Cost-Sensitive Learning with Conditional Markov Networks

Prithviraj Sen* sen@cs.umd.edu
Lise Getoor* getoor@cs.umd.edu

Abstract

There has been a recent, growing interest in classification and link prediction in structured domains. Methods such as conditional random fields and relational Markov networks 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 two new cost-sensitive structured classifiers based on maximum entropy principles. The first determines the cost-sensitive classification by minimizing the expected cost of misclassification. The second directly determines the cost-sensitive classification without going through a probability estimation step. We contrast these approaches with an approach which employs a standard 0/1-loss structured classifier to estimate class conditional probabilities followed by minimization of the expected cost of misclassification and with a cost-sensitive IID classifier that does not utilize the correlations present in the link structure. We demonstrate the utility of our cost-sensitive structured classifiers with experiments on both synthetic and real-world data.

1 Introduction

Traditional machine learning classifiers assume that data consists of numerous independent and identically distributed (IID) examples. Many domains give rise to data that does not satisfy this assumption and is more naturally represented by a graph or network structure (e.g., classifying hypertext documents connected via hyperlinks, sequence data arising from natural language processing, 2D images in computed vision, etc.). There has been a recent, growing interest in classification and link prediction in such structured domains. Methods such as conditional random fields (CRFs) (Lafferty et al., 2001) and relational

*Department of Computer Science, University of Maryland, College Park, MD 20783.

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 misclassifications is important for many practical applications. There is a rich tradition of cost-sensitive learning (Domingos, 1999; Elkan, 2001) for applications such as targeted marketing and fraud and intrusion detection that assume that the data is best viewed as a set of IID examples. Using a series of motivating examples, we show that when the data is structured, the misclassification costs may also be structured. Specifically, besides the costs associated with misclassifying each example we now have *costs associated with misclassifying groups of related examples*.

In this paper, we propose two cost-sensitive structured classifiers based on maximum entropy principles. The first classifier is a simple extension of 0/1-loss structured classifiers using Bayes risk theory where the cost-sensitive classification is obtained 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 and Elkan, 2001). This is of particular concern in structured classifiers where it is common to utilize approximate inference methods since exact inference is usually too expensive especially in the case when we are dealing with networked data devoid of any special properties or regularities (irregular graphs). To address this issue, we propose a novel cost-sensitive structured classifier that does not make explicit use of class conditional probabilities. We present learning and inference algorithms for both proposed classifiers and compare their performance with other approaches.

This paper builds on our earlier work in Sen and Getoor (2006). In comparison with the earlier paper, this manuscript contains additional details, new technical advances and a significantly expanded experimental evaluation. Specifically, we develop new approaches for learning structured cost-sensitive conditional Markov networks based on minimizing expected costs of misclassification using the costs on the training set. In addition, we also present new experiments with real-world sensor data quantifying the conditions when the various models succeed in reducing misclassification costs.

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 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 that 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 that 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.

3 Related Work

Research on 0/1-loss structured classifiers has seen a considerable amount of activity in various research communities ranging from machine learning, information retrieval, spatial statistics, computer vision and the database research community each with slightly different areas of focus. For instance, researchers in computer vision and spatial statistics tend to concentrate on graphs with regular structure such as 2D grids or 3D grids whereas researchers in machine learning and information retrieval tend to work with irregular graphs. We refer the interested reader to leading conferences in each of the above research communities to get a taste of their specific brands of research. Here, we report recent research performed mainly by the machine learning community since that is most closely related to our work in this paper.

One of the most popular applications for 0/1-loss structured classifiers is the classification of hypertext and webpage classification where the objective is to classify wepbages and the assumption is that the hyperlinks that connect them represent correlations in labels across links. Chakrabarti et al. (1998) is perhaps one of the earliest and most well known works that looked at this application using an extension of the Naive Bayes classifier that could exploit the hyperlink structure. Slattery and Craven (1998) also considered the same application but utilized techniques from inductive logic programming combined with a logistic regression classifier to utilize hyperlinks and text-based attributes of webpages. These works showed that classification accuracy can be significantly improved by exploiting the graph structure.

Lafferty et al. (2001) was perhaps the first to provide a coherent, principled probabilistic framework utilizing undirected graphical models or Markov networks based on maximum entropy principles as a means for constructing structured classifiers and applied them to the task of labeling sequence data available in the natural language processing domain. Taskar et al. (2002) extended these ideas to classify data with arbitrary graph structures. When the graph structure is linear, efficient inference is possible using techniques based on

Viterbi decoding (Lafferty et al., 2001), the same cannot be said when dealing with arbitrary graphs. Taskar et al. proposed the use of approximate inference techniques to get around this problem. Since then there has been a spate of work in this field including iterative structured classifiers (Neville and Jensen, 2000; Lu and Getoor, 2003), generative models for both link structure and attributes (Cohn and Hofmann, 2001; Getoor et al., 2002), max-margin Markov networks (Taskar et al., 2003), associative Markov networks (Taskar et al., 2004), learning structured classifiers in unsupervised and semi-supervised settings (Xu et al., 2006) etc.

Research in cost-sensitive learning can be differentiated into three main approaches. The first approach includes specific ideas to make particular classifiers cost-sensitive such as decision trees (Bradford et al., 1998; Knoll et al., 1994), neural networks (Geibel and Wysotzki, 2003) and support vector machines (Fumera and Roli, 2002). The second approach utilizes Bayes risk theory and assigns each example to the class with the lowest expected misclassification cost (Domingos, 1999; Zadrozny and Elkan, 2001). This requires the use of classifiers that can produce accurate class-conditional probabilities. There has been some work extending such approaches to the case when we do not know the misclassification costs exactly (Zadrozny and Elkan, 2001). The third approach involves modifying the distribution of training examples so that learned classifier becomes cost-sensitive (Chan and Stolfo, 1998; Zadrozny et al., 2003; Abe et al., 2004). Of the three, the third approach is perhaps most closely connected to the assumption that the data consists of IID examples and thus it is not obvious how to extend it to the case of structured classifiers where the IID assumption is not made.

In this paper, we propose two cost-sensitive structured classifiers. The first one is a simple extension of conditional Markov networks (Taskar et al., 2002) using Bayes risk theory. The second one is derived by adding misclassification costs to the maximum entropy formulation itself which forms the basis of conditional Markov networks. In the next section, we describe the notation used in the rest of the paper and provide some relevant background on Markov networks and related algorithms.

4 Preliminaries

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 $\mathcal{C}_G$ denote a set of (not necessarily maximal) cliques in G. For each $c \in \mathcal{C}_G$, 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 $\Psi = \{\psi_c(V_c)\}_{c \in \mathcal{C}_G}$. For classification problems, we are often interested in conditional models. Let $\mathbf{X}$ be the set of observed random variables on which we condition and let $\mathbf{x}$ denote the observed values of $\mathbf{X}$. Let X_c denote the observed random variables in clique $c \in \mathcal{C}_G$ 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 $\mathbf{Y}$. Finally, let Y_c denote the set of target random variables in clique $c \in \mathcal{C}_G$ and let y_c denote an assignment to it.

Definition 4.1 [Conditional Markov Networks (Taskar et al., 2002)]. *Let $G = (\mathbf{V}, E)$ be an undirected graph with Ψ, $\mathbf{y}$, $\mathbf{x}$, $\mathcal{C}_G$, x_c and y_c as defined above. A* conditional Markov network *is a Markov network (G, Ψ) which defines the distribution $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}_G} \psi_c(x_c, y_c)$ where $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_c \psi_c(x_c, y'_c)$.*

Notice that the above definition of conditional Markov networks is defined using clique-specific potential functions $\Psi = \{\psi_c(V_c)\}_{c \in \mathcal{C}_G}$. This is not very useful in the supervised learning scenario where our goal is to learn a classifier from given labeled data and apply the classifier to determine the labels for unseen test data. For instance, having learned the potential functions on the cliques of the training data one would not know how to extend them to the previously unseen test data. To get around this problem it is common to define the potential functions in terms of a small set of features:

$$\psi_c(V_c = v_c) = \exp\left[\sum_{k=1}^{K} w_k f_k(x_c, y_c)\right] \tag{1}$$

where f_k denotes the k^{th} feature, K is the number of features used, y_c is the assignment to the target random variables, x_c are the observed attributes in V_c as given in v_c and $\{w_k\}_{k=1}^{K}$ are the parameters of the conditional Markov network that assigns a weight to each feature.

Conditional Markov networks, as defined above, are based on the principles of maximum entropy (Jaynes and Rosenkrantz (ed.), 2003) that aims to learn the probability distribution with the maximum entropy whose expected feature counts are equal to the empirical feature counts (feature counts on the labeled training data). Expressed in symbols this can be written as:

$$\begin{aligned} \max & \sum_{\mathbf{y}} -P(\mathbf{y} \mid \mathbf{x}) \log P(\mathbf{y} \mid \mathbf{x}) \\ \text{s.t.} & \sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) = 1, \\ & \sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) f_k(\mathbf{x}, \mathbf{y}) = A_k \quad \forall k = 1, \ldots, K \end{aligned} \tag{2}$$

where A_k is the count of the k^{th} feature from the labeled training data.

Solving the above constrained optimization for $P(\mathbf{y}|\mathbf{x})$ gives us the expression for $P(\mathbf{y}|\mathbf{x})$ shown in Definition 4.1 as a product of clique potentials.

For the cost-sensitive version of the structured classification problem, in addition to (G, Ψ) as in traditional 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 $\mathcal{C}_H$ denote the set of (not necessarily

maximal) cliques in H. Let Y_h denote the set of target random variables present in clique $h \in \mathcal{C}_H$ and let y_h denote an assignment to Y_h. For each clique $h \in \mathcal{C}_H$, there is a clique loss function $l_h(y_h, \tilde{y}_h)$. l_h is determined by the cost matrices $\{\text{Cost}_h(y_h, \tilde{y}_h)\}_{h \in \mathcal{C}_H}$ 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:

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

Our aim is to determine $\mathbf{y}$ which corresponds to the minimum misclassification cost. In this paper, we consider the special case where $\mathcal{C}_G = \mathcal{C}_H$. Note that the same random variable may be involved in many different cliques, and according to the above formulation (mis)classification of random variables with higher degrees has a greater affect on the overall misclassification cost associated with the complete assignment.

While describing learning and inference algorithms for the new classifiers we will frequently require two types of quantities that need to be computed from Markov networks: marginal probabilities for each target random variable and expectations of feature counts. Both these quantities can be computed using inference procedures. Unfortunately, most relational datasets and their underlying Markov networks consist of thousands of nodes and exact inference is infeasible unless the Markov network has special properties such as being a linear chain, being a tree, having low treewidth or consisting of very few nodes. In such cases, the usual approach is to resort to approximate inference methods. We next describe a very simple yet general approximate inference algorithm known as *Loopy Belief Propagation* (LBP) (Yedidia et al., 2000) that is widely regarded for its efficacy (Murphy et al., 1999; Berrou et al., 1993; McEliece et al., 1998; Kschischang and Frey, 1998). Note that there exist other approximate inference algorithms, e.g., (Hummel and Zucker, 1983; Besag, 1986; Yedidia et al., 2000; Minka, 2001), and we are not suggesting that LBP is the only algorithm which can be used. For completeness and to provide context, we provide a brief description of LBP next.

4.1 Loopy Belief Propagation (Yedidia et al., 2000)

In this section, for brevity, we assume that the set of cliques $\mathcal{C}_G$ in the conditional Markov network $G = (\mathbf{V}, E)$ is composed of only the set of nodes and the set of edges, otherwise known as a *pairwise Markov network*. Most of the material in this section is based on Yedidia et al. (2000). There exist various methods to extend the basic LBP procedure to larger clique sizes and for those details we refer the interested reader to Yedidia et al. (2005).

We will use indices to enumerate the target random variables in $\mathbf{Y}$. Thus Y_i represents the i^{th} random variable in $\mathbf{Y}$. When we restrict $\mathcal{C}_G$ to just the set of

nodes and edges, $P(\mathbf{y}|\mathbf{x})$ (Definition 4.1), can be rewritten as:

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \phi_i(y_i) \prod_{(i,j)\in E} \psi_{ij}(y_i, y_j)$$

where ϕ_i represents the product of clique potentials defined on the node Y_i and any edges containing Y_i as the only target random variable.

The LBP procedure is very simple and involves exchanging messages among target random variables connected via edges. Each message is a function of the label on the destination node. Let $m_{i\to j}(y_j)$ denote the message sent from the i^{th} target random variable to the j^{th} target random variable. The fixed point equation describing LBP is:

$$m_{i\to j}(y_j) = \alpha \sum_{y_i} \phi_i(y_i)\psi_{ij}(y_i, y_j) \prod_{k\in\mathcal{N}_i\setminus j} m_{k\to i}(y_i) \quad \forall y_j, \forall (ij) \in E$$

where α is a normalizing constant such that $\sum_{y_j} m_{i\to j}(y_j) = 1$ and $\mathcal{N}_i$ denotes the set of target random variables Y_i is linked to.

Once the messages stabilize, we can compute the marginal probabilities as we show next. In practice, one usually measures the square of the L2-norm of the change in the message entries summed over all messages at the end of each iteration. If this change falls below a user-defined threshold then LBP is said to have stabilized. Sometimes, this criterion may not be met, in which case we stop after a user-defined number of iterations have elapsed.

4.1.1 Extracting marginal probabilities

Let $\mu_i(y_i)$ denote the marginal probability of target random variable Y_i being labeled with y_i. One can compute $\mu_i(y_i)$ (approximately) from the messages $m_{i\to j}(y_j)$ (once they have stabilized) as follows:

$$\mu_i(y_i) = \alpha\phi_i(y_i) \prod_{j\in\mathcal{N}_i} m_{j\to i}(y_i) \quad \forall i, \forall y_i$$

where α is a normalization constant such that $\sum_{y_i} \mu_i(y_i) = 1$.

LBP also gives us a method to compute the approximate edge marginal probabilities for an edge between the i^{th} and j^{th} target random variables:

$$\mu_{ij}(y_i, y_j) = \alpha\phi_{ij}(y_i, y_j) \prod_{k\in\mathcal{N}_i\setminus j} m_{k\to i}(y_i) \prod_{k\in\mathcal{N}_j\setminus i} m_{k\to j}(y_j) \quad \forall (ij) \in E, \forall y_i, y_j$$

where $\phi_{ij}(y_i, y_j) = \phi_i(y_i)\phi_j(y_j)\psi_{ij}(y_i, y_j)$ and α is a normalization constant such that $\sum_{y_i,y_j} \mu_{ij}(y_i, y_j) = 1$.

4.1.2 Computing expectations of feature counts

While learning the parameters of a conditional Markov network we will also require the expectations of feature counts. Let $f(y_i)$ denote a feature that is

applied to every target random variable in the conditional Markov network. We would like to compute the expected value of $\sum_i f(y_i)$ under the distribution defined by the conditional Markov network. The brute force way of performing this computation is to compute:

$$E_{P(\mathbf{y}|\mathbf{x})}\left[\sum_i f(y_i)\right] = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \sum_i f(y_i)$$

The outer summation on $\mathbf{y}$ makes this computation difficult unless the Markov network consists of a very small number of nodes and we are dealing with very few class labels.

An easier way to compute the above quantity would be to first rewrite it using linearity of expectations so that it is expressed in terms of marginal probabilities:

$$\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \sum_i f(y_i) = \sum_{i,y_i} f(y_i) \sum_{\mathbf{y} \sim y_i} P(\mathbf{y}|\mathbf{x}) = \sum_{i,y_i} f(y_i)\mu_i(y_i) \tag{3}$$

where $\mathbf{y} \sim y_i$ denotes the complete labelings $\mathbf{y}$ that label Y_i with y_i, in other words, $\mathbf{y} \sim y_i$ can be read as 'complete assignments $\mathbf{y}$ that agree with the local assignment y_i'. Note that the summations in the final expression are linear in the number of nodes and the number of class labels, respectively, hence much easier to compute.

Thus to compute the expectations of feature counts, we can simply use LBP to estimate (approximate) marginal probabilities and use these estimates to compute (approximate) expectations of feature counts. Similarly, for a feature defined over the edges in the Markov network we can compute (approximate) expectations for feature counts using the edge marginals from LBP: $E_{P(\mathbf{y}|\mathbf{x})}[\sum_{ij} f(y_i, y_j)] = \sum_{(ij)\in E, y_i, y_j} f(y_i, y_j)\mu_{ij}(y_i, y_j)$.

Note that the only property we assumed in the above rewrites is that the features distribute over the Markov network as a sum. If the Markov network can be expressed as a product of clique potentials (Definition 4.1) and if each clique potential can be defined in terms of features (Eq. 1) then this assumption will always hold.

5 Cost-Sensitive Markov Networks by Minimizing Expected Cost of Misclassification

Traditionally, conditional Markov network classifiers aim to predict the labeling that maximizes $P(\mathbf{y}|\mathbf{x})$ (Definition 4.1). When we are dealing with varying misclassification costs, the above objective function does not make sense. In fact, given unseen test data, the optimal labeling in such a case is to compute the *Bayes optimal prediction* (Domingos, 1999) that minimizes the *conditional risk* (Duda et al., 2001) $R(\mathbf{y}|\mathbf{x})$:

$$R(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{y}'} P(\mathbf{y}'|\mathbf{x})\text{Cost}(\mathbf{y}, \mathbf{y}') \tag{4}$$

In the 0/1-loss setting, the parameters of a conditional Markov network $\{w_k\}_{k=1}^K$ are estimated by maximizing the probability of a fully labeled training set with correct class labels $\tilde{\mathbf{y}}$, e.g. by computing $\text{argmax}_w P(\tilde{\mathbf{y}}|\mathbf{x})$. In the case when we have varying misclassification costs, even this optimization does not make sense. We would like to learn a classifier that penalizes the configurations corresponding to higher misclassification costs which is an idea that has been proposed in various other work dealing with cost-sensitive learning (Domingos, 1999; Brefeld et al., 2003; Geibel and Wysotzki, 2003). One way to do this is to alter the objective function while learning the parameters $\{w_k\}_{k=1}^K$. Instead of maximizing the given labeling we will now minimize the weighted sum of probabilities corresponding to each labeling where each configuration's probability is weighted by its misclassification cost w.r.t. to the correct set of class labels:

$$\text{argmin}_w \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})\text{Cost}(\mathbf{y}, \tilde{\mathbf{y}}) \tag{5}$$

Since each configuration's probability is weighted by its misclassification cost, the higher the cost the more it contributes to the objective function. By minimizing the objective function we are thus going to minimize the probability mass allocated to configurations with higher misclassification costs in effect penalizing configurations with higher misclassification costs more than configurations with low misclassification costs.

Combining these two ideas we will get a classifier that uses costs both during learning and testing. We next derive algorithms that can perform the new learning and testing optimizations.

5.1 Learning

Given fully labeled training data, we would like to estimate the weight vector $\{w_k\}_{k=1}^K$ by performing the following optimization:

$$\text{argmin}_w \sum_{\mathbf{y}} \frac{1}{Z(\mathbf{x})} \exp\left[\sum_{c \in \mathcal{C}_G} \sum_{k=1}^{K} w_k f_k(x_c, y_c)\right] \text{Cost}(\mathbf{y}, \tilde{\mathbf{y}}) \tag{6}$$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \exp\left[\sum_{c \in \mathcal{C}_G} \sum_{k=1}^{K} w_k f_k(x_c, y_c)\right]$ and $\tilde{\mathbf{y}}$ is the labeling on the training data.

A simple approach to learning conditional Markov networks is to utilize one of the various first-order gradient-based optimization methods, e.g., gradient descent, conjugate-gradient descent, etc. (Taskar et al., 2002). We will follow the same approach. The key step to utilizing gradient-based optimization methods is to show how to compute the gradient of the objective function.

To compute the gradient, we take log and differentiate w.r.t. w_k.

$$\frac{\partial}{\partial w_k}\left[\log\left\{\sum_{\mathbf{y}}\text{Cost}(\mathbf{y},\tilde{\mathbf{y}})\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y_c)\right)\right\}-\log Z(\mathbf{x})\right]$$
$$= \frac{\partial}{\partial w_k}\log\left\{\sum_{\mathbf{y}}\text{Cost}(\mathbf{y},\tilde{\mathbf{y}})\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y_c)\right)\right\}-\frac{\partial\log Z(\mathbf{x})}{\partial w_k} \quad (7)$$

The first term in the above expression reduces to:

$$\frac{\partial}{\partial w_k}\log\left\{\sum_{\mathbf{y}}\text{Cost}(\mathbf{y},\tilde{\mathbf{y}})\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y_c)\right)\right\}=$$
$$\sum_{\mathbf{y}}\frac{\text{Cost}(\mathbf{y},\tilde{\mathbf{y}})\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y_c)\right)\sum_{c\in\mathcal{C}_G}f_k(x_c,y_c)}{\sum_{\mathbf{y}'}\text{Cost}(\mathbf{y}',\tilde{\mathbf{y}})\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y'_c)\right)} \quad (8)$$

which is simply the expectation of $\sum_{c\in\mathcal{C}_G}f_k(x_c,y_c)$ under the following distribution:

$$Q_1(\mathbf{y}|\mathbf{x})\propto\text{Cost}(\mathbf{y},\tilde{\mathbf{y}})\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y_c)\right)$$

such that $Q_1(\mathbf{y}|\mathbf{x})\geq 0$, $\sum_{\mathbf{y}}Q_1(\mathbf{y}|\mathbf{x})=1$.

The second term in Eq. 7 (after expanding $Z(\mathbf{x})$) reduces to:

$$\frac{\partial\log Z(\mathbf{x})}{\partial w_k}=$$
$$\sum_{\mathbf{y}}\frac{\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y_c)\right)\sum_{c\in\mathcal{C}_G}f_k(x_c,y_c)}{\sum_{\mathbf{y}'}\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y'_c)\right)}$$

which is the expectation of $\sum_{c\in\mathcal{C}_G}f_k(x_c,y_c)$ under the distribution $P(\mathbf{y}|\mathbf{x})$. Thus Eq. 7 can be expressed as:

$$\frac{\partial}{\partial w_k}\left[\log\left\{\sum_{\mathbf{y}}\text{Cost}(\mathbf{y},\tilde{\mathbf{y}})\exp\left(\sum_{c\in\mathcal{C}_G}\sum_{k=1}^{K}w_k f_k(x_c,y_c)\right)\right\}-\log Z(\mathbf{x})\right]$$
$$= E_{Q_1(\mathbf{y}|\mathbf{x})}[\sum_{c\in\mathcal{C}_G}f_k(x_c,y_c)]-E_{P(\mathbf{y}|\mathbf{x})}[\sum_{c\in\mathcal{C}_G}f_k(x_c,y_c)]$$

We have already seen how to use approximate inference techniques to compute the expectations of feature counts (Section 4.1). We can use those techniques to compute the expectations under the distribution $P(\mathbf{y}|\mathbf{x})$. Unfortunately, the expectations under the distribution $Q_1(\mathbf{y}|\mathbf{x})$ cannot be computed by the techniques described in Section 4.1 because the probability under the

distribution is not in product form due the presence of the term $\text{Cost}(\mathbf{y}, \tilde{\mathbf{y}})$. $\text{Cost}(\mathbf{y}, \tilde{\mathbf{y}})$ does not distribute as a product over the cliques in the network and does not match a Markov network probability distribution. We get around this issue by applying Jensen's inequality and making a lower bound approximation that allows us to approximate $\text{Cost}(\mathbf{y}, \tilde{\mathbf{y}})$ as a product of terms:

$$\text{Cost}(\mathbf{y}, \tilde{\mathbf{y}}) = \sum_{c \in \mathcal{C}_G} \text{Cost}_{c \in \mathcal{C}_G}(y_c, \tilde{y}_c) \geq |\mathcal{C}_G| \prod_{c \in \mathcal{C}_G} \text{Cost}_{c \in \mathcal{C}_G}(y_c, \tilde{y}_c)^{1/|\mathcal{C}_G|} \tag{9}$$

where $|\mathcal{C}_G|$ denotes the number of cliques in the network.

With this approximation we can now apply the techniques described in Section 4.1 and thus compute the gradient.

5.2 Inference

Given unseen test data and the parameters of a conditional Markov network we would like to compute:

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

Note that the set of conditional probabilities $P(\mathbf{y}'|\mathbf{x})$ required in the above equation can be quite large – so large that no classifier might be able to list them out. For instance, consider a conditional Markov network with 100 target random variables each of which can be assigned one of 10 class labels. The number of complete assignments (and hence the number of conditional probabilities $P(\mathbf{y}'|\mathbf{x})$) for this network is 10^{100}. In general, producing a listing of all the conditional probabilities $P(\mathbf{y}'|\mathbf{x})$ will only be possible for the smallest of conditional Markov networks. On the other hand, the number of marginal probabilities for each target random variable Y in the network is linear in the number of class labels and the size of the network (in the above example 10×100). In such a case, it is more feasible to work with the marginal probabilities (by first computing them using an approximate inference method) instead of the conditional probabilities $P(\mathbf{y}'|\mathbf{x})$. Thus it is more useful to rewrite the above expression in terms of the marginals:

$$\text{argmin}_{\mathbf{y}} \sum_{c \in \mathcal{C}_G, y'_c} \text{Cost}_c(y_c, y'_c) \mu_c(y'_c \mid \mathbf{x}) \tag{10}$$

where $\mu_c(y'_c \mid \mathbf{x}) = \sum_{\mathbf{y}' \sim y'_c} P(\mathbf{y}' \mid \mathbf{x})$ and $\mathbf{y}' \sim y'_c$ denotes a full assignment $\mathbf{y}'$ consistent with partial assignment y'_c.

Any energy minimization technique (exact or approximate) can be used to perform the above optimization. A simple way to achieve this is to utilize the methods described in Section 4.1. Consider the following probability distribution:

$$Q_2(\mathbf{y}|\mathbf{x}) \propto \exp\left[- \sum_{c \in \mathcal{C}_G, y'_c} \text{Cost}_c(y_c, y'_c) \mu_c(y'_c \mid \mathbf{x}) \right] \tag{11}$$

where $Q_2(\mathbf{y}|\mathbf{x}) \geq 0$ and $\sum_{\mathbf{y}} Q_2(\mathbf{y}|\mathbf{x}) = 1$.

Notice that determining the configuration $\mathbf{y}$ with the lowest value of Eq. 10 corresponds to determining the configuration corresponding to highest probability under the distribution $Q_2(\mathbf{y}|\mathbf{x})$. Since $Q_2(\mathbf{y}|\mathbf{x})$ distributes as a product over the cliques of the Markov network we can utilize the techniques described in Section 4.1 to find the most probable configuration. This gives us a simple two-step inference procedure where we first estimate the marginals under $P(\mathbf{y}|\mathbf{x})$ and then estimate the cost-sensitive labeling using those marginals. Both steps can be performed using techniques described in Section 4.1.

5.3 Issues

The problem with the above approach is that it requires accurate estimates of the marginal probabilities (Abe et al., 2004). Note that estimating class conditional probabilities is a harder problem than 0/1-loss classification since 0/1-loss classification only requires that we identify the class with the maximum class conditional probability and need not estimate the probabilities accurately. 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. Instead of minimizing expected cost of misclassification, 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. (2003) 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 conditional Markov network classifier which penalizes labelings associated with high misclassification costs based on maximum entropy principles.

6 Cost-Sensitive Markov Networks

To come up with a structured cost-sensitive classifier that avoids the use of probability estimates we need to go in a different direction. We will begin by taking a closer look at the constraints in the maximum entropy formulation that forms the basis of conditional Markov networks. The essential 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:

$$\sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \sum_c \underbrace{(f_k(x_c, y_c) - f_k(x_c, \tilde{y}_c))}_{\text{clique specific constraint}} = 0, \qquad \forall k = 1, \ldots, K \tag{12}$$

Eq. 12 attempts to set the sum of $P(\mathbf{y}|\mathbf{x})$ weighted by the difference in total feature counts to 0. We would now like to modify Eq. 12 so that it involves the misclassification costs into the maximum entropy formulation. 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. 12 to:

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

The new maximum entropy formulation can now be expressed as:

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

The dual of the above maximum entropy formulation is:

$$\min \ \log \left[\sum_{\mathbf{y}'} \exp \left(\sum_{k,c} w_k l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \right) \right] \tag{13}$$

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

Eq. 13 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.

6.1 Learning

Given fully labeled training data, we can learn the above model by minimizing Eq. 13 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.
Differentiating with respect to w_k we get:

$$\sum_{\mathbf{y}'} \left[\frac{1}{Z} \exp\left(\sum_k w_k \sum_c l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \right) \right.$$
$$\left. \sum_c l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \right] \quad (14)$$

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. 14) computation as a standard inference problem, let us now define the following probability distribution:

$$Q_3(\mathbf{y}') \propto \exp\left(\sum_k w_k \sum_c l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c)) \right)$$

such that $Q_3(\mathbf{y}') \geq 0$ and $\sum_{\mathbf{y}'} Q_3(\mathbf{y}') = 1$.

The gradient in Eq. 14 is thus simply the expectation of $\sum_c l_c(y'_c, \tilde{y}_c)(f_k(x_c, y'_c) - f_k(x_c, \tilde{y}_c))$ under the distribution $Q_3(\mathbf{y}')$ which can be computed using techniques described in Section 4.1.

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.

6.2 Inference

For inference, we need to determine the optimal labeling $\mathbf{y}$ which minimizes Eq. 13:

$$\text{argmin}_{\mathbf{y}} \log \left[\sum_{\mathbf{y}'} \exp\{ \sum_{k,c} w_k l_c(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c))\} \right] \quad (15)$$

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. In such cases, we resort to approximate inference.

In order to obtain a lower bound approximation we apply Jensen's inequality to Eq. 15:

$$\begin{aligned}
&\log\left[\sum_{\mathbf{y}'}\exp\{\sum_{k,c} w_k l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c))\}\right] \\
&\quad \geq \sum_{\mathbf{y}'} Q_4(\mathbf{y}') \sum_k w_k \sum_c l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) \\
&\quad\quad - \sum_{\mathbf{y}'} Q_4(\mathbf{y}') \log Q_4(\mathbf{y}') \qquad (16)
\end{aligned}$$

where $Q_4(\mathbf{y}')$ is a distribution over all $\mathbf{y}'$ ($\sum_{\mathbf{y}'} Q_4(\mathbf{y}') = 1$, $Q_4(\mathbf{y}') \geq 0$) and we show how to compute it subsequently.

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. 16 w.r.t $Q_4(\mathbf{y}')$ 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. 16 is:

$$\begin{aligned}
&\sum_{\mathbf{y}'} Q_4(\mathbf{y}') \sum_k w_k \sum_c l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) \\
&- \sum_{\mathbf{y}'} Q_4(\mathbf{y}') \log Q_4(\mathbf{y}') - \tau\left(\sum_{\mathbf{y}'} Q_4(\mathbf{y}') - 1\right) \qquad (17)
\end{aligned}$$

To maximize w.r.t to $Q_4(\mathbf{y}')$, we differentiate with respect to $Q_4(\mathbf{y}')$ and set the derivative to 0 to get:

$$Q_4(\mathbf{y}') \propto \exp\left[\sum_k w_k \sum_c l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c))\right]$$

where $\sum_{\mathbf{y}'} Q_4(\mathbf{y}') = 1$.

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. 17 (since this is the only term which involves $\mathbf{y}$):

$$\begin{aligned}
&\sum_{\mathbf{y}'} Q_4(\mathbf{y}') \sum_k w_k \sum_c l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c)) = \\
&\qquad \sum_{y'_c, c} \sum_k w_k l(y'_c, y_c)(f_k(x_c, y'_c) - f_k(x_c, y_c))\mu_c^{Q_4}(y'_c) \qquad (18)
\end{aligned}$$

where $\mu_c^{Q_4}(y'_c)$ is the marginal probability of labeling clique c with y'_c under the $Q_4(\mathbf{y}')$ distribution.

One way to minimize Eq. 18 w.r.t. $\mathbf{y}$ is to define the following distribution and determine the configuration corresponding to the highest probability:

$$Q_5(\mathbf{y}) \propto \exp\left[\sum_{y'_c,c}\sum_{k} w_k l(y'_c, y_c)(f_k(x_c, y_c) - f_k(x_c, y'_c))\mu_c^{Q_4}(y'_c)\right]$$

where $\sum_{\mathbf{y}} Q_5(\mathbf{y}) = 1, Q_5(\mathbf{y}) \geq 0$. To determine the $\mathbf{y}$ corresponding to the maximum $Q_5(\mathbf{y})$, we can use inference algorithms.

6.3 Inference for Cost-Sensitive Pairwise Markov Networks

Algorithm 1 describes the pseudo-code for the above inference procedure for pairwise Markov networks using loopy belief propagation (Section 4.1). The pseudo-code contains comments that clearly demarcate the use of loopy belief propagation and we hope that this will aid the user who wants to use the cost-sensitive inference procedure but may want to substitute LBP with an alternate inference procedure.

7 Experiments

We performed experiments on synthetic random graph data and real-world sensor network data to substantiate the following claims: misclassification costs can be reduced by exploiting correlations across links and structured cost-sensitive classifiers perform better than traditional machine learning cost-sensitive classifiers, the probability estimates of 0/1-loss structured classifiers using approximate inference methods may not be dependable and thus utilizing techniques that require accurate estimates of class-conditional probabilities may return poor results, and structured cost-sensitive classifiers can exploit structured cost functions to return decreased misclassification costs.

In all our experiments we compare misclassification costs achieved by four different classifiers: *LOGREG* - logistic regression which classifies each sample based on the attributes followed by minimization of expected cost of misclassification (the minimization of expected cost of misclassification is simply a wrapper around standard logistic regression), *MN* - conditional Markov networks followed by minimization of expected cost of misclassification, *CSMN1* conditional Markov networks with the extensions described in Section 5 and *CSMN2* - the classifier described in Section 6.

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 $p(w_k) \propto \exp(-\lambda w_k^2)$. 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 examples in the training set) gave the best results (which matches with Zhang

Algorithm 1 Cost-sensitive inference for pairwise cost-sensitive Markov network $G(\mathbf{V}, E)$ using loopy belief propagation.

Let $\mathbf{y}$ denote our best guess of the labels
Initialize $\mathbf{y}$ randomly
num_itn $\leftarrow 0$
while $\mathbf{y}$ shows change and num_itn $<$ MAX_ITN **do**
  {Estimate Q_4 using loopy belief propagation}
  num_Q_4_itn $\leftarrow 0$
  while messages $m_{i\to j}^{Q_4}$ show change and num_Q_4_itn $<$ MAX_Q_4_ITN **do**
    $m_{i\to j}^{Q_4}(y'_j) = \alpha \sum_{y'_i} \phi_i^{Q_4}(y'_i)\psi_{ij}^{Q_4}(y'_i, y'_j) \prod_{h\in\mathcal{N}_i\setminus j} m_{h\to i}^{Q_4}(y'_i), \forall y'_j \forall (ij) \in E$
    where $\phi_i^{Q_4}(y'_i) = \prod_{x,st.(xi)\in E} \exp[\sum_k w_k l(y'_i, y_i)(f_k(x, y'_i) - f_k(x, y_i))]$
    $\psi_{ij}^{Q_4}(y'_i, y'_j) = \exp\left[\sum_k w_k l(\{y'_i, y'_j\}, \{y_i, y_j\})(f_k(\{y'_i, y'_j\}) - f_k(\{y_i, y_j\}))\right]$
    α is a normalization constant s.t. $\sum_{y'_j} m_{i\to j}^{Q_4}(y'_j) = 1$
    num_Q_4_itn $\leftarrow$ num_Q_4_itn $+ 1$
  end while
  $\mu_i^{Q_4}(y'_i) = \alpha\phi_i^{Q_4}(y'_i) \prod_{j\in\mathcal{N}_i} m_{j\to i}^{Q_4}(y'_i), \forall y'_i \forall i$
  $\mu_{ij}^{Q_4}(y'_i, y'_j) = \alpha\phi_{ij}^{Q_4}(y'_i, y'_j) \prod_{h\in\mathcal{N}_i\setminus j} m_{h\to i}^{Q_4}(y'_i) \prod_{h\in\mathcal{N}_j\setminus i} m_{h\to j}^{Q_4}(y'_j), \forall y'_i, y'_j, \forall (ij)$
  where $\phi_{ij}^{Q_4}(y'_i, y'_j) = \phi_i^{Q_4}(y'_i)\phi_j^{Q_4}(y'_j)\psi_{ij}^{Q_4}(y'_i, y'_j)$
  {Estimate Q_5 using loopy belief propagation}
  num_Q_5_itn $\leftarrow 0$
  while messages $m_{i\to j}^{Q_5}$ show change and num_Q_5_itn $<$ MAX_Q_5_ITN **do**
    $m_{i\to j}^{Q_5}(y_j) = \alpha \sum_{y_i} \phi_i^{Q_5}(y_i)\psi_{ij}^{Q_5}(y_i, y_j) \prod_{h\in\mathcal{N}_i\setminus j} m_{h\to i}^{Q_5}(y_i), \forall y_j \forall (ij) \in E$
    where $\phi_i^{Q_5}(y_i) = \prod_{xst.(xi)\in E} \exp\left[\sum_{y'_i,k} w_k l(y'_i, y_i)(f_k(x, y_i) - f_k(x, y'_i))\mu_i^{Q_4}(y'_i)\right]$
    $\psi_{ij}^{Q_5}(y_i, y_j) = \exp\left[\sum_{y'_i,y'_j,k} w_k l(\{y'_i, y'_j\}, \{y_i, y_j\})(f_k(y_i, y_j) - f_k(y'_i, y'_j))\mu_{ij}^{Q_4}(y'_i, y'_j)\right]$
    α is a normalization constant s.t. $\sum_{y_j} m_{i\to j}^{Q_5}(y_j) = 1$
    num_Q_5_itn $\leftarrow$ num_Q_5_itn $+ 1$
  end while
  for each node i **do**
    $\mu_i^{Q_5}(y_i) = \alpha\phi_i^{Q_5}(y_i) \prod_{j\in\mathcal{N}_i} m_{j\to i}^{Q_5}(y_i), \forall y_i$
    Let $y_i \leftarrow \text{argmax}_y \mu_i^{Q_5}(y)$ and add y_i to $\mathbf{y}$
  end for
  num_itn $\leftarrow$ num_itn $+ 1$
end while

and Oles (2001)'s suggestion), for *MN* and *CSMN1* $\lambda = 10$ gave the best results (Taskar et al. (2002) report using a regularization constant of the same magnitude $\lambda \approx 5.5$) and for *CSMN2* $\lambda = 20$ returned the best results.

7.1 Synthetic Data Generation

Commonly available real-world networks exhibit properties like preferential attachment and correlations among 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 2.

The synthetic data generator (Algorithm 2) "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. We experimented with graphs of difference sizes and found the results to be similar. For the experiments reported here, we set $numNodes = 100$.

Algorithm 2 Synthetic data generator

SynthGraph($numNodes$, α, ρ)

$i \leftarrow 0$
$G \leftarrow \emptyset$
while $i < numNodes$ **do**
  sample $r \in [0, 1]$ uniformly at random
  if $r <= \alpha$ **then**
    $v \leftarrow$ select any node uniformly at random from G
    connectNode(v, G, ρ)
  else
    add a new node v to G
    choose $v.label$ from $\{0, 1\}$ uniformly at random
    connectNode(v, G, ρ)
    $i \leftarrow i + 1$
  end if
end while
for i = 1 to numNodes **do**
  $v \leftarrow i^{th}$ node in G
  genAttributes(v)
  genNodeCostMatrix(v)
end for
for each edge e in G **do**
  genEdgeCostMatrix(e)
end for
return G

We generated binary class data using our synthetic data generator; this

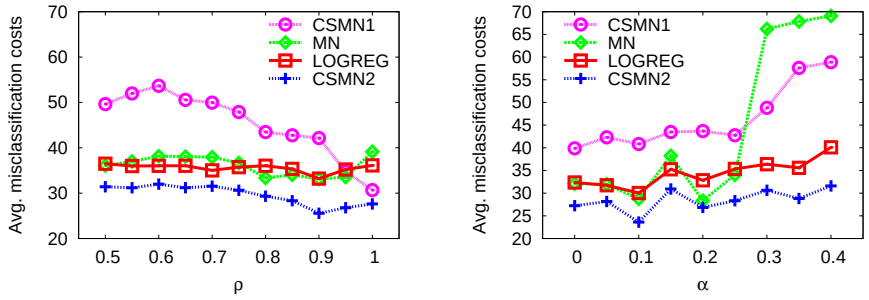
is a common case in cost-sensitive applications ("loan granted"/"loan denied", "good customer"/"bad customer", "terrorist"/"non-terrorist" etc.). Algorithm 2 proceeds through iterations and in each iteration it either connects a newly created node to the graph or connects two existing nodes in the graph. Each time Algorithm 2 creates an edge it makes a call to Algorithm 3. Algorithm 3 implements a rudimentary form of *preferential attachment* where a node can choose which nodes to link based on their labels. This introduces correlations among the labels across links. The strength 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 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 many 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. More specifically, for each node we generated 5 attribute values. Each attribute can either be present or absent (boolean-valued). We sampled for attribute ids from class-specific binomial distributions. In particular, for a specific node in the graph we begin by setting all 5 of its attribute values to 0. Then we sample from the binomial distribution with p=(1+c)/3, n=5 (where c denotes the class label of the node) and set the attribute with the sampled id to 1. We do this four times for each node.

Algorithm 3 Generating an edge in the synthetic data graph

connectNode(v, G, ρ)

1: sample r uniformly at random from $[0, 1]$
2: **if** $r \leq \rho$ **then**
3: $\quad c_n \leftarrow v.label$
4: **else**
5: $\quad c_n \leftarrow (v.label + 1)$ `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. Since we considered cliques only up to size 2 (nodes and edges), 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 $[0, \frac{\text{ham}(y_c, \tilde{y}_c)}{2}]$ 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*.



(a) (b)

Figure 1: (a) 10-fold cross validation results of varying ρ (X-axis). α was kept constant at 0.25.(b) 10-fold cross validation results of varying α (X-axis). ρ was kept constant at 0.85.

For each experiment, we produced 10 datasets and performed 10-fold cross validation. Each number we report is the average misclassification cost obtained. For all runs of *CSMN2*, we set the clique loss matrices equal to the cost matrices: $l_c(y_c, \tilde{y}_c) = \text{Cost}_c(y_c, \tilde{y}_c)$.

7.2 Performance Comparisons

For our first experiment, we varied the correlation among labels across links to find out if structured classifiers actually help decrease misclassification costs. We varied the value of ρ from 0.5 to 1.0 keeping α constant at 0.25. 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 labels.

Figure 1 (a) shows that structured classifiers (*CSMN2* and *MN*) can exploit correlations in the link structure to reduce misclassification costs. The plot shows that *CSMN2* manages to produce the lowest misclassification costs compared to all the other classifiers on all settings of ρ. *CSMN2* achieves 9.96% reduction in costs over *LOGREG* at $\rho = 0.7$ which increases to 23.5% at $\rho = 1.0$. Also, at $\rho = 0.75$, *CSMN2* achieves an avg. accuracy of 80.1% whereas the 0/1-loss *MN* achieves an avg. accuracy of 80.9% indicating that a higher avg. accuracy does not necessarily imply a lower avg. misclassification cost.

In Figure 1 (a), at $\rho = 1.0$, *MN* shows an increase in misclassification costs when it should be expected to utilize the correlations across links to show better performance. This is most likely the result of the inference algorithms used. As indicated earlier, we used loopy belief propagation (LBP) (Yedidia et al., 2000) for inference in our implementation as described in Section 4.1. LBP is a message passing algorithm that is known to return very poor estimates

of marginal 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 *CSMN2* avoids this pitfall because it does not rely on the explicit use of probabilities.

In Figure 1 (a), *CSMN1* tends to fare the worst among the four models and this may be due to the use of the approximation described in Eq. 9 while learning. The equality in the lower bound approximation holds when all costs are equal and the bound becomes progressively looser as this condition is violated. In the case where we have sample dependent randomly generated costs, this bound is not very tight thus causing problems during *CSMN1*'s learning phase. Despite this fact Figure 1 (a) shows that *CSMN1* performs better when the correlations between class labels across links aid cost-sensitive classification as indicated by the dip shown in the plot when ρ is increased. As we show in the experiments on real-world data, when the bound in Eq. 9 is tight and the network is sparse *CSMN1* can produce good results. Table 1 (a) shows the result of two-tailed paired t-tests comparing the performance of *CSMN2* with the other models. Note that since we compare misclassification costs, less is better. Bold values indicate 95% level of significance or above.

The previous experiment was performed on datasets with a constant number of edges ($\approx$ 133 edges in a graph of 100 nodes). We also wanted to see how varying edge density affects the performance of the classifiers. In our second experiment, we varied α from 0 to 0.4 and kept ρ 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 1 (b) shows the results. At $\alpha = 0.0$, the misclassification cost due to the misclassification of edges is small but this fraction increases as α increases. Since *LOGREG* does not care about the edge cost matrices (does not use them to compute the cost-sensitive classification, only uses them to compute the final misclassification cost) it performs well at $\alpha = 0.0$ but poorly at higher settings of α. *CSMN2* consistently returns the lowest misclassification costs. Increasing edge density increases the number of short cycles. At $\alpha = 0.3$ and higher, due to the large number of cycles in the graph, *MN* and *CSMN1*, due to their dependence on LBP, return inaccurate estimates of class conditional probabilities thus resulting in very poor results. Once again, due to the presence of sample dependent misclassification costs, Eq. 9 is not very tight and *CSMN1* does not produce very good results. Table 1 (b) shows the result of two-tailed paired t-tests comparing the performance of *CSMN2* with the other models for these experiments. Bold values indicate 95% level of significance or above.

7.3 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

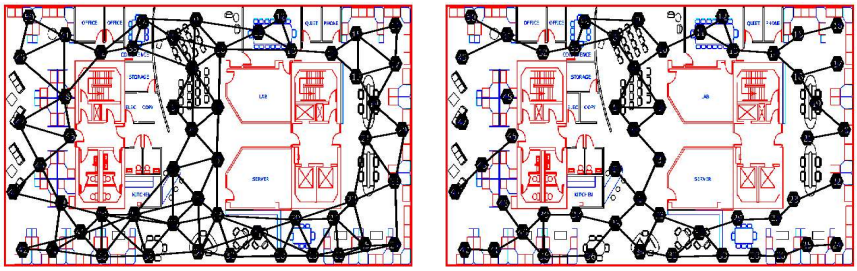
ρ	CSMN2 vs. LOGREG	CSMN2 vs. MN	CSMN2 vs. CSMN1
0.50	**-5.0**	**-3.5**	**-8.8**
0.55	**-4.1**	**-3.7**	**-12.5**
0.60	**-2.73**	**-3.3**	**-12.9**
0.65	**-4.1**	**-2.99**	**-2.5**
0.70	**-2.8**	**-2.33**	**-7.8**
0.75	**-2.8**	**-2.5**	**-9.9**
0.80	**-5.7**	-2.0	**-6.0**
0.85	**-4.6**	**-2.9**	**-7.5**
0.90	**-6.6**	**-2.4**	**-4.4**
0.95	**-3.0**	**-4.3**	**-2.3**
1.00	**-3.1**	**-4.4**	-1.0

(a)

α	CSMN2 vs. LOGREG	CSMN2 vs. MN	CSMN2 vs. CSMN1
0.00	**-3.2**	**-2.7**	**-6.4**
0.05	**-2.3**	-2.0	**-3.2**
0.10	**-6.1**	**-2.7**	**-5.1**
0.15	**-3.2**	**-2.4**	**-5.7**
0.20	**-2.6**	**-2.8**	**-6.4**
0.25	**-4.6**	**-2.9**	**-7.5**
0.30	**-5.9**	**-17.2**	**-3.8**
0.35	**-2.27**	**-7.2**	**-8.4**
0.40	**-3.0**	**-8.0**	**-9.5**

(b)

Table 1: Results of two-tailed paired t-tests for experiments on synthetic data with (a) varying ρ and (b) varying α. Bold values indicate 95% level of significance or more.

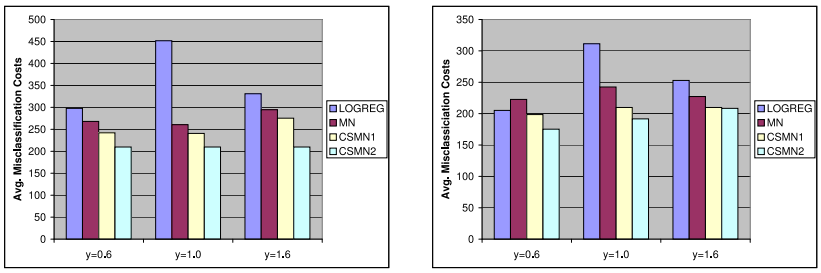


(a) (b)

Figure 2: Lab from which the Intel lab dataset (Bodik et al., 2004) was collected. The figures show the lab layout at two different edge densities with sensors and edges shown in black. (a) Network obtained by connecting each pair of sensors within 7 meters of each other. (b) Network obtained by connecting each pair of sensors within 5 meters of each other.

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 d meters of each other and experimented with two settings: $d = 7$ (Figure 2 (a)) and $d = 5$ (Figure 2 (b)). 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 10 sets containing, roughly, the same distribution of values for the light attribute (*high_light*, *medium_light* and *low_light*) and performed 10-fold cross validation. Unfortunately, the dataset does not come with cost matrices. We augmented the dataset with, hopefully realistic, label-dependent 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 γ to define the cost due to occupant discomfort in units of electricity and defined the cost matrices in terms of γ. See Appendix A for the complete cost matrices used in the following experiments. 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 γ; if the predicted value of light is *low_light* and the correct class label is *medium_light* (cost due to excess electricity usage) then we pay a cost of 1. This gave us a 3×3 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:



(a) (b)

Figure 3: (a) 10-fold cross validation results on *Intel lab* (Bodik et al., 2004) dataset, X-axis shows various settings of γ when $d = 7$. (b) 10-fold cross validation results on *Intel lab* (Bodik et al., 2004) dataset, X-axis shows various settings of γ when $d = 5$.

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 γ (occupant discomfort caused due to lack of light in a location near the occupied location). This gave us a 9×9 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 γ is a difficult problem. Intuitively, if we set γ 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 γ 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*, *CSMN1* and *CSMN2* at three different settings of γ (Figure 3).

As Figure 3 shows, *CSMN2* produces the lowest average misclassification costs on both settings $d = 7$ and $d = 5$. At $\gamma = 1$ and $d = 7$, *CSMN2* achieved a 53.5% reduction in average misclassification costs over *LOGREG*, a 19.4% reduction in average misclassification costs over *MN* and 12.8% reduction in average misclassification costs over *CSMN1*. Further, *CSMN1* does much better in these experiments compared to the experiments with synthetic data we reported earlier since the use of label-dependent misclassification costs (as opposed to sample-dependent misclassification costs) introduces a number of cliques with similar costs thus allowing the lower bound approximation (Eq. 9) to be much tighter. At $d = 5$ and $\gamma = 0.6$, *LOGREG* shows better performance than *MN* and this may be due to the fact that the misclassification costs due to the edges are only incurred due to occupant discomfort and the cost due to occupant discomfort is small when γ is set to a small number (like 0.6) thus

allowing a classifier such as *LOGREG* to get away with only classifying the nodes correctly.

The results at $d = 7$ (Figure 3 (a)) show that when the edge density is higher *CSMN2* still holds an advantage over the other two structured classifiers because *MN* and *CSMN1* still depend on probability estimation which is difficult in dense Markov networks. This is further corroborated by the fact that the differences in results are less pronounced when $d = 5$ (Figure 3 (b)) where, due to the sparse network structure, inference is easier and *MN* and *CSMN1* perform much better producing results close to *CSMN2*'s; in fact, at $\gamma = 1.6$ and $d = 5$ *CSMN1* produces almost identical results to *CSMN2*'s across all 10 splits. Table 2 shows the results of two-tailed paired t-tests for these experiments.

8 Discussion and 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 may also be structured. Existing unstructured IID cost-sensitive classification methods do not provide a natural means to handle structured cost functions. We proposed an approach based on Bayes optimal prediction extending existing 0/1-loss structured classifiers and IID cost-sensitive classifiers that 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 quantify the conditions when the proposed approaches work well and to show that they can lead to significant reductions in misclassification costs.

Our experiments show that structured cost-sensitive classifiers can achieve significant improvements compared to their unstructured counterparts and that these benefits are higher when the graph structure is sparse and accurate inference is feasible. It may not always be possible to judge just by looking at the data whether the graph structure is too dense for approximate inference algorithms to produce probability estimates that are accurate enough and for this reason we believe the cost-sensitive classifier proposed in Section 6 (*CSMN2*) is a more prudent alternative because in all our experiments it produced equivalent, if not better, results than the other cost-sensitive models we tried. On the other hand, recall that in this paper we only concentrated on problems whose cost graph was identical to the underlying Markov network. It is easy to see that this assumption is only a limitation for the cost-sensitive Markov network presented in Section 6 and not for the classifier presented in Section 5 (*CSMN1*). If we are faced with an application that has special properties that allow the use of accurate inference methods (e.g., the application produces sequences only) then it may be a good idea to utilize the model presented in Section 5 since it is based on Bayes optimal prediction which is guaranteed to return optimal results (Domingos, 1999).

As indicated in Section 3, research in the structured classifiers has been grow-

CSMN2 vs.	$\gamma = 0.6$	$\gamma = 1.0$	$\gamma = 1.6$
LOGREG	**-2.6**	**-5.4**	**-3.4**
MN	**-2.4**	**-2.3**	**-4.9**
CSMN1	**-2.6**	-2.0	**-4.7**

(a)

CSMN2 vs.	$\gamma = 0.6$	$\gamma = 1.0$	$\gamma = 1.6$
LOGREG	-2.13	**-4.8**	**-3.2**
MN	**-2.6**	**-3.0**	-1.3
CSMN1	-1.6	-1.9	-0.1

(b)

Table 2: Results of two-tailed paired t-tests for experiments on *Intel lab* (Bodik et al., 2004) dataset with (a) $d = 7$ and (b) $d = 5$. Bold values indicate 95% level of significance or above.

ing at a rapid pace and a number of IID classifiers have now been extended to handle structured inputs. Most notably, max-margin Markov networks (Taskar et al., 2003) and associative Markov networks (Taskar et al., 2004) are extensions of support vector machines that assume that each misclassification is equally costly. An interesting line of future research would be to develop cost-sensitive versions of these classifiers so that they utilize varying misclassification costs during inference. Another line of future work would be to apply the proposed classifiers to newer domains. Research in intelligent light control for buildings has benefited greatly in the recent past due to advances in sensor network technologies and we believe there are significant opportunities to apply cost-sensitive decision theoretic models in this area. Another exciting area for cost-sensitive applications is the field of social networks arising from various domains such as counter-terrorism where we would like to classify people as terrorists or non-terrorists and different people are connected via communication links but the cost of misclassifying a terrorist as a non-terrorist is different from the cost of misclassifying a non-terrorist as a terrorist. Given the diversity of the domains that give rise to data requiring classification in the presence of varying misclassification costs, we believe, there is room for considerable research to be done in the area of structured cost-sensitive classification.

Acknowledgements: This work was supported by the National Science Foundation (NSF #0423845). The authors would also like to thank all the anonymous reviewers for their helpful comments.

References

N. Abe, B. Zadrozny, and J. Langford. An iterative method for multiclass cost-sensitive learning. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2004.

C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo codes. In *Proceedings of IEEE International Communications Conference*, 1993.

J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 1986.

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

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

J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. Brodley. Pruning decision trees with misclassification costs. In *Proceedings of the European Conference on Machine Learning*, 1998.

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

S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *International Conference on Management of Data*, 1998.

P. Chan and S. Stolfo. Toward scalable learning with non-uniform class and cost distributions. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1998.

D. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *Advances in Neural Information Processing Systems*, 2001.

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

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

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, 2001.

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

G. Fumera and F. Roli. Cost-sensitive learning in support vector machines. In *Convegno Associazione Italiana per L'Intelligenza Artificale*, 2002.

P. Geibel and F. Wysotzki. Perceptron based learning with example dependent and noisy costs. In *Proceedings of the International Conference on Machine Learning*, 2003.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 2002.

R. Hummel and S. Zucker. On the foundations of relaxation labeling processes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983.

E. T. Jaynes and R. D. Rosenkrantz (ed.). *E. T. Jaynes: Papers on Probability, Statistics and Statistical Physics*. Springer, 2003.

U. Knoll, G. Nakhaeizadeh, and B. Tausend. Cost-sensitive pruning of decision trees. In *Proceedings of the European Conference on Machine Learning*, 1994.

F. R. Kschischang and B. J. Frey. Iterative decoding of compound codes by probability progation in graphical models. *IEEE Journal on Selected Areas in Communication*, 1998.

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

Q. Lu and L. Getoor. Link based classification. In *Proceedings of the International Conference on Machine Learning*, 2003.

R. J. McEliece, D. J. C. MacKay, and J. F. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE Journal on Selected Areas in Communication*, 1998.

T. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, 2001.

K. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, 1999.

J. Neville and D. Jensen. Iterative classification in relational data. In *AAAI Workshop on Learning Statistical Models from Relational Data*, 2000.

P. Sen and L. Getoor. Cost-sensitive learning with conditional markov networks. In *Proceedings of the International Conference on Machine Learning*, 2006.

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

S. Slattery and M. Craven. Combining statistical and relational methods for learning in hypertext domains. In *International Conference on Inductive Logic Programming*, 1998.

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

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proceedings of the International Conference on Machine Learning*, 2004.

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

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

L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans. Discriminative unsupervised learning of structured predictors. In *Proceedings of the International Conference on Machine Learning*, 2006.

J. Yedidia, W.T.Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, 2000.

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

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

B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the IEEE International Conference on Data Mining*, 2003.

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

A Cost Matrices for Experiments with Sensor Network Data

A.1 Node Cost Matrix

Table 3 shows the complete node cost matrix used for the Intel Lab data experiments. Rows indicate assigned class labels and columns indicate true class

	l	m	h
l	0	1	2
m	γ	0	1
h	γ	γ	0

Table 3: Node cost matrix used for sensor network data experiments. Each class label is abbreviated to its first letter, e.g., 'l' denotes *low_light*.

labels. For instance, row 1 column 3 provides the misclassification cost incurred when we misclassify a location as having low light when the true class label is high light. In this case what can happen is that, an occupant might come in and the building control system might still turn on lights even though there is sufficient light to begin with thus incurring unwanted electricity costs. When the misclassification is 'milder', a location with high light being labeled medium then we only incur a cost of 1 (row 2 column 3). On the other hand, when a location with low light is assigned the label high light then the control system may not turn on lights even if an occupant occupies the location thus leading to occupant discomfort, incurring a cost of γ (row 3 column 1).

A.2 Edge Cost Matrix

Table 4 shows the complete edge cost matrix used for the sensor network data experiments. The edge cost matrix assumes that all occupants would like to have a medium light or high light area nearby. Each row/column is annotated with 2 class labels, one each for two locations next to one another. Once again, rows indicate assigned labels whereas columns indicate true labels. As opposed to the node cost matrix, the edge cost matrix only returns cost due to occupant discomfort (the excess electricity cost will be returned by the node cost matrices). For instance, for row 1 column 2, the second location is misclassified as having low light when in fact it has medium light which means that the control system is going to turn on lights if an occupant occupies the first location thus incurring no occupant discomfort cost indicated by a 0 in the edge cost matrix (note that this misclassification will still incur an excess electricity cost of 1 from the node cost matrix). Also, for row 4 column 2, if an occupant comes into the second location then the control system will not turn on lights at the first location even though it should (since the true label is *low_light*) thus incurring occupant discomfort cost.

	(l,l)	(l,m)	(l,h)	(m,l)	(m,m)	(m,h)	(h,l)	(h,m)	(h,h)
(l,l)	0	0	0	0	0	0	0	0	0
(l,m)	γ	0	0	γ	0	0	γ	0	0
(l,h)	γ	0	0	γ	0	0	γ	0	0
(m,l)	γ	γ	γ	0	0	0	0	0	0
(m,m)	γ	γ	γ	γ	0	0	γ	0	0
(m,h)	γ	γ	γ	γ	0	0	γ	0	0
(h,l)	γ	γ	γ	0	0	0	0	0	0
(h,m)	γ	γ	γ	γ	0	0	γ	0	0
(h,h)	γ	γ	γ	γ	0	0	γ	0	0

Table 4: Edge cost matrix used for sensor network data experiments. Each row/column is annotated with a pair of class labels and each class label is abbreviated to its first letter, e.g., '(m,h)' represents (*medium_light, high_light*).