Probabilistic Classification and Clustering in Relational Data

Ben Taskar
Computer Science Dept.
Stanford University
Stanford, CA 94305
btaskar@cs.stanford.edu

Eran Segal
Computer Science Dept.
Stanford University
Stanford, CA 94305
erans@cs.stanford.edu

Daphne Koller
Computer Science Dept.
Stanford University
Stanford, CA 94305
koller@cs.stanford.edu

Abstract

Supervised and unsupervised learning methods have traditionally focused on data consisting of *independent* instances of a single type. However, many real-world domains are best described by relational models in which instances of multiple types are *related* to each other in complex ways. For example, in a scientific paper domain, papers are related to each other via citation, and are also related to their authors. In this case, the label of one entity (e.g., the topic of the paper) is often correlated with the labels of related entities. We propose a general class of models for classification and clustering in relational domains that capture probabilistic dependencies between related instances. We show how to learn such models efficiently from data. We present empirical results on two real world data sets. Our experiments in a *transductive* classification setting indicate that accuracy can be significantly improved by modeling relational dependencies. Our algorithm automatically induces a very natural behavior, where our knowledge about one instance helps us classify related ones, which in turn help us classify others. In an unsupervised setting, our models produced coherent clusters with a very natural interpretation, even for instance types that do not have any attributes.

1 Introduction

Most supervised and unsupervised learning methods assume that data instances are independent and identically distributed (IID). Numerous classification and clustering approaches have been designed to work on such "flat" data, where each data instance is a fixed-length vector of attribute values (see [Duda *et al.*, 2000] for a survey). However, many real-world data sets are much richer in structure, involving instances of multiple types that are related to each other. Hypertext is one example, where web pages are connected by links. Another example is a domain of scientific papers, where papers are related to each other via citation, and are also related to their authors. The IID assumption is clearly violated for two papers written by the same author or two papers linked by citation, which are likely to have the same topic.

Recently, there has been a growing interest in learning techniques for more richly structured datasets. Relational links between instances provide a unique source of information that has been proved useful for both classification and clustering in the hypertext domain [Slattery and Craven, 1998; Kleinberg, 1998]. Intuitively, relational learning methods attempt to use our knowledge about one object to reach conclusions about other, related objects. For example, we would like to propagate information about the topic of a paper p to papers that it cites. These, in turn, would propagate information to papers that they cite. We would also like to use information about p's topic to help us reach conclusion about the research area of p's author, and about the topics of other papers written by that author.

Several authors have proposed relational classification methods along the lines of this "influence propagation" idea. Neville and Jensen [2000] present an *iterative classification* algorithm which essentially implements this process exactly, by iteratively assigning labels to test instances the classifier is confident about, and using these labels to classify related instances. Slattery and Mitchell [2000] propose an iterative algorithm called FOIL-HUBS for the problem of classifying web pages, e.g., as belonging to a university student or not. However, none of these approaches proposes a single coherent model of the correlations between different related instances. Hence they are forced to provide a purely procedural approach, where the results of different classification steps or algorithms are combined without a unifying principle.

In clustering, the emphasis so far has been on *dyadic* data, such as word-document co-occurrence [Hofmann and Puzicha, 1999], document citations [Cohn and Chang, 2000], web links [Cohn and Hofmann, 2001; Kleinberg, 1998], and gene expression data. Kleinberg's "Hubs and Authorities" algorithm exploits the link structure to define a mutually reinforcing relationship between hub and authority pages, where a good hub page points to many good authorities and a good authority page is pointed to by many good hubs.

These techniques can be viewed as relational clustering methods for one or two "types" of instances (e.g., web pages, documents and words), with a single relation between them (e.g., hyperlinks, word occurrence). However, we would like to model richer structures present in many real world domains with multiple types of instances and complex relationships between them. For example, in a movie database the instance types might be movies, actors, directors, and producers. Instances of the same type may also be directly related. In a scientific paper database, a paper is described by its set of words and its relations to the papers it cites (as well as to the authors who wrote it). We would like to identify, for each instance type, sub-populations (or segments) of instances that are similar in both their attributes and their relations to other instances.

In this paper, we propose a general class of generative probabilistic models for classification and clustering in relational data. The key to our approach is the use of a single probabilistic model for the entire database that captures interactions between instances in the domain. Our work builds on the framework of *Probabilistic Relational Models (PRMs)* of Koller and Pfeffer [1998] that extend Bayesian networks to a relational setting. PRMs provide a language that allows us to capture probabilistic dependencies between related instances in a coherent way. In particular, we use it to allow dependencies between the class variables of related instances, providing a principled mechanism for propagating information between them.

Like all generative probabilistic models, our models accommodate the entire spectrum between purely supervised classification and purely unsupervised clustering. Thus, we can learn from data where some instances have a class label and other do not. We can also deal with cases where one (or more) of the instance types does not have an observed class attribute by introducing a new latent class variable to represent the (unobserved) cluster. Note that, in relational models, it is often impossible to segment the data into a training and test set that are *independent* of each other since the training and test instances may be interconnected. Using naive random sampling to select training instances is very likely to sever links between instances in the training and test set data. We circumvent this difficulty by using a transductive learning setting, where we use the test data, albeit without the labels, in the training phase. Hence, even if all the instance types have observed class attributes, the training phase involves learning with latent variables.

We provide an approximate EM algorithm for learning such PRMs with latent variables from a relational database. This task is quite complex: Our models induce a complex web of dependencies between the latent variables of all of the entities in the data, rendering standard approaches intractable. We provide an efficient approximate algorithm that scales linearly with the number of instances, and thus can be applied to large data sets.

We present experimental results for our approach on two domains: a dataset of scientific papers and authors and a database of movies, actors and directors. Our classification experiments show that the relational information provides a substantial boost in accuracy. Applied to a clustering task, we show that our methods are able to exploit the relational structure and find coherent clusters even for instance types that do not have any attributes.

2 Generative models for relational data

Probabilistic classification and clustering are often viewed from a generative perspective as a density estimation task. Data instances are assumed to be independent and identically distributed (IID) samples from a mixture model distribution. Each instance belongs to exactly one of k classes or clusters. In clustering, a latent class random variable is associated with the instance to indicate its cluster. Other attributes of an instance are then assumed to be samples from a distribution associated with its class. A simple yet powerful model often used for this distribution is the Naive Bayes model. In the Naive Bayes model, the attributes of each instance are assumed to be conditionally independent given the class variable. Although this independence assumption is often unrealistic, this model has nevertheless proven to be robust and effective for classification and clustering across a wide range of applications [Duda *et al.*, 2000; Cheeseman and Stutz, 1995]. Both classification and clustering involve estimation of the parameters of the Naive Bayes model; however, clustering is significantly more difficult due to the presence of latent variables.

The IID assumption made by these standard classification and clustering models is inappropriate in rich relational domains, where different instances are related to each other, and are therefore likely to be correlated. In this section, we describe a probabilistic model for classification and clustering in relational domains, where entities are related to each other. Our construction utilizes the framework of *probabilistic relational models (PRMs)* [Koller and Pfeffer, 1998; Friedman *et al.*, 1999].

2.1 Probabilistic Relational Models

A PRM is a template for a probability distribution over a relational database of a given schema. It specifies probabilistic models for different classes of entities, including probabilistic dependencies between related objects. Given a set of instances and relations between them, a PRM defines a joint probability distribution over the attributes of the instances.

Relational Schema. A relational schema describes attributes and relations of a set of instance types $\mathcal{X} = \{X_1, \ldots, X_n\}$. Each type X is associated with a set of *attributes* $\mathcal{A}(X)$. Each type X is also associated with a set $\mathcal{R}(X)$ of typed binary relations $R(X, Y)$. We associate each relation R with the type X of its first argument, allowing us to use the relation as a set-valued function, whose value $x.R$ is the set of instances $y \in Y$ related to an instance x. For example, for an actor a, a.*Role* is the set of movies in which the actor has appeared.

In certain cases, relations might have attributes of their own. For example, the "Role" relation might be associated with the attribute *Credit-Order*, which indicates the ranking of the actor in the credits. We can introduce an explicit type corresponding to the relation. In this case, a relation object is itself related to both of its arguments. For example, if one of the role objects is "Meryl Streep in Sophie's Choice", this role object would be related to the actor object "Meryl Streep" and the movie object "Sophie's Choice". By definition, these relations are many-to-one. It will be useful to distinguish between *entity* types (such as Person or Movie), and *relation* types (such as Role).

An *instantiation* $\mathcal{I}$ specifies the set of objects in each type, the relations that hold between them, and the values of the attributes for all of the objects. A *skeleton* σ specifies only the objects and the relations. We will use $\sigma(X)$ to denote the set of objects of type X.

Probabilistic Model. A probabilistic relational model Π specifies a probability distribution over a set of instantiations $\mathcal{I}$ of the relational schema. More precisely, a PRM is a tem-

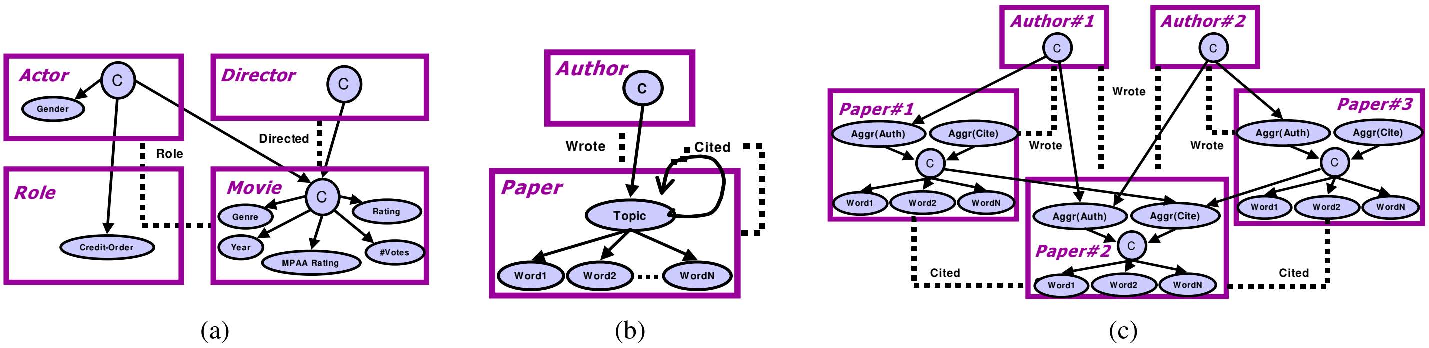


Figure 1: (a) Model for IMDB domain; (b) Model for Cora domain; (c) Fragment of unrolled network for Cora model.

plate, which is instantiated for different skeletons σ. The result of this instantiation is a probabilistic model over a set of random variables corresponding to all of the attributes of all of the objects in the skeleton. We can view a PRM as a compact way of representing a Bayesian network for any skeleton over this schema.

A PRM consists of a qualitative dependency structure, $\mathcal{S}$, and the parameters associated with it, $\boldsymbol{\theta}_{\mathcal{S}}$. The dependency structure is defined by associating with each attribute $X.A$ a set of *parents* $\text{Pa}(X.A)$. Each parent of $X.A$ has the form of either $X.B$ or $X.R.B$ for $R \in \mathcal{R}(X)$. (PRMs also allow dependencies along chains of relations, but we have chosen to omit those for simplicity of presentation.)

For a given skeleton σ, the PRM structure induces an *unrolled* Bayesian network over the random variables $x.A$. For every object $x \in \sigma(X)$, $x.A$ depends probabilistically on parents of the form $x.B$ or $x.R.B$. Note that if R is not single-valued, then $x.R.B$ is actually a set of random variables, one for each $y \in x.R$. We address this problem by interpreting the dependence of $X.A$ on $X.R.B$ as dependence on an aggregate function (e.g., mode or mean) of the multiset of values of these variables (see below).

The quantitative part of the PRM specifies the parameterization of the model. Given a set of parents for an attribute, we can define a local probability model by associating with it a *conditional probability distribution (CPD)*. For each attribute we have a CPD that specifies $P(X.A \mid \text{Pa}(X.A))$. When one of the parents is of the form $X.R.B$ and R is many-valued, the CPD represents the dependence on the value of the aggregate. The CPD for $X.A$ is used for $x.A$ in the unrolled network, for every x in X. Thus, the CPD for $X.A$ is repeated many times in the network.

Aggregates. There are many possible choices of aggregation operator to allow dependencies on a set of variables. An obvious choice for categorical variables is the mode aggregate, which computes the most common value of its parents. More precisely, consider some variable Y whose parents $\mathbf{X} = \{X_1 \ldots X_n\}$ we wish to aggregate into a single variable A. Let the domain of each X_i be $\{v_1, \ldots, v_k\}$, and note that A has the same domain. The effect of the mode aggregator is as follows: We define a distribution $P(Y \mid v_i)$ for each i; given a multiset of values for $X_1, \ldots, X_n$, we use the distribution $P(Y \mid v_i)$ for the value v_i which is the most common in this multiset.

The mode aggregator is not very sensitive to the distribution of values of its parents; for example, it cannot differentiate between a highly skewed and a fairly uniform set of values that have the same most frequent value. An aggregate that better reflects the value distribution is a *stochastic mode* aggregator. In this case, we still define a set of distributions $P(Y \mid v_i)$, but the effect of the aggregator is that $P(Y \mid X_1, \ldots, X_n)$ is a *weighted average* of these distributions, where the weight of v_i is the frequency of this value within $X_1, \ldots, X_n$. We accomplish this behavior by using an aggregate variable $A = \textit{Stochastic-Mode}(X_1, \ldots, X_n)$, defined as follows. The aggregate variable also takes on values in $\{v_1, \ldots, v_k\}$. Let $t_j(\mathbf{X})$ be the number of variables X_i that take on the value v_j. Then we define $P(A = v_j \mid \mathbf{X}) = t_j(\mathbf{X})/n$. It is easy to verify that this aggregator has exactly the desired effect.

We note that this aggregate can also be viewed as a randomized selector node that chooses one of its parents uniformly at random and takes on its value. One appealing consequence is that, like min or max, the stochastic model can be decomposed to allow its representation as a CPD to scale linearly with the number of parents. We simply decompose the aggregate in a cascading binary tree. The first layer computes aggregates of disjoint pairs, with each aggregate randomly selecting the value of one of its parents; the following layer repeats the procedure for disjoint pairs of results from the first layer, and so on. (This construction can also be extended to cases where the number of variables is not a power of 2; we omit details for lack of space.)

2.2 Classification and clustering models

We use the PRM framework as the basis of our models for relational classification and clustering. As in the "flat" probabilistic generative approaches, our approach is based on the use of a special variable to represent the class or the cluster. This variable is the standard "class" variable in the classification task. As usual, we deal with the clustering task by introducing a new latent class variable. Thus, for each entity class X we have a designated attribute $X.C$ in $\mathcal{A}(X)$.

As in flat classification and clustering, we define the attributes of X to depend on the class variable. For simplicity, we choose the Naive Bayes dependency model for the other attributes: For each attribute $X.A$, the only parent of $X.A$

is $X.C$. Note that we have only defined class attributes for entity types. We connect the attributes of relation types to the class attributes of the two associated entity types. Thus, for example, an attribute such as *Credit-Order* in the relation class Role will depend on the class attributes of Actor and Movie. Note that, as the dependence in this case is single-valued by definition, no aggregates are necessary. Most interestingly, we also allow direct dependence between class attributes of related entities. Thus, for example, we could allow a dependence of Movie.C on Actor.C, or vice versa. In this case, as the relation is many-to-many, we use aggregates, as described above.

Fig. 1(a) shows a simple model for a movie dataset, extracted from the *Internet Movie Database (IMDB)* (`www.imdb.com`). We see that "Role" is both a class on its own, as well as defining the relation between movies and actors. We have chosen, in this case, not to have the attribute Role.*Credit-Order* depend on the class of movies, but only of actors. Fig. 1(b) shows a model for a domain of scientific papers and authors, derived from the Cora dataset [McCallum *et al.*, 2000] (`cora.whizbang.com`). In this case, we see that the "Cites" relation connects two objects of the same type. We have chosen to make the class attribute of the cited paper depend on the class attribute of the citing paper. Note that this dependency appears cyclic at the type level. However, recall that this model is only a template, which is instantiated for particular skeletons to produce an unrolled network; Fig. 1(c) shows a fragment of such a network. If we do not have "citation cycles" in the domain, then this unrolled network is acyclic, and the PRM induces a coherent probability model over the random variables of the skeleton. (See [Friedman *et al.*, 1999] for more details.)

We can also use latent variable models to represent dyadic clustering. Consider, for example, a domain where we have people and movies, and a relation between them that corresponds to a person rating a movie. In this case, we will have a class Vote, corresponding to the relation, with the attribute *Rating* representing the actual rating given. This attribute will depend on the cluster attributes of both Movie and Person, leading naturally to a two-sided clustering model. However, our approach is flexible enough to accommodate a much richer model, e.g., where we also have other attributes of person, and perhaps an entire relational model for movies, such as shown in Fig. 1(a). Our approach will take all of this information into consideration when constructing the clusters.

3 Learning the models

We now show how we learn our models from data. Our training set D consists of a partial instantiation of the schema, one where everything except the values of some or all the class attributes is given. We can view this data as a single large "mega-instance" of the model, with a large number of missing values. Note that we cannot view the data as a set of independent instances corresponding to the objects in the model. In our setting, we typically assume that the structure of our latent variable model is given, as described in Section 2.2. Thus, our task is parameter estimation.

3.1 Parameter estimation

In this case, we assume that we are given the probabilistic dependency structure $\mathcal{S}$, and need only estimate the parameters $\boldsymbol{\theta}_\mathcal{S}$, i.e., the CPDs of the attributes. A standard approach is to use *maximum likelihood* (ML) estimation, i.e., to find $\boldsymbol{\theta}_\mathcal{S}$ that maximize $P(D \mid \boldsymbol{\theta}_\mathcal{S})$.

If we had a complete instantiation $\mathcal{I}$, the likelihood function has a unique global maximum. The maximum likelihood parameters can be found very easily simply by counting occurrences in the data. Recall that all of the objects in the same class share the same CPD. Thus, to estimate the parameter for $P(X.A \mid \text{Pa}(X.A))$, we simply consider all objects x of class X, and count the number of times that each combination $v, \mathbf{u}$ that $x.A$ and its parents jointly take. These counts are known as *sufficient statistics*. See [Friedman *et al.*, 1999] for details.

The case of incomplete data is substantially more complex. In this case, the likelihood function has multiple local maxima, and no general method exists for finding the global maximum. The *Expectation Maximization (EM)* algorithm [Dempster *et al.*, 1977], provides an approach for finding a local maximum of the likelihood function. Starting from an initial guess $\boldsymbol{\theta}^{(0)}$ for the parameters, EM iterates the following two steps. The E-step computes the distribution over the unobserved variables given the observed data and the current estimate of the parameters. Letting $\mathbf{C}$ be the set of unobserved cluster variables, we compute $P(\mathbf{C} \mid D, \boldsymbol{\theta}^{(t-1)})$, from which it can compute the *expected* sufficient statistics:

$$N_{X.A}[v, \mathbf{u}] = \sum_{x \in \sigma(X)} P(x.A = v \mid \text{Pa}(x.A) = \mathbf{u}, D, \boldsymbol{\theta}^{(t-1)})$$

To compute the posterior distribution over the hidden variables, we must run inference over the model. The M-step re-estimates the parameters by maximizing the likelihood with respect to the distribution computed in the E-step.

$$\theta^{X.A}_{v|\mathbf{u}} = \frac{N_{X.A}[v, \mathbf{u}]}{\sum_v N_{X.A}[v, \mathbf{u}]}$$

3.2 Belief Propagation for E step

To perform the E step, we need to compute the posterior distribution over the unobserved variables given our data. This inference is over the unrolled network defined in Section 2.2. We cannot decompose this task into separate inference tasks over the objects in the model, as they are all correlated. (In some cases, the unrolled network may have several connected components that can be treated separately; however, it will generally contain one or more large connected components.)

In general, the unrolled network can be fairly complex, involving many objects that are related in various ways. (In our experiments, the networks involve tens of thousands of nodes.) Exact inference over these networks is clearly impractical, so we must resort to approximate inference. There is a wide variety of approximation schemes for Bayesian networks. For various reasons (some of which are described below), we chose to use *belief propagation*. Belief Propagation (BP) is a local message passing algorithm introduced by Pearl [Pearl, 1988]. It is guaranteed to converge to the correct marginal probabilities for each node only for singly connected Bayesian networks. However, empirical results [Murphy and Weiss, 1999] show that it often converges in general networks, and when it does, the marginals are a good approximation to the correct posteriors. (When BP does not converge, the marginals for some nodes can be very inaccurate. This happens very rarely in our experiments and does not affect convergence of EM.)

We provide a brief outline of one variant of BP, referring to [Murphy and Weiss, 1999] for more details. Consider a Bayesian network over some set of nodes (which in our case would be the variables $x.A$). We first convert the graph into a *family graph*, with a node F_i for each variable X_i in the BN, containing X_i and its parents. Two nodes are connected if they have some variable in common. The CPD of X_i is associated with F_i. Let ϕ_i represent the factor defined by the CPD; i.e., if F_i contains the variables $X, Y_1, \ldots, Y_k$, then ϕ_i is a function from the domains of these variables to $[0, 1]$. We also define ψ_i to be a factor over X_i that encompasses our evidence about X_i: $\psi_i(X_i) \equiv 1$ if X_i is not observed. If we observe $X_i = x$, we have that $\psi_i(x) = 1$ and 0 elsewhere. Our posterior distribution is then $\alpha \prod_i \phi_i \times \prod_i \psi_i$, where α is a normalizing constant.

The belief propagation algorithm is now very simple. At each iteration, all the family nodes simultaneously send message to all others, as follows:

$$m_{ij}(F_i \cap F_j) \leftarrow \alpha \sum_{F_i - F_j} \phi_i \psi_i \prod_{k \in N(i) - \{j\}} m_{ki}$$

where α is a (different) normalizing constant and $N(i)$ is the set of families that are neighbors of F_i in the family graph. At any point in the algorithm, our marginal distribution about any family F_i is $b_i = \alpha \phi_i \psi_i \prod_{k \in N(i)} m_{ki}$. This process is repeated until the beliefs converge.

After convergence, the b_i give us the marginal distribution over each of the families in the unrolled network. These marginals are precisely what we need for the computation of the expected sufficient statistics.

We note that occasionally BP does not converge; to alleviate this problem, we start the EM algorithm from several different starting points (initial guesses). As our results in Section 5 show, this approach works well in practice.

4 Influence propagation over relations

Among the strong motivations for using a relational model is its ability to model dependencies between related instances. As described in the introduction, we would like to propagate information about one object to help us reach conclusions about other, related objects. Recently, several papers have proposed a process along the lines of this "influence propagation" idea. Neville and Jensen [2000] propose an *iterative classification* algorithm which builds a classifier based on a fully observed relational training set; the classifier uses both base attributes and more relational attributes (e.g., the number of related entities of a given type). It then uses this classifier on a test set where the base attributes are observed, but the class variables are not. Those instances that are classified with high confidence are temporarily labeled with the predicted class; the classification algorithm is then rerun, with the additional information. The process repeats several times. The classification accuracy is shown to improve substantially as the process iterates.

Slattery and Mitchell [2000] propose an application of this idea to the problem of classifying web pages, e.g., as belonging to a university student or not. They first train a classifier on a set of labeled documents, and use it to classify documents in the test set. To classify more documents in the test set, they suggest combining the classification of the test set pages and the relational structure of the test set. As a motivating example, they describe a scenario where there exists a page that points to several other pages, some of which were classified as student home pages. Their approach tries to identify this page as a student directory page, and conclude that other pages to which it points are also more likely to be student pages. They show that classification accuracy improves by exploiting the relational structure.

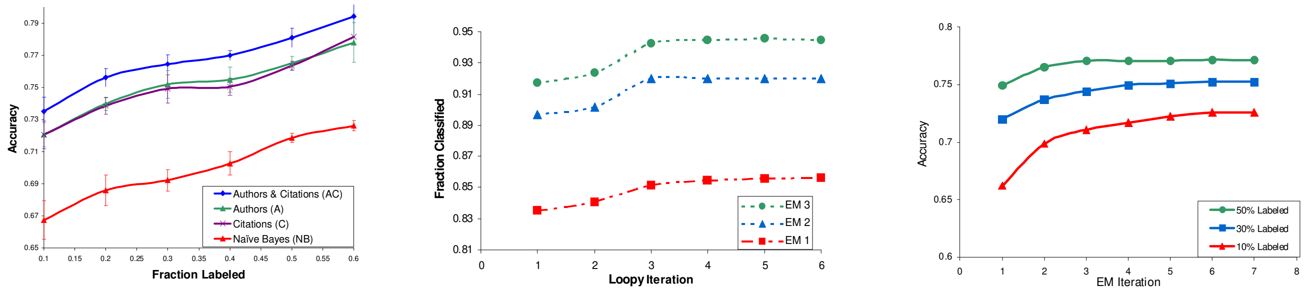
Neither of these approaches proposes a single coherent model of the dependencies between related objects and thus combine different classification steps or algorithms without a unifying principle. Our approach achieves the influence propagation effect through the probabilistic influences induced by the unrolled Bayesian network over the instances in our domain. For example, in the Cora domain, our network models correlations between the topics of papers that cite each other. Thus, our beliefs about the topic of one paper will influence our beliefs about the topic of its related papers. In general, probabilistic influence "flows" through active paths in the unrolled network, allowing beliefs about one cluster to influence others to which it is related (directly or indirectly). Moreover, the use of belief propagation implements this effect directly. By propagating a local message from one family to another in the family graph network, the algorithm propagates our beliefs about one variable to other variables to which it is directly connected. We demonstrate this property in the next section.

This spreading influence is particularly useful in our framework due to the application of the EM algorithm. The EM algorithm constructs a sequence of models, using the probabilities derived from the belief propagation algorithm to train a new model. Hence, we not only use our probabilistic inference process to spread the information in the relational structure, we then use the results to construct a better classifier, which in turn allows us to obtain even better results, etc. From a different perspective, we are using the structure in the test set not only to provide better classifications, but also to learn a better classifier. As we show below, this process results in substantial improvements in accuracy over the iterations of EM. We note that this bootstrapping ability arises very naturally in the probabilistic framework, where it is also associated with compelling convergence guarantees.

5 Experiments

We evaluated our method on the Cora and IMDB data sets.

Cora. The structure of the Cora dataset, and the model we used, are shown in Fig. 1(b,c). For our experiments, we selected a subset of 4187 papers from the Machine Learning category, along with 1454 of their authors. These papers are classified into seven topics: Probablistic Methods, Neural networks, Reinforcement Learning, Rule Learning, Case-Based, and Theory.



(a) (b) (c)

Figure 2: (a) Comparison of classification accuracies; (b) Influence propagation in BP; (c) Accuracy improvement in EM.

We evaluated the ability of our algorithm to use the relational structure to aid in classification. We took our entire data set, and hid the classifications for all but a fraction of the papers. We then constructed our model based on all of this data, including the documents whose topics were unobserved. The resulting model was used to classify the topic for the test documents. In effect, we are performing a type of *transduction*, where the test set is also used to train the model (albeit without the class labels).

To investigate how our method benefits from exploiting the relational structure, we considered four different models which vary in the amount of relational information they use. The baseline model does not use relational information at all. It is a standard multinomial Naive Bayes model (**NB**) over the set of words (bag of words model) in the abstract. The full model (**AC**) was shown in Fig. 1(b); it makes use of both the authors and citations. The other two models are fragments of **AC**: model **A** incorporates only the author information (eliminating the citation relation from the model), and model **C** only citations. All four models were trained using EM; model **NB** was trained using exact EM and the others using our algorithm of Section 3. We initialized the CPDs for the word attributes using the CPDs in a Naive Bayes model that was trained only on the observed portion of the data set. All models were initialized with the same CPDs.

We varied the percentage of labeled papers, ranging from 10% to 60%. For each different percentage, we tested the classification accuracy over five random training/test splits. The results are shown in Fig. 2(a). Each point is the average of the accuracy on the five runs, and the error bars correspond to the standard error. As can be seen, incorporating more relational dependencies significantly improves classification accuracy. Both **A** and **C** outperform the baseline model, and the combined model **AC** achieves by far the highest accuracy.

As discussed in Section 4, the local message passing of loopy belief propagation (BP) resembles the process of "spreading" the influence of beliefs for a particular instance to its related instances. For example, suppose paper x cites several labeled papers. Upon initialization, we have some initial belief about the topic of x from its words alone. However, after the first iteration, this belief will be updated to reflect the labels of the papers it cites, and is likely to become more peaked around a single value, increasing the confidence in x's topic. In the following iteration, unlabeled papers that cite x (as well as unlabeled papers that x cites) will be updated to reflect the increased confidence about the topic of x, and so on. To measure this effect, we examine the belief state of the topic variable of the unlabeled papers after every iteration of loopy belief propagation. For every iteration, we report the fraction of variables whose topic can be determined with high confidence, i.e., whose belief for a single topic is above a threshold of 0.9. Fig. 2(b) shows several series of these measurements on a dataset with 10% labeled papers. The series show BP iterations performed within the first, third and seventh iteration of EM. Each series shows a gradual increase of the fraction of papers whose topics we are confident in. The accuracy on those high-confidence papers is fairly constant over the iterations — around 0.7, 0.735, and 0.74 for the first, third and seventh iteration of EM, respectively.

Loopy belief propagation is an approximation to the inference required for the E step of EM. Although loopy BP is not guaranteed to converge, in our experiments, it generally converges to a solution which is good enough to allow EM to make progress. Indeed, Fig. 2(c) shows that the classification accuracy improves for every EM iteration. This figure also demonstrates the performance improvement obtained from bootstrapping the results of iterative classification, as discussed in Section 4.

IMDB. The attributes and relations in the IMDB database, and the latent variable model we used, are shown in are shown in Fig. 1(a); the *Genre* attribute actually refers to a set of 18 binary attributes (action, comedy, ...). Note that actors and directors have almost no descriptive attributes and hence cannot be clustered meaningfully without considering their relations. We selected a subset of this database that contains 1138 movies, 2446 actors, and 734 directors. In Fig. 5, we show two example clusters for each class, listing several highest confidence members of the clusters.

In general, clusters for movies consist of movies of predominantly of a particular genre, time period and popularity. For example, the first movie cluster shown can be labeled as classic musicals and children's films. The second cluster corresponds roughly to action/adventure/sci-fi movies. In our model, the clusters for actors and directors are relational in nature, since they are induced by the movie attributes. For example, the first cluster of actors consists primarily of action

Swiss Family Robinson	Cinderella
Sound of Music, The	Love Bug, The
Wizard of Oz, The	Pollyanna
Parent Trap, The	Mary Poppins
Robin Hood: Prince of Thieves	Batman
Hunt for Red October, The	Batman Forever
Mission: Impossible	GoldenEye
Terminator 2: Judgment Day	Starship Troopers

Stallone, Sylvester	Russell, Kurt
Schwarzenegger, Arnold	Costner, Kevin
Van Damme, Jean-Claude	Willis, Bruce
Ford, Harrison	Seagal, Steven
Jones, Tommy Lee	De Niro, Robert
Hopkins, Anthony	Keitel, Harvey
Freeman, Morgan	Oldman, Gary

Hitchcock, Alfred
Kubrick, Stanley
Coppola, Francis Ford
Lean, David
Forman, Milos
Gilliam, Terry
Wyler, William
Spielberg, Steven
Cameron, James
McTiernan, John
Burton, Tim
Scott, Tony
Schumacher, Joel

Figure 3: Clusters of Movies, Actors, and Directors

movie actors and the second of actors who primarily play in dramas. Similarly for directors, the first cluster corresponds to directors of dramas and while the second to directors of popular action and adventure films.

6 Discussion and Conclusions

Many real-world domains have a rich relational structure, with complex webs of interacting entities: the web; papers and authors; biomedical datasets; and more. Traditional machine learning algorithms ignore this rich relational structure, flattening it into a set of IID attribute vectors. Recently, however, there has been growing interest in learning methods that exploit the relational structure of the domain.

In this paper, we provide a general method for classification and clustering in richly structured data with instances and relations. Our approach has coherent probabilistic semantics, allowing us to build on powerful tools for probabilistic reasoning and learning. Our algorithm uses an effective combination of these techniques to provide linear scaling in the number of instances; it can thus be applied to very large domains.

We have shown in a transduction task that the relational information allows us to achieve substantially better accuracies than a standard "flat" classification scheme. We have also shown anecdotally that our algorithm constructs interesting clusters based on relational information. Finally, our approach induces a compelling behavior unique to relational settings: Because instances are *not* independent, information about some instances can be used to reach conclusions about others. Our approach is the first to provide a formal framework for this behavior.

There are many interesting extensions to this work. Most obvious is the problem of *model selection*. In this paper, we have used predetermined models, both the edges in the dependency model and the number of clusters of each latent class attribute. Our framework allows us to incorporate into our models a great deal of prior knowledge about the semantics of the underlying domains. However, when domain expertise is lacking, automatic model construction becomes crucial. We can extend our approach using techniques for model selection in Bayesian networks [Friedman, 1998; Cheeseman and Stutz, 1995], allowing our learning algorithm to select the model structure that best suits the data.

Acknowledgments. This work was supported by ONR contract N66001-97-C-8554 under DARPA's HPKB program. Eran Segal was also supported by a Stanford Graduate Fellowship (SGF).

References

[Cheeseman and Stutz, 1995] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In Fayyad U., Piatesky-Shapiro G., Smyth P., and Uthuruasmy R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, Menlo Park, CA, 1995.

[Cohn and Chang, 2000] D. Cohn and H. Chang. Probabilistically identifying authoritative documents. In *Proc. SIGIR*, 2000.

[Cohn and Hofmann, 2001] D. Cohn and T. Hofmann. The missing link: A probabilistic model of document content and hypertext connectivity. In *Proc. NIPS*, 2001. To appear.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–39, 1977.

[Duda *et al.*, 2000] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2000.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. IJCAI*, 1999.

[Friedman, 1998] N. Friedman. The Bayesian structural EM algorithm. In *Proc. UAI*, 1998.

[Hofmann and Puzicha, 1999] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. IJCAI*, 1999.

[Kleinberg, 1998] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[Koller and Pfeffer, 1998] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proc. AAAI*, 1998.

[McCallum *et al.*, 2000] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. *Automating the construction of internet portals with machine learning*. Information Retrieval Journal, 3:127, 2000.

[Murphy and Weiss, 1999] K. Murphy and Y. Weiss. Loopy belief propagation for approximate inference: an empirical study. In *UAI*, 1999.

[Neville and Jensen, 2000] J. Neville and D. Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20. AAAI Press, 2000.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Slattery and Craven, 1998] S. Slattery and M. Craven. Combining statistical and relational methods in hypertext domains. In *Proc. ILP*, 1998.

[Slattery and Mitchell, 2000] S. Slattery and T. Mitchell. Discovering test set regularities in relational domains. In *Proc. ICML*, 2000.