# Adversarial Relational Learning with Antagonistic Costs

Ali Torkamani, University of Oregon
<ali@cs.uoregon.edu >

Daniel Lowd, 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 features for each example can be manipulated independently. Collective classification violates this assumption, since object labels depend on the attributes or labels of related objects as well as their own attributes. In this poster, we formulate adversarial collective classification as a game between a learner and an adversary in which the learner selects a relational classifier and the adversary selects a transformation of the data. We present an algorithm to find a Nash equilibrium in the special case of an antagonistic (zero-sum) game where the learner's reward is a regularized conditional log likelihood. We compare our approach experimentally to non-adversarial Markov logic networks and non-relational adversarial classifiers on simulated data.

## Introduction

- Many domains are both relational and adversarial:
  - Web spam
  - Social network spam
  - Online auction fraud
  - Terrorism
- Statistical relational learning:
  - Markov Logic Networks (MLNs)
  - Applications: Link Prediction [Richardson and Domingos, 2006], Entity Resolution [Singla and Domingos, 2006], Information Extraction [Poon and Domingos, 2007], and more…
  - Markov logic decision networks (MLDNs) [Nath & Domingos, 2009]
- Adversarial machine learning:
  - Construct a game: Learner selects a classifier and adversary selects a transformation.
  - Solve it: Minimax [Globerson and Roweis, 2007], Nash equilibrium [Brückner and Scheffer, 2009], Stackelberg equilibrium [Brückner and Scheffer, 2011]

## MLDN Formulation of Adversarial Collective Classification

Features: Features are the number of true groundings of each of the first order formulas in the MLN. Let $\phi_k^L$ be the feature corresponding to the learner's $k$th formula. In vector format, let $\phi^L = [\phi_1^L, \ldots, \phi_K^L]^T$. Analogously, let $\phi^A$ be the adversary's feature vector, constructed from the adversary's formulas.

PDF's: The joint probability density function of estimated values of x,y given the true values can be factorized as:

$$P(\dot{x}, \dot{y} | x, y) = P_{w^A}(\dot{x}|x,y) P_{w^L}(\dot{y}|\dot{x})$$

In which:

$$P_{w^L}(\dot{y}|\dot{x}) = \frac{1}{Z_{w^L}(\dot{x})} e^{w_L^T \phi^L(\dot{x},\dot{y})}$$

$$P_{w^A}(\dot{x}|x,y) = \frac{1}{Z_{w^A}(x,y)} e^{w_A^T \phi^A(x,y,\dot{x})}$$

According to the MLDN framework we'll need to maximize the utility of each player:

$$\theta_L(w_L) = E_{\dot{x},\dot{y}\sim P_{A,L}}\left[U^L(\dot{x},\dot{y},x,y)\right] - \lambda_L \frac{w_L^T w_L}{2}$$

$$\theta_A(w_A) = E_{\dot{x},\dot{y}\sim P_{A,L}}\left[U^A(\dot{x},\dot{y},x,y)\right] - \lambda_A \frac{w_A^T w_A}{2}$$

In which $U^L$ and $U^A$ are the learner and adversary's utility functions respectively. These functions can be arbitrary functions of the true $x$ and $y$ and $(\dot{x},\dot{y}) \sim P_{A,L}$, or can be a weighted sum of utility features, which may also be defined by first-order formulas.

Given the strategies and rewards of the learner and adversary, we can model their interaction as a game and search for a Nash or Stackelberg equilibrium. This representation of adversarial relational learning is very general, but also very difficult to solve. Thus, in the rest of this poster we address the simpler case where the adversary selects a single $\dot{x}$ instead of a distribution, and the utilities are an antagonistic (zero-sum) function that incorporates the label log-likelihood, the learner's weights, and the adversary's modifications $(\dot{x} - x)$ .

## Specific Example of Features

As an example, we can represent the important relationships in web spam using the following first order formulas:

```
Linked(pi, pj) ^ Spam(pi) => Spam(pj)
Linked(pi, pj)^ HasWord(pi,+w) => Spam(pj)
HasWord(pi, +w) => Spam(pi)
```

We can use these to construct an MLN, a log-linear model in which the features are the true counts of each formula.

## Antagonistic Regularized MLE Cost

We define the Antagonistic Regularized MLE Cost Optimization Problem as:

$$\begin{aligned}\theta(w,\dot{x}) &= \log P(y|\dot{x}) - \lambda_L \frac{w^T w}{2} + \lambda_A \frac{\mathrm{tr}((\dot{x}-x)^T(\dot{x}-x))}{2} \\ \textit{subject to} \quad & \dot{x} \le 1 \\ & \dot{x} \ge 0\end{aligned}$$

which the learner tries to maximize over *w* and the adversary tries to minimize over $\dot{x}$ . Since the program is not convex with respect to the adversary's action, the existence of a unique Nash equilibrium is not guaranteed.

## Greedy Algorithm for Finding a Nash Equilibrium

**Algorithm 1** Greedy Search to find Nash equilibrium Antagonistic Game

```
Require: $\theta(w,\dot{x}), x, \lambda_A, \lambda_L$.
  Initialize: $\dot{x}^k := x, w^0 \in \mathrm{R}^K$ and $k := 0, w_{acum} = 0$.
  repeat
    repeat
      Set $\Delta_L^k = \nabla_w \theta(w^k, \dot{x}^k)$
      Do a backtracking line search and find step size $a_L^{k+1}$.
      Set $w^{k+1} := w^k + a_L^{k+1}\Delta_L^k$
    until $\|w^{k+1} - w^k\|_2 \le \epsilon_L$

    Set $w_{acum} := \frac{k*w_{acum}+w}{k+1}$
    Set $w := w_{acum}$

    for $i = 1$ to nIter do
      Change $\dot{x}_{ij}$ where $ij$ is the index of the coordinate, that decreases the antagonistic cost more than all other coordinates. (Start from random initial points as well as true $x$ and also the previous adversary's best point locations)
    end for

    Set k:=k+1
  until $|cost^{k+1} - cost^k| \le \epsilon$
```

This algorithm is inspired by [Brückner and Scheffer, 2009], who use a similar approach to find Nash equilibria in non-relational domains. With our cost function and relational features, convergence is no longer guaranteed since the adversary's optimization is non-convex. We address this by averaging the learner's weights over all iterations so far, which damps oscillations.

## Parameter Estimation

The greedy algorithm needs the gradient of the cost function with respect to the learner's action:

$$\nabla_w \theta(w,\dot{x}) = \frac{\partial \theta(w,\dot{x})}{\partial w} = \phi(\dot{x},y) - E_{y\sim P_w}[\phi(\dot{x},y)] - \lambda_L w$$

Importance Sampling: For gradients and also for finding the normalization parameter of the pdf, we need to compute expectations and integrals over y. We approximate these expectations using importance sampling with 2500 samples from a uniform distribution over y. (When y has fewer than 2500 possible configurations, we perform exact inference instead.)

Regularization Parameters: In order to determine the regularization parameter of the learner, $\boldsymbol{\lambda}_L$, we use grid search and choose the value that gives the closest train and test costs.

The adversary's regularization weight is an attribute of the domain, and not something the learner can control. However, the game is uninteresting when $\boldsymbol{\lambda}_A$ is very large, since the adversary cannot change any features, or very small, since the adversary can change all features. We selected an intermediate value of $\boldsymbol{\lambda}_A$ that led to interesting behavior on our domain.

.

## Experiments

We conducted preliminary experiments in a simulated web spam domain. Model features were the same ones described above. We compared our method to baselines that were either non-adversarial, non-relational, or both.

**Methods:**
C – Classifier (logistic regression)
CC – Collective Classifier (MLN)
AC – Adversarial Classifier (Our method, ignoring links)
ACC – Adversarial Collective Classifier (Our method)

We evaluate models by log-likelihood in 8 different settings:
- Training and test graphs
- With and without adversarial manipulation of the data

For each of these settings we report the final log-likelihoods.

### Data Sets

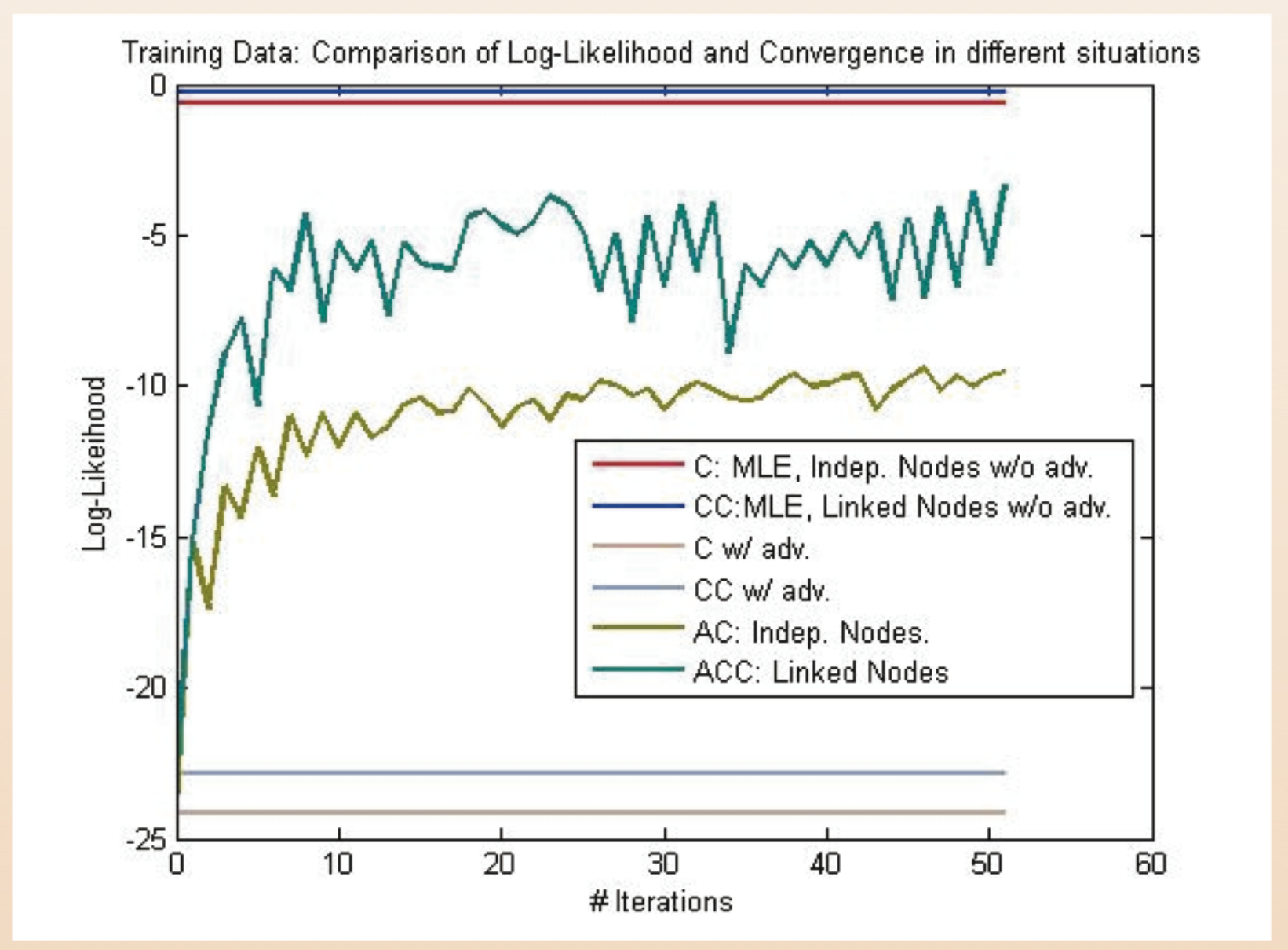
| | #Nodes | Avg #links: Spam-Spam | Avg #links: Spam-Non | Avg #links: Non-Spam | Avg #links: Non-Non | #Words |
|---|---|---|---|---|---|---|
| ToyDS1Linked | 10 | 2.2 | 0.6 | 0.2 | 0.8 | 30 |
| ToyDS1NotLinked | 10 | 0 | 0 | 0 | 0 | 30 |

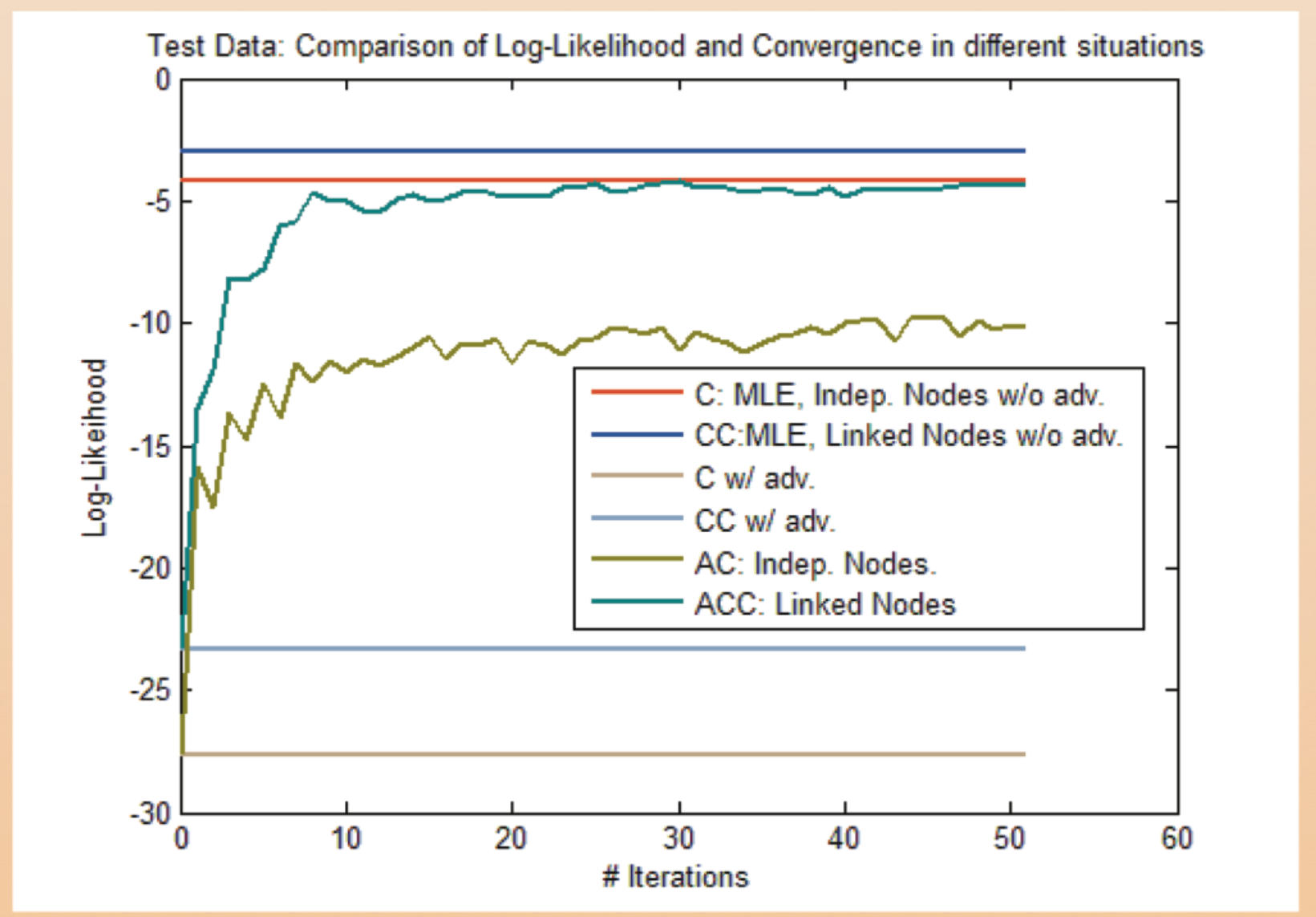
### Log-Likelihood in Different Settings

| | Training Network | | Test Network | |
|---|---|---|---|---|
| | w/o adv. | w/ adv. | w/o adv. | w/ adv. |
| C | -0.60068 | -24.1661 | -4.15538 | -27.6895 |
| CC | -0.22082 | -22.8646 | -3.00058 | -23.3053 |
| AC | -8.26708 | -9.47902 | -8.2748 | -10.1924 |
| ACC | -2.29453 | -3.367 | -4.11902 | -4.39254 |

ACC does much better than both AC and CC on both the training and test networks, demonstrating that both link information and adversarial modeling are necessary in order to obtain good results on this problem.

The existence of oscillations in our learning graphs suggests that we are not yet converging to an equilibrium, and may need to explore mixed strategies or other solution techniques in future work.





## Future Work and Conclusion

In this poster, we introduced adversarial relational learning and presented an algorithm to solve the antagonistic regularized log-likelihood case. Experiments on simulated data confirmed that both links and adversarial modeling are necessary to obtain good results. We also found that there were significant oscillations, even when we used weight averaging, suggesting that equilibria may exist that are better than the best weights we found.

**Ongoing/Future Work:**
- Mixed strategies for learner and adversary
- Non-antagonistic MLDN utility functions
- Scaling up
- Real-world datasets
- Irrational or improperly modeled adversaries

Goal: Robust method for making any MLN adversarial, given MLDNs or other utilities for learner and adversary.