
Convex Adversarial Collective Classification

Ali Torkamani

Dept. of Computer and Information Science
University of Oregon
ali@cs.uoregon.edu

Daniel Lowd

Dept. of Computer and Information Science
University of Oregon
lowd@cs.uoregon.edu

Abstract

Many real-world domains, such as web spam, auction fraud, and counter-terrorism, are both relational and adversarial. Existing work on adversarial machine learning assumes that the attributes of each instance can be manipulated independently. Collective classification violates this assumption, since object labels depend on the labels of related objects as well as their own attributes. 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 [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. Statistical relational learning (SRL) methods such as Markov logic [2] 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 [3,4], other explicitly adversarial domains include counter-terrorism, online auction fraud [5], 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 [6], reverse engineering classifiers [7,8] (and later: [9]), and building classifiers robust to feature deletion or other invariants [10,11]. More recently, Brückner and Scheffer showed that, under modest as-

sumptions, Nash equilibria can be found for domains such as spam [12]. However, current adversarial methods assume that instances 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 [13] and convex learning with invariants [11]. Unlike previous work in learning graphical models, CACC selects the most effective weights *assuming a worst-case adversary* which 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 solution 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, Markov logic networks, and max-margin parameter estimation. 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

Markov networks (MNs) represent the joint distribution over a set of random variables $\mathbf{X} = \{X_1, \dots, X_N\}$ 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 i th factor, and $\mathbf{D}_i \subset \mathbf{X}$ is the scope of the i th 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) [13] 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.

Markov logic [2] is a language for conveniently defining Markov networks over relational domains. A *Markov logic network* (MLN) is a set of weighted first-order formulas (w_i, F_i) . Together with a set of logical constants representing objects in the domain, an MLN induces a log-linear model where the features n_i are the number of satisfied groundings of the formulas F_i :

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

Here, the random variables in \mathbf{X} are all the ground atoms that can be defined with the given set of logical constants. For example, if our domain includes a binary predicate $\text{Friends}(x, y)$ and our set of constants is $\{\text{Anna}, \text{Bob}\}$, then the random variables are given by the ground atoms $\text{Friends}(\text{Anna}, \text{Anna})$, $\text{Friends}(\text{Anna}, \text{Bob})$, $\text{Friends}(\text{Bob}, \text{Anna})$, and $\text{Friends}(\text{Bob}, \text{Bob})$. MLNs can also be viewed as templates for creating large MNs, where each grounding of the formula F_i defines a factor with value e^{w_i} when that grounding formula is satisfied and 1 otherwise.

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

MLNs make it easy to compactly describe very complex distributions. For example, a simple collective classification model can be defined using relatively simple formulas, as shown in Table 1. The subscript j and superscript k indicate that different formulas are defined for each attribute $j \in \{1, \dots, M\}$ and object label $k \in \{1, \dots, K\}$. The formula from the first line define features for the prior distribution over labels in the absence of any attributes or links. The next line relates each object’s attributes to its label. The third line relates the labels of neighboring objects. Note that

Table 1: MLN formulas for a simple collective classification model.

Weight	Formula
w_0^k	$\text{Label}(o) = k$
w_j^k	$\text{Attribute}_j(o) \Rightarrow \text{Label}(o) = k$
w_e^k	$\text{Link}(o, o') \wedge (\text{Label}(o) = k) \Rightarrow (\text{Label}(o') = k)$

the first formula may be omitted as a special case of the second if we assume that a special bias attribute $\text{Attribute}_0(o)$ is true for every object o .

The MLN from Table 1 can be represented as the following log-linear model:

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

where x_{ij} represents the value of the j th attribute of the i th object and y_i^k is an indicator variable which equals 1 when the i th object is assigned the k th label and 0 otherwise. E denotes the set of pairs (i, j) such that the i th and j th object are linked.

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:

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

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 [14]. 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., [15]). An alternative is to maximize the margin between the correct labeling and all alternative labelings, as done by max-margin Markov networks (M^3Ns) [16] and max-margin Markov logic networks (M^3LNs) [17]. 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 [13]. We now briefly describe the solution of Taskar et al. [13], which will later motivate our adversarial extension of AMNs. (We use slightly different notation from the original presentation in Taskar et al. [13] 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, $h(\mathbf{w}, \mathbf{x}, \mathbf{y})$. Margin scaling is used to enforce a wider margin from labelings that are more different, according to some difference function $\Delta(\mathbf{y}, \hat{\mathbf{y}})$. We thus obtain the following minimization problem with an exponential number of constraints (one for each \mathbf{y}):

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

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. For our problem, h is defined by the simple collective classification model, and Δ is Hamming distance:

$$\begin{aligned} h(\mathbf{w}, \mathbf{x}, \mathbf{y}) &= \sum_{ijk} w_j^k x_{ij} y_i^k + \sum_{(i,j) \in E, k} w_e^k y_i^k y_j^k \\ \Delta(\mathbf{y}, \hat{\mathbf{y}}) &= N - \sum_{i,k} y_i^k \hat{y}_i^k \end{aligned}$$

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 $\mathbf{y} \in \mathcal{Y}'$:

$$\mathcal{Y}' = \{\mathbf{y} : y_i^k \geq 0; \sum_k y_i^k = 1; y_{ij}^k \leq y_i^k; y_{ij}^k \leq y_j^k\}$$

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:

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

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. [13]. 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, 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, there are many bots that are engineered to attack network computers and change their behavior such that intrusion detection systems fail to detect them.

Designing machine learning algorithms that are robust to malicious adversaries is an area of growing interest [18]. 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. [6] note that finding a Nash equilibrium is often intractable and propose a strategy to anticipate the adversary’s next move instead. Brückner and Schefter [12] 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 [19].

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 [10, 20–22]. Teo et al. [11] 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 [10] and Teo et al. [11], 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 handle collective classification, in which the label of each object depends on the labels of its neighbors.

4 Convex Adversarial Collective Classification

The collective classification problems are hard because there are exponential number of possible joint label assignments that grow by the number of nodes. By introducing AMN’s, Taskar et al. [13] show that if neighboring nodes are more likely to have the same labels then the MAP inference problem can be efficiently solved by a linear program, this property is called “associativity”. To construct an adversarial collective classifier, we start with the AMN formulation (Equation 1) and incorporate an adversarial invariant, similar to the approach of Globerson and Roweis [10]. 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 can optionally restrict the changed features to some subset of indices, \mathcal{A} , in order to prevent the simulated adversary from modifying certain features in certain examples. 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. For clarity of exposition, we omit all such restrictions for now, but they are easy to add back in. Our set of valid \mathbf{x} is thus defined 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 fraction values, much like the

set \mathcal{Y}' defined by Taskar et al.

In our adversarial formulation, we want the true labeling $\hat{\mathbf{y}}$ to be separated from any alternate labeling $\mathbf{y} \in \mathcal{Y}'$ given any $\mathbf{x} \in \mathcal{X}'$. We can accomplish this by performing the maximization over both \mathbf{x} and \mathbf{y} , in order to find the “highest-scoring” combination of \mathbf{x} and \mathbf{y} out of all possibilities:

$$\begin{aligned} & \max_{\mathbf{y} \in \mathcal{Y}', \mathbf{x} \in \mathcal{X}'} \sum_{i,j,k} w_j^k x_{ij} (y_i^k - \hat{y}_i^k) + \\ & \sum_{(i,j) \in E, k} w_e^k (y_{ij}^k - \hat{y}_{ij}^k) - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \end{aligned} \quad (2)$$

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 $z_{ij}^k \leq y_i^k$. This 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:

$$\begin{aligned} & \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} \sum_{i,j,k} w_j^k (z_{ij}^k - \hat{y}_i^k x_{ij}) + \sum_{(i,j) \in E, k} w_e^k y_{ij}^k - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \\ \text{s.t. } & 0 \leq x_{ij} \leq 1; \quad \sum_{i,j} x_{ij} + \hat{x}_{ij} - 2x_{ij}\hat{x}_{ij} \leq D \\ & 0 \leq y_i^k; \quad \sum_k y_i^k = 1; \quad y_{ij}^k \leq y_i^k; \quad y_{ij}^k \leq y_j^k \\ & z_{ij}^k \leq x_{ij}; \quad z_{ij}^k \leq y_i^k \quad \forall i, j, k \end{aligned} \quad (3)$$

Given the model’s weights, this linear program allows the adversary to change binary features up to a fixed budget D . Recall that, in AMN’s formulation, M³N relaxation of exponential number of constraints results to a single non-linear constraint which has a maximization program in its heart. We have a similar scenario here but this time the margin can also be altered in its worst possible form by the changes that can be performed on the binary features. Thus substituting this new MAP inference task with the inner maximization program in the original AMN’s formulation, the resulting program’s optimal solution will be also robust to worst case invariants of input feature vectors. This new formulation converts the the AMN’s original quadratic program to an adversarially robust formulation:

$$\begin{aligned} & \min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C\xi \quad \text{s.t. } \mathbf{w} \geq 0; \\ & \xi - N \geq \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} \sum_{i,j,k} w_j^k (z_{ij}^k - \hat{y}_i^k x_{ij}) + \sum_{(i,j) \in E, k} w_e^k y_{ij}^k \\ & \quad - \sum_{i,k} y_i^k \cdot \hat{y}_i^k \\ \text{s.t. } & 0 \leq y_i^k; \quad \sum_k y_i^k = 1; \quad y_{ij}^k \leq y_i^k; \quad y_{ij}^k \leq y_j^k \\ & 0 \leq x_{ij} \leq 1; \quad \sum_{i,j} x_{ij} + \hat{x}_{ij} - 2x_{ij}\hat{x}_{ij} \leq D \\ & z_{ij}^k \leq x_{ij}; \quad z_{ij}^k \leq y_i^k \end{aligned} \quad (4)$$

Clearly the mathematical program in Eq. (4) is not convex because of the bilinear terms, and since it has an bilevel Stackelberg nature, it hard to solve. The good news is that we can use strong duality property of linear programs and resolve both of these difficulties. The dual of a primal maximization linear program is a minimization linear program, whose objective is an upper bound for the primal problem, and the gap between the value of both primal and dual objectives is zero at the optimal solution point. Based on this property for the second constraint of Eq. (4) after substituting the dual, “ $\xi - N \geq \text{dual objective}$ ” entails “ $\xi - N \geq \min \text{dual objective}$ ”, that also entails “ $\xi - N \geq \min \text{dual objective} \geq \max \text{primal objective}$ ” which is equal to “ $\xi - N \geq \max \text{primal objective}$ ”. This approach is also used in [10, 13] to remove the bilinearity from the program.

Therefore, we can substitute its dual into the original program to obtain a single convex quadratic program for learning adversarial AMN weights. As long as this relaxed program has an integral optimum, it is equivalent maximizes only over integral \mathbf{x} and \mathbf{y} . Thus, the overall program will find optimal weights.

Taskar et al. [13] prove that the inner maximization in a 2-class AMN always has an integral solution. We can prove a similar result for the adversarial AMN:

Theorem 1. *Equation 3 has an integral optimum when $\mathbf{w} \geq 0$ and the number of classes is 2.*

Proof Sketch. The structure of our argument is to show that an integral optimum exists by taking an arbitrary adversarial AMN problem and constructing an equivalent AMN problem that has an integral solution. Since the two problems are equivalent, the original adversarial AMN must also have an integral solution.

First, we use a Lagrange multiplier to incorporate the constraint $\Delta(\mathbf{x}, \hat{\mathbf{x}}) \leq D$ directly into the maximization. The extra term acts as a “per-change” penalty, which remains linear in \mathbf{x} . Minimizing over the Lagrange multiplier effectively adjusts this per-change penalty until there are at most D changes between \mathbf{x} and $\hat{\mathbf{x}}$, but does not affect the integrality of the inner maximization.

Next, we replace all \mathbf{x} variables with equivalent variables \mathbf{v} . Assume that either $w_j^1 = 0$ or $w_j^2 = 0$, for all j . (If both are positive, then we can subtract the smaller value from both to obtain a new set of weights with the same optimum as before.) We define \mathbf{v} as follows:

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

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} . \square

5 Experiments

In this section, we describe our experimental evaluation of CACC. Since CACC is both adversarial and relational, we compared it to three baselines: AMNs [13], which are relational but not adversarial; SVMInvar [11], which is adversarial but not relational; and SVMs with a linear kernel, which are neither. All three baselines 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 Data sets

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. The other half of the nodes had an ambiguous distribution consisting mainly of the neutral words. 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 data set collected by Adamic and Glance [23]. The original data set contains 1490 online blogs captured during the 2008 election cycle, their political affiliation (liberal or conservative), and

their linking relationships to other blogs. We extended this data set with word information from four different crawls at different dates: early February (EFeb), late February (LFeb), early May (EMay) and late May (LMay), all in 2012. We used mutual information to select the 100 words that best predict the class label [24], 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 data set were no longer active, and had been replaced by empty or spam web pages. We manually removed these from consideration, along with all blogs that had zero of the selected words. Finally, we partitioned the blogs into two disjoint subsets and removed all edges between nodes in the different subsets.

Reuters. As our third data set, we prepared a Reuters data set similar to the one used by Taskar et al. [13]. 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 100 words with highest mutual information as features. We linked each document to the two most similar documents based on TF-IDF weighted cosine distance.

5.2 Methodology and Metrics

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 as well. All parameters were selected to maximize performance on the tuning set in the absence of any explicit 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 $t + 2$.

In addition to evaluating on the unmodified test data, we also modified the test data by applying simulated adversaries of different strengths, in order to see how each model degraded under direct attack.

We used CPLEX to solve all quadratic and linear pro-

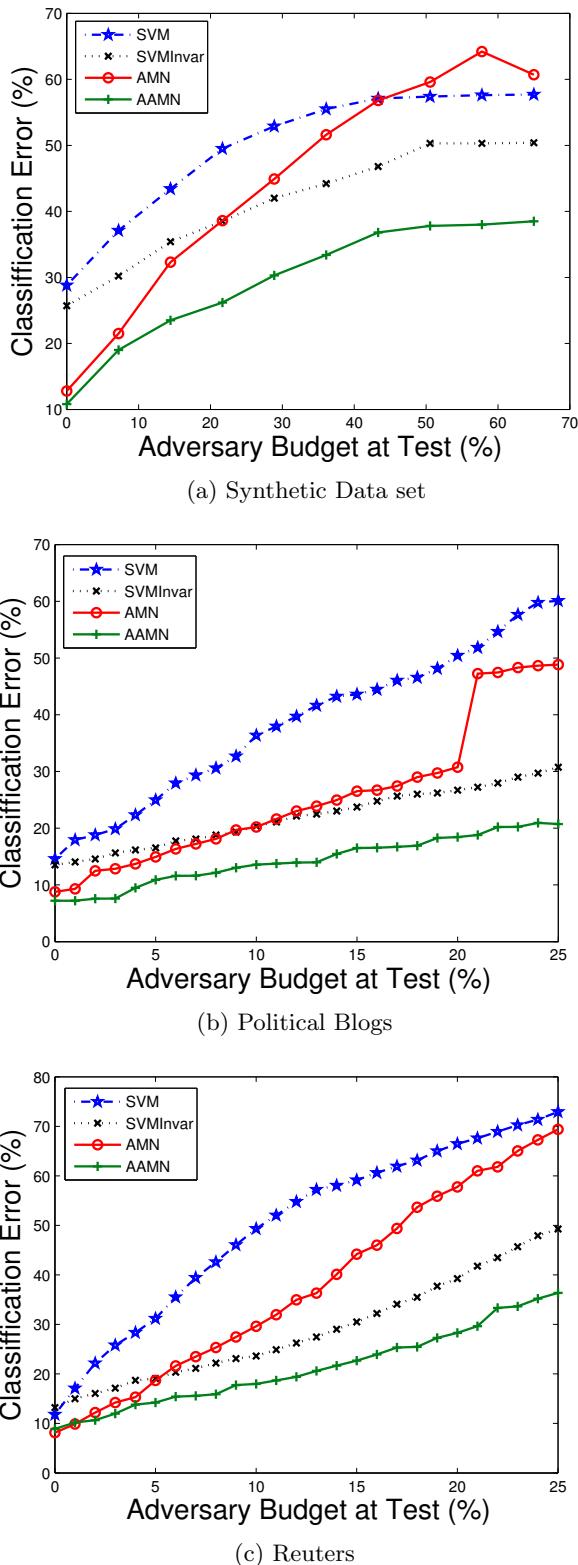


Figure 1: We clearly observe the trend that as adversary gets stronger the classification error of naive methods increases, also as it can be seen in 1b when there exist a concept drift the robust method outperforms other methods even if there exist no active adversary.

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

5.3 Results and Discussion

Figures 1a, 1c, and 1b show the performance of all four methods on test data against adversaries of varying strength. 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 an 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 data set.

We observe that up to some certain point AMN outperforms SVMInvar, but as the adversary is allowed to change more features, it will exploit the structure of the graph to even mislead the classifier even more. Therefore the AMN, performs very poorly in presence of an active adversary, while CACC is still robust against adversaries of any strength.

In Fig. 1b we observe as the adversaries budget transits from 20% to 21%, the AMN classification error jumps by 59%, and this is when the adversary has been to exploit the neighbors misclassified labels to convince the classifier to label more nodes incorrectly; in fact this is the point that in associative Markov networks, links not only are not helpful but also can be misleading as well. Such break points are different in different data sets, for example this effect can also be observed in synthetic data set when data adversary's budget at test time is approximately 30%. One of the nice properties of CACC is that it does not suffer from this effect and we observe that not only does it always outperform SVM and SVMInvar, but also it is robust to very strong adversaries and manipulation of the features at test time.

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. This is similar to what was observed by Taskar et al. [13].

6 Conclusion

In this paper we provide a generalization of SVMInvar [11] and AMN [13] which combines the robustness of SVMInvar with the ability to reason about interrelated objects. In experiments on real and synthetic data, we find that 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.

References

- [1] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, “Collective classification in network data,” *AI Magazine*, vol. 29, no. 3, p. 93, 2008.
- [2] P. Domingos and D. Lowd, *Markov Logic: An Interface Layer for AI*. San Rafael, CA: Morgan & Claypool, 2009.
- [3] J. Abernethy, O. Chapelle, and C. Castillo, “Graph regularization methods for web spam detection,” *Machine Learning*, vol. 81, no. 2, pp. 207–225, 2010.
- [4] I. Drost and T. Scheffer, “Thwarting the nigritude ultramarine: Learning to identify link spam,” in *Proceedings of the Sixteenth European Conference on Machine Learning*, pp. 96–107, Springer, 2005.
- [5] D. Chau, S. Pandit, and C. Faloutsos, “Detecting fraudulent personalities in networks of online auctioneers,” *Knowledge Discovery in Databases: PKDD 2006*, pp. 103–114, 2006.
- [6] N. Dalvi, P. Domingos, Mausam, S. Sanghavi, and D. Verma, “Adversarial classification,” in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (Seattle, WA), pp. 99–108, ACM Press, 2004.
- [7] D. Lowd and C. Meek, “Good word attacks on statistical spam filters,” in *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, pp. 125–132, 2005.
- [8] D. Lowd and C. Meek, “Adversarial learning,” in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 641–647, ACM, 2005.
- [9] B. Nelson, B. Rubinstein, L. Huang, A. Joseph, S. Lau, S. Lee, S. Rao, A. Tran, and J. Tygar, “Near-optimal evasion of convex-inducing classifiers,” in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010*, vol. 9, (Chia Laguna Resort, Sardinia, Italy), 2010.
- [10] A. Globerson and S. Roweis, “Nightmare at test time: robust learning by feature deletion,” in *Proceedings of the Twenty-Third International Conference on Machine Learning*, (Pittsburgh, PA), pp. 353–360, ACM Press, 2006.
- [11] C. Teo, A. Globerson, S. Roweis, and A. Smola, “Convex learning with invariances,” in *Advances in Neural Information Processing Systems 21*, 2008.
- [12] M. Brückner and T. Scheffer, “Nash equilibria of static prediction games,” in *Advances in Neural Information Processing Systems 22*, 2009.
- [13] B. Taskar, V. Chatalbashev, and D. Koller, “Learning associative Markov networks,” in *Proceedings of the twenty-first international conference on machine learning*, ACM Press, 2004.
- [14] V. Kolmogorov and R. Zabin, “What energy functions can be minimized via graph cuts?,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 2, pp. 147–159, 2004.
- [15] D. Lowd and P. Domingos, “Efficient weight learning for Markov logic networks,” in *Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, (Warsaw, Poland), pp. 200–211, Springer, 2007.
- [16] B. Taskar, M. F. Wong, P. Abbeel, and D. Koller, “Max-margin Markov networks,” in *Advances in Neural Information Processing Systems 16* (S. Thrun, L. Saul, and B. Schölkopf, eds.), Cambridge, MA: MIT Press, 2004.
- [17] T. Huynh and R. 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*, pp. 564–579, Springer, 2009.
- [18] P. Laskov and R. Lippmann, “Machine learning in adversarial environments,” *Machine learning*, vol. 81, no. 2, pp. 115–119, 2010.
- [19] M. Brückner and T. 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.
- [20] G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. Jordan, “Learning the kernel matrix with semidefinite programming,” *The Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.
- [21] L. El Ghaoui, G. Lanckriet, and G. Natsoulis, *Robust classification with interval data*. Computer Science Division, University of California, 2003.
- [22] S. Kim, A. Magnani, and S. Boyd, “Robust fisher discriminant analysis,” *Advances in Neural Information Processing Systems*, vol. 18, p. 659, 2006.
- [23] L. Adamic and N. Glance, “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.
- [24] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1226–1238, 2005.