tilde{k}(x_i,\xi_i,x',\zeta)\right] = \sum_i y_i\alpha_i E_\zeta[\tilde{k}(x_i,\xi_i,x',\zeta)]$$

Each expected kernel value can be computed in linear time by iterating over the non-dropped-out features of the support vector.

APPENDIX F

NOTATIONS AND SYMBOLS

$\|\mathbf{A}\|$	The determinant of matrix $\mathbf{A}$
$\|\mathbf{x}\|_0$	The L_0 norm of vector $\mathbf{x}$. $\|\mathbf{x}\|_0$ is used to indicate the number of non-zero elements in $\mathbf{x}$.
$\|\mathbf{x}\|_p$	The L_p norm of vector $\mathbf{x}$. $\|\mathbf{x}\|_p = (\sum_{i=0}^{m} \|x_i\|^p)^{\frac{1}{p}}$
$\mathbf{0}$	A vector of zeros
$\mathbf{1}$	A vector of ones
$\mathbf{A}$	Matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ for some positive integers n and m; bold capital letters are used for matrices.
$\mathbf{A}^T$	The transpose of matrix $\mathbf{A}$
A_{ij}	The element of $\mathbf{A}$ in row i and column j
$\mathbf{A} \odot \mathbf{B}$	Hadamard (element-wise) product of matrices $\mathbf{A} \odot \mathbf{B}$: $(\mathbf{A} \odot \mathbf{B})_{ij} = A_{ij}B_{ij}$. Same notation is used for the Hadamard product of vectors.
$\mathrm{diag}(\mathbf{x})$	The $m \times m$ diagonal matrix, in which the elements of vector $\mathbf{x}$ form the diagonal.
m	The dimension of a random vector or a feature function
n	The number of samples in a dataset
x	A scalar $x \in \mathbb{R}$

$\mathbf{x}$	Vector $\mathbf{x} \in \mathbb{R}^m$ for some positive integer m; bold lowercase letters are used to indicate vectors. Vectors are assumed to be $m \times 1$ dimensional matrices (i.e. column vector): $\mathbf{x} = [x_1, \ldots, x_m]^T$
x_i	The ith element of $\mathbf{x}$
$\mathbf{x} \sim f_x(\theta)$	Random variable (vector) $\mathbf{x}$ is drawn from the probability distribution f_x, which is parameterized by vector θ.
$\mathcal{N}(\mu, \mathbf{\Sigma})$	A Gaussian (normal) distribution with mean μ and covariance matrix $\mathbf{\Sigma}$
$\phi(\mathbf{x}; \mu, \mathbf{\Sigma})$	The probability density function (PDF) of the normal distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$. $\phi(\mathbf{x}; \mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{\lvert 2\pi\mathbf{\Sigma}\rvert}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)}$
$\phi(x)$	The standard normal PDF with mean 0 and variance 1 (i.e. $\mathcal{N}(0,1)$). $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$. For univariate random variable $x \sim \mathcal{N}(0,1)$, we have: $\phi(x; \mu, \sigma^2) = \phi(\frac{x-\mu}{\sigma})$.
$\Phi(\mathbf{t}; \mu, \mathbf{\Sigma})$	The cumulative density function (CDF) of the normal distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$. $\Phi(\mathbf{t}; \mu, \mathbf{\Sigma}) = \int_{\mathbf{x} \leq \mathbf{t}} \phi(\mathbf{x}; \mu, \mathbf{\Sigma}) d\mathbf{x}$

REFERENCES CITED

Abernethy, J., Chapelle, O., and Castillo, C. (2010). Graph regularization methods for web spam detection. *Machine Learning*, 81(2):207–225.

Adamic, L. and Glance, N. (2005). The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43. ACM.

Altun, Y., Tsochantaridis, I., Hofmann, T., et al. (2003). Hidden markov support vector machines. In *ICML*, volume 3, pages 3–10.

An, B., Kempe, D., Kiekintveld, C., Shieh, E., Singh, S., Tambe, M., and Vorobeychik, Y. (2012). Security games with limited surveillance. *Ann Arbor*, 1001:48109.

Bakir, G. H., Hofmann, T., Scholkopf, B., Smola, A. J., Taskar, B., and Vishwanathan, S. V. N. (2007). *Predicting Structured Data*. MIT Press.

Bartlett, P. L., Collins, M., Taskar, B., and McAllester, D. A. (2004). Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems (NIPS) 18*.

Basilico, N., Gatti, N., and Amigoni, F. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 57–64. International Foundation for Autonomous Agents and Multiagent Systems.

Begleiter, R., El-Yaniv, R., and Yona, G. (2004). On prediction using variable order markov models. *J. Artif. Intell. Res.(JAIR)*, 22:385–421.

Ben-Tal, A. and Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805.

Ben-Tal, A. and Nemirovski, A. (1999). Robust solutions of uncertain linear programs. *Operations research letters*, 25(1):1–13.

Ben-Tal, A. and Nemirovski, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424.

Ben-Tal, A. and Nemirovski, A. (2001). On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26(2):193–205.

Bertsimas, D., Pachamanova, D., and Sim, M. (2004). Robust linear optimization under general norms. *Operations Research Letters*, 32(6):510–516.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations research*, 52(1):35–53.

Bhattacharyya, C., Pannagadatta, K., and Smola, A. J. (2004). A second order cone programming formulation for classifying missing data. *Advances in neural information processing systems*, 17:153–160.

Biggio, B., Corona, I., Nelson, B., Rubinstein, B. I., Maiorca, D., Fumera, G., Giacinto, G., et al. (2014). Security evaluation of support vector machines in adversarial environments. *arXiv preprint arXiv:1401.7727*.

Biggio, B., Fumera, G., and Roli, F. (2013a). Security evaluation of pattern classifiers under attack.

Biggio, B., Nelson, B., and Laskov, P. (2011). Support vector machines under adversarial label noise. *Journal of Machine Learning Research-Proceedings Track*, 20:97–112.

Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., and Roli, F. (2013b). Is data clustering in adversarial settings secure? In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, pages 87–98. ACM.

Bilmes, J., Zweig, G., Richardson, T., Filali, K., Livescu, K., Xu, P., Jackson, K., Brandman, Y., Sandness, E., Holtz, E., et al. (2001). Discriminatively structured graphical models for speech recognition. In *Report of the JHU 2001 Summer Workshop*.

Bishop, C. M. (1995). Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blanzieri, E. and Bryl, A. (2008). A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, 29(1):63–92.

Blum, A. (2006). Random projection, margins, kernels, and feature-selection. In *Subspace, Latent Structure and Feature Selection*, pages 52–68. Springer.

Boyd, S. P. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Brückner, M., Kanzow, C., and Scheffer, T. (2012). Static prediction games for adversarial learning problems. *the Journal of Machine Learning Research*, 13(1):2617–2654.

Brückner, M. and Scheffer, T. (2009). Nash equilibria of static prediction games. In *Advances in Neural Information Processing Systems 22*.

Brückner, M. and Scheffer, T. (2011). Stackelberg games for adversarial prediction problems. In *Proceedings of the Seventeenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press.

Califf, M. E. and Mooney, R. J. (2003). Bottom-up relational learning of pattern matching rules for information extraction. *The Journal of Machine Learning Research*, 4:177–210.

Chang, K.-W., Hsieh, C.-J., and Lin, C.-J. (2008). Coordinate descent method for large-scale l2-loss linear support vector machines. *The Journal of Machine Learning Research*, 9:1369–1398.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Chau, D., Pandit, S., and Faloutsos, C. (2006). Detecting fraudulent personalities in networks of online auctioneers. *Knowledge Discovery in Databases: PKDD 2006*, pages 103–114.

Chen, N., Zhu, J., Chen, J., and Zhang, B. (2014a). Dropout training for support vector machines. In *Proceedings of Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1752–1759, Quebec, Canada.

Chen, S., Feng, Z., Lu, Q., Mahasseni, B., Fiez, T., Fern, A., and Todorovic, S. (2014b). Play type recognition in real-world football video. *WACV*.

Chieu, H. L. and Ng, H. T. (2002). A maximum entropy approach to information extraction from semi-structured and free text. *AAAI/IAAI*, 2002:786–791.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Philadelphia, PA. ACL.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

Collins, M., Globerson, A., Koo, T., Carreras, X., and Bartlett, P. L. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *The Journal of Machine Learning Research*, 9:1775–1822.

Collins, M. and Koo, T. (2005). Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Dalvi, N., Domingos, P., Mausam, Sanghai, S., and Verma, D. (2004). Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, Seattle, WA. ACM Press.

Daumé III, H. (2009a). Semi-supervised or semi-unsupervised? In *NAACL Workshop on Semi-supervised Learning for NLP*. Citeseer.

Daumé III, H. (2009b). Unsupervised search-based structured prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 209–216. ACM.

Daumé Iii, H., Langford, J., and Marcu, D. (2009). Search-based structured prediction. *Machine learning*, 75(3):297–325.

Daumé III, H. and Marcu, D. (2005). Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM.

Decoste, D. and Schölkopf, B. (2002). Training invariant support vector machines. *Machine learning*, 46(1-3):161–190.

Dekel, O., Shamir, O., and Xiao, L. (2010). Learning to classify with missing and corrupted features. *Machine learning*, 81(2):149–178.

Derezinski, M. and Warmuth, M. K. (2014). The limits of squared Euclidean distance regularization. In *Advances in Neural Information Processing Systems*, pages 2807–2815.

Dickerson, J. P., Simari, G. I., Subrahmanian, V., and Kraus, S. (2010). A graph-theoretic approach to protect static and moving targets from adversaries. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 299–306. International Foundation for Autonomous Agents and Multiagent Systems.

Domingos, P. and Lowd, D. (2009a). *Markov Logic: An Interface Layer for AI.* Morgan & Claypool, San Rafael, CA.

Domingos, P. and Lowd, D. (2009b). *Markov logic: An interface layer for artificial intelligence*, volume 3. Morgan & Claypool Publishers.

Domke, J. (2013). Structured learning via logistic regression. In *Advances in Neural Information Processing Systems*, pages 647–655.

Doppa, J. R., Fern, A., and Tadepalli, P. (2012). Output space search for structured prediction. *arXiv preprint arXiv:1206.6460*.

Dreves, A., Facchinei, F., Kanzow, C., and Sagratella, S. (2011). On the solution of the kkt conditions of generalized nash equilibrium problems. *SIAM Journal on Optimization*, 21(3):1082–1108.

Dritsoula, L., Loiseau, P., and Musacchio, J. (2012). A game-theoretical approach for finding optimal strategies in an intruder classification game. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 7744–7751. IEEE.

Drost, I. and Scheffer, T. (2005). Thwarting the nigritude ultramarine: Learning to identify link spam. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 96–107. Springer.

El Ghaoui, L., Lanckriet, G., and Natsoulis, G. (2003). *Robust classification with interval data*. Computer Science Division, University of California.

Fang, F., Jiang, A. X., and Tambe, M. (2013). Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 957–964. International Foundation for Autonomous Agents and Multiagent Systems.

Fua, P., Li, Y., Lucchi, A., et al. (2013). Learning for structured prediction using approximate subgradient descent with working sets. In *Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-185082.

Globerson, A., Koo, T. Y., Carreras, X., and Collins, M. (2007). Exponentiated gradient algorithms for log-linear structured prediction. In *Proceedings of the 24th international conference on Machine learning*, pages 305–312. ACM.

Globerson, A. and Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 353–360, Pittsburgh, PA. ACM Press.

Gong, D., Zhao, X., and Medioni, G. (2012). Robust multiple manifolds structure learning. *ICML*.

Gupta, K. K., Nath, B., and Kotagiri, R. (2010). Layered approach using conditional random fields for intrusion detection. *Dependable and Secure Computing, IEEE Transactions on*, 7(1):35–49.

Gupta, K. K., Nath, B., and Ramamohanarao, K. (2007). Conditional random fields for intrusion detection. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 1, pages 203–208. IEEE.

Gurobi Optimization, I. (2014). Gurobi optimizer reference manual.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580.*

Huynh, T. and Mooney, R. (2009). Max-margin weight learning for Markov logic networks. In *In Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09). Bled*, pages 564–579. Springer.

Jain, M., Kardes, E., Kiekintveld, C., Ordóñez, F., and Tambe, M. (2010a). Security games with arbitrary schedules: A branch and price approach. In *AAAI.*

Jain, M., Tsai, J., Pita, J., Kiekintveld, C., Rathi, S., Tambe, M., and Ordóñez, F. (2010b). Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40(4):267–290.

Jensen, D. and Neville, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 259–266, Sydney, Australia. Morgan Kaufmann.

Jiang, A. X., Nguyen, T. H., Tambe, M., and Procaccia, A. D. (2013a). Monotonic maximin: A robust stackelberg solution against boundedly rational followers. In *Decision and Game Theory for Security*, pages 119–139. Springer.

Jiang, A. X., Yin, Z., Zhang, C., Tambe, M., and Kraus, S. (2013b). Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 207–214. International Foundation for Autonomous Agents and Multiagent Systems.

Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

Kivinen, J. and Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63.

Kloft, M. and Laskov, P. (2007). A poisoning attack against online anomaly detection. In *NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security*. Citeseer.

Koller, B., Carlos, T., and Daphne, G. (2003). Max-margin markov networks. In *Advances in Neural Information Processing Systems (NIPS) 17*, Vancouver, BC, Canada.

Kolmogorov, V. and Zabin, R. (2004). What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159.

Korolev, V. and Shevtsova, I. (2012). An improvement of the berry–esseen inequality with applications to poisson and mixed poisson random sums. *Scandinavian Actuarial Journal*, 2012(2):81–105.

Korzhyk, D., Conitzer, V., and Parr, R. (2011). Solving stackelberg games with uncertain observability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1013–1020. International Foundation for Autonomous Agents and Multiagent Systems.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, Williamstown, MA. Morgan Kaufmann.

Lanckriet, G. R., Ghaoui, L. E., Bhattacharyya, C., and Jordan, M. I. (2003). A robust minimax approach to classification. *The Journal of Machine Learning Research*, 3:555–582.

Laskov, P. and Kloft, M. (2009). A framework for quantitative security analysis of machine learning. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, pages 1–4. ACM.

Laskov, P. and Lippmann, R. (2010). Machine learning in adversarial environments. *Machine learning*, 81(2):115–119.

Lauritzen, S. L. (1996). *Graphical models*. Oxford University Press.

Le, Q., Sarlós, T., and Smola, A. (2013). Fastfood-computing hilbert space expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning*, pages 244–252.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lin, H.-T. and Lin, C.-J. (2003). A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. *Technical report, Department of Computer Science, National Taiwan University.*

Lippmann, R. P. (1987). An introduction to computing with neural nets. *ASSP Magazine, IEEE*, 4(2):4–22.

Livni, R. and Globerson, A. (2012). A simple geometric interpretation of SVM using stochastic adversaries. *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics.*

Lowd, D. and Domingos, P. (2007). Efficient weight learning for Markov logic networks. In *Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 200–211, Warsaw, Poland. Springer.

Lowd, D. and Meek, C. (2005a). Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647. ACM.

Lowd, D. and Meek, C. (2005b). Good word attacks on statistical spam filters. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, pages 125–132.

Lowd, D. and Shamaei, A. (2011). Mean field inference in dependency networks: An empirical study. In *AAAI*.

Lu, Q. and Getoor, L. (2003). Link-based classification. In *ICML*, volume 3, pages 496–503.

Maaten, L., Chen, M., Tyree, S., and Weinberger, K. Q. (2013). Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 410–418.

McAllester, D., Collins, M., and Pereira, F. (2004). Case-factor diagrams for structured probabilistic modeling. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 382–391. AUAI Press.

McAuley, J. J., Caetano, T. S., and Smola, A. J. (2008). Robust near-isometric matching via structured learning of graphical models. In *Advances in Neural Information Processing Systems (NIPS) 22*, pages 1057–1064.

McCallum, A., Freitag, D., and Pereira, F. C. (2000). Maximum entropy markov models for information extraction and segmentation. In *ICML*, pages 591–598.

McDonald, R., Hall, K., and Mann, G. (2010). Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464. Association for Computational Linguistics.

McDonald, R., Hannan, K., Neylon, T., Wells, M., and Reynar, J. (2007). Structured models for fine-to-coarse sentiment analysis. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 432.

McDonald, R. and Pereira, F. (2005). Identifying gene and protein mentions in text using conditional random fields. *BMC bioinformatics*, 6(Suppl 1):S6.

McDowell, L. K., Gupta, K. M., and Aha, D. W. (2009). Cautious collective classification. *The Journal of Machine Learning Research*, 10:2777–2836.

Nelson, B. (2010). *Behavior of Machine Learning Algorithms in Adversarial Environments*. PhD thesis, Electrical Engineering and Computer Sciences University of California at Berkeley, California, United States.

Nelson, B., Rubinstein, B., Huang, L., Joseph, A., Lau, S., Lee, S., Rao, S., Tran, A., and Tygar, J. (2010). Near-optimal evasion of convex-inducing classifiers. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010*, volume 9, Chia Laguna Resort, Sardinia, Italy.

Neville, J. and Jensen, D. (2007). Relational dependency networks. *The Journal of Machine Learning Research*, 8:653–692.

Nguyen, T. H., Yang, R., Azaria, A., Kraus, S., and Tambe, M. (2013). Analyzing the effectiveness of adversary modeling in security games. In *Conf. on Artificial Intelligence (AAAI)*.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., et al. (2003). Syntax for statistical machine translation. In *Johns Hopkins University 2003 Summer Workshop on Language Engineering, Center for Language and Speech Processing, Baltimore, MD, Tech. Rep.*

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA.

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238.

Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., and Kraus, S. (2008). Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems.

Pita, J., Tambe, M., Kiekintveld, C., Cullen, S., and Steigerwald, E. (2011). Guards: innovative application of game theory for national airport security. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2710–2715. AAAI Press.

Punyakanok, V. and Roth, D. (2001). The use of classifiers in sequential inference. *arXiv preprint cs/0111003*.

Qiao, Y., Xin, X., Bin, Y., and Ge, S. (2002). Anomaly intrusion detection method based on hmm. *Electronics Letters*, 38(13):663–664.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Ranjbar, M., Lan, T., Wang, Y., Robinovitch, S. N., Li, Z.-N., and Mori, G. (2013). Optimizing nondecomposable loss functions in structured prediction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(4):911–924.

Ravichandran, D., Hovy, E., and Och, F. J. (2003). Statistical qa-classifier vs. re-ranker: what's the difference? In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 69–75. Association for Computational Linguistics.

Ross, S., Gordon, G. J., and Bagnell, J. A. (2010). A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1011.0686*.

Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). No-regret reductions for imitation learning and structured prediction. In *In AISTATS*. Citeseer.

Rudin, W. (2011). *Fourier analysis on groups.* John Wiley & Sons.

Sawade, C., Scheffer, T., et al. (2013). Bayesian games for adversarial regression problems. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 55–63.

Schölkopf, A., Simard, P., Vapnik, V., and Smola, A. (1997). Improving the accuracy and speed of support vector machines. *Advances in neural information processing systems*, 9:375–381.

Schölkopf, D. (2002). Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, volume 1, page 335. MIT Press.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, 29(3):93.

Shalev-Shwartz, S. (2011). Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30.

Shen, D., Sun, J.-T., Li, H., Yang, Q., and Chen, Z. (2007). Document summarization using conditional random fields. In *IJCAI*, volume 7, pages 2862–2867.

Shen, L., Sarkar, A., and Och, F. J. (2004). Discriminative reranking for machine translation. In *HLT-NAACL*, pages 177–184.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., and Meyer, G. (2012). Protect: A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 13–20. International Foundation for Autonomous Agents and Multiagent Systems.

Shivaswamy, P. K., Bhattacharyya, C., and Smola, A. J. (2006). Second order cone programming approaches for handling missing and uncertain data. *The Journal of Machine Learning Research*, 7:1283–1314.

Smith, G. D. (1985). *Numerical solution of partial differential equations: finite difference methods.* Oxford University Press.

Smola, A. J., Vishwanathan, S., and Le, Q. V. (2007). Bundle methods for machine learning. In *Advances in Neural Information Processing Systems (NIPS) 21*, pages 1377–1384.

Song, Y., Wen, Z., Lin, C.-Y., and Davis, R. (2013). One-class conditional random fields for sequential anomaly detection. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1685–1691. AAAI Press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Taskar, B., Chatalbashev, V., and Koller, D. (2004a). Learning associative Markov networks. In *Proceedings of the twenty-first international conference on machine learning*. ACM Press.

Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. (2005). Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM.

Taskar, B., Wong, M. F., Abbeel, P., and Koller, D. (2004b). Max-margin Markov networks. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Teo, C., Globerson, A., Roweis, S., and Smola, A. (2008). Convex learning with invariances. In *Advances in Neural Information Processing Systems 21*.

Theil, H. and Fiebig, D. G. (1984). *Exploiting continuity: Maximum entropy estimation of continuous distributions*. Ballinger Cambridge, MA.

Tian, Y., Huang, T., and Gao, W. (2006). Robust collective classification with contextual dependency network models. In *Advanced Data Mining and Applications*, pages 173–180. Springer.

Torkamani, M. (2014). Adversarial structured output prediction. Available athttp://www.cs.uoregon.edu/Reports/ORAL-201406-Torkamani.pdf. Oral Comprehensive Exam.

Torkamani, M. and Lowd, D. (2013). Convex adversarial collective classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 642–650.

Torkamani, M. A. and Lowd, D. (2014). On robustness and regularization of structural support vector machines. *Proceedings of the Thirty-First International Conference on Machine Learning (ICML)*, pages 577–585.

Toutanova, K., Haghighi, A., and Manning, C. D. (2005). Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 589–596. Association for Computational Linguistics.

Tsai, J., Kiekintveld, C., Ordonez, F., Tambe, M., and Rathi, S. (2009). Iris-a tool for strategic security allocation in transportation networks.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2006). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453.

Wager, S., Fithian, W., Wang, S., and Liang, P. S. (2014). Altitude training: Strong bounds for single-layer dropout. In *Advances in Neural Information Processing Systems*, pages 100–108.

Wager, S., Wang, S., and Liang, P. (2013). Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pages 351–359.

Wang, S. and Manning, C. (2013). Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 118–126.

Wang, S., Wang, M., Wager, S., Liang, P., and Manning, C. D. (2013). Feature noising for log-linear structured prediction. In *EMNLP*, pages 1170–1179.

Wang, S. I. and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the ACL*, pages 90–94.

Xu, H., Caramanis, C., and Mannor, S. (2009). Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510.

Xu, H., Caramanis, C., and Mannor, S. (2010). Robust regression and lasso. *IEEE Transactions on Information Theory*, 56(7):3561–3574.

Xu, H. and Mannor, S. (2012). Robustness and generalization. *Machine learning*, 86(3):391–423.

Yin, Z., Jain, M., Tambe, M., and Ordóñez, F. (2011). Risk-averse strategies for security games with execution and observational uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Yin, Z., Jiang, A. X., Johnson, M. P., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., Tambe, M., and Sullivan, J. P. (2012). Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*.

Yu, C.-N. J. and Joachims, T. (2009). Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176. ACM.

Zhang, S.-X., Gales, M. J., et al. (2011). Structured support vector machines for noise robust continuous speech recognition. In *INTERSPEECH*, pages 989–990.

Zhang, S.-X., Ragni, A., and Gales, M. J. F. (2010). Structured log linear models for noise robust speech recognition. *Signal Processing Letters, IEEE*, 17(11):945–948.