KDD Cup 2001 Report

Jie Cheng
Canadian Imperial Bank of Commerce
Global Analytics
Jie.cheng@cibc.ca

Christos Hatzis
Silico Insights, Inc.
Woburn, MA
christos@silicoinsights.com

Hisashi Hayashi
Department of Computer Science
University of Tokyo
hisashi@gi.k.u-tokyo.ac.jp

Mark-A. Krogel
School of Computer Science
Otto von Guericke University
Magdeburg, Germany
krogel@iws.cs.uni-magdeburg.de

Shinichi Morishita
Dept. of Computer Science and Dept. of Complexity Science and Engineering
University of Tokyo
moris@k.u-tokyo.ac.jp

David Page
Dept. of Biostatistics and Medical Informatics and Dept. of Computer Sciences
University of Wisconsin
page@biostat.wisc.edu

Jun Sese
Dept. of Complexity Science and Engineering
University of Tokyo
sesejun@gi.k.u-tokyo.ac.jp

ABSTRACT

This paper presents results and lessons from KDD Cup 2001. KDD Cup 2001 focused on mining biological databases. It involved three cutting-edge tasks related to drug design and genomics.

Keywords

Competition, biology, drug design, genomics

1. INTRODUCTION

Since its inception five years ago, the KDD Cup has become an event of international stature. It is recognized widely not only within the community but among scientists from other fields as well. While much attention surrounds winning the cup, its primary contribution still is to serve as a laboratory from which the community can learn many valuable lessons and identify key areas for further research. Several important lessons can be gleaned from KDD Cup 2001. They include the following, at least the first two of which will be surprising to many readers.

- A Bayes net learner can outperform the leading classification algorithms on a pure classification task.
- A feature-based learner can outperform relational learners on a multi-relational task, if the features are constructed by a clever "propositionalization" algorithm.
- More research is needed on comprehensibility, human-computer interaction, and performance evaluation.
- While interest is booming in mining data on networks of interacting genes or proteins, interaction so far has not proven to be the most useful piece of information for predicting important protein or gene properties such as function and localization.

This paper reviews KDD Cup 2001, describes the winning approaches, and concludes with a discussion of the aforementioned lessons. Other more specific lessons also can be found at the conclusion of each section about the winning approach to a task.

Because of the rapid growth of interest in mining biological databases, KDD Cup 2001 was focused on data from genomics and drug design. Sufficient information was provided so that detailed domain knowledge was not a requirement for entry. A total of 136 groups participated to produce a total of 200 submitted predictions over the 3 tasks: 114 for Thrombin, 41 for Function, and 45 for Localization. The remainder of this section discusses these three tasks and statistics of the entries. The subsequent three sections describe the winning approaches to the three tasks and are written by the winners themselves.

KDD Cup 2001 involved three tasks, based on two data sets, one of which was a little over half a gigabyte in size and the other a little over seven megabytes. While the first consisted of a single relational table, or flat file, the second consisted of two relational tables that were produced from an original seven relational tables. A single flat file also was provided with the second data set, but users were warned that they might get better results by staying with the multiple relational tables or by doing their own conversion. Further details on the tasks along with the complete datasets and answer keys and also copies of the presentations

given by the cup organizers and the winning teams can be found at the KDD Cup 2001 web site (http://www.cs.wisc.edu/ ~dpage/ kddcup2001/).

1.1 Dataset 1: Prediction of Molecular Bioactivity for Drug Design – Binding to Thrombin

Drugs are typically small organic molecules that achieve their desired activity by binding to a target site on a receptor. The first step in the discovery of a new drug is usually to identify and isolate the receptor to which it should bind, followed by testing many small molecules for their ability to bind to the target site. This leaves researchers with the task of determining what separates the active (binding) compounds from the inactive (non-binding) ones. Such a determination can then be used in the design of new compounds that not only bind, but also have all the other properties required for a drug (solubility, oral absorption, lack of side effects, appropriate duration of action, toxicity, etc.).

The thrombin training data set consisted of 1909 compounds tested for their ability to bind to a target site on thrombin, a key receptor in blood clotting.[1] Of these compounds, 42 are active (bind well) and the others are inactive. Each compound was described by a single feature vector comprised of a class value (A for active, I for inactive) and 139,351 binary features, which describe three-dimensional properties of the molecule. The definitions of the individual bits were not included -entrants were told only that they were generated in an internally consistent manner for all 1909 compounds. Biological activity in general, and receptor binding affinity in particular, correlate with various structural and physical properties of small organic molecules. The task was to determine which of these properties are critical in this case and to learn to accurately predict the class value.

This task is exemplary of a large family of related tasks arising within both the pharmaceutical industry and university research laboratories. In fact, in some cases the number of feature-to-example ratio can be even greater than here. To further simulate the real-world drug design environment, the test set contained 636 additional compounds that were in fact generated based on the assay results recorded for the training set. The goal was to see who (if anyone) could outperform the chemists who decided on this new set of 636 compounds to make and test. If a data mining system could perform better, it could be used to help select the next round of compounds to synthesize and test for similar applications. Because more compounds prove to be inactive than active, a predictor that always says “inactive” will achieve a relatively high accuracy even though it is completely useless. Therefore, in evaluating the accuracy for this task only, a differential cost model was used. The simplest description of this model that we have found is as follows. We compute the accuracy on true actives and the accuracy on true inactives and compute the (unweighted) average these two. Hence it was just as important for entrants to minimize their error rate on the actives as to minimize their error rate on the inactives,

This last point regarding evaluation is worth a bit more discussion. Ideally, we would compare approaches using ROC curves (comparing areas under the curves). But this was infeasible because participants submit their predictions, not their predictors; furthermore, some predictors are not easily modified to be more or less conservative, as is required to generate ROC curves. John Elder (private communication) proposed using ``lift." Perhaps this should be considered for some future KDD Cup tasks, but it requires participants to rank-order their predictions (most likely active, second-most likely active, etc.), which some predictors are not easily modified to do.

1.2 Dataset 2: Prediction of Gene/Protein Function and Localization

The genomes of several organisms have now been sequenced, including the human genome. Interest within bioinformatics therefore has shifted away from classical genomics mainly concerned with sequence assembly and gene identification, to functional genomics, i.e. learning about the genes encoded in the sequence. Genes code for proteins, and these proteins tend to localize in various parts of cells and interact with one another, in order to perform crucial functions. Although genes that encode for 6449 yeast proteins are already known, only 52% of these proteins have been characterized. Of the remaining, only 4% show a strong similarity at the sequence level that can form a basis for inferring function. It therefore becomes imperative to use information beyond sequence similarity to characterize the unknown genes. Data set 2 consisted of a variety of gene-level and protein-level information from the yeast genome.[2] Gene names were anonymized and a subset of the genes (about 1/3) were withheld for testing.

Two tasks were associated with data set 2. These two tasks were to predict the functions and localizations of the proteins encoded by the genes. The functions actually are fifteen broad functional categories, while the locations are fifteen different parts of the cell. A gene/protein can have more than one function, but rarely (in this data set) more than one localization. The other information provided about the genome is described in the following two paragraphs.

One relational table specified which genes (or their protein products) interact with which other genes. An interaction may be physical, in which the protein for which one gene codes is known to bind in some way with the protein for which another gene codes. Or an interaction may be genetic, in which the presence or absence of a protein affects the level of expression of the gene

[1] We thank DuPont Pharmaceuticals Research Laboratories for graciously providing this data set for the KDD Cup 2001 competition. All publications referring to analysis of this data set should acknowledge DuPont Pharmaceuticals Research Laboratories and KDD Cup 2001.

[2] We thank the team at MIPS (Munich Information Center for Protein Sequences) for making the protein interaction data available (http://mips.gsf.de/proj/yeast/CYGD/db/index.html)

encoding for another. Or an interaction may be both physical and genetic. This relational table also presents the degree of correlation between the expression levels of the two genes under a given experimental condition, as measured by gene expression microarrays.[3]

The other relational table specified a variety of properties of individual genes or proteins. These included the chromosome on which the gene appears, whether organisms with a mutation in this gene can survive, phenotype (observable characteristics) of organisms with differences in this gene, structural category of the protein for which this gene codes (what general shape does the protein take), the existence of characteristic motifs in the amino acid sequence of the protein, and whether the expression of this gene complexes with others to form a larger protein. It further would have been desirable to use sequence information about each gene.

But in contrast to the Thrombin dataset, yeast genome data already was in the public domain. Had we provided the sequence, it would have been too easy for anyone to identify each gene in the test set and "look up" its functions and localization. Indeed, the cup co-chairs considered even anonymizing all the field names and values, to completely ensure the answers could not be found in this way. But in the end the co-chairs decided this decision would take too much away from the interestingness of the data set and tasks.

An additional challenge with using protein interactions as a predictor of function is the fact that high-throughput approaches, such as the yeast two-hybrid system, that are used to screen physical interactions can generate many false-positive relations.

For example, a protein might have the right sequence complementarity to interact with another protein, but the second protein is localized in a different compartment in the cell and therefore cannot interact under physiological conditions.

A final challenge was that, because both function and localization had to be withheld in the test set, predictors for function could not use localization (even though it was present in the training set) and predictors for localization could not use function. In actual practice, it is common to have localization information available when one is trying to predict function.

Evaluation of localization was the simplest of the three tasks. Each entry was permitted to predict only one localization per gene. If more than one localization was predicted, the first prediction was used. The score of an entry was simply the fraction of genes for which the correct localization was predicted. Evaluation of function was slightly more complicated because a gene could have more than one function. An entry could contain arbitrarily many function predictions per gene. Our key consisted of all

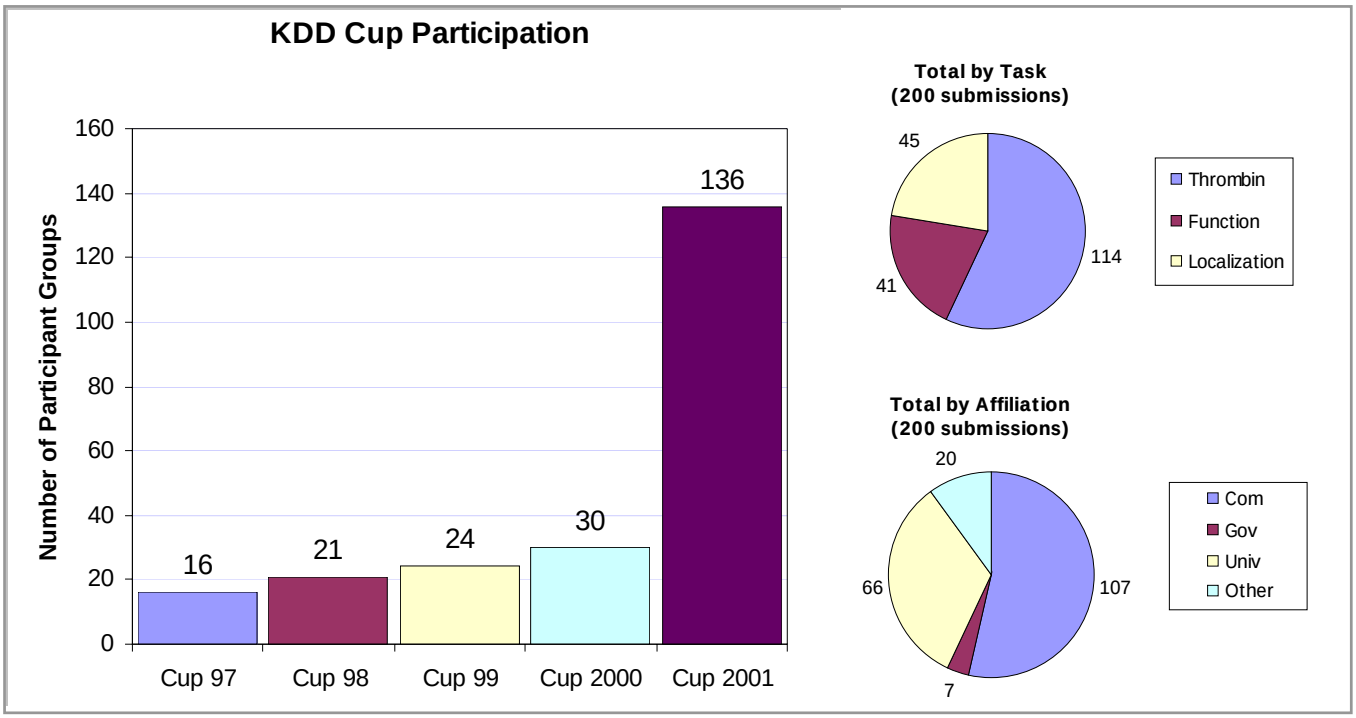


Figure 1. Number of participant groups and distribution of submissions by task and group affiliation.

[3] The gene expression profiles were from the diauxic study growth of DeRisi et al. (1997) and are available at the Stanford Microarray Database (http://genome-www5.stanford.edu/MicroArray/SMD/)

(gene,function) pairs where the gene was known to have the function. A (gene,function) pair was counted correct for an entry if it either (1) appeared in both our key and the prediction or (2) appeared in neither our key nor the prediction. Otherwise it was

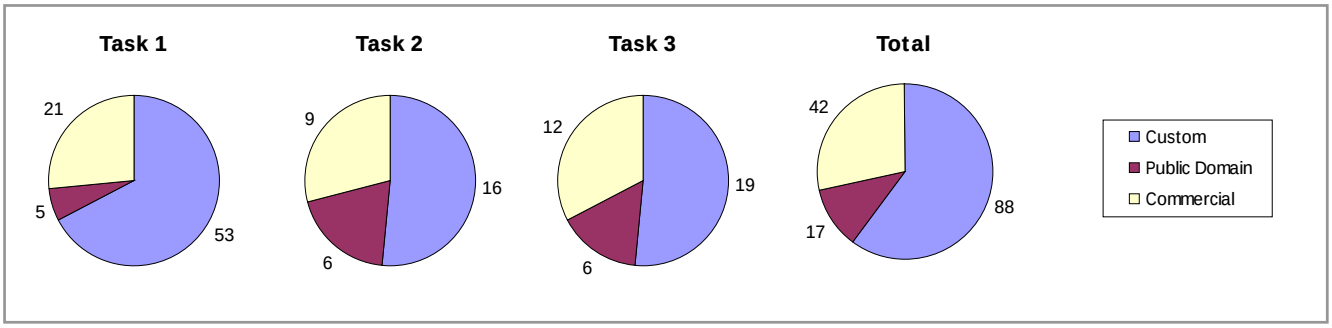


Figure 2. Origin of software used by the participants.

counted incorrect. The score of an entry was the number of correct pairs over the total number of (gene,function) pairs.

1.3 Submission Statistics

As shown in Fig. 1, a total of 136 groups participated to produce a total of 200 submitted predictions over the 3 tasks: 114 for Thrombin, 41 for Function, and 45 for Localization. This represents a five-fold increase in participation over previous years with more than half of the entries being contributions from the commercial sector. We believe that the increased level of participation could very likely represent a growing interest by the data mining community in bioinformatics-related knowledge discovery problems. This is supported by the workshop and keynote talk on the same subject in this year's conference.[4]

We asked the participants to send a paragraph describing their approach in exchange of an email report of their performance along with the performance of everyone else. We received 156 responses and these provided the basis for the summary report on the approach and software used for the data mining tasks. The breakdown by origin of data mining software used by the participants in the three tasks is shown in Figure 2. About 60% of the competitors used custom software, i.e. software that they wrote. It is interesting to note that, especially for the Thrombin task, only 5 out of 79 groups (6%) used commercial or proprietary software. This reverses the trend observed over the previous competitions, where 77% of the participants in the KDD-Cup 2000 used commercial software.[5] The use of custom software in KDD-Cup 2001 was dictated by the nature of the problems, as most commercial software systems cannot handle the excessive number of features in the Thrombin dataset and the multi-class nature of the Function prediction task. In contrast, for Localization prediction, which lends itself better as a standard single-class classification task, 32% of the competitors used commercial data mining packages. These observations on software usage seems to indicate that although many mature commercial data mining packages are available, data from the bioinformatics-related domains pose unique challenges that commercial packages cannot address adequately.

As mentioned, both the relational tables and a denormalized flat file were made available for the Function and Localization tasks. Of the 57 competitors who provided information on the dataset used, 33 used the fully-denormalized set, 21 used both the genes and interaction relational tables, and 3 used only the genes table. Most likely, this was due to lack of access to data transformation tools that can deal easily with relational data.

Figure 3 summarizes the algorithmic approaches used in the data mining tasks by the competitors. Not surprisingly, almost 70% of the competitors in the Thrombin task used feature selection in combination with some other algorithm, but only a very small number of them created new features for that dataset. However, about 15% of the competitors in the Function and Localization tasks constructed new features mainly to capture the relational nature of the data. Decision trees and ensemble classifiers based on more than one algorithms were by far the most popular for all three tasks, followed by Naïve Bayes and k-nearest neighbor classifiers. The k-nearest neighbor classifiers offered certain advantages in dealing with the significant fraction of missing data in Dataset 2. Surprisingly, only two groups used relational approaches, such as inductive logic programming (ILP), despite the fact that such techniques can deal "naturally" with relational data. Finally, a relatively small number of competitors used cross-validation techniques in an iterative fashion to improve algorithm selection and tuning.

[4] KDD-2001 Workshop: "Data Mining in Bioinformatics" organized by M.J. Zaki, H.T.T. Tiovonen and J.T.L. Wang. Keynote talk: "Challenges for Knowledge Discovery in Biology" by Russ Altman.

[5] Kohavi *et al.* (2001) KDD-Cup 2000 Organizers' Report: Peeling the Onion. *SIGKDD Explorations* 2(2):86-98.

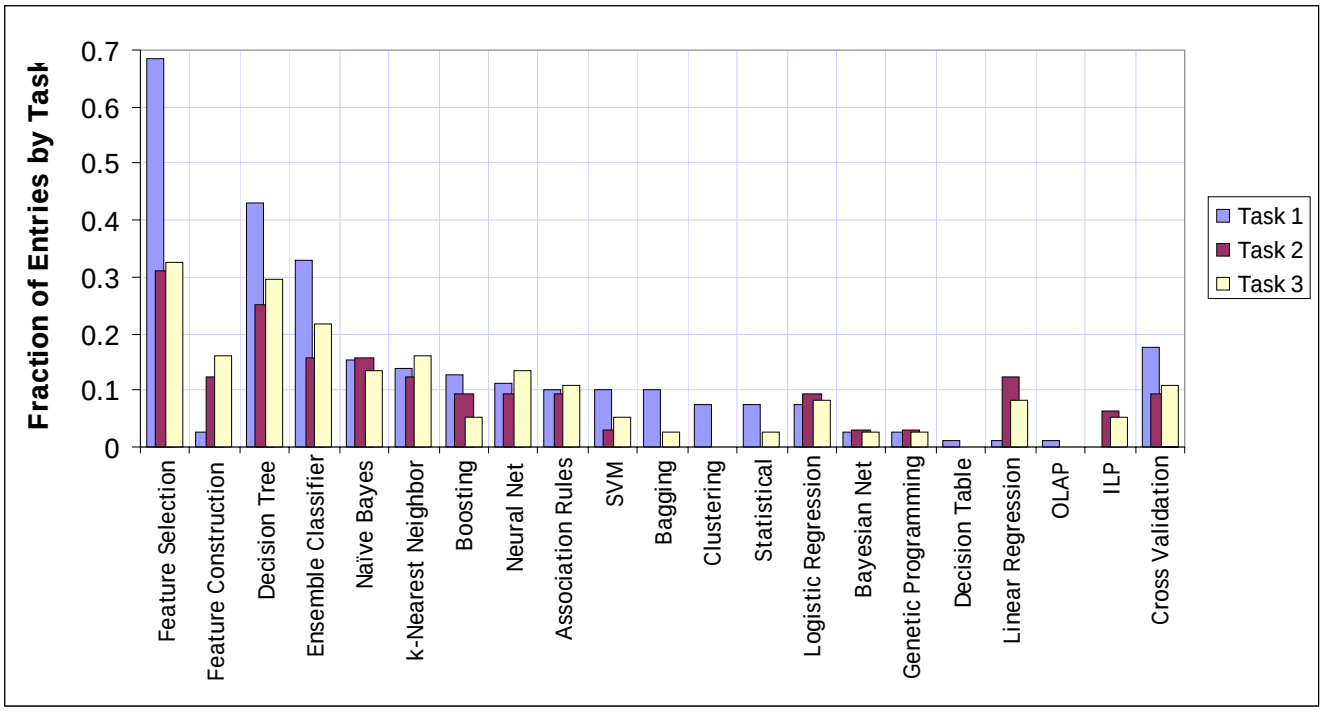


Figure 3. Algorithmic approaches used in the three tasks.

The remainder of the paper is organized as follows. Section 2 is discusses the winning approach to Task 1: Thrombin and is written by Jie Cheng. Section 3 discusses the winning approach to Task 2: Function and is written by Mark Krogel. Section 4 discusses the winning approach to Task 3: Localization and is written by Hisashi Hayashi, Shinichi Morishita, and Jun Sese. Section 5 discusses broad lessons from the competition; Sections 1 and 5 are written by the co-chairs Christos Hatzis and David Page. Before continuing, we would like to provide honorable mention to several teams that either came very close to the winning score for a task or provided particularly important insights. These are:

- Task 1: Tomi Silander, University of Helsinki.
- Task 2: Christophe Lambert, Golden Helix; Jun Sese, Hisashi Hayashi, and Shinichi Morishita, University of Tokyo; David Vogel and Ramanujan Srinivasan, A.I. Insights; Sara Pocinki, Robert Wilkinson, and Patrick Gaffney, Lubrizol.
- Task 3: Matthias Schonlau, William DuMouchel, Chris Volinsky, and Corina Cortes, RAND and AT&T; Brian Frasca, Zijian Zheng, Rajesh Parekh, and Ronny Kohavi, Blue Martini.

2. THROMBIN RESULT

2.1 Overview

The aim of this task is to learn a classifier that can effectively predict whether an organic molecule can bind well to a target site on thrombin -- a key receptor in blood clotting, given the chemical structure of the compound.

From the description of this task in Section 1.2, we can see that this is a highly challenging problem for three reasons. First, the training data set is extremely imbalanced and the number of positive examples is very small – only 42 compounds (2.2 percent of the total 1909) are active. This prevents us from using standard model evaluation and selection techniques such as creating separate validation set or using k-fold cross validation. Second, the feature vector contains 139,351 features, which is a lot more than most learning systems can handle. Therefore, having the proper techniques for effective dimension reduction is crucial here. Third, because the examples in the test set are not drawn from the same distribution as the training set, the distribution of test set is expected to be very different from that of the training set. This means that we do not have a properly defined misclassification cost function. As a result, finding the cut-point to separate the active compounds from the inactive compounds using their posterior probabilities can be quite tricky.

Because we have a lot of experience in learning Bayesian network classifiers and our previous work showed that Bayesian networks can be excellent classifiers and have unique advantage in feature selection [4][5], we decided to apply Bayesian network learning techniques to this task.

2.2 Introduction to Bayesian Network

A Bayesian network $B = \langle N, A, \Theta \rangle$ is a directed acyclic graph (DAG) $\langle N, A \rangle$ where each node $n \in N$ represents a domain variable (eg, a dataset attribute), and each arc $a \in A$ between nodes represents a probabilistic dependency, quantified using a conditional probability distribution (CP table) $\theta_i \in \Theta$ for each node n_i (see [16,17]). A BN can be used to compute the conditional probability of one node, given values assigned to the other nodes; hence, a BN can be used as a classifier that gives the *posterior probability distribution* of the class node given the values of other attributes. A major advantage of BNs over many other types of predictive models, such as neural networks, is that the Bayesian network structure represents the inter-relationships among the dataset attributes (Fig. 4). Human experts can easily understand the network structures and if necessary modify them to obtain better predictive models.

We will later use the idea of a *Markov boundary* of a node *y* in a BN, where *y*'s Markov boundary is a subset of nodes that "shields" *n* from being affected by any node outside the boundary. One of *y*'s Markov boundaries is its *Markov blanket*, which is the union of *y*'s parents, *y*'s children, and the parents of *y*'s children. When using a BN classifier on complete data, the Markov blanket of the classification node forms a natural feature subset, as all features outside the Markov blanket can be safely deleted from the BN. This can often produce a much smaller BN without compromising the classification accuracy.

Figure 4 shows the structure of a Bayesian net classifier learned from the *Adult* data set, which is extracted from the census bureau database and commonly used as a benchmark dataset for learning models that predict whether a person's salary is over 50K per year. The *Adult* data set contains 12 features and the class attribute (a.k.a. target variable) is "Salary". In Figure 4, we ignored five attributes that are outside the Markov blanket of the target node since those attributes will not affect the outcome of the classification when there is no missing value. Here we can see that feature selection is a natural byproduct of the Bayesian net learning.

Although the arrows in Bayesian network are commonly explained as causal links, in classifier learning, the class attribute is normally placed at the root of the structure in order to reduce the total number of parameters in the CP tables. For convenience, we can imagine that the actual class of a sample 'causes' the values of other attributes. Using the classifier in Figure 4, we achieved one of the best predictive accuracies ever reported on the *Adult* data set [5].

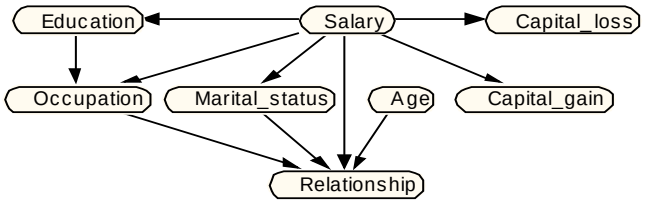


Figure 4. Bayesian network structure of 'Adult' data set.

The two major tasks in learning a BN are: learning the graphical structure, and then learning the parameters (CP table entries) for that structure. As it is trivial to learn the parameters for a given structure that are optimal for a given corpus of complete data – simply use the empirical conditional frequencies from the data – we will focus on learning the BN structure.

There are two ways to view a BN, each suggesting a particular approach to learning. First, a BN is a structure that encodes the joint distribution of the attributes. This suggests that the best BN is the one that best fits the data, and leads to the *scoring-based* learning algorithms, that use heuristic search to seek a structure that maximizes the Bayesian, MDL or Kullback-Leibler (KL) entropy scoring function [7,11].

Second, the BN structure encodes a group of conditional independence relationships among the nodes, according to the concept of *d-separation* [17]. This suggests learning the BN structure by identifying the conditional independence relationships among the nodes. These algorithms are referred as *CI-based* algorithms or constraint-based algorithms [6,19].

We developed a unique constraint-based three-phase dependency analysis algorithm, which is especially suitable for data mining in high dimensional data sets due to its efficiency. (The complexity is roughly O(N^2) where N is the number of features; see [6] for details.) In [4,5], we studied various aspects of learning Bayesian networks as classifiers. The empirical results on a set of standard benchmark datasets show that Bayesian networks are excellent classifiers. We have also developed two Bayesian network learning systems: BN PowerConstructor [2], which is used for general Bayesian network learning, and BN PowerPredictor [3], which is used for classifier learning. PowerPredictor is the tool that we use in this KDD CUP competition.

2.3 Our Approach to Thrombin Data

2.3.1 Data Pre-processing

Because there are only 42 positive examples in the training data, it is quite obvious that the data cannot support a complex model that uses many features. Therefore, it is justified to apply dimension reduction techniques. The first thing we would like to do is feature filtering. The goal is to create a feature subset that is large enough to include all the important features and is small enough for our learning system to handle easily. At this stage, we are not trying to come up with the final feature subset but to exclude the features that are not strongly correlated with the target variable.

To achieve this, we applied the standard information gain feature filtering. We computed the mutual information between each of the 139,351 features and the target variable ("Activity") using Equation 1, and then sorted these features by their information gain from large to small. In Figure 5, we plotted the top 30,000 features. It is easy to see that only a very small portion of the features on the top have strong correlation with the target variable. Based on this plot, we decided to filter out the features that have information gain less than 0.035. This left us with around 200 features.

$$I(A,B) = \sum_{a,b} P(a,b) \log \frac{P(a,b)}{P(a)P(b)} \quad (1)$$

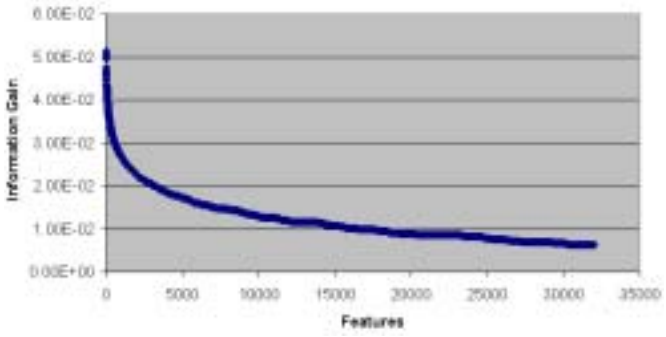


Figure 5. The information gain of the top 30000 features

2.3.2 *Learning and evaluating Bayesian Network models*

The BN PowerPredictor system allows users to control the complexity of the learned network by adjusting a threshold value. It can also use a wrapper to search for an optimal model based on the model's performance on a validation set. The system allows users to choose from two commonly used performance measures: the prediction accuracy and the area under ROC curve (AUC) (see [10]). For this problem, the AUC measure is more appropriate since the cost function is actually unknown.

Since there are only 42 positive examples in the training set, we cannot afford to use a part of the training set as the validation set to search for an optimal model. We cannot even afford to use 10-fold cross validation. Although leave-one-out cross validation might work, it is very expensive computationally and the time we could spend on the KDD Cup was very limited. Therefore, we decided to abandon the wrapper approach. Instead, we decided we would have to create several candidate models and pick the one that seemed most appropriate.

Based on the above analysis, we generated five candidate models from the preprocessed training data set that had 200 features and 1909 instances. Learning each model took about 8 minutes on a Pentium III PC. Although each model had many connections among the 200 features, we could safely ignore most of each network by only studying the sub-network that contained the class node and the features in the *Markov blanket*. Each of the five candidates had from two to twelve features. For each candidate, we used it to classify the whole training set and measured its AUC scores. Then we picked the simplest model that had a "decent" AUC score – a model with only four features. Because the performances were not measured using out-of-sample data, we could not simply pick the one with the highest score. In fact, all of the more complex candidates give slightly better scores. The chosen model is shown in Figure 6.

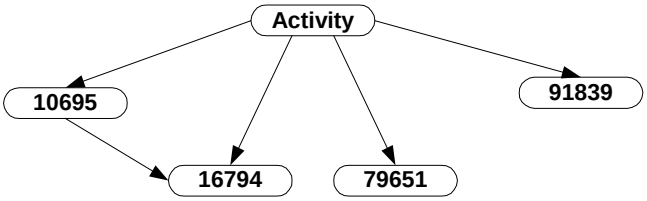


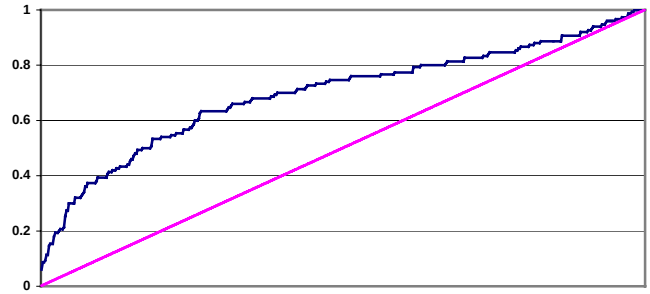
Figure 6. The Bayesian network classifier

2.3.3 *Classifying the testing set*

Using the chosen model, we created the posterior probabilities of each instance in the test data set. The next thing to do was to decide the cut point to classify the test cases into either active or inactive. If the test data were drawn from the same distribution as the training data, and the cost function were known, we could then calculate the optimal cut point from the ROC curve. However, this was not the case. Fortunately, when we examined the probabilities of the test cases, we could see that the model only gave nine distinct probability values due to its simplicity, which meant there were only eight possible cut points to choose from – we could either classify 32 tests cases as active, or 71 cases, or 72, 74, 75, 215, 223, 550. Since we knew that there were still more inactive cases than active cases, we decided to classify 223 cases as active. We did not want to classify a smaller number of cases as active because misclassifying a true active case cost more.

2.3.4 *Analyzing the result*

After the true labels of the test cases were released, we measured the performance of all the candidate models. We were glad to see that the model we chose was actually the best, which means the more complex models were overfitting models. Our final model gave classification accuracy 0.711 and weighted accuracy 0.684. Its ROC curve and confusion matrix are shown in Figure 7.



	Predicted Positive	Predicted Negative
Actual Positive	**95**	**55**
Actual Negative	**128**	**356**

Figure 7. The ROC curve and the confusion matrix of the chosen model

2.4 Discussion

There are three things we have learned from working on the KDDCUP project:

1. The combination of information gain based feature filtering and the Bayesian net based feature selection is a novel, effective approach for analyzing high-dimensional data. If the data had contained only the four features our model used, we believe many classifier learning techniques would have attained similar performance.
2. We gained awareness of the overfitting problem when out-of-sample validation is impossible, especially when the sample size is small.
3. One should carefully choose performance measures that are cost function independent when a well-defined cost function is not available, such as the AUC.

3. PREDICTING FUNCTION

This section presents a description of an approach to KDD-Cup 2001 tasks 2 and 3, prediction of gene/protein function and localization. The approach includes the application of the software system RELAGGS, which was developed at Magdeburg University. Due to this system, we were able to win the Cup for task 2, which forms the focus of this section.

3.1 Motivation

The main objective for our participation in KDD-Cup 2001 was to evaluate our approach to data analysis RELAGGS, which was first presented in more detail shortly after KDD-2001 [13]. We were especially interested in the opportunity to compare our approach to others and their results.

RELAGGS is intended to deal with relational data, i.e. data spread over multiple tables as is usually the case in a relational database. The data for tasks 2 and 3 were announced as relational data, so our attention was drawn here.

Before the Cup, our approach had been tested on several learning tasks concerning relational datasets from financial domains. Although these data had been used in data mining competitions in previous years, there was limited information about the results of their participants, so we were lacking chances for comparisons with our results.

Moreover, the data from the Cup's domain of biology offered a new challenge compared to those from insurance companies and banks. These biological data were more complex in terms of recursive structures introduced by interactions of proteins. Furthermore, we had only dealt with concept learning so far, not yet with multi-class problems.

3.2 Preprocessing with SQL

Tasks 2 and 3 asked for models to predict among 15 functions and 15 localizations of proteins, respectively. Two variants of the data were provided: firstly, a single table with single lines per example and almost 3.000 columns, and secondly, two tables as a relational dataset with less than 10 attributes each. Because of our interest in relational data mining, we focused on the latter variant of the data. The two tables given were named genes_relation and interactions_relation, cf. Figure 8 on the left.

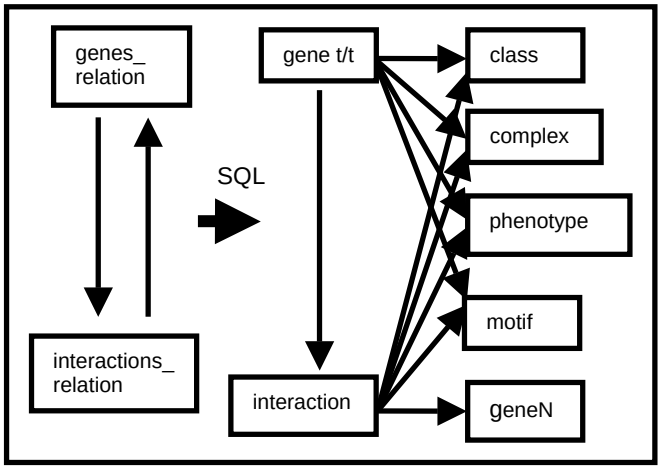


Figure 8. Renormalization of the original data (rectangles: tables, arrows between tables: foreign links, t/t: separate tables for training and test genes).

The genes_relation contained 862 training examples, the corresponding relation for testing included 381 test examples. There could be more than one line in the tables describing each gene. Genes could be identified with the help of a key attribute gene_id. RELAGGS takes as input a dataset in normal form (NF), especially demanding for single lines per example in the target table. This is why the original data had to be transformed with the help of SQL view definitions into a different representation, cf. Figure 8 on the right. Note that renormalization should be unnecessary in many practical applications, since databases are usually in normal form.

The data from the original table genes_relation was now represented in tables gene, class, complex, phenotype, and motif. Moreover, the original function and localization attributes in the gene table were split into a number of attributes: one for each function and one for each localization, with possible values 0 (false) and 1 (true) for training examples, and ? (unknown) for test examples. This was another prerequisite for the application of RELAGGS and SVM^{light}. Now, there is a one-to-many relationship between table gene and each of the other four tables. This is indicated by so-called *foreign links*, which correspond to foreign key relationships but differ from those wrt. the definition of direction [20].

Note that a natural join of gene, class, complex, phenotype, and motif on gene_id reproduces the original genes_relation, apart from differences due to the new representation of the target attributes for functions and localizations. Note also that a

universal join like this is problematic in the general case because of the dangers of exponential explosions or losing data.

Concerning the interactions_relation, we first tried to find out what kind of relation was established here, i.e. properties such as reflexivity, symmetry, and transitivity. We sent our questions concerning the former two properties to the Cup organizers and received a prompt and helpful reply, which also appeared in the Questions & Answers section of the Cup Web site. The property of reflexivity was not valid for the interactions relation, but symmetry was.

Since RELAGGS cannot (yet) deal with background knowledge in the form of general rules, e.g. a rule for establishing symmetry, this feature of interactions had to be made explicit. For each pair of genes (gene1, gene2) included in interactions_relation, we also inserted the pair (gene2, gene1) into our interaction table, if it was not already there.

The issue of transitivity became relevant to us only after the first question period of the Cup was over. We did not want to consider only direct "neighbors," i.e. genes/proteins interacting according to the orignal table interactions_relation (plus symmetry). However, RELAGGS cannot (yet) deal with cycles in the graph of tables and foreign links as contained in the original data. In this situation, we decided to take the following measures:

1. We tried to find a way to also have "neighbors of neighbors," etc. explicit in our interaction table. For non-direct neighbors, we computed the product of the correlation values of the corresponding interactions. An example: with (gene1, gene2) and (gene2, gene3) already in the interaction table, (gene1, gene3) and the product of the correlations of the original two interactions could be also inserted into the interaction table.

2. In order to avoid a very large interaction table, we concentrated on those interactions for which high correlation values were given or computed (> 0.5) and which did not exceed four levels of neighborhood.

3. We created an extra table geneN that included all genes which where "neighbors" of others according to our interaction table.

In this context, we introduced foreign links to the tables for class, complex, phenotype, and motif, in a special way. They would have had their origin in the table geneN, analogous to foreign links from the gene table. However, we had them originate directly from the interaction table. This "flattening" of the database schema made the computation of joins in the later steps of the KDD process more efficient.

Finally, after the test data had been provided, we integrated training and test data such that properties such as class information for test genes, which were neighbors of training examples, could be of influence on the models to be found. We just kept separate gene tables for training and testing because this simplified procedures like 10-fold cross-validation at later stages.

3.3 Preprocessing with RELAGGS

The name of our approach hints at the form of the input data to the system, viz. relations, and the application of aggregate functions to this input. (Occasionally, we now translate it into "*rel*y on *agg*regation, sometimes.")

So, what does RELAGGS do? It takes as input a description of the tables including their names and numbers of attributes, the names and types of attributes, lists of possible values for categorical attributes, and furthermore, a description of the foreign links and the target attribute. Finally, the contents of the tables also forms part of the input.

RELAGGS uses the foreign link information to compute join definitions, e.g. for a join of the tables gene and class, gene and complex, etc. Then, these joins are computed and SQL standard aggregation functions are applied (*count* for relations and categorical values; *avg*, *max*, *min*, and *sum* for numeric attributes) to collapse the joins into one line per gene. In the example, this means just counting the possible values for class, complex, etc. Unknown values are handled by simply not counting them. Finally, the target attribute (here localization) is concatenated with all the single lines per gene originating from the different joins.

The RELAGGS output consists of a single table, optionally formatted as input for C4.5 or SVM^{light}, cf. Figure 9. In inductive logic programming, the transformation of multiple tables into a single table is called propositionalization. For the final training and testing, a run of RELAGGS took about a minute, and the output table contained 938 columns.

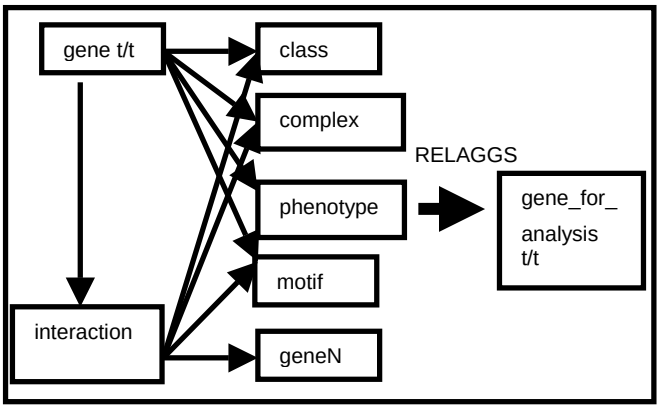


Figure 9. Propositionalization.

3.4 Data Mining with SVM^{light}

As mentioned above, RELAGGS can output files for C4.5 [12] and SVM^{light} [18]. In our earlier experiments, we had used these learners with their default parameter settings [13]. We used these settings here as well.

Actually, the first focus for optimizations was not on function prediction but on localization prediction, and especially on the

most frequent class nucleus (42.5% of all localizations). A series of more than 50 10-fold cross-validation experiments showed that

1. SVM^{light} performed better then both C4.5 and C4.5rules here (10.1% ± 2.6% vs. 14.9% ± 4.0% / 14.1% ± 3.9%), which was different from our earlier experiments with financial data, and with differences being statistically significant according to paired t-tests at 0.05 level.

2. SVM^{light} standard parameters performed well; among other findings was that a change from the default linear kernel to quadratic or kubic ones resulted in overfitting.

3. Other representations of the input data, such as a variant without interaction data, did not perform as well as the representation described above, although differences were usually small and statistically significant for some variants only.

An SVM^{light} run on the RELAGGS output took a few seconds and resulted in model files from the training genes and in prediction files for the test genes as depicted in Figure 10. Predictions are either above zero for positive cases or below zero for negative cases. The real-valued differences from zero express a degree of confidence in the predictions.

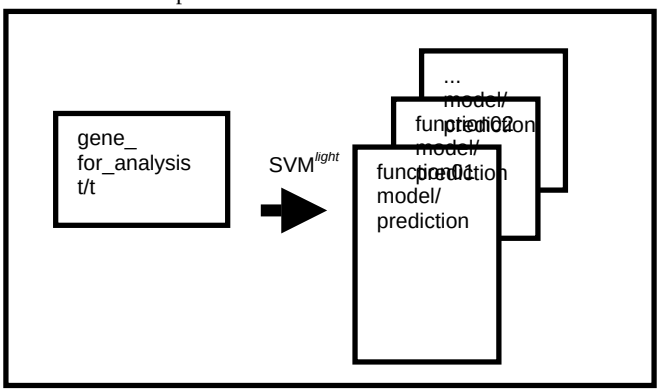


Figure 10. Application of SVM^{light}

3.5 Postprocessing with SQL

The predictions for single functions and localizations had to be integrated into a final solution. There was a simple opportunity for localization prediction: we chose the value for localization with the highest confidence value provided by SVM^{light}, regardless if positive or negative.

In the context of function prediction, we made the observation that the table for test genes included several identical lines per example after class, complex, etc. information had been extracted to different tables. We interpreted this circumstance as an information about the number of functions to be predicted per gene/protein.

When it came to function prediction, we at first chose those functions with a confidence higher than zero. In a next step, we tried to also include the function with a negative confidence closest to zero, if we had not yet as many functions predicted as indicated by the number of the identical lines in the gene table mentioned above. If we had too many functions predicted already, we removed the one with the smallest confidence. Based on 10-fold cross-validation, we found out that only the latter step of prediction removals improved prediction accuracy. So, we did only this for the construction of the final predictions, cf. also Figure 11.

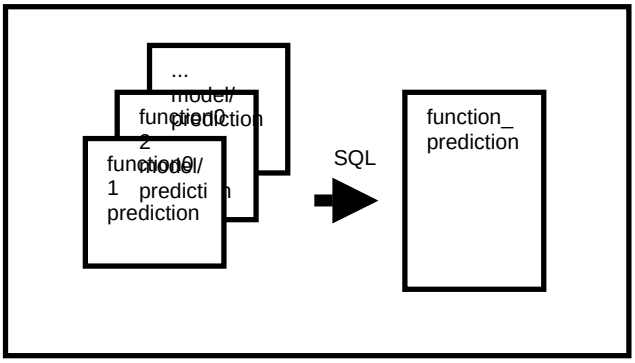


Figure 11. Construction of the final predictions.

3.6. Results, discussion, conclusion

From 10-fold cross-validation, we received an estimate of 92.9% accuracy for function prediction and 72.5% accuracy for localization prediction. From the Cup organizers, we received the actual results of 93.6% accuracy for function prediction and 69.8% accuracy for localization prediction. While the former made us the winner on task 2, the latter meant rank 4 on task 3.

It seems that it was a good strategy here to include all the information we could get about the genes/proteins, including the integration of training and test data and the usage of assumptions about the number of functions to be predicted.

Overall, we invested about 160 hours of work into this knowledge discovery project. Considering the fact that much of this time was devoted to things like replacing international decimal points by German decimal commas in order to get the data from one tool the other without sacrificing semantics, we consider it useful to integrate our tools in a database environment as a homogeneous platform. This should also be of advantage for handling larger datasets.

During the project, we learned that RELAGGS has some potential for further improvements such as support for multi-class problems. However, even the first version of RELAGGS was fast and simple to apply due to the opportunity to work immediatly with multiple tables and the easy specification of foreign links, and it was competitive on tasks 2 and 3 of the KDD-Cup 2001.

4. TASK 3: LOCALIZATION

Dataset 2, which was prepared for the KDD Cup 2001, has three interesting features: (1) the dataset contains many missing values; (2) the domain of the objective attribute contains fifteen non-ordered values; and (3) the dataset is a mixture of two types of data: one table correlates the features of individual genes and the other represents a binary relation that describes interactions between the genes. Although these are typical features of biological data, it is difficult to manipulate the data to achieve highly accurate predictions. In order to solve this problem, we used traditional approaches, such as decision trees, AdaBoost, and nearest neighbor methods. The nearest neighbor method represented the most promising approach because majority voting to predict multiple classes, and space reduction to handle missing values appeared to be straightforward, although the selection of optimal distances from the data required serious investigation. To overcome this problem, we calculated an "optimal" neighborhood, thereby maximizing the prediction accuracy against a test dataset (a subset of the given training dataset). Although this optimization problem is computationally intractable, we propose an efficient solution that involves a branch-and-bound searching strategy. This method was used successfully to generate the most accurate predictor in the KDD Cup 2001 Task 3 competition. In this section, we discuss the advantages of this method in terms of predictive accuracy and computational performance.

4.1 Task Overview

First, we give a brief overview of Task 3. The task was to predict the location in a cell where a given gene is active (where the protein for which the gene codes is located). The goal was to select one of the fifteen candidate locations within a cell, i.e., cell wall, cytoplasm, cytoskeleton, endosome, ER, extracellular milieu, Golgi, integral membrane, lipid particle, mitochondrion, nucleus, peroxisome, plasma membrane, transport vesicle, and vacuole. The experiment included 862 training genes and 381 test genes.

Each gene was assigned values for six attributes: Essential, Class, Complex, Phenotype, Motif, and Chromosome. The 'Chromosome' parameter corresponded to one of 16 chromosome numbers. The domains of the other five attributes were multiple sets. For instance, the value of 'Class' was a subset of 24 protein categories, such as 'Cyclins' and 'Transcription Factors'. However, 70% of the Class values were empty sets {}, and thus the values were missing. The 'Complex' attribute indicates members of the 56 protein complexes that are encoded by individual genes. 'Phenotype' and 'Motif' assign a gene to one of the 11 phenotypes or 351 types, respectively. An example from dataset 2 is shown below.

Gene	G234064
Essential	{Essential}
Class	{"GTP/GDP-exchange factors"}
Complex	{"Translation complexes"}
Phenotype	{}
Motif	{PS00824, PS00825}
Chromosome	1
Function	{"CELLULAR ORGANIZATION", "PROTEIN SYNTHESIS"}
Localization	cytoplasm

The dataset describes interactions between all of the 1243 gene pairs, which consist of 862 training genes and 381 test genes. The interactions between gene pairs can be classified as 'Physical', 'Genetic', 'Genetic-Physical' or 'No'. In the first three types of interactions, the strengths of the interactions are associated. It would be interesting to see if the type and strength of the observed interactions could be applied to increasing prediction accuracy. However, the total number of interactions was not sufficient to perform such a precise analysis, since information on most of the interactions between the genes is missing. Therefore, we decided to simplify the analysis by treating Physical, Genetic and Genetic-Physical as indicative of an observed interaction, and No as indicative of a failure to observe the interaction. We did not take into account the strength of each interaction. In this way, we generated binary pairs that described the interactions between genes, and we called this the *binary interaction relationship*.

4.1.1 Coping with Missing Values

We developed an approach for handling missing values, thereby allowing more accurate predictions. First, we observed that the Class, Complex, and Motif attributes were highly related to localization, while the other three, Essential, Phenotype, and Chromosome Number, did not correlate strongly with the objective attribute. Furthermore, with regard to the binary interaction relationship, we observed that genes that interacted with the focusing gene were usually located in the same part of the cell. This is a further indication that the binary interaction relationship is useful in predicting localization.

We also noted that although many attribute values for the 862 training genes were missing, at least one of the three attributes (Class, Complex, or Motif) was usually defined. Indeed, among the 381 test genes, 367 had one of these three values or interacted with other genes. Therefore, we decided to compensate for the missing information by using information on the three attributes and the binary interaction relationship.

4.1.2 Different Test Approaches

Once the four features had been selected, we applied three independent approaches to the data analysis. In [15], we have previously developed an efficient method of mining correlated association rules that have strong relevance to the objective attribute, such as localization. We tested the usefulness of this approach in tackling this particular problem.

First, we generated a decision tree [18] that was labeled with correlated association rules [1,15]. This approach made a good score in the Task 2, and our prediction was ranked third. Second, we made combinations of several correlated association rules using the AdaBoost strategy [14,15] in order to boost the overall

prediction accuracy. Third, we used the nearest neighbor strategy [9]. Of these three approaches, the nearest neighbor method worked best for the training dataset. Finally, we applied the nearest neighbor method to the test dataset.

Gene	Essential	Class	Complex	Motif	Chromosome	Localization
G234126	{Non-Essential}	{GTP-binding}	{Translation}	{PS00017}	2	cytoplasm
G235065	{Non-Essential}	{GTP-binding}	{Translation}	{PS00301}	16	cytoplasm
G234064	{Essential}	{GTP/GDP-exchange}	{Translation}	{PS00824, PS00825}	1	cytoplasm
G235357	{Non-Essential}	{}	{Translation}	{PS00017, PS00190}	7	mitochondria

Table 1. Example of genes.

Gene	Gene	Type	Strength
G234064	G234126	Genetic-Physical	0.914095071
G234064	G235065	Genetic-Physical	0.751584888
G235357	G239653	Genetic	0.891039915

Table 2. Binary interaction relation between pairs of genes.

4.2 Nearest Neighbor Analysis

4.2.1 Attribute Agreement of Records

Let R be a relation that is a set of records with which certain attributes (features) $f_1,f_2,...,f_k$ are associated. We assume here that R has two attributes of special. One is the key attribute used for the unique identification of each record. The other is the objective attribute of a classification problem. Let D_i be the domain of f_i. Let r be a record in R. We then denote the f_i's value in r by $r[f_i] \in D_i$. Without loss of generality, we assume that $r[f_i]$ is a set of values. In the case where the f_j's value of r is a single value c, we formally regard it as the singleton set $\{c\}$, but we simply describe the set as c for readability. r_1 and r_2 in R are called to *agree* on f_i if $r_1[f_i]$ and $r_2[f_i]$ share some common elements; namely, $r_1[f_i] \cap r_2[f_i] \neq \phi$.

For example, consider the four records in Table 1, in which the key attribute is Gene and the objective is Localization. The four records in the table are members of dataset 2. Observe that G234126 agrees with G235065 with respect to the Essential, Class, and Complex attributes, because

$$G234126[Class] \cap G235065[Class] = \{GTP\text{-}binding\}$$

G234126 also agrees with G235357 in terms of Essential, Complex, and Motif attributes.

One might consider defining the degree of agreement, because the interaction of $r_1[f_i]$ and $r_2[f_i]$ possibly involves more than one element, and the number of common elements would be expected to indicate the strength of agreement of r_1 and r_2 on f_i. However, the number of shared elements in dataset 2 is typically no greater than two, and therefore we cannot define the strength of agreement. Nevertheless, it is interesting to see whether the use of agreement strength contributes to improvements in prediction accuracy.

Binary interactions between pairs of genes are also represented by a relationship. Table 2 illustrates a *binary interaction relation*; that is, a set of binary pairs of genes and the strengths of their associated interactions. These interactions are also from dataset 2. Two genes are deemed to *agree with* a binary interaction relation if the pair is listed in the relation. For instance, G234064 and G234126 agree with Table 2.

4.2.2 Neighbors

We are now in a position to define neighborhoods among the records. We focus on G234126 in Table 1 and define its neighbors. We see that the top three genes (G234126, G235065, and G234064) in the table are located in the cytoplasm. In order

to define the neighborhood of the focusing gene G234126, we utilize the notion of attribute agreement between two genes; that is, two records are *neighbors* if they agree with respect to certain attributes. It is then possible to select attributes that are useful in the prediction of localization.

The selection of Chromosome in Table 1 is not effective, because none of the other three genes match G234126 with regard to this parameter, and therefore it is impossible to infer the localization of G234126. By choosing Motif, the bottom gene G235357 becomes the neighbor, but it is located in mitochondria. The choice of Complex designates all three genes as neighbors of G234126, and most of these are located in the cytoplasm. One can also consider the binary interaction relation in Table 2. G234064, which is located in cytoplasm, corresponds only to the focusing attribute G234126 in this table. The choice of Class appears to be appropriate, since G235065, which is located in the cytoplasm, becomes the neighbor of the focusing gene.

4.2.3 Nearest Neighbor Assignment by Prioritizing Attributes

In practice, there are many missing values, such as those seen in dataset 2, and thus the use of only a single attribute may not be sufficient for accurate prediction. Therefore, in order to assign neighborhoods that are based on agreement we need to examine more than one attribute. For instance, in Table 1, one can combine Class, Complex, and Motif attributes together with the binary interactions in Table 2. We see that the bottom three genes can be treated as neighbors since they agree with G234126 with regard to Class, Complex and Motif.

In cases where the number of neighbors is large, we perform a further selection among the neighbors. In order to achieve this, we prioritize the attributes. As an example, we choose Complex as the primary attribute and extract all the genes that agree with G234126 on Complex; that is, all of the bottom three genes. We then select Class as the secondary attribute, and we extract the genes that agree with G234126 with regard to Class, thus revealing G235065, which is located in the cytoplasm.

We now present a formal description of the method used to restrict neighbors. Suppose that we select some features from f_1, f_2, ..., f_k, and prioritize them to yield a sequence of features $[g_1,...,g_m]$ that is ordered from left to right (we denote order by enclosing a sequence within square brackets). Let $r \in R$ be a test record, and let N_i denote a set of neighbors of r. As a training dataset, let us select a non-empty set $R^{train} \subseteq R$, such that r is not a member of R^{train}. In the initial step, we assign R^{train} to N_0, and we continue to restrict N_i by using the priority list $[g_1,...,g_m]$, as illustrated in Figure 12.

$N_0 := R^{train}$;
for each i = 1,2,...,m **begin**
  $N_i := \{x \in N_{i-1} \mid x$ and r agree on $g_i\}$;
  if $N_i = \phi$ **then** $N_i := N_{i-1}$;
end
return N_m;

Figure 12: Computing the nearest neighborhood for a priority list

In each step, we compute the elements in N_{i-1} that agree with r on the next attribute g_i, and we assign the set to N_i. The new set N_i may be empty due to many missing attribute values. In order to avoid losing all the neighbors of r when N_i is empty, we continue the calculation by re-assigning the previous non-empty N_{i-1} to N_i. We repeat these steps until $i=m$. The final set, N_m, contains neighbors that have survived agreement tests on attributes through as many higher priorities as possible. Therefore, we call the members of N_m the *nearest neighbors of r with respect to the initial training dataset* R^{train} *and the priority list* $[g_1,...,g_m]$. We denote the final answer N_m as: $NN(r, R^{train}, [g_1,...,g_m])$.

4.2.4 Classification by Nearest Neighborhood Analysis

Let *obj* be an objective attribute, such as Localization, and let D_{obj} be its domain. We calculate the objective value of r, $r[obj]$ from the majority of objective values of nearest neighbors in $NN(r,R^{train}, [g_1,...,g_m])$ using the formula:

$$arg\ \max_{d \in D_{obj}} |\ \{\ x \in NN(r, R^{train}, [g_1,...,g_m]) \mid x[obj]=d\ \}\ |,$$

which is denoted by $predict(r, R^{train}, [g_1,...,g_m])$.

Let R^{test} be a test dataset, such that $R^{test} \subseteq R$, and R^{test} is disjoint from the training dataset R^{train}; that is, $R^{test} \cap R^{train} = \phi$. The prediction accuracy of our classification method using the test dataset R^{test} is

$$\frac{|\ \{\ r \in R^{test} \mid r[obj]=predict(r, R^{train}, [g_1,...,g_m])\ \}\ |}{|R^{test}|}$$

which is referred to as $accuracy(R^{test}, R^{train}, [g_1,...,g_m])$ in the following discussion.

So far we have assumed that a priority list $[g_1,...,g_m]$ is provided for the definition of nearest neighborhoods. In fact, the choice of priority list significantly affects the prediction accuracy by selecting an optimal priority list that maximizes accuracy. Generally, this optimization problem is NP-hard.

Theorem: It is NP-hard to compute $[g_1,...,g_m]$ that optimizes $accuracy(R^{test}, R^{train}, [g_1,...,g_m])$.

The proof of this theorem is given at the end of this section.

4.3 Computing Optimal Priority

In this section, we present an efficient branch-and-bound search technique for solving the optimization problem.

First, it is noteworthy that if the objective value of any record $x \in NN(r, R^{train}, [g_1,...,g_m])$ does not coincide with $r[obj]$ ($x[obj] \neq r[obj]$), it is impossible to predict $r[obj]$ correctly. In this case, r is classified *unpredictable* using the nearest neighborhood technique. The ratio of unpredictable records allows us to bound the prediction accuracy:

$$accuracy(R^{test}, R^{train}, [g_1,...,g_m])$$
$$\leq \quad 1 - \frac{|\{x \in R^{test} \mid x \text{ is unpredictable.}\}|}{|R^{test}|}. \qquad (2)$$

The upper bound in the right-hand side is denoted by $ub([g_1,...,g_m])$.

Second, the above upper bound decreases monotonically for any extension $[g_1,...,g_m,...,g_n]$ of $[g_1,...,g_m]$; that is,

$$ub([g_1,...,g_m,...,g_n]) \leq ub([g_1,...,g_m]). \qquad (3)$$

By using the procedure for computing the nearest neighborhood (shown in Figure 11), it is easy to see that the nearest neighborhood shrinks for extensions:

$$NN(r, R^{train}, [g_1,...,g_m,...,g_n]) \subseteq NN(r, R^{train}, [g_1,...,g_m])$$

It immediately follows that if r is unpredictable when using $NN(r,R^{train},[g_1,...,g_m])$, then r is unpredictable when using $NN(r,R^{train},[g_1,...,g_m,...,g_n])$. Thus the number of unpredictable

records increases monotonously if a priority list is extended by adding attributes, which lends proof to (3) above.

Finally, suppose that we find a priority list P, such that

$$ub(Q) \leq accuracy(R^{test}, R^{train}, P)$$

From Equations (1) and (2), for any extension Q' of Q

$$accuracy(R^{test}, R^{train}, Q') \leq accuracy(R^{test}, R^{train}, P).$$

This property motivates us to develop a branch-and-bound searching algorithm that can be used to compute the optimal priority list, and thereby maximize the prediction accuracy. Consider a search tree of priority lists in which the root is the empty list [], and any child priority list corresponds to its parent list with one new attribute added at the tail. Starting at the root empty list [], we gradually expand the ensemble of candidate lists and maintain the priority list, say P_{max}, which temporarily maximizes the prediction accuracy. In each step, we select a frontier node in the ensemble, and we investigate its children (in this instance Q). If $ub(Q) \leq accuracy(R^{test},R^{train},P_{max})$, we can safely prune the subtree rooted at Q without fear of losing the optimal solution.

One may wonder if this branch-and-bound heuristic is effective, especially in cases where the objective attribute is Boolean. However, in the case of dataset 2, the number of values in the objective attribute is fifteen, and thus there are many opportunities to determine whether the records are unpredictable, in which case the upper bound is sharply lowered. Actually, our branch-and-bound search technique investigates about 10% of all the possible orders of the seven attributes in dataset 2.

4.4 Experimental Results

4.4.1 Predictive Accuracy

Let S^{train} and S^{test} denote the training data and the test data, respectively, of dataset 2, which was provided by the KDD Cup 2001. We generated the optimal priority list of attributes by treating S^{train} as both the training dataset and the test dataset, and found that the priority list [Complex, Class, Interaction, Motif] ($=P_{max}$) maximized the prediction accuracy at 79%; that is,

$$accuracy(S^{train},S^{train},P_{max}) = 79\%.$$

A prediction that used [Complex, Class, Interaction, Motif] also achieved 72% accuracy against the test dataset S^{test};

$$accuracy(S^{test},S^{train},P_{max}) = 72\%.$$

Since the prediction accuracy decreased by 7%, the optimal priority list slightly overfitted the training data. The priority list indicates that incorporation of the other three attributes, Essential, Phenotype, and Chromosome, into the list does not lead to improvements in the prediction accuracy.

Since all the values of Localization were available, we performed additional experiments. The second column of Table 3 includes the prediction accuracy of the singleton set of each attribute. In order to show that the prediction accuracy improves when attributes are appended to each singleton set, the third column shows the optimal priority list, starting with each attribute listed in the first column, and the fourth column lists their prediction accuracies. Figure 13 also illustrates how the prediction accuracy improves step-by-step until the optimal priority list in Table 3 is generated.

Singleton List	Acc1(%)	Multiple List	Acc2(%)
[es]	45.7	[es,co,in,mo]	69.0
[cl]	48.8	[cl,in,co,es,ph,ch]	69.7
[co]	62.5	[co,cl,in,mo]	72.2
[ph]	47.0	[ph,co,in,es,cl]	69.6
[mo]	49.1	[mo,co,in,cl,ph]	70.2
[ch]	44.9	[ch,co,in]	59.6
[in]	65.1	[in,mo,co,cl,es]	69.7

Table 3: Accuracy improvement for optimal priority list. Acc1 = accuracy(S^{test}, S^{train}, "Singleton List"). Acc2 = accuracy(S^{test}, S^{train}, "Multiple List"). es=Essential, cl=Class, co=Complex, ph=Phenotype, mo=Motif, ch=Chromosome, in=Interaction.

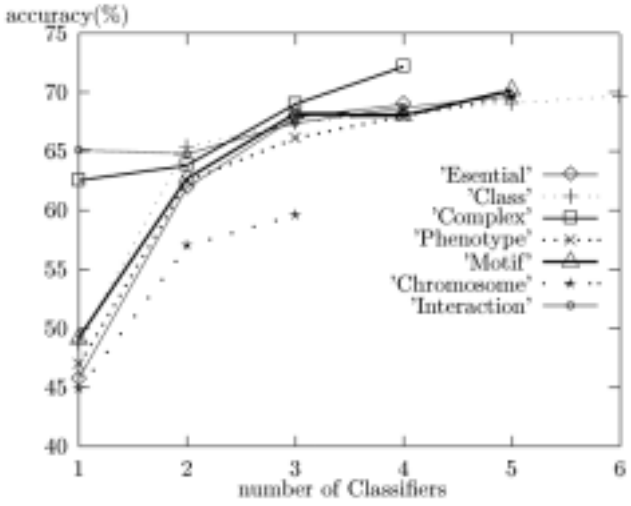


Figure 13: Accuracy improvement by increasing the number of attributes.

After considering all the possible priority lists, we decided that the priority list [Complex, Class, Interaction, Motif] was optimal for the given training dataset. Although it is worth considering whether another priority list might out-perform this priority list with respect to prediction accuracy against the test dataset, Table 3 indicates that this is not the case. We further investigated whether permutations of [Complex, Class, Interaction, Motif] might make good markers, and the results are shown in Table 4.

We implemented the branch-and-bound strategy in C++. We implemented the entire system and evaluated its performance on a Dell Inspiron 5000e containing a Pentium-III processor with a clock rate of 750 MHz and memory capacity of 384 MB. It took about 6 seconds to compute the optimal priority and to output the prediction for Task 3 against dataset 2, which comprised 862 training records and 381 test records.

Permutation				Accuracy(%)
cl	co	mo	in	69.4
cl	co	in	mo	69.5
cl	mo	co	in	68.6
cl	mo	in	co	67.6
cl	in	mo	co	68.9
cl	in	co	mo	69.2
co	cl	mo	in	71.0
co	cl	in	mo	72.2
co	mo	cl	in	70.5
co	mo	in	cl	70.3
co	in	mo	cl	71.0
co	in	cl	mo	71.0
mo	cl	co	in	67.4
mo	cl	in	co	66.3
mo	co	cl	in	67.9
mo	co	in	cl	70.0
mo	in	co	cl	67.9
mo	in	cl	co	67.9
in	cl	co	mo	68.7
in	cl	mo	co	69.2
in	co	cl	mo	68.9
in	co	mo	cl	68.9
in	mo	co	cl	69.5
in	mo	cl	co	69.2

Table 4: Accuracies of permutations of the optimal priority list [Complex, Class, Interaction, Motif].

4.5 Discussion and Technical Proof

From the biological viewpoint, proteins interact and work together to achieve certain functions at a specific location in the cell. Thus, the binary interaction relation and the Complex attribute should correlate with localization, though it is rather surprising to see that the binary interaction is less important than Complex and Class in terms of prediction accuracy improvement. On the other hand, it has been observed that eucaryotic genes with similar functions are not necessarily coded on the same chromosome. Even in the absence of this knowledge, our data mining method automatically selects biologically important features to ensure better predictions.

We close this section with a proof of the result mentioned earlier.

Theorem It is NP-hard to compute $[g_1,...,g_m]$ that optimizes $accuracy(R^{test}, R^{train}, [g_1,...,g_m])$.

Proof We present transformation from the minimum cover problem. Let E be a collection of subsets of a finite set V. A *cover* is a subset of E such that any element in V belongs to at least one member of the cover. Deciding whether or not there exists a cover of size K for a given positive integer $K \le |E|$ is NP-complete. We show that an algorithm for computing the priority list P minimizing $accuracy(R^{test}, R^{train}, P)$ is also able to calculate the minimum cover.

We first define test records. We call a record *positive* (respectively, *negative*) if its objective value is 1 (0). We regard all the records in V as positive test records, while from each $e \in E$, we create a unique negative test record neg_e; that is, $v[obj]=\{1\}$ if $v \in V$, and $v[obj]=\{0\}$ if $v=neg_e$ for $e \in E$.

Let R^{test} denote the set of test records, $V \cup \{neg_e | e \in E\}$. We then regard $e \in E$ as an attribute that assigns $\{1\}$ to members in e and neg_e but assigns the empty set to the others, for the purpose of distinguishing members of e and neg_e from the others; that is, $v[obj]=\{1\}$ if $v \in e$ or $v=neg_e$, and $v[obj]=\{\}$, otherwise.

We next define the set of training records. We first generate $|E|$ positive training records a_e ($a_e[obj]=1$) for $e \in E$ such that any test record $v \in e$ and neg_e become the neighbors of a_e; that is, $a_e[f]=\{1\}$ if $e=f$, and $a_e[f]=\{\}$, otherwise.

Next, we create $(|E|+1)$ negative records b_i ($b_i[obj]=0$, $i=1,..., |E|+1$) that do not agree with any training record on any attribute (that is, $b_i[e]=\{\}$ for $e \in E$). Now the majority of the training records is negative. Let R^{train} denote the set of all the training records, $\{a_e \mid e \in E\} \cup \{b_i \mid i=1,...,|E|+1\}$.

Predicting whether $v \in R^{test}$ is either positive or negative is performed as follows:

- If $v \in e$ or $v=neg_e$ for some $e \in P$, v agrees with $a_e \in R^{train}$ on e, and hence $predict(v, R^{train}, P)=1$.
- Otherwise, the whole training set R^{train} becomes the neighborhood of v. Because negative records are major in R^{train}, $predict(v, R^{train}, P)=0$.

If P is empty, according to the latter case, the prediction is incorrect for positive test records but is correct for negative test records. In order to improve the prediction accuracy, one may include e into P to reverse the prediction for $v \in e$ and neg_e, which makes the prediction correct for $v \in e$ but incorrect for neg_e. Inclusion of e increases the prediction accuracy if the number of elements in e is more than one. To ensure this improvement, let us make one copy v' of $v \in V$ so that $v \in e$ if and only if $v' \in e$.

Suppose that $\cup\{f \mid f \in P\}$ fails to include V. To improve the prediction accuracy, for each $v \in V - \cup\{f \in P\}$, we ought to find e such that $v \in e$ and add e into P. Since inclusion of e mis-predicts the value of neg_e, to achieve the best prediction accuracy, we must cover V using the minimum number of elements in E, which is equivalent to compute the minimum cover. *Q.E.D*

5. CONCLUSIONS

The biological community has seen numerous technological breakthroughs in the last decade. These include (but are not limited to) fast genome sequencing techniques, high-throughput screening robots for testing large libraries of small molecules for binding to target proteins, and gene expression microarrays to measure gene transcription. Still more breakthroughs are on the horizon in areas such as proteomics, where the amount of protein made from a gene can be measured directly, rather than merely measuring the amount of mRNA as in current microarrays. All these technologies are providing vast amounts of data, of a wide variety of forms. In addition, much domain knowledge already exists, such as information about some metabolic pathways. It may even be said that biology is both data rich and knowledge rich, with the challenge being to synthesize these to produce still more knowledge. As a result, biology is not only a major user of data mining tools, but also a driving force for the development of future data mining algorithms. For this reason KDD Cup 2001 focused on biological applications of data mining. This section concludes by summarizing lessons for mining biological databases in particular, followed by general lessons for data mining, that have been touched on in this paper.

5.1 Lessons for mining biological databases

The expression of one gene frequently regulates the expression of other genes, resulting in a network of various regulatory pathways. In addition, one protein may interact with a number of other proteins in metabolic or signaling pathways. Therefore, there is widespread belief in biology that as we learn more about pathways, this new knowledge will help us determine the functions of many genes. Such is the motivation, for example, for predicting regulatory pathways from gene expression microarray experiments. Hence, it is very surprising that protein interaction information was not more useful in Tasks 2 and 3. Section 3 notes that interaction information was of some value for the winning approach to Task 2 but not particularly great value. And while interaction information was one of four items of information used in the winning approach to Task 3, interaction was the least important of these four. It might be perhaps that we still are missing too much interaction information for it to be highly useful. But that would be a disappointing conclusion given that the organism we were studying, yeast, has far more interaction information available than does any other organism. Or perhaps still better techniques are needed for using this information. For example, perhaps a Bayesian approach could be taken to predicting pathways, with associated probabilities, and these predicted pathways could be used to help with function prediction. The question of whether we can get more out of the available interaction information is of major importance.

A second lesson for mining biological databases is the issue of interacting with the laboratory. For Task 1, the test set was drawn from a very different distribution than the training set. This is because the chemists selected the molecules to make in the second round of experiments based on the results of the first round; they chose to synthesize molecules in the second round that "looked" like the active molecules from the first round. It has been the Cup co-chairs' experience that when collaborating with biologists and chemists, data often comes in "waves," and the selection of data points in the next "wave" is dependent on the results (labels of the data) in the previous wave. Because data mining tools tacitly assume training and test data are drawn from the same distribution, they often perform worse on the new wave than cross-validation on the previous wave would predict. But this iterative process also is a great opportunity for data mining to influence experimentation. This influence is not according to the simple "membership query" model that many adopt, where the system gets to ask the label of one data point at a time. And the experimenter is not simply at the service of the data mining system but instead may need to be told why an experiment is useful, e.g., why certain molecules should be made and tested. Real-world experience into such interaction between machine and scientist is a key area for data mining research, particularly with respect to biological applications.

5.2 General lessons for data mining

The co-chairs' intention for KDD Cup 2001 was that biological knowledge would not be a prerequisite for entry into the competition. It appears from the diversity of backgrounds of entrants and winners that we succeeded in that regard. But an equally important intention was that lessons would arise that would be of value to data mining outside of biological applications. We now consider several such lessons.

First, conventional wisdom holds that while Bayes nets are good for modeling the distribution from which data are drawn, for a pure classification task it is hard to beat modeling approaches designed specifically for classification, such as decision trees, SVMs, or ensembles. For example, we use Bayes nets to model gene expression data when we want to identify clusters of genes that appear highly interrelated. But if we want to distinguish between two types of patients (e.g., cancer vs. non-cancer), we turn to classification algorithms. In light of this conventional wisdom, it is quite surprising that of 114 entries for Task 1 (a pure 2-valued classification task) the winning approach was a Bayes net. The lesson is that Bayes nets should not be rejected out of hand for pure classification tasks. Indeed, as a direct consequence of this result one of the co-chairs (Page) is now employing Bayes net learning in a study of gene expression data for a classification task dealing with multiple myeloma, a blood cancer. The end of Section 3 provides several insightful comments about why the application of Bayes nets to Task 1 was successful.

The second lesson is one that has been recognized before within the inductive logic programming (ILP) community but is little known outside it. This is that *propositionalization* often is a good approach to a relational learning task. Tasks 2 and 3 are classic relational learning tasks, in which the information about a data point (gene) includes not only features of that data point itself but also how it relates to other data points. Many data mining tasks in relational databases have a similar nature; one

needs only to have a many-many relationship between entities of interest in the database to have a true relational task. Even though Task 2 and Task 3 are relational, the winning approach to Task 2 used an SVM and the winning approach to Task 3 used a neighborhood strategy. The key in Task 2 was a good algorithm for feature construction from relations. The key in Task 3 was to change the typical definition of distance. Ordinarily, distance is a measure of how similar two feature vectors are. The modification made was to incorporate the interaction relation into the distance measure, by making two data points nearer to each other if they were known to interact.

Finally, in Subsection 5.1 we noted challenges raised by interacting with an experimental laboratory. These include the need for improved human-computer interaction (e.g., in proposing *and motivating* a new round of experiments) and the question of how to handle a changing distribution over data. Note that this latter point seems related to "concept drift" but is different. The concept does not change, e.g., there is no change in what makes a molecule bind to thrombin. This issue of changing distributions raises questions about performance evaluation. If the change merely affects the ratio of one class to another then AUC, or area under a (ROC) curve, addresses the problems one would have if using only accuracy.[6] But what if the changes for example "focus in" on one area of the space, making some potential data points much more probable? This is what happened in Task 1. Is there a single measure, such as AUC, that can tell us one model or modeling approach will be more robust to these changes than another, or is better over some space of possible changes?

6. ACKNOWLEDGMENTS

Christos Hatzis wishes to thank Chris Kostas of Silico Insights for help in organizing, normalizing and cleaning the various datasets for Dataset 2. Mark Krogel wishes to thank his supervisor, Stefan Wrobel, who encouraged his participation in the Cup, and Anja Rohleder and Christiane Mikosch who supported tests of hypotheses. David Page wishes to acknowledge NSF grant 9987841.

7. REFERENCES

1. Agrawal, R., Imielinski, T., and A.N. Swami. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, D.C., May 26-28, 1993}, pages 207--216. ACM Press, 1993.
2. Cheng, J. (1998). *PowerConstructor* System. http://www.cs.ualberta.ca/~jcheng/bnpc.htm.
3. Cheng, J. (2000). *PowerPredictor* System. http://www.cs.ualberta.ca/~jcheng/bnpp.htm.
4. Cheng, J., Greiner, R. (1999). Comparing Bayesian network classifiers. In *UAI-99*.
5. Cheng, J. and Greiner, R., Learning Bayesian Belief Network Classifiers: Algorithms and System. *Proceedings of 14th Biennial conference of the Canadian society for computational studies of intelligence*, 2001.
6. Cheng, J. *et al*. (2001). Learning Bayesian networks from data: An information-theory based approach. *To appear in Artificial Intelligence Journal.*
7. Cooper, G.F. and Herskovits, E. (1992). A Bayesian Method for the induction of probabilistic networks from data. *Machine Learning*, 9 (pp. 309-347).
8. Freund, Y. and Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119--139, Aug. 1997.
9. Hand, D., Mannila, H. and Smyth, P. *Principles of Data Mining*. MIT Press, 2001.
10. Hanley, J.A. and McNeil B.J. (1982). The meaning and use of the area under a Receiver Operating Characteristic (ROC) curve. *Radiology*, 143, pp. 29-36.
11. Heckerman, D. (1995). A tutorial on learning Bayesian networks. *Technical Report MSR-TR-95-06*. Microsoft Research.
12. Joachims, T. Making Large-Scale SVM Learning Practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.
13. Krogel, M.-A. and Wrobel, S. Transformation-Based Learning Using Multirelational Aggregation. In C.Rouveirol and M.Sebag, editors, *Proceedings of the Eleventh International Conference on Inductive Logic Programming (ILP)*, LNAI 2157. Springer-Verlag, 2001.
14. Morishita, S. Computing optimal hypotheses efficiently for boosting. *Springer LNAI: Progresses in Discovery Science*, in press.
15. Morishita, S. and Sese, J. Traversing itemset lattices with statistical metric pruning. *Proc. of ACM SIGACT-SIGMOD-SIGART Symp. on Database Systems (PODS)*, pages 226--236, May 2000.
16. Neapolitan, R.E. (1990), *Probabilistic reasoning in expert systems: theory and algorithms*, John Wiley & Sons.
17. Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: networks of plausible inference*, Morgan Kaufmann.
18. Quinlan, J.R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
19. Spirtes, P., Glymour, C. and Scheines, R. (1993). *Causation, Prediction ,and Search*. Springer Lecture Notes in Statistics.
20. Wrobel, S. Inductive Logic Progamming for Knowledge Discovery in Databases. In N. Lavrac and S. Dzeroski, editors, *Relational Data Mining*. Springer-Verlag, 2001.

[6] It would be nice to use a measure such as AUC in future KDD Cups instead of accuracy, but this is impractical. With accuracy, entries can be simple files of predictions, whereas AUC would require Cup chairs to run everyone's predictors multiple times. "Lift" could be used, but as mentioned earlier, it requires participants to rank their predictions, and some approaches are ill-suited to this.

About the authors:

Jie Cheng is a senior data modeler at the Risk Management Division of Canadian Imperial Bank of Commerce. He received

his Ph.D. from the Faculty of Informatics of University of Ulster, UK. He then worked as a post-doc fellow with Professor Russell Greiner at the Computing Science Dept. of the University of Alberta. More about his research can be found at www.cs.ualberta.ca/~jcheng.

Christos Hatzis is the Vice President of Informatics and co-founder of Silico Insights, an informatics-based drug discovery company. Prior to Silico Insights, he was involved in developing and deploying real-time expert systems and knowledge discovery platforms for biotechnology companies. He earned his Ph.D. from the Department of Chemical Engineering and Materials Science at the University of Minnesota, Minneapolis.

Hisashi Hayashi is a Masters student of Dept. of Computer Science at the University of Tokyo. He has an interest in theoretical and practical problems of improving prediction accuracy and computational performance of data mining algorithms.

Mark-A. Krogel received his first degree in computer science from Magdeburg University, Germany, and an M.Sc. in cognitive science from Edinburgh University, Scotland. After several years in database application development, he is now a Ph.D. student in Stefan Wrobel's research group for Machine Learning & Knowledge Discovery at Magdeburg University.

Shinichi Morishita is an associate professor in the Dept. of Complexity Science and Engineering and Dept. of Computer Science at the University of Tokyo. He has been working on bio-informatics, data mining algorithms, database systems, and computational logic. URL: http://www.gi.k.u-tokyo.ac.jp/~moris/.

David Page is an assistant professor of Biostatistics and Medical Informatics and of Computer Science at the University of Wisconsin at Madison. He earned his Ph.D. from the Computer Science Department at the University of Illinois at Urbana-Champaign. More about his research and background can be found at www.cs.wisc.edu/~dpage.

Jun Sese is a Ph.D. student at the Dept. of Complexity Science and Engineering at the University of Tokyo. He is now developing an efficient algorithm for computing correlated association rules that can be naturally incorporated into decision-trees to achieve highly accurate predictions.