Convex Adversarial Collective Classification

MohamadAli Torkamani ALI@CS.UOREGON.EDU
Daniel Lowd LOWD@CS.UOREGON.EDU
Department of Computer and Information Science, University of Oregon

Abstract

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

1. Introduction

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

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

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

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

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

stances are independent, ignoring the relational nature of many domains.

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

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

The rest of our paper is organized as follows. In Section 2, we present a brief overview of Markov networks and associative Markov networks as applied to collective classification. In Section 3, we review previous work on adversarial machine learning. We introduce our formulation and algorithm in Section 4. Section 5 contains our experiments on real and synthetic data, and we conclude in Section 6 with a discussion of ongoing and future work.

2. Max-Margin Relational Learning

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The goal of the AMN optimization problem is to maximize the margin between the log probability of the true labeling, $h(\mathbf{w}, \mathbf{x}, \hat{\mathbf{y}})$, and any alternative labeling, $h(\mathbf{w}, \mathbf{x}, \mathbf{y})$. For our problem, h follows from (Eq. 1): $h(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \sum_{i,j,k} w_j^k x_{ij} y_i^k + \sum_{(i,j)\in E,k} w_e^k y_i^k y_j^k$. We can omit the $\log Z(\mathbf{x})$ term because it cancels in the difference. Margin scaling is used to enforce a wider margin from labelings that are more different. We defined this difference as the Hamming distance: $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = N - \sum_{i,k} y_i^k \hat{y}_i^k$ where N is the total number of objects. We thus obtain the following minimization problem with an exponential number of constraints (one for each $\mathbf{y}$):

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

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

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \mathbf{w} \geq 0; \\ & \xi - N \geq \max_{\mathbf{y}\in\mathcal{Y}'} \sum_{i,j,k} w_j^k x_{ij} (y_i^k - \hat{y}_i^k) \\ & \qquad + \sum_{(i,j)\in E,k} w_e^k (y_{ij}^k - \hat{y}_{ij}^k) - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \end{aligned} \tag{3}$$

Finally, since the inner maximization is itself a linear program, we can replace it with the minimization of its dual to obtain a single quadratic program (not shown). For the two-class setting, Taskar et al. prove that the inner program always has an integral solution, which guarantees that the weights found by the outer quadratic program are always optimal.

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

3. Adversarial Machine Learning

Most classic learning algorithms assume that training and test data are drawn from the same distributions. However, in many real world applications, an adversary will actively change its behavior to avoid detection, leading to significantly worse performance in practice. For example, spammers add and remove words from their email messages in order to bypass spam filters, and web spammers try to deceive search engines by creating "link farms" to make a web site seem more important. In computer and network security, many bots are engineered to attack network computers and change their behavior so that intrusion detection systems fail to detect them.

Designing machine learning algorithms that are robust to malicious adversaries is an area of growing interest (Laskov & Lippmann, 2010). One approach is to formulate the problem as a game between the learner and an adversary, each with its own set of strategies and rewards. Dalvi et al. (2004) note that finding a Nash equilibrium is often intractable and propose a strategy to anticipate the adversary's next move instead. Brückner and Scheffer (2009) present a method to find a Nash equilibrium for non-zero sum games that satisfy certain convexity conditions. In later work, they present results for finding Stackelberg equilibria as well (Brückner & Scheffer, 2011). One special case of adversarial manipulation is feature deletion, in which the adversary chooses the features to remove that would most harm the classifier's performance. This results in a zero-sum game between the learner and adversary that can be solved using robust minimax methods as in (Lanckriet et al., 2004; El Ghaoui et al., 2003; Kim et al., 2006; Globerson & Roweis, 2006). Teo et al. (2008) is more general, allowing any set of adversarial actions that afford an efficient numerical solution to be represented as an invariant while learning.

We take particular inspiration from Globerson and Roweis (2006) and Teo et al. (2008), which take the quadratic program of a max-margin learning problem and substitute in the adversary's worst-case modification of the evidence. By formulating the adversary's modification as a linear program and taking the dual, the learning problem remains convex.

However, none of these methods handles collective classification, in which the label of each object depends on the labels of its neighbors.

4. Convex Adversarial Collective Classification

Collective classification problems are hard because the number of joint label assignments is exponential in the number of nodes. As discussed in Section 2, if neighboring nodes are more likely to have the same label, then the collective classification problem can be represented as an associative Markov network (AMN), in which max-margin learning and MPE inference are both efficient. To construct an adversarial collective classifier, we start with the AMN formulation (Eq. 3) and incorporate an adversarial invariant, similar to the approach of Globerson and Roweis (2006). Specifically, we assume that the adversary may change up to D binary-valued features x_{ij}, for some positive integer D that we select in advance. We use $\hat{\mathbf{x}}$ to indicate the true features and $\mathbf{x}$ to indicate the adversarially modified features. The number of changes can be written as: $\Delta(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{i,j} x_{ij} + \hat{x}_{ij} - 2x_{ij}\hat{x}_{ij}$

We define the set of valid $\mathbf{x}$ as $\mathcal{X}' = \{\mathbf{x} : 0 \leq x_{ij} \leq 1; \Delta(\mathbf{x}, \hat{\mathbf{x}}) \leq D\}$. Note that $\mathcal{X}'$ is a relaxation that allows fractional values, much like the set $\mathcal{Y}'$ defined by Taskar et al. We will later show that there is always an integral solution when both the features and labels are binary-valued.

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

$$\begin{aligned}
&\max_{\mathbf{y}\in\mathcal{Y}',\mathbf{x}\in\mathcal{X}'} h(\mathbf{w},\mathbf{x},\mathbf{y}) - h(\mathbf{w},\mathbf{x},\hat{\mathbf{y}}) + \Delta(\mathbf{y},\hat{\mathbf{y}}) \\
&= \max_{\mathbf{y}\in\mathcal{Y}',\mathbf{x}\in\mathcal{X}'} \sum_{i,j,k} w_j^k x_{ij} y_i^k + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k \\
&\qquad - \sum_{i,j,k} w_j^k x_{ij} \hat{y}_i^k - \sum_{(i,j)\in E,k} w_e^k \hat{y}_{ij}^k \\
&\qquad + N - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \qquad (4)
\end{aligned}$$

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

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

following linear program:

$$\begin{aligned}\max_{\mathbf{x},\mathbf{y},\mathbf{z}} \quad & \sum_{i,j,k} w_j^k(z_{ij}^k - \hat{y}_i^k x_{ij}) + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \\ \text{s.t.} \quad & 0 \le x_{ij} \le 1; \quad \sum_{i,j} x_{ij} + \hat{x}_{ij} - 2x_{ij}\hat{x}_{ij} \le D \\ & 0 \le y_i^k; \quad \sum_k y_i^k = 1; \quad y_{ij}^k \le y_i^k; \quad y_{ij}^k \le y_j^k \\ & z_{ij}^k \le x_{ij}; \quad z_{ij}^k \le y_i^k \quad \forall i,j,k \end{aligned} \tag{5}$$

Given the model's weights, this linear program allows the adversary to change up to D binary features. Recall that, in the AMN formulation, the exponential number of constraints separating the true labeling from all alternate labelings are replaced with a single non-linear constraint that separates the true labeling from the best alternate labeling (Eqs. 2,3). This non-linear constraint contains a nested maximization. We have a similar scenario, but here the margin can also be altered by changing the binary features, affecting the probabilities of both the true and alternate labelings. By substituting this new MPE inference task (Eq. 5) into the original AMN's formulation, the resulting program's optimal solution will be robust to the worst manipulation of the input feature vector:

$$\begin{aligned}\min_{\mathbf{w},\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \quad \text{s.t.} \quad \mathbf{w} \ge 0; \\ \xi - N \ge \max_{\mathbf{x},\mathbf{y},\mathbf{z}} & \sum_{i,j,k} w_j^k(z_{ij}^k - \hat{y}_i^k x_{ij}) + \sum_{(i,j)\in E,k} w_e^k y_{ij}^k \\ & - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \qquad \text{s.t.} \\ & 0 \le y_i^k; \ \sum_k y_i^k = 1; \ y_{ij}^k \le y_i^k; \ y_{ij}^k \le y_j^k \\ & 0 \le x_{ij} \le 1; \ \sum_{i,j} x_{ij} + \hat{x}_{ij} - 2x_{ij}\hat{x}_{ij} \le D \\ & z_{ij}^k \le x_{ij}; \quad z_{ij}^k \le y_i^k \end{aligned} \tag{6}$$

The mathematical program in Eq. (6) is not convex because of the bilinear terms and the nested maximization (similar to solving a bilevel Stackelberg game). Fortunately, we can use the strong duality property of linear programs to resolve both of these difficulties. The dual of the maximization linear program is a minimization linear program with the same optimal value as the primal problem. Therefore, we can replace the inner maximization with its dual minimization problem to obtain a single convex quadratic program that minimizes over $\mathbf{w}$, ξ, and the dual variables (not shown). A similar approach is used by Globerson and Roweis (2006). As long as this relaxed program has an integral optimum, it is equivalent to maximizing only over integral $\mathbf{x}$ and $\mathbf{y}$. Thus, the overall program will find optimal weights. Taskar et al. (2004a) prove that the inner maximization in a 2-class AMN always has an integral solution. We can prove a similar result for the adversarial AMN:

Theorem 1. *Eq. 5 has an integral optimum when* $\mathbf{w} \ge 0$ *and the number of classes is 2.*

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

$$v_{ij}^1 = \begin{cases} x_{ij} & \text{if } w_j^1 > 0, \\ 1 - x_{ij} & \text{if } w_j^1 = 0. \end{cases}$$
$$v_{ij}^2 = 1 - v_{ij}^1$$

By construction:

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

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

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

Many extensions of our model are possible. One extension is to restrict the adversary to only changing certain features of certain objects. For example, in a web spam domain, we might assume that the adversary will only modify spam pages. We could also have different budgets for different types of changes, such as a separate budget for each web page, or even

separate budgets for changing the title of a web page and changing its body. These are easily expressed by changing the definition of $\mathcal{X}'$ and adding the appropriate constraints to the quadratic program. Our model can also support higher-order cliques, as described by Taskar et al. (2004a), as long as they are associative. For simplicity, our exposition and experiments focus on the simpler case described above.

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

5. Experiments

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

5.1. Datasets

We evaluated our method on three collective classification problems.

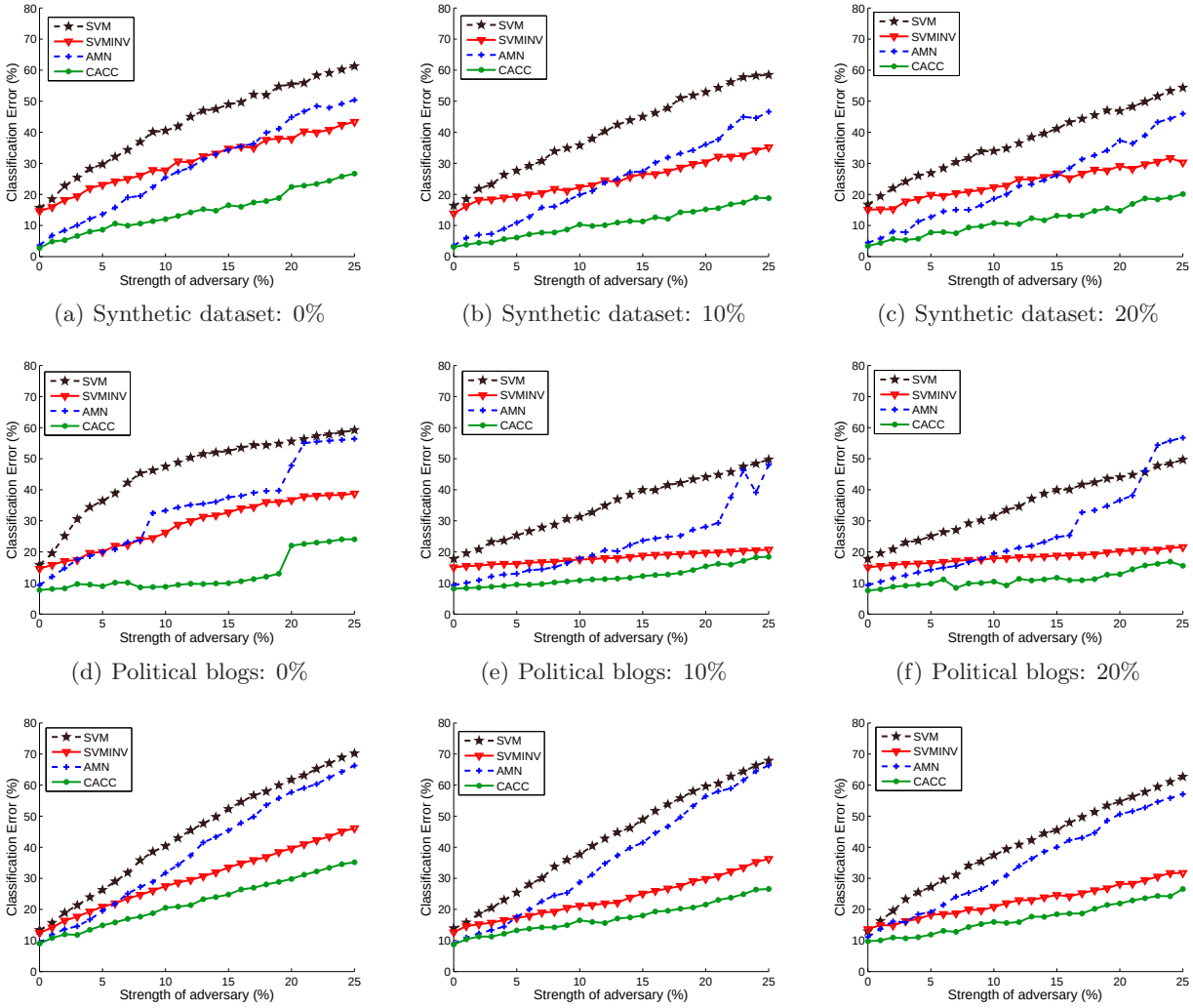
Synthetic. To evaluate the effectiveness of our method in a controlled setting where the distribution is known, we constructed a set of 10 random graphs, each with 100 nodes and 30 Boolean features. Of the 100 nodes, half had a positive label ('+') and half had a negative label ('−'). Nodes of the same class were more likely to be linked by an edge than nodes with different classes. The features were divided evenly into three types: positive, negative, and neutral. Half of the positive and negative nodes had different feature distributions based on their class; that is, the positive nodes had more positive attributes and the negative nodes had more negative attributes, on average. In such nodes, on average there are 6 words, one of which is of the opposite class's words, two words are consistent with the class label and three words are neutral. The other half of the nodes had an ambiguous distribution consisting mainly of the neutral words (on average one word is consistent with class label, one word is not consistent and 3 words are neutral). Therefore, an effective classifier for these graphs must rely on both the attributes and relations. On average, each node had 8 neighbors, 7 of which had the same class and 1 of which had a different class.

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

Reuters. As our third dataset, we prepared a Reuters dataset similar to the one used by Taskar et al. (2004a). We took the ModApte split of the Reuters-21578 corpus and selected articles from four classes: crude, grain, trade, and money-fx. We used the 200 words with highest mutual information as features. We linked each document to the two most similar documents based on TF-IDF weighted cosine distance. We split the data into 7 sets based on time, and performed the tuning and then the training phases based on this temporal order (as explained in 5.2).

5.2. Methodology and Metrics

In order to evaluate the robustness of these methods to malicious adversaries, we applied a simulated adversary to both the tuning data and the test data. We assumed the worst-case scenario, in which the adversary has perfect knowledge of the model parameters and only wants to maximize the error rate of the classifier. Since exactly maximizing the error rate is typically NP-hard, our intelligent adversary instead maximizes the margin loss by solving the linear program in Eq. (5) for a fixed budget. Each model was attacked separately. On the validation data, we used adversarial budgets of 0% (no adversarial manipulation), 10%,



(a) Synthetic dataset: 0% (b) Synthetic dataset: 10% (c) Synthetic dataset: 20%

(d) Political blogs: 0% (e) Political blogs: 10% (f) Political blogs: 20%

(g) Reuters: 0% (h) Reuters: 10% (i) Reuters: 20%

Figure 1. Accuracy of different classifiers in presence of worst-case adversary. The number following the dataset name indicates the adversary's strength at the time of parameter tuning. The x-axis indicates the adversary's strength at test time. Smaller is better.

and 20% of the total number of features present in the data. This allowed us to tune our models to "expect" adversaries of different strengths. Of course, we rarely know the exact strength of the adversary in advance. Thus, on the test data, we used budgets that ranged from 0% to 25%, in order to see how well different models did against adversaries that were weaker and stronger than expected.

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

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

We used CPLEX to solve all quadratic and linear pro-

gramming problems. Most problems were solved in less than 1 minute on a single core.

All of our code and datasets are available upon request.

5.3. Results and Discussion

Figure 1 shows the performance of all four methods on test data manipulated by rational adversaries of varying strength (0%-25%), after being tuned against adversaries of different strengths (0%, 10%, and 20%). Lower is better. On the far left of each graph is performance without an adversary. To the right of each graph, the strength of the adversary increases.

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

As expected, tuning against stronger adversaries (10% and 20%) makes CACC more effective against stronger adversaries at test time. Surprisingly, tuning against a stronger adversary does not significantly reduce performance against weaker adversaries: CACC remains nearly as effective against no adversary when tuned for a 20% adversary as when tuned for no adversary. Specifically, when there is no adversary at test time, the increase in error rate from training against a 20% adversary is less than 1% on Synthetic and Reuters, and on Political the error rate actually decreases slightly. Thus, this additional robustness comes at a very small cost.

In Figures 1(d), 1(e), and 1(f), the AMN classification error jumps sharply as the adversary budget increases. This is the point when enough nodes are mis-classified that links are actively misleading in one or two of the eight cross-validation folds, leading to worse performance than the SVM for those folds. This demonstrates that relational classifiers are potentially more vulnerable to adversarial attacks than non-relational classifiers. A smoother version of this effect can also be observed on both the synthetic dataset and Reuters.

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

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

6. Conclusion

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

In future work, we intend to extend our methods to learn adversarially regularized variants of non-associative relational models, using approximate inference and constraint generation methods as necessary to cope with the intractability of inference. We would also like to apply our methods to larger, more realistic adversarial problems, such as web-spam. In addition to larger size, many of these problems are semi-supervised and include numeric attributes, which would require some modifications to CACC.

Acknowledgments

We thank the anonymous reviewers for useful comments. This research was partly funded by ARO grant W911NF-08-1-0242 and NSF grant OCI-0960354. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, NSF, or the U.S. Government.

References

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

Adamic, L.A. and Glance, N. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43. ACM, 2005.

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

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

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

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

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

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

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

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

Huynh, T.N. and Mooney, R.J. 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*, pp. 564–579. Springer, 2009.

Kim, S.J., Magnani, A., and Boyd, S. Robust fisher discriminant analysis. *Advances in Neural Information Processing Systems*, 18:659, 2006.

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

Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., and Jordan, M.I. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

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

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

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

Lowd, D. and Meek, C. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 641–647. ACM, 2005b. ISBN 159593135X.

Nelson, B., Rubinstein, B.I.P., Huang, L., Joseph, A.D., Lau, S., Lee, S.J., Rao, S., Tran, A., and Tygar, JD. 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.

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

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29(3):93, 2008. ISSN 0738-4602.

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

Taskar, B., Wong, M. F., Abbeel, P., and Koller, D. Max-margin Markov networks. In Thrun, S., Saul, L., and Schölkopf, B. (eds.), *Advances in Neural Information Processing Systems 16.* MIT Press, Cambridge, MA, 2004b.

Teo, C.H., Globerson, A., Roweis, S., and Smola, A. Convex learning with invariances. In *Advances in Neural Information Processing Systems 21*, 2008.