University of Oregon

Adversarial Structured Output Prediction

by

Mohamad Ali Torkamani

Oral Comprehensive Report
University of Oregon

Advisor:
Prof. Daniel Lowd
Computer and Information Science Department

August 2014

Abstract

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 difficult 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. Many real world applications can be abstracted as an adversarial structured output prediction problem. In this oral exam, I study the state-of-the-art methods for solving the problem of structured learning and output prediction in adversarial settings. In particular, I will mention the strengths and weaknesses of the existing methods, and point to the open problems in the field.

Contents

Chapter 1

Introduction

Traditional machine learning methods assume that both training and test data samples are independently drawn from the same distribution. If the data samples are "Independently and Identically Drawn" (IID) from a distribution, then the mathematical modeling of their joint distribution is simpler and cleaner. Although the IID assumption *rarely* holds in practice, many of statistical approaches, including classical machine learning, still suppose that it is satisfied; this allows simpler math, and more tractability. Due to this fact, the classical machine learning algorithms fail in many of the existing and emerging real-world problems. The IID assumption can be violated for two reasons: first, the underlying distribution that generates the data is constantly changing; therefore, the train and test data might be drawn from two different distributions. The change in the distributions at train and test time may be due to different reasons: random noise, natural concept drift such as change of topic in discussion forums, or malicious manipulation of the data by some adversary. Inter-dependent 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; or in a graph data, where 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, the labeling of a graph, the segmentation of an image, etc.

To date, most of the modern methods in machine learning are designed to solve only one of these two challenges; i.e., either they solve the problem of inter-related data, or they are designed to be robust against noise, and natural or adversarial changes in the distribution of samples. In this report, I study the state-of-the-art methods in machine learning where both of the IID assumptions are violated: the samples are not independent and are not drawn from a static distribution at train and test time. In particular, I focus on the worst-case scenario, where the test data is intentionally manipulated by some adversary to deceive the machine learning algorithm. In the rest of the report, I refer to this problem as adversarial structured prediction[1].

A motivating example of adversarial structured prediction is collective classification in adversarial settings. In collective classification [1], 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 [2, 3], and their relational extensions, such as Markov logic networks [4], can handle both uncertainty and complex relationships in a single model, making them well-suited to collective classification problems.

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 in order to avoid detection. For example, when the search engines began to use incoming links for ranking of the 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 wep-page, manufacturing incoming links makes a spammy web site appear more legitimate. Web-spam [5, 6] is one of many examples with explicitly adversarial domains; some other examples are counter-terrorism, online auction fraud [7], and spam in online social networks.

At this point point in time, there exists no complete theory of adversarial structured prediction. In the rest of this section, I provide a high-level overview of the previous studies and their limitations. Early research in adversarial machine learning included methods for blocking the adversary by anticipating their next move [8], reverse engineering classifiers [9, 10, 11], and building robust classifiers in presence of feature deletion or other variations of the input data [12, 13]. Brückner and Scheffer showed that, under modest assumptions, Nash equilibria can be found for domains such as spam [14]; however, these assumptions are hard to satisfy in real world problems. In a later paper, they showed how we can find Stackelberg equilibria [15]. In a Stackelberg game, one of the players (called "the leader", e.g. the machine learning algorithm) takes action first, and then the other player (called "the follower", e.g. the adversary)

[1]Prediction of dependent outputs is a special case of structured prediction, which is formally defined in the next chapter.

takes action. However, most of the current adversarial methods assume that instances are independent, ignoring the relational nature of many domains.

Teo et al. [13] proposed a robust learning method in the presence of an adversary. Their method was not primarily addressing structured output prediction, but if there is an efficient way of simulating the adversary, then their method is applicable to structured learning as well. In their formulation they have a summation over all possible adversarial actions, which is intractable if the number of adversarial actions is exponential in the input size. By "Convex Adversarial Collective Classification" (CACC), Torkamani and Lowd introduced one of the early algorithms for learning to predict structured outputs in adversarial environments [16]. Unlike the previous research in adversarial machine learning, CACC allows for dependencies among the labels of different objects.

Some of the studies have tried to use robust optimization for achieving stable models in the presence of an adversary. Xu et al. [17] have studied the connection between robustness and regularization in Support Vector Machines (SVMs). In their approach they show that there is strong connection between regularization and robustness in support vector machines for independent data points. Torkamani and Lowd [18] showed that the same results hold for structured output prediction with SVMs; related this phenomenon to the type of perturbations that the adversarial manipulation of data can generate; and generalized this effect to a wide class of possible perturbations. Even this work still has several limitations: for example, the robustness is guaranteed for certain possible adversarial manipulation of data, and only works for structural support vector machines.

A different view of adversarial machine learning, is about the securing the algorithms against training data points that are adversarially engineered to mislead the learning algorithm. This branch of adversarial manipulation of training data is called *poisoning attacks* [19, 20, 21, 22].

While there are few works that directly address the problem of adversarial structured prediction, there are many papers that are somehow related to this topic. For example, there are many application domains, especially in security, that seek solutions in structured prediction algorithms; hence, robustness is a critical issue. There are new optimization algorithms that try to solve the problem of structured prediction more accurately and robustly.

Regret minimization is another field that studies robustness against adversarial components (i.e. nature). This field is relatively new, and there is a good potential of studying the regret minimization methods for structured prediction. The only work that refers to regret minimization for structured prediction is done by Ross et al. [23], but even this work refers to a very specific structured prediction problem where the desired output can be decomposed to a sequence of simple outputs.

One important aspect of adversarial machine learning that is currently missing in the literature of adversarial structured prediction is deep analysis of 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 addition to above-mentioned problems, there are still many open questions. For example, only the vulnerability of a few machine learning algorithms is studied in depth, but many algorithms remain unexplored. The ideal goal is to design a global robustness recipe that is applicable to most of the machine learning algorithm.

In this study, I have tried to cover theoretical foundations of both adversarial machine learning and structured prediction, and then have grouped the work that is related to both of these fields. The following is the outline of this report:

Chapter 2, Structured Prediction: I address structured prediction problems in this chapter. The chapter starts by some motivating examples of structured prediction problems and theoretical foundations of supervised learning. Then, I formally define structured prediction, and the concept of a feature function vector. I continue a brief explaination of the state-of-the-art structured prediction methods. The discussed methods are: structured perceptron; structured prediction based on the maximum entropy approach and log-linear models; the max-margin approach; re-ranking; and search-based methods. I also briefly talk about the “structured prediction cascades” algorithm which results in more efficient use of computational resources. The chapter concludes by several optimization methods that are mostly used for parameter learning in structured prediction.

Chapter 3, Adversarial Machine Learning: A high-level overview of adversarial machine learning is presented in this chapter. In fact, the adversary’s negative effects on the learner results in poor performance of the learned model in the future. In order to be robust to such negative effects, we should know the capabilities of the adversaries. In this chapter, a theoretical model for the adversary is defined, and the properties of the adversary are categorized based on different criteria. One criterion is the type of the adversarial problem, in terms of the order in which the learner, or the adversary react to each others' actions. Another important factor in adversarial problems is the knowledge of the learner about the adversary, and vice-versa. In most of the real-world scenarios the knowledge of each party about the other one is limited, and needs to be somehow inferred. In majority of existing methods, the assumption is that in the worst-case, the adversary has complete knowledge about the learned parameters of the machine learning algorithm, which is not necessarily a realistic assumption. In this chapter I also briefly talk about regret minimization.

Chapter 4, Structured Prediction: This chapter is dedicated to existing methods for adversarial structured prediction and some of the applications. There is a small body of research work that address both adversarial and structural problems at the same time. This chapter covers the adversarial machine learning methods that are either particularly designed for structured prediction, or have the capability of being adapted for this task. This chapter also refers to applications of adversarial structured prediction in real-world problems: many security problems fall into this category, along with other problems in computer vision, speech recognition, outlier detection, etc.

Chapter 5, Structured Prediction: This chapter summarizes the report, and highlights the remaining questions: Scaling-up to large-size problems is an important challenge for machine learning in general, and adversarial structured prediction should address it in the future as well. Inferring the adversary's utility functions is another important problem that can have impact on real-world problems, and there is much room for improvement of existing algorithms. The chapter concludes by listing the future directions that can be explored.

Chapter 2

Structured Prediction

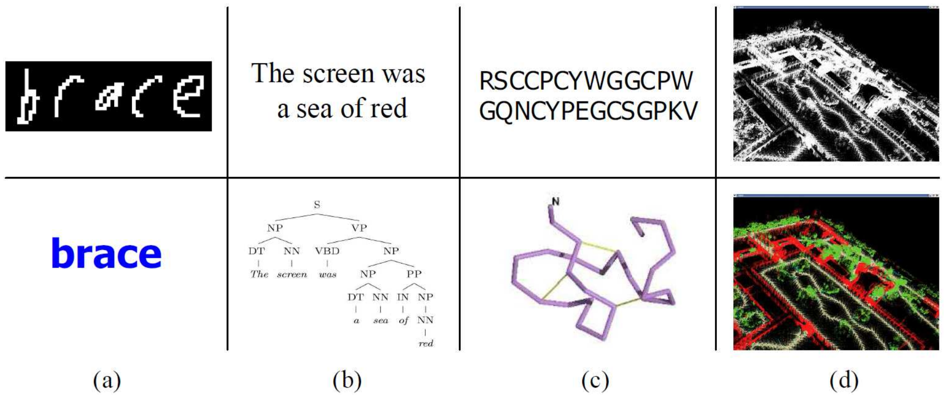
Most of the classical prediction algorithms in machine learning are designed to solve prediction problems whose outputs are a fixed number of binary or real valued variables. In contrast, there are problems with a strong interdependence among the output variables, often with sequential, graphical, or combinatorial structure. 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.

Structured prediction [24] provides a unified treatment for dealing with structured outputs. The structured prediction algorithms root back in seminal work by McCallum et al. [25]; Lafferty et al. [26]; Punyakanok et al. [27]; Collins [28]; Taskar et al. [3]; Altun et al. [29]; McAllester et al.[30]; Tsochantaridis et al. [31], among others.

In this chapter, the foundational background on structured prediction is explained. In this chapter, I start by a brief explanation of the basics of supervised learning in general, and then move on to definition of structured prediction. Then, I present several training algorithms for structured predictors, and also discuss the underlying optimization problems.

2.1 Supervised Learning

In machine learning, output prediction is the procedure of observing the input x (state of some phenomenon) and using our understanding of the concept (the model that is trained on past data) to predict the output y (some hidden property of the observed data).



The screen was a sea of red
RSCCPCYWGGCPW GQNCYPEGCSGPKV
brace
S NP VP DT NN VBD NP The screen was NP PP DT NN IN NP a sea of NN red
N
(a)
(b)
(c)
(d)

FIGURE 2.1: Examples of structured prediction problems (top row:inputs, bottom row: outputs), table from Ben Taskar [32]: (a) handwriting recognition; (b) natural language parsing; (c) disulfide bond prediction in proteins; (d) 3D point cloud classification

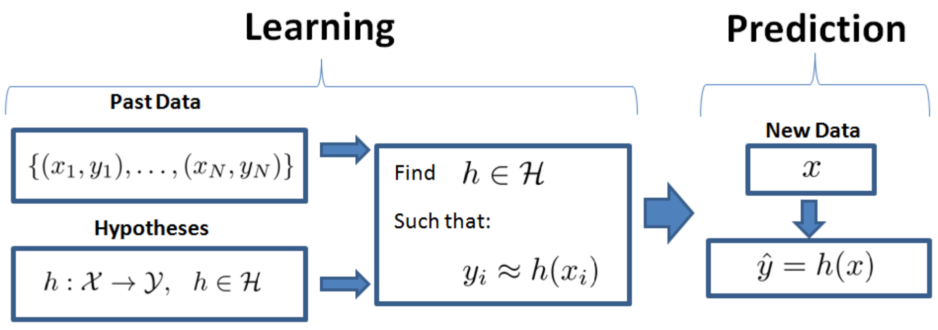


FIGURE 2.2: Supervised learning procedure

The goal is to find a mapping function (a.k.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 $x \in \mathcal{X}$, the predicted output is $\hat{y} = h(x) \in \mathcal{Y}$. 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. In structured prediction $|\mathcal{Y}|$ is usually very large.

The goal of the learning algorithm is to find a mapping function that produces accurate predictions. The training data samples $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\} \in (\mathcal{X} \times \mathcal{Y})^N$, are input-output pairs from the past (empirical distribution). We assume that each sample (x_i, y_i) is drawn from an underlying joint distribution over inputs and outputs: $P(\mathcal{X}, \mathcal{Y})$. Also, we usually assume that y_i is the correct output for the input x_i. For an input x_i, the predicted output $\hat{y}_i = h(x_i)$ should be “close” to the true output y_i. This closeness if usually defined by some non-negative loss function $l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathrm{R}$ that determines the distance of $\hat{y}$ to y. 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 would like to find $h^* \in \mathcal{H}$, such that the expected loss is minimized:

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

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:

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

The term $\frac{1}{N} \sum_{i=1}^{N} \ l \left(h(x_i), y_i \right)$ is called the empirical risk. Fig. 2.2 shows the procedure of supervised learning. If the number of observed samples $|\mathcal{D}|$ is small or $|\mathcal{H}|$ (the number of possible hypotheses) is extremely large, then the learned hypothesis h^* in Eq. 2.2 is likely to "overfit" the data; i.e., we achieve zero (or very small) empirical risk, but large losses (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 adding a penalty function $\Omega_{\mathcal{H}}(h)$ to the optimization problem, that controls the flexibility of h[1]:

$$h^* = \underset{h \in \mathcal{H}}{\arg\min} \ \ \Omega_{\mathcal{H}}(h) + \frac{1}{N} \sum_{i=1}^{N} \ l \left(h(x_i), y_i \right) \tag{2.3}$$

This approach is called regularized risk minimization. Some times the loss function $l(y', y)$ is not convex– and therefore the optimization problem in Eq. 2.3 is not tractable; then, a convex surrogate function for $l(y', y)$ is used instead.

[1]Flexibility of h mostly depends on the function space $\mathcal{H}$. In this report, I assume that $h(x)$ is linear in its parameter, therefore the regularization function penalizes the parameters that are large in magnitude.

The probabilistic interpretation of the regularized risk minimization is based on estimating a parametric form for the data distribution; then the regularization can be interpreted as the log of some prior distribution over the possible hypotheses, and the loss function is usually the log of probability of independent sample pairs (x_i, y_i) (For brevity, I skip the details. The reader can refer to Bishop [33] for further details.).

2.2 Motivation of Structured Prediction

Before the emergence of the structured prediction algorithms, probabilistic graphical models (PGMs) [34] were the most successful methods for solving problems with strongly interdependent outputs. PGMs provide a framework for inference about dependent variables and confounding factors by combining tools from statistics and graph theory. 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.

While PGMs can be used for a wide class of problems, they are very general purpose; this results in some inevitable costs. Using the probability distribution function of variables in the model is desirable in theory, but estimating the parameters of the distribution function– especially the normalization constant (a.k.a. partition function), can be intractable. Structured prediction algorithms do not estimate the probability distribution of the variables explicitly, and particularly avoid the calculation of the normalization constants. Therefore, learning the parameters of structured prediction models is usually more tractable, especially when tailored for specific problems.

Before formally defining "structured prediction", some motivating examples of structured prediction problems are listed in the following:

- Collective classification: given a graph that involves a set of nodes and edges, the goal is to predict a joint labeling for the nodes [1]. Maximum a posteriori inference in many relational learning problems reduces to collective classification [2, 3]. For example Markov logic network (MLN) is a strong method in relational machine learning [4, 35, 36], where some inference tasks can some times be formulated as a collective classification problem [16, 37].
- Sequence labeling: given an input sequence, the goal is to produce a label sequence. Each label is drawn from a small finite set. This problem has many applications in natural language processing for part-of-speech tagging [38, 39, 40], and can be seen as a specific case of the larger class of conditional random fields

(CRFs) and hidden Markov models (HMMs) [26]. A related problem is syntactic parsing for a given input sequence; the goal is to build a tree whose leaves are the elements in the sequence and whose structure is determined by some language grammar [38, 41].

- Bipartite matching: given a bipartite graph, find the best possible matching. This task shows up in the word alignment problem in natural language processing, and protein structure prediction in computational biology [39, 42, 43, 44].

- Computer vision: images and videos are consisted of pixels which are highly correlated, and form objects; understanding the image implies identification of the objects in the image. There are several problems in computer vision that are so far formulated as structured prediction problem. For example, determining the human body part is one of the main problems that is solved by using structured output prediction methods [45, 46, 47]. A few other examples– among many, that use structured output prediction for solving computer vision problems: Taskar et al. used structured prediction for labeling the points in a 3D point cloud which was sampled from objects like trees, builings, etc. [2]; Based on the fact that the objects are hierarchically built from smaller parts, Girshick et al. formulated the object detection problem as an structured prediction problem [48]; Activity detection in videos can be formulated as an structured prediction problem [49, 50, 51].

- Security: The structured prediction algorithms also have applications in security problems. I will expand these applications later in Chapter 4, but here I refer to one security related application: the optimal placement of police in different locations in the airport to minimize the possibility of a terrorist attack. This task requires structure output prediction, because of the combinatorial nature of possible placement choices [52].

There are many other problems in NLP, computer vision, and security that fall under the heading of structured prediction. These include entity detection and tracking, automatic document summarization, machine translation and question answering, video segmentation, occlusion detection, surveillance systems and etc.

The main common theme in all structured output prediction problems is the combinatorial nature of the output. 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.

2.3 Feature Space for Structured Prediction

A key concept in the state-of-the-art structured prediction algorithms is the notion of an extended feature function. The inputs of the feature functions are both the original input $\boldsymbol{x} \in \mathcal{X}$ and a hypothesized output $\tilde{\boldsymbol{y}} \in \mathcal{Y}$. Therefore, we define $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ as the feature function, which again is a vector, and depends on both the input and the output. The mathematical details of $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ are problem-specific. For example, in graphical models [53], the feature function is the same as the vector of all potential functions [2, 16, 54], and in maximum entropy (MaxEnt) models [55] (or equivalently in log-linear models), the sufficient statistics are used as the feature functions.

In general, the choice of $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ 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 $y_i \in \{\text{“spam”}, \text{“non-spam”}\}$ 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:

$$\phi_{jk}(\boldsymbol{x}, \boldsymbol{y}) = \sum_i x_{ij} y_{ik} \tag{2.4}$$

$$\phi_{ekk'}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i,j} e_{ij} y_{ik} y_{ik'} \tag{2.5}$$

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

$$score(\boldsymbol{x}, \tilde{\boldsymbol{y}}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}, \tilde{\boldsymbol{y}}) \tag{2.6}$$

$\boldsymbol{w}$ is called the model weight vector, and the goal of the machine learning algorithm, is to learn such that the true labeling $\boldsymbol{y}$ gains the maximum score when plugged into the score function. Unfortunately, it is possible that in some cases an alternate labeling $\tilde{\boldsymbol{y}}$ which is very different than $\boldsymbol{y}$ also gain a high score, therefore the learning algorithm needs to select a $\boldsymbol{w}$ that penalizes such scenarios. We want to learn $\boldsymbol{w}$ such that the closer $\tilde{\boldsymbol{y}}$ is to $\boldsymbol{y}$, the higher the score of $\tilde{\boldsymbol{y}}$ is. Therefore $\Delta(\tilde{\boldsymbol{y}}, \boldsymbol{y})$ is defined as a measure of dissimilarity between $\tilde{\boldsymbol{y}}$ and $\boldsymbol{y}$. The Hamming distance between $\tilde{\boldsymbol{y}}$ and $\boldsymbol{y}$ is one of the popular choices. The difference function $\Delta(\tilde{\boldsymbol{y}}, \boldsymbol{y})$ 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 $\boldsymbol{w}$, one will need to find the best output for a given input. This is the "argmax problem" defined in 2.7 and referred to as maximum a posteriori (MAP) inference:

$$\hat{\boldsymbol{y}}_{prediction} = h_{\boldsymbol{w}}(\boldsymbol{x}) = \arg\max_{\tilde{\boldsymbol{y}} \in \mathcal{Y}} \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}, \tilde{\boldsymbol{y}}) \tag{2.7}$$

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

2.4 Structured Prediction Methods

In this section, I briefly explain the state-of-the-art methods for weight learning in structured prediction methods.

2.4.1 Structured Perceptron

The structured perceptron is an extension of the standard perceptron [56] to structured prediction [28, 57, 58]. The algorithm of learning bmw is shown in Alorithm (1).

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 $\arg\max_{\tilde{\boldsymbol{y}} \in \mathcal{Y}} \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}, \tilde{\boldsymbol{y}})$ is not equal to the true $\boldsymbol{y}$. Note that the algorithm is only applicable when the resulting output is either equal to the true output or not. In other words, the difference function $\Delta(\tilde{\boldsymbol{y}}, \boldsymbol{y}) \in \{0, 1\}$. The consequence of this fact, is poor generalization of this algorithm to unseen data.

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

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

2.4.2 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 $\tilde{\boldsymbol{y}}$ (or sometimes generatively over both the input $\boldsymbol{x}$ and the hypothetic output $\tilde{\boldsymbol{y}}$), the feature function $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ is seen as the sufficient statistics of this distribution:

$$p(\tilde{\boldsymbol{y}}; \boldsymbol{x}, \boldsymbol{w}) = \frac{1}{z(\boldsymbol{x}, \boldsymbol{w})} e^{\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}, \tilde{\boldsymbol{y}})} \tag{2.8}$$

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

$$z(\boldsymbol{x}, \boldsymbol{w}) = \sum_{\tilde{\boldsymbol{y}} \in \mathcal{Y}} e^{\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}, \tilde{\boldsymbol{y}})} \tag{2.9}$$

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

$$\begin{aligned} L(\tilde{\boldsymbol{y}}; \boldsymbol{x}, \boldsymbol{w}) &= -\log p(\tilde{\boldsymbol{y}}; \boldsymbol{x}, \boldsymbol{w}) \\ &= -\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}, \tilde{\boldsymbol{y}}) + log(\sum_{\tilde{\boldsymbol{y}} \in \mathcal{Y}} e^{\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}, \tilde{\boldsymbol{y}})}) \end{aligned} \tag{2.10}$$

The maximum entropy framework is one of the most successful methods in structured prediction. For example McCallum et al. applied this method to sequence labeling problems [25], and a lot of follow-up work applied maximum entropy structured prediction in different disciplines [27, 59, 60, 61, 62, 63, 64].

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.

2.4.3 Maximum Margin Markov Networks

The Maximum Margin Markov Network ($\mathrm{M}^3\mathrm{N}$) class of structured prediction methods are a generalization of max-margin methods in classic machine learning (also known as support vector machines (SVM)) to structured output prediction settings. The early work by Taskar et al. [2, 3, 39] was followed by a large amount of other progress in development of max margin methods [1, 31, 45, 65, 66].

To date, the state-of-the-art structural SVM is the 1-slack formulation [67], which solves the following optimization program:

$$\begin{aligned} &\underset{\boldsymbol{w},\zeta}{\text{minimize}}\ f(\boldsymbol{w}) + C\zeta \quad \text{subject to} \\ &\zeta \geq \max_{\tilde{\boldsymbol{y}}}\ \boldsymbol{w}^T(\boldsymbol{\phi}(\boldsymbol{x},\tilde{\boldsymbol{y}}) - \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})) + \Delta(\boldsymbol{y},\tilde{\boldsymbol{y}}) \end{aligned} \tag{2.11}$$

$f(\boldsymbol{w})$ is a regularization function, that penalizes "large" weights. Depending on the application, $f(\boldsymbol{w})$ can be any convex function in general. Semi-homogeneous functions such as norms, or positive-powers of norms, 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.) $f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$ 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 (that's why it's called 1-slack) which should be minimized, along with the regularization function.

2.4.4 Reranking

Reranking is a common method that is used in natural language processing problems. Assume that we have an oracle that solves some inference problem, but instead of generating "the best" output generates a list of "n best" outputs. Then, the learner's goal is to build a second model for choosing "one output" from this "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 [57, 68].

This method considers a constant-sized list and optimizes a reranker to a loss function closer to the one that we are interested in. Reranking has been applied in a variety of NLP problems including parsing [57, 68, 69], machine translation [70, 71], question answering [72], semantic role labeling [73], and other tasks. A main feature of reranking is that different loss functions can be easily embedded into the algorithm and immediately tested. There are also some drawbacks. For example, in reranking algorithm, one should have an oracle for choosing n-best initial ranking which may not be available, or n maybe too large to be useful.

2.4.5 Search Based Structured prediction

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. [41] 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 classier may be applied. SEARN is able to learn prediction functions for different loss functions and different features functions.

The structured output is produced by search algorithm that generates a sequence of decisions at each time step. This process of search aims to find a structured output that maximizes a scoring function in general. The goal of search-based structured prediction is to find a function π that guides us through search. Formally, given an input $x \in \mathcal{X}$ and a state s in a search space $\mathcal{S}$, the function $\pi(x; s)$ tells us the next state or action to choose. The function π is called a policy.

Training of SEARN is performed in an iterative way. At each iteration a known policy is used to create new cost-sensitive classification examples. These examples are the decisions that the optimal policy should make to help us search in the right direction.

There are several other related work by the same authors [38, 74], in particular, Daumé' et al. extend SEARN to unsupervised and semi-supervised settings [74, 75].

Recently Doppa et al. [76], introduced the HC-search method for structured prediction. This search-based algorithm consists of two main components; a recurrent classifier and a cost function of the loss function on the training data that mimics the search behavior. A recurrent classifier constructs structured outputs based on a series

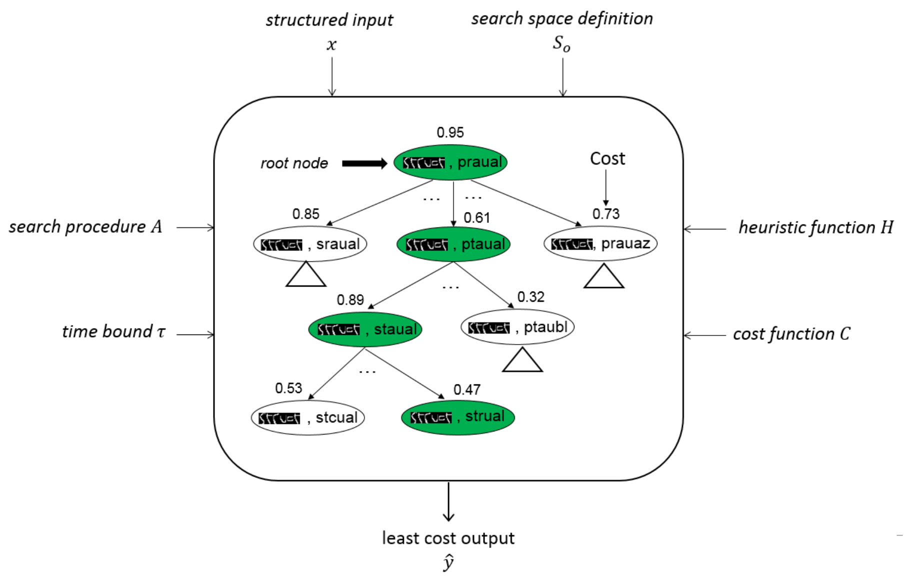


FIGURE 2.3: HC-search, Doppa et al. [76]

of discrete decisions. In order to learn a cost function, the authors choose a hypothesis space of the cost functions; if the performance of the cost function is good on the training data, then we hope that the performance on the testing data is also acceptable. The recurrent classifier and the cost function are trained sequentially. First, they train the classifier, and then use it to define the search space over output space for every training input. Secondly, they train the cost function to score outputs for a given combination of the search space and a search procedure. At test time, for each test input a search space is calculated by the recurrent classifier and the search procedure guided by the cost function returns the best output that is found in a limited time bound. There are several other follow-up papers by the same group [77, 78, 79].

Fig. 2.3 is a simple example that shows how HC-search method works in a high level. Given the input x and the search space S_0, HC-search starts with a search space that consists of all possible outputs. Then for a limited time of τ, a greedy algorithm $\mathcal{A}$ searches for the best high scoring outputs. This search is guided by the heuristic function $\mathcal{H}$, which heuristically determines which nodes should be expanded. The green nodes in Fig. 2.3 show the trajectory of node expansions. After reaching the time bound, the node expansion will be stopped. Then the best high-scored node based on the learned cost function will be returned as the final prediction.

Indeed, in all structured prediction problems– due to hardness of inference the quality of the solution is not acceptable unless a considerable amount computation time is spent. A main advantage of the HC-search algorithm is that it is capable of

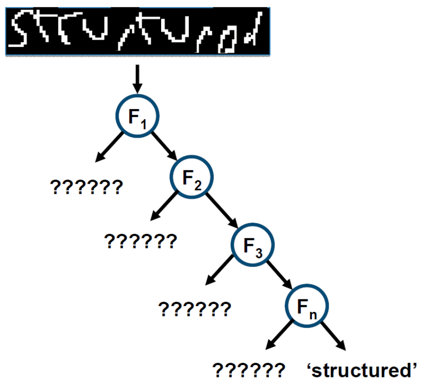


FIGURE 2.4: Structured prediction cascades, in each level some filtering on the output space is made based on the cheaper computation of the features [81]

controlling the trade-off between the quality of the solution and the invested time for solving the problem. From a critical point of view, the advantage that HC-search advertises for is mainly gained by heuristic branch and bound algorithm which has been known for a long time in the operational research community. As an example, commercial integer solvers such as Gurobi [80], also have the property of managing the trade-off between the quality of solution and the spent time[2]. However, HC-search is one of the first algorithms that supports the time bound, and is specifically tailored for structured prediction.

2.4.6 Structured Prediction Cascades

As mentioned in the previous subsection, inference time is the main bottleneck for both training and testing of the structured prediction algorithms. Since the feature vector is a function of both the inputs and the hypothetical outputs, it has to be calculated for every single candidate output. Calculation of some components of the feature vector are cheaper, but for some other components the computation of the feature can be quite expensive. As an example, consider a social network graph with one million nodes, and one hundred billion edges, then computing the number of the nodes that have the property "foo" takes about one million time steps, but checking the number of the neighboring node pairs that both have this property takes billions of time steps.

Structured prediction cascades [81] is a new method that is based on the following idea: avoid computing the expensive features when they don't help, especially when there are computational constraints. Since we should balance expected error

[2]Especially since the combinatorial problems can be reduced to integer programs.

with inference time, structured prediction cascades uses natural approach of using the pruning techniques when expanding the search tree for the high scoring outputs. In this method a tree (cascade) of possible solutions is build, where the closer you are to the root, the cheaper features are used for computing the score. Then, only the nodes that have higher score will be expanded (i.e. the more features will be calculated for the corresponding outputs of those nodes). This method is also further developed to ignore the effect of the features that do not have considerable contribution to accuracy of the prediction. Fig. 2.4 is a schematic overview of how we avoid expanding the nodes that are rejected early on.

Structured prediction cascades is mainly inspired by the Viola-Jones face detection algorithm [82], where a sliding window moved around the image, and if a patch is not face it is easy to reject that patch before computing all other feature from neighboring patches. The are several other related work that use heuristic methods for pruning the search space of outputs. In the previous subsection, I talked about HC-search [76]. Several other researchers also uses a similar coarse to fine technique for natural language processing problems [69, 83, 84].

2.5 Optimization Algorithms

In most of the methods that I 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 that itself needs to be solved. In the following, I briefly explain some of the state-of-the-art optimization algorithms that are used for structured learning.

- Cutting plane algorithm
  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 [3, 31, 39, 45, 65, 67]

- 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 [13, 39, 85, 86].

- Exponentiated gradient
  The exponentiated gradient algorithm also solves the optimization program in its dual form, and uses a gradient ascend 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 [87, 88, 89, 90].

There are some other optimization algorithms that are either not state of the art in the field of optimization, or are not widely used in existing structured prediction algorithms. But there are some rare works that have used these algorithms. For example, Fua et al. [91] use a trust region based optimization method for structured learning.

Chapter 3

Adversarial Machine Learning

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

Adversarial machine learning studies machine learning techniques that are robust against adversarial components, that rule over the process of input data generation. The need for adversarial machine learning algorithms is becoming more obvious these days, as security challenges are increasing [21]. In analogy with security problems, adversarial machine learning can be seen as a game between two players, where one player wants to protect the healthy functionality of a system, and the other player wants to somehow pursue its own 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) [8]. There has been a rich set 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 [9, 10, 12, 13, 14, 15, 92, 93, 94, 95, 96, 97].

In the following section, I briefly address some of the main aspects of the state-of-the-art methods, and will discuss the common themes in adversarial machine learning algorithms. I will also talk about regret minimization algorithms, that are somehow 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 from some space" had been taken at all the times. The "best fixed action" is the action that would have been chosen, if all the costs were known in hindsight.

Blaine Nelson's PhD thesis covers an extensive collection of possible threats that make most of the classical machine learning algorithms vulnerable [93]; in this report, my perspective is mostly from the learner's point of view, and I categorize the adversarial attacks based on higher level properties of an adversary.

3.1 Adversary's Theoretical Model

I start this section by some definitions:

Antagonistic adversary and zero-sum games: The adversary's goals are explicitly against the duties of the learner. By "explicit", I mean the adversary's win equally means the learner's loss and vice-versa; these adversarial games are called zero-sum games, and such an adversary is known as an antagonistic adversary.

Non-antagonistic adversary and non-zero-sum games: If the adversary's goals are implicitly against the learner's goals, then the adversary is seeking its own benefits, that may or may not be directly harmful for 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 main goal of the adversary, then it is a non-antagonistic adversary.

The non-antagonistic adversarial machine learning is not studied much yet for general cases, and most of the existing work focus on antagonistic adversarial opponents that actively try to defeat the machine learning algorithm, so that it fails by decreasing some performance measure.

Modeling the non-zero-sum game is relatively simple. Let $\boldsymbol{w} \in \mathcal{W}$ be the parameters of the learners model, and $\boldsymbol{a} \in \mathcal{A}$ be the parameters of 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 $r_a(\boldsymbol{w}, \boldsymbol{a})$ be the loss function that the learner wants to minimize by choosing the right $\boldsymbol{w}$[1]. An antagonistic adversary wants to maximize the loss of the learner by selecting appropriate $\boldsymbol{a}$. Therefore, the adversarial game can be formulated as:

$$\min_{\boldsymbol{w}\in\mathcal{W}} \max_{\boldsymbol{a}\in\mathcal{A}} \quad r_a(\boldsymbol{w}, \boldsymbol{a}) \tag{3.1}$$

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

A more general abstraction of adversarial games is presented in Algorithm 2.

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 $\boldsymbol{w}$ of the model $\mathcal{M}$ based on $\hat{\mathcal{A}}$ and the
      previous observations.
  • Adversary's prior belief:
    – The adversary chooses a set of strategies $\mathcal{A}$ based on its own prior knowl-
      edge and restrictions.
    – The adversary initializes its initial belief on the learner's model $\hat{\mathcal{M}}$, and
      its belief on the model parameters $\hat{\boldsymbol{w}}$.
    – 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 $\boldsymbol{w}$ based on $\hat{\mathcal{A}}$, and the observed adversary's
    action $\boldsymbol{a}$.
  • The learner gains reward $r_l(\boldsymbol{w}, \boldsymbol{a}) \in \mathcal{R}$
  Attack:
  • The adversary chooses an attack $a \in \mathcal{A}$
  • The adversary gains reward $r_a(\boldsymbol{w}, \boldsymbol{a}) \in \mathcal{R}$
  • The adversary updates $\hat{\mathcal{A}}$ based on the observed reward $r_a(\boldsymbol{w}, \boldsymbol{a})$ and its new
    understanding of $\mathcal{R}$.
  Nature:
  • nature updates $\mathcal{R}$.
end while
```

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 based on the adversary's prior belief about the learner's choice for 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, which decides on the amount of positive or negative pay-offs 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 important 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 (in this example users are a part nature).

Algorithm 2 encapsulates many details and possible assumptions; and, the order of the itemized events can be completely arbitrary; this makes the abstraction of the adversarial game in Algorithm 2 too generic to be useful. Each of the existing approaches on adversarial machine learning is designed based on some assumptions about Algorithm 2. In the following, I briefly categorize the main possible assumptions.

3.2 Type of Adversarial Problems

The assumptions about the order of occurrence of events in Algorithm 2 are a key point of difference in algorithmic approaches, that are designed for the adversarial machine learning. In particular, the existing studies are mostly based on three general assumptions about the possible orders of occurrence of events:

- **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 the 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 [12, 13, 15, 16, 97].
- **Based on Nash Equilibria:** In these models, although the order of events is arbitrary, but hypothetically, there exist optimum joint strategies of both players, where no player gains more rewards by deviating from its current strategy. It is a known fact from Game Theory that such optima do not necessarily exist among pure strategies[14, 94, 95, 96].
- **Based on Poisoning the Training Data:** The adversary generates several especially designed data points, and injects them into the training data. The adversary's goal in this kind of attacks, is to make the machine learning algorithm learn a wrong model in the first place. There are several research papers that explain how such attacks can be designed to target certain machine learning algorithms [22, 100, 128, 129, 130]. I will discuss about the poisoning attacks a little more in Section 4.1.

- **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; compared to the time, that if it knew all of the costs imposed by nature in hindsight and had chosen a fixed strategy as the response [23, 98, 99].

In general, it is not easy to find the Nash equilibrium when the dimensionality of the players' actions is large and the utility functions are arbitrary. Brückner and Scheffer 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 [14]. Therefore, Stackelberg competitions are more approachable techniques, because the learner should select the strategy that restricts the worst-case adversary in a mini-max 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 [12, 13, 14, 15, 94, 95, 96, 97], is the continuity of feature functions, which does not hold in many domains.

Globerson and Roweis [12] 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. In a later paper with Teo et al. [13], they generalized the former method to all invariants of input data, as long as there exist an efficient numerical procedure for calculating the invariants[2]. In Teo et al. the notion of invariant is seen as a transformation function with limited number of possible outcomes [13], and the formulation has a summation over all possible transformations. This is not practical whenever the number of possible transformation is exponential in the input size (or sometimes infinite). Torkamani and Lowd solved this problem for collective classification in associative Markov networks[3], which is a specific domain of structured prediction [16].

There are other related work, that have made small changes in the objective of the optimization program or have used a slightly different technical representation to the framework that Teo et al. [13], introduced. For example, Dekel et al. [100] also look at feature deletion and corruption, but in their formulation they use an L_1 norm for penalizing large weights instead of a more common L_2 norm. This formulation leads to

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

[3]In an associative Markov networks, the vertices of the data graph that are neighbor to each other are more likely to have the same label.

a linear program that can be solved by the simplex method [101] instead of a quadratic program.

There are some other papers that focus on the distribution of the train and the test data: for example Livni et al. [102] give an explanation for the reason , that why the common L_2 regularization is efficient for SVMs by assuming the existence of stochastic adversaries. There are also some related work on covariate shift when both the train and the test data are available at train time [103, 104].

3.3 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 learners model and try to infer parameters of the algorithm which is working behind the scene, by attacking the system, and observing the outcomes. Active adversaries have full access to the learner's model and the parameters that the learner has selected for the model [9, 10, 92]. 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. To the best of my knowledge, all of the existing methods in the field of adversarial structured prediction assume that there exists an active adversary that has access to the learner's model (except in the regret minimization setting where the scenario is different); I think, focusing on the passive scenarios for structured prediction, is a potential field of research in adversarial machine learning.

It is also important for the learner to know the adversaries 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 [105].

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 [12, 13, 16, 102]. Some other recent papers have considered the relationship between regularization and robustness to restricted adversaries in SVMs. Xu et al. [17] 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 work that expand this idea in different directions [106, 107]. For example in follow-up work Xu et al. expand their approach to the robust regression problem[106]. I will expand regularization methods in Chapter 4.

3.4 The Role of nature

In Algorithm 2, I have separated the adversary and nature. I believe that the adversary is in fact following the rules that nature sets. For example, in stock markets there are definitely some traders (adversaries) who want to increase their pay-offs by choosing the right portfolio, but the demands in 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 a 2 feet tall piece of rock is definitely 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.

Please note, that there can be many variations in the order of execution of events, and assumptions about the learner, adversary and nature in Algorithm 2, which I skip for brevity.

3.5 Regret Minimization

In a high-level definition, the "regret minimization" algorithms are designed to choose the right sequence of actions when the cost of each action is not known in advance. The goal is to compete well against one single action that would be chosen if the costs were known in hindsight. The measure of regret is the difference between the performance of the online algorithm– that chooses actions at each time step, and the best benchmark in the class [108].

As I will discuss in Section 4.1, the sequential nature of the prediction in the regret minimization framework, makes it a potential method for performing structure prediction. If the structured prediction problem is reducible to a sequence prediction problem, then a regret minimization framework is a good match.

3.5.1 Motivation and Definition of Regret Minimization

The first motivation of regret minimization research was based on the idea of forecasting a sequence with the advice of some oracle, that unveils the true output when the learner has made its decision. As an example, suppose there is a horse race in Eugene every Sunday morning, where 20 horse-riders with $horse_1, \ldots, horse_{20}$ compete against each other. You can bet on a number from 1 to 20, and after the game is over the horse that you have bet on stands on jth position. The game rule is: if $j - 6 > 0$, then you should pay $j - 6$ dollars to the organizer, otherwise you will wager $6 - j$ dollars (Yes! not a fair game.). Now after 10 weeks of losing different amounts of money, you may ask yourself: "what if I had chosen $horse_9$ all the times? She has had the maximum number of winnings on average!"

In this example, the amount of regret is the money that you have lost minus the money that you would have lost (winnings can be considered as negative lost) if you had bet on $horse_9$ every single time. The regret minimization algorithms aim for decreasing the expected amount of regret in sequential decision making processes.

More formally, the learner (forecaster ideally wants to predict a sequence of outputs $y_1, y_2, \ldots \quad y_i \in \mathcal{Y}$, where $\mathcal{Y}$ is some output space. The learner's predictions $\hat{y}_1, \hat{y}_2, \ldots$ will approximate the goal sequence. After the learner makes the decision $\hat{y}_i$ at the ith time step, it has to pay some cost $c_i(\hat{y}_i)$, where the cost function c_i, is (adversarially) determined by nature, and $c_i(\hat{y}_i) \geq c_i(y_i)$.

Now the regret measure is defined by:

$$Regret(\hat{y}_1, \ldots, \hat{y}_n) = \sum_{i=1}^{n} c_i(\hat{y}_i) - min_{y^*} \sum_{i=1}^{n} c_i(y^*) \tag{3.2}$$

therefore, the goal of regret minimization is to find $(\hat{y}_1, \ldots, \hat{y}_n)^*$ such that:

$$(\hat{y}_1, \ldots, \hat{y}_n)^* = \underset{(\hat{y}_1, \ldots, \hat{y}_n)}{\text{minimize}} \;\; Regret(\hat{y}_1, \ldots, \hat{y}_n) \tag{3.3}$$

Note that the amount of regret can be negative in theory. In the next subsetion, I briefly explain a few of the base-line and state-of-the-art algorithms for solving the regret minimization problem.

3.5.2 Regret Minimization Algorithms

The sequential nature of the predictions in regret minimization's setting suggests its connection to online learning[4] [98]. Therefore, all of the guarantees that online learning provides are aslo applicable to regret minimization algorithms.

3.5.2.1 Follow The Leader

The "Follow The Leader" (FTL) algorithm is the most basic strategy for regret minimization. At each iteration the FTL algorithm makes a decision, that is the best choice based on the data that is observed so far. In other words the algorithm tries to find the fixed point that minimizes the average of all previous costs:

$$\hat{y}_{(t+1)} = \arg\min_{y \in \mathcal{Y}} \sum_{i=1}^{t} c_i(y) \tag{3.4}$$

This basic algorithm can easily get trapped in the loop of switching strategies [109]. As a simple example, assume that the output space $\mathcal{Y} = [-1, 1]$, then if from the second iteration, nature chooses the the cost function $c_i(y) = y_1(-1)^i y$, then the FTL algorithm will prediction the wrong answer in each following iteration.

3.5.2.2 Regularized Follow The Leader

Since the predictions of FTL may have wild variations from one iteration to the next one, the FTL algorithm is know to be flawed. The regularized FTL (RFTL) is a modification of FTL to make it more stable. RFTL solves the FTL's problem by regularizing its objective:

$$\hat{y}_{(t+1)} = \arg\min_{y \in \mathcal{Y}} \; \lambda\Omega(y) + \sum_{i=1}^{t} c_i(y) \tag{3.5}$$

[4]Online learning is a branch of machine learning, where all of the data samples are not available to the learning in advance, and the learner observes a new sample at each iteration. In online algorithms the model parameters are updated iteratively as the learner observes new samples. Online learning algorithms provide efficient ways of data processing, which make them appropriate for large-scale data processing in machine learning. Another benifit of using online algorithms is the convergence guarantees that are comparable to batch data processing algorithms.

The regularization function $\Omega(y)$ is assumed to be strongly convex (i.e. its Hessian should be positive definite). Algorithm (3) shows an sketch RFTL.

Algorithm 3 Regularized Follow The Leader Algorithm

```
**Input:** Regularization coefficient $\lambda$, strongly convex regularization function $\Omega(y)$, output space $\mathcal{Y}$, The horizon of predictions $T$
**Initialize:** $\hat{y}_1 = \arg\min_{y \in \mathcal{Y}} \ \Omega(y)$
for $t = 2$ to $T$ do
    Observe the last cost $c_{t-1}(y)$
    Update: $\hat{y}_t = \arg\min_{y \in \mathcal{Y}} \ \lambda\Omega(y) + \sum_{i=1}^{t} c_i(y)$
end for
```

A special case of RFTL is know as "multiplicative updates", where $\boldsymbol{y}$ is a discrete probability distribution (i.e. $\mathcal{Y} = \{\boldsymbol{y} | \ y_i \geq 0, \ \sum_i y_i = 1\}$), the chosen regularization function is a negative entropy measure: $\Omega(\boldsymbol{y}) = \sum_i y_i \log y_i$, and $c_t(y) = \boldsymbol{y}^T \boldsymbol{\phi}_t$, then the updates will be:

$$\hat{\boldsymbol{y}}_{(t+1)} = \frac{\hat{\boldsymbol{y}}_t \bullet e^{\lambda \boldsymbol{\phi}_t}}{\hat{\boldsymbol{y}}^T e^{\lambda \boldsymbol{\phi}_t}} \tag{3.6}$$

where $e^{\lambda \boldsymbol{\phi}_t}$ is a vector made by stacking the element-wise exponentiations of elements of $\boldsymbol{\phi}_t$ multiplied by λ, and the operation $\bullet$ is the Hadamard product of two vectors.

A second special case when $\Omega(\boldsymbol{y}) = \|\boldsymbol{y}\|_2^2$, and $\mathcal{Y}$ is a unit L_2 ball (i.e. $\|\boldsymbol{y}\|_2 \leq 1$), then the updates will be:

$$\hat{\boldsymbol{y}}_{(t+1)} = \frac{\hat{\boldsymbol{y}}_t - \lambda \boldsymbol{\phi}_t}{\|\hat{\boldsymbol{y}}_t - \lambda \boldsymbol{\phi}_t\|_2} \tag{3.7}$$

In the first special case, a multiplicative correction is made on the output, and in the second special case a translation correction is applied. Then, in both cases, the output is projected back to the feasible space.

3.5.2.3 Regret Bounds

It can be proven that for $c_t(y) = \boldsymbol{y}^T \boldsymbol{\phi}_t$, the regret that is incurred after T iterations of running Algorithm (3) is less than or equal to $2\sqrt{2\gamma DT}$[5]:

$$Regret_T = \sum_{t=1}^{T} \boldsymbol{\phi}_t^T(\boldsymbol{y}_t - \boldsymbol{y}) \leq 2\sqrt{2\gamma DT} \tag{3.8}$$

where $\boldsymbol{\gamma} = \max_{t,y\in\mathcal{Y}}\ \boldsymbol{\phi}_t^T H^{-1}\boldsymbol{\phi}_t$, $D = \max_{\boldsymbol{y}\in\mathcal{Y}}\ \Omega(\boldsymbol{y}) - \Omega(\boldsymbol{y}_1)$,and H^{-1} is the inverse of the Hessian matrix of the regularization function.

Having a bound on the regret is very important, because it give us an upper-bound on the maximum loss that we might incur. There is an active research on theoretically tightening the regret bounds for different algorithms.

3.5.3 Applications of Regret Minimization

In the following, I list some of the applications of regret minimization:

- **Prediction from experts advice:**In this class of problems, leaner has to select one on n options (experts advice). After selecting one of them in an iteration, the leaner incurs a loss; and this process repeats again. The goal is to choose a sequence which does as well as the best expert would do in hindsight.
- **Online shortest paths:** In the online shortest path problem a directed graph $G = (V, E)$, a source node s, and a sink node t are given. At the tth iteration, the leaner selects a path from s to t, and then an adversary chooses the edge weights. The cost of the learner will be the sum of the edge wights of the chosen path. The goal is to incur the minimum cost compared to one fixed path in hindsight [111]. This setting has applications in traffic control problems. Takimoto et al. [111], use the multiplicative updates on the relaxed program to solve this problem.
- **Portfolio selection:** In the portfolio selection problem, the goal is to choose a distribution $\boldsymbol{y}_t \in \mathrm{R}^{\mathrm{n}}$, $\sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{y_{i,t}} = 1$ of wealth over n assets at the tth iteration. nature (adversary) chooses the market return vector r_t for all assets; and the learner will be rewarded $\boldsymbol{y}_t^T r_t$. The goal is to maximize the overall reward (or

[5]The proof of this theorem is based on the Bregman divergence of two possibles outputs $\boldsymbol{y}, \boldsymbol{z}\mathcal{Y}$ with respect to the regularization function, and the primal-dual optimization algorithm for updating $\boldsymbol{y}_t$. I skip the proof here, for the proof please refer to Hazan [110].

minimize the negative loss), over T iterations. Hazan et al. use a Newton based minimization algorithm fo rsolving this problem [112].

Chapter 4

Adversarial Structured Output Prediction

The broad impacts of adversarial structured prediction are in the problems where space of possible outputs is large and the robustness of the machine learning algorithm is important. While, many areas of security need to deal with these two issues; recently, researchers have encountered other emerging problems with similar difficulties. Most of these problems have emerged in the field of computer vision. It is worthwhile to mention that, many of the computer vision problems such as video surveillance have implicit applications in security problems.

As I mentioned earlier in this report, there are only a few existing work that directly refer to the problem of adversarial structured output prediction. In this section, I explain the existing applications, and at the same time refer to some of the limitations of these works.

I should mention the existence of some work that study robustness from the generalization point of view [113]. Most of these results are theoretical extensions of generalization bounds from PAC-learnability, and I don't focus on them in this report [1].

[1]Probably-Approximately-Correct (PAC) learning, refers to some theorems that are foundation of computational learning theory. In simple words, PAC-learnability relates the generalization error of the machine learning algorithm at test time to the (number of) samples that are observed at train time[114].

4.1 The State-of-the-Art Techniques

4.1.1 Utility-based Adversarial Structured Learning

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 own utility functions. The utility functions can be a likelihood function or some other reward function. In a game theoretic framework each of the players tries to maximize its own utility. Brückner and Scheffer have several papers that are based on this approach. They have shown that for certain non-antagonistic utility functions, the prediction game has a unique Nash equilibrium, and derive a simulation-based algorithms for finding the equilibrial prediction models [14, 95]. 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 [15]. This framework is in fact a minimax scenario where the learner tries to minimize the maximum possible negative damage that the adversary can cause. None of these works are designed for structured prediction problems. Since, the framework that these methods suggest is based on simulation of the game; it is applicable to structured settings as well. Although, 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 even non-structured) output spaces.

There are other works that expand the analysis of the conditions for finding the Nash equilibrium, for example Dreves et al. have analyzed the Karush-Kuhn-Tucker (KKT) conditions for which the generalized Nash equilibrium exists [94].

4.1.2 Max-margin-based Adversarial Structured Learning

I explained the basics of the max-margin structured prediction in Chapter 2. The max-margin based algorithms include the large class of structural SVMs, and therefore is the main branch of methods for which learning in adversarial settings is studied.

4.1.2.1 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. Schoÿlkopf et al. [115] were one the first authors who used the idea of using virtual (noisy) samples for training the model. This idea was first used as an embedded part of the algorithm for binary SVMs by Globerson and Roweis [12], when a limited number of the features could be set to zero by the adversary at test time. Later Teo et al. [13] expanded this idea to include a wider class possible adversaries. The main limitation of this 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. CACC [16] addressed this issue for a more specific structured prediction problem.

There are other approaches that have a similar nature, for example Biggio et al. [116] formulate the problem in the dual form and model the adversarial noise as 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 those kind of perturbation [102, 117].

4.1.2.2 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 [118, 119, 120, 121] showed that there exist a wide range of applications that could be formulated in a robust convex optimization framework. Robust linear programming is a central method in most of the robust formulations. Bertsimas et al. [122] 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 Bertsimal et al. [123], the authors focus on the case when the disturbance of the inputs is restricted to an ellipsoid around the true 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.,[124, 125, 126, 127]).

Recently, Xu et al. [17] showed that regularization of support vector machines can be derived from a robust formulation, and also argue that robustness in feature space entails robustness in sample space.

Recently, we came up with a new method for improving the robustness of structured SVMs in presence of worst-case adversary [18]. As in Bertsimas et al. [123], we also assumed that the disturbances in feature space are restricted to the interior (and the boundary) of some ellipsoid. We showed that the robust formulation of structural

SVMs is equivalent to some regularization of the models' weights. This extends similar results by Xu et al. [17] from binary SVMs to structural SVMs. We also explored situations in which the disturbances are restricted to an arbitrary polyhedron. Finally, we showed that we can generalize both formulations to a single program where disturbances are constrained by the intersection of an ellipsoid and a polyhedron. By showing these equivalences between robust structural SVMs and regularized or transformed versions of non-robust SVMs, we achieve a deeper understanding of how robustness affects the structured learning problems as well as suggesting new methods for solving these optimization problems.

Based on the insight that I mentioned above, we developed a general theory of robustness for arbitrary structural support vector machines (SVMs). Structural SVMs address a wide class of problems that include collective classification problem that we had previously solved with CACC. But this time, we showed that by selecting the appropriate regularization function for a specific perturbation set in the feature space, the desired robustness to that perturbation set can be achieved.

4.1.2.3 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 this kind of attacks. I think, filling this gap is worthwhile, and is specifically important to applications where the number of training samples is limited. Biggio et al. [22] have studied this problem for non-structural prediction. They investigate a family of poisoning attacks against SVMs. Note that most of the learning algorithms assume that their training data comes from a natural distribution, and therefore are vulnerable to this 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 and Shamir [100] 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. But I think, the reason that this regularization works better for them, is because of the effect of dual norms that we have explained in our submission to ICML 2014 (See the previous subsection for details.). There are some other works in the literature, that attempt to train models that are robust to poisoning attacks [128, 129, 130].

4.1.3 Online Learning and Regret Minimization

There are not much work that use online learning [98] for structured prediction. I have found a few that I list here. Online learning can be seen both as a frame work for decomposing the problem of structured prediction to small non-structured problems and solving them individually and also as an efficient and scalable optimization framework for learning. The former application of online learning, i.e. decomposing the problem to smaller problems, is mostly used for sequence prediction, which is highly related to similar regret-minimization methods. For example Arindam Banerjee uses online updates for learning a logistic regression model for structured output prediction [131]. In this paper, Banerjee uses online updates in order to learn the parameters of a log-linear model for structured prediction in a CRF-based setting. There are other similar methods that use the same framework[2] for structured prediction, but Banerjee's main result is to show that, the online updates can provide the same convergence guarantees as batch updates. There is several works that use similar techniques. For example, Hall et al. perform online updates to train structured perception [132].

Ross et al. have several papers that address structured prediction in regret minimization framework; their basic idea is to decompose the structured output into a sequence prediction problem and solve it by the regret minimization framework [23, 99].

Sequence prediction problems arise in many different settings, but one can reduce some of the structured prediction problems to sequence prediction problems. For example, a robot system needs to predict a list of sub-actions in order to perform a super-action such as: drive a car from location “A” to location “B”. This super-action can be seen as an structured output, but it is decomposable to a sequence of sub-actions. Imitation leaning– where the machine learning system is trained to mimic some expert's behavior, is a good fit for such problems [99].

Ross et al. [99] use a regret minimization setting for learning to drive a computer simulated car, where the output is a sequence of actions in limited horizon. In their problem, the true cost of taking action a in state s: $C(s, a)$ is not known, but they use some expert's knowledge about the loss $l(s, \pi)$ incurred by policy $a = \pi(s)$ [3], and try to minimize the empirical expectation of this loss function:

$$\hat{\pi} = \arg\max_{\pi \in \Pi} \; \mathrm{E}[l(s, \pi)] \tag{4.1}$$

[2]E.g. Domke's paper on structured prediction using logistic regression [64]

[3]The term “policy” is commonly used in the reinforcement learning community for the function $\pi : \mathcal{S} \to \mathcal{D}(\mathcal{A})$ that maps an state to an action or a distribution over actions, and is almost equivalent hypothesis function $h : \mathcal{X} \to \mathcal{Y}$.

The expert's loss $l(s, \pi)$ is measured by the average number of times that the simulated car falls out of the road. Their approach is similar to regularized follow the leader algorithm that I explained in Section 3.

To conclude, I think there is a great potential of using regret minimization framework, for solving structured output predictions; where, the output can be decomposed to a sequence of simpler prediction problems.

4.2 Applications

4.2.1 Collective Classification

Many real-world relational learning problems can be formulated as a collective classification problem. For example, web-spam detection can be formulated as a collective classification problem where each web-page is either spam or non-spam, and the label of each web-page not only depends on its contents, but also depends on the label of neighboring web-pages that are linked to it [1, 5, 16].

To the best of my knowledge, our paper "Collective Adversarial Collective Classification" [16], is the only published work in the field of structured output prediction, that is designed to be directly robust against adversarial manipulation of data at test time. In that work, we assumed that the adversary can change up to D attributes of all web-pages, and by incorporating this limitation of the adversary [4] in a robust optimization program, we came up with an efficient method[5] for robustly solving the problem of collective classification in associative Markov networks [2].

There are some other papers that try to solve this problem with implicit effort to address the robustness issue. Sen et al. [1] discuss that the "Iterative Classification Algorithm" [133, 134] is fairly robust to the order that the nodes are visited (not the manipulation of test data). Tian et al. [135] introduce a heuristic additional weight on top of a dependency network [136, 137] to model the strength of the dependencies; although, this paper claims that this additional weight makes the method robust to random noise, but this method is only ad-hoc and particularly not robust to malicious noise. McDowell et al. [138] introduce cautious iterative classification algorithm, where at each local classification, the classifier also generates a confidence criterion about the classification that it has made; if this criterion is less than some threshold, the predicted

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

[5]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.

label is ignored by the algorithm. This method is also heuristic and does not rely on related literature of robust machine learning. Bilgic et al. [139] introduce an iterative method that combines collective classification and link prediction, in their work they only show that the proposed method is robust different homophily parameters in the network.

Abernethy et al. have done several works in the field of adversarial machine learning, but most of them do not address the adversarial attacks that can be performed against collective classification algorithms. In one of their papers they introduce "WITCH" algorithm [5], that uses a graph regularization approach to use the link information for regularizing the model parameters. Since their approach gains some implicit robustness due to regularization, I think it is also a related work that needs to be mentioned here.

4.2.2 Anomaly Detection

Anomaly detection is the problem of detecting unusual samples among a number of ordinary samples. For example detecting network intrusions or credit card fraud detection is known examples of anomaly detection. There exist a large literature in both statistics and machine learning on anomaly detection; in this part, I refer to some of these works use structured output prediction (especially conditional random fields) for performing the anomaly detection task. 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 some sort of structured prediction to determine the role of each agent in the network. There are a group of papers that use conditional random fields or hidden Makov model to perform this task [140, 141, 142], a main drawback of these method 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. introduce a one-class classification approach for detecting the sequential anomalies [143]; their method is robust to outliers in the training data; I find it very interesting. The main fact that makes this work less applicable to adversarial settings is that, the adversarially manipulated samples are different than outliers. In particular, the adversary manipulates them as a response to the learned parameters of the classification method.

4.2.3 Game Theory for Security

Security issues are becoming more serious and critical problem these days, and naturally machine learning tools are used to solve some of these problems. As explained in section 3, these security challenges can be formulated as a game between the defender (or learner) and the attacker (or the adversary), therefore we call them security games. Not only the action space in security games is large, but also the limited resources of the defender is a challenge in most of the security games. In fact, in real world security problems there are not enough agents to patrol all the targets that the adversary could attack to; therefore, deciding the placement of the resources is highly important. As an example, Pita et al. have developed and algorithm called ARMOR [52, 144] which is now deployed at the Los Angeles International Airport (LAX) to randomize the checkpoints on the roadways that enter the airport. Randomization in this context basically means that the strategies are drawn from some mixture of strategy distributions rather than a fixed pure strategy. In game theory random or mixed strategies are important, because then the opponent will not be able to surely determine your next action.

There are several other papers from Prof. Milind Tambe's lab from University of Southern California, that address similar problems; I bring some these examples in the following: IRIS is a randomized large-scale scheduling system for US Federal Air Marshal Service [145]. This algorithm is used for scheduling air marshals to be allocated to flights departing from United States, and prioritizes the flights by considering different factors such as the number of passengers, population of destination and destination itself, etc.İn IRIS the targets are the number of flights at each time step (day), and since the number of possible schedules are exponential in the number of flights; some other embedded algorithms are used for fast generation of the schedules [146]. PROTECT is designed for allocating US Coast Guard for securing coasts, ports, and inland waterways, due to threats such as terrorism and drug trafficking [147].

Recently, in a follow-up version of PROTECT, Fang et. al. introduced an algorithm where the resources can be mobile [148]. Another application called GUARDS (Game-theoretic Unpredictable and Randomly Deployed Security) is designed by this group for the US Transportation Security Administration that protects airports. Unlike ARMOR, GUARDS it is designed to address the diversity of attacks, e.g. existence of heterogeneous adversarial activities and diverse potential threats [149]. There are several other contributions by this group and other groups that all use Stackleberg games for solving other interesting but similar challenges that I skip for brevity [150, 151, 152, 153, 154, 155, 156]. Dickerson et al. [157] look at security games from a graph theoretic approach and propose a greedy algorithm for protecting the moving targets from adversaries.

4.2.4 Computer Vision

Computer vision applications of adversarial structured predication are probably the most well studied ones for one important reason. Both robustness and structured output prediction are hot research topics in the computer vision community. In the following, I cite some the work that are related to robust structured prediction in the field of computer vision.

As mentioned earlier, these methods are not designed to be directly robust against adversarial manipulation of data, but have achieved some better robustness in comparison to previous works. For example Fua et al. [91] propose a working set based approximate sub-gradient 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. The randomly sample new constraints instead of computing them using the more expensive approximate inference techniques, and gain more robustness in the prediction by this method. 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. [158] propose a structured prediction method where the output space is subset of two distinct manifolds, and their method tries to be robust to noise and choose the output from the right manifold. This method is shown to be efficient in human motion capturing from videos. There are also some work, that robust features are chosen in advance to gain robustness in the structured prediction [159]. Exploiting the domain knowledge is also a method that can be used in order to gain robustness, Chen et al. use this ad-hoc approach for robust play type recognition in a football game, which is recorded by noisy sensors [160].

4.2.5 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 interesting example of such applications. Zhang et al. have parameterized a noise model, and have embedded it into the their optimization program. They optimize for this noise control parameter as well [161, 162]. In their problem the noise in the speech signal is not adversarial, and I think adversarial speech recognition is also one of the fields that have important applications in real world problems.

Chapter 5

Discussion and Future Plan

5.1 Discussion and Conclusion

In the following I highlight the key points that I mentioned throughout the report.

5.1.1 Improving Adversarial Machine Learning

As Algorithm (2) demonstrates there exist a wide range of possible directions in the field of adversarial machine learning that is not explored yet. I think, there is a possibility of theoretical expansion in adversarial machine learning. In particular:

- 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 opponents utility, then we will have a more realistic model of the adversarial game. Besides that we will be able to use decision theoretic approaches to model non-zero-sum games. Note that solving non-zero-sum games in adversarial settings is another important issue that needs to be addressed.
- Efficient use of knowledge about the opponent
  In CACC paper, and our submission to ICML2014 we have shown that by taking advantage of adversary's limitations, we can design more robust algorithms; but, there are still many details about how-to of translating 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.

5.1.2 Expansion of Existing Work to Structural Settings

There exist a large amount of work in adversarial machine learning that are designed for specific problems. I believe that 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 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 take benefit of 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.

5.2 Ongoing and Future Work

Our goal is to develop a complete theory of adversarial structured prediction. My current direction is exploring the possibilities from robust optimization point of view.

In longer term, we have planned to explore robustness in presence of non-antagonistic adversaries. We also want to analyze how robustness can be achieved in regret minimization and online learning algorithms, and also how these algorithms can be deployed to gain robustness. One of our goals is to evaluate our methods on data from several real-world adversarial problems, including Twitter spam, YouTube comment spam, and fake reviews. We also want to expand our current published and submitted works to wider range of possible adversarial manipulations on the raw data, and at the same time keep the optimization program as efficient as possible. Focusing on more game theoretic approaches is also our next priority.

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

Bibliography

[1] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29 (3):93, 2008.

[2] Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative Markov networks. In *Proceedings of the twenty-first international conference on machine learning*. ACM Press, 2004.

[3] Ben Koller, Taskar Carlos, and Guestrin Daphne. Max-margin markov networks. In *Advances in Neural Information Processing Systems (NIPS) 17*, Vancouver, BC, Canada, 2003.

[4] Pedro Domingos and Daniel Lowd. *Markov logic: An interface layer for artificial intelligence*, volume 3. Morgan & Claypool Publishers, 2009.

[5] Jacob Abernethy, Olivier Chapelle, and Carlos Castillo. Graph regularization methods for web spam detection. *Machine Learning*, 81(2):207–225, 2010.

[6] Isabel Drost and Tobias Scheffer. Thwarting the nigritude ultramarine: Learning to identify link spam. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 96–107. Springer, 2005.

[7] Duen Chau, Shashank Pandit, and Christos Faloutsos. Detecting fraudulent personalities in networks of online auctioneers. *Knowledge Discovery in Databases: PKDD 2006*, pages 103–114, 2006.

[8] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, Seattle, WA, 2004. ACM Press.

[9] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, pages 125–132, 2005.

[10] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647. ACM, 2005. ISBN 159593135X.

[11] Blaine Nelson, Benjamin Rubinstein, Ling Huang, Anthony Joseph, Shing-hon Lau, Steven Lee, Satish Rao, and Anthony Tran. 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, 2010.

[12] Amir Globerson and Sam Roweis. 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, 2006. ACM Press.

[13] Choon Hui Teo, Amir Globerson, Sam Roweis, and Alexander Smola. Convex learning with invariances. In *Advances in Neural Information Processing Systems (NIPS) 22*, 2008.

[14] Michael Brückner and Tobias Scheffer. Nash equilibria of static prediction games. In *Advances in Neural Information Processing Systems 22*, 2009.

[15] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the Seventeenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM Press, 2011.

[16] MohamadAli Torkamani and Daniel Lowd. Convex adversarial collective classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 642–650, 2013.

[17] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510, 2009.

[18] MohamadAli Torkamani and Daniel Lowd. On robustness and regularization of structural support vector machines. In *Proceedings of the 31th International Conference on Machine Learning (ICML-14)*, 2013.

[19] Marius Kloft and Pavel Laskov. A poisoning attack against online anomaly detection. In *NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security.* Citeseer, 2007.

[20] Pavel Laskov and Marius Kloft. 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, 2009.

[21] Pavel Laskov and Richard Lippmann. Machine learning in adversarial environments. *Machine learning*, 81(2):115–119, 2010.

[22] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[23] Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. In *In AISTATS*. Citeseer, 2011.

[24] Gökhan Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander Smola, and Ben Taskar. *Predicting Structured Data*. MIT Press, 2007.

[25] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, pages 591–598, 2000.

[26] John Lafferty, Andrew McCallum, and Fernando Pereira. 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, 2001. Morgan Kaufmann.

[27] Vasin Punyakanok and Dan Roth. The use of classifiers in sequential inference. *arXiv preprint cs/0111003*, 2001.

[28] Michael Collins. 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, 2002. ACL.

[29] Yasemin Altun, Ioannis Tsochantaridis, Thomas Hofmann, et al. Hidden markov support vector machines. In *ICML*, volume 3, pages 3–10, 2003.

[30] David McAllester, Michael Collins, and Fernando Pereira. Case-factor diagrams for structured probabilistic modeling. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 382–391. AUAI Press, 2004.

[31] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

[32] Ben Taskar. *Learning Structured Prediction Models:A Large Margin Approach*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 2005.

[33] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[34] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Francisco, CA, 1988.

[35] Matthew Richardson, Daniel Lowd, and Pedro Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[36] Pedro Domingos, Stanley Kok, Daniel Lowd, Hoifung Poon, Matthew Richardson, and Parag Singla. Markov logic. In *Probabilistic inductive logic programming*, pages 92–117. Springer, 2008.

[37] Tuyen Huynh and Raymond Mooney. 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, 2009.

[38] Hal Daumé III and Daniel Marcu. 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, 2005.

[39] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM, 2005.

[40] Tamir Hazan and Raquel Urtasun. A primal-dual message-passing algorithm for approximated large scale structured prediction. In *Advances in Neural Information Processing Systems 23*, volume 2, page 4, 2010.

[41] Hal Daumé Iii, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[42] Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I Jordan. Word alignment via quadratic assignment. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 112–119. Association for Computational Linguistics, 2006.

[43] Benjamin Taskar, Simon Lacoste-Julien, and Michael Jordan. Structured prediction via the extragradient method. In *Advances in Neural Information Processing Systems (NIPS) 19*, 2005.

[44] Ben Taskar, Simon Lacoste-Julien, and Michael I Jordan. Structured prediction, dual extragradient and bregman projections. *The Journal of Machine Learning Research*, 7:1627–1653, 2006.

[45] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176. ACM, 2009.

[46] Long Zhu, Yuanhao Chen, Alan Yuille, and William Freeman. Latent hierarchical structural learning for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1062–1069. IEEE, 2010.

[47] Ian Endres, Vivek Srikumar, Ming-Wei Chang, and Derek Hoiem. Learning shared body plans. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3130–3137. IEEE, 2012.

[48] Ross B Girshick, Pedro F Felzenszwalb, and David A McAllester. Object detection with grammar models. In *Advances in Neural Information Processing Systems 24*, volume 5, page 6, 2011.

[49] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1250–1257. IEEE, 2012.

[50] Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. Video summarization via transferrable structured learning. In *Proceedings of the 20th international conference on World wide web*, pages 287–296. ACM, 2011.

[51] Xinxiao Wu and Yunde Jia. View-invariant action recognition using latent kernelized structural svm. In *Computer Vision–ECCV 2012*, pages 411–424. Springer, 2012.

[52] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40(4):267–290, 2010.

[53] Steffen L Lauritzen. *Graphical models*. Oxford University Press, 1996.

[54] Jeff Bilmes, Geoff Zweig, T Richardson, Karim Filali, K Livescu, Peng Xu, K Jackson, Y Brandman, E Sandness, E Holtz, et al. Discriminatively structured graphical models for speech recognition. In *Report of the JHU 2001 Summer Workshop*, 2001.

[55] Henri Theil and Denzil G Fiebig. *Exploiting continuity: Maximum entropy estimation of continuous distributions*. Ballinger Cambridge, MA, 1984.

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

[57] Michael Collins and Nigel Duffy. 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, 2002.

[58] Ryan McDonald, Keith Hall, and Gideon Mann. 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, 2010.

[59] Mary Elaine Califf and Raymond J Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *The Journal of Machine Learning Research*, 4:177–210, 2003.

[60] Ryan McDonald and Fernando Pereira. Identifying gene and protein mentions in text using conditional random fields. *BMC bioinformatics*, 6(Suppl 1):S6, 2005.

[61] Ron Begleiter, Ran El-Yaniv, and Golan Yona. On prediction using variable order markov models. *J. Artif. Intell. Res.(JAIR)*, 22:385–421, 2004.

[62] Hai Leong Chieu and Hwee Tou Ng. A maximum entropy approach to information extraction from semi-structured and free text. *AAAI/IAAI*, 2002:786–791, 2002.

[63] Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. Document summarization using conditional random fields. In *IJCAI*, volume 7, pages 2862–2867, 2007.

[64] Justin Domke. Structured learning via logistic regression. In *Advances in Neural Information Processing Systems*, pages 647–655, 2013.

[65] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[66] Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. Structured models for fine-to-coarse sentiment analysis. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 432, 2007.

[67] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[68] Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

[69] Eugene Charniak and Mark Johnson. 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, 2005.

[70] Libin Shen, Anoop Sarkar, and Franz Josef Och. Discriminative reranking for machine translation. In *HLT-NAACL*, pages 177–184, 2004.

[71] Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, et al. 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*, 2003.

[72] Deepak Ravichandran, Eduard Hovy, and Franz Josef Och. 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, 2003.

[73] Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 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, 2005.

[74] Hal Daumé III. Unsupervised search-based structured prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 209–216. ACM, 2009.

[75] Hal Daumé III. Semi-supervised or semi-unsupervised? In *NAACL Workshop on Semi-supervised Learning for NLP*. Citeseer, 2009.

[76] Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. Output space search for structured prediction. *arXiv preprint arXiv:1206.6460*, 2012.

[77] Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. Structured prediction via output space search. *Journal of Machine Learning Research*, 15:1–34, 2014.

[78] Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. Hc-search: Learning heuristics and cost functions for structured prediction. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[79] Michael Lam, Janardhan Rao Doppa, Xu Hu, Sinisa Todorovic, Thomas Dietterich, Abigail Reft, and Marymegan Daly. Learning to detect basal tubules of nematocysts in sem images. *International Conference on Computer Vision (ICCV)*, 2013.

[80] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2014. URL `http://www.gurobi.com`.

[81] David Weiss, Benjamin Sapp, and Ben Taskar. Structured prediction cascades. *arXiv preprint arXiv:1208.3279*, 2012.

[82] Michael Jones and Paul Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3:14, 2003.

[83] Adwait Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. *IRCS Technical Reports Series*, page 81, 1997.

[84] Xavier Carreras, Michael Collins, and Terry Koo. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16. Association for Computational Linguistics, 2008.

[85] Alex J Smola, SVN Vishwanathan, and Quoc V Le. Bundle methods for machine learning. In *Advances in Neural Information Processing Systems (NIPS) 21*, pages 1377–1384, 2007.

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

[87] Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[88] Peter L Bartlett, Michael Collins, Benjamin Taskar, and David A McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2004.

[89] Amir Globerson, Terry Y Koo, Xavier Carreras, and Michael Collins. Exponentiated gradient algorithms for log-linear structured prediction. In *Proceedings of the 24th international conference on Machine learning*, pages 305–312. ACM, 2007.

[90] Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L Bartlett. Exponentiated gradient algorithms for conditional random fields and

max-margin markov networks. *The Journal of Machine Learning Research*, 9: 1775–1822, 2008.

[91] Pascal Fua, Yunpeng Li, Aurélien Lucchi, et al. Learning for structured prediction using approximate subgradient descent with working sets. In *Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-185082, 2013.

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

[93] Blaine Nelson. *Behavior of Machine Learning Algorithms in Adversarial Environments*. PhD thesis, Electrical Engineering and Computer Sciences University of California at Berkeley, California, United States, 2010.

[94] Axel Dreves, Francisco Facchinei, Christian Kanzow, and Simone Sagratella. On the solution of the kkt conditions of generalized nash equilibrium problems. *SIAM Journal on Optimization*, 21(3):1082–1108, 2011.

[95] Michael Brückner, Christian Kanzow, and Tobias Scheffer. Static prediction games for adversarial learning problems. *the Journal of Machine Learning Research*, 13(1):2617–2654, 2012.

[96] Lemonia Dritsoula, Patrick Loiseau, and John Musacchio. 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, 2012.

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

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

[99] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1011.0686*, 2010.

[100] Ofer Dekel, Ohad Shamir, and Lin Xiao. Learning to classify with missing and corrupted features. *Machine learning*, 81(2):149–178, 2010.

[101] John A Nelder and Roger Mead. A simplex method for function minimization. *Computer journal*, 7(4):308–313, 1965.

[102] Roi Livni and Amir Globerson. A simple geometric interpretation of svm using stochastic adversaries. 2013.

[103] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[104] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *The Journal of Machine Learning Research*, 10:2137–2155, 2009.

[105] Thanh H Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe. Analyzing the effectiveness of adversary modeling in security games. In *Conf. on Artificial Intelligence (AAAI)*, 2013.

[106] Huan Xu, Constantine Caramanis, and Shie Mannor. Robust regression and lasso. *IEEE Transactions on Information Theory*, 56(7):3561–3574, 2010.

[107] Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

[108] Nicolo Cesa-Bianchi. *Prediction, learning, and games.* Cambridge University Press, 2006.

[109] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[110] Elad Hazan. 10 the convex optimization approach to regret minimization. *Optimization for machine learning*, page 287, 2012.

[111] Eiji Takimoto and Manfred K Warmuth. Path kernels and multiplicative updates. *The Journal of Machine Learning Research*, 4:773–818, 2003.

[112] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[113] Ben London, Bert Huang, Ben Taskar, and Lise Getoor. Collective stability in structured prediction: Generalization from one example. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 828–836, 2013.

[114] Michael Kearns and Umesh Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, Cambridge, MA, 1994.

[115] Andreas Schölkopf, P Simard, V Vapnik, and AJ Smola. Improving the accuracy and speed of support vector machines. *Advances in neural information processing systems*, 9:375–381, 1997.

[116] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. *Journal of Machine Learning Research-Proceedings Track*, 20:97–112, 2011.

[117] Laurens Maaten, Minmin Chen, Stephen Tyree, and Kilian Q Weinberger. Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 410–418, 2013.

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

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

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

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

[122] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

[123] Dimitris Bertsimas, Dessislava Pachamanova, and Melvyn Sim. Robust linear optimization under general norms. *Operations Research Letters*, 32(6):510–516, 2004.

[124] Gert RG Lanckriet, Laurent El Ghaoui, Chiranjib Bhattacharyya, and Michael I Jordan. A robust minimax approach to classification. *The Journal of Machine Learning Research*, 3:555–582, 2003.

[125] Laurent El Ghaoui, Gert R.G. Lanckriet, and Georges Natsoulis. *Robust classification with interval data.* Computer Science Division, University of California, 2003.

[126] Chiranjib Bhattacharyya, KS Pannagadatta, and Alexander J Smola. A second order cone programming formulation for classifying missing data. *Advances in neural information processing systems*, 17:153–160, 2004.

[127] Pannagadatta K Shivaswamy, Chiranjib Bhattacharyya, and Alexander J Smola. Second order cone programming approaches for handling missing and uncertain data. *The Journal of Machine Learning Research*, 7:1283–1314, 2006.

[128] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. 2013.

[129] Battista Biggio, Ignazio Pillai, Samuel Rota Bulò, Davide Ariu, Marcello Pelillo, and Fabio Roli. Is data clustering in adversarial settings secure? In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, pages 87–98. ACM, 2013.

[130] Battista Biggio, Igino Corona, Blaine Nelson, Benjamin IP Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, et al. Security evaluation of support vector machines in adversarial environments. *arXiv preprint arXiv:1401.7727*, 2014.

[131] Arindam Banerjee. An analysis of logistic models: Exponential family connections and online performance. In *SDM*. SIAM, 2007.

[132] Keith Hall, Ryan McDonald, Jason Katz-Brown, and Michael Ringgaard. Training dependency parsers by jointly optimizing multiple objectives. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1489–1499. Association for Computational Linguistics, 2011.

[133] David Jensen and Jennifer Neville. 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, 2002. Morgan Kaufmann.

[134] Qing Lu and Lise Getoor. Link-based classification. In *ICML*, volume 3, pages 496–503, 2003.

[135] Yonghong Tian, Tiejun Huang, and Wen Gao. Robust collective classification with contextual dependency network models. In *Advanced Data Mining and Applications*, pages 173–180. Springer, 2006.

[136] J. Neville and D. Jensen. Relational dependency networks. *The Journal of Machine Learning Research*, 8:653–692, 2007. ISSN 1532-4435.

[137] Daniel Lowd and Arash Shamaei. Mean field inference in dependency networks: An empirical study. In *AAAI*, 2011.

[138] Luke K McDowell, Kalyan Moy Gupta, and David W Aha. Cautious collective classification. *The Journal of Machine Learning Research*, 10:2777–2836, 2009.

[139] Mustafa Bilgic, Galileo Mark Namata, and Lise Getoor. Combining collective classification and link prediction. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 381–386. IEEE, 2007.

[140] Kapil Kumar Gupta, Baikunth Nath, and Kotagiri Ramamohanarao. 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, 2007.

[141] Kapil Kumar Gupta, Baikunth Nath, and Ramamohanarao Kotagiri. Layered approach using conditional random fields for intrusion detection. *Dependable and Secure Computing, IEEE Transactions on*, 7(1):35–49, 2010.

[142] Yan Qiao, XW Xin, Yang Bin, and S Ge. Anomaly intrusion detection method based on hmm. *Electronics Letters*, 38(13):663–664, 2002.

[143] Yale Song, Zhen Wen, Ching-Yung Lin, and Randall Davis. 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, 2013.

[144] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. 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, 2008.

[145] Jason Tsai, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Shyamsunder Rathi. Iris-a tool for strategic security allocation in transportation networks. 2009.

[146] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordóñez, and Milind Tambe. Security games with arbitrary schedules: A branch and price approach. In *AAAI*, 2010.

[147] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. 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, 2012.

[148] Fei Fang, Albert Xin Jiang, and Milind Tambe. 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, 2013.

[149] James Pita, Milind Tambe, Christopher Kiekintveld, Shane Cullen, and Erin Steigerwald. 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, 2011.

[150] Z. Yin, M. Jain, M. Tambe, and F. Ordóñez. Risk-averse strategies for security games with execution and observational uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[151] Zhengyu Yin, Albert Xin Jiang, Matthew Paul Johnson, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, Milind Tambe, and John P Sullivan. Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*, 2012.

[152] Albert Xin Jiang, Zhengyu Yin, Chao Zhang, Milind Tambe, and Sarit Kraus. 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, 2013.

[153] Albert Xin Jiang, Thanh H Nguyen, Milind Tambe, and Ariel D Procaccia. Monotonic maximin: A robust stackelberg solution against boundedly rational followers. In *Decision and Game Theory for Security*, pages 119–139. Springer, 2013.

[154] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. 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, 2009.

[155] Bo An, David Kempe, Christopher Kiekintveld, Eric Shieh, Satinder Singh, Milind Tambe, and Yevgeniy Vorobeychik. Security games with limited surveillance. *Ann Arbor*, 1001:48109, 2012.

[156] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. 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, 2011.

[157] John P Dickerson, Gerardo I Simari, VS Subrahmanian, and Sarit Kraus. 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, 2010.

[158] Dian Gong, Xuemei Zhao, and Gérard Medioni. Robust multiple manifolds structure learning. *ICML*, 2012.

[159] Mani Ranjbar, Tian Lan, Yang Wang, Stephen N Robinovitch, Ze-Nian Li, and Greg Mori. Optimizing nondecomposable loss functions in structured prediction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(4):911–924, 2013.

[160] Sheng Chen, Zhongyuan Feng, Qingkai Lu, Behrooz Mahasseni, Trevor Fiez, Alan Fern, and Sinisa Todorovic. Play type recognition in real-world football video. *WACV*, 2014.

[161] Shi-Xiong Zhang, Anton Ragni, and Mark John Francis Gales. Structured log linear models for noise robust speech recognition. *Signal Processing Letters, IEEE*, 17(11):945–948, 2010.

[162] Shi-Xiong Zhang, Mark JF Gales, et al. Structured support vector machines for noise robust continuous speech recognition. In *INTERSPEECH*, pages 989–990, 2011.