The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

ARC: Adversarial Robust Cuts for Semi-Supervised and Multi-Label Classification

Sima Behpour, Wei Xing, Brian D. Ziebart
Department of Computer Science
University Of Illinois at Chicago
{sbehpo2,wxing3,bziebart}@uic.edu

Abstract

Many structured prediction tasks arising in computer vision and natural language processing tractably reduce to making minimum cost cuts in graphs with edge weights learned using maximum margin methods. Unfortunately, the hinge loss used to construct these methods often provides a particularly loose bound on the loss function of interest (e.g., the Hamming loss). We develop Adversarial Robust Cuts (ARC), an approach that poses the learning task as a minimax game between predictor and "label approximator" based on minimum cost graph cuts. Unlike maximum margin methods, this game-theoretic perspective always provides meaningful bounds on the Hamming loss. We conduct multi-label and semi-supervised binary prediction experiments that demonstrate the benefits of our approach.

Introduction

Structured prediction—the task of predicting multiple interrelated variables—is increasingly important for machine learning applications in computer vision (Kolmogorov and Zabin 2004), natural language processing (Flake, Tarjan, and Tsioutsiouliklis 2004), computational biology, and other areas (Blum and Chawla 2001; Blum et al. 2004). Though intractable in general (even when only pairwise relationships exist (Wainwright and Jordan 2008)), certain restricted relationships and structures (e.g., chains, trees, and other low-treewidth structures) do have efficient inference algorithms. The most general of these—binary-valued associative Markov networks (Taskar, Chatalbashev, and Koller 2004) and the special case of attractive pairwise relationships (Boykov, Veksler, and Zabih 2001)—use minimum graph cuts (Greig, Porteous, and Seheult 1989) for inference and maximum margin methods (Tsochantaridis et al. 2004; Joachims 2005) for training. Unfortunately, the hinge loss surrogate employed by this approach can be quite loose, often providing meaningless performance guarantees in practice. For example, the surrogate loss may be worse than random predictions or even the worst possible loss.

Seeking to tighten this gap between training objective and evaluative loss function for structured prediction tasks, we present Adversarial Robust Cuts (ARC), an approach to learning for binary associative Markov networks and attractive pairwise relationships. ARC takes the form of a zero-sum game between a predictor trying to minimize an additive loss over predicted variables and an adversarial training label approximator that seeks to maximize this same loss. It provides loss bounds that are always meaningful since the game outcome is always within the range of the evaluation loss function. Using the double oracle method (McMahan, Gordon, and Blum 2003), which is a constraint generation method for zero-sum games, inference reduces to solving a sequence of minimum-cut/linear programming problems. Learning corresponds to a convex optimization problem that we address using standard gradient-based methods. We demonstrate the benefits of our approach on multi-label and semi-supervised binary classification tasks. In the latter tasks, our approach combines inductive parametric modeling with similarity-based reasoning.

Adversarial learning methods have seen a recent resurgence, particularly for deep generative models (Goodfellow et al. 2014). Early methods sought to make predictors robust to manipulations of the training data (Dalvi et al. 2004). Our work follows a line of research with a less powerful adversary: the inherent uncertainty of true distributions in inductive learning settings given limited training data (Topsøe 1979; Grünwald and Dawid 2004). We contribute an extension of structured prediction using this minimax perspective beyond the previous work on classification problems with zero-one loss (Fathony et al. 2016), ordinal regression (Fathony, Bashiri, and Ziebart 2017), more general cost-sensitive losses (Asif et al. 2015), multivariate losses (Wang et al. 2015), and chain structures (Li et al. 2016).

Background

Notation and Learning Task

We consider n predicted variables, $\mathbf{y} = (y_1, \ldots, y_n)$, chosen from a fixed set of labels $y_i \in \mathcal{Y}, \forall i \in [n]$, where $[n] = \{1, \ldots, n\}$. We denote the corresponding random variables for these label variables using capitalization, $\mathbf{Y} = (Y_1, \ldots, Y_n)$, and denote vectors and multivariate variables in bold. We denote given information or side information variables using a single vector, $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, with a corresponding random variable denoted as $\mathbf{X}$. (Strict sub-portions of $\mathbf{x}$ may be relevant to each variable y_i, but for notational sim-

$$\mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

Figure 1: An example image and its multilabel annotation vector for label set: *sky*, *clouds*, *trees*, *sunset*, *sea*, *ship*, *mountains*, *desert*, . . .

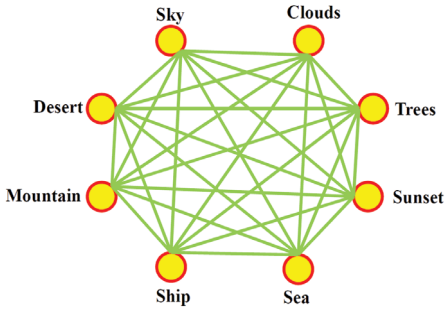

Figure 2: The Markov network corresponding to the multilabel annotation prediction of Figure 1.

plicity, we do not denote such partitions in our formulation.)

Our task in this setting is to make predictions for $\mathbf{y}$ given an input $\mathbf{x}$ and a set of m training example pairs, $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})_{j\in[m]}$, where we index training examples using a parenthetical superscript notation whenever necessary to disambiguate between different examples or denote this distribution as $\tilde{P}(\mathbf{X}, \mathbf{Y})$. Aiding in this task are a set of features relating the input variables to the predicted variables and to one another. We generically denote these feature vectors as $\boldsymbol{\phi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}, \mathbf{x})$ for relationships over variables in some subset of the $\mathbf{y}$ variables denoted by $\mathbf{c} \in \mathcal{C} \subseteq 2^{[n]}$. For a subset $\mathbf{c} = \{c_1, \ldots, c_l\}$ which contains l variables, $\mathbf{y}_{\mathbf{c}} = \{y_{c_1}, \ldots, y_{c_l}\}$ is the corresponding set of label values for the variables in the subset. For pairwise relationships between y_i and y_j that also incorporate input variables, this reduces to feature functions denoted as $\boldsymbol{\phi}_{i,j}(y_i, y_j, \mathbf{x})$ and to $\boldsymbol{\phi}_i(y_i, \mathbf{x})$ for univariate feature functions.

For many datasets, variables that are closely related to one another tend to have the same label. For example, pixels with similar characteristics in the same region of an image tend to belong to the same image segment. To capture this property, we define pairwise features that reflect the difference when two variables have different labels, and use a generalized Potts model (Potts 1952) to penalize assignments that do not have the same label across the edges:

$$\boldsymbol{\phi}_{i,j}(y_i, y_j, \mathbf{x}) = I(y_i \neq y_j)\boldsymbol{\delta}_{i,j}(y_i, y_j, \mathbf{x}), \tag{1}$$

where $I()$ is a indicator function whose value is 1 only if the inner logical expression is true.

We consider the multilabel prediction task of annotating images as a running example. Each training image, $\mathbf{x}$, has an associated vector of labels, $\mathbf{y}$, corresponding to different descriptors of the image, as illustrated in Figure 1. We define unary and pairwise features for the labels as:

$$\boldsymbol{\phi}_{sky}(y_{sky}, \text{  }) = I(y_{sky}=1)\,\mathbf{imgFeatures}(\text{  }) \tag{2}$$

$$\boldsymbol{\phi}_{sky,clouds}(y_{sky}, y_{clouds}) = I(y_{sky} \neq y_{clouds}) \tag{3}$$
$$|\,\mathbf{word2vec}(sky) - \mathbf{word2vec}(clouds)|^{-1},$$

using features from the Mulan dataset (Tsoumakas et al. 2011) for image representations and a deeply learned word embedding[1] for word semantics.[2]

In this paper, we focus on problems evaluated using the Hamming loss:

$$\text{loss}(\hat{\mathbf{y}}, \check{\mathbf{y}}) = \frac{1}{n}\sum_i I(\hat{y}_i \neq \check{y}_i), \tag{4}$$

which measures the fraction of the labels that are correctly predicted in the multilabel annotation task.

Markov Networks and Intractability

Estimating the conditional probability of label variables using a Markov network is one powerful approach to this structured prediction task. Markov networks can be written as log-linear models when their densities are positive. A Markov network has the following probability distribution:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}e^{\boldsymbol{\Psi}(\mathbf{y},\mathbf{x})}, \tag{5}$$

where the potential function $\boldsymbol{\Psi}$ decomposes into a set of potential functions over subsets of the $\mathbf{y}$ variables, $\boldsymbol{\Psi}(\mathbf{y}, \mathbf{x}) = \sum_{\mathbf{c}\in\mathcal{C}} \boldsymbol{\psi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}, \mathbf{x})$, with these subset potentials defined as $\boldsymbol{\psi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}, \mathbf{x}) = \theta_{\mathbf{c}} \cdot \boldsymbol{\phi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}, \mathbf{x})$ using a vector of estimated weights $\theta_{\mathbf{c}}$ that is specific to each $\mathbf{c}$. Parameter sharing with clique $\mathbf{c}'$, $\boldsymbol{\phi}_{\mathbf{c}} = \boldsymbol{\phi}_{\mathbf{c}'}$, can be employed to reduce the effective number of learned parameters of the model. The structure of these potentials corresponds to an undirected graphical model in which the variables in set $\mathbf{c}$ are connected by undirected edges, forming cliques in the graph.

In our running example, all unary and pairwise subsets of variables are inlcuded in $\mathcal{C} = \{\{sky\}, \{clouds\}, \{trees\}, \ldots, \{sky, clouds\}, \{sky, trees\}\}$, and the corresponding Markov network is the complete graph over all of these class labels (Figure 2). Unfortunately, even when restricted to pairwise and unary potential functions, the most probable assignment of values, $\mathbf{y}^* = \text{argmax}_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x})$, and the normalization term, $Z(\mathbf{x}) = \sum_{\mathbf{y}\in\mathcal{Y}} e^{\sum_{\mathbf{c}\in\mathcal{C}} \boldsymbol{\psi}_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}},\mathbf{x})}$, are both intractable to compute for Markov networks in general (Wainwright and

[1] https://code.google.com/p/word2vec/

[2] We use element-wise operations to compute and invert the differences between each embedded dimension.

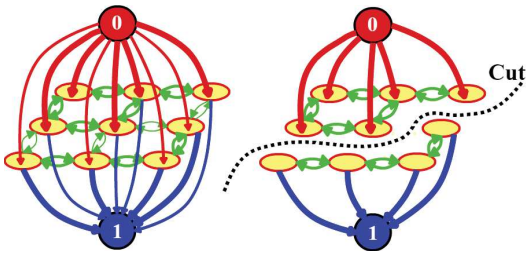

Figure 3: A directed graph used to augment a Markov network (left) so that the minimum cut (right) provides the most probable assignment of each variable based on its connection to the source node (0) or target node (1).

Jordan 2008). Restrictions are often placed on the potential functions so that the corresponding undirected graph has low tree-width (e.g., chains, trees), which enables efficient maximization and normalization computations (Wainwright and Jordan 2008).

Minimum-Cuts and Associative Markov Networks

Another direction for realizing tractable Markov networks exploits potential functions for which maximization can be solved efficiently, even though normalization is intractable due to the large tree-widths of their graphs. Binary-valued Markov networks with non-negative pairwise potentials are one example of this. Their maximum value assignments can be obtained using minimum-cut/maximum-flow algorithms, as shown in Figure 3. Edges from the source and to the sink nodes are weighted based on unary feature potentials, and edges between predicted variables are weighted based on the pairwise feature potentials. Large potentials prevent certain edge cuts (and corresponding value assignments to the connected sink or source) from the solution. In our running example, for instance, two semantically related words (e.g., *sky* and *clouds*) are likely to have large learned potentials that prevent one from being an included label without the other. This class of models has been employed extensively in computer vision applications for binary image denoising and segmentations problems (Greig, Porteous, and Seheult 1989; Boykov, Veksler, and Zabih 2001).

Associative Markov networks (AMNs) (Taskar, Chatalbashev, and Koller 2004) allow larger cliques with potential functions that are only non-zero if all variables in the clique share the same value. This has the effect of "attracting" the same values across the predicted variables. These generalizations of binary structured prediction beyond pairwise potentials to arbitrary cliques can be solved using a linear program (Taskar et al. 2005) that is closely related to the minimum-cut problem. Though multiclass inference problems are generally much more difficult than binary problems, there are also some classes of potential functions with certain "convexity" properties (Ishikawa 2003) that can be solved efficiently as min-cut problems.

Maximum Margin Learning and Loose Bounds

Maximum margin methods for learning, like the structured support vector machine (Tsochantaridis et al. 2004; Taskar et al. 2005), operate by introducing a hinge loss surrogate to optimize in place of the loss function of interest, which is generally non-convex and possibly not even continuous. For structured prediction, this takes the following form:

$$\min_{\boldsymbol{\theta},\boldsymbol{\epsilon}\geq\mathbf{0}} ||\boldsymbol{\theta}|| + \lambda \sum_i \epsilon_i \text{ such that:}$$
$$\epsilon_i \geq \max_{\mathbf{y}'} \text{loss}(\mathbf{y}', \mathbf{y}^{(i)}) + \boldsymbol{\Psi}(\mathbf{y}', \mathbf{x}) - \boldsymbol{\Psi}(\mathbf{y}^{(i)}, \mathbf{x}), \quad (6)$$

where loss() can be any loss function of interest, $\boldsymbol{\theta}$ is a compact representation of all estimated potential function parameters, λ is a fixed regularization parameter, and ϵ_i is the amount of hinge loss incurred by the i^{th} training example. Conceptually, this optimization seeks to make the potential of the true label vector $\boldsymbol{\Psi}(\mathbf{y}^{(i)}, \mathbf{x})$ greater than all alternatives $\boldsymbol{\Psi}(\mathbf{y}', \mathbf{x})$ by a margin that depends on the (Hamming) loss between $\mathbf{y}'$ and $\mathbf{y}^{(i)}$. Since the objective of this optimization is a convex function of the model parameters, $\boldsymbol{\theta}$, standard gradient-based methods can be employed to optimize $\boldsymbol{\theta}$. Achieving low ϵ_i ensures small amounts of incurred loss on example i, since $\epsilon_i \geq \text{loss}(\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} \boldsymbol{\Psi}(\mathbf{y}), \mathbf{y}^{(i)})$.

Finding parameters $\boldsymbol{\theta}$ that make the total hinge loss, $\sum_i \epsilon_i$, small can be difficult. When the size of the label space $|\boldsymbol{\mathcal{Y}}|$ is much larger than the number of parameters being learned, there may not be any non-trivial choice of $\boldsymbol{\theta} \neq \mathbf{0}$ that makes the potential of $\mathbf{y}^{(i)}$ optimal. When multiple training examples are considered, reducing the hinge loss for one example tends to increase the hinge losses for other examples. Due to these issues, the hinge loss for a particular example $\mathbf{y}^{(i)}$ can be much greater than not only the actual loss of the predicted labels, $\text{loss}(\hat{\mathbf{y}}, \mathbf{y}^{(i)})$, but random guessing and the worst possible loss, $\max_{\mathbf{y}} \text{loss}(\mathbf{y}, \mathbf{y}^{(i)})$, as well. When this is the case, the hinge loss bounds provide no meaningful guarantees on the predictor's performance.

Adversarial Robust Cuts (ARC)

The motivating idea of minimax robust learning is to approximate the training data labels with a worst-case distribution that must still resemble training data properties (Topsøe 1979; Grünwald and Dawid 2004; Asif et al. 2015). The advantage of this approach versus employing the hinge loss or other convex surrogate is that: (1) the training error on the actual loss function of interest (e.g., the Hamming loss) is upper bounded by the game value; and (2) optimizing this upper bound more closely aligns the predictor's construction with its predictive performance.

Minimax Game Formulation

We introduce the distribution $\check{P}(\check{\mathbf{Y}}|\mathbf{X})$ controlled by the adversary to produce worst-case approximations of the actual training example label, $\mathbf{y}^{(i)}$. The adversary seeks to maximize the loss subject to certain constraints based on

the training data sample, while the predictor player chooses $\hat{P}(\hat{\mathbf{Y}}|\mathbf{X})$ to minimize the expected loss:

$$\min_{\hat{P}(\hat{\mathbf{Y}}|\mathbf{X})} \max_{\check{P}(\check{\mathbf{Y}}|\mathbf{X})} \mathbb{E}_{\mathbf{X}\sim\tilde{P};\hat{\mathbf{Y}}|\mathbf{X}\sim\hat{P};\check{\mathbf{Y}}|\mathbf{X}\sim\check{P}} \left[\text{loss}(\hat{\mathbf{Y}},\check{\mathbf{Y}})\right] \text{ such that:}$$
$$\mathbb{E}_{\mathbf{X}\sim\tilde{P};\check{\mathbf{Y}}|\mathbf{X}\sim\check{P}} \left[\boldsymbol{\phi}_i(\check{Y}_i,\mathbf{X})\right] = \mathbb{E}_{\mathbf{X},\mathbf{Y}\sim\tilde{P}} \left[\boldsymbol{\phi}_i(Y_i,\mathbf{X})\right], \forall i \in [n];$$
$$\mathbb{E}_{\mathbf{X}\sim\tilde{P};\check{\mathbf{Y}}|\mathbf{X}\sim\check{P}} \left[\boldsymbol{\phi}_{i,j}(\check{Y}_i,\check{Y}_j,\mathbf{X})\right] \leq \mathbb{E}_{\mathbf{X},\mathbf{Y}\sim\tilde{P}} \left[\boldsymbol{\phi}_{i,j}(Y_i,Y_j,\mathbf{X})\right], \quad \forall i,j \in [n], \tag{7}$$

where $\tilde{P}$ is the empirical distribution of $\mathbf{X}$ [and $\mathbf{Y}$] in the training data set.

Conceptually, the adversary seeks to construct a label distribution that is as uncertain as possible, since this forces the predictor to incur large amounts of expected loss. However, the constraints placed on the adversary to match statistics ϕ relating the actual training data labels to inputs $\mathbf{x}$ and one another (such as mean or higher order moments) prevent the adversary from doing this, and, when chosen carefully, can restrict the adversary to be highly predictable.

We reduce and generalize the restrictions on the adversary by redefining the constraints so that they share clique features. For example, rather than having unique pairwise constraints for each $i \neq j$, we can have a single constraint:

$$\mathbb{E}_{\mathbf{X}\sim\tilde{P};\check{\mathbf{Y}}|\mathbf{X}\sim\check{P}} \left[\sum_{i\neq j} \boldsymbol{\phi}_{i,j}(\check{Y}_i,\check{Y}_j,\mathbf{X})\right] \leq \mathbb{E}_{\mathbf{X},\mathbf{Y}\sim\tilde{P}} \left[\sum_{i\neq j} \boldsymbol{\phi}_{i,j}(Y_i,Y_j,\mathbf{X})\right]. \tag{8}$$

Slack for the constraints can also be incorporated to deal with the variance that results from having small amounts of training data, leading to regularization in the resulting optimization problems (Dudík and Schapire 2006).

Using the method of Lagrange multipliers and strong duality (Boyd and Vandenberghe 2004), the constraints of this formulation can be incorporated as Lagrangian potentials with parameters $\{\boldsymbol{\theta}_c\}$:

$$\min_{\{\boldsymbol{\theta}_i\},\{\boldsymbol{\theta}_{i,j}\}\leq\mathbf{0}} \min_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \max_{\check{P}(\check{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\substack{\mathbf{X},\mathbf{Y}\sim\tilde{P},\\ \hat{\mathbf{Y}}|\mathbf{X}\sim\hat{P},\\ \check{\mathbf{Y}}|\mathbf{X}\sim\check{P}}} \Big[\text{loss}(\hat{\mathbf{Y}},\check{\mathbf{Y}}) + \sum_i \boldsymbol{\theta}_i \cdot (\boldsymbol{\phi}_i(\check{Y}_i,\mathbf{X}) - \boldsymbol{\phi}_i(Y,\mathbf{X})) + \sum_{i\neq j} \boldsymbol{\theta}_{i,j} \cdot (\boldsymbol{\phi}_{i,j}(\check{Y}_i,\check{Y}_j,\mathbf{X}) - \boldsymbol{\phi}_{i,j}(Y_i,Y_j,\mathbf{X}))\Big]. \tag{9}$$

These Lagrangian potentials exactly match those of the Markov random field (Equation (5)) in form, and can similarly be written as $\Psi(\check{\mathbf{y}},\mathbf{x}) = \sum_i \psi_i(y_i,\mathbf{x}) + \sum_{i\neq j} \psi_{i,j}(y_i,y_j,\mathbf{x})$, where $\psi_i(y_i,\mathbf{x}) = \theta_i \cdot \phi_i(y_i,\mathbf{x})$ and $\psi_{i,j}(y_i,y_j,\mathbf{x}) = \theta_{i,j} \cdot \phi_i(y_i,y_j,\mathbf{x})$.

Under the Hamming loss (Equation (4)), the terms of this optimization problem in Equation (9) involving $\hat{P}$ and $\check{P}$ for a specific input $\mathbf{x}$ can be re-written as a bilinear function of the predictor's distribution, the Lagrangian-augmented loss function, and the adversary's distribution,

$$\underbrace{\begin{bmatrix} \hat{P}(000) \\ \hat{P}(001) \\ \hat{P}(010) \\ \vdots \end{bmatrix}^{\mathrm{T}}}_{\hat{\mathbf{p}}_{\mathbf{x}}} \underbrace{\begin{bmatrix} 0+\Psi(000,\mathbf{x}) & \frac{1}{3}+\Psi(001,\mathbf{x}) & \cdots \\ \frac{1}{3}+\Psi(000,\mathbf{x}) & 0+\Psi(001,\mathbf{x}) & \cdots \\ \frac{1}{3}+\Psi(000,\mathbf{x}) & \frac{2}{3}+\Psi(001,\mathbf{x}) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}}_{\mathbf{C}_{\boldsymbol{\theta},\mathbf{x}}=\{\text{loss}(\hat{\mathbf{y}},\check{\mathbf{y}})+\Psi(\check{\mathbf{y}},\mathbf{x})\}} \underbrace{\begin{bmatrix} \check{P}(000) \\ \check{P}(001) \\ \check{P}(010) \\ \vdots \end{bmatrix}}_{\check{\mathbf{p}}_{\mathbf{x}}},$$

which can then be compactly denoted in terms of vectors of conditional probabilities ($\hat{\mathbf{p}}_{\mathbf{x}}$ and $\check{\mathbf{p}}_{\mathbf{x}}$), and a payoff matrix ($\mathbf{C}_{\boldsymbol{\theta},\mathbf{x}}$) for each example $\mathbf{x}$.

With the Lagrangian potentials incorporated into the payoff matrix $\mathbf{C}_{\boldsymbol{\theta},\mathbf{x}}$, the joint game over all training instances can be solved independently for each one:

$$\min_{\{\boldsymbol{\theta}_i\},\{\boldsymbol{\theta}_{i,j}\}\leq 0} \mathbb{E}_{\mathbf{X},\mathbf{Y}\sim\tilde{P}} \left[\left(\max_{\check{\mathbf{p}}_{\mathbf{X}}} \min_{\hat{\mathbf{p}}_{\mathbf{X}}} \hat{\mathbf{p}}_{\mathbf{X}}^{\mathrm{T}}\, \mathbf{C}_{\boldsymbol{\theta}_1,\mathbf{X}}\, \check{\mathbf{p}}_{\mathbf{X}}\right) - \sum_i \boldsymbol{\theta}_i \cdot \boldsymbol{\phi}_i(Y_i,\mathbf{X}) - \sum_{i\neq j} \boldsymbol{\theta}_{i,j} \cdot \boldsymbol{\phi}_{i,j}(Y_i,Y_j,\mathbf{X})\right]. \tag{10}$$

Unfortunately, naïvely constructing the game in this manner requires time and space that grow exponentially with the number of predicted variables to include every possible vector of values. We address this computational bottleneck using the constraint generation methods in a later sections.

Double Oracle for ARC

Unlike traditional probabilistic treatments of Markov random fields, which require computing the normalizing partition function, $Z(\mathbf{x})$, our approach obtains an equilibrium over value assignments for the set of predicted variables between the predictor and the adversary. Conceptually, this game is played over the set of all possible label assignments as strategies. However, in practice the game equilibria have sparse support (i.e., $P(\mathbf{y}|\mathbf{x}) = 0$ for most $\mathbf{y}$). We employ the double oracle algorithm (McMahan, Gordon, and Blum 2003), a constraint generation approach, to uncover a set of strategies for each player that support the equilibrium.

Algorithm 1 outlines the behavior of this procedure. It operates by maintaining a set of label vectors $\hat{\mathcal{S}}$ and $\check{\mathcal{S}}$ for each player that it greedily expands until the game's equilibrium is supported by the label vectors of these sets. This is accomplished by repeatedly finding the game's equilibrium for the current set of strategies and then alternatively adding each player's best response against the other player's equilibrium distribution as a new row or column for the game. Within the algorithm, we denote the potentials for the label vectors of $\check{\mathcal{S}}$ as $\boldsymbol{\Psi}(\check{\mathcal{S}}) \triangleq \{\Psi(\check{\mathbf{y}},\mathbf{x})\}_{\check{\mathbf{y}}\in\check{\mathcal{S}}}$ and denote the loss matrix between all pairs of label vectors across sets as $\text{loss}_{\text{Ham}}(\hat{\mathcal{S}},\check{\mathcal{S}}) \triangleq \{\text{HammingLoss}(\check{\mathbf{y}},\hat{\mathbf{y}})\}_{\check{\mathbf{y}}\in\check{\mathcal{S}},\hat{\mathbf{y}}\in\hat{\mathcal{S}}}$.

We solve the zero-sum games (solveGame) as linear programs (von Neumann and Morgenstern 1947), which can be solved in polynomial time. The key remaining challenge is finding the best response for each player.

Algorithm 1 Double Oracle Algorithm for ARC Equilibria

Require: Node features $\{\phi_i(\cdot, \mathbf{x})\}$; Pairwise features $\{\phi_{i,j}(\cdot, \cdot, \mathbf{x})\}$; Parameters $\boldsymbol{\theta}$; Initial label $\mathbf{y}_{\text{initial}}$
Ensure: Nash equilibrium, $(\hat{P}, \check{P})$

$\check{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \leftarrow \{\mathbf{y}_{\text{initial}}\}$
repeat
  $(\hat{P}, \check{P}, \check{V}) \leftarrow \text{solveGame}(\boldsymbol{\Psi}(\check{\mathcal{S}}), \text{loss}_{\text{Ham}}(\check{\mathcal{S}}, \hat{\mathcal{S}}))$
  $(\check{\mathbf{y}}_{new}, V_{\max}) \leftarrow \max_{\check{\mathbf{y}}} \mathbb{E}_{\hat{\mathbf{Y}} \sim \hat{P}}[\text{loss}_{\text{Ham}}(\check{\mathbf{y}}, \hat{\mathbf{Y}}) + \Psi(\check{\mathbf{y}})]$
  if $(\check{V} \neq V_{\max})$ **then**
    $\check{\mathcal{S}} \leftarrow \check{\mathcal{S}} \cup \check{\mathbf{y}}_{new}$
  end if
  $(\hat{P}, \check{P}, \hat{V}) \leftarrow \text{solveGame}(\boldsymbol{\Psi}(\check{\mathcal{S}}), \text{loss}_{\text{Ham}}(\check{\mathcal{S}}, \hat{\mathcal{S}}))$
  $(\hat{\mathbf{y}}_{new}, V_{\min}) \leftarrow \min_{\hat{\mathbf{y}}} \mathbb{E}_{\check{\mathbf{Y}} \sim \check{P}}[\text{loss}_{\text{Ham}}(\check{\mathbf{Y}}, \hat{\mathbf{y}})]$
  if $(\hat{V} \neq V_{\min})$ **then**
    $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \cup \hat{\mathbf{y}}_{new}$
  end if
until $\check{V} = V_{\max} = \hat{V} = V_{\min}$
return $(\hat{P}, \check{P})$

Adversary's best responses: Given the predictor's distribution over value vectors, $\hat{P}(\hat{\mathbf{y}}|\mathbf{x})$, over $\hat{\mathbf{y}} \in \hat{\mathcal{S}}$ the adversary's best response is a single vector of values $\check{\mathbf{y}}_{\text{new}}$ with the largest possible expected value under $\hat{P}$. It is determined both by the expected loss against the predictor's distribution and the Lagrangian potential terms:

$$\operatorname*{argmax}_{\check{\mathbf{y}}} \mathbb{E}_{\hat{\mathbf{Y}}|\mathbf{x} \sim \hat{P}} \left[\frac{1}{n} \sum_i I(\hat{Y}_i \neq \check{y}_i) \right] + \sum_i \boldsymbol{\psi}_i(\check{y}_i) + \sum_{i \neq j} \boldsymbol{\psi}_{i,j}(\check{y}_i, \check{y}_j, \mathbf{x}). \quad (11)$$

This is similar to loss-augmented potential maximization problems addressed in maximum margin methods (Taskar et al. 2005). When there are only binary labels, the problem can be transformed into a min-cut problem by expanding the expectation as:

$$\operatorname*{argmin}_{\check{y}} \sum_i - \left(\frac{1}{n} \sum_{\hat{y}} \hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \cdot I(\hat{y}_i \neq \check{y}_i) + \boldsymbol{\psi}_i(\check{y}_i) \right) + \sum_{i \neq j} -\boldsymbol{\psi}_{i,j}(\check{y}_i, \check{y}_j, \mathbf{x}). \quad (12)$$

If we use values 0 and 1 as binary labels, we can separate the formula by the values of $\check{\mathbf{y}}$ and apply Eq. (1), obtaining:

$$\operatorname*{argmin}_{\check{y}} \sum_{\{i:\check{y}_i=1\}} \left(-\frac{1}{n} \sum_{\hat{y}} \hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \cdot \left[I(\hat{y}_i = 0) - \boldsymbol{\psi}_i(1, \mathbf{x}) \right] \right) + \sum_{\{i:\check{y}_i=0\}} \left(-\frac{1}{n} \sum_{\hat{y}} \hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \cdot \left[I(\hat{y}_i = 1) - \boldsymbol{\psi}_i(0, \mathbf{x}) \right] \right) + \sum_{i \neq j} I(y_i \neq y_j) - \boldsymbol{\theta}_{i,j} \boldsymbol{\delta}_{i,j}(y_i, y_j, \mathbf{x}). \quad (13)$$

We construct a directed graph G that contains nodes representing each variable, along with two special nodes: source (*Src*) and target (*Trg*). Solving Eq. (13) is equivalent to the min-cut problem on G if the following weights (capacities) are all non-negative: edges from the source node to each predicted variable are weighted by

$$w_{\text{Src},i} = -\frac{1}{n} \sum_{\hat{y}} \hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \cdot I(\hat{y}_i = 1) - \boldsymbol{\psi}_i(0, \mathbf{x}), \quad (14)$$

directed edges from each predicted variable to the target node are similarly weighted by:

$$w_{i,\text{Trg}} = -\frac{1}{n} \sum_{\hat{y}} \hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \cdot I(\hat{y}_i = 0) - \boldsymbol{\psi}_i(1, \mathbf{x}), \quad (15)$$

and each pair of different nodes except source and target are weighted by:

$$w_{i,j} = w_{j,i} = -\boldsymbol{\theta}_{i,j} \cdot \boldsymbol{\delta}_{i,j}(\check{y}_i, \check{y}_j, \mathbf{x}). \quad (16)$$

After removing edges in the cut-set, nodes connected to *Src* have label 0 and the other nodes have label 1.

To make the weights non-negative, we first consider weights for edges connected to either *Src* or *Trg* nodes. One important property is that we can add the same constant to both the capacity from *Src* to node i and the capacity from node i to *Trg* without changing the cut-set. That is because one and only one of the two edges will be in the cut-set. Based on this property, we can simply subtract the smaller of $w_{\text{Src},i}$ and $w_{i,\text{Trg}}$ from each one's pairwise weights. Next, for the pairwise weights, from Eq. (9) we know that $\boldsymbol{\theta}_{i,j} \leq 0$. So as long as we choose non-negative $\boldsymbol{\delta}_{i,j}(y_i, y_j, \mathbf{x})$, $w_{i,j}$ will be non-negative.

Predictor's best response: Given the adversary's distribution of value vectors $\check{P}(\check{\mathbf{y}}|\mathbf{x})$ from $\check{\mathbf{y}} \in \check{\mathcal{S}}$, the predictor's best response is a single vector of values with the smallest expected value against $\check{P}$. Since the Lagrangian potential does not rely on predictor's strategy, and the Hamming loss is additively decomposable, we have:

$$\min_{\hat{\mathbf{y}}} \mathbb{E}_{\check{\mathbf{Y}}|\mathbf{x} \sim \check{P}} \left[\frac{1}{n} \sum_i I(\hat{y}_i \neq \check{Y}_i) \right] = \frac{1}{n} \sum_i \min_{\hat{y}_i} \sum_{\check{\mathbf{y}}} \check{P}(\check{\mathbf{y}}|\mathbf{x}) I(\hat{y}_i \neq \check{y}_i). \quad (17)$$

Thus, the predictor's best responses can be independently made for each variable: $\hat{y}_i = \operatorname{argmax}_y \check{P}(\check{Y}_i = y|\mathbf{x})$.

Using these best response subroutines, the double oracle algorithm can be applied to obtain a compact equilibrium distribution for the robust min-cut game.

Parameter Learning

The final aspect of our approach is optimizing the model parameters $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_{i,j}$, that incentivize the adversary. We accomplish this using standard tools from convex optimization. Specifically, we employ AdaGrad (Duchi, Hazan, and Singer 2011) and compute the gradients $\mathbf{g}_i$ and $\mathbf{g}_{i,j}$ for a single example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$ as the difference between the

Table 1: Semi-supervised classification dataset characteristics; training/testing hinge loss and testing Hamming loss for SSVM; number of testing-time cuts, training/testing game value and testing Hamming loss for our ARC approach.

Dataset Information				SSVM			ARC			
Name	#training	#testing	#features	Hinge$_{tr}$	Hinge$_{te}$	Hamming$_{te}$	#cuts	Value$_{tr}$	Value$_{te}$	Hamming$_{te}$
Diabetes	600	168	8	1.27	1.22	0.37	9	0.39	0.35	0.31
Breast Cancer	500	183	10	0.44	0.58	0.12	8	0.42	0.23	0.10
Gisette	800	200	4971	1.26	0.54	0.21	17	0.45	0.29	0.16
Spect	187	80	22	1.30	1.28	0.29	10	0.38	0.34	0.26

expected features under the adversary's distribution and the empirical features calculated from the training data:

$$\left\{\mathbb{E}_{\check{\mathbf{Y}}|\mathbf{x}^{(k)}\sim\tilde{P}}[\boldsymbol{\phi}_i(\check{Y}_i,\mathbf{x}^{(k)})]-\boldsymbol{\phi}_i(y_i^{(k)},\mathbf{x}^{(k)})\right\} \text{ and} \quad (18)$$

$$\left\{\mathbb{E}_{\check{\mathbf{Y}}|\mathbf{x}^{(k)}\sim\tilde{P}}[\boldsymbol{\phi}_{i,j}(\check{Y}_i,\check{Y}_j,\mathbf{x}^{(k)})]-\boldsymbol{\phi}_{i,j}(y_i^{(k)},y_j^{(k)},\mathbf{x}^{(k)})\right\}. \quad (19)$$

Extension to Associative Features

Following associative Markov network formulations (Taskar, Guestrin, and Koller 2004), we can also incorporate higher-order features of the variables rather than unary and pairwise features, transforming Eq. (7) to:

$$\min_{\hat{P}(\hat{\mathbf{Y}}|\mathbf{X})}\max_{\tilde{P}(\check{\mathbf{Y}}|\mathbf{X})}\mathbb{E}_{\mathbf{X}\sim\tilde{P},\check{\mathbf{Y}}|\mathbf{X}\sim\tilde{P},\check{\mathbf{Y}}|\mathbf{X}\sim\tilde{P}}\left[\text{loss}(\hat{\mathbf{Y}},\check{\mathbf{Y}})\right] \text{ such that:}$$

$$\mathbb{E}_{\mathbf{X}\sim\tilde{P},\check{\mathbf{Y}}|\mathbf{X}\sim\tilde{P}}\left[\boldsymbol{\phi}_i(\check{Y}_i,\mathbf{X})\right]=\mathbb{E}_{\mathbf{X},\mathbf{Y}\sim\tilde{P}}\left[\boldsymbol{\phi}_i(Y_i,\mathbf{X})\right],\forall i\in[n]$$

$$\mathbb{E}_{\mathbf{X}\sim\tilde{P},\check{\mathbf{Y}}|\mathbf{X}\sim\tilde{P}}\left[\boldsymbol{\phi}_c(\check{\mathbf{Y}}_c,\mathbf{X})\right]\leq\mathbb{E}_{\mathbf{X},\mathbf{Y}\sim\tilde{P}}\left[\boldsymbol{\phi}_c(\mathbf{Y}_c,\mathbf{X})\right],$$
$$\forall c\in\mathcal{C},|c|>1. \quad (20)$$

We define the higher order features on clique c as:

$$\boldsymbol{\phi}_c(\mathbf{y}_c,\mathbf{x})=\sum_k\left(\prod_j I(y_{c_j}=k)\right)\boldsymbol{\delta}_c(k,\mathbf{x}). \quad (21)$$

Different from the pairwise setting, we assign non-zero feature values only to cases where all the labels of the variables in the subset are the same (i.e., $\boldsymbol{\delta}_c(k,\mathbf{x})$ is the feature when the labels of all the variables in subset c are k). Finding the adversary's best response now becomes:

$$\operatorname*{argmax}_{\check{\mathbf{y}}}\mathbb{E}_{\hat{\mathbf{Y}}|\mathbf{x}\sim\hat{P}}\left[\sum_i I(\hat{Y}_i\neq\check{y}_i)\right]+\sum_i\psi_i(\check{Y}_i) \quad (22)$$
$$+\sum_{c:|c|>1}\psi_c(\check{\mathbf{y}}_c,\mathbf{x}).$$

To solve Eq. (22), we introduce indicator variables $u_{i,y_i'}$ and $u_c^k\in\{0,1\}$. $u_{i,y_i'}$ is 1 only if $y_i=y_i'$, and u_c^k is 1 only if the labels of the variables in c are all k. Combining them all into a single vector $\mathbf{u}$ of size q, the problem (22) can be formulated as an integer linear programming (ILP) problem:

$$\operatorname*{argmax}_{\mathbf{u}\in\{0,1\}^q}\sum_i\sum_{\check{y}_i'}u_{i,\check{y}_i'}\cdot\alpha_i(\check{y}_i')+\sum_{c:|c|>1}\sum_{k\in}u_c^k\cdot\alpha_c(\check{\mathbf{y}}_c',\mathbf{x})$$

$$\text{such that: }\sum_{\check{y}_i'}u_{i,\check{y}_i'}=1,\forall i\in[n]$$

$$u_c^k\leq u_{i,k},\forall|c|>1,i\in c,k, \quad (23)$$

where α_i and α_c are defined as:

$$\alpha_i(\check{y}_i,\mathbf{x})=\frac{1}{n}\sum_{\hat{\mathbf{y}}}\hat{P}(\hat{\mathbf{y}}|\mathbf{x})I(\hat{y}_i\neq\check{y}_i)+\boldsymbol{\psi}_i(\check{y}_i,\mathbf{x}),\forall i\in[n]$$

$$\alpha_c(\check{\mathbf{y}}_c,\mathbf{x})=\sum_k\left(\prod_j I(y_{c_j}=k)\right)\boldsymbol{\theta}_c\cdot\boldsymbol{\delta}_c(k,\mathbf{x}),\forall|c|>1.$$

Eq. (23) can be approximated by Linear Programming relaxation. For binary classification problems, when $\boldsymbol{\delta}_c(k,\mathbf{x})$ is non-positive, it can be optimized as a normal linear programming problem (without integer constraints) and will provide the integer solution (Taskar, Chatalbashev, and Koller 2004), which means it can be solved in polynomial time.

Applications

We focus our attention on binary-valued structured prediction tasks for which inference can be efficiently performed without extremely restrictive limitations on the potential functions or using approximation.

Semi-Supervised Classification

We first consider cut-based semi-supervised classification (Blum and Chawla 2001) using four datasets from the UCI repository (Lichman 2013). The characteristics of these datasets are summarized in Table 1. The vector $\mathbf{y}$ corresponds to the examples of each dataset. Following previous work, we seek to leverage the relationships between each example, in terms of input values $\mathbf{x}_i$, and its label, y_i, along with relationships between pairs of labels (y_i, y_j) (and their inputs ($\mathbf{x}_i,\mathbf{x}_j$)). We construct unary features directly from each example's input vector $\mathbf{x}_i$ and pairwise features as the inverse of the absolute difference of the two corresponding nodes features. We share the same unary and pairwise parameters across all edges so that these potentials can be applied to previously unseen examples at test time. During training time, we only incorporate labeled training examples. At testing time, we incorporate both the training set and the unlabeled testing set on which predictions are desired.

We compare our ARC approach with a structured support vector machine (Taskar et al. 2005; Tsochantaridis et al. 2004) on the same feature representation. Thus, the only difference between these methods is the learner's objective function being optimized. We evaluate performance using the Hamming loss on unlabeled testing datapoints, which corresponds to the misclassification rate for semi-supervised learning. We see from the results in Table 1 that the hinge

Table 2: Multi-label dataset information and average testing Hamming loss for binary relevance (BR), multi-label KNN (ML-KNN), and Rank-based support vector machines (Rank SVM), and our ARC approach (with average number of cuts).

Dataset Information					Test Hamming Loss				
Name	Domain	#Instances	#Features	#Labels	BR	MLKNN	Rank SVM	ARC	#cuts
Bibtex	text	7395	1836	159	0.015 ± 0.001	0.017 ± 0.001	0.120 ± 0.014	0.015 ± 0.001	20.8
Bookmarks	text	87856	2150	202	0.238 ± 0.018	0.149 ± 0.011	0.176 ± 0.016	0.141 ± 0.014	19.3
Birds	audio	645	260	19	0.156 ± 0.106	0.063 ± 0.001	0.124 ± 0.106	0.062 ± 0.010	7.4
CAL500	music	502	68	174	0.159 ± 0.016	0.113 ± 0.012	0.124 ± 0.016	0.102 ± 0.018	14.9
Emotions	music	593	72	6	0.261 ± 0.018	0.198 ± 0.016	0.183 ± 0.012	0.174 ± 0.010	13.7
Flags	images	194	19	7	0.271 ± 0.220	0.236 ± 0.014	0.234 ± 0.011	0.212 ± 0.010	18.5
Scene	images	2407	294	6	0.139 ± 0.010	0.144 ± 0.012	0.241 ± 0.015	0.110 ± 0.016	12.1
Yeast	biology	2417	103	14	0.238 ± 0.015	0.195 ± 0.110	0.210 ± 0.090	0.186 ± 0.014	11.6
NUS-WIDE	images	269648	128	81	0.120	0.028	0.102	0.020	14.5
				Average	0.177	0.127	0.168	0.113	14.8

loss of the trained SSVM model almost always exceeds 0.5, corresponding to a meaningless bound on the Hamming loss (i.e., randomly guessing achieves this value). Indeed, the hinge loss is often entirely above the range of the loss function $[0, 1]$). In contrast, the game values of our approach (ARC) always provide a meaningful upper bound on the Hamming loss that is below random guessing (0.5). This tighter bound translates into meaningful bounds from the game values on testing data and testing Hamming losses that are lower than SSVM across all datasets.

Additionally, we find that the equilibria are obtained from performing constraint generation a relatively small number of times. These range from performing 8 to 17 minimum cuts during the testing time prediction task. In terms of running time, this is the main difference from the SSVM model, which makes predictions based on the results of a single minimum cut problem. Though we do not investigate it in this paper, the similarity between minimum cut instances in each prediction task may allow reuse and acceleration of the inference procedure.

Multi-Label Prediction

The second application that we investigate is multi-label classification, like our running example. In this setting, multiple labels can be attached to each example and the prediction task is that of predicting some subset of the label set for each example. We treat each of the labels as a binary variable and follow the structure presented in the previous section to train an adversarial multi-label predictor by learning to make adversarial cuts. Most of the features we employ as unary and pairwise features are taken from the Mulan dataset (Tsoumakas et al. 2011). However, we additionally extracted word2vec[3] features for datasets with meaningful labels including Bibtex, Bookmarks, Cal500, Emotions, Scene and NUS-WIDE, and define a vector of features that is inversely proportioned to the distance between each dimension of the word2vec feature representation. When augmenting the features in this manner, we omit a small number of labels with no word2vec features from the dataset label representation, as described in Equation (3). As shown in Table 2, we have considered datasets with different sizes from a variety of application areas to show the general performance of our approach.

We compare the performance of ARC with Multi-label KNN (Zhang and Zhou 2007), Binary relevance (BR) (Tsoumakas, Katakis, and Vlahavas 2009), and Rank SVM (Elisseeff and Weston 2002) on nine different benchmark datasets. We perform 10-fold cross-validation and report both the mean and standard deviation of the Hamming loss, except for the extremely large NUS-WIDE dataset, for which we only compute the Hamming loss for a singe testing sample due to its size. As shown in Table 2, our ARC approach performs at least as well as the other methods on each individual dataset, and much better on average.

We also find that a relatively small number of minimum cuts are needed by our ARC algorithm on average (between 7.4 and 20.8) to find the equilibria. We see no clear relationship between the number of constraints generated and the number of class labels, number of instances, or Hamming loss. This suggests that the amount of support for the equilibrium does not depend heavily on the size of the full game representation nor the difficulty of the prediction game.

Discussion

We investigated a robust approach for learning to make cuts in graphs. It operates by making worst-case approximations to the training labels. This has benefits theoretically—providing meaningful bounds on losses—and in practice, as illustrated by our experiments. In future work, we plan to investigate the benefits of our game formulation for multiclass problems where only approximately optimal graph cuts can be obtained. We expect that because the equilibrium is defined over many different cuts, rather than the single best alternative (as in structured SVM's hinge loss), that approximations will have a less detrimental impact on our approach.

Acknowledgments

This research was supported in part by NSF CAREER grant #1652530 and RI grant #1526379.

[3]https://code.google.com/p/word2vec/

References

Asif, K.; Xing, W.; Behpour, S.; and Ziebart, B. D. 2015. Adversarial cost-sensitive classification. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Blum, A., and Chawla, S. 2001. Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning*, 19–26. Morgan Kaufmann Publishers Inc.

Blum, A.; Lafferty, J.; Rwebangira, M. R.; and Reddy, R. 2004. Semi-supervised learning using randomized mincuts. In *Proceedings of the twenty-first international conference on Machine learning*, 13. ACM.

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

Boykov, Y.; Veksler, O.; and Zabih, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence* 23(11):1222–1239.

Dalvi, N.; Domingos, P.; Sanghai, S.; Verma, D.; et al. 2004. Adversarial classification. In *KDD*, 99–108. ACM.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* 12:2121–2159.

Dudík, M., and Schapire, R. E. 2006. Maximum entropy distribution estimation with generalized regularization. In *Learning Theory*. Springer Berlin Heidelberg. 123–138.

Elisseeff, A., and Weston, J. 2002. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, 681–687.

Fathony, R.; Bashiri, M. A.; and Ziebart, B. 2017. Adversarial surrogate losses for ordinal regression. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc. 563–573.

Fathony, R.; Liu, A.; Asif, K.; and Ziebart, B. 2016. Adversarial multiclass classification: A risk minimization perspective. In *Advances in Neural Information Processing Systems*, 559–567.

Flake, G. W.; Tarjan, R. E.; and Tsioutsiouliklis, K. 2004. Graph clustering and minimum cut trees. *Internet Mathematics* 1(4):385–408.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2672–2680.

Greig, D. M.; Porteous, B. T.; and Seheult, A. H. 1989. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)* 271–279.

Grünwald, P. D., and Dawid, A. P. 2004. Game theory, maximum entropy, minimum discrepancy, and robust Bayesian decision theory. *Annals of Statistics* 32:1367–1433.

Ishikawa, H. 2003. Exact optimization for markov random fields with convex priors. *IEEE transactions on pattern analysis and machine intelligence* 25(10):1333–1336.

Joachims, T. 2005. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, 377–384. ACM.

Kolmogorov, V., and Zabin, R. 2004. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence* 26(2):147–159.

Li, J.; Asif, K.; Wang, H.; Ziebart, B. D.; and Berger-Wolf, T. 2016. Adversarial sequence tagging. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1690–1696.

Lichman, M. 2013. UCI machine learning repository.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the International Conference on Machine Learning*, 536–543.

Potts, R. B. 1952. Some generalized order-disorder transformations. In *Mathematical proceedings of the cambridge philosophical society*, volume 48, 106–109. Cambridge Univ Press.

Taskar, B.; Chatalbashev, V.; Koller, D.; and Guestrin, C. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning*, 896–903. ACM.

Taskar, B.; Chatalbashev, V.; and Koller, D. 2004. Learning associative markov networks. In *Proceedings of the twenty-first international conference on Machine learning*, 102. ACM.

Taskar, B.; Guestrin, C.; and Koller, D. 2004. Max-margin markov networks. In *Advances in neural information processing systems*, 25–32.

Topsøe, F. 1979. Information theoretical optimization techniques. *Kybernetika* 15(1):8–27.

Tsochantaridis, I.; Hofmann, T.; Joachims, T.; and Altun, Y. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning*, 104. ACM.

Tsoumakas, G.; Spyromitros-Xioufis, E.; Vilcek, J.; and Vlahavas, I. 2011. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research* 12:2411–2414.

Tsoumakas, G.; Katakis, I.; and Vlahavas, I. 2009. Mining multi-label data. In *Data mining and knowledge discovery handbook*. Springer. 667–685.

von Neumann, J., and Morgenstern, O. 1947. *Theory of Games and Economic Behavior*. Princeton University Press.

Wainwright, M. J., and Jordan, M. I. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* 1(1-2):1–305.

Wang, H.; Xing, W.; Asif, K.; and Ziebart, B. 2015. Adversarial prediction games for multivariate losses. In *Advances in Neural Information Processing Systems*, 2728–2736.

Zhang, M.-L., and Zhou, Z.-H. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition* 40(7):2038–2048.