Rule-Based Data Mining Methods for Classification Problems in Biomedical Domains

Jinyan Li Limsoon Wong
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
Email: {jinyan, limsoon}@i2r.a-star.edu.sg

July 26, 2004

A tutorial note for the 15th European Conference on Machine Learning (ECML) and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Pisa, Italy September 20-24, 2004.

Abstract

This is an introductory-to-intermediate level tutorial that takes about three and a half hours. It aims to introduce the importance of rule-based data mining methods for solving biomedical classification problems. We expect it to be suitable for final-year and post-graduate students in computer science or computational biology, as well as post-doctorates starting work on general data mining or bioinformatics topics. We do not assume any special deep background in statistics, mathematics, or computer science.

This tutorial begins with a description of a data set repository where we have stored about 20 high-dimensional biomedical data sets. Then we talk about decision trees, committees of decision trees, and also about how they are used in bioinformatics. Then we discuss some interesting rules and patterns discovered from the data sets. Finally, we demonstrate how to use a software package that implements a wide range data mining and machine learning algorithms including our own algorithms. This software is free for academic use.

1 Introduction

Many bio-medical applications—such as diagnosis using gene expression profiles [2], relapse studies [98], and subtype distinction of a heterogeneous disease [105]—are typical supervised learning problems. In these applications, a learning algorithm is usually presented with a set of training samples where each sample is described by a vector of feature values and a class label. For example, in gene expression cancer diagnosis problems the features are genes, the values are the genes' expression levels, and the class label may indicate whether or not a cell is normal or cancerous. The learning has two primary goals. One is to induce from the training data a *classifier* that can classify future samples with high accuracy. In addition to inducing a highly accurate classifier, the other purpose of the learning is to obtain explicit, easily comprehensible, and reliable decision rules or patterns from the training data for potential clinical use.

Supervised learning has been extensively studied in machine learning and data mining. So there are many choices of learning algorithms to achieve the first goal of inducing accurate classifiers from training data. For example, neural networks [39] and its extension concept—support vector machines (SVMs) [16, 18]—can approximate the training data to an almost error-free [17, 42] degree, and can exhibit good generalization capability on test samples. However, the structure of these classifiers is very complicated especially when the input feature space is large. This complexity is due to two reasons: (1) All the input features have to be used in these classifiers. If a dataset is described by 1000 features, then the mapping—*i.e.*, the classifier—has 1000 variables. (2) The kernel functions are usually set as non-linear functions for the purpose of achieving good accuracy. Even when they are set as linear functions, the hyperplanes—say with 1000 dimensions—are still hard to understand.

Therefore, an ideal classifier should be a classifier that is accurate and that uses only a small proportion of the original features in a systematic way. A decision tree is a good learning algorithm that can induce classification rules that use a small number of features. Suppose a gene expression cancer diagnosis dataset have n' number of features. Using a tree induction algorithm such as C4.5 [82], we can usually get a decision tree containing a very small n'' number of features. Therefore, the interpretation of the decision results needs only the n'' features rather than the n' features. From our previous studies, we find that a decision tree derived from gene expression data usually contains less than 10 features. The concentration on these recommended features may help biomedical experts to understand better and deeper about some biomedical mechanism.

2 Some Bioinformatics Problems and Data Sets

Large scale high-throughput biomedical data are increasingly accumulated and analysed at different medical centers, hospitals, research institutes, and universities world-wide. In this tutorial, we first introduce a data set repository where tens of different high-dimensional biomedical data sets have been stored in one place. It is named the Kent Ridge Biomedical Data Set Repository and is available at `http://sdmc.i2r.a-star.edu.sg/rp`. This is an online repository of high-dimentional biomedical data sets, including gene expression data, protein profiling data and genomic sequence data that are published recently in *Science*, *Nature*, and other prestigious journals. All of these data are for classification purposes, such as disease diagnosis, subtype classification, relapse study, genomic feature prediction, and so on. These biomedical applications are also challenging problems to the machine learning and data mining community.

The original file formats of the original raw data are different from common ones used in most

machine learning softwares. In this repository, all these original raw data files have been converted into plain text data files under the same relational data schema where every row in the new file is a comma-punctuated string, like a vector, representing a labelled data sample—*e.g.*, a tumor sample with expression values of 1000 genes—and where every column represents a feature or variable—*e.g.*, a gene. Note that the last column in the new files are all class labels of a particular application. The re-formatted data files are named with the extension of `.data`; and the feature names are saved in a separate file named with the extension of `.names`. The equivalent `.arff` format is also provided. Such transformed data files can be directly fed to common machine learning software packages such as C5.0, MLC++ [52], and the WEKA machine learning software package developed at the University of Waikato. Detailed documentation of WEKA can be found at `http://www.cs.waikato.ac.nz/~ml/weka`. This data reformatting should save plenty of time in data pre-processing for users who want to evaluate their algorithms using a group of benchmark high-dimentional biomedical data sets.

Let us describe below some of the data sets and their underlying biological or medical problems.

- The breast cancer outcome prediction gene expression data set of van't Veer [98]. The training data contains 78 patient samples, 34 of which are from patients who develop distance metastases within 5 years, the other 44 samples are from patients who remain healthy from the disease after their initial diagnosis for interval of at least 5 years. Correspondingly, there are 12 relapse and 7 non-relapse samples in the testing data set. The number of genes or features is 24481.
- The central nervous system embryonal tumour outcome prediction gene expression data set of Pomeroy [79]. Only the dataset C mentioned in the paper that is used to analyse the outcome of treatment is provided in our repository. In this data set, survivors are patients who are alive after treatment, while the failures are those who succumb to their disease. The data set contains 60 patient samples, 21 are survivors and 39 are failures. There are 7129 genes in the dataset.
- An ovarian disease data set [77] consisting of proteomic profiles of 253 serum samples. The goal of this biomedical application is to identify proteomic patterns in serum that can distinguish between ovarian cancer and non-cancer. This study is significant to women who have a high risk of ovarian cancer due to family or personal history of cancer. The proteomic spectra are generated by mass spectroscopy and the data set provided here is 6-19-02, which includes 91 controls (Normal) and 162 ovarian cancers Another reason to highlight this data set is that the best 10-fold cross-validation accuracy is 100% [59].
- A lung cancer data set [37] from the Brigham and Women's Hospital of the Harvard Medical School. The purpose of this gene expression study is for classification between malignant pleural mesothelioma (MPM) and adenocarcinoma (ADCA) of lung. There are 181 tissue samples, comprising 31 MPM and 150 ADCA. The training set contains 32 samples: 16 MPM and 16 ADCA. The rest 149 samples are used for testing. Each sample is described by 12533 genes. It is highlighted because its training data are class-balanced and small, but the test data are large and contain odd numbers of samples in the two classes.
- The childhood acute lymphoblastic leukemia gene expression data set of Yeoh [105]. This study is about classifying subtypes of childhood acute lymphoblastic leukemia. The data has been divided into six diagnostic groups—BCR-ABL, E2A-PBX1, Hyperdiploid>50, MLL, T-ALL and TEL-AML1—and one that contains diagnostic samples that do not fit into any one of the above groups. There are 12558 genes. According to the above publication, each group of samples has been randomized into training and testing parts. There are 215 training samples and 112 testing samples in total.

- The translation initiation sites prediction data set of Liu and Wong [61]. This data set is converted from sequence data. The original data [76] is a selected set of vertebrates genomic sequences extracted from GenBank. It is used to find the Translation Initiation Site (TIS), at which the translation from mRNA to protein initiates. In total, there are 3312 sequences in the original raw data. There are various ways to build a feature space from sequences. The data set generated by Liu and Wong [61] is produced as follow. A window is centered at each ATG, and spans 100 bases both upstream and downstream from the ATG. So there are 203 bases indicated by A, T, C, and G in each window. A total of 3312 true ATG and 10063 false ATG segments are thus obtained. When building feature space for classification, 3 nucleotides are matched to 1 amino acid and the frequency of each amino acid is counted. The amino acids are distinguished as upstream or downstream based on whether it appears before or after the centered ATG. Besides using single amino acids as features, pairs of amino acids are also used. Thus, the number of features based on amino acids is $(21 + 21 * 21) * 2 = 924$. Furthermore, according to domain knowledge, a true ATG often has G at position 1 of its downstream side, A or G at position 3 of its upstream side, and has no other upstream ATG within a short distance from the centered ATG. These 3 features are included in the feature space, giving 927 features in total.

In this part of the tutorial presentation, we also introduce the basic concepts behind gene expression and proteomic profiling.

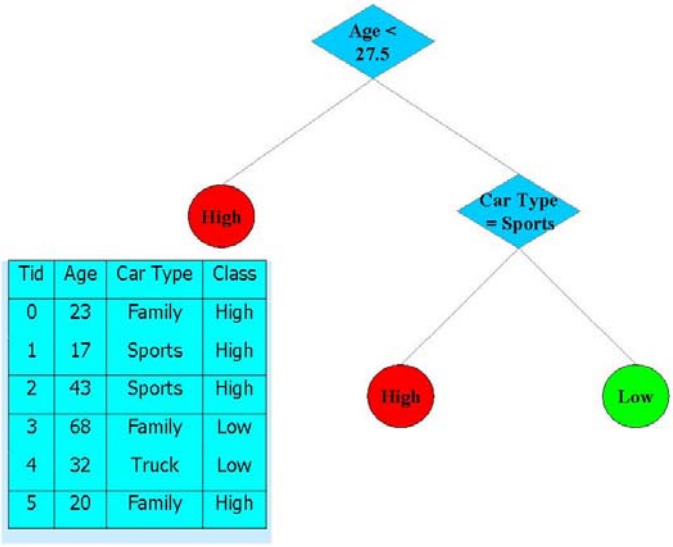
3 A Basic Classification Method Based on Rules

The most popular group of classification techniques is the idea of decision tree induction [14, 80, 82]. These techniques have an important advantage over other machine learning methods—such as k-nearest neighbour [19] and SVM—in the qualitative dimension: rules produced by decision tree induction are easy to understand and hence can help greatly in appreciating the underlying reasons that separate the different classes of samples.

An example of a decision tree is given in Figure 1. The dataset, shown at the lower-left quadrant, consists of 6 training cases, with one numeric attribute (Age), one categorical attribute ($CarType$), and the class that we need to predict. The decision tree mined from this data is shown on the right. Each internal node corresponds to a test on an attribute, whereas the leaf nodes indicate the predicted class. For example, assume we have a new record, $\{Age = 40, CarType = Family\}$, whose class we need to predict. We first apply the test at the root node. Since $Age < 27.5$ is not true, we proceed to the right subtree and apply the second test. Since $CarType = Sports$ is not true, we again proceed to the right, which is a leaf that predicts the label to be Low.

It is also clear that a set of decision or classification rules can be derived from a decision tree. In fact, every path from the root to a leaf gives rise to a rule. For example, the decision tree in Figure 1 yields the following rules, corresponding to the leaves in a left-to-right order:

1. if $Age < 27.5$ then risk is $High$,
2. if $Age \not< 27.5$ and $CarType = Sports$ then risk is $High$, and
3. if $Age \not> 27.5$ and $CarType \neq Sports$ then risk is Low.



Tid	Age	Car Type	Class
0	23	Family	High
1	17	Sports	High
2	43	Sports	High
3	68	Family	Low
4	32	Truck	Low
5	20	Family	High

Figure 1: Example of a decision tree.

Although a large number of algorithms exist for this approach [14, 80, 82], they share two common main steps. The first is the "build tree" step:

1. Start with data at root node.
2. Select an attribute and formulate a logical test on that attribute. This selection is based on the so-called "split-points", which must be evaluated for all attributes.
3. Create a child node on each outcome of the test, and move the subset of examples satisfying that outcome to the corresponding child node.
4. Recursively process each child node. The recursion stops for a child node if it is "pure"—*i.e.*, all examples it contains are from a single class—or "nearly pure"—*i.e.*, most examples it contains are from the same class. The child node at which a recursion stops becomes a leaf of the tree.

The second is the "prune tree" step, which removes subtrees that do not improve classification accuracy and helps avoid over-fitting. More details on tree pruning can be found in [72, 73, 81, 84]. We focus here only on the tree building step.

A visualization of how decision trees split the data is given in Figure 2. Assume that we have 42 points in the two dimensional space shown. The circles indicate the low risk points and the stars the high risk points. Decision trees try to formulate axis-parallel splits to best separate the points into relatively "pure" partitions. One such example is shown in the figure, where we split on $Age < 25$ first and then on $CarType = Sports$.

The critical issue in the "build tree" step is the formulation of a good split test and selection measure for attributes in Substep (2). A general idea is to prefer the simplest hypothesis that fits the data. One concept of simplest "hypothesis" is the so-called "minimum message length" principle.

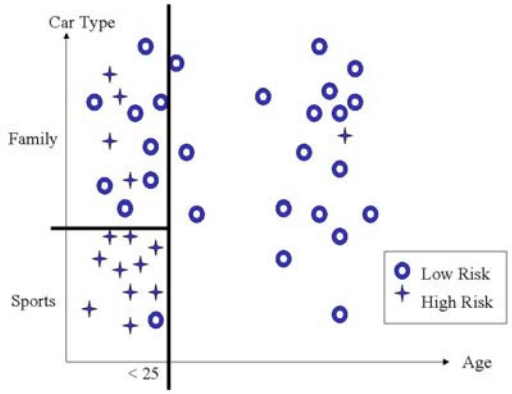


Figure 2: A visualization of axis-parallel decision tree splits.

Definition 3.1 *Given a dataset D. Let there be hypotheses $\mathcal{H} = \{H_1, H_2, ..., H_n\}$ proposed for describing D. The minimal message length principle is to pick the hypothesis H that minimizes message length. That is,*

$$H = argmin_{H_i \in \mathcal{H}}(ML(H_i) + ML(D|H_i))$$

where $ML(\cdot)$ is a measure of message length.

There are a number of ways to define message length, such as Gini index [36], information gain [80], information gain ratio [82], *etc.* Let us illustrate using the Gini index concept familiar from the study of economics. Let $\mathcal{U} = \{C_1, ..., C_k\}$ be all the classes. Suppose we are currently at a node and D is the set of those samples that have been moved to this node. Let f be a feature and $d[f]$ be the value of the feature f in a sample d. Let S be a range of values that the feature f can take. Then the Gini index for f in D for the range S is defined as

$$gini_f^D(S) = 1 - \sum_{C_i \in \mathcal{U}} \left(\frac{|\{d \in D \mid d \in C_i,\ d[f] \in S\}|}{|D|} \right)^2$$

The purity of a split of the value range S of an attribute f by some split-point into subranges S_1 and S_2 is then defined as

$$gini_f^D(S_1, S_2) = \sum_{S \in \{S_1, S_2\}} \frac{|\{d \in D \mid d[f] \in S\}|}{|D|} * gini_f^D(S)$$

In Substep (2) of the "build tree" step, we choose the feature f and the split-point p that minimizes $gini_f^D(S_1, S_2)$ over all possible alternative features and split-points. An illustration is given in Figure 3. This example shows the two best axis-parallel split points on each of the two dimensions in our example: Age and $CarType$. In reality, we have to test all possible split points for both dimensions. When we compare the number of points from each class on either side of the split and then derive the Gini index, we find that the split on the left has a lower value (0.31) and is thus the best overall split. This also corresponds to our intuition, since it is easy to see that the split on Age results in a pure right partition—with the exception on one star—while the split on $CarType$ results in a less pure top partition—5 stars mixed with the majority of circles.

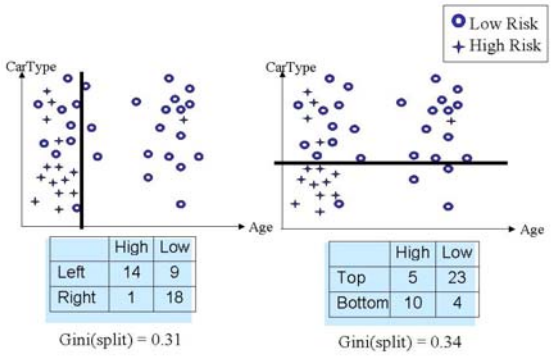


Figure 3: An example of how to find good split-points using Gini index. The chosen split is the vertical one given on the left.

The other two commonly used ideas for Substep (2) of the "build tree" step are information gain and information gain ratio. Let $\mathcal{U} = \{C_1, C_2, ..., C_k\}$ be all the classes of samples. Let f be a feature and S be the range of values that f can take in the samples. Let S be partitioned into two subranges S_1 and S_2. Then the difference between the information needed to identify the class of a sample in $\mathcal{U}$ before and after the value of the feature f is revealed is

$$Gain(f, \mathcal{U}, S_1, S_2) = Ent(f, \mathcal{U}, S_1 \cup S_2) - E(f, \mathcal{U}, \{S_1, S_2\})$$

where

- $Ent(f, \mathcal{U}, S)$ is the class entropy of a range S with respect to a feature f and a collection of classes $\mathcal{U}$. It is defined as

$$Ent(f, \mathcal{U}, S) = -\sum_{C_i \in \mathcal{U}} \frac{|\{d \in C_i \mid d[f] \in S\}|}{|\{d \in \bigcup \mathcal{U} \mid d[f] \in S\}|} * \log_2 \left(\frac{|\{d \in C_i \mid d[f] \in S\}|}{|\{d \in \bigcup \mathcal{U} \mid d[f] \in S\}|} \right)$$

- $E(f, \mathcal{U}, \{S_1, S_2\})$ is the class information entropy of the partition (S_1, S_2). It is defined as

$$E(f, \mathcal{U}, S) = \sum_{S_i \in S} \frac{|\{d \in \mathcal{U} \mid d[f] \in S_i\}|}{|\{d \in \mathcal{U} \mid d[f] \in \bigcup S\}|} * Ent(f, \mathcal{U}, S_i)$$

Then the information gain is the amount of information that is gained by looking at the value of the feature f, and is defined as

$$InfoGain(f, \mathcal{U}) = \max\{Gain(f, \mathcal{U}, S_1, S_2) \mid (S_1, S_2) \text{ is a partitioning of the values of } f \text{ in } \bigcup \mathcal{U} \text{ by some point } T\}$$

We can consider a feature f to be more relevant than a feature f' if $InfoGain(f, \mathcal{U}) > InfoGain(f', \mathcal{U})$. The ID3 decision tree induction algorithm uses it as the measure for picking discriminatory features for tree nodes [80].

However, $InfoGain(f, \mathcal{U})$ tends to favour features that have a large number of values. The information gain ratio is a refinement to compensate for this disadvantage. Let

$$GainRatio(f, \mathcal{U}, S_1, S_2) = \frac{Gain(f, \mathcal{U}, S_1, S_2)}{SplitInfo(f, \mathcal{U}, S_1, S_2)}$$

where $SplitInfo(f, \mathcal{U}, S_1, S_2) = Ent(f, \{\mathcal{U}_f^{S_1}, \mathcal{U}_f^{S_2}\}, S_1 \cup S_2)$, and $\mathcal{U}_f^S = \bigcup_{C_i \in \mathcal{U}} \{d \in C_i \mid d[f] \in S\}$. Then the information gain ratio is defined as

$$InfoGainRatio(f, \mathcal{U}) = \max\{GainRatio(f, \mathcal{U}, S_1, S_2) \mid (S_1, S_2) \text{ is a partitioning of the values of } f \text{ in } \bigcup \mathcal{U} \text{ by some point } T\}$$

The information gain ratio can be used for selecting features that are relevant because we can consider a feature f to be more relevant than a feature f' if $InfoGainRatio(f, \mathcal{U}) > InfoGainRatio(f', \mathcal{U})$. The C4.5 decision tree induction algorithm uses it as the measure for picking discriminatory features for tree nodes [82].

Other algorithms for decision tree induction besides ID3 and C4.5 include CART [14], SLIQ [71], FACT [63], QUEST [62], PUBLIC [84], CHAID [50], ID5 [97], SPRINT [91], and BOAT [34]. This group of algorithms are most successful for analysis of clinical data and for diagnosis from clinical data. Some examples are diagnosis of central nervous system involvement in hematooncologic patients [65], prediction of post-traumatic acute lung injury [83], identification of acute cardiac ischemia [90], prediction of neurobehavioral outcome in head-injury survivors [95], and diagnosis of myoinvasion [64]. More recently, they have even been used to reconstruct molecular networks from gene expression data [93].

4 Advanced Classification Methods Based on Decision-Tree Ensembles

Consider the simple situation where we have three independently constructed classifiers h_1, h_2, and h_3 for predicting whether a new sample is in class C_1 or C_2. Suppose each of these classifiers has an accuracy of 60%. Let $h(t) = \text{argmax}_{C \in \{C_1, C_2\}} |\{h_i \mid h_i \in \{h_1, h_2, h_3\}, h_i(t) = C\}|$. That is, h is a classifier based on a majority vote of h_1, h_2, and h_3. Then given a new sample $t \in C_1$, the probability of $h(t) = C_1$ is

$$\begin{aligned}
Prob(h(t) = C_1) &= Prob(h_1(t) = C_1 \wedge\ h_2(t) = C_1 \wedge\ h_3(t) = C_1) + \\
&\quad Prob(h_1(t) = C_1 \wedge\ h_2(t) = C_1 \wedge\ h_3(t) = C_2) + \\
&\quad Prob(h_1(t) = C_1 \wedge\ h_2(t) = C_2 \wedge\ h_3(t) = C_1) + \\
&\quad Prob(h_1(t) = C_2 \wedge\ h_2(t) = C_1 \wedge\ h_3(t) = C_1) + \\
&= 60\% * 60\% * 60\% + 60\% * 60\% * 40\% + 60\% * 40\% * 60\% + 40\% * 60\% * 60\% \\
&= 64.8\%
\end{aligned}$$

In general, the accuracy of a classifier that is based on a majority vote of several independently constructed component classifiers can give a higher accuracy than all its component classifiers, provided each of these component classifiers have greater than 50% accuracy. In contrast, if these component classifiers have less than 50% accuracy, then their majority vote can lead to a worse

accuracy. For example, if each of h_1, h_2, and h_3 above has accuracy of 40%, then $Prob(h(t) = C_1) = 35.2\%$.

Because of this phenomenon, a number of approaches to create classifiers based on an ensemble of component or base classifiers have been tried. Those that usually use decision tree induction as the base classifier include bagging [12], boosting [31], random forests [13], randomized trees [23], and CS4 [58, 59].

4.1 Bagging

The technique of bagging was introduced by Breiman [12]. Suppose a training data set S having n samples is given. Bagging applies a base classifier—usually C4.5—multiple times to generate a committee $\mathcal{H}$ of classifiers h_1, ..., h_k using bootstrapped versions S_1, ..., S_k of S as follows:

1. Set $i = 1$.
2. Make the i-th bootstrapped version S_i of S by drawing n samples with replacement from S.
3. Make the i-th decision tree h_i by applying C4.5 or other classifier construction method to S_i.
4. Increment i, goto Step (2), until $i > k$.

Then given a new test sample T, each of h_i is applied, and the class that is predicted most often is assigned to T. In other words, the final classifier is defined as follows, where $\mathcal{U} = \{C_1, ..., C_r\}$ are the classes:

$$bagged(T) = \text{argmax}_{C_j \in \mathcal{U}} |\{h_i \in \mathcal{H} \mid h_i(T) = C_j\}|$$

Some recent studies have shown that bagging is a useful approach to the analysis of biomedical data such as gene expression data clustering [26], classification of ovarian cancer using mass spectrometry data [103], and automated classification of fluorescence microscope images for location proteomics [43].

4.2 Boosting

Similar to bagging, boosting also uses a committee of classifiers for classification by voting. Here, the construction of the committee of classifiers is different. While bagging builds the individual classifiers separately, boosting builds them sequentially such that each new classifier is influenced by the performance of those built previously. In this way, those data incorrectly classified by previous models can be emphasized in the new model, with an aim to mold the new model to an "expert" for classifying those hard instances. A further difference between the two committee techniques is that boosting weights the individual classifiers' output depending on their performance, while bagging gives equal weights to all the committee members. AdaBoost.M1 [31] is a very good example of the boosting idea.

There are two ways that AdaBoost.M1 manipulates the weights to construct a new training set S_i. One way is called "boosting by sampling", where samples are drawn with replacement from the original data set S with probability proportional to their weights. Another way is called "boosting by

weighting", where the sample weights change the error calculation of the i-th tree classifier h_i—the sum of the weights of the misclassified samples divided by the total weight of all samples, instead of the fraction of samples that are misclassified. The AdaBoost.M1 training process is as follows:

1. Set $i = 1$.
2. Make training set S_i by assigning equal weight $1/n$ to each sample in S, where n is size of S.
3. Let $w_i(x)$ denote the weight of sample x in S_i. Note that $\sum_{x \in S_i} w_i(x) = 1$.
4. Make classifier h_i by applying C4.5 or other classifier construction method to S_i.
5. Let X_i be the samples in S_i misclassified by h_i, and let Y_i be the samples in S_i correctly classified by h_i.
6. Calculate errors $e_i = \sum_{x \in X_i} w_i(x)$. Note that $1 - e_i = \sum_{y \in Y_i} w_i(y)$, because $\sum_{x \in X_i} w_i(x) + \sum_{y \in Y_i} w_i(y) = 1$.
7. Stop if $e_i = 0$.
8. Stop if $e_i > 0.5$.
9. Adjust weights so that the error rate of h_i under weights $w_{i+1}(\cdot)$ is exactly $1/2$. This is achieved by
$$w_{i+1}(x) = \begin{cases} \dfrac{w_i(x)}{2 * e_i} & \text{if } x \in X_i \\ \dfrac{w_i(x)}{2 * (1 - e_i)} & \text{if } x \in Y_i \end{cases}$$
Note that the above weight adjustment formula is equivalent to generating weights $w_{i+1}(x)$ by multiplying the weights $w_i(x)$ of samples that h_i classifies correctly by the factor $\beta_i = e_i/(1-e_i)$ and then renomalizing so that $\sum_{x \in X_i} w_{i+1}(x) + \sum_{y \in Y_i} w_{i+1}(y) = 1$. To see this, let α_i be the normalization factor. Thus, $\sum_{x \in X_i} w_{i+1}(x) + \sum_{y \in Y_i} w_{i+1}(y) = 1$ means $\sum_{x \in X_i} \alpha_i * w_i(x) + \sum_{y \in Y_i} \alpha_i * \beta_i * w_i(y) = 1$. This rewrites to $\alpha_i = 1/(\sum_{x \in X_i} w_i(x) + \beta_i * \sum_{y \in Y_i} w_i(y))$. By definition, $\sum_{x \in X_i} w_i(x) = e_i$ and $\sum_{y \in Y_i} w_i(y) = 1 - e_i$. Thus, $\alpha_i = 1/(e_i + \beta_i * (1 - e_i))$. Substituting the definition of β_i into this formula, we get $\alpha_i = 1/(2 * e_i)$. Thus $w_{i+1}(x) = \alpha_i * w_i(x) = w_i(x)/(2 * e_i)$ for $x \in X_i$ and $w_{i+1}(x) = \alpha_i * \beta_i * w_i(x) = w_i(x)/(2 * (1 - e_i))$ for $x \in Y_i$, as given in the adjustment formula above.
10. Increment i, goto (4), until $i = k$.

Then to classify a new sample T, we do

$$boosting(T) = \text{argmax}_{C_j \in \mathcal{U}} \sum_{h_i \in \mathcal{H}, h_i(T) = C_j} \log\left(\frac{1 - e_i}{e_i}\right)$$

Note that if $e_i = 0$ in Step (6), the construction of the ensemble of decision is stopped. In some training data sets, this can happen in the first iteration—*i.e.*, $e_1 = 0$—and the ensemble ends up with only 1 decision tree. In such a situation, the result of boosting is the same decision tree as C4.5.

Recent applications of the boosting idea to bioinformatics can be found at [11, 22, 43, 59, 96, 103].

4.3 Random Forest and Randomization Trees

The technique of random forest [13] is closely related to that of bagging. The only key difference is that when we are constructing the i-th decision in the ensemble, the choice of the features to be considered for splitting a node is restricted to a subset of the m_{try} features randomly selected for that node. The procedure for constructing a committee $\mathcal{H}$ of classifiers h_1, ..., h_k using bootstrapped versions S_1, ..., S_k of S is as follows:

1. Let m_{try} be a small integer.
2. Set $i = 1$.
3. Make a new training set S_i by drawing n samples with replacement from S, where n is the size of S.
4. Make a decision tree h_i, where at each node, m_{try} features are selected randomly, and the best split for the node is chosen one among them.
5. Increment i, goto (3), until $i = k$.

Then to classify a new sample T, we do

$$randomforest(T) = \text{argmax}_{C_i \in \mathcal{U}} |\{h_i \in \mathcal{H} \mid h_i(T) = C_i\}|$$

While the parameter m_{try} implies some additional amount of tuning needed to get higher classification accuracy out of a random forest classifier, in practice it is not very sensitive to the value of m_{try}. It is also believed to be more robust to outliers and noise, faster than bagging and boosting, and is simple and easy to parallelize [13].

Recent applications of random forests in bioinformatics can be found at [44, 88, 103].

The randomization decision trees [23] is a modified version of the C4.5 learning algorithm. Basically, the decision about which split to introduce at each internal node of the tree is randomized. With a different random choice, a new tree is then constructed. Twenty of the best splits, in terms of information gain ratio, are considered to be the pool of random choices. Note that the 20 best splits may all involve splitting on the same feature. Consequently, every member of a committee of randomized trees always shares the same root nodes and features—the only difference between the members is their different splitting points.

4.4 CS4

This subsection describes CS4 (cascading and sharing for decision trees)—our new method to discover significant rules using the concept of tree committees—and to use the rules as an ensemble for reliable predictions. Let $\mathcal{U}$ be a training data set having two classes of samples, *positive* and *negative*. We use the steps below to iteratively derive k trees from $\mathcal{U}$. Here, k is significantly less than the number of features used in $\mathcal{U}$, and usually we set k as 20:

1. Use the information gain ratio measure to rank all the features into an ordered list with the best feature at the first position.

2. Set $i = 1$.

3. Use the i-th feature as the root node to construct the i-th tree.

4. Increase i by 1 and goto Step 3, until $i = k$.

Then rules can be directly generated from these trees by depth-first traversals as usual. In order to identify significant rules, we just rank all the rules according to each rule's coverage in the training data set $\mathcal{U}$. The top-ranked ones are significant. The significant rules may then be used for understanding possible interactions between the features involved in these rules.

Next, we describe how to use these rules for class prediction. We call the k trees generated during the training process above a committee or an ensemble of decision trees. Now let a new test sample T be given. Clearly, each of the k trees in the committee has a specific rule to tell us a predicted class label for this test sample. Denote these k rules from the tree committee as:

$$rule_1^{pos}, rule_2^{pos}, \cdots, rule_{k_1}^{pos},$$
$$rule_1^{neg}, rule_2^{neg}, \cdots, rule_{k_2}^{neg}.$$

Here, $k_1 + k_2 = k$. Each of $rule_i^{pos}$ predicts T to be in the positive class. Each of $rule_i^{neg}$ predicts T to be in the negative class. Sometimes, the k predictions can be unanimous—*i.e.*, either $k_1 = 0$ or $k_2 = 0$. In these situations, the predictions from all the k rules agree with one another, and the final decision is obvious. Oftentimes, the k decisions are mixed with either a majority of positive classes or a majority of negative classes. In these situations, we use the following formulas to calculate two classification scores based on the coverages of these rules:

$$\begin{aligned} Score^{pos}(T) &= \sum_{i=1}^{k_1} coverage(rule_i^{pos}) \\ Score^{neg}(T) &= \sum_{i=1}^{k_2} coverage(rule_i^{neg}) \end{aligned}$$

If $Score^{pos}(T)$ is larger than $Score^{neg}(T)$, we assign the positive class to the test sample T. Otherwise, T is predicted as negative. By using the rules' coverage as weights, we avoid the pitfalls of simple equal voting adopted by bagging [12]. Our weighting policy allows the tree committee to automatically distinguish the contributions from the minor rules and from the significant rules in the prediction process.

It is quite easy to generalize this method for multi-class problems. In this situation, the classification score for a specific class, say class $C_j \in \mathcal{U}$, is calculated as:

$$Score^{C_j}(T) = \sum_{i=1}^{k_{C_j}} coverage(rule_i^{C_j})$$

The class that receives the highest score is then predicted as the test sample's class. In other words, the resulting CS4 classifier is:

$$cs4(T) = \text{argmax}_{C_j \in \mathcal{U}}\ Score^{C_j}(T)$$

CS4 differs from traditional committee classifiers—such as bagging and boosting—in the management of the original training data. Bagging and boosting generate bootstrapped or modified training

data for every iteration's construction of trees. CS4 uses the original training data throughout the whole training process. As a result, the rules produced by CS4 always reflect precisely the nature of the original data. In contrast, because of the use of bootstrapped training data, some bagging or boosting rules may not be true when applied to the original training data.

CS4 also differs from randomized decision trees [23]. By construction, every member of a committee of randomized trees may share the same nodes and features. The only difference between the members is their different splitting points. In contrast, the trees in a CS4 committee differs from one another not only at splitting points but also at features. Thus CS4 has a much larger potential for diversity than randomized trees.

5 Other Classification Methods

Let us also introduce a few non-rule-based classifiers—such as support vector machines [99], Naive Bayes [25], k nearest neighbours [19], and hidden Markov models [53]—so that we can better appreciate the advantages of rule-based classification methods for analysing biomedical data.

5.1 K Nearest Neighbour Classifier

In the preceeding sections, we considered classification prediction from the perspective of first constructing a prediction model from training data and then using this model to assign class labels to new samples. There is another perspective to classification prediction that is quite different where no prediction model is constructed beforehand, and every new sample is assigned a class label in an instance-based manner.

Representatives of this perspective of classification prediction include the "k Nearest Neighbors" classifier (kNN) [19], which is particularly suitable for numeric feature vectors; the "Decision making by Emerging Patterns" classifier (DeEPs) [57], which is more complicated but also works well on categorical feature vectors; and other classifiers [1]. We describe the simplest of these classifiers—kNN—in this subsection.

The kNN classification technique to assign a class to a new example x is as follows:

1. Find k nearest neighbours of x in the existing dataset, according to some distance or similarity measure. That is, we compare the new sample x to all known samples in the existing dataset and determine which k known samples are most similar to x.
2. Determine which class C_i is the class to which most of those k known samples belong.
3. Assign the new sample x to the class C_i.

A distance measure between two samples x and y that is commonly used with kNN is the Euclidean distance $\sqrt{\sum_f (x[f] - y[f])^2}$, where f ranges over the features of the samples. The working of kNN classifier is illustrated in Figure 4

As mentioned earlier, it is a characteristic of kNN that it locates some training instances or their prototypes in the existing training dataset without any extraction of high-level patterns or rules.

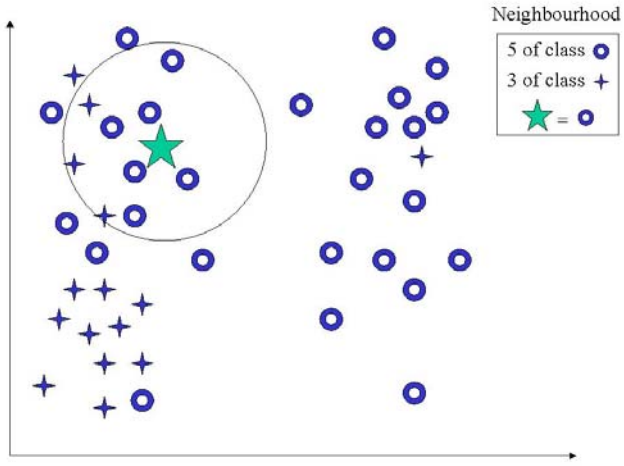


Figure 4: This diagram illustrates the working of the kNN classifier. The new sample is the large "star". We consider $k = 8$ nearest neighbours. As can be seen in the diagram, 5 of the neighbours are in the "circle" class and 3 are in "cross" class. Hence the new sample is assigned to the "circle" class.

Such a classifier that is based solely on distance measure may be insufficient for certain types of applications that require a comprehensible explanation of prediction outcomes. Some in-road to this shortcoming has been made in more modern instance-based classifiers such as DeEPs [57].

Instead of focusing on distance, DeEPs focuses on the so-called emerging patterns [24]. Emerging patterns are regular patterns contained in an instance that frequently occurs in one class of known data but that rarely—or even never—occurs in all other classes of known data. DeEPs then makes its prediction decision for that instance based on the relative frequency of these patterns in the different classes. These patterns are usually quite helpful in explaining the predictions made by DeEPs.

5.2 Support Vector Machines

Let $\phi : D \to D'$ be a function that embeds a training sample in the input space D to a higher dimensional embedding space D'. A kernel $k(x, y) = (\phi(x) \cdot \phi(y))$ is an inner product in D'. It can often be computed efficiently. By formulating a learning algorithm in a way that only requires knowledge of the inner products between points in the embedding space, one can use kernels without explicitly performing the embedding ϕ.

A support vector machine is one such learning algorithm. It first embeds its data into a suitable space and then learns a discriminant function to separate the data with a hyperplane that has maximum margin from a small number of critical boundary samples from each class [16, 99]. A SVM's discriminant function $svm(T)$ for a test sample T is a linear combination of kernels computed at the training data points and is constructed as

$$svm(T) = sign\left(\sum_i \alpha[i] * Y[i] * k(T, X[i]) + b\right)$$

where $X[\_]$ are the training data points, $Y[\_]$ are the class labels (which are assumed to have been mapped to 1 or –1) of these data points, $k(\cdot,\cdot)$ is the kernel function, and b and $\alpha[\_]$ are parameters to be learned from the training data.

As it turns out, the training of a SVM is a quadratic programming problem on maximizing the Lagrangian dual objective function

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{l} \alpha[i] - \frac{1}{2} * \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha[i] * \alpha[j] * Y[i] * Y[j] * k(X[i], X[j])$$

subject to the constraint that $\forall i.0 \leq \alpha[i] \leq C$ and $\sum_{i=1}^{l} \alpha[i] * Y[i] = 0$. Here, C is a bound on errors; $X[i]$ and $X[j]$ are the ith and jth training samples; $Y[i]$ and $Y[j]$ are the corresponding class labels, which are mapped to 1 and –1; and $k(\cdot,\cdot)$ is the kernel function. The kernel function normally used is a polynomial kernel

$$k(x, x') = (x \cdot x')^d = (\sum_i x[i] * x'[i])^d$$

In the popular WEKA data mining software package [101], the quadratic programming problem above is solved by sequential minimal optimization [78]. Once the optimal solution $\alpha[\_]$ is obtained from solving the quadratic programming problem above, we can substitute it into the SVM decision function $svm(T) = sign(\sum_i \alpha[i] * Y[i] * k(T, X[i]) + b)$ given earlier.

It remains now to find the threshold b. We can estimate b using the fact that the optimal solution $\alpha[\_]$ must satisfy the so-called Karush-Kuhn-Tucker conditions [38, 78, 89]. In particular, for each $\alpha[j] > 0$, $Y[j] * svm(X[j]) = 1$. Hence $Y[j] = sign(\sum_i \alpha[i] * Y[i] * k(X[j], X[i]) + b)$. So we estimate b by averaging $Y[i] - \sum_i \alpha[i] * Y[i] * k(X[j], X[i])$ for all $\alpha[j] > 0$.

A SVM is largely characterized by the choice of its kernel function. Thus SVM connects the problem they are designed for to a large body of existing research on kernel-based methods [16, 85, 99]. Besides the polynomial kernel function $k(x, y) = (x \cdot y)^d$, other examples of kernel function include the Gaussian radial basis kernel function [7, 99], sigmoid kernel function [89], B_n-spline kernel function [89], locality-improved kernel function [106], and so on.

Some recent applications of SVM in the biomedical context include protein homology detection [46], microarray gene expression data classification [15], breast cancer diagnosis [33, 68], protein translation initiation site recognition [106, 107], as well as prediction of molecular bioactivity in drug design [100].

5.3 Naive Bayesian Classifier

Another important group of classification techniques [6, 25, 40, 47, 48, 56, 74, 86] are based on the Bayes theorem. The theorem states that

$$P(h|d) = \frac{P(d|h) * P(h)}{P(d)}$$

where $P(h)$ is the prior probability that a hypothesis h holds, $P(d|h)$ is the probability of observing data d given some world that h holds, and $P(h|d)$ is the posterior probability that h holds given the observed data d.

Let H be all the possible classes. Then given a test instance with feature vector $\{f_1 = v_1, ..., f_n = v_n\}$, the most probable classification is given by

$$\text{argmax}_{h_j \in H} P(h_j | f_1 = v_1, \ldots, f_n = v_n)$$

Using the Bayes theorem, this is rewritten to

$$\text{argmax}_{h_j \in H} \frac{P(f_1 = v_1, \ldots, f_n = v_n | h_j) * P(h_j)}{P(f_1 = v_1, \ldots, f_n = v_n)}$$

Since the denominator is independent of h_j, this can be simplified to

$$\text{argmax}_{h_j \in H} P(f_1 = v_1, \ldots, f_n = v_n | h_j) * P(h_j)$$

However, estimating $P(f_1 = v_1, \ldots, f_n = v_n | h_j)$ accurately may not be feasible unless the training set is sufficiently large.

One of the popular ways to deal with the situation above is that adopted by the Naive Bayes classification method (NB) [25, 47]. NB assumes that the effect of a feature value on a given class is independent of the values of other features. This assumption is called class conditional independence. It is made to simplify computation and it is in this sense that NB is considered to be "naive". Under this class conditional independence assumption,

$$\text{argmax}_{h_j \in H} P(f_1 = v_1, \ldots, f_n = v_n | h_j) * P(h_j)$$

$$= \text{argmax}_{h_j \in H} \prod_i P(f_i = v_i | h_j) * P(h_j)$$

where $P(h_j)$ and $P(f_i = v_i | h_j)$ can often be estimated reliably from typical training sets.

Some example applications of Bayesian classifiers in the biomedical context are mapping of locus controlling a genetic trait [35], screening for macromolecular crystallization [41], classification of cNMP-binding proteins [69], prediction of carboplatin exposure [45], prediction of prostate cancer recurrence [21], prognosis of femoral neck fracture recovery [55], prediction of protein secondary structure [3, 51, 94], and reconstruction of molecular networks [32].

5.4 Hidden Markov Model Classifier

Related to Bayesian classifiers are hidden Markov models (HMM) [6, 27, 28, 53]. A HMM is a stochastic generative model for sequences defined by a finite set S of states, a finite alphabet A of symbols, a transition probability matrix T, and an emission probability matrix E. The system moves from state to state according to T while emitting symbols according to E. In an n-th order HMM, the matrices T and E depend on all n previous states. Thus, for a first order HMM, given a sequence of emissions $X = x_1, x_2, ...$, and states $S = s_1, s_2, ...$, we can compute the probability of this sequence as

$$Prob(X, S) = \prod_i Prob(x_i | s_i) = \prod_i E(x_i | s_i) * T(s_{i-1}, s_i)$$

Typically, the sequence of emissions X is given, and we use an algorithm such as Viterbi [49] to obtain the sequence of states S such as that $S = \text{argmax}_S Prob(X, S)$. In the situation where the emission or the transition matrices are not fully known, we use an algorithm such as Baum-Welch [20, 70].

In order to appreciate HMMs, let us use the example of a dishonest casino told to us by Ken Sung. Suppose the casino has a fair dice, where $P(i) = 1/6$ for $i = 1, ..., 6$; and a loaded dice, where

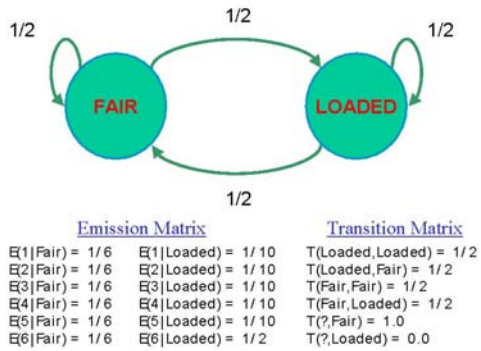


Figure 5: A visualization of the dishonest casino model.

$P(i) = 1/10$ for $i = 1, ..., 5$, and $P(6) = 1/2$. The casino player switches between the fair and loaded dice with probability $1/2$, and initially the dice is always fair. This set up is shown in Figure 5.

Now suppose a game is as follows: you bet \$1, you roll, the casino player rolls, the person with the highest number wins \$2. Given a sequence of rolls, we would like to know how probable is the sequence. For example, suppose we have played 2 games and the sequence of rolls is $X = 1, 6, 2, 6$. Then

$$\begin{aligned} Prob(X, S = Fair, Fair, Fair, Fair) &= E(1|Fair) * T(?, Fair) * \\ &\quad E(6|Fair) * T(Fair, Fair) * \\ &\quad E(2|Fair) * T(Fair, Fair) * \\ &\quad E(6|Fair) * T(Fair, Fair) \\ &= \frac{1}{6} * 1 * \frac{1}{6} * \frac{1}{2} * \frac{1}{6} * \frac{1}{2} * \frac{1}{6} * \frac{1}{2} \\ &= 9.6451 * 10^{-5} \end{aligned}$$

and

$$\begin{aligned} Prob(X, S = Fair, Loaded, Fair, Loaded) &= E(1|Fair) * T(?, Fair) * \\ &\quad E(6|Loaded) * T(Fair, Loaded) * \\ &\quad E(2|Loaded) * T(Loaded, Fair) * \\ &\quad E(6|Loaded) * T(Fair, Loaded) \\ &= \frac{1}{6} * 1 * \frac{1}{2} * \frac{1}{2} * \frac{1}{6} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} \\ &= 8.6806 * 10^{-4} \end{aligned}$$

In fact, $S = Fair, Loaded, Fair, Loaded$ is the most likely sequence of states corresponding to the rolls 1, 6, 2, 6. It is very likely that the casino player has cheated.

HMMs have been applied to a variety of problems in sequence analysis, including protein family classification and prediction [5, 8, 54], tRNA detection in genomic sequences [66], methylation guide

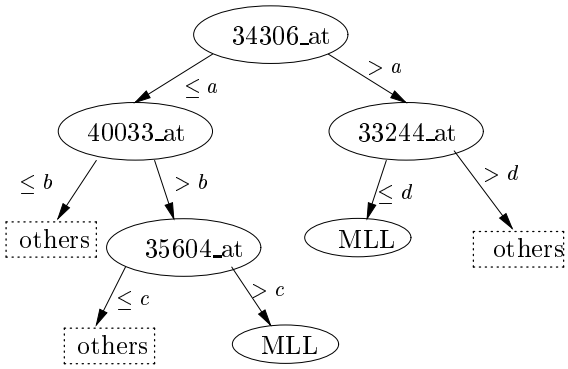


Figure 6: A decision tree induced by C4.5 from gene expression dataset for differentiating the subtype MLL against other subtypes of childhood leukemia. Here $a = 13683.6, b = 3691.4, c = 986.9, d = 846.6$.

snoRNA screening [67], gene finding and gene structure prediction in DNA sequences [4, 9, 10, 53, 87], protein secondary structure modeling [30], and promoter recognition [75, 104].

6 Some Interesting Results

In this section, we give examples of decision trees induced from high-dimensional biomedical data to see the power of decision trees for concisely characterizing the massive data. We also give detailed results of CS4 when running on two gene expression datasets. Finally in this section, we will give performance comparison between single decision trees, bagging, boosting, CS4, SVM, and kNN.

6.1 Examples of decision trees

To demonstrate the typical structure of decision trees and to show where are built-in features in a tree, we present an example decision tree induced by C4.5 [82] from a gene expression profiling dataset. The dataset consists of 215 two-class training samples—14 MLL plus 201 OTHERS—for differentiating the MLL subtype from the other subtypes of childhood leukemia disease [105]. The data are described by 12558 features. Each feature is a gene having continuous expression values. The structure of this tree is depicted in Figure 6. Observe that there are only 4 built-in features in this tree residing at the 4 non-leaf nodes—namely 34306_at, 40033_at, 33244_at, and 35604_at. Compared to the total 12558 input features, the 4 built-in features is a very tiny fraction. Each of the five leaf nodes corresponds to a rule, the rule's predictive term is the class label contained in the leaf node. As an example, the rule at the most-left side of the tree is:

$$\text{if } 34306\_at \leq 13683.6 \text{ and } 40033\_at \leq 3691.4,$$
$$\text{then this sample is OTHERS.}$$

Consequently, the interpretation of the decision results by this tree classifier needs only some of the 4 built-in features rather than the entire set of 12588 features. To see this clearly, we decompose

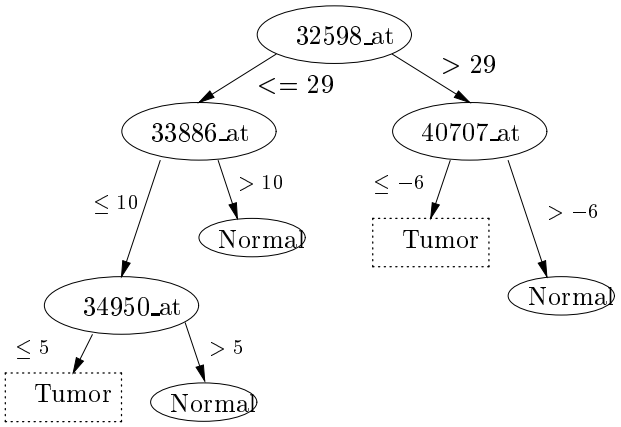


Figure 7: A decision tree induced from the prostate disease data set of gene expression profiles of 102 cells.

the tree into 5 rules and use the rules to establish a function as follows.

$$f(x_1, x_2, x_3, x_4) = \begin{cases} -1 & \text{if } x_1 \le a,\ x_2 \le b \\ -1 & \text{if } x_1 \le a,\ x_2 > b,\ x_3 \le c \\ 1 & \text{if } x_1 \le a,\ x_2 > b,\ x_3 > c \\ 1 & \text{if } x_1 > a,\ x_4 \le d \\ -1 & \text{if } x_1 > a,\ x_4 > d \end{cases}$$

where x_1, x_2, x_3, and, x_4 represent 34306_at, 40033_at, 33244_at, and 35604_at, respectively; $a = 13683.6$, $b = 3691.4$, $c = 986.9$, $d = 846.6$; the two values of this function -1 and 1 represent OTHERS and MLL respectively. Given a test sample, at most 3 of the 4 built-in genes' expression values are needed to determine $f(x_1, x_2, x_3, x_4)$. If the function value is -1, then the test sample is predicted as OTHERS, otherwise it is predicted as MLL.

This example tells us that though gene expression data are high-dimensional, there indeed exist many simple rules containing only 2 or 3 features that can be used to *concisely* characterize the data. These simple rules also have much potential to be translated into clinical knowledge.

As another example, the decision tree shown in Figure 7 is induced by C4.5 from a prostate disease data set that comprises expression profiles from 52 tumor cells and 50 normal cells [92]. There are two significant rules in this tree: One is

$$\text{if } 32598\_at \le 29 \text{ and } 33886\_at \le 10 \text{ and } 34950\_at \le 5,$$
$$\text{then this is a tumor cell.}$$

This rule is a significant rule with a coverage of 94% (49/52) in the tumor class. The other rule is dominant in the normal class:

$$\text{if } 32598\_at > 29 \text{ and } 40707\_at > -6,$$
$$\text{then this is a normal cell.}$$

This rule is significant with an 82% (41/50) coverage in the normal class.

Class	# of Training Samples	# of Test Samples	# of Features
Hyperdip>50	42	22	12558
Others	52	27	12558

Figure 8: Training and test data for differentiation between the subtype Hyperdip>50 and some other subtypes of childhood leukemia [105].

Tree No.	1	2	3	4	5	**6**	7	8	9	10
Training Errors	0	3	1	2	1	**0**	1	1	1	1
Test Errors	13	9	7	12	12	**5**	8	11	9	7
Tree No.	11	12	13	14	15	16	17	18	19	20
Training Errors	1	0	3	2	1	2	2	1	2	1
Test Errors	14	8	12	10	6	6	9	8	8	8

Figure 9: The training and test errors of our 20 single cascading decision trees on the Hyperdip>50 vs Others data set that consist of 94 training and 49 test samples.

6.2 The performance of CS4

We illustrate four main ideas of CS4 using a real example: (1) whether top-ranked features have similar gain ratios; (2) whether our cascading trees have similar training performance; (3) whether they have similar, but perhaps not perfect accuracy on test data; and (4) whether their combinations converge to the true optimal function—that is, whether the expanding tree committees reduce the test errors gradually.

The data used here are a collection of gene expression profiles of 143 cells used to differentiate the subtype Hyperdip>50 against some other subtypes of the childhood leukemia disease [105]. There are 12558 features describing the data profiles. Details about the 94 training samples and 49 test samples are shown in Figure 8.

The gain ratios of the 20 top-ranked features in this data set are close to each other. The first 10 features' gain ratios are: 0.39, 0.36, 0.35, 0.33, 0.33, 0.33, 0.33, 0.32, 0.31, and 0.30; The second 10 features' gain ratios are: 0.30, 0.30, 0.30, 0.29, 0.29, 0.28, 0.28, 0.28, 0.28, and 0.28. The biggest difference of the ratios, between the first and the 20th feature, is only 0.11—in fact, these two features' partitionings are different in only a few samples. Naturally, we should try each of them as root nodes to construct decision trees.

Using the cascading idea on these top-ranked features, we constructed a committee of 20 trees. Figure 9 shows the training errors—*i.e.*, the number of samples wrongly classified over the training data—and test errors—*i.e.*, the number of samples wrongly classified over the test data—of the 20 trees.

Clearly, the performance of these trees is not perfect and fluctuates very much even on the training samples. So, each of them is far from the true optimal function. In particular, observe that the first tree, which is rooted by the best feature, makes no training errors but 13 test errors. However, the 6th tree, which is *not* rooted by the best feature, makes no training errors but only 5 test errors. This observation supports the following two points:

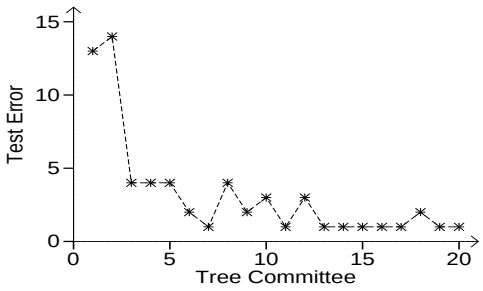


Figure 10: The performance trend when the committee is expanded by adding our cascading trees one by one. The test error becomes almost stable at the level of one mistake after the 13th tree is added into the committee.

1. The first tree does not always have the best performance.
2. Alternative trees rooted by other top-ranked features may have better performance than the first tree.

In fact, from our previous experience, this phenomenon occurs quite often in bio-medical data. We randomly examine ten pairs of our first and second trees, and find 4 pairs where the first tree wins, 3 pairs where the second tree wins, and 3 pairs where the two trees tie in performance. This is also a reason why the cascading idea is a good idea to construct tree committees.

Next, we combine the trees to form committees and to see whether there is a convergence path of test errors. When combining the first two trees, the committee makes 14 mistakes on the test data. After adding the third, the test errors are reduced to 4. Sequentially adding one by one until the 20th tree, the corresponding tree committees make 4, 4, ..., 1, and finally only one mistake on the 49 test samples. Note also that all the committees make no mistake on the training data. Figure 10 shows the error reduction curve when the tree committees are expanded by adding our cascading trees, indicating a perfect convergence route to the true function.

In comparison, bagging and boosting make 2 and 13 test errors respectively on this data set. In this case, bagging also significantly reduces the error numbers of the single decision trees, just like our method. But we will show later that our overall accuracy is better than bagging on a wide range of datasets. Boosting makes the same number of mistakes as the single C4.5 tree—a poor performance of 13 errors. This is because the first tree of the boosting committee makes no mistakes on the training data, causing the algorithm to stop generating other base classifiers. As a result, this boosting committee is a singleton, unable to take real committee power.

6.3 Performance Comparison

In this section, we present a performance comparison between six classifiers: SVM, NB, kNN, CS4, C4.5 Bagging, and C4.5 Boosting on 15 data sets. The data sets are listed in Figure 11. The performance is measured by error numbers. An error number of a classifier is the number of samples that are wrongly classified by the classifier. We give the performance of the classifiers when different features of a data set are used. These include the case of selecting the whole feature space, the case of selecting all entropy-discretized features [60], and the case of all mean-entropy discretized features [60]. An *entropy-discretized feature* is defined as a feature that is discretized into two

Data sets	# classes	# samples	# features	Usage
T-ALL	2	327	12558	Subtype distinction of leukemia
E2A-PBX1	2	327	12558	Subtype distinction of leukemia
TEL-AML1	2	327	12558	Subtype distinction of leukemia
BCR-ABL	2	327	12558	Subtype distinction of leukemia
MLL	2	327	12558	Subtype distinction of leukemia
Hyperdip>50	2	327	12558	Subtype distinction of leukemia
Ovarian cancer	2	253	15154	Ovarian disease diagnosis
Prostate cancer	2	102	12600	Prostate disease diagnosis
Colon tumor	2	62	2000	Colon tumor diagnosis
ALL-AML	2	72	7129	Subtype classification of leukemia
Subtype lymphoma	2	47	4026	Subtype classification of lymphoma
Stjude testing	2	112	12558	Subtype classification of leukemia
Lung cancer testing	2	149	12533	Lung cancer diagnosis
ALL-AML testing	2	34	7129	Subtype classification of leukemia
Armstrong testing	3	15	12582	Subtype classification of leukemia

Figure 11: Data set description. The total number of samples for each of the first 11 data sets are shown in the 3rd column and for 10-fold cross-validation. For the remaining 4 data sets, only test samples are shown here.

or more intervals by an entropy-discretization method [29]. Similarly, mean-entropy discretized features are features whose entropy value is *smaller* than the mean entropy value of all the entropy-discretized features. Usually, entropy-discretized features consist of only 5%–10% of the original features; and mean-entropy discretized features consist of only 2%–3% of the original features for different datasets [60].

Figure 12 reports the classification errors of SVM, NB, and kNN on the 15 data sets when the three scenarios—all the original features, all the entropy-discretized features, and only the mean-entropy discretized features—are used in the 10-fold cross-validation, except the last 4 data sets in Figure 12. We can see that:

- SVM makes much less number of mistakes after the feature selections—in the first round or after the second round—on the data sets BCR-ABL, MLL, Subtype lynphoma, Stjude testing, and ALL-AML testing. On the other data sets, SVM maintains its performance. From the first round to the second round of feature selection, SVM often decreases or maintains the errors.
- NB almost constantly improves its performance very much on most of the data sets. In general, the trend of the error numbers goes to a lower level when less features are used.
- kNN also improves its performance very much on most of the data sets. In general, the trend of the error numbers goes to a lower level when less features are used.

Figure 13 reports the error change trend of three rule-based committee classifiers—our CS4 classifier, C4.5 Bagging, and C4.5 Boosting. We can see that:

- The three committee classifiers do not change as much in their performance, before and after

Data sets	SVM			NB			kNN		
	all	entropy	M-entropy	all	entropy	M-entropy	all	entropy	M-entropy
T-ALL	1	0	0	29	0	0	8	3	0
E2A-PBX1	1	1	1	25	0	0	1	1	1
TEL-AML1	4	3	4	62	6	4	14	4	4
BCR-ABL	12	8	8	15	20	14	15	9	10
MLL	7	5	2	19	5	3	9	5	4
Hyperdip>50	11	9	11	57	15	17	21	13	16
Ovarian	0	0	0	19	16	14	15	11	10
Prostate	7	8	6	40	15	8	18	10	8
Colon Tumor	11	9	8	25	18	14	19	9	11
ALL-AML	1	2	2	2	0	2	10	2	1
Subtype lymphoma	6	3	2	10	3	3	13	5	5
Stjude testing	32	1	2	82	14	16	20	5	2
Lung cancer	1	1	0	7	2	2	3	1	1
ALL-AML testing	5	1	1	3	1	1	10	6	2
Armstrong	0	0	0	0	0	0	2	2	2
Total Errors	99	51	47	395	115	98	178	85	77

Figure 12: The change trend of the error numbers of SVM, NB, and kNN after feature selections. Here, "all" represents a classifier considering all the features; "entropy" represents the entropy discretized features; and "M-entropy" represents the mean-entropy discretized features.

feature selection. Nonetheless, there is still a slight decrease in total errors after the feature selections; see the last row of Figure 13. Such a trend is different from the changes occurred in SVM, NB, and kNN, where the total errors are significantly reduced after features selections; see the last row of Figure 12. This interesting phenomenon is an issue about the stability of classifiers in relation to feature selections, and also an issue about wrapped features in a classifier.

- Compared to bagging and boosting, our newly proposed CS4 committee classifier outperforms them with only one exception case on the Hyperdip>50 data set, where boosting makes 14 errors, but CS4 makes 15 errors.

From both Figure 12 and Figure 13, we can see that feature selections have constantly helped the classifiers to improve their performance, even though only 2–10% of the original features are used.

Next, we discuss which classifier wins the best performance.

- If using the whole feature space without any selection, CS4, SVM, bagging, kNN, boosting, and NB respectively make 85, 99, 119, 178, 185, and 395 total errors on the 15 data sets.
- If using all the entropy discretized features, SVM, CS4, kNN, bagging, NB, and boosting respectively make 51, 75, 85, 113, 115, and 174 total errors.

Data sets	CS4			Bagging			Boosting		
	all	entropy	M-entropy	all	entropy	M-entropy	all	entropy	M-entropy
T-ALL	1	1	1	1	1	1	1	1	1
E2A-PBX1	1	1	1	1	1	1	1	1	1
TEL-AML1	6	6	6	12	11	10	9	13	14
BCR-ABL	8	7	6	13	12	12	22	18	15
MLL	7	5	6	10	9	8	13	14	18
Hyperdip>50	14	14	15	19	19	20	23	24	14
Ovarian	0	0	1	7	6	5	10	9	8
Prostate	9	9	8	10	9	10	14	10	8
Colon Tumor	14	11	12	12	10	12	12	10	12
ALL-AML	1	2	2	5	6	5	13	11	12
Subtype lymphoma	5	5	5	6	6	7	11	11	10
Stjude testing	12	7	6	14	12	9	26	22	19
Lung cancer	3	3	3	4	5	5	27	26	26
ALL-AML testing	4	4	3	3	4	4	3	3	3
Armstrong	0	0	0	2	2	2	0	1	1
Total Errors	85	75	75	119	113	111	185	174	162

Figure 13: The change trend of the error numbers of the three rule-based committee classifiers.

- If using only the mean-entropy discretized features, SVM, CS4, kNN, NB, bagging, and boosting respectively make 47, 75, 77, 98, 113, and 162 total errors on the 15 data sets.

So, SVM and our CS4 classifier are the best two classifiers irrespective of whether there is feature selection or not.

7 A Useful Software Package

Here, we would like to introduce you a widely-used software package for machine learning, data mining and bioinformatics. This software package is developed by *WEKA Machine Learning Project*; see `ttp://www.cs.waikato.ac.nz/~ml/index.`tml. It is called WEKA, for Waikato Environment for Knowledge Analysis. According to its documentations, "with it, a specialist in a particular field is able to use machine learning to derive useful knowledge from databases that are far too large to be analysed by hand."

WEKA contains machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. WEKA also contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. WEKA is open source software issued under the GNU General Public License. As such, we implemented our own CS4 algorithm and

integrated it into the WEKA software package.

8 Closing Remarks

In this tutorial, we have introduced some biomedical applications and their related bioinformatics and machine learning problems. We have shown the importance of rule-based classification methods for the analysis of biomedical data. These methods include a base rule-induction method C4.5, and ensemble approaches such as Bagging, Boosting, Random forest, Randomization trees, and our own CS4. For comparison, we have also introduced some other learning algorithms such as kNN, SVM, NB, and Hidden Markov Model. They are generally known as non-linear classification methods, usually not generating comprehensible rules suitable for understanding by non-specialist for example by doctors. We have also presented interesting decision trees induced from gene expression profiling data sets. This tutorial is enriched by a detailed performance comparison between the classifiers on 15 data sets when taking 3 different sets of features. A closely related topic to this tutorial is feature selection for classifications. Interested readers are referred to a recent bioinformatics book [102].

References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] U. Alon, et al. Broad patterns of gene expression revealed by clustering analysis of tumor colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA*, 96:6745–6750, 1999.

[3] G. E. Arnold, et al. Use of conditional probabilities for determining relationships between amino acid sequence and protein secondary structure. *Proteins*, 12(4):382–399, 1992.

[4] P. Baldi, et al. Hidden Markov models for human genes: Periodic patterns in exon sequences. In *Theoretical and Computational Methods in Genome Research*, pages 15–32, 1997.

[5] P. Baldi and Y. Chauvin. Hidden Markov models of G-protein-coupled receptor family. *Journal of Computational Biology*, 1:311–335, 1994.

[6] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. MIT Press, 1999.

[7] O. A. Bashkirov, E. M. Braverman, and I. B. Muchnik. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:629–631, 1964.

[8] A. Bateman, et al. Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic Acids Research*, 27(1):260–262, 1999.

[9] M. Borodovsky and J. D. McIninch. GENEMARK: Parallel gene recognition for both DNA strands. *Computers and Chemistry*, 17(2):123–133, 1993.

[10] M. Borodovsky, et al. Detection of new genes in a bacterial genome using Markov models for three gene classes. *Nucleic Acids Research*, 23:3554–3562, 1995.

[11] A.-L. Boulesteix, G. Tutz, and K. Strimmer. A CART-based approach to discover emerging patterns in microarray data. *Bioinformatics*, 19(18):2465–2472, 2003.

[12] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[13] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] L. Breiman, et al. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

[15] M. P. Brown, et al. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. USA*, 97(1):262–267, 2000.

[16] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[17] T. Chen and H. Chen. Universal approximation to non-linear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6:911–917, 1995.

[18] C. Cortez and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–279, 1995.

[19] T. M. Cover and P. E. Hart. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[21] J. Demsar, et al. Naive Bayesian-based nomogram for prediction of prostate cancer recurrence. *Studies in Health Technolnology Informatics*, 68:436–441, 1999.

[22] M. Dettling and P. Buhlmann. Boosting for tumor classification with gene expression data. *Bioinformatics*, 19:1061–1069, 2003.

[23] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[24] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 15–18, 1999.

[25] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[26] S. Dudoit and J. Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19:1090–1099, 2003.

[27] R. Durbin, et al., editors. *Biological Sequence Analysis: Probabilistics Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[28] S. R. Eddy. Hidden Markov models. *Current Opinion in Structural Biology*, 6:361–365, 1996.

[29] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

[30] V. Di Francesco, J. Granier, and P.J. Munson. Protein topology recognition from secondary structure sequences—applications of the hidden Markov models to the alpha class proteins. *Journal of Molecular Biology*, 267:446–463, 1997.

[31] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of 13th International Conference on Machine Learning*, pages 148–156, 1996.

[32] N. Friedman, et al. Using Bayesian networks to analyse expression data. *Journal of Computational Biology*, 7:601–620, 2000.

[33] T.-T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. In *Proceedings of 15th International Conference on Machine Learning*, 1998.

[34] J. Gehrke, et al. BOAT—optimistic decision tree construction. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 169–180, 1999.

[35] S. Ghosh and P. P. Majumder. Mapping a quantitative trait locus via the EM algorithm and Bayesian classification. *Genetic Epidemiology*, 19(2):97–126, 2000.

[36] C. Gini. Measurement of inequality of incomes. *The Economic Journal*, 31:124–126, 1921.

[37] G. J. Gordon, et al. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research*, 62(17):4963–4967, 2002.

[38] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[39] S. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[40] D. Heckerman. Bayesian networks for knowledge discovery. In *Advances in Knowledge Discovery and Data Mining*, pages 273–305, 1996.

[41] D. Hennessy, et al. Statistical methods for the objective design of screening procedures for macromolecular crystallization. *Acta Crystallogr. D Biol. Crystallogr.*, 56(7):817–827, 2000.

[42] K. Hornik. Some new results on neural network approximation. *Neural Networks*, 6:1069–1072, 1993.

[43] K. Huang and R. F. Murphy. Boosting accuracy of automated classification of fluorescence microscope images for location proteomics. *BMC Bioinformatics*, 5, 2004.

[44] X. Huang, et al. Modeling the relationship between LVAD support time and gene expression changes in the human heart by penalized partial least squares. *Bioinformatics*, 20:888–894, 2004.

[45] A. D. Huitema, et al. Validation of techniques for the prediction of carboplatin exposure: application of Bayesian methods. *Clinical Pharmacology & Therapeutics*, 67(6):621–630, 2000.

[46] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1–2):95–114, 2000.

[47] F. V. Jensen. *An Introduction to Bayesian Networks*. Springer-Verlag, 1996.

[48] G. H. John. *Enhancements to the Data Mining Process*. PhD thesis, Stanford University, 1997.

[49] G. D. Forney Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, 1973.

[50] G. V. Kaas. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29:119–127, 1980.

[51] S. Kasif and A. L. Delcher. Modeling biological data and structure with probabilistic networks. In *Computational Methods in Molecular Biology*, pages 335–352. 1998.

[52] R. Kohavi, et al. MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 740–743, 1994.

[53] A. Krogh. An introduction to hidden Markov models for biological sequences. In *Computational Methods in Molecular Biology*, pages 45–62, 1998.

[54] A. Krogh, et al. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

[55] M. Kukar, I. Kononenko, and T. Silvester. Machine learning in prognosis of the femoral neck fracture recovery. *Artificial Intelligence in Medicine*, 8(5):431–451, 1996.

[56] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

[57] J. Li, G. Dong, and K. Ramamohanarao. DeEPs: Instance-based classification using emerging patterns. In *Proceedings of 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 191–200, 2000.

[58] J. Li and H. Liu. Ensembles of cascading trees. In *Proceedings of 3rd IEEE International Conference on Data Mining*, pages 585–588, 2003.

[59] J. Li, H. Liu, S.-K. Ng, and L. Wong. Discovery of significant rules for classifying cancer diagnosis data. *Bioinformatics*, 19(Supplement 2):ii93–ii102, 2003.

[60] J. Li, H. Liu, and L. Wong. Mean-entropy discretized features are effective for classifying high-dimensional biomedical data. In *Proceedings of 3rd ACM SIGKDD Workshop on Data Mining*, pages 17–24, 2003.

[61] H. Liu and L. Wong. Data mining tools for biological sequences. *Journal of Bioinformatics and Computational Biology*, 1(1):139–168, 2003.

[62] W. Y. Loh and Y. S. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840, 1997.

[63] W. Y. Loh and N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis. *Journal of American Statistical Association*, 83:715–728, 1988.

[64] T.A. Longacre, et al. Proposed criteria for the diagnosis of well-differentiated endometrial carcinoma. A diagnostic test for myoinvasion. *American Journal of Surgical Pathology*, 19(4):371–406, 1995.

[65] I.S. Lossos, et al. Cerebrospinal fluid lactate dehydrogenase isoenzyme analysis for the diagnosis of central nervous system involvement in hematooncologic patients. *Cancer*, 88(7):1599–1604, 2000.

[66] T. M. Lowe and S. R. Eddy. tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Research*, 25(5):955–964, 1997.

[67] T. M. Lowe and S. R. Eddy. A computational screen for methylation guide snoRNAs in yeast. *Science*, 283:1168–1171, 1999.

[68] O. L. Mangasarian, W. Nick Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

[69] L. A. McCue, K. A. McDonough, and C. E. Lawrence. Functional classification of cnmp-binding proteins and nucleotide cyclases with implications for novel regulatory pathways in mycobacterium tuberculosis. *Genome Research*, 10(2):204–219, 2000.

[70] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* John Wiley, 1997.

[71] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In *Proceedings of International Conference on Extending Database Technology*, pages 18–32, 1996.

[72] M. Mehta, J. Rissanen, and R. Agrawal. MDL-based decision tree pruning. In *Proceedings of 1st International Conference on Knowledge Discovery in Databases and Data Mining*, pages 216–221, 1995.

[73] J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989.

[74] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[75] A. G. Pedersen, et al. Characterization of prokaryotic and eukaryotic promoters using hidden Markov models. *Intelligent Systems for Molecular Biology*, 4:182–191, 1996.

[76] A. G. Pedersen and H. Nielsen. Neural network prediction of translation initiation sites in eukaryotes: Perspectives for EST and genome analysis. *Intelligent Systems for Molecular Biology*, 5:226–233, 1997.

[77] E. F. Petricoin, et al. Use of proteomic patterns in serum to identify ovarian cancer. *Lancet*, 359:572–577, 2002.

[78] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods—Support Vector Learning.* MIT Press, 1998.

[79] S. L. Pomeroy et al. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415:436–442, 2002.

[80] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[81] J. R. Quinlan. Simplifying decision trees. *Interational Journal of Man-Machine Studies*, 27:221–334, 1987.

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

[83] T. H. Rainer, et al. Derivation of a prediction rule for post-traumatic acute lung injury. *Resuscitation*, 42(3):187–196, 1999.

[84] R. Rastogi and K. Shim. Public: A decision tree classifier that integrates building and pruning. In *Proceedings of 24th International Conference on Very Large Data Bases*, pages 404–415, 1998.

[85] S. Raudys. How good are support vector machines? *Neural Network*, 13(1):17–19, 2000.

[86] S. Russell, et al. Local learning in probabilistic networks with hidden variables. In *Proceedings of 14th Joint International Conference on Artificial Intelligence, volume 2*, pages 1146–1152, 1995.

[87] S. L. Salzberg, et al. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.

[88] G. A. Satten, et al. Standardization and denoising algorithms for mass spectra to classify whole-organism bacterial specimens. *Bioinformatics*, in press, 2004.

[89] B. Scholkopf and A. J. Smola. *Learning with Kernels.* MIT Press, 2002.

[90] H. P. Selker, et al. A comparison of performance of mathematical predictive methods for medical diagnosis: identifying acute cardiac ischemia among emergency department patients. *Journal of Investigative Medicine*, 43(5):468–476, 1995.

[91] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of 22nd International Conference on Very Large Data Bases*, pages 544–555, 1996.

[92] D. Singh et al. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:203–209, 2002.

[93] L. A. Soinov, M. A. Krestyaninova, and A. Brazma. Towards reconstruction of gene networks from expression data by supervised learning. *Genome Biology*, 4(1):R6.1–9, 2003.

[94] C. M. Stultz, et al. Predicting protein structure with probabilistic models. In *Protein Structural Biology in Biomedical Research*, pages 447–506, 1997.

[95] N. R. Temkin, et al. Classification and regression trees (CART) for prediction of function at 1 year following head trauma. *Journal of Neurosurgery*, 82(5):764–771, 1995.

[96] J. B. Tobler, et al. Evaluating machine learning approaches for aiding probe selection for gene-expression arrays. *Bioinformatics*, 18:164S–171S, 2002.

[97] P. E. Utgoff. An incremental ID3. In *Proceedings of 5th International Conference on Machine Learning*, pages 107–120, 1988.

[98] L. J. van't Veer, et al. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–536, 2002.

[99] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[100] J. Weston, et al. Feature selection and tranduction for prediction of molecular bioactivity for drug design. *Bioinformatics*, 19(6):764–771, 2003.

[101] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.

[102] L. Wong, editor. *The Practical Bioinformatician*. World Scientific, 2004.

[103] B. Wu, et al. Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics*, 19:1636–1643, 2003.

[104] T. Yada, et al. Extraction of hidden Markov model representations of signal patterns in DNA sequences. In *Proceedings of Pacific Symposium on Biocomputing*, pages 686–696, 1996.

[105] E.-J. Yeoh, et al. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1:133–143, 2002.

[106] A. Zien, et al. Engineering support vector machine kernels that recognize translation initiation sites. In *Proceedings of German Conference on Bioinformatics*, pages 37–43, 1999.

[107] A. Zien, et al. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.