Iterative Classification

Applying Bayesian Classifiers in Relational Data

Jennifer Neville
jneville@cs.umass.edu

June 2000

Technical Report
TR-00-31

Computer Science Department
140 Governor's Drive
University of Massachusetts
Amherst, MA 01003-4601

Abstract

Relational data offer a unique opportunity for improving the classification accuracy of statistical models. If two objects are related, inferring something about one object can aid inferences about the other. We present an iterative classification procedure that exploits this characteristic of relational data. This approach uses simple Bayesian classifiers in an iterative fashion, dynamically updating the attributes of some objects as inferences are made about related objects. Inferences made with high confidence in initial iterations are fed back into the data and are used to strengthen subsequent inferences about related objects. We evaluate the performance of iterative classification on a corporate dataset, using a binary classification task. Experiments indicate that iterative classification significantly increases accuracy when compared to a single-pass approach.

1. Introduction

The past two decades have seen a dramatic increase in the amount of stored information, creating a need for a new generation of automated and intelligent data analysis techniques. Many of the data being captured are relational in nature, yet most analysis techniques work with "flattened" attribute-value data. Attribute-value data record the characteristics of a set of homogeneous and statistically independent objects, whereas relational data record characteristics of heterogeneous objects and the relations among those objects. The inherent structure of relational data presents a unique opportunity to use knowledge of one object to infer something about related objects. The goal of this work is to explore how conventional analysis techniques can be used in new ways to exploit this opportunity.

Classification is a well-studied problem in the field of data analysis for which many machine-learning techniques have been developed. One of these techniques is the simple Bayesian classifier. This classifier offers good performance in many domains and it is also simple to train and easy to understand. We investigate using simple Bayesian classifiers in an iterative fashion to improve classification accuracy by taking advantage of relational information in the data.

The hypothesis underlying this approach is that if two objects are related, inferring something about one object can help you infer something about the other. Inferences made with high confidence in initial iterations can be fed back into the data to strengthen inferences about related objects in subsequent iterations. Experimental evidence reported here shows that iterative classification leads to a significant increase in classification accuracy when compared with a single-pass approach. This suggests that there are distinctive characteristics of relational data that can be used to improve classification accuracy.

2. Relational Knowledge Discovery

The amount of data being collected by organizations and businesses around the world has grown explosively in the past 10 years. AT&T logs over 35 gigabytes of telephone call data every day; The Second Palomar Observatory Sky Survey (POSS-II) has over 3 terabytes of high-resolution image data from an astronomical sky survey; WalMart has transaction detail from over 2,900 stores in its 7.5 terabyte data warehouse (Pregibon 2000, Brodley and Smyth 1996, Palace 1996). In addition to the many business, government, and scientific databases growing at an unparalleled rate, the World Wide Web grows by roughly a million electronic pages every day, developing into the world's largest interconnected information base (Chakrabarti et al. 1999).

For structured data captured and stored on a daily basis, a relational model is the most commonly chosen representation (Friedman et al. 1999). Relational databases are used routinely to store everything from marketing and sales transactions, to scientific observations and medical records. The Web's network of hyperlinked pages is another good example of a relational data structure. The power of relational data lies in combining intrinsic information about objects in isolation, with information about the connections among objects.

Hidden in this expanse of information are vital clues and general regularities that could be used to improve decision making, if they could be discovered and represented. For example, customer demographic, lifestyle, and purchasing information could be used to reduce the cost of a direct-mail marketing campaign by targeting a set of consumers most likely to buy the product being promoted. Credit card companies could use transaction histories, purchase patterns, and examples of past fraudulent activity to predict possible cases of fraud in current credit card transactions.

The volume and pace of data storage far exceeds our ability to summarize and evaluate data without the use of automated analysis techniques. Knowledge discovery is the field evolving to provide automated tools for these growing amounts of data, building on techniques from statistics, artificial intelligence and databases. Knowledge discovery is the process of automatically identifying previously unknown, and potentially useful patterns in data (Fayyad et al. 1996). Knowledge discovery techniques are finding wide-ranging applications in science, government and business.

Most current knowledge discovery techniques take attribute-value data as input, where each instance is assumed to be structurally identical and statistically independent. For example, current techniques can learn simple rules to use for diagnosis from a collection of independent medical records with identical fields. However, in some cases a useful diagnostic rule will need to consider genetic and environmental factors in addition to patient records. In this situation we would like to examine an interconnected group of patients, their medical records, family members, home environments, and workplaces. Such a relational data structure violates the previously mentioned assumptions by allowing numerous objects of varying structure to have associations among themselves.

Relational data can be converted into attribute-value data by transforming relationships into attributes. For example, <father's blood type> and <mother's blood type> could be represented as attributes of a patient rather than as associations among related family members. In order to flatten the data we must decide in advance which combinations of relationships may be useful and which of the large number of potential relational attributes to include. The flattening process removes the richer relational structure from the data, and in doing so, may omit information crucial to the discovery of useful patterns. Relational knowledge discovery looks for new ways to make use of relational structure in discovery techniques, in order to support and enhance the pattern detection process.

3. Classification in Relational Data

Relational knowledge discovery builds on existing work in machine learning, statistics and social network analysis to exploit the additional knowledge implicit in relations. There are at least two approaches to take when developing relational knowledge discovery techniques. One approach is to devise entirely new techniques for relational data structures, leaving behind the conventional attribute-value framework. An alternative approach is to adapt and extend current attribute-value techniques for use with relational data. The latter approach stands to benefit from the advances made in the field of knowledge discovery over the past 10 years, utilizing known techniques that have been fine-tuned on attribute-value data. The work reported here takes the second approach, using a conventional classification technique in new ways on relational data.

Classification is probably the oldest and most widely studied of all the knowledge discovery tasks, and it is one area where techniques stand to benefit from the relations among instances. Classification takes a set of labeled training examples and builds a model to map previously unseen, unlabeled examples to discrete class values. For example, we may want to identify cases of cellular phone fraud using call data and customer histories (Fawcett and Provost 1997), or we may want to predict whether a person is a potential money launderer based on bank deposits, international travel documents and known associations (Jensen 1997).

The goal of this work is to explore and evaluate a specific application of classification techniques on relational data: simple Bayesian classifiers. Simple (or "naive") Bayesian classifiers (SBCs) make the simplifying assumption that all attributes are independent given the class. Empirically, it has been found that SBCs perform surprisingly well in many domains. Domingos and Pazzani (1997) provide a formal analysis of the reasons for this robust behavior and show that SBCs produce optimal predictions of class labels, even when the assumption of independence is violated by a wide margin. An attractive feature of SBCs is they require no explicit search through the space of possible models. Instead, a model is built using simple probabilities, estimated from the training examples. This results in a classifier that is both incremental and relatively easy to code.

Simple Bayesian classifiers take traditional attribute-value data as input. In order to use SBCs with relational data we have to flatten the data first. As mentioned above, flattening

the data removes much of the richer relational structure. Flattening relational data is task-specific. Once flattened for one classification problem, it is often misleading or impossible to use the flattened data to analyze an alternative classification problem. However, if we maintain a relational representation of the data and flatten dynamically only when needed, then the relational structure is still accessible and it is possible to leverage it in order to improve predictions.

Keeping the data in a relational format preserves the relationships among objects so they can be used in analysis dynamically. A relational representation makes it possible to extract data, perform a series of calculations and then feed the results back into the relational structure for use in future calculations. The ability to perform iterative calculations in this manner is one of the benefits of maintaining a relational data representation. For example, some measures of centrality in social network analysis (Wasserman and Faust 1994) can only be calculated in such an iterative fashion. Kleinberg's Hubs and Authorities algorithm for Web searching (1998) also uses iterative calculations in this manner. Below, we will examine how to use an iterative application of SBCs to improve classification accuracy in relational data by using initial inferences to aid later inferences about related objects.

4. Iterative Application of Selective Bayesian Classifiers

4.1. Learning a Simple Bayesian Classifier

The simple Bayesian classifier (SBC) is a probabilistic method for classification that makes predictions as follows (Mitchell 1997). Let A_1, A_2, ..., A_k be attributes used to predict a discrete class **C**. For a given set of attribute values v_1 through v_k, the classifier calculates the conditional probability for each class label c_i. The optimal prediction, under a zero-one loss function, is the class label c_i for which $P(c_i \mid v_1 \wedge v_2 \wedge \ldots \wedge v_k)$ is maximized.

Bayes' Rule states that we can determine the probability that a particular example, represented by the attribute vector $\mathbf{E} = \{ v_1 , v_2 , \ldots , v_k \}$, is of class c_i with the following formula (for notational simplicity, we will substitute the notation $P(c_i \mid \mathbf{E})$ for $P(\mathbf{C} = c_i \mid \mathbf{E})$ along with other similar substitutions):

$$P(c_i \mid \mathbf{E}) = P(\mathbf{E} \mid c_i) \, P(c_i) / P(\mathbf{E})$$

The probability $P(\mathbf{E})$ in the denominator is the same for each class label c_i; consequently, the denominator can be dropped from the equation if the probabilities $P(c_i \mid \mathbf{E})$ are normalized across all class labels c_i. Also, instead of estimating the joint, conditional probability $P(\mathbf{E} \mid c_i)$, the SBC makes the assumption that the individual attributes of **E** are conditionally independent given **C**. As such, the conditional probability can be rewritten as follows:

$$P(c_i \mid \mathbf{E}) = P(c_i) \, \Pi_k \, P(v_k \mid c_i)$$

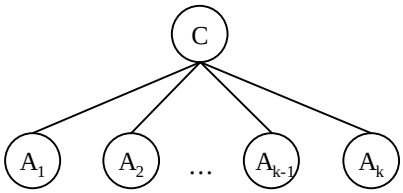

Figure 1: Graphical representation of a Simple Bayesian Classifier

An SBC model consists of a set of associated probability distributions, which the model uses to make its class predictions. In particular, a model is comprised of a single discrete distribution for the class, and a set of associated conditional probabilities that characterize probability distributions for each attribute given a particular class label. For discrete attributes, the model maintains a discrete distribution ranging over all possible values of that attribute, storing the probabilities $P(v_k | c_i)$ for each value v_k, given an instance of class c_i.

Discrete distributions will contain zero counts when a class label and attribute value never occur together in the training data. A zero count results in a zero probability estimate, causing the entire joint probability to become zero. For this reason, a Laplace correction is incorporated into all discrete probability estimates to adjust for zero counts (Domingos and Pazzani 1997). The uncorrected estimate of $P(v_k | c_i)$ is n_{ik} / n_i where n_{ik} is the number of times class c_i and value v_k occur together, and n_i is the total number of times class c_i occurs in the training set. The Laplace corrected estimate of $P(v_k | c_i)$ is $(n_{ik} + f) / (n_i + f n_k)$ where n_k is the number of distinct values of attribute A_k and $f = 1 / n$ where n is the number of examples in the training set.

For continuous attributes, we model the probability distributions using kernel density estimators (Silverman 1986). Many implementations of SBCs either discretize continuous attributes or model them with Gaussian distributions, which can be conveniently represented in terms of their mean and variance. However, choosing appropriate discretizations can be difficult and the assumption that an attribute obeys a Gaussian distribution may not hold in all domains. It has been shown that Bayesian classifiers using kernel density estimators for continuous probability distributions perform better than those that assume a single normal distribution (John & Langley 1995). Our implementation of a kernel density estimator stores every value of an attribute seen during training. When asked for an estimate of $P(v_k | c_i)$, the estimator calculates a Gaussian distribution (kernel) around each observed value and returns a density averaged over the set of all kernels. This method of kernel estimation produces more accurate probability calculations in domains where the Gaussian distribution assumption is violated, with only moderate computational cost.

Training an SBC model is quite simple and can easily be done incrementally. For each new labeled training instance the model increments the counts for the class label and each of the discrete attribute values, and stores the values of each of the continuous attributes. Once the model is learned it can be applied to predict classes of previously unseen instances.

4.2. Selective Bayesian Classifier

Although SBCs are known to be robust in the presence of irrelevant features (Duda & Hart 1973), we would like the model to select and use only the relevant attributes to improve the comprehensibility of the results. Relational data have a large number of potential attributes since each object has not only its own intrinsic attributes, but also relational attributes involving objects to which it is linked. Removing useless attributes before classification, and consequently reducing the number of attributes used by the final classifier, simplifies the results and can improve the comprehension of human analysts.

The model can use either a filter approach or a wrapper approach to detect and remove irrelevant attributes (Kohavi & John 1997). In the filter approach, a subset of features is selected as a preprocessing step, ignoring the effects of the selected set on the performance of the model. In the wrapper approach, the model itself is used as a black box in the evaluation function, guiding the search for a good subset of features.

The algorithm that we implemented for selectivity can be described as a wrapper model. In order to reduce the size of the feature subset search space, a "relevance" weight is calculated for each attribute, which signifies the correlation of the attribute to the target class. To use this approach, the weights must be comparable for discrete and continuous attributes. White & Liu (1994) suggest the use of p-values for this reason. P-values calculate the probability of incorrectly rejecting the null hypothesis when it is true — where the null hypothesis is that the attributes and the class are independent. P-values can be calculated easily for both discrete and continuous distributions using simple statistical significance tests. We use the G-statistic to obtain p-values for discrete attributes, and the Kolmogorov-Smirnoff test for continuous attributes (Sachs 1982). The resulting p-values are used to order the attributes and the SBC model is then used to determine a p-value threshold that maximizes cross-validated accuracy on the training set. The attributes with p-values less than, or equal to, the chosen threshold are used in the final classifier; all other attributes are ignored.

Overview of learning a selective SBC model:

1. For each labeled training instance
   a. Update class distribution
   b. Update conditional distribution for each attribute
2. Calculate p-values for discrete and continuous attributes
3. Determine threshold that maximizes cross-validated accuracy
4. Select attributes with p-values ≤ threshold

4.3. Iterative Application of Bayesian Classifiers

Relational datasets present a special opportunity for improving classification. The opportunity exists if, when two objects are related, inferring something about one object can help you infer something about the other. For example, if two people are involved in business together and one of them is identified as a money launderer then it is more likely that the other is also involved in money laundering. In this situation, knowledge inferred

about one object can be used to improve inferences about related objects. The ability to exploit associations among objects in this manner to "discover knowledge" has wide-ranging applications in any field with relational data, including epidemiology, fraud detection, ecological analysis and sociology.

A relational classification technique, which uses information implicit in relationships, should classify more accurately than techniques that only examine objects in isolation. Relational classification techniques could be particularly useful in domains where we have more information about relationships among objects than about their intrinsic properties in isolation. For example, we may be interested in identifying potential money-laundering operations based on bank deposits and business connections (Jensen 1997). In such a situation, the existence of an employee making large cash deposits for more than one business gives little information as to the legitimacy of those businesses. Many service and retail companies have high volumes of cash sales and it's not uncommon for a person to be employed by more than one company. However, if one of the businesses is discovered to be a front company for money laundering, then the related businesses are more likely to be front companies as well. In this case, the relationship of a common depositor is more useful in the context of knowledge about the related companies.

There are multiple ways to approach classification in a relational context. One can ignore related objects and classify based only on the properties of an object in isolation. One can look at the properties of both the object and its related objects in a static manner, by taking a snapshot of the relational context at some time prior to classification. Or one can employ a more dynamic approach, using properties of related objects and updating those properties as predictions about those related objects change. *Iterative classification* is a dynamic method of relational classification, which uses SBCs in a dynamic way to fully leverage the structure of relational data.

For example, in a data set we describe in section 5.1, relational data structures represent companies, their subsidiaries, corporate stockholders, officers and board members. Companies are linked indirectly through stockholders and through people serving simultaneously on several boards (see figure 2). In such an interlocking structure we have both *intrinsic* and *relational* attributes. Intrinsic attributes record characteristics of objects in isolation, for example, company type or officer salary. Relational attributes summarize characteristics of one or more related objects, for example, a company's number of subsidiaries or the maximum salary of any board member.

Relational attributes fall into two categories which we will call *static relational* and *dynamic relational*. Any intrinsic attribute has the potential to be predicted by an SBC model; from the same company data we could predict any of the intrinsic attributes mentioned above. Static relational attributes involve "known" intrinsic attributes of related objects and as such they can be computed without the need for inference. The values of static relational attributes remain constant over the course of classification. Dynamic relational attributes involve "inferred" intrinsic attributes of related objects so they require that at least some related objects be classified before the attribute can be

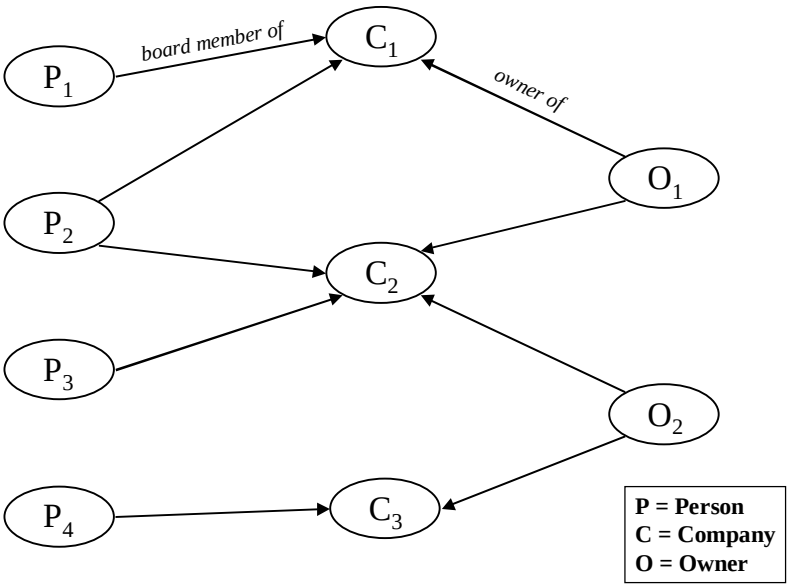

Figure 2: Graphical representation of corporate data linkage

computed. The values of dynamic relational attributes may change as classification progresses and inferences are made about related objects.

For example, if we were predicting company type:

- Static relational attributes
  - number of board members who have the title CEO
  - average salary of officers.
- Dynamic relational attributes
  - most prevalent *type* of corporate stockholder
  - maximum number of subsidiaries that share the same *type*.

For notational simplicity, for the remainder of this paper we will refer to intrinsic and static relational attributes as static attributes, and dynamic relational attributes as dynamic attributes.

In a relational corporate data set, knowing the type of one company might help us infer the type of another company to which it is related, and vice versa. For instance, we may find that individuals tend to serve on boards of companies with the same type, so if a person is on the board of both company X and company Y, and company X is a bank, then company Y is more likely to also be a bank. Or we may find that companies tend to own stock in companies with the same type, so if a company owns company X and company Y, and company X is a bank, then company Y is more likely to also be a bank. In situations of this type, the relations among objects assist the inferences.

In iterative classification, a model is built using a variety of static and dynamic attributes. Classifiers that include dynamic attributes rely on the previous (inferred) classification of

related objects. When training the model, the class labels of all objects are known and consequently the values of all dynamic attributes are also known.

The trained classifier is then applied to previously unseen examples in which the class labels are unknown. Initially, because class labels of related objects are unknown, values of dynamic attributes are also unknown, but their values can be estimated as the classification progresses. At the onset, the classifier makes predictions for all objects based only on the values of static attributes. Classifications made with high confidence are accepted as valid and are written into the data as "known" class labels. SBCs are useful for iterative classification because each prediction has an associated probability estimate that can be used as a confidence score.

After some percentage of the most certain classifications are "accepted" the classifier starts the next iteration, recalculating all dynamic attributes in light of this new information and proceeding with classification once again. At each iteration, additional dynamic attributes are filled in and a greater percentage of classifications are accepted.

Because each prediction is both recalculated and reevaluated for each iteration, a prediction about a given object may change over the course of iterations. If the probability associated with a particular prediction falls out of the top percentage of accepted predictions, the inference will be removed from the data. Also, if the predicted class label changes for a particular object (and the prediction is accepted), the new class label will be written into the data for that object.

After a given number of cycles, when all classifications have been accepted, the process terminates. We conjecture that iterative classification will produce more accurate predictions of class values than conventional classification involving intrinsic and static relational attributes alone.

Overview of iterative application of SBC model:

1. Build SBC model on fully labeled training set
2. Apply trained model to test set of N instances. For each iteration i : 1 to m
   a. Calculate values for dynamic relational attributes
   b. Use model to predict class labels
   c. Sort inferences by probability
   d. Accept k class labels, where $k = N (i / m)$
3. Output final inferences made by model on test set

4.4. Necessary Conditions

We conjecture that a relational dataset must exhibit several characteristics before an iterative classification approach will improve on a single-pass technique. An initial outline of these characteristics is given below; however, further investigation is needed to determine the exact nature and scope of these conditions.

First, the floor classification accuracy, using only static attributes, should not be too high. If a classifier can make highly accurate predictions without dynamic attributes, there is little room for improvement via iteration. For this reason, there must be insufficient predictive power in the static attributes of the dataset for iteration to exhibit an increase in accuracy over a single-pass approach.

Next, inferences made by the model must be relevant to the classification task. If when objects are related, an inference about one object *does not* help subsequent inferences about the other objects, then dynamic attributes will not aid classification. The relevance of dynamic attributes can be gauged with a single "full knowledge" classification pass — where the true class labels of related objects are used to calculate the values of dynamic attributes. Such a test indicates how effective the dynamic attributes would be if the inferences made by the model were 100% accurate; the test reveals the ceiling accuracy for the chosen set of attributes. If the ceiling accuracy is not significantly higher than the floor accuracy (using only static attributes), iteration will not produce a discernible effect.

Also, the dataset must be sufficiently connected. An iterative approach uses relational structure to maximize the use of its inferences. Because the results of classification are spread through the relational structure by way of dynamic attributes, if the dataset has insufficient linkage, there is less opportunity to make use of prior inferences. However, what constitutes "sufficient" linkage is not clear, and it may vary significantly across datasets. Both the degree of linkage, as well as the type of linkage, may affect the results of iterative classification. Further exploration is needed to determine the success of iterative classification for various types of relational structures.

Finally, there must be information present in the data to start off the iteration process. Initial classifications are made using only static attributes; therefore the classification model must have a way of making some initial inferences accurately. If none of the initial inferences are correct, then all subsequent predictions will be misled by those inferences that are accepted. In order to make accurate predictions in the first iteration but to still have room for improvement over the course of iterations, we must have attributes that represent "islands of certainty."

Islands of certainty denote knowledge from which some, but not all, objects can be classified accurately, with high confidence. Examples of islands of certainty include:

- A highly predictive static attribute that is missing in many instances but known for some
- A static attribute for which some values are highly predictive of particular class labels but other values are not
- A partially labeled dataset

The inferences made from islands of certainty jump-start the iterative procedure, feeding dynamic attribute calculations and improving predictions about related objects. In this way, knowledge spreads out through the data. Without such islands, the performance of iterative classification may degrade. Future work includes exploring extent of this

degradation and determining the size, type and number of islands needed for successful iterative classification.

Before there is an opportunity to gain from iterative classification we should be operating in a domain that meets these conditions. Otherwise flattening the data and classifying a single time will result in accuracies comparable to those achieved with iterative classification.

5. Experimental Evaluation

5.1. Corporate Dataset

The data set used for these experiments records the intrinsic and relational features of publicly traded corporations. The data are drawn from documents filed with the US Securities Exchange Commission (SEC). Due to the size of the entire database, we chose to work with data from only two industries, banks and chemicals companies. Data are maintained separately for each industry in the SEC database, so substantial consolidation was needed to combine data from two industries.

The data consist of companies, their board members and officers, stockholders, contractors and subsidiaries. The data set contains 2142 central companies (892 chemical companies and 1250 banks). It also contains 18679 related companies: 5201 corporate owners, 969 contractors, and 12509 subsidiaries. Owners, contractors, and subsidiaries do not have the same intrinsic attributes as the banks and chemical companies, so we chose to represent then as separate objects. In addition to these objects, the data set also contains 25591 people who serve as officers and directors of the companies.

We selected a relatively simple task: to classify companies as to their industry, either bank or chemical, using both relational and intrinsic attributes. Classification of companies by type is a surrogate task intended to illustrate the potential of iterative classification in other domains with similar organizational structure, such as fraud detection or money laundering analysis. Iterative classification is not restricted to binary classification tasks. Because the SBC makes prediction for each class label, the approach could easily be used for classes with more than two labels. Multiple class labels however, would make the queries for calculating and updating attribute values more complex, and would make the ROC curve analysis (section 5.4) more difficult.

The data ontology is displayed below in figure 3. Nodes in the graph represent the objects in the data set. Links in the graph correspond to possible relationships among objects in the data set. Italicized labels indicate link or object type. All other labels correspond to intrinsic data associated with the links and objects. A distinctive feature of this ontology is that companies are never linked to directly to other companies; they are only linked indirectly through people, owners and contractors.

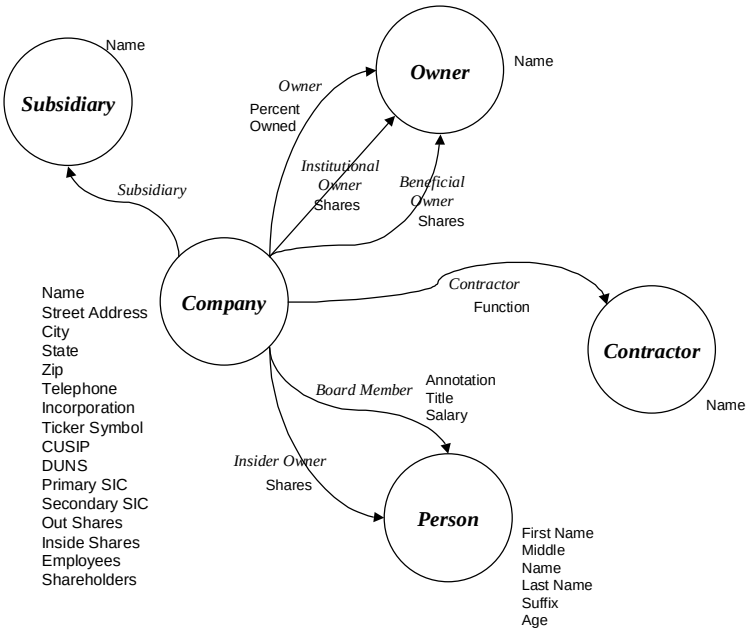

Figure 3: Corporate data ontology

In the experiment reported below, we used four attributes for each company:

- state of incorporation (static)
- number of subsidiaries (static)
- company *X* is linked to more than one chemical company through its board members: true/false (dynamic)
- company *X* is linked to more than one chemical company through its insider owners: true/false (dynamic)

A few informal tests with additional attributes showed no substantial improvement in accuracy, so for efficiency reasons the attributes were limited to these four. Although the SBC classifier itself is fast, there are efficiency issues regarding attribute calculation in the data set. Each dynamic attribute calculation involves aggregating information about company objects two links away. Recalculating a single dynamic attribute for companies in a sample takes approximately three minutes of computation. Because each iteration involves recalculating every dynamic attribute, experiments were greatly facilitated by keeping the number of attributes to a minimum.

5.2. Sampling

Devising a disjoint training and test set was a challenging task. It has been shown that partial sampling of linked data can bias statistical estimates of relational attributes (Jensen 1998). Fractional sampling of linkage in the data can produce under- and over-estimates of attributes that will reduce the effectiveness of an induction algorithm. SBCs assume that the distribution of features is comparable between training and tests sets, so their effectiveness depends on a sampling procedure that produces similarly linked training and tests sets. Also, because the iterative classification involves inferences made

about linked companies, a desirable sampling procedure would retain as much linkage to other companies as possible, to avoid handicapping iterative classification.

The sampling procedure used is similar to the exhaustive approach described by Jensen (1998). The process for creating two samples A & B from the set of all companies X is as follows.

1. Do until *X* is empty:
    a. Do until a company is placed in sample A:
        i. Randomly pick a company *x* and remove from *X*.
        ii. Gather all objects one link away from *x*.
        iii. If any of these objects is in sample B, discard *x*. Otherwise place *x* in sample A, along with all objects one link away from *x*.
    b. Do until a company is placed in sample B:
        i. Randomly pick a company *y* and remove from *X*.
        ii. Gather all objects one link away from *y*.
        iii. If any of these objects is in sample A, discard *y*. Otherwise place *y* in sample B, along with all objects one link away from *y*.
2. For all discarded companies, randomly place half in sample A and half in sample B.
3. Label all companies in sample A that have no links to sample B as objects in the core of sample A. Label sample B similarly.

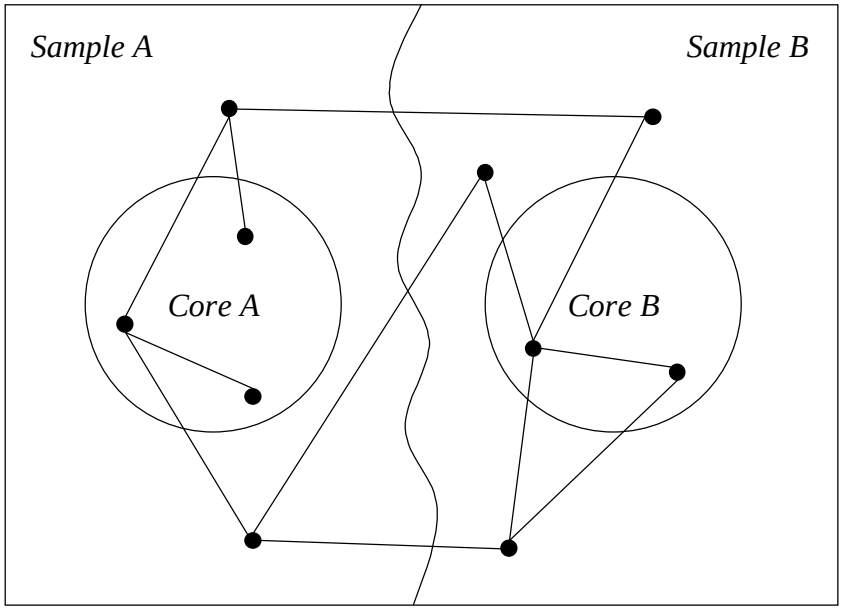

Figure 4: Graphical representation of indirect company linkage in samples A and B

This approach produces two disjoint subsets — the core of each sample. By definition companies in core A have no links to companies in sample B. Likewise, companies in core B have no links to companies in sample A (see figure 4). The resulting size of the

cores depends on the degree of linkage in the dataset. If the objects are highly linked then there will be very few objects in the core.

Because the success of iterative classification in the corporate data depends on linkage among companies, we removed all companies from the sample with no links to other companies. This improved the statistical power of our evaluation by focusing on the portion of the task to which iterative classification is most applicable. It also reduced the total number of companies in the dataset to 1088. In order to increase the number of companies in the core of each sample, the definition of the core was relaxed. Because the only dynamic attributes used for classification involved links through people (insider owners or board members), the core objects were defined as those that have no links *through people* to companies in the other sample. Links to companies in the other sample through corporate owners and contractors however, were allowed. Core A therefore consists of those companies in sample A that have no links through people, to companies in sample B. The distribution of banks and chemical companies in both the samples and the cores are outlined in table 1.

	Number of banks	Number of chemicals	Total number of companies
Sample A	230	316	546
Core A	170	113	283
Sample B	236	306	542
Core B	189	113	302

Table 1: Distribution of samples and cores

5.3. Experimental Procedure

Using the two samples A and B we performed a two-fold cross validation test of iterative classification. The small number of objects in the resulting cores, when sampled for more than two sets, prohibited the use of more than two disjoint samples. The SBC classifier was trained on a fully labeled sample A and then tested on sample B with 10 iterations. Because the 10^{th} iteration has only 90% of the inferences available for dynamic attribute calculation, a final classification pass (11^{th} iteration) was also included which used 100% of the inferred class labels.

During training, the dynamic attributes of sample A make use of some of the class labels in sample B but this does not include any of the companies in core B. When testing on sample B, the classifier makes inferences about all the companies in sample B; however, accuracy is measured only on the fully disjoint companies in core B. The companies of sample A must be fully labeled during the testing process in order to prevent biasing the attribute calculation of companies in sample B that are not in core B. In the second test, the classifier is trained on sample B and tested on sample A.

5.4. Results

Accuracy results for the two test sets are shown in the table below; accuracy refers to the rate of correct predictions made by the model for the objects in the test set. The "Static" accuracy results are from a single classification pass using only static attributes of the test set, where the values for the dynamic relational attributes are all missing. "Iteration 1" and "Iteration 10" are the accuracy results after the first and tenth iteration respectively. "Full knowledge" indicates the accuracy results of a single classification pass using all attributes, where the dynamic attributes are calculated with complete knowledge of the true class labels of all related companies.

	% Accuracy on Core B	% Accuracy on Core A
Static	69.2	68.6
Iteration 1	72.2	78.1
Iteration 10	75.2	80.9
Full Knowledge	78.1	80.9

Table 2: Classification accuracies

McNemar's test (Sachs 1982) was used the compare the difference in classification accuracy between the 1st iteration and 10th iteration. The McNemar statistic tests the null hypothesis that the differences in frequencies of correct and incorrect classifications in each iteration represent random variations in the class labels. Let b be the number of instances that change from correct to incorrect classification from the 1st to the 10th iteration. Let c be the number of instances that change from incorrect to correct classification from the 1st to the 10th iteration. For $(b + c) \geq 30$ the McNemar statistic is $(b - c)^2 / (b + c + 1)$ and it is distributed as χ^2 with one degree of freedom.

Combining the results from both cross-validation trials, the value of the McNemar statistic was 5.558, which indicates the difference in classifications from the 1st to the 10th iteration is significant at the 2% level.

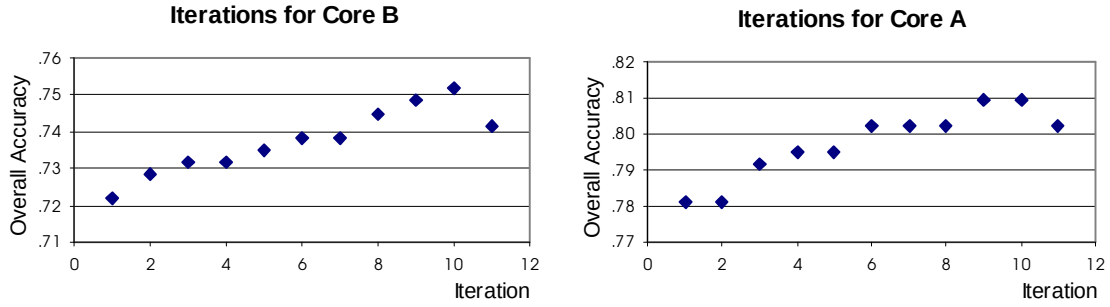

Figure 5: Accuracy results on sample core objects for each iteration.

Accuracy results over the course of iterations for each cross-validation run are shown in figure 5. Accuracy increases steadily throughout the classification procedure except for a drop in the final pass (11th iteration). Dynamic attribute calculations in the final pass include the inferences for which the SBC model is most unsure about — the bottom 10%. This suggests that an improvement in classification could be achieved by the use of a threshold for accepting predictions, instead of accepting the top percentage.

ROC Curve Analysis

Because accuracy maximization assumes equal misclassification cost for false positive and false negative errors, it has been shown that the use of classification accuracy as a primary metric to compare classifiers is not always an indication of superior performance for other costs and class distributions (Provost et al. 1998). Receiver Operating Characteristic (ROC) analysis, an analysis method taken from signal detection theory, is an alternative means to evaluate the error tradeoffs associated with a given model.

ROC curves show the predictive ability of a classifier across all possible error costs and class distributions. Each SBC model is represented in ROC space by a curve corresponding to its true positive rates and false positive rates (TP, FP), as the probability threshold between classes is varied between zero and one. In contrast to the SBC model, which associates probabilities with each class label, for ROC curve analysis one class label is considered positive (in this case bank) and the probabilities associated with the predictions of that class label are used to graph the ROC curves.

An ROC curve maps a classifier's performance as the confidence threshold for acceptance of its predictions is varied between the extremes of accepting no classifications to accepting all classifications. If a model dominates the ROC space it can be regarded as the "best" predictive model for all domains, no matter what the cost and class distributions are in the test environment.

The ROC curves for each cross-validation run are shown in figure 6.

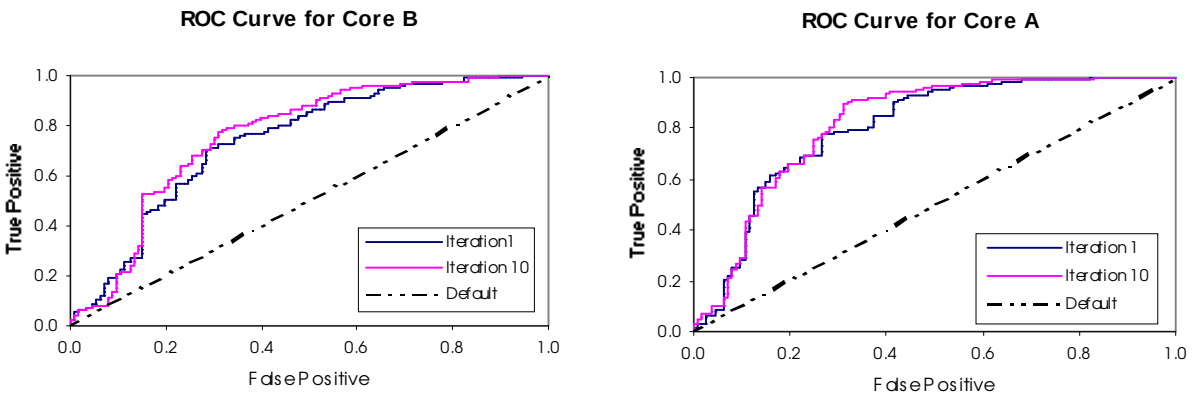

Figure 6: ROC Curves for classification on sample core objects.

5.5. Discussion

Looking at the accuracy results we can make some interesting observations regarding iterative classification in this domain. First, the window for improvement in this dataset is quite small, with approximately a 10% difference between the floor and ceiling accuracies. The floor accuracy can be lowered artificially by dropping static attributes. This was attempted but the iterative approach failed without the inclusion of both static attributes. This indicates the importance of having strong static attributes as islands of certainty from which to jumpstart the iterative process. The limited variety of links in the data set constrained the number of potentially predictive dynamic attributes, so raising the ceiling accuracy was difficult.

Next, the improvement of accuracy in the 1st iteration compared to the static approach is noteworthy. The difference between classification in the 1st iteration and the static test is that during the 1st iteration some dynamic attributes values are known. For companies with less than two links to other companies through people, we can return a value of false for the dynamic attributes without any knowledge of the company type. This suggests that dynamic attributes whose value can be determined with certainty from a small amount of evidence may be quite helpful to the iterative process.

Also, it is worth mentioning that in the second trial on Core A, iterative classification was able to match the accuracy of classification with full knowledge. This shows the power of iterative classification to classify as if it had full knowledge of the surrounding environment.

Finally, the ROC curves show that the 10th iteration performs better than, or equal to, the 1st iteration for most thresholds. However, the ROC curves show that the primary effect of iteration occurs late in the curve when the probability of a company being a bank is relatively low. This may indicate that dynamic attributes are more helpful in the case of predicting chemical companies and do little to increase the probabilities associated with predictions of banks.

6. Conclusions and Future Work

A number of conclusions can be drawn from this work about the potential of iterative classification. We have shown that there is an opportunity to use relations in data to increase classification accuracy, and that an iterative approach exploiting this opportunity can produce a significant improvement in accuracy for a binary classification task in the corporate data set.

We have outlined several necessary conditions for successful application of iterative classification. For iterative classification to improve on a static approach, a data set should exhibit the following characteristics: insufficient predictive power from static attributes and useful dynamic attributes, rich relational structure, and islands of certain knowledge from which to jump start the iterative process. Expansion and formal verification of these ideas is an important area for further investigation.

In addition to presenting opportunities for discovery, relational data also offer several challenges. Devising a sampling procedure that doesn't bias statistical estimates of relational attributes is a difficult task. As the relational data structure becomes more complex, our opportunities for improving classification increase, but so do the challenges of sampling. Future work would be aided by the use of naturally disjoint datasets with similar distributions such as the university web sites used by Slattery (2000).

Formulating helpful dynamic attributes is also challenging. It is difficult to define the value of a dynamic attribute when some, but not all of the related class labels have been inferred. Because the classifier is trained on full knowledge, dynamic attribute values expressing partial knowledge can bias or mislead the predictions of the classifier. A few incorrect inferences could have a "snowball effect" with the dynamic attributes cascading the mistakes throughout the test set. For this reason it is important to use dynamic attributes whose values are either known with complete certainty or not at all. *Threshold* attributes are a good example of this type of "robust" attribute, where the value is known as soon as a particular value threshold is exceeded. Both dynamic attributes used in this experiment are examples of threshold attributes.

Consider the attribute "company *X* is linked to more than one chemical company through its board members". The calculation for this attribute is as follows (where n is the total number of companies linked through board members to *X*, n_b is the number of known banks, n_c is the number of known chemical companies, and '?' designates a missing value):

if $n \leq 1$ then return false
else if $n_c > 1$ then return true
else if $n - n_b \leq 1$ then return false
else return '?'

When designing threshold attributes it is important to keep in mind that different sources of evidence can be used to determine the value of the attribute but some of the values must be known with certainty early on in the process for iteration to have a starting point. If none of the values can be determined in early iterations then either the process will stall and no gains will be made, or incorrect predictions will begin to reduce classification accuracy. Future work includes both establishing the effects of threshold attributes on iterative classification, and determining other types of robust attributes.

Attributes that combine probabilistic evidence of all related class labels are a potential alternative to threshold attributes. Instead of accepting the top percentage of predictions, or those exceeding a threshold, the algorithm would accept all predictions. The values of these probabilistic attributes are then determined by a combination of the probabilities associated with the inferred class labels of related objects. As the certainty of predictions change over the course of iterations, the attribute values could be dynamically updated. This is an area that requires additional exploration; it is not clear that the probabilities produced by the SBC are accurate enough to be used effectively in this fashion.

A potential pitfall of the specific variety of iterative classification explored here is that SBCs often produce biased probability estimates. SBCs are known to produce optimal class predictions in a wide variety of domains; however, SBC probability estimates are biased except under conditions of attribute independence. Future work includes exploring iterative classification with other methods that output more accurate probabilities such as Bayesian networks or PRMs (Freidman et al. 1999). We will also investigate the use of a threshold for accepting predictions instead of accepting a percentage determined by the number of iterations.

Another direction for future work involves extending the iterative procedure for prediction of multiple object types by simply combining the results of multiple classifiers. Each classifier would make use of the dynamic attributes filled in through the efforts of the other classifiers. In this sense the classifiers would collaborate with each other to improve accuracies for both classification tasks. Caruana (1997) has investigated the collaboration of multiple models for learning under the hypothesis that multiple, related learning tasks share the same representation, and learning one helps with learning another. A relational approach would be similar but would involve the collaborative application of models instead.

7. Related Work

Previous work of the WebKB project investigated classification in a relational context (Craven et al. 1998). WebKB uses FOIL, a greedy covering algorithm for learning function-free Horn clauses, to label web pages automatically. Relationships among pages, as encoded by their hyperlinks, are used along with intrinsic attributes to improve classification accuracy. While this approach uses relational attributes as inputs to the learned model, the values of those attributes remain static throughout the entire process. What is known about the test instances does not change dynamically during classification as it does in an iterative approach.

Freidman et al. (1998) have also investigated the use of a relational framework to make complicated inferences. They have shown how to learn probabilistic relational models (PRMs) from relational databases. PRMs are sophisticated relational models, similar to Bayesian networks (Heckerman 1995) that allow the properties of an object to depend probabilistically on both properties of the object in isolation and on properties of other related objects. However, as with WebKB, the knowledge in these models remains fixed; the data representation is not updated to reflect the inferences made by the model. This approach does not exploit the relationships among objects as fully as iterative classification.

The Expectation-Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977) is similar to in spirit to iterative classification, but it addresses a somewhat different problem. The EM algorithm uses a two-step iterative procedure to find the maximum-likelihood estimate of the parameters of an underlying distribution (a model) from a data set containing incomplete or missing data (Bilmes 1998). The first step of EM (the "expectation" step) finds the expected value of missing data values, given the current

model. The second step of EM (the "maximization" step) finds the maximum-likelihood model, given the inferred data. After replacing the current model with the new model, the process repeats. In contrast to iterative classification, EM readjusts the model in the second step, rather than adjusting the values of attributes that serve as inputs to the model. Thus, it is a method of learning a model given attribute-value data, rather than a method of applying a learned model to relational data.

"Co-training" is another iterative approach to learning models (Blum and Mitchell 1998, Mitchell 1999). Mitchell showed that a large number of unlabeled instances can be used to boost the performance of a learning algorithm when only a small set of labeled instances is available. Multiple classifiers are learned on independent sets of attributes, from the all the available training data. Each classifier is run and its most confidently predicted positive and negative instances are added to the common training set. By using the same training data, the classifiers each profit from the predictions of other classifiers. The classifiers are relearned with the larger, augmented training set, and the process is repeated. Co-training is tested in a relational context; however, it does not require relational knowledge for the process to operate, it can be applied to attribute-value data as well. As with the EM algorithm, this method uses iteration for learning models instead of using iteration in the application of learned models, as does iterative classification.

Boosting and other methods of ensemble classification (Dietterich 1997) are related to iterative classification in the sense that what is known about the data changes over the course of the procedure. Boosting uses multiple classifiers collaboratively for a single classification task and manipulates the sampling distributions of the training sets used to learn each model in order to increase overall classification accuracy. Ensemble approaches such as these, change the knowledge available to models as they are learned. In contrast, iterative classification changes the knowledge available to the model as it is applied.

Kleinberg (1998) developed an iterative algorithm, called Hubs & Authorities, for Web searching based on the network structure of hyperlinked pages on the Web. The algorithm uses a graph structure, with nodes corresponding to web pages and directed links indicating the presence of hyperlinks between pages. Given the task of identifying authoritative pages, two mutually-reinforcing attributes are defined: hub weight and authority weight. The weights are calculated in an iterative fashion by feeding the values of one attribute into the calculations of the other. The iterative nature of this algorithm is similar to our approach in that it maintains and updates attribute values throughout the procedure. However, the algorithm assumes the values of both attributes are known for each instance and starts by assigning equal weights to all pages, it does not use a predictive model to assign weight values.

In work concurrent with our own, Slattery (2000) has investigated using relational information in the test set to classify web pages more accurately. FOIL-HUBS is an extension of FOIL inspired by the Hubs & Authorities algorithm (Kleinberg 1998). FOIL-HUBS identifies the existence of hubs for each target class (e.g., student-hubs point to many student pages) and hub weights contribute to the probability that pages

pointed to by the hubs are of a particular class. FOIL-HUBS employs an iterative classification scheme to predict class labels and estimate hub weights, which is very similar to our own algorithm for iterative classification, but it is limited to domains where uniform hub nodes exist. In contrast, our work represents an initial attempt to provide a uniform framework for the calculation and use of a wider range of dynamic attributes, albeit within a simpler model representation (SBCs as opposed to function-free Horn clauses).

8. Acknowledgments

The author would like to thank the David Jensen, Matt Cornell, and Hannah Blau for their important contributions to the work reported herein. Thanks also go to Foster Provost and James Allan for valuable comments and suggestions on drafts of this work.

This research is supported, in part, by a University of Massachusetts Faculty Research Grant and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Office of Scientific Research (AFOSR) under contract F49620-97-1-0485. 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 the University of Massachusetts, DARPA, AFOSR, or the U.S. Government.

9. References

Bilmes, J. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. TR-97-021. International Computer Science Institute, Berkeley, California.

Blum, A. and T. Mitchell. (1998). Combining labeled and unlabeled data with cotraining. In Proceedings of the 11th Annual Conference on Computational Learning Theory. ACM.

Brodley, C. E., and P. Smyth (1997). Applying classification algorithms in practice. Statistics and Computing 7: 45-56.

Caruana, R. (1997). Multitask learning. Machine Learning 28: 41-75.

Chakrabarti, S., B. Dom, S. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, J. Kleinberg, and D. Gibson (1999). Hypersearching the Web. Scientific American. June. 54-60.

Craven M., D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery (1998). Learning to extract symbolic knowledge from the World Wide Web. Proceedings of the 15th National Conference on Artificial Intelligence.

Dempster, A., N. Laird, and D. Rubin (1977). Maximum-likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B. 39.

Dietterich, T. G. (1997). Machine learning research: Four current directions. AI Magazine 18(4):97-136.

Domingos, P., and M. Pazzani (1997). On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning 29:103-130.

Duda, R., and P. Hart (1973). Pattern classification and scene analysis. New York, NY: Wiley.

Fawcett, T., and F. Provost (1997). Adaptive Fraud Detection. Data Mining and Knowledge Discovery 1:291—316.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). From data mining to knowledge discovery in databases. AI Magazine. Fall: 37-54.

Friedman N., L. Getoor, D. Koller, and A. Pfeffer (1999). Learning probabilistic relational models. Proceedings of the International Joint Conference on Artificial Intelligence. pp. 1300-1307.

Jensen, D. (1997). Prospective assessment of AI technologies for fraud detection: A case study. AAAI Workshop on AI Approaches to Fraud Detection and Risk Management.

Jensen, D. (1998). Statistical challenges to inductive inference in linked data. Seventh International Workshop on Artificial Intelligence and Statistics.

John G., and P. Langley (1995). Estimating Continuous Distributions in Bayesian Classifiers. Proceeding of the Eleventh Conference on Uncertainty on Artificial Intelligence.

Heckerman, D. (1995). A tutorial on learning with Bayesian networks. Microsoft Research Technical Report MSR-TR-95-06.

Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms.

Mitchell, T. (1997). Machine Learning, New York, NY: McGraw Hill.

Mitchell, T. (1999). The role of unlabeled data in supervised learning. In Proceedings of the 6th International Colloquium on Cognitive Science.

Palace, B. (1996). Data mining: Technology note prepared for management 274A, Anderson Graduate School of Management UCLA. http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/index.htm. June.

Pregibon, D. (2000). Personal communication. AT&T Laboratories. February 11.

Provost, F., and T. Fawcett, and R. Kohavi (1998). The case against accuracy estimation for comparing induction algorithms. Proceedings of the Fifteenth International Conference on Machine Learning.

Sachs, L. (1982). Applied statistics: A handbook of techniques. New York, NY: Springer-Verlag.

Silverman, B.W. (1986). Density Estimation. London: Chapman and Hall.

Slattery, S. (2000). Unsupervised structural inference for web page classification. To appear in: 17th International Conference on Machine Learning.

Wasserman, S., and K. Faust (1994). Social network analysis: Methods & applications. Cambridge, UK: Cambridge University Press.

White, A. and W. Liu (1994). Bias in information-based measures in decision tree induction. Machine Learning, 15(3), 321-329.