J. Neville, D. Jensen, L. Friedland and M. Hay (2002). Learning Relational Probability Trees.
University of Massachusetts, Technical Report 02-55. Revised 02/2003.

Learning Relational Probability Trees

Jennifer Neville David Jensen Lisa Friedland Michael Hay

Computer Science Department
University of Massachusetts
Amherst, MA 01003 USA
[jneville | jensen | lfriedl | mhay]@cs.umass.edu

ABSTRACT

Classification trees are widely used in the machine learning and data mining communities for modeling propositional data. Recent work has extended this basic paradigm to probability estimation trees (PETs). Traditional tree learning algorithms assume that instances in the training data are homogenous and independently distributed. Relational probability trees (RPTs) extend standard probability estimation trees to a relational setting in which data instances are heterogeneous and interdependent. Our algorithm for learning the structure and parameters of an RPT searches over a space of relational features that use aggregation functions (e.g. AVERAGE, MODE, COUNT) to dynamically flatten relational data and create dichotomous divisions within the RPT. Previous work has identified a number of statistical biases due to common characteristics of relational data such as autocorrelation and degree disparity. The algorithm uses a novel form of randomization test to adjust for these biases. On a variety of relational learning tasks, we show that RPTs built using randomization tests are significantly smaller than other models and achieve equivalent, or better, performance.

1. INTRODUCTION

Classification trees are used widely in the machine learning and data mining community for propositional data. Due to the selective nature of classification models and their intuitive representation of knowledge, the learned trees are often easily interpretable. This makes classification trees an attractive modeling approach for the knowledge discovery community. However, conventional tree learning algorithms were designed for data sets where the instances are homogeneous and statistically independent. The past two decades have seen a dramatic increase in the amount of stored information and much of the data being captured are relational in nature. This has created a need for a new generation of analysis techniques for relational data that are heterogeneous and interdependent. In this paper we present an algorithm for learning classification trees over relational data.

Because of the popularity of classification tree models, there is a large body of research detailing the results of various algorithm design choices. For example, it has been shown that cross-validation can be used to avoid attribute selection biases [9] and that misclassification costs are generally insensitive to choice of split criteria [19]. Recent work has extended the basic classification tree paradigm to probability estimation trees (PETs) and has focused on improving probability estimates in leaves [19]. We can leverage this body of research to construct a new algorithm for relational data where many sources of potential variance have been addressed. Choosing a well-studied algorithm to modify for use with relational data reduces the range of effects we can expect due to other unexplored biases inherent in the model choice and allows us to focus of the characteristics of relational data.

Our recent work on relational learning has concentrated on the unique challenges of learning probabilistic models in relational data, where the traditional assumption of instance independence is violated [10], [11]. Specifically, we have examined how particular characteristics of relational data affect the statistical inferences necessary for accurate learning. We have identified three characteristics of relational data—concentrated linkage, degree disparity, and relational autocorrelation—and have shown how these can greatly complicate efforts to construct good statistical models. In particular, they can lead to feature selection bias and discovery of spurious correlations, leading to overly complex models with excess structure.

Excess structure in models is harmful for several reasons. First, such models are factually incorrect. They indicate that some variables are related when they are not. Some applications use induced models to support additional reasoning (e.g. [2]), so correctness can be a central issue. Second, such models require more space and more computational resources than models that do not contain unnecessary components. Third, using a model with excess structure can require the collection of unnecessary features for each instance, increasing the cost and complexity of making predictions. For example, medical diagnosis with unnecessarily large models would require unnecessary medical tests. Fourth, large models are more difficult to understand. The unnecessary components complicate attempts to integrate induces models with existing knowledge derived from other sources. Overfitting avoidance has sometimes been justified solely on the grounds of producing comprehensible models [21].

Many techniques common to machine learning, data mining, and statistical modeling use the underlying assumption that data instances are independent. Techniques for learning statistical models of relational data need to address the problems associated with violating these assumptions. We have developed a promising class of techniques, based on randomization tests and resampling-based methods, and have shown that these computationally-intensive statistical procedures allow us to adjust for the unique characteristics of a given relational data set, and make accurate parameter estimates and hypothesis tests [12]. We have incorporated these approaches into our algorithm for constructing relational probability trees (RPTs). To our knowledge, RPTs are the only statistical models for relational data that adjust for potential biases due to the unique characteristics of relational data.

We begin by describing a specific data set, an example analysis task—predicting whether a web page belongs to a student page—and present a simplified version of an RPT model learned for this task. Next, we describe some of the basic features of relational data and approaches to modeling data that do not meet the assumptions of conventional techniques. The next section discusses the set of statistical issues unique to relational data, the biases that may result from ignoring those issues and a solution to adjust for those biases. Then, we cover the details of the RPT algorithm and finish with an experimental section that evaluates RPTs against C4.5 [20] and RBCs [18] on a number of classification tasks.

2. EXAMPLE TASK

Recent research has examined methods for constructing statistical models of complex relational data [7]. Examples of such data include social networks, genomic data, and data on interrelated people, places, things, and events extracted from text documents. Consider the data set collected by the WebKB Project [4]. It consists of a set of web pages from four computer science departments. The web pages have been manually classified into the categories: course, faculty, staff, student, research project, or other. The category "other" denotes a page that is not a home page (e.g. a curriculum vitae linked from a faculty page or homework description linked from a course page). The collection contains approximately 4,000 web pages and 8,000 hyperlinks among those pages. The task is to determine whether a page belongs to a student (P(+)=0.51).

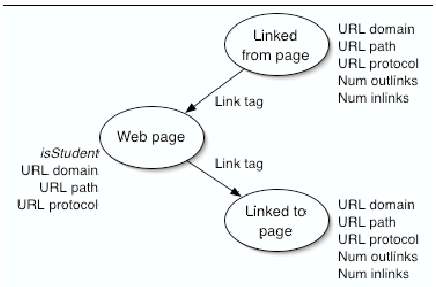


Figure 1: WebKB schema.

The data are shown schematically in Figure 1. The data consist of web pages and associated pages one hyperlink away. The pages are connected in the ways that you would expect (e.g. out-links and in-links). In addition to the hyperlink relations, the database contains attributes associated with each page and hyperlink, including the path and domain of urls, and the directionality of the link in the department hierarchy (e.g. lateral, up, down).

Our learning algorithm for relational probability trees constructs trees such as the one shown in Figure 2. The tree represents a series of questions to ask about a web page and the pages in its relational neighborhood. The leaf nodes contain probability distributions over the values of the isStudent attribute. They show, of the training set instances that reach the leaf, the number of pages of each class, as well as their respective probabilities. In this tree, the root node asks whether the page is linked to from a page with more than 111 out-links (e.g. directory page). If so, the page travels down the left-hand branch and result in a 99% probability of being a student page. If not, the page travels down the right-hand branch. The next node asks whether the page is linked to from a page without any path appended to the url (e.g. department home page). If so, the page is unlikely to belong to a student. If not, the next node asks whether the page is linked to from pages with a high average number of out links (e.g. directory pages, research group pages). If so, the page is likely to belong to a student.

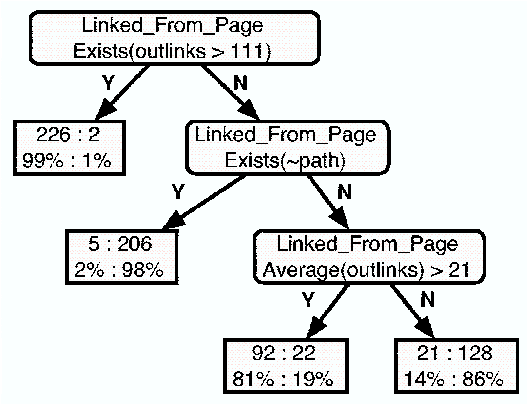


Figure 2: Example probability estimation tree.

This tree is a simplified version of an RPT learned on pages from three departments. The actual tree had 7 nodes but we collapsed the lower levels for readability and space considerations. When tested on the fourth held-out department, this tree had both perfect accuracy and perfect area under the ROC curve. The tree construction algorithm is described in section 5, and the details of this experiment are reported in section 6.

3. MODELING RELATIONAL DATA

To generate the RPT discussed above, we had to account for the unique characteristics of relational data relative to propositional data. Relational data violate two assumptions of conventional classification techniques. First, algorithms designed for propositional data assume the data are independent and identically distributed (i.i.d.). Relational data, however, have dependencies both as a result of direct relations (e.g. hyperlinked pages) and through chaining multiple relations together (e.g. pages linked to by the same directory page).

Figure 3 presents two simple relational data sets represented as subgraphs. Each set contains a unique object x with class label {+,-}. A propositional data set would consist only of objects x. In propositional models, each object x is independent and the model is conditioned on the attribute values of x. Figure 3 illustrates relational data sets where the examples are represented as subgraphs. Each subgraph contains a unique object x connected to one or more other objects y. In this case, objects y contain attributes that may be used to condition the model. The first set of data represents i.i.d. relational data, where objects x and y have a one-to-one relationship and where the class labels on instances are independent. The second set shows instances where objects x and y have a many-to-one relationship and where the class labels are dependent. This paper will focus on relational data similar in structure to figure 3b where each subgraph consists of multiple relations and each relation may produce dependencies among the instances. We use a graph representation throughout the paper, but complex rela-

tional data may also be represented in logical statements or relational database tables.

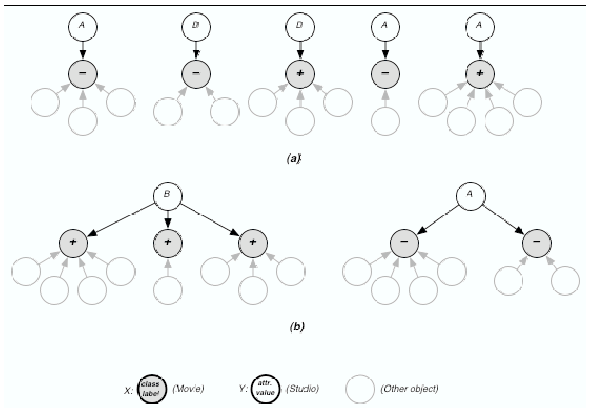


Figure 3: Example relational data sets with independent instances (a) and dependent instances (b).

Relational data have more information available with which to build better models, but the data often have complex structures which are more difficult to model. For example, the subgraph in figure 4a shows the data available to predict movie success (receipts>$2mil) in a relational dataset from the cinematic domain. In addition to information about the movie itself, there is information regarding the actors, directors, producers, and studios that participated in making the movie. For example, actors have gender, age and award information. Each movie subgraph may have a different number of related objects, resulting in diverse structures. For example, some movies may have 10 actors and others may have 1000. A relational classification technique needs to contend with heterogeneous data instances for both learning and inference.

There are a number of approaches to using conventional machine learning techniques on relational data. Aggregation is widely used as a means to homogenize relational data for modeling whether it is applied as a pre-processing step (e.g. [16]) or applied dynamically during the learning process (e.g. [8]). Heterogeneous data is transformed into homogenous records by aggregating multiple values into a single value (e.g. average actor age). Figure 4b contains a portion of an aggregated relational data set where multiple values have been averaged (for discrete attribute we take the modal value). RPTs use relational features constructed from dynamic application of aggregation functions. We define relational features below but defer the details of the RPT algorithm until section 5.

3.1 Relational features

We define *feature* as a mapping between raw data and a low-level inference. For example, a feature for a movie might be *genre=comedy*. In this case, a feature combines an attribute (genre), an operator, and a value. Typically, many features are combined into a higher-level model such as a tree or rule set.

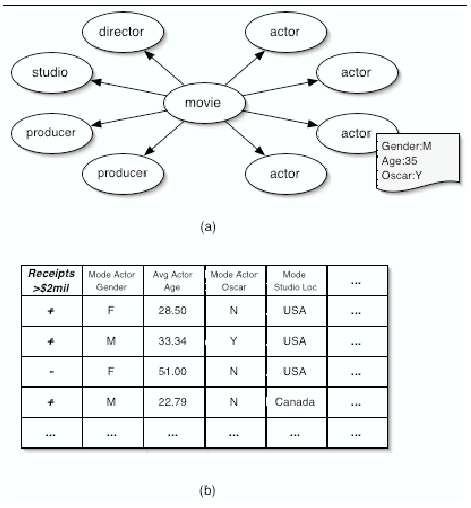


Figure 4: Some objects and attributes in IMDb .

Specifically, we focus here on relational features. Relational features are used by RPTs to predict the value of an attribute on particular types of objects (e.g., the box office receipts of movies) based on the attributes of related objects (e.g., characteristics of the movie's director, producer, and actors, and the studio that released the movie). Relational features are similar to the features described above in that they identify both an attribute and a way of testing the values of the attribute. However relational features may also identify a particular relation (e.g. ActedIn(x,y)) that links a single object x (e.g. movie) to a set of other objects Y (e.g. actors). If this is the case, the attribute referenced by the feature may belong to the related objects Y (e.g. actor age), and the test is conducted on the set of attribute values on the objects in Y. For example, the relational feature:

$$Movie(x), Y = \{y \mid ActedIn(x, y)\} : Max(Age(Y)) > 65$$

determines whether the oldest of the actors in movie x is over 65.

When the relation between x and Y is one-to-many, a relational feature must consider a set of attribute values on the objects Y. In this situation, standard database aggregation functions can be used to map sets of values into single values. The RPT algorithm uses the following aggregation functions to create binary splits in trees: AVERAGE, MODE, COUNT, and PROPORTION. MINIMUM, MAXIMUM, and EXISTS are special cases of these aggregation functions.

The graph structure of relational data may also be used in features. For example, a feature could count the number of actors associated with a movie, instead of counting a particular attribute value. We will refer to such features as DEGREE features throughout the rest of this paper. DEGREE features can be used to count the degree of links (e.g. number of phone calls between two people), the degree of objects (e.g. number of children for a given parent), or even the degree of attribute values (e.g. number of aliases for a given person). As we will show later in section 4.2, DEGREE features capture an important structural aspect of relational data – the heterogeneity of the relations. In certain restricted types of relational data, such as spatial or temporal data, the structure is homogeneous. There, it is not as important to be able to represent degree because all objects have similar structure. However, we have analyzed a num-

ber of relational data sets where degree features are highly correlated with the class label (see section 6).

4. STATISTICAL BIASES

Our recent work has shown that feature selection will be biased when relational learning algorithms ignore concentrated linkage and relational autocorrelation, two common characteristics of relational data. Specifically, these two characteristics can cause learning algorithms to be strongly biased toward certain features, irrespective of their predictive power.

4.1 Linkage and autocorrelation

Informally, concentrated linkage occurs when many objects are linked to a common neighbor, and relational autocorrelation occurs when the values of a given attribute are highly uniform among objects that share a common neighbor. See [10] for more details. Figure 5 illustrates the concept of concentrated linkage. Objects in set (a) are each linked to single, unique neighbor (low linkage); objects in set (b) are linked to a few common neighbors (high linkage).

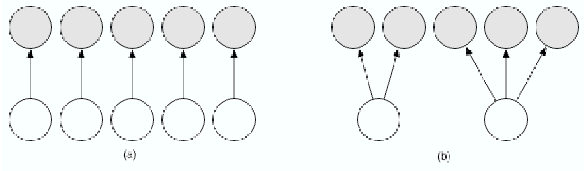


Figure 5: Low linkage vs. high linkage.

Concentrated linkage is common in relational data sets. For example, while studying relationships among publicly traded companies in the banking and chemical industries, we found that nearly every company in both industries uses one of only seven different accounting firms. In work on fraud in mobile phone networks, we found that 800 numbers, 900 numbers, and some public numbers (e.g., 911) produced concentrated linkage among phones. Many articles in the scientific literature are published in a single journal and many basic research articles are cited in single review articles. On the Web, many content pages are linked to single directory pages on sites such as Yahoo. In the IMDb, many movies are link to a small set of studios.

Relational autocorrelation is depicted in figure 6. The class labels of objects in set (a) show no uniformity (low autocorrelation). On the other hand, the class labels of objects in set (b) are highly correlated among objects that share a common neighbor (high autocorrelation). We have defined relational autocorrelation in a similar way to existing definitions of temporal and spatial autocorrelation (see, for example, [5]). Autocorrelation in these specialized types of relational data has long been recognized as a source of increased variance. However, the more general types of relational data commonly analyzed by relational learning algorithms pose even more severe challenges because the amount of linkage can be far higher than in temporal or spatial data and because that linkage can vary dramatically among objects.

In our analysis of relational data, we have encountered many examples of high relational autocorrelation. For example, in our study of publicly traded companies, we found that when persons served as officers or directors of multiple companies, the companies were often in the same industry. Similarly, in biological data on protein interactions we analyzed for the 2001 ACM SIGKDD Cup Competition, the proteins located in the same place in a cell (e.g., mitochondria or cell wall) had highly autocorrelated functions (e.g., transcription or cell growth). Such autocorrelation has been identified in other domains as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated [3]. The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as "hubs" [13]. Similarly, the topics of articles in the scientific literature are highly autocorrelated when linked through a common journal.

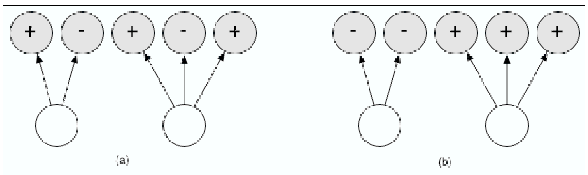


Figure 6: No autocorrelation vs. high autocorrelation.

Linkage and autocorrelation cause feature selection bias in a two-step chain of causality. First, linkage and autocorrelation combine to reduce the *effective sample size* of a data set, thus increasing the variance of scores estimated using that set. Relational data sets with high linkage and autocorrelation contain less information than an equivalently sized set of independent data. This reduction in effective sample size increases the variance of parameter estimates made with the data. Just as small data samples can lead to inaccurate estimates of the scores used to select features, concentrated linkage and autocorrelation can cause the scores of some features to have high variance. Second, increased variance of score distributions increases the probability that features formed from objects with high linkage and autocorrelation will be selected as the best feature, even when these features are random. To our knowledge, with the exception of RPTs, no current relational learning algorithm adjusts for this bias.

4.2 Degree disparity

Degree disparity is another common characteristic of relational data. When aggregation is used on data with degree disparity and autocorrelation, it can lead data mining algorithms to include completely spurious elements in their models (Type I errors) and to completely miss very useful elements (Type II errors) [11]. These errors occur with degree disparity because many aggregation functions (e.g., MAX) will produce apparent correlation between the aggregated values (e.g., maximum movie receipts) and a class label (e.g., studio location) whenever degree disparity occurs, regardless of whether movie receipts have any correlation with studio location.

We define degree disparity as "systematic variation in the distribution of the degree for one entity type with respect to the class label on another entity type." For example, actor degree disparity exists with respect to the box office receipts of movies if successful movies tend to have more (or fewer) actors than unsuccessful movies. Figure 7 illustrates degree disparity schematically. Objects in set (a) link to the same number of objects irrespective of degree (no disparity); objects in set (b) with a class label of {-} link to a greater number of objects than those with a class label of {+} (high disparity).

We have found examples of degree disparity in a number of relational data sets. For example, in movie data, US-based studios are systematically linked to a larger number of movies than foreign studios ($p<0.0001$). In corporate data, the number of owners differs systematically among publicly traded companies in different industries. In web data, the number of hyperlinks differs systematically among different classes of web pages at university web sites.

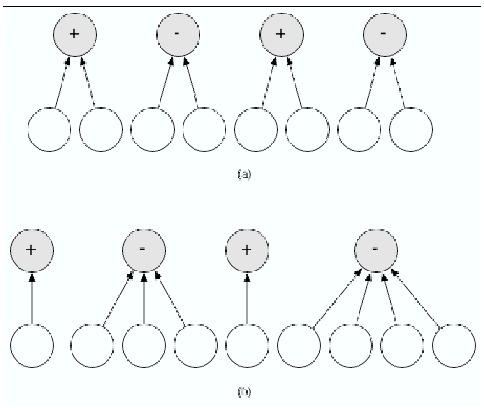


Figure 7: No degree disparity vs. degree disparity.

Degree disparity can reflect a wide variety of different influences. For example, the degree disparity observed in IMDb may reflect either prevailing practices in the movie industry (e.g., movie producers could make relatively good estimates of future box office receipts and thus be willing to invest in larger cast sizes) or it could reflect data collection practices (IMDb data are contributed by movie enthusiasts who may show disproportionate interest to popular movies and thus list a larger percentage of all actors). Similarly, the degree disparity observed in scientific literature databases may indicate that researchers in particular areas cite a smaller number of papers, or that the database contains more conference papers in those areas (as opposed to journal articles which cite a larger number of papers). One can imagine degree disparity that is far more extreme than these cases. For example, the "publication degree" of researchers with respect to their seniority will exhibit degree disparity merely because of the cumulative nature of a researcher's publication list. Similarly, the "incoming-link degree" of web pages with respect to their popularity will exhibit degree disparity due to the pages' popularity itself.

Given degree disparity, nearly any aggregated attribute can show apparent correlation with the class label. This is true regardless of which of a large class of aggregation functions are used—count, exists, sum, max, min, avg, mode—although the amount of correlation depends on the aggregation function employed, the extent of degree disparity, and the distribution of the attribute being aggregated. Some of these effects are obvious. For example, the sum of a continuous attribute such as actor age will be much higher for movies with many actors than for those with few. Similarly, the count of a particular value of a discrete attribute (gender=female) will be systematically higher for movies with more actors. Other effects are relatively clear when you consider the effects of degree disparity. For example, the min or max values of a particular continuous attribute of actors will tend to be larger given the opportunity to select from a larger number of actors. Similarly, the probability that a particular value exists will be higher given a larger number of actors.

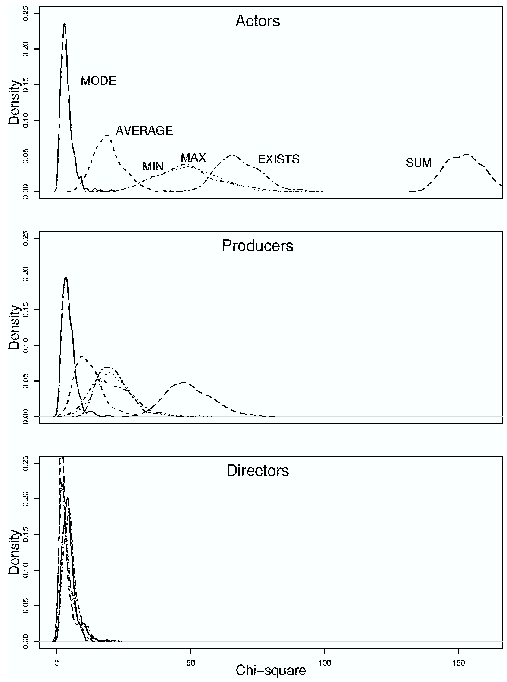


Figure 8: Simulation results for different types of aggregation functions on random attributes.

The use of aggregation functions on data with degree disparity can lead to apparent correlation between the aggregated feature and the class label even if the individual attribute values are independent of the class label. Such correlation reflects degree disparity alone, and it can have strong negative effects on model learning. First, this type of correlation produces models that are easily misunderstood as representing correlation between the attribute values themselves and the class label. At the very least, correlation due to degree disparity introduces an added level of indirection into a user's understanding of an induced model. Second, correlation due to degree disparity can vastly increase the number of apparently useful features, making induced models much more complex. This added complexity makes models correspondingly much less understandable and much less computationally efficient to use. For many techniques, particularly graphical models such as probabilistic relational models [8], the identification of conditional independence among attributes is a central goal, because it improves both interpretability and efficiency. Both these goals are impaired by added complexity. In addition, the large number of surrogate features for degree will cause some types of models to spread the credit for the predictive ability of degree across a large number of other features, making it appear that many features are weakly predictive rather than the truth—that a single structural feature (degree) is strongly predictive.

4.3 Randomization tests

Randomization tests [12] provide a method for hypothesis testing in relational data with linkage, autocorrelation and degree disparity. A randomization test is a type of computationally intensive statistical test [6]. Other types include re-

sampling and Monte Carlo procedures. Each of these tests involves generating many replicates of an actual data set—typically called pseudosamples—and using the psuedosamples to estimate a sampling distribution. In the case of a randomization test, pseudosamples are generated by randomly reordering (or permuting) the values of one or more variables in an actual data set. Each unique permutation of the values corresponds to a unique pseudosample. A score is then calculated for each pseudosample, and the distribution of these randomized scores is used to estimate a sampling distribution for the score calculated from the actual data. Randomization tests are also called *permutation tests*.

To construct pseuodsamples in relational data, we randomize the attribute values prior to aggregation. Using this approach, pseudosamples retain the linkage present in the original sample and the autocorrelation among the class labels. Retaining the linkage also preserves any degree disparity present in the data. Randomizing attribute vectors destroys the correlation between the attributes and the class in pseudosamples, thus making them appropriately conform to the null hypothesis—that there is no correlation between the attribute values and the class label. In addition to adjusting for the effects of autocorrelation and degree disparity, randomization tests adjust for the effects of attribute selection errors [9].

Randomization tests provide a method of hypothesis testing that can adjust for the effects of degree disparity, linkage and autocorrelation on parameter estimates. Another alternative is to estimate the parameters (feature scores) more accurately. We have explored adjustments to chi-square calculations which "factor out" the bias introduced by degree disparity [11]. However, adjustments are difficult to calculate for some combinations of aggregation function and attribute distributions. Also, we do not yet know how to adjust for the high variance associated with objects having high linkage and autocorrelation. For now, we use hypothesis testing to make accurate assements of significance when choosing among features with differing levels of bias and variance. This approach facilitates unbiased feature selection and prevents excessive tree structure.

5. RELATIONAL PROBABILITY TREES

RPT models estimate probability distributions over possible attribute values. The task of estimating probability distributions over the values of a given attribute would appear to differ little from traditional propositional learning. However, algorithms for relational learning typically look beyond the item for which the attribute is defined, to consider the effect of related objects on the probability distribution. For example, in order to predict the box-office success of a movie, a relational model would consider not only the attributes of the movie, but attributes of the actors in the movie and the director, producer, and studio that made the movie. A model might go even further and consider attributes of much more "distant" objects (in the sense of a graph neighborhood).

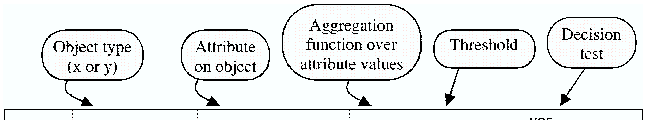
Classification tree algorithms are easily modified for relational data—the examples simply consist of subgraphs instead of independent objects. Feature specifications need to be enhanced to consider (object, attribute) pairs but the basic algorithm structure remains relatively unchanged. Indeed, several other decision tree algorithms for relational data have already been developed including TILDE [1], Multi Relational Decision Trees [14] and Structural Regression Trees (SRTs) [15]. These systems focus on extending decision tree algorithms to work in the first-order logic framework used by inductive logic programming systems (ILP). Although these systems can be used to build classification trees with relational data, the space of feature classes they consider is restricted to a subset of first-order logic. We chose to implement a classification tree algorithm with the capacity to explore a wider range of feature families. In return for this flexibility we gave up some of the expressiveness of the systems mentioned above. In particular RPTs are not able to string together a series of conjunctions about a particular object.

5.1 Algorithm Overview

The RPT algorithm takes a collection of subgraphs as input (e.g., those in figure 4a). Each subgraph contains a single target object to be classified; the other objects and links in the subgraph form its relational neighborhood. For example, an RPT can be used to predict the box office success of a movies based on the attributes of the movie and related objects, including the movie's actors, directors and producers.

The RPT algorithm constructs a probability estimation tree to predict the target class label given (1) the attributes of the target objects, (2) the attributes of other objects and links in the relational neighborhood of the target objects, and (3) graph attributes specifying the structure of relations.

The RPT algorithm is a recursive partitioning algorithm that searches over a space of binary relational features to split the data. The algorithm considers the attributes of different object/link types in the subgraphs and multiple methods of aggregating the values of those attributes, creating binary splits on the aggregated values. In particular the RPT algorithm considers the following aggregation functions for discrete attributes: MODE, COUNT, PROPORTION, DEGREE. The following aggregation functions are considered for continuous attributes: AVERAGE, COUNT, PROPORTION, DEGREE. COUNT, PROPORTION and DEGREE features consider a number of different thresholds (e.g. PROPORTION>10%). Features of continuous attributes search for the best binary discretization of the attribute (e.g. COUNT(age>15)). Figure 9 lists a number of example features formed from movie data and web page data.



IMDB	movie	genre	EXISTS(Drama)	yes
	studio	mostPrevalentGenre	MODE(Horror)	yes
	actor	gender	PROPORTION(!Male)	> 0.75
	director	hasAward	PROPORTION(No)	≤ 0.5
	actor	ag	COUNT(≤54.0)	> 48
WebKB	pageLinkedFrom	numOutlinks	COUNT(≤10.0)	> 2
	pageLinkedFrom	urlPath	MODE(grads)	yes
	pageLinkedTo	numInlinks	AVERAGE	≤ 3

Figure 9: Example features formed by RPT.

Each feature produces a binary split of the training data. For example, MODE(actor-gender)=female tests whether a movie's actors are predominantly female. For a given feature, the algorithm searches for the value and threshold that best partition the class label. Feature scores are calculated from the class counts after splitting the data, using chi-square to measure correlation.

Table 1 gives the pseudocode for the RPT algorithm. The algorithm takes a set of subgraphs examples as input, a target class label to predict on the core object of the subgraphs and a list of attributes available to the algorithm for constructing features. A p-value cutoff is supplied as a stopping criterion.

The algorithm recursively partitions the subgraphs choosing features and binary splits greedily until further partitions no longer change the class distributions significantly. To choose a feature test the algorithm looks at each possible feature in turn.

The algorithm uses randomization tests to adjust for biases towards particular features due to the characteristics of the relational data (e.g. degree disparity). The attribute values are repeatedly randomized. For each randomization trial, the chi-square score for the best settings is recorded. These trials produce a sampling distribution of the maximum chi-square scores expected under the null hypothesis—that there is no association between the class label and the attribute values. This sampling distribution is then used to calculate an empirical p-value for the observed chi-square feature scores.

The current implementation of the RPT algorithm does not do any post-pruning of the tree, instead it used pre-pruning in the form of a p-value cutoff. If the p-value associated with the maximum chi-square score exceeds the cutoff the method returns without splitting the node and growing stops. Because the features considered are not necessarily independent we use a Bonferroni adjustment on the p-value cutoff to account for any dependence. All experiments reported in this paper used an adjusted alpha of 0.05/|attributes|. This threshold may be too conservative. Future work will explore using cross-validation to determine correct p-value threshold.

Once the tree-growing phase is halted the algorithm calculates the class distribution of the examples and stores it in the node. Laplace correction is applied to the distribution to improve the probability estimates [19].

```
RPT( examples, classLabel, attributes, pValueCutoff)
  create root node of tree
  growTree(root, examples, attributes)

growTree( node, examples, attributes )
  create featureTests for examples with attributes
    for each featureTest in featureTests
      determine possibleSettings for featureTest
      for each setting in possibleSettings
        calculate score of featureTest on examples
      maxSetting <- setting with max score
      calc pValue for featureTest w/maxSetting
      if pValue > pValueCutoff
        drop featureTest from consideration
    if all featureTests have been dropped
      nodeClassDist <- dist of classLabel in examples
      return //stop growing
    else
      nodeFeatureTest <- featureTest with min p-value
      create nodes leftChild and rightChild
      lChildExamples <- examples that pass nodeFeatureTest
      rChildExamples <- examples that fail nodeFeatureTest
      //recurse on partitioned data
      growTree(leftChild, leftChildExamples, attributes)
      growTree(rightChild, rightChildExamples, attributes)
```

Table 1: RPT algorithm.

Once an RPT model is learned over a set of training examples, the model can be applied to unseen subgraphs for prediction. The chosen feature tests are applied to the subgraphs and the examples travel down the tree appropriately. Each example ends up at a leaf node. The model then returns the probability distributions estimated for that leaf node and uses it to make a prediction about the class label of the example.

6. EVALUATION

6.1 Data sets

The first data set is drawn from the Internet Movie Database (www.imdb.com). We collected a sample of 1,364 movies released in the United States between 1996 and 2001. In addition to movies, the data set contains objects representing actors, directors, producers, and studios. In total, this sample contains approximately 46,000 objects and 68,000 links. We discretized movie opening-weekend box office receipts so that a positive class label indicates a movie earned more than $2 million in opening-weekend receipts (P(+)=0.45).

Figure 11 (top) shows degree disparity of three types of entities. We tested those differences using the Kolmogorov-Smirnoff (K-S) distance. The degree disparities for two types of entities—actors and producers—are statistically significant ($p<0.0001$); the degree disparity for directors is not significant. As discussed in section 4.2, even a degree disparity this small can have large effects. Figure 10 shows the linkage and autocorrelation of each object type. While directors have the highest value of relational autocorrelation, their linkage is quite low. As discussed in section 4.1, when linkage and autocorrelation are both high—as they are with studio objects—they bias learning algorithms toward features formed from objects of that type.

The second data set is drawn from Cora, a database of computer science research papers extracted automatically from the web using machine learning techniques [17]. We selected a set of 1,511 machine-learning papers along with associated authors, cited papers, and journals. The resulting collection contains 6,798 objects and 13,225 links. Figure [x] shows degree disparity for three types of entities in a collection of machine learning papers. The class label indicates whether a particular paper was assigned the topic of "neural networks" (P(+)=0.32). The degree disparities for all three types of entities—references, authors, and journals—are statistically significant ($p<0.0001$). Figure 11 (bottom) shows the linkage of papers with respect to other types of objects and the relational autocorrelation of paper topic (e.g., neural networks) with respect to those other object types.

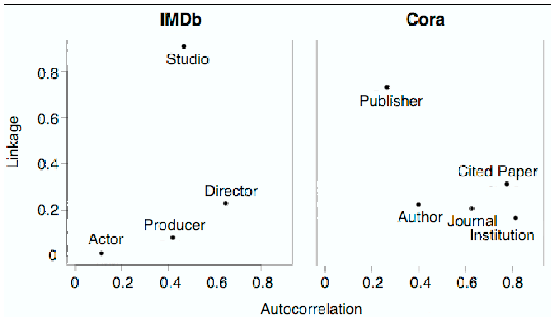


Figure 10: Linkage and autocorrelation for IMDb and Cora.

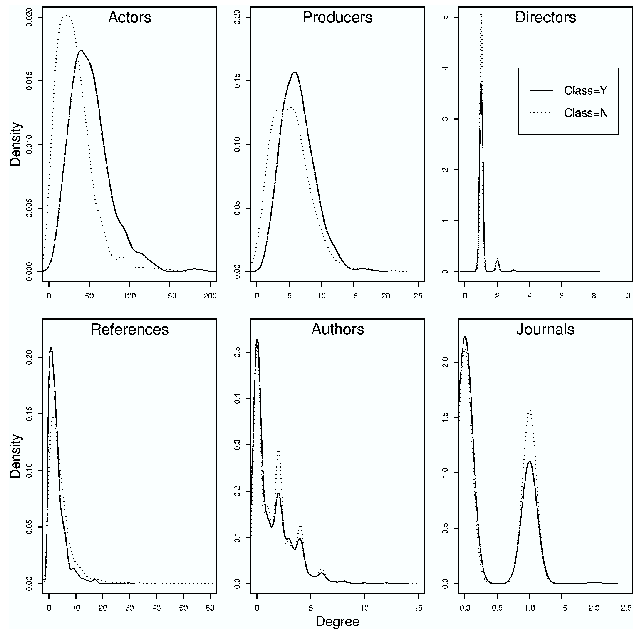


Figure 11: Degree disparity in IMDb (movies and their receipts) and Cora (papers and their topics).

The third data set is a relational data set containing information about the yeast genome at the gene and the protein level (www.cs.wisc.edu/~dpage/kddcup2001/). The data set contains information about 1,243 genes and 1,734 interactions among their associated proteins. The class label indicates whether or not a gene's functions include transcription (P(+)=0.31).

The fourth data set is the WebKB described in section 2. The data consists of a set of 3,877 web pages from computer science departments. The class label indicated whether a page is a student page (P(+)=0.51).

6.2 Classification tasks

Our first task uses the IMDb data set described above where the class label indicates the movie earned more than $2 million in opening-weekend receipts. We created a classification task where the only feature correlated with the class label was the degree of the objects in the relational data structure. Recall that movies with a positive class label tend to have higher degree with respect to actors and producers (there is no significant difference in director degree nor studio degree). On each actor, director, producer, and studio object we added 6 random attributes (3 discrete and 3 continuous). Discrete attributes were drawn from a uniform distribution of ten values; continuous attribute values were drawn from a uniform distribution in the range [0,10]. The model considered 4 degree features, one for each type of object linked to the movie.

The second task also used the IMDb dataset, but used both the structure and the attributes in the original data. The models were to predict movie success based on 8 attributes, such as the genre of the movie and the year of a producer's first film. The model also considered 4 degree features, one for each type of object linked to the movie.

The third task used the Cora data set where the class label indicates whether a paper's topic is "neural networks." In addition to papers, the data contained objects representing authors, cited papers and journals. The models had 15 attributes available for classification, including attributes such as a cited paper's high-level topic (e.g. Artificial Intelligence) and an author's number of publications. Three degree attributes counted the number of cited papers, authors, and journals.

The fourth task used the gene data set where the models were to predict whether a gene's functions include transcription. The models used 7 attributes for prediction, including gene phenotype, complex, and interaction type; and two degree attributes for links to interactions and other genes.

The fifth classification task used the WebKB data set and the models were to predict whether a page is a student page. The models used a total of 10 attributes for prediction. These included attributes like the URL path and host, as well as structural attributes such as the number of in links and out links (directional degree) of each page.

For each of the five tasks, we tested four models: (1) an RPT model that used conventional significance tests (CTs) to evaluate feature splits and (2) an RPT model that used randomization tests (RTs) to measure significance. As a baseline of comparison, we flattened the relational data and applied (3) the C4.5 algorithm [20]. Finally, we applied (4) a relational Bayes classifier (RBC) [18].

The Relational Bayesian Classifier (RBC) is a modification of the Simple Bayesian Classifier (SBC) for relational data. The algorithm uses an independent value estimator (INDEPVAL) which assumes each value of a multiset is independently drawn from the same distribution. For estimation, each value of each set (e.g. actor ages) is considered to be an independent instance. The class label is duplicated and paired with each attribute value. Each pair is considered to be independent evidence. During inference the probability of each value is computed and multiplied into the overall probability independently. See [18] for more details.

To evaluate the models, we measured accuracy, area under the ROC curve (AUC), and the number of attributes used in the model. Note that the RBC model includes all attributes whereas the classification trees are selective. Also, because the C4.5 models return classifications only (not probability estimates) we could not calculate AUC for the C4.5 models. For the tree models, we measured the size of the tree and the proportion of structural features, such as degree features. The proportion is weighted to reflect the proportion of training instances which travel through that node (e.g. leaf nodes have less weight than root node). The experiments used ten-fold cross-validation. In the IMDb data set—because studio objects have such high linkage—the test sets were created by stratified sampling on studios. We randomly sampled studios from three sets (studios with high, medium, and low degree) thus creating test sets of roughly equal proportion.

6.3 Results

Figure 12 shows accuracy and AUC results for each of the four models on the five classification tasks. We used two-tailed, paired t-tests to assess the significance of the results obtained from the ten-fold cross-validation trials. The null hypothesis is that there is no difference between two approaches; the alternative is that there is a difference between two approaches. We compared RTs to each of the other three aproaches (CTs, C4.5, RBCs). An asterisk above the model indicates a significantly different performance from the RTs.

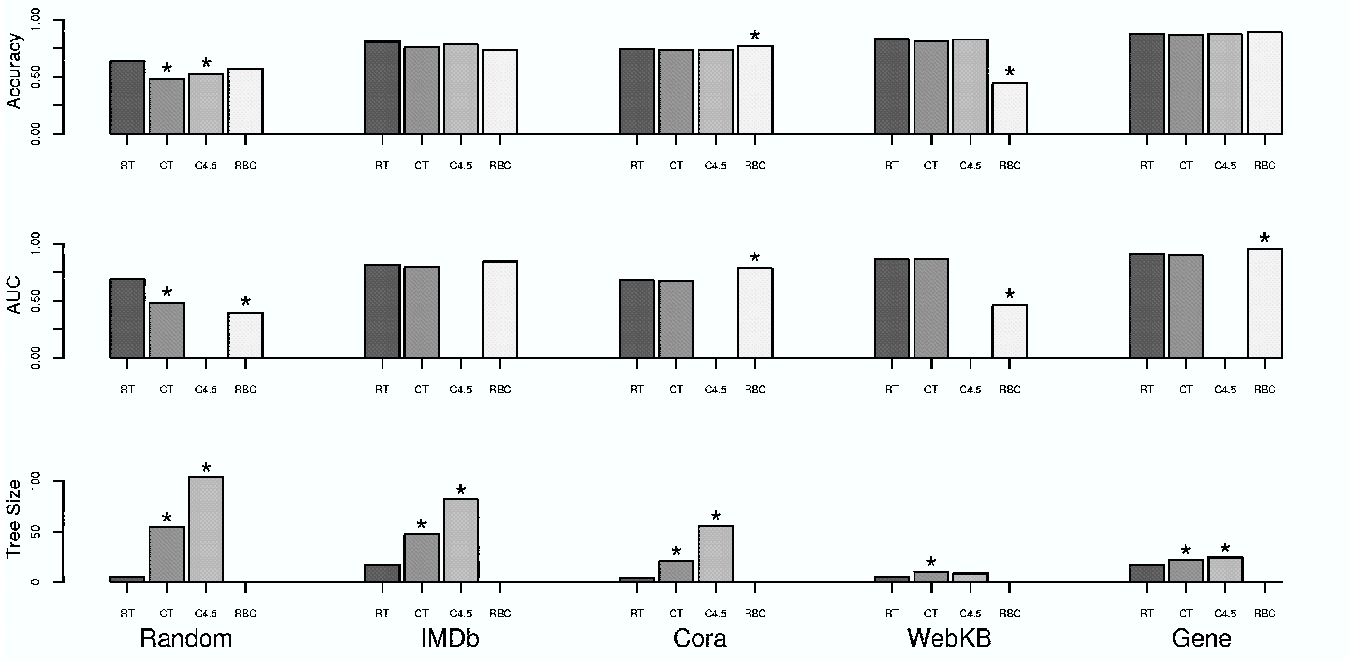


Figure 12: Accuracy, AUC and tree size for four models across the various classification tasks.

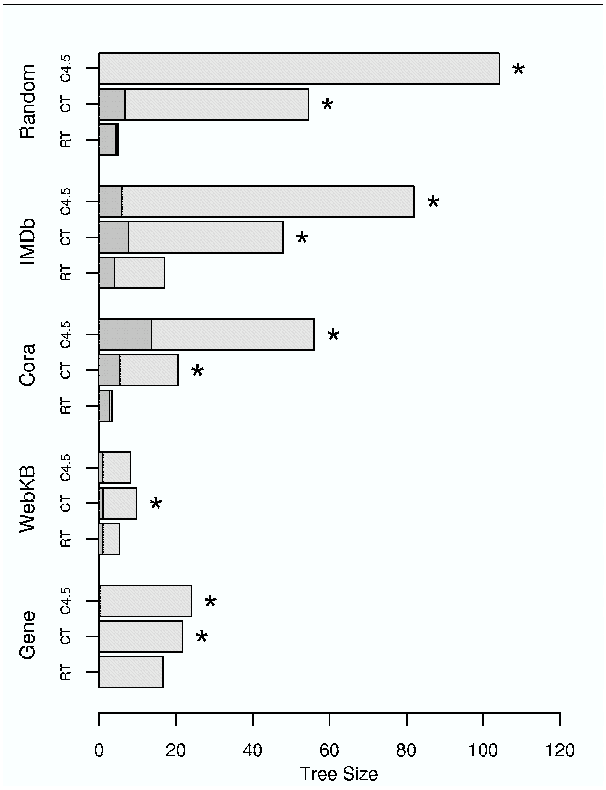


Figure 13: Number and composition of nodes in trees. Shading represents weighted proportion of degree features.

The results from the IMDb data set with random attributes (Random in Figure 12 and 13) support three claims. First, RTs can adjust for linkage and autocorrelation and build more accurate models. RPTs using randomization tests (RTs) perform significantly better than the other three models. RT models had higher accuracy than both CT models and C4.5 models. Although the accuracies of RT models were not significantly different than the RBC models, the RTs did have significantly higher AUC. These results indicate the potential for biased models to select attributes that hinder performance. The lower performance is due to selection of random attributes on studio objects. Features involving these attributes have high variance because of the low effective sample size of studios. If objects with high linkage and autocorrelation, such as studios, are present in the data, attributes of those objects will be selected even if they have no correlation with the class. Second, aggregation functions can cause misleading correlations in the presence of degree disparity. In the IMDb data set with random attributes, the only predictive structure is the degree disparity of "actor" and "producer" objects. By our measure of weighted proportion, approximately 4/5 of the features in trees built with conventional tests (CT) consisted of features derived from random attributes that served as surrogates for degree. Third, the results from the Random data set show that models that do not adjust for these biases can add unnecessary complexity. Trees built with conventional tests were, on average, an order of magnitude larger than the size of the trees built with randomization tests.

The results from experiments on real data show that RTs achieve comparable performance to CTs and C4.5 models, both in accuracy and AUC. However, the trees had radically different structure. Figure 13 summarizes the features used in RPT trees built with conventional tests and randomization tests, as well as trees built with C4.5. Each bar expresses both the size of the tree and the weighted proportion of degree attributes, averaged over the cross validation trials. Note that in data sets where degree disparity is predictive (IMDb, and Cora), RTs give higher weight to degree features.

7. CONCLUSIONS

We have shown that it is possible to extend conventional probability estimation tree algorithms to work with relational data. The RPT models built using randomization tests performs equivalently to RPT models using conventional hypothesis tests but the trees are significantly smaller. This supports our claim that common characteristics of relational can bias feature selection and result in excessively complex models. Randomization tests adjust for both the increased bias due to degree disparity and the increased variance due to linkage and autocorrelation.

Models that are not selective (e.g. RBC) do not suffer from these biases. This can result in significantly better models, but we then lose the interpretability of the selective models. RBCs exhibited significantly lower performance on datasets where degree was the only feature correlated with the class (Random, WebKB). More work needs to be done to explore the situations in which RBC performance is distinct from RPT. We may be able to combine the strengths of RPT feature construction and selection methods with the low variance parameter estimates of RBC models.

Future work will investigate further enrichments to the RPT model. Extending the algorithm to consider multiway feature splits and alternative methods of modeling continuous attributes should improve performance. We will conduct a series of more focused ablation studies to determine which characteristics of the algorithm are most beneficial and to further understand the complexities of relational data and their effect on modeling choices.

8. ACKNOWLEDGMENTS

Helpful comments and assistance were provided by Amy McGovern. This research is supported by DARPA and NSF under contract numbers F30602-01-2-0566 and EIA9983215, respectively. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA, NSF, or the U.S. Government.

9. REFERENCES

[1] A Blockeel, H., and De Raedt, L. Top-down induction of first-order logical decision trees. Artificial Intelligence, 101: 285-297, 1998.

[2] C. Brodley and E. Rissland. Measuring concept change. Training Issues in Incremental Learning: *Papers from the 1993 Spring Symposium* (pp. 99-108). Menlo Park, CA: AAAI Press, 1993.

[3] C. Cortes, D. Pregibon, and C. Volinsky. Communities of Interest. *Proceedings of the Fourth International Symposium on Intelligent Data Analysis*, 2001.

[4] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam and S. Slattery. Learning to Extract Symbolic Knowledge from the World Wide Web. *Proceedings of the 15th National Conference on Artificial Intelligence*, 1998.

[5] N. Cressie. *Statistics for Spatial Data*. Wiley, 1993.

[6] E. Edgington. *Randomization Tests*. New York: Marcel Dekker, 1980.

[7] S. Dzeroski and N. Lavrac, editors. Relational Data Mining. Springer-Verlag, 2001.

[8] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *IJCAI'99*. 1300-1309, 1999.

[9] D. Jensen and P. Cohen. Multiple comparisons in induction algorithms. *Machine Learning* 38(3):309-338, 2000.

[10] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proc. of the 19th Int. Conf. on Machine Learning*. Morgan Kaufmann. 259-266, 2002.

[11] D. Jensen, J. Neville and M. Hay. Avoiding bias when aggregating relational data with degree disparity. Submitted to the *20th Int. Joint Conf. on Machine Learning*, 2003.

[12] D. Jensen, J. Neville and M. Rattigan. Randomization tests for relational learning. Submitted to the *18th Int. Joint Conf. on Artificial Intelligence*, 2003.

[13] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM* 46:604-632, 1999.

[14] J. Knobbe, A. Siebes and D. Van der Wallen. Multi-relational decision tree induction. In *Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, 1999.

[15] S. Kramer. Structural regression trees. In *Proceedings of the 13th National Conference on Artificial Intelligence*, 1996.

[16] S. Kramer, B. Pfahringer, and C. Helma. Stochastic propositionalization of non-determinate background knowledge. In *Proc. of the 8th International Workshop on Inductive Logic Programming*, pp. 80--94. Springer Verlag, 1998.

[17] A. McCallum, K. Nigam, J. Rennie, & K. Seymore. A machine learning approach to building domain-specific search engines. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence*, 1999.

[18] J. Neville, D. Jensen, B. Gallagher and R. Fairgrieve. Simple estimators for relational Bayesian classifiers. Submitted to the *18th Int. Joint Conf. on Artificial Intelligence*, 2003.

[19] F. Provost and P. Domingos. Well-trained PETs: Improving probability estimation trees. CDER Working Paper #00-04-IS, Stern School of Business, NYU, 2000.

[20] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[21] J.R. Quinlan. Simplifying decision trees. International Journal of Man-Machine Studies, 27: 221-234, 1987.