# Cost-Sensitive Learning with Conditional Markov Networks

**Prithviraj Sen** SEN@CS.UMD.EDU
Department of Computer Science, University of Maryland, College Park, MD 20742.

**Lise Getoor** GETOOR@CS.UMD.EDU
Department of Computer Science, University of Maryland, College Park, MD 20742.

## Abstract

There has been a recent, growing interest in classification and link prediction in structured domains. Methods such as CRFs (Lafferty et al., 2001) and RMNs (Taskar et al., 2002) support flexible mechanisms for modeling correlations due to the link structure. In addition, in many structured domains, there is an interesting structure in the risk or cost function associated with different misclassifications. There is a rich tradition of cost-sensitive learning applied to unstructured (IID) data. Here we propose a general framework which can capture correlations in the link structure and handle *structured* cost functions. We present a novel cost-sensitive structured classifier based on Maximum Entropy principles that directly determines the cost-sensitive classification. We contrast this with an approach which employs a standard 0/1 loss structured classifier followed by minimization of the expected cost of misclassification. We demonstrate the utility of our proposed classifier with experiments on both synthetic and real-world data.

## 1. Introduction

There has been a recent, growing interest in classification and link prediction in structured domains. Methods such as Conditional Random Fields (CRFs) (Lafferty et al., 2001) and Relational Markov Networks (RMNs) (Taskar et al., 2002) have been introduced which use discriminative training to optimize collective classification. Most of these methods implicitly assume that all errors are equally costly.

Classification in the presence of varying costs associated with different types of misclassification is important for many practical applications. There is a rich tradition of cost-sensitive learning (Elkan, 2001; Domingos, 1999) applied to independent and identically distributed (IID) data for applications such as targeted marketing and fraud & intrusion detection. These approaches assume the traditional machine learning setting, where the data is viewed as a set of IID samples. Using a series of motivating examples, we show that when the data is structured the misclassification costs may also be structured. Specifically, besides the misclassification costs associated with misclassifying each sample we now have *misclassification costs associated with misclassifying groups of related samples*.

In this paper, we describe extensions which enable the use of 0/1 loss structured classifiers for cost-sensitive classification by minimizing the expected cost of misclassification. The problem with this approach is that it requires precise estimates of class conditional probabilities. One of the main concerns in traditional cost-sensitive learning is how to extract good estimates of probabilities from standard machine learning classifiers (Domingos, 1999; Zadrozny & Elkan, 2001). This is of particular concern in structured classifiers which, as we demonstrate in Section 6, can produce very poor estimates of class conditional probabilities under certain conditions. Our main contribution is a novel structured classifier based on Maximum Entropy principles that does not make explicit use of class conditional probabilities. We present learning and inference algorithms for the proposed classifier and compare its performance with other approaches.

## 2. Motivation

Here we describe two real-world scenarios that can be modeled as structured cost-sensitive classification problems.

### 2.1. Motivating Example #1: Loan Applicants

Consider the classic cost-sensitive classification problem of granting loans. This is a 2-class classification problem in which the bank needs to classify each loan application as

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. 

either "granted" or "denied". Let us assume that there is only one applicant per application and that each applicant also has an account in the bank with some savings in it. The cost of misclassifying a loan application which should have been granted is the loss of the interest on the loan and the loss of the amount in the applicant's bank account since there is a chance that the disgruntled applicant might close the account and move it to another bank. The cost of misclassifying a loan application which should have been denied is the loss of the principal amount of the loan in question (the applicant might default her/his payment).

Consider a small extension to the above scenario. Applicants can now have joint accounts which can have more than one account holder. Now, the cost of misclassifying a loan which should have been granted is the loss of the interest on the loan, loss of the amounts in the accounts singly owned by the applicant of the loan *and loss of the amount in any joint accounts of the applicant* (the disgruntled applicant may close these as well). More importantly, the amount in the joint accounts can be lost if *any* of the loans applied for by any of its account holders end up being denied. In fact, the amount in the joint account is a cost that is associated with related loans and is a *relational* cost. *Relational costs* can be modeled as a cost matrix $\text{Cost}(y_c, \tilde{y}_c)$ which specifies the cost incurred when a set of related entities 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 the set of account holders associated with any particular joint account, $\tilde{y}_c$ denotes the set of correct labels for those account holders (whether their loans are granted or denied) and $y_c$ denotes the labels assigned by the bank.

### 2.2. Motivating Example #2: Intelligent Light Control in Buildings

Building control strategies for *intelligent light control* aim to increase user comfort and reduce the energy consumption of lighting a building. Two aspects require special attention: first, we should utilize ambient light fully and make sure that lights are not turned "on" for locations in the building which already have enough light from external sources (a process Singhvi et al. (2005) refers to as *daylight harvesting*) and second, we should respect the occupant's preferences. Any control strategy needs to keep track of the amount of light at various locations in the building in order to achieve daylight harvesting. The use of light meters may not be feasible since sensing light via light meters is expensive both in terms of battery power and time. In such cases, a more prudent approach suggested by Deshpande et al. (2005) is to predict light at various locations by observing cheaper attributes (such as temperature and sensor voltage) and exploiting spatial correlations (attribute values of nearby sensors) via a statistical model rather than measure the expensive attributes directly.

We can also frame this problem as an instance of *structured cost-sensitive classification*. Consider the case where the light at an occupied location is predicted to be "low" when in fact it is well-lit. In this case, the control strategy will turn on lights and incur unwanted electricity costs. The opposite case is when a poorly lighted occupied location is classified as well-lit. In this case, the control strategy will refrain from turning on lights, and this results in losses in terms of user comfort. Most people would prefer to have lights turned on *in and around* the occupied location. Consider the following misclassification: suppose we correctly classify a well-lit occupied location but misclassify a nearby location which requires lighting as being well-lit. This will cause the occupant discomfort since s/he would prefer to have light in the nearby location but the control strategy will refrain from providing it. This cost, in terms of occupant discomfort, was incurred even though we correctly classified the occupied location. The misclassification cost associated with the pair of locations is in addition to the misclassification cost attached to each location described previously and these are, in fact, *relational costs* which can be modeled using structured cost matrices.

Many traditional cost-sensitive classification problems give rise to relational costs when extended to the structured case. Another structured domain that presents a rich source of varying misclassification costs is classification in social networks, e.g., predicting suspicious links in a communication network or classifying entities in a terrorist network. We can encode structured classification problems using a *Markov network* and we next provide relevant notation.

## 3. Preliminaries

We review the definitions of conditional Markov networks from Taskar et al. (2002). 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 $\Psi$. 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 and let $\mathbf{x}$ denote the observed values of $\mathbf{X}$. Let $X_c$ denote the observed random variables in clique $c \in C(G)$ and let $x_c$ denote the observed values of $X_c$. Let $\mathbf{Y}$ be the set of target random variables to which we want to assign labels and let $\mathbf{y}$ denote an assignment to $\mathbf{Y}$. Let $Y_c$ denote the set of target random variables in clique $c \in C(G)$ and let $y_c$ denote an assignment to it. A *conditional Markov network* is a Markov network $(G, \Psi)$ which defines the distribution $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C(G)} \psi_c(x_c, y_c)$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_c \psi_c(x_c, y'_c)$.

For the cost sensitive version of this problem, in addition to $(G, \Psi)$ 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 (possibly) 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:

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

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 performing 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 rewrite the above expression in terms of the marginals:

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

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.

The problem with the above approach is that it requires accurate estimates of the marginal probabilities. In fact, Domingos (1999) and Zadrozny and Elkan (2001) show how to improve the class conditional probabilities obtained from decision trees and naive bayes classifiers, respectively, using traditional ideas of bagging and smoothing for use in cost-sensitive classification. A slightly different idea proposed in Brefeld et al. (2003) is to learn a classifier function which associates a higher penalty term corresponding to misclassifications associated with higher costs . Brefeld et al. only consider IID input. Recent research in learning for structured domains has also concentrated on loss augmented learning of classifier functions (Taskar et al., 2003; Tsochantaridis et al., 2004) but they do not involve the loss function during inference. Our aim is to design a classifier for structured domains that penalizes configurations corresponding to higher costs both during learning and inference without the explicit computation of probabilities. We next derive a classifier which penalizes labelings associated with high misclassification costs based on Maximum Entropy principles.

## 5. Cost-Sensitive Markov Networks

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, $A_k$ is the $k^{th}$ empirical feature count and we employ $K$ such features.

We assume that the features distribute over the cliques and thus $f_k(\mathbf{x}, \mathbf{y}) = \sum_c f_k(x_c, y_c)$. 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, $A_k = \sum_c f_k(x_c, \tilde{y}_c)$. With these assumptions the above equation can be rewritten as:

$$\sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \sum_c \underbrace{(f_k(x_c, y_c) - f_k(x_c, \tilde{y}_c))}_{\text{clique specifi c constraint}} = 0, \qquad \forall k = 1, \ldots, K \tag{1}$$

Eq. 1 attempts to set the sum of $P(\mathbf{y}|\mathbf{x})$ weighted by the difference in total feature counts to 0. A configuration $\mathbf{y}$ with a large difference in total feature count is likely to be assigned a lower probability $P(\mathbf{y}|\mathbf{x})$ so that the sum over all the configurations remains 0. Thus, the difference in feature counts acts as a constraint. We would now like to modify Eq. 1 so that configurations associated with higher misclassification costs are assigned low probabilities. A natural way to do this is to scale the *clique specific constraint* with the loss associated with the misclassification of the clique and modify Eq. 1 to:

$$\sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \sum_c \underbrace{l_c(y_c, \tilde{y}_c)\,(f_k(x_c, y_c) - f_k(x_c, \tilde{y}_c))}_{\text{scaled clique specifi c constraint}} = 0, \qquad \forall k = 1, \ldots, K$$

The new Maximum Entropy formulation can now be expressed as:

$$\max \sum_{\mathbf{y}} -P(\mathbf{y} \mid \mathbf{x}) \log P(\mathbf{y} \mid \mathbf{x})$$
$$\text{s.t.} \sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) = 1,$$
$$\sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \sum_{c} l_c(y_c, \tilde{y}_c)(f_k(x_c, y_c) - f_k(x_c, \tilde{y}_c)) = 0, \quad \forall k = 1, \ldots, K$$

The dual of the above Maximum Entropy formulation is:

$$\min \log \left[ \sum_{\mathbf{y}'} \exp \left( \sum_{k,c} w_k l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \right) \right] \tag{2}$$

where $\{w_k\}_{1,\ldots K}$ are the parameters of the classifier.

Eq. 2 is our objective function. Note that this objective function is not log-linear . Thus the standard methods of inference and learning do not apply. We next describe learning and inference algorithms for our classifier.

### 5.1. Learning

Given fully labeled training data, we can learn the above model by minimizing Eq. 2 w.r.t $\{w_k\}_{1,\ldots K}$ :

$$\text{argmin}_w \log \left[ \sum_{\mathbf{y}'} \exp \{ \sum_{k,c} w_k l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \} \right]$$

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

Differentiating with respect to $w_k$ we get:

$$\sum_{\mathbf{y}'} \left[ \frac{1}{Z} \exp \left( \sum_{k} w_k \sum_{c} l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \right) \sum_{c} l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \right] \tag{3}$$

where $Z = \sum_{\mathbf{y}''} \exp \left( \sum_{k,c} w_k l_c(y''_c, \tilde{y}_c)(f_k(x_c, y''_c) - f_k(x_c, \tilde{y}_c)) \right)$.

In order to formulate the gradient (Eq. 3) computation as a standard inference problem, let us now define the following probability distribution:

$$q(\mathbf{y}') = \frac{1}{Z} \exp \left( \sum_{k} w_k \sum_{c} l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \right)$$

The gradient with respect to $w_k$ (Eq. 3) can now be expressed in terms of $q(\mathbf{y}')$ as:

$$\sum_{\mathbf{y}'} q(\mathbf{y}') \sum_{c} l_c(y'_c, \tilde{y}_c)[f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)] \tag{4}$$

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. Fortunately, we can rewrite Eq. 4 as:

$$\sum_{c, y'_c} \mu_c^q(y'_c) l_c(y'_c, \tilde{y}_c)[f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)]$$

where $\mu_c^q(y'_c)$ 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 $\mathbf{y}'$. The marginals $\mu_c^q(y'_c)$ can be estimated using standard inference algoritms (e.g., Yedidia et al. (2000)).

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

### 5.2. Inference

For inference, we need to determine the optimal labeling $\mathbf{y}$ which minimizes Eq. 2:

$$\text{argmin}_{\mathbf{y}} \log \left[ \sum_{\mathbf{y}'} \exp \{ \sum_{k,c} w_k l_c(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) \} \right] \tag{5}$$

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 apply Jensen's inequality to Eq. 5:

$$\log \left[ \sum_{\mathbf{y}'} \exp \{ \sum_{k,c} w_k l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) \} \right] \geq \sum_{\mathbf{y}'} q(\mathbf{y}') \sum_{k} w_k \sum_{c} l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) - \sum_{\mathbf{y}'} q(\mathbf{y}') \log q(\mathbf{y}') \tag{6}$$

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. 6 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. 6 is:

$$\sum_{\mathbf{y}'} q(\mathbf{y}') \sum_k w_k \sum_c l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) - \sum_{\mathbf{y}'} q(\mathbf{y}') \log q(\mathbf{y}') - \mu \left( \sum_{\mathbf{y}'} q(\mathbf{y}') - 1 \right) \quad (7)$$

To maximize w.r.t to $q(\mathbf{y}')$, we differentiate with respect to $q(\mathbf{y}')$ and set the derivative to 0 to get:

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

where $\sum_{\mathbf{y}'} q(\mathbf{y}') = 1$. As discussed before, computing $q(\mathbf{y}')$ for every complete assignment $\mathbf{y}'$ may not be feasible.

The second step of the optimization requires minimizing with respect to $\mathbf{y}$. Thus, we only need to look at the first term in Eq. 7 (since this is the only term which involves $\mathbf{y}$):

$$\sum_{\mathbf{y}'} q(\mathbf{y}') \sum_k w_k \sum_c l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) = \sum_{y'_c, c} \sum_k w_k l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) \mu_c^q(y'_c) \quad (8)$$

where $\mu_c^q(y'_c)$ 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 step of the optimization. Again, these marginal probabilities $\mu_c^q(y'_c)$ can be estimated using inference algorithms (e.g., Yedidia et al. (2000)).

One way to minimize Eq. 8 is to define the following distribution and determine the $\mathbf{y}$ corresponding to the highest probability (note the multiplication of the exponent by a negative sign):

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

where $\sum_{\mathbf{y}} r(\mathbf{y}) = 1$. To determine the $\mathbf{y}$ corresponding to the maximum $r(\mathbf{y})$, we can use inference algorithms.

## 6. Experiments

We performed experiments on synthetic random graph data and real-world sensor network data to address the following questions: Can misclassification costs be reduced by exploiting correlations across links and do structured cost-sensitive classifiers have an advantage over traditional machine learning cost-sensitive classifiers? Are the probability estimates of 0/1 loss structured classifiers dependable? How do cost-sensitive methods perform in the presence of structured cost functions?

In all our experiments we compare misclassification costs achieved by three different classifiers: *LOGREG* - logistic regression which classifies each sample based on the attributes followed by minimization of expected cost of misclassification, *MN* - relational markov networks (Taskar et al., 2002) followed by minimization of expected cost of misclassification as described in Section 4 and *CSMN* - the proposed classifier described in Section 5.

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, we assumed that different parameters are a priori independent and define $p(w_i) = \lambda w_i^2$. We tried a range of regularization constants for each classifier and report the best results. Typically, we found that for *LOGREG* $\lambda = |\mathbf{Y}| \times 10^{-4}$ (where $|\mathbf{Y}|$ is the number of samples in the training set) gave the best results (which matches with Zhang and Oles (2001)'s suggestion), for *MN* $\lambda = 10$ gave the best results (Taskar et al. (2002) report using a regularization constant of the same magnitude $\lambda \approx 5.5$) and for *CSMN* $\lambda = 20$ returned the best results.

### 6.1. Synthetic Data Generation

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 structured classifiers will perform on such networks, we chose to model our synthetic data generation algorithm along the lines of the evolutionary network model described in Bollobas et al. (2003). The algorithm is outlined in Algorithm 1.

The synthetic data generator (Algorithm 1) "grows" a graph from an empty set of nodes. The number of nodes in the final graph is controlled by the parameter *numNodes*. $\alpha$ 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. For all our experiments, we set *numNodes* $= 300$.

We generated binary class data using our synthetic data generator; this is a common case in cost-sensitive applications ("loan granted"/"loan denied", "good customer"/"bad customer", "terrorist"/"non-terrorist" etc.). Algorithm 1 proceeds through iterations and in each iteration it either connects a newly created node to the graph or connects two existing nodes in the graph. Each time Algorithm 1 creates an edge it makes a call to Algorithm 2. Algorithm 2 implements a rudimentary form of *preferential attachment* where a node can choose which nodes to link based on their labels. This introduces correlations amongst the labels across links. The strength of these correlations is controlled by the parameter $\rho$. Each node can link to nodes of its own class with probability $\rho$. With probability $1 - \rho$, a node

can choose to link to a node of the other class. In addition, nodes with higher out-degree have 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. (2003) for more details regarding this aspect of our synthetic data generation algorithm. After generating the graph, we generate attributes for each node using fixed, class-specific multivariate Bernoulli distributions.

**Algorithm 1** Synthetic data generator

SynthGraph(*numNodes*, $\alpha$, $\rho$)
1: $i \leftarrow 0$
2: $G \leftarrow \emptyset$
3: **while** $i <$ *numNodes* **do**
4: sample $r \in [0,1]$ uniformly at random
5: **if** $r <= \alpha$ **then**
6: $v \leftarrow$ select any node uniformly at random from $G$
7: connectNode($v$, $G$, $\rho$)
8: **else**
9: add a new node $v$ to $G$
10: choose $v.label$ from $\{0,1\}$ uniformly at random
11: connectNode($v$, $G$, $\rho$)
12: $i \leftarrow i+1$
13: **end if**
14: **end while**
15: **for** i = 1 to numNodes **do**
16: $v \leftarrow i^{th}$ node in $G$
17: genAttributes($v$)
18: genNodeCostMatrix($v$)
19: **end for**
20: **for** each edge $e$ in $G$ **do**
21: genEdgeCostMatrix($e$)
22: **end for**
23: return $G$

**Algorithm 2** Generating an edge in the synthetic data graph

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

Finally, we generate sample dependent cost matrices for the data. 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 (`genNodeCostMatrix`) and one for the edges (`genEdgeCostMatrix`). For the node cost matrices $\text{Cost}(y,\tilde{y})$, we set the diagonal entries to 0 and 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 $[0, \frac{\text{ham}(y_c,\tilde{y})}{2}]$ where $\text{ham}(y_c,\tilde{y}_c)$ denotes the hamming distance between $y_c$ and $\tilde{y}_c$. The factor of $\frac{1}{2}$ with the hamming distance reduces the disadvantage of *LOGREG*.

For each experiment, we produced three datasets and performed 3-fold cross validation. Each number we report is the average misclassification cost obtained. For all runs of *CSMN*, we set the clique loss matrices equal to the cost matrices: $l_c(y_c,\tilde{y}_c) = \text{Cost}_c(y_c,\tilde{y}_c)$.

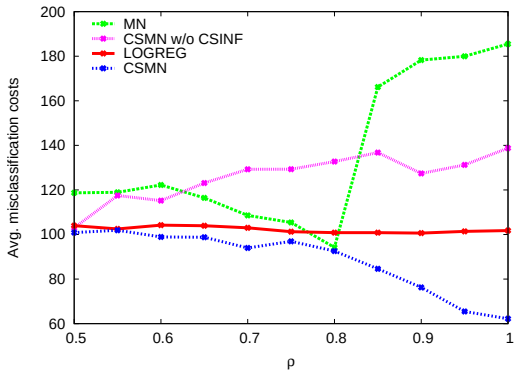


*Figure 1.* 3-fold cross validation results of varying $\rho$ (X-axis). $\alpha$ was kept constant at 0.25.

### 6.2. Performance Comparisons

For our first experiment, we varied the correlation amongst labels across links to find out if structured classifiers actually help decrease misclassification costs. We varied the value of $\rho$ from 0.5 to 1.0 keeping $\alpha$ constant at 0.25. Recall that $\rho$ 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 labels.

Figure 1 shows that structured classifiers (*CSMN* and *MN*) can exploit correlations in the link structure to reduce misclassification costs. The plot shows that *CSMN* manages to produce lower misclassification costs than *MN* on all settings of $\rho$. *CSMN* achieves 8.19% reduction in costs over *LOGREG* at $\rho = 0.8$ which increases to 38.87% at $\rho = 1.0$. Note that at $\rho = 0.75$, *CSMN* achieves an avg. accuracy of 80.44% whereas the 0/1 loss *MN* achieves an avg. accuracy of 81.88% indicating that a higher avg. accuracy does not necessarily imply a lower avg. misclassification cost.

We also show a plot labeled *CSMN w/o CSINF* that uses the parameters learned by CSMN but performs inference by simply maximizing the sum of the feature counts weighted by the parameters (ignores costs completely during inference). As is clear, *CSMN w/o CSINF* performs quite poorly even in the case when there are significant correlations in the links ($\rho = 0.7, 0.8$) and shows the importance of involving the costs during inference.

In Figure 1, at $\rho = 0.85$ and above, *MN* shows very high misclassification costs. This is most likely the result of the inference algorithms used. We used loopy belief propaga-

tion (LBP) (Yedidia et al., 2000) for inference in our implementation as suggested by Taskar et al. (2002). LBP is a message passing algorithm that is known to return very poor estimates of class conditional probabilities when the graph has a large number of short cycles (Yedidia et al., 2005). As we increase the strength of the correlations in the link structure, new nodes attach themselves to the same nodes in the graph (the nodes with the highest out-degree with the same label). This introduces cycles making it very difficult for LBP to estimate the marginal probabilities. Note that *CSMN* avoids this pitfall because it does not rely on the explicit use of probabilities.

The previous experiment was performed on datasets with a constant number of edges ($\approx$ 400 edges in a graph of 300 nodes). We also wanted to see how varying edge density affects the performance of the classifiers. In our second experiment, we varied $\alpha$ from 0 to 0.4 and kept $\rho$ constant at 0.85. Recall that the number of edges in the graph is roughly $\frac{1}{1-\alpha}$ times the number of nodes. Figure 2 shows the results. At $\alpha = 0.0$, the misclassification cost due to the misclassification of edges is small but this fraction increases as $\alpha$ increases. Since *LOGREG* does not care about the edge cost matrices it performs well at $\alpha = 0.0$ but poorly at higher settings of $\alpha$. *CSMN* consistently returns lower misclassification costs than *MN*. Increasing edge density increases the number of short cycles. At $\alpha = 0.25$ and higher, due to the large number of cycles in the graph, *MN* returns inaccurate estimates of class conditional probabilities thus resulting in very poor results. At $\alpha = 0.4$, *CSMN* achieves a 63.52% reduction in costs over *MN* and 13.26% reduction in costs over *LOGREG*.

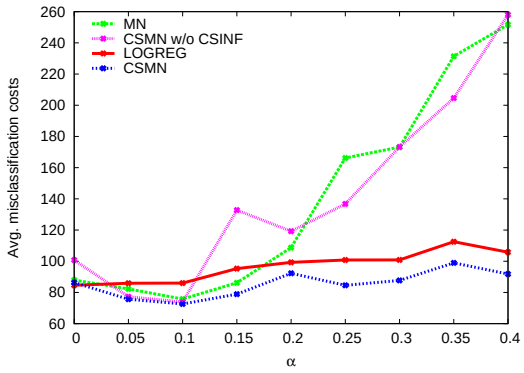


*Figure 2.* 3-fold cross validation results of varying $\alpha$ (X-axis). $\rho$ was kept constant at 0.85.

### 6.3. Experiments on Sensor Network Data

Next, we report experiments comparing the performance of the various cost-sensitive classifiers on the problem of providing intelligent light control discussed in Section 2.2. The *Intel lab* dataset (Bodik et al., 2004) contains more than 2,000,000 readings consisting of *temperature*, *humidity*, *light*, *battery voltage*, *time* and *date* recorded from a sensor network of about 54 different sensors/motes along with the sensors' ids and $(x,y)$ coordinates. To introduce links between the sensors, we defined an edge for every pair of sensors which were within 5 meters of each other and obtained a dependency graph with 122 edges. We performed experiments to predict light at various locations using the other three attributes, temperature, humidity and battery voltage, and spatial correlations. We discretized each attribute into three nominal values and removed all readings with missing values. To generate relational graphs, we organized the dataset into *snapshots* such that each snapshot consists of readings from at least 50 different sensors with no two readings from the same sensor. Finally, We divided the set of *snapshots* into three sets containing, roughly, the same distribution of values for the light attribute (*high_light*, *medium_light* and *low_light*) and performed 3-fold cross validation. Unfortunately, the dataset does not come with cost matrices. We augmented the dataset with, hopefully realistic, costs and varied the misclassification costs to illustrate the performance of the various classifiers under different settings.

For simplicity, we considered cliques up to size 2 and generated label dependent cost matrices for nodes and edges. We introduced a parameter $\gamma$ to define the cost due to occupant discomfort in units of electricity and defined the cost matrices in terms of $\gamma$. To generate the node cost matrix $\text{Cost}_{\text{node}}(y,\tilde{y})$, we used simple intuitions such as: if the predicted value of light is *high_light* and the correct class label is *low_light* (cost due to occupant discomfort) then we pay a cost of $\gamma$, if the predicted value of light is *low_light* and the correct class label is *medium_light* (cost due to excess electicity usage) then we pay a cost of 1. This gave us a $3 \times 3$ node cost matrix. To define the edge cost matrix $\text{Cost}_{\text{edge}}(y_c,\tilde{y}_c)$, we used simple intuitions like the one introduced in Section 2.2: if the predicted class labels of a pair of linked sensors is (*high_light*, *high_light*) whereas the correct class labels are (*high_light*, *low_light*) then we pay a cost of $\gamma$ (occupant discomfort caused due to lack of light in a location near the occupied location). This gave us a $9 \times 9$ cost matrix with 32 non-zero entries. For these experiments we set the clique loss matrices equal to the cost matrices: $l_c(y_c,\tilde{y}_c) = \text{Cost}_c(y_c,\tilde{y}_c)$.

Determining the appropriate value of $\gamma$ is a difficult problem. Intuitively, if we set $\gamma$ too high then the classifiers will tend to label all nodes with a low light value since the cost associated with occupant discomfort is too high. During our experiments this phenomenon occurred at $\gamma = 2$. On the other hand, if we set $\gamma$ too low ($= 0.2$) then the classifiers tend to label all nodes with a high light value because the cost of turning "on" the lamp at some location is too high. We demonstrate the performance of the various classifiers

*LOGREG*, *MN* and *CSMN* at three different settings of $\gamma$ (Figure 3). As Figure 3 shows, *CSMN* produces the lowest average misclassification costs. At $\gamma = 1$, *CSMN* achieved a 30.15% reduction in average misclassification costs over *LOGREG* and a 9.94% reduction in average misclassification costs over *MN*.

## 7. Conclusion

In this paper, we have formulated the cost-sensitive classification problem for structured data. Using a series of motivating examples, we showed that in structured classification the misclassification costs also tend to be structured. Existing unstructured IID cost-sensitive classification methods do not provide a natural means to handle structured cost functions. We proposed a baseline based on existing 0/1 loss structured classifiers which minimizes the expected cost of misclassification. We also proposed a novel classifier which does not explicitly depend on the accurate estimation of class conditional probabilities. We compared the performance of various cost-sensitive classifiers on synthetic and real-world data to show that the proposed classifier can lead to significant reductions in misclassification costs.

**Acknowledgements:** This work was supported by the National Science Foundation, NSF #0423845, with additional support from the National Geospatial Agency and the ITIC KDD program. The authors would like to thank all the anonymous reviewers for their helpful comments.

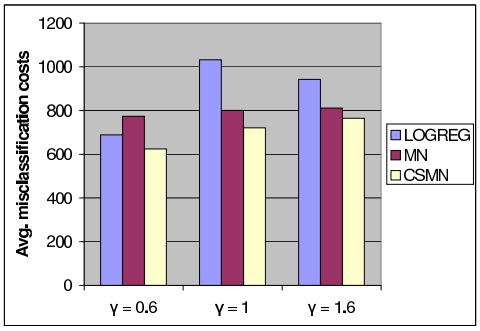


*Figure 3.* 3-fold cross validation results on *Intel lab* (Bodik et al., 2004) dataset, X-axis shows various settings of $\gamma$.

## References

Bodik, P., Hong, W., Guestrin, C., Madden, S., Paskin, M., & Thibaux, R. (2004). Intel lab dataset. `http://berkeley.intel-research.net/labdata/`.

Bollobas, B., Borgs, C., Chayes, J. T., & Riordan, O. (2003). Directed scale-free graphs. *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*.

Brefeld, U., Geibel, P., & Wysotzki, F. (2003). Support vector machines with example dependent costs. *Proceedings of the European Conference on Machine Learning*.

Deshpande, A., Guestrin, C., Madden, S., & Hong, W. (2005). Exploiting correlated attributes in acquisitional query processing. *International Conference on Data Engineering*.

Domingos, P. (1999). Metacost: A general method for making classifiers cost sensitive. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

Elkan, C. (2001). The foundations of cost-sensitive learning. *Proceedings of the International Conference on Artificial Intelligence*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the International Conference on Machine Learning*.

Singhvi, V., Krause, A., Guestrin, C., Garrett, J., & Matthews, H. S. (2005). Intelligent light control using sensor networks. *Conference on Embedded Networked Sensor Systems*.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *Advances in Neural Information Processing Systems*.

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

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2000). Generalized belief propagation. *Advances in Neural Information Processing Systems*.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

Zhang, T., & Oles, F. (2001). Text categorization based on regularized linear classification methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.