Cost-Sensitive Learning with Conditional Markov Networks

Prithviraj Sen[*] Lise Getoor[*]

Abstract
Link analysis and social network analysis frequently require combining attribute-based and link-based information. Relational classifiers are a simple yet effective means that use attribute and link information for collective classification of nodes and/or links. However, in many cases, real world network classification tasks are accompanied by varying *relational* misclassification costs. We give a number of motivating examples, and propose the relational cost-sensitive learning problem. Our main contribution is to develop a novel relational cost-sensitive classifier which directly optimizes the relational misclassification costs. We compare our proposed relational cost-sensitive classifier to existing relational classifiers and show that it can help lower misclassification costs.
Keywords: Relational Classification, Link Analysis, Social Networks, Discriminative Models, Maximum Entropy Classifiers, Cost-Sensitive Learning.

1 Introduction

Social Network Analysis has long been an important field of research in the social sciences. Recent developments such as the proliferation of the online communities and communication networks has shown the need for scalable techniques for extracting, analyzing and mining large real-world social networks. These networks consist of *entities* linked by various *relations*. Predictive models which exploit both the attributes of entities and relations and their relational patterns are important for identifying key actors and important (or anomalous) links.

There has been a recent, growing interest amongst researchers in the machine learning community for classification and link prediction in relational domains. A host of methods like Conditional Random Fields (CRFs) [3] and Relational Markov Networks (RMNs) [1] have been introduced which are designed to enable users to build accurate classifiers for diverse graph datasets with a minimum of effort. In this paper, we develop extensions of relational classifiers to perform various tasks involving social networks and highlight their advantages over other methods.

Research in areas such as link analysis, social network analysis and text analysis frequently need to combine information from the attributes of entities and the information represented by relations to solve various problems such as identifying a set of nodes with some given common properties (e.g., identifying the terrorists in a given affiliation network). Discriminative relational classifiers are very well suited for such tasks. Moreover, the output of such classifiers usually leads us to take certain actions (e.g., classifying the entities into terrorists and non-terrorists may lead to arresting the terrorists). Incorrect classifications lead to undesirable actions with different consequences. If we are to build accurate classifiers that are to be applied to real-world networks then we need to take care of the varying *misclassification costs*. Most relational classifiers implicitly assume that all misclassifications are equally costly.

Within the machine learning community, there is a rich tradition of cost-sensitive learning [8, 7] applied to independent identically distributed data (non-relational data) which can handle varying misclassification costs. Our main contribution is to develop cost-sensitive relational classifiers which can handle varying misclassification costs.

We motivate the use of relational classifiers and the need for cost-sensitive relational classifiers using a series of examples. Our examples are modeled upon a specific type of social network known as a *communication network* ([4]) because they offer the appropriate amount of complexity that allows us to highlight the various aspects of cost-sensitive relational classification. However, none of the methods we discuss are specific to communication networks; they are equally applicable to any type of network data.

2 Link-based Classification

In the following subsections, we introduce three link-based classification problems, using communication networks to motivate each task.

2.1 Link-based Object Classification Communication networks are networks where nodes represent entities and edges between nodes represent some form of communication between entities ([4]). Different modes of communication (eg: email, phone call etc.) can be represented as different types of relations in the communication network but for simplicity we will consider networks with a single type of relation. We emphasize however that the approaches described in this paper can easily be applied in the case of multiple relation types in the network.

Traditionally, in machine learning, classification is the problem of predicting some unobserved, discrete valued attribute (referred to as the classification, class label or simply, label) given all the other attributes of the data. Consider the

[*]Department of Computer Science, University of Maryland, College Park, MD 20912.

problem of classifying entities in a communication network. Each entity may have some observed attributes associated with it describing characteristics of the entity. For example, in a corporate dataset network such as the Enron email corpus where the entities are the employees in the corporation, some examples of attributes could be the age, education and position of the employee which can be used to classify the employees as management or support.

Link-based object classification is the problem of predicting entity classifications based on both entity attributes and the classification of related entities. The relationships can represent possible correlations between classifications of connected entities and these correlations may be exploited to obtain correct classifications. As an example, suppose our task is to predict the roles of the entities in a communication network. It might be the case that entities with a certain role usually communicate with a restricted subset of entities having specific roles. For example, dentists usually communicate with patients and hygenists ([5]). Traditional machine learning has focussed on classifying independent and identically distributed (IID) entities solely on the basis of its observed attribute values. This completely ignores the relations in the network. Exploiting correlations among entity labels according to their relations has been shown to improve classification accuracies [26, 17, 1, 29].

In addition, the communications between entities can contain various attributes such as the words in an email. A good classifier should be able to exploit all three forms of evidence: 1) entity attribute values, 2) relations in the network and 3) the attribute values of the relations to achieve the correct classification. Relational classifiers are a simple yet powerful means to this end.

Link-based classification is an active area of machine learning research and many types of relational classifiers have been proposed. Here we provide a high-level introduction to relational classifiers based on *Markov networks* [1]; we provide a formal treatment and all the required definitions in Section 3. A *Markov network* consists of two parts: a qualitative part and a quantitative part. The qualitative part consists of a graph formed by nodes representing *random variables* each associated with a domain from which it can be assigned values. When classifying entities, the Markov network should contain a random variable for each entity that needs to be classified. These random variables are *unobserved* or *target* random variables and our problem is to determine the correct value that needs to be assigned to them from their respective domains. The Markov network represents attributes and their values by using *observed* random variables whose values are known and connecting them to the random variables corresponding to the entities. Besides the random variables, the Markov network also contains edges connecting the *target* random variables. These edges represent correlations and two random variables connected via an edge obey the correlation represented by the edge. If we believe that the links representing communications in communication networks represent correlations between the labels of the corresponding entities then we can connect the random variables of those entities with an edge in the Markov network. The quantitative part of the Markov network describes the correlations in the network and are stored in the form of *clique potentials* which we define in Section 3. A Markov network defines a joint probability distribution and the optimum joint labeling for all the *target* random variables in the graph is obtained by maximizing the distribution.

2.2 Link-based Edge Classification Link-based classification has been used in various domains to classify various types of entities. The importance of relations is raised to another level when one is interested in classifying, not the entities but, the relations themselves. For example, consider classifying communication links in a communication network into links which require surveillance and links which do not. We next describe how the problem of link-based *edge* classification can be described in terms of a Markov network.

As part of the ongoing efforts to reduce terrorism, a considerable amount of communication (e.g., email, phone calls etc.) is analyzed. Given a communication network, one needs to determine the key links in the network which require monitoring since it might be infeasible to monitor all communication links for a sustained period of time. For example, after the London bombings ([6]) a considerable amount of security was dedicated to prevent any such occurrence in major cities of the USA. In such situations, one way to improve surveillance would be to identify the communication links which had been discussing the relevant topics ("subway", "suicide", "bombs" etc.) and devote more resources to keep tabs on them. Note that even though we are still dealing with a communication network, we are no longer interested in classifying the entities themselves. We are more interested in classifying the links in the communication networks to determine whether they are suspicious or unsuspicious.

To classify links in a communication network we need to construct a Markov network which contains *target* random variables corresponding to each communication link. Just like the problem of classifying entities in a communication network, we have all three forms of evidence which require combining:

- Every communication link may have certain observed attributes (eg: words in a telephone conversation etc.) and these can be represented by *observed* random variables and connecting them to the *target* random variable representing the communication link they belong to.

- Each entity in the communication network may have one or more communication link emanating from it and the labels on communication links which emanate from the same entity might be correlated. For example, a terrorist who is discussing how to make a bomb with terrorist *A* is probably going to discuss where to plant the bomb with some other terrorist *B* and thus both these communication links are suspicious. One way to represent such correlations is to introduce an edge between two *target* random variables if and only if the corresponding communication links share an entity in common. Figure 1 shows an example of a communication network and its corresponding Markov network to facilitate communication link classification.

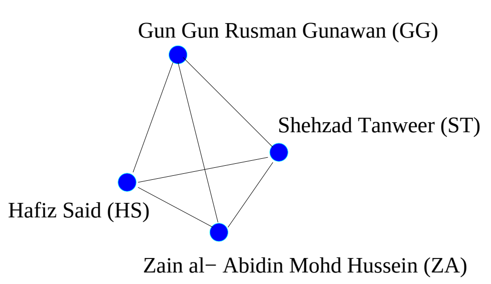
- The third form of evidence is the attribute values belonging to the entities themselves and these attribute values may help us classify each and every communication link emanating from the entity. Once again we can represent these attribute values using *observed* random variables and connecting them to each and every *target* random variable representing all the communication links which emanate from the entity.

2.3 Cost-sensitive link-based classification As mentioned in the introduction, in many cases, the output of classification is used to take certain actions. Incorrect outputs will lead to suboptimal actions which may lead to undesirable consequences. To build an accurate classifier which can be applied to real-world social networks one needs to take into account the varying costs associated with each misclassification. Unfortunately, most relational classifiers assume that all misclassifications are equally costly and are thus referred to as 0/1 loss relational classifiers. Traditional cost-sensitive learning [8, 7] has worked on the problem of classifying IID data with different misclassification costs for various domains like targeted marketing, fraud and intrusion detection etc. Next we motivate the need for *cost-sensitive relational classifiers* which can handle varying misclassification costs in the context of classification in social networks.

Consider the earlier problem of classifying communication links in a communication network comprising of terrorists. Suppose we want to classify each link into one of "suspicious" (links which may be discussing terrorist activities associated with the bombing of a subway station) and "unsuspicious" (links which discuss harmless topics). Consider a particular communication link l in the communication network. l can be misclassified in two ways: first labeling l as "suspicious" when it is not and second, labeling l as being "unsuspicious" when it actually is. The first type of misclassification will cause us to devote resources like manpower, wire-taps etc. to monitor l even though this surveillance is uncalled for. The second type of misclassification is more serious and might cause the terrorist activity (like the bombing of a subway station) to succeed resulting in high costs like damage to life and property. Note that the two misclassification costs can be measured in common units (eg: monetory units) and may differ considerably. Such varied costs associated with different misclassifications is the hallmark of cost-sensitive learning.

A novel aspect of *relational cost-sensitive learning* is the presence of *relational misclassification costs*. In other words, besides the misclassification costs associated with each misclassification, we now have *misclassification costs associated with misclassifying groups of related target random variables*. Consider an entity e in the communication network. The result of misclassifying **any** of the communication links emanating from e as being "suspicious" when in fact it is not might prompt e to move to court for invasion of privacy and claim compensation resulting in a misclassification cost. This misclassification cost is associated with **all** of the target random variables representing the communication links emanating from e and is in fact an instance of a *relational misclassification cost* associated with a group of related random variables. *Relational costs* can be modeled as a cost matrix $\mathrm{Cost}_c(y_c, \tilde{y}_c)$ which specifies the cost incurred when a set of related random variables denoted by *clique* c whose correct set of labels is $\tilde{y}_c$ is labeled with the set y_c. In the above case c denotes a set of random variables corresponding to the communication links emanating from a single entity in the communication network, $\tilde{y}_c$ denotes the set of correct labels corresponding to those communication links and y_c denotes the set of labels predicted by our classifier.

In this paper we consider the problem of link-based classification in the presence of unequal misclassification costs. In fact, we are not aware of any other work which considers cost-sensitive classification of relational data. Note that 0/1 loss relational classifiers are a special case of cost-sensitive classifiers obtained by setting all misclassifications to be equally costly. We devote the rest of the paper to the development of the more general cost-sensitive relational classifier. In Section 3 we begin with some preliminary definitions and notation. In Section 4 we describe how traditional 0/1 loss relational classifiers based on Markov networks can be used to perform relational cost-sensitive classification. The drawback of using a 0/1 loss relational classifier to perform cost-sensitive classification is the requirement to estimate accurate class conditional probabilities. In Section 5 we propose a cost-sensitive relational classifier which does not need to estimate class conditional probabilities. In Section 6 we compare the performance of various cost-sensitive classifiers on synthetic data. In Section 7 we describe some of the related work in this area and finally, conclude with a discussion in Section 8.


Gun Gun Rusman Gunawan (GG)
Shehzad Tanweer (ST)
Hafiz Said (HS)
Zain al− Abidin Mohd Hussein (ZA)

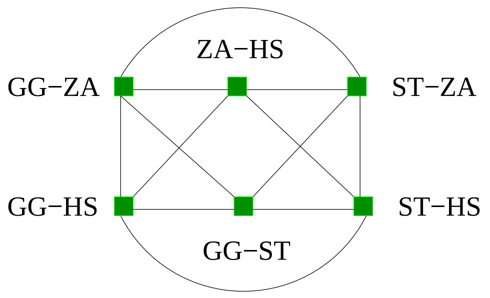


Figure 1: An example showing the conversion from a communication network on the left ([9]) to a Markov network on the right where each random variable represents a communication link (blue circles represent entities in the communication network and green squares represent random variables).

3 Preliminaries

We review the definitions of conditional Markov networks from Taskar et al [1]. Let $\mathbf{V}$ be a set of discrete random variables, and let $\mathbf{v}$ be an assignment of values to the random variables. A Markov network is described by a graph $G = (\mathbf{V}, E)$ and a set of parameters Ψ. Let $C(G)$ denote a set of (not necessarily maximal) cliques in G. For each $c \in C(G)$, let V_c denote the nodes in the clique. Each clique c has a clique potential $\psi_c(V_c)$ which is a non-negative function on the joint domain of V_c and let $\Psi = \{\psi_c(V_c)\}_{c \in C(G)}$. For classification problems we are often interested in conditional models. Let $\mathbf{X}$ be the set of observed random variables we condition on, let $\mathbf{x}$ denote the observed values of $\mathbf{X}$ and let X_c denote the observed random variables in clique $c \in C(G)$. Let $\mathbf{Y}$ be the set of target random variables we want to assign labels to, let $\mathbf{y}$ denote an assignment to $\mathbf{Y}$ and let Y_c denote the set of target random variables in clique $c \in C(G)$. A *conditional Markov network* is a Markov network (G, Ψ) which defines the distribution $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C(G)} \psi_c(\mathbf{x}, y_c)$ where $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_c \psi_c(\mathbf{x}, y'_c)$.

For the cost sensitive version of this problem, in addition to (G, Ψ) as in ordinary Markov Networks, we also have a *cost graph* $H = (\mathbf{V}, E')$ which is defined over the same set of random variables $\mathbf{V}$ but has a different edge set E'. Let $C(H)$ denote the set of (not necessarily maximal) cliques in H. Let Y_h denote the set of target random variables present in clique $h \in C(H)$ and let y_h denote an assignment to Y_h. For each clique $h \in C(H)$ there is a clique loss function $l_h(y_h, \tilde{y}_h)$. l_h is determined by the cost matrices $\{\text{Cost}_h(y_h, \tilde{y}_h)\}_{h \in C(H)}$ involved in the problem and it is not necessary that they be the same. $\text{Cost}_h(y_h, \tilde{y}_h)$ is a measure of how severe the misclassification is if Y_h is labeled with y_h when its correct labels are $\tilde{y}_h$.

The misclassification cost of a complete assignment $\mathbf{y}$ relative to the correct assignment $\tilde{\mathbf{y}}$ is:

$$\text{Cost}(\mathbf{y}, \tilde{\mathbf{y}}) = \sum_{h \in C(H)} \text{Cost}(y_h, \tilde{y}_h).$$

Our aim is to determine $\mathbf{y}$ which corresponds to the minimum misclassification cost. In this paper, we consider the special case where $C(G) = C(H)$.

4 Cost Sensitive classification with Conditional Markov Networks

One approach to perform cost-sensitive classification is to use a classifier which can output conditional probabilities associated with each possible complete assignment to $\mathbf{Y}$. We can use these probabilities to compute the complete assignment $\mathbf{y}$ which minimizes the expected cost of misclassification:

$$\text{argmin}_{\mathbf{y}} \sum_{\mathbf{y}'} P(\mathbf{y}' \mid \mathbf{x}) \text{Cost}(\mathbf{y}, \mathbf{y}')$$

Note that the set of conditional probabilities required in the above equation can be quite large, so large that no classifier might want to list them out. It is more useful to express the problem in terms of the marginals:

$$\text{argmin}_{\mathbf{y}} \sum_{h \in C(H), y'_h} \text{Cost}(y_h, y'_h) \mu_h(y'_h \mid \mathbf{x})$$

where $\mu_h(y'_h \mid \mathbf{x}) = \sum_{\mathbf{y}' \sim y'_h} P(\mathbf{y}' \mid \mathbf{x})$ and $\mathbf{y}' \sim y'_h$ denotes a full assignment $\mathbf{y}'$ consistent with partial assignment y'_h. Any energy minimization technique can be used to perform this optimization.

5 Cost Sensitive Markov Networks

In this section we outline the design of a classifier function which computes the complete assignment $\mathbf{y}$ given the graph $G = (\mathbf{V}, E)$, the nodes we condition on $\mathbf{X}$ and the clique loss matrices $\{l_c(y'_c, y_c)\}_{c \in C(G)}$. We first derive the form of the classifier function from maximum entropy principles. The basic idea is to modify the constraints of the maximum entropy framework so that an assignment with higher loss is assigned a correspondingly lower probability. The traditional maximum entropy constraints can be expressed as:

$$\sum_{\mathbf{y}} f_k(\mathbf{x},\mathbf{y})P(\mathbf{y} \mid \mathbf{x}) = A_k,\ \forall k = 1,\ldots,K$$

where f_k is the k^{th} feature and we employ K such features.

We assume that the features distribute over the cliques and thus $f_k(\mathbf{x},\mathbf{y}) = \sum_c f_k(\mathbf{x}, y_c)$. Also we assume that the constants $\{A_k\}_{1,\ldots,K}$ come from counting the features of the fully labeled training data set labeled $\tilde{\mathbf{y}}$ and so, $A_k = \sum_c f_k(\mathbf{x}, \tilde{y}_c)$. With these assumptions the above equation can be rewritten as:

$$\sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \sum_c \underbrace{(f_k(\mathbf{x}, y_c) - f_k(\mathbf{x}, \tilde{y}_c))}_{\text{clique specific penalty term}} = 0, \quad \forall k = 1,\ldots,K$$

Our basic idea is to modify the clique specific penalty term by scaling it with the loss incurred by the misclassification of the clique.

$$\sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \sum_c \underbrace{l_c(y_c, \tilde{y}_c)\,(f_k(\mathbf{x}, y_c) - f_k(\mathbf{x}, \tilde{y}_c))}_{\text{scaled clique specific penalty term}} = 0, \quad \forall k = 1,\ldots,K$$

The new maximum entropy formulation can now be expressed as:

$$\begin{aligned}
&\max \sum_{\mathbf{y}} -P(\mathbf{y}) \log P(\mathbf{y}) \\
&\text{subject to:} \\
&\sum_{y} P(\mathbf{y}) = 1 \\
&\sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \sum_c l_c(y_c, \tilde{y}_c)\,(f_k(\mathbf{x}, y_c) - f_k(\mathbf{x}, \tilde{y}_c)) = 0, \\
&\qquad \forall k = 1,\ldots,K
\end{aligned}$$

The lagrangian formulation is thus:

$$\begin{aligned}
L = &-\sum_{\mathbf{y}} P(\mathbf{y}) \log P(\mathbf{y}) - \mu \left(\sum_{y} P(\mathbf{y}) - 1\right) \\
&- \sum_k w_k \left(\sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \right. \\
&\left. \sum_c l_c(y_c, \tilde{y}_c)[f_k(\mathbf{x}, y_c) - f_k(\mathbf{x}, \tilde{y}_c)]\right)
\end{aligned}$$

where μ and $w_k\ \forall k = 1 \ldots K$ are lagrangian multipliers.

Differentiating with respect to $P(\mathbf{y})$:

$$\begin{aligned}
\frac{\partial L}{\partial P(\mathbf{y})} = &\\
&- \log P(\mathbf{y}) - 1 - \mu \\
&- \sum_k w_k \sum_c l_c(y_c, \tilde{y}_c)[f_k(\mathbf{x}, y_c) - f_k(\mathbf{x}, \tilde{y}_c)]
\end{aligned}$$

Setting the derivative to 0 we obtain:

$$P(\mathbf{y}) = \frac{1}{Z} \exp\left[-\sum_k w_k \sum_c l_c(y_c, \tilde{y}_c)(f_k(\mathbf{x}, y_c) - f_k(\mathbf{x}, \tilde{y}_c))\right]$$

where

$$Z = \sum_{\mathbf{y}'} \exp\left[-\sum_k w_k \sum_c l_c(y'_c, \tilde{y}_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c))\right]$$

Substituting the expression for $P(\mathbf{y})$ back in the lagrangian L gives us:

$$L = \log Z$$

Thus the dual of our maximum entropy formulation is:

$$\min \sum_{\mathbf{y}'} \exp\left(\sum_k w_k \sum_c l_c(y'_c, \tilde{y}_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c))\right) \tag{5.1}$$

where $\{w_k\}_1^K$ are the parameters of the classifier. Eq. (5.1) is the basic form of our classifier. Note that this classifier is not a log-linear classifier. Thus the standard methods of inference and learning don't apply. We next describe learning and inference algorithms for our classifier.

5.1 Learning. Given fully labeled training data we can learn the model by solving the following optimization problem:

$$\operatorname{argmin}_w \sum_{\mathbf{y}'} \exp\left(\sum_k w_k \sum_c l_c(y'_c, \tilde{y}_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c))\right)$$

where $\tilde{y}$ is the complete assignment of the labeled training data. Note that this problem is convex for fully labeled training data.

Differentiating with respect to w_k:

$$l = \log\left[\sum_{\mathbf{y}'} \exp\left(\sum_k w_k \sum_c l_c(y'_c, \tilde{y}_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c))\right)\right]$$

$$\frac{\partial l}{\partial w_k} = \sum_{\mathbf{y}'}\left[\frac{1}{Z}\exp\left(\sum_k w_k \sum_c l_c(y'_c, \tilde{y}_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c))\right) \sum_c l_c(y'_c, \tilde{y}_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c))\right]$$

Let us now define the following probability distribution (which is, of course, normalized):

$$q(\mathbf{y}') \propto \exp\left(\sum_k w_k \sum_c l_c(y'_c, \tilde{y}_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c))\right)$$

$\frac{\partial l}{\partial w_k}$ can be expressed in terms of $q(\mathbf{y}')$ as:

$$\begin{aligned}\frac{\partial l}{\partial w_k} &= \sum_{\mathbf{y}'} q(\mathbf{y}') \sum_c l_c(y'_c, \tilde{y}_c)[f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c)] \\ &= \sum_{c, y'_c} \mu^q_c(y'_c) l_c(y'_c, \tilde{y}_c)[f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c)]\end{aligned}$$

where $\mu^q_c(y'_c)$ is the marginal probability of labeling clique c with y'_c under the q distribution. So if we can compute the marginals then we can compute the gradient without having to sum up for each possible complete assignment $\mathbf{y}'$.

One way to compute the marginals is to run loopy belief propagation [2] (for pairwise markov networks) or its extensions (for markov networks with larger cliques) for the q distribution by defining the following clique potentials:

$$\psi^q_c(y'_c) = \exp\left[\sum_k w_k l_c(y'_c, \tilde{y}_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, \tilde{y}_c))\right]$$

Having computed the gradient with respect to the weights $\{w_k\}_{1,\ldots K}$ we can use any gradient based optimization method (like conjugate gradient descent) to perform the learning.

5.2 Inference. The inference problem is to compute:

(5.2)
$$\text{argmin}_{\mathbf{y}} \sum_{\mathbf{y}'} \exp\left(\sum_k w_k \sum_c l_c(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c))\right)$$

Unless the underlying Markov network has special properties (e.g., being a tree, a sequence or a network with a low treewidth) exact inference may be infeasible. For the domains described in Section 1, the Markov network might consist not only of thousands of nodes but may also be densely connected. In such cases we resort to approximate inference.

In order to obtain a lower bound approximation we take log of Eq. (5.2) and apply Jensen's inequality to get:

(5.3)
$$\begin{aligned}&\log\left[\sum_{\mathbf{y}'} \exp\left(\sum_k w_k \sum_c l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c))\right)\right] \\ &\quad \geq \sum_{\mathbf{y}'} q(\mathbf{y}') \sum_k w_k \sum_c l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c)) \\ &\quad\quad - \sum_{\mathbf{y}'} q(\mathbf{y}') \log q(\mathbf{y}')\end{aligned}$$

where $q(\mathbf{y}')$ is a distribution over all $\mathbf{y}'$ ($\sum_{\mathbf{y}'} q(\mathbf{y}') = 1$, $q(\mathbf{y}') \geq 0$).

To find the optimal complete assignment $\mathbf{y}$ we will employ a 2-step iterative procedure. In each iteration, first, we will obtain the best approximation by maximizing the right hand side in Eq. (5.3) w.r.t $q(\mathbf{y}')$ and, second, we will minimize w.r.t. $\mathbf{y}$. We will keep iterating between these two steps until our objective function stabilizes.

The Lagrangian of the RHS in Eq. (5.3) is:

$$\begin{aligned}l = &\sum_{\mathbf{y}'} q(\mathbf{y}') \sum_k w_k \sum_c l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c)) \\ &- \sum_{\mathbf{y}'} q(\mathbf{y}') \log q(\mathbf{y}') - \mu\left(\sum_{\mathbf{y}'} q(\mathbf{y}') - 1\right)\end{aligned}$$

Differentiating with respect to $q(\mathbf{y}')$:

$$\begin{aligned}\frac{\partial l}{\partial q(\mathbf{y}')} = &\sum_k w_k \sum_c l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c)) \\ &-1 - \log q(\mathbf{y}') - \mu\end{aligned}$$

Setting the derivative to 0:

$$q(\mathbf{y}') \propto \exp\left[\sum_k w_k \sum_c l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c))\right]$$

where $\sum_{\mathbf{y}'} q(\mathbf{y}') = 1$.
Let

$$Z'(\mathbf{y}) = \sum_{\mathbf{y}''} \exp\left[\sum_k w_k \sum_c l(y''_c, y_c)(f_k(\mathbf{x}, y''_c) - f_k(\mathbf{x}, y_c))\right]$$

Thus:

$$q(\mathbf{y}') = \frac{1}{Z'(\mathbf{y})} \exp\left[\sum_k w_k \sum_c l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c))\right]$$

Note that computing $q(\mathbf{y}')$ for every complete assignment $\mathbf{y}'$ is not feasible because the set of all complete assignments could be very large. We need to see if we can compute the optimal complete assignment without explicitly computing all $q(\mathbf{y}')$.

The second stage of the optimization requires minimizing with respect to $\mathbf{y}$. Thus we only need to look at the first term in the lagrangian (since this is the only term which involves $\mathbf{y}$):

$$\sum_{\mathbf{y}'} q(\mathbf{y}') \sum_k w_k \sum_c l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c)) =$$

$$\sum_{y'_c, c} \sum_k w_k l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c))\mu^q_c(y'_c) \tag{5.4}$$

where $\mu^q_c(y'_c)$ is the marginal probability of labeling clique c with y'_c under the q distribution. Thus we only need the marginal probabilities to perform the second stage of the optimization. We can perform the second stage of the optimization by using the marginals of the $q(\mathbf{y}')$ distribution and determining the $\mathbf{y}$ which maximizes Eq. (5.4). This becomes our new best guess of the solution. We can iterate between the two steps until our guess for the optimal complete assignment stabilizes.

One way to compute the marginal probabilities under the $q(\mathbf{y}')$ distribution is to run loopy belief propagation [2] (for pairwise markov networks) or one of its extensions (for markov networks with larger cliques) with the following clique potential:

$$\psi^q_c(y'_c) = \exp\left[\sum_k w_k l_c(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c))\right]$$

where $\mathbf{y}$ is the current guess of the optimal complete assignment.

One way to maximize Eq. (5.4) is to define the following distribution for $\mathbf{y}$:

$$r(\mathbf{y}) \propto \exp\left[\sum_{y'_c, c} \sum_k w_k l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c))\mu^q_c(y'_c)\right]$$

where $\sum_{\mathbf{y}} r(\mathbf{y}) = 1$. Thus the $\mathbf{y}$ which maximizes Eq. (5.4) corresponds to the most optimal $\mathbf{y}$ under the $r(\mathbf{y})$ distribution. To compute the optimal $\mathbf{y}$ under the $r(\mathbf{y})$ distribution we can run loopy belief propagation [2] or or one of its extensions with the following clique potentials:

$$\psi^r_c(y_c) = \sum_{y'_c} \sum_k w_k l(y'_c, y_c)(f_k(\mathbf{x}, y'_c) - f_k(\mathbf{x}, y_c))\mu^q_c(y'_c)$$

and choose the $\mathbf{y}$ with the highest marginal probabilities.

6 Experiments

We performed experiments on synthetic random graph data with misclassification costs. Commonly available real world networks exhibit properties like preferential attachment and correlations amongst the labels across links. Since our aim is to find out how relational classifiers will perform on such networks we chose to model our synthetic data generating algorithm according to the evolutionary network model described in Bollobas et al [27] which can give rise to power-law graphs. Further, we were also interested in varying graph characteristics like the strength of the correlation amongst labels across links and link density in the network to find out how these variations affect classifier performance.

In all our experiments we compare misclassification costs achieved by different classifiers. The first model we considered is a non-relational model which only looks at the attribute values in the data and builds many one-against-the-rest logistic regression classifiers ([28]) to predict the probability of the node belonging to each class choosing the one with the highest probability (*LOGREG*). The second model we report results for is a discriminative relational classifier based on conditional markov networks which minimizes the expected cost of misclassification as described in Section 4 (*MN*). The third and final model we report results for is the model described in Section 5 (*CSMN*).

For simplicity we consider cliques of maximum size 2 in all our experiments. For each classifier, we assumed a "shrinkage" prior and compute the MAP estimate of the parameters. More precisely, for *LOGREG* we assumed that different parameters are a priori independent and define $p(w_i) = \lambda w_i^2$. After trying out a range of regularization constants we found that $\lambda = 0.03$ gave the best result. For *MN*, we assumed that different parameters are a priori independent and define $p(w_i) = \lambda w_i^2$. After trying a range of regularization constants we found that $\lambda = 10$ returned the best results. For *CSMN*, we assumed that different parameters are a priori independent and define $p(w_i) = \lambda w_i^2$. We tried regularization constant in the range [1,200] and found that $\lambda = 20$ returned the best results.

For all runs of *CSMN*, we set the clique loss matrices equal to the cost matrices:

$$l_c(y_c, \tilde{y}_c) = \mathrm{Cost}_c(y_c, \tilde{y}_c)$$

6.1 Synthetic data generation. Our algorithm for generating synthetic datasets closely follows the algorithm described in Bollobas et al [27]. For all our experiments we generated binary class random graph data since this is what is commonly encountered in most cost-sensitive applications (terrorist/non-terrorist, good-customer/bad-customer etc.). The algorithm is outlined in Algorithm 6.1.

ALGORITHM 6.1. (SYNTHETIC GRAPH DATA GENERATION)
SynthGraph($numNodes$, α, ρ, $vocabSize$, $numObs$, $attrNoise$)

```
1: Set i=0
2: $G = \emptyset$
3: **while** $i < numNodes$ **do**
4:     Sample $r \in [0,1]$ uniformly at random
5:     **if** $r <= \alpha$ **then**
6:         connectNode($G$, $\rho$)
7:     **else**
8:         addNode($G$, $\rho$)
9:         $i \leftarrow i + 1$
10:    **end if**
11: **end while**
12: **for** i = 1 to numNodes **do**
13:     $v \leftarrow i^{th}$ node in $G$
14:     genAttributes($v$, $vocabSize$, $numObs$, $attrNoise$)
15:     genNodeCostMatrix($v$)
16: **end for**
17: **for** each edge $e$ in $G$ **do**
18:     genEdgeCostMatrix($e$)
19: **end for**
20: return $G$
```

ALGORITHM 6.2. (ADDING AN EDGE TO THE GRAPH)
connectNodes(G, ρ)

```
1: $v \leftarrow$ select any existing node uniformly at random from $G$
2: sample $r$ uniformly at random from $[0,1]$
3: **if** $r \le \rho$ **then**
4:     $c_n \leftarrow v.label$
5: **else**
6:     $c_n \leftarrow (v.label + 1)$ mod $2$
7: **end if**
8: $w \leftarrow$ select a node from $G$ with $w.label = c_n$ and probability of selection proportional to its out-degree
9: introduce an edge from $v$ to $w$
```

ALGORITHM 6.3. (ADDING A NODE TO THE GRAPH)
addNode(G, ρ)

```
1: add a new node $v$ to $G$
2: choose $v.label$ from $\{0,1\}$ uniformly at random
3: sample $r$ uniformly at random from $[0,1]$
4: **if** $r \le \rho$ **then**
5:     $c_n \leftarrow v.label$
6: **else**
7:     $c_n \leftarrow (v.label + 1)$ mod $2$
8: **end if**
9: $w \leftarrow$ select a node from $G$ with $w.label = c_n$ and probability of selection proportional to its out-degree
10: introduce an edge from $v$ to $w$
```

The algorithm "grows" a graph from an empty set of nodes. The number of nodes in the final graph is controlled by the parameter $numNodes$. α is a parameter which controls the number of links in the graph. Roughly, the final graph should contain $\frac{1}{1-\alpha} numNodes$ number of links. As mentioned before, we experimented with binary class data. We used a uniform set of class priors.

The algorithm implements a rudimentary form of *preferential attachment* where a node can choose the label of the node it wants to link to. This introduces correlations amongst the labels across links. The degree of these correlations is controlled by the parameter ρ. Each node can link to nodes of its own class with probability ρ. With probability $1 - \rho$ a node can choose a node of the other class to link to. If ρ is close to 1 then the generated dataset will primarily feature links between nodes with the same class labels, whereas, if ρ is close to 0.5 then the generator will produce datasets where nodes have equal chance of linking to nodes of their own class or nodes of the other class. A further aspect of our synthetic data generator's preferential attachment is that nodes with higher out-degree have a a higher chance of getting linked to. This introduces the power-law degree distribution commonly observed in most real world networks. We refer the interested reader to Bollobas et al [27] for more details regarding this aspect of our synthetic data generation algorithm.

After generating the graph, we generate attributes for each node (`genAttributes`). For every node, an attribute can either be present or absent. The total number of attributes is controlled by the parameter $vocabSize$. For each node we sample the ids of the attributes which are present from a noisy class-specific binomial distribution. The noise in the attributes is controlled by the parameter $attrNoise$. With probability $attrNoise$ we sample an attribute id uniformly from the set $\{0, \ldots, vocabSize - 1\}$. With probability $1 - attrNoise$, we sample an attribute id from the distribution

name	value
$numNodes$ for training set	300
$numNodes$ for test set	300
$numLabels$	2
$vocabSize$	5
$attrNoise$	0.3
$numObs$	4

Table 1: Parameter settings for our synthetic data generator

$Binomial(p = 1/3, vocabSize)$ if the node belongs to class 0 and $Binomial(p = 2/3, vocabSize)$ if the node belongs to class 1. For each node we sample $numObs$ such attribute ids. For our experiments we set $vocabSize = 5$, $attrNoise = 0.3$ and $numObs = 4$. Intuitively, nodes with class label 0 will have present attributes with ids between 0 and 3 whereas nodes with class label 1 will have present attributes with ids between 1 and 4 with some noise introduced in the process.

Finally, we generate the cost matrices for the data (`genNodeCostMatrix` and `genEdgeCostMatrix`). For simplicity, we considered cliques only upto size 2 (nodes and edges) and thus we needed to generate only two types of cost matrices: one for the nodes and one for the edges. For the node cost matrices $\text{Cost}(y, \tilde{y})$ we set the diagonal entries to 0 and we sampled the off-diagonal entries uniformly from $[0, 2]$. For the edge cost matrices $\text{Cost}(y_c, \tilde{y}_c)$, we set the diagonal entries to 0 and sampled the off-diagonal elements uniformly from $[0, \frac{\text{ham}(y_c, \tilde{y}_c)}{2}]$ where $\text{ham}(y_c, \tilde{y}_c)$ denotes the hamming distance between y_c and $\tilde{y}_c$.

Table 1 describes our parameter settings. For each of our experiments we produced one training and three testing datasets. Each number we report is the misclassifi cation cost averaged over three test sets. All graph datasets were disjoint from each other.

6.2 Performance Comparisons. For our fi rst experiment we varied the correlation amongst labels across links to fi nd out if relational classifi ers actually help decrease misclassifi - cation costs. We varied the value of ρ from 0.5 to 1.0 while keeping α constant at 0.3. Recall that ρ controls the chance of a node with label c linking to another node with label c. Setting $\rho = 1$ will cause nodes with label c to exclusively link with other nodes of label c whereas setting $\rho = 0.5$ will cause nodes with label c to randomly choose nodes to link to irrespective of their class labels.

Figure 2 shows that when $\rho = 0.5$ (no correlations) all three classifi ers tend to perform equally, once we begin to introduce correlations in the link structure the two relational classifi ers begin to show some savings in misclassifi cation costs over the non-relational method but this doesn't happen

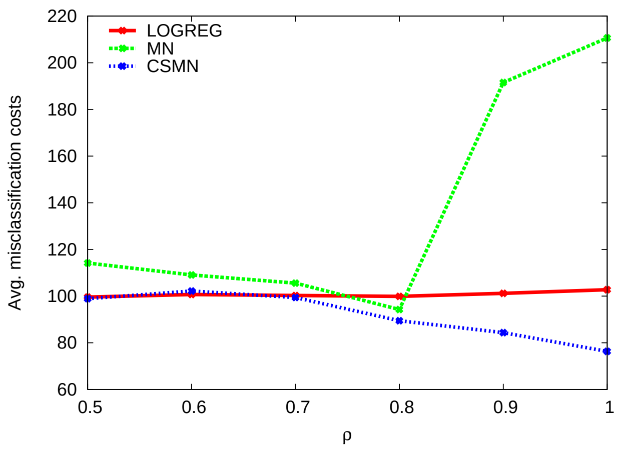


Figure 2: Avg. misclassifi cation costs attained by varying ρ (X-axis). α was kept constant at 0.3.

	LOGREG	*MN*	*CSMN*
Training	0.27	48.41	10.81
Testing	0.066	0.126	0.11

Table 2: Average run times (seconds) for the various classifi ers with α kept constant at 0.3 and ρ kept constant at 0.7. All runs were done on a 2.4 GHz Xeon.

until ρ reaches a value of 0.7. The plot shows that *CSMN* manages to produce lower misclassifi cation costs that *MN* on all settings of ρ. *CSMN* achieves 10.6% reduction in costs over *LOGREG* at $\rho = 0.8$ which increases to 24.6% at $\rho = 1.0$. Note that the 0/1 loss *MN* classifi er achieves an avg. classifi cation accuracy of 79.76% at $\rho = 0.5$ and this improves to 80.83% at $\rho = 0.7$ which shows that 0/1 loss relational classifi ers can exploit correlations in the link structure to improve classifi cation accuracy. Moreover at $\rho = 0.8$, *CSMN* achieves an avg. accuracy of 81.1% whereas the 0/1 loss *MN* achieves an avg. accuracy of 82.65% indicating that a higher avg. accuracy does not imply a lower avg. misclassifi cation cost.

We also report training and test times required by the various classifi ers in Table 2. Training time for *MN* and *CSMN* are greater than *LOGREG* which is not surprising due to the additional complexity introduced by learning from the link structure. Interestingly, *CSMN* tends to train faster than *MN*. Test times are roughly the same for all three classifi ers.

In Figure 2, at $\rho = 0.9$ and above, *MN* shows very high misclassifi cation costs. This has more to do with the inference algorithms used in our implementation of *MN*. We used loopy belief propagation (LBP) [2] for inference in our implementation which is the same inference algorithm that was used in Taskar et al [1]. LBP is a message passing algorithm which suffers (returns very poor estimates of class conditional probabilities) when the graph has a number loops with small clusters of nodes [30]. To observe this more carefully, we looked at the degree distributions of two test sets one generated with $\rho = 0.8$ (when *MN* does well) and another generated with $\rho = 1.0$ (when *MN* performs poorly). The test set generated with $\rho = 1.0$ contained more nodes with degrees > 1 than the test set generated with $\rho = 0.8$. Specifically, the test set generated with $\rho = 1.0$ contained a higher fraction of nodes with degrees 2,3,4,6,7 than the test set generated with $\rho = 0.8$ respectively. Note that both test sets have roughly the same number of links (411 and 414) due to the same setting of α. A lower value of ρ tends to distribute links around a bit while the graph is evolving and thus manages to avoid a few loops while a higher value of ρ causes new edges to form using the same nodes with the highest out-degree. These loops cause *MN* (which uses LBP) to return extremely poor estimates of class conditional probabilities thus returning a very high misclassification cost. *CSMN* avoids this pitfall because it does not rely on estimating probabilities and is more robust to loops. Inference in graphical models with loops is still an active field of reseach which aims to develop more accurate inference algorithms.

Another way to produce tightly linked clusters in the generated data is to simply increase the number of links in the graph. In our second experiment we varied α from 0 to 0.8 and kept ρ constant at 0.8. Recall that α is directly proportional to the frequency with which `connectNode` is called and that the number of links in the graph is roughly $\frac{1}{1-\alpha}$ times the number of nodes. Figure 3 shows that at low link densities ($\alpha = 0, 0.1, 0.2$), all three classifiers produce comparable results. With an increase in the number of links comes an increase in the number of edge cost matrices thus increasing the total misclassification costs as shown in the plot for *LOGREG*. At $\alpha = 0.2$ the relational classifiers exploit these correlations to produce results slightly better than *LOGREG* (*MN*'s 91.6 and *CSMN*'s 94.5 compared to *LOGREG*'s 94.8). But at $\alpha = 0.4$ and higher, *MN* begins to show signs of poor estimation of probabilities due to an excess of links (and loops) in the data. *CSMN*, on the other hand, shows more resilience to the increase in links and exploits the correlations in the link structure to produce results better than the other two classifiers until $\alpha = 0.6$. Beyond $\alpha = 0.6$, however, even *CSMN* falters due to an excess in link density.

In the final set of experiments we explored the effects of correlations present in the cost matrices themselves. Consider the earlier problem of classifying communication links into one of "suspicious" and "unsuspicious". Recall that we have three types of costs: the cost associated with resources for surveillance incurred when an unsuspicious link was misclassified, the cost associated with damage to life and property incurred when a suspicious communication link was la-

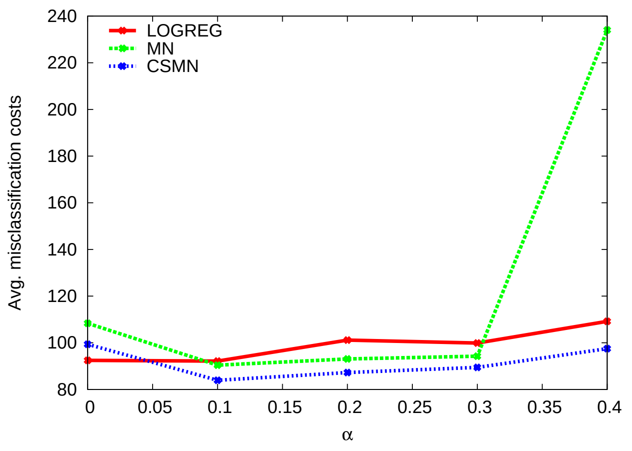


Figure 3: Avg. misclassification costs attained by varying α (X-axis). ρ was kept constant at 0.8.

beled otherwise and the relational cost incurred when any one of a non-terrorist's communication link was classified as "suspicious". An honest, non-terrorist entity in the communication network is going to be more confident and more protective of his rights to privacy than a terrorist who does not want to attract attention by moving to court if anyone of her/his communication links is being monitored. Thus the relational misclassification cost associated with a communication link belonging to a non-terrorist is going to be higher than the relational misclassification cost of a terrorist's communication link. This points to the fact that in some cases correlations present in the cost matrices might provide us information about the class labels and we need to exploit these correlations also. In our last set of experiments we tried to mimic this scenario and observed the performance of the different classifiers.

We introduced another parameter γ in our synthetic data generator which controlled the generation of the relational (edge) cost matrices. For every edge cost matrix $\text{Cost}(y_c, \tilde{y}_c)$, if the labels at both ends of the edge are 0 ("unsuspicious") then with probability γ we sample the off-diagonal elements uniformly from $[0, \text{ham}(y_c, \tilde{y}_c)]$ whereas with probability $1 - \gamma$ we sample uniformly from $[0, \frac{\text{ham}(y_c,\tilde{y}_c)}{10}]$, where $\text{ham}(y_c, \tilde{y}_c)$ denotes the hamming distance between y_c and $\tilde{y}_c$; otherwise we sample from $[0, \frac{\text{ham}(y_c,\tilde{y}_c)}{10}]$. The diagonal elements are still set to 0.

In the first experiment we kept α constant at 0.3, γ constant at 1.0 and varied ρ from 0.5 to 1.0, in other words, we varied the correlations in the link structure in the presence of strong correlations in the cost matrices. Figure 4 shows that *CSMN* thrives on such correlations in the cost matrices while *MN* improves slightly when the correlations in the links are pronounced ($\rho = 0.8$) but falters when there are too many loops in the graph ($\rho = 0.9, 1.0$).

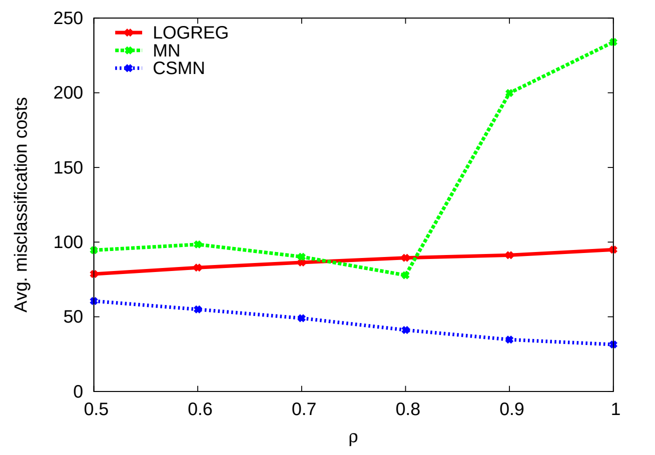


Figure 4: Avg. misclassification costs attained by varying ρ (X-axis). α and γ were kept constant at 0.3 and 1.0 resp.

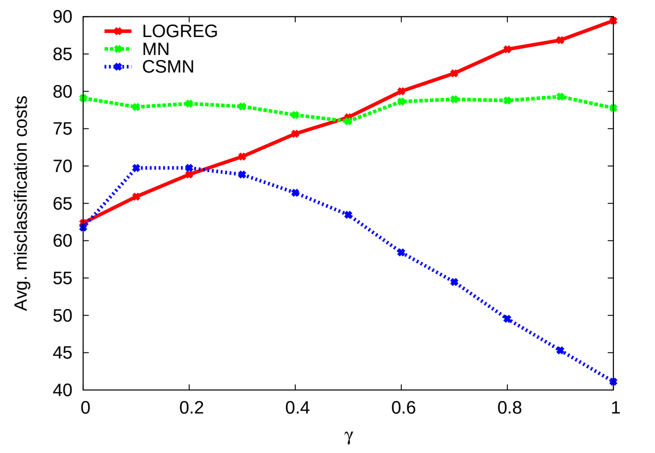


Figure 5: Avg. misclassification costs attained by varying γ (X-axis). α and ρ were kept constant at 0.3 and 0.8 resp.

In our last experiment we varied γ from 0 to 1.0 while keeping α constant at 0.3 and ρ constant at 0.8. Figure 5 shows the results. Note that there are two things happening here. As we increase γ, we increase the chances of larger costs appearing in the edge cost matrices and thus the misclassification cost results go higher as shown in the plot for *LOGREG*. As we increase γ we also increase the correlations in the cost matrices which can be exploited. *MN* neither takes advantage nor is adversely affected by the change in γ and shows roughly the same misclassification cost throughout the plot. *CSMN* shows a slight increase in misclassification costs initially ($\gamma = 0.1$) but quickly recovers and shows progressive improvements in results returning lower and lower misclassification costs with increasing correlations in the cost matrices.

7 Related Work

Link-based classification has been a topic of interest in many research communities at different points in time. Researchers in computer vision, natural language processing and machine learning have looked at this problem at length and developed a variety of methods. Chakrabarti et al [26] was one of the first to notice that exploiting correlations present amongst the labels of related entities significantly improves classification accuracy. Lafferty et al [3] followed up by proposing Conditional Random Fields (CRFs) for classifying entities which form linear chains which is a problem frequently encountered in the field of natural language processing. Taskar et al [1] extended Lafferty et al's work to handle irregular graphs.

Much of the research on link-based classification in the machine learning community has concentrated on classifying text corpora with links like hypertext corpora with hyperlinks or scientific publications with citations. One reason for this is the ease of availability of text classification datasets. Other application domains include classifying email text corpora ([16]), classifying datasets with information regarding various corporations ([17]), classification of links in hypertext datasets ([18]), predicting links in friendship networks ([18]), predicting links in text corpora ([19]), optical character recognition ([20]) etc.

The cost-sensitive learning community has yet to embrace link-based classification. Much of the research in cost-sensitive learning concentrates IID data. One approach to perform cost-sensitive learning is to use a standard 0/1 loss classifier to predict class conditional probabilities using which one attempts to find the labels which minimize the expected cost of misclassification ([7], [21]). Another approach is to make particular classifiers cost sensitive. Specific methods have been developed for decision trees ([23, 22]), support vector machines ([25]), neural networks ([24]) etc. None of these approaches consider relational data.

Social network analysts mainly use network connectivity information to acquire knowledge. Link analysis has acknowledged the need to combine text and link information ([13]). Yet most of the work either exploits only content information ([15, 10]) or only link information ([14, 11]). One exception is McCallum et al [12] which combines both content and link information to generate topic mixtures of emails and determine roles. McCallum et al uses a probabilistic graphical model which requires considerable efforts to design and involves making a number of independence assumptions that rarely hold in the real world. In contrast, the methods discussed in this paper involve discriminative models which are a much easier way to design classifiers involving both link and content information for relational data and make none of the independence assumptions. Moreover, to the best of our knowledge, none of the previous work in link analysis considers varied misclassification costs.

8 Conclusion

In this paper, we proposed the use of relational classifiers as a method to combine different types of information (e.g., link and attribute) to solve various classification tasks in social networks. We showed that classification can be used to formulate a number of problems in social networks like identifying terrorists in a terrorist network and identifying communication links which need monitoring in communication networks. Most of these problems come with various costs of misclassification which need to be kept in mind if we are to solve these problems correctly. To this end, we developed relational classifiers to perform cost-sensitive classification. We demonstrated the performance of relational classifiers and our proposed cost-sensitive relational classifier on synthetic data. Our main observations were that the presence of correlations in labels across links can be exploited by relational classifiers to achieve results better than standard non-relational classifiers. We showed that increasing correlations in the link structure of the data improves the results of relational classifiers. We also showed that when the number of links is high or when there are too many loops in the graph, standard relational classifiers falter due to their reliance on class conditional probabilities while our proposed cost-sensitive classifier is more robust to these problems. Finally, we showed that the proposed cost-sensitive relational classifier can exploit correlations in the cost matrices, something the other two classifiers (non-relational classifiers and standard relational classifiers) showed no evidence of being able to do.

References

[1] B. Taskar, P. Abbeel and D. Koller. *Discriminative Probabilistic Models for Relational Data*, UAI, 2002.

[2] J.S. Yedidia, W.T. Freeman and Y. Weiss. *Generalized Belief Propagation*, NIPS, 2000.

[3] J. Lafferty, A. McCallum, and F. Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. ICML, 2001.

[4] J. Diesner and K.M. Carley. *Exploration of Communication Networks from the Enron Email Corpus*. SIAM International Conference on Data Mining, Workshop on Link Analysis, Counterterrorism and Security, 2005.

[5] A. P. Wolfe and D. Jensen. *Playing Multiple Roles: Discovering Overlapping Roles in Social Networks*. In Proceedings of the ICML-04 Workshop on Statistical Relational Learning and its Connections to Other Fields, 2004.

[6] http://en.wikipedia.org/wiki/7_July_2005_London_bombings

[7] P. Domingos. *MetaCost: A general method for making classifiers cost sensitive*. In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, 1999.

[8] C. Elkan. *The Foundations of Cost-Sensitive Learning*. In Proceedings of the Seventeenth International Conference on Artificial Intelligence, 2001.

[9] http://profilesinterror.mindswap.org/

[10] M.W. Berry and M. Browne. *Email Surveillance Using Non-negative Matrix Factorization*. SIAM International Conference on Data Mining, Workshop on Link Analysis, Counterterrorism and Security, 2005.

[11] Y. Duan, J. Wang, M. Kam and J. Canny. *A Secure Online Algorithm for Link Analysis on Weighted Graph*. SIAM International Conference on Data Mining, Workshop on Link Analysis, Counterterrorism and Security, 2005.

[12] A. McCallum, A. Corrada-Emmanuel and X. Wang. *The Author-Recipient-Topic Model for Topic and Role Discovery in Social Networks, with Application to Enron and Academic Email*. SIAM International Conference on Data Mining, Workshop on Link Analysis, Counterterrorism and Security, 2005.

[13] M. Ben-Dov, W. Wu, R. Feldman and P. A. Cairns. *Improving Knowledge Discovery by Combining Text-Mining and Link Analysis Techniques*. SIAM International Conference on Data Mining, Workshop on Link Analysis, Counterterrorism and Security, 2004.

[14] C. Faloutsos, K. S. McCurley and A. Tomkins *Connection Subgraphs in Social Networks*. SIAM International Conference on Data Mining, Workshop on Link Analysis, Counterterrorism and Security, 2004.

[15] D. B. Skillicorn. *Detecting Related Message Traffic*. SIAM International Conference on Data Mining, Workshop on Link Analysis, Counterterrorism and Security, 2004.

[16] V. Carvalho and W. W. Cohen. *On the Collective Classification of Email Speech Acts*. SIGIR, 2005.

[17] J. Neville and D. Jensen. *Iterative classification in relational data*. AAAI, 2000.

[18] B. Taskar, M. F. Wong, P. Abbeel and D. Koller. *Link Prediction in Relational Data*. NIPS, 2003.

[19] L. Getoor, N. Friedman, D. Koller and B. Taskar. *Learning Probabilistic Models of Link Structure*. JMLR, 2002.

[20] B. Taskar, C. Guestrin and D. Koller. *Max-Margin Markov Networks*. NIPS, 2003.

[21] B. Zadrozny and C. Elkan. *Learning and making decisions when costs and probabilities are both unknown*. KDD, 2001.

[22] U.Knoll, G.Nakhaeizadeh and B.Tausend. *Cost-sensitive pruning of decision trees*. ECML, 1994.

[23] J. Bradford, C. Kunz, R. Kohavi, C. Brunk and C. Brodley. *Pruning Decision Trees with misclassification costs*. ECML, 1998.

[24] P.Geibel and F.Wysotzki. *Perceptron based learning with example dependent and noisy costs*. ICML, 2003.

[25] U. Brefeld, P. Geibel and F. Wysotzki. *Support Vector Machines with Example Dependent Costs*. ECML, 2003.

[26] S. Chakrabarti, B. Dom and P. Indyk. *Enhanced hypertext categorization using hyperlinks*. SIGMOD, 1998.

[27] B. Bollobas, C. Borgs, J. T. Chayes and O.Riordan. *Directed scale-free graphs*. SODA, 2003.

[28] T. Zhang and F.J. Oles. *Text Categorization based on regularized linear classification methods*. PAMI, 2001.

[29] Qing Lu and Lise Getoor, *Link Based Classification*. ICML, 2003.

[30] J.S. Yedidia, W.T. Freeman and Y. Weiss, *Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms*. IEEE Transactions on Information Theory, 2005.