

Chapter 1

Introduction

The genomes of several organisms have now been completely sequenced, including the human genome -- depending on one's definition of "completely" :-). Interest within bioinformatics is therefore shifting somewhat away from sequencing, to learning about the genes encoded in the sequence. Gene's 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. The present data set consists of a variety of details about the various genes of one particular type of organism. Gene names have been anonymized and a subset of the genes have been withheld for testing. The task is to predict the localizations of the proteins encoded by the genes. A gene/protein can have more than one function, but (at least in this data set) only one localization. The other information from which function and localization can be predicted includes the class of the gene/protein, the phenotype (observable characteristics) of individuals with a mutation in the gene (and hence in the protein), and the other proteins with which each protein is known to interact.

This thesis describes a case study that was carried out by us for the KDD Cup 2001 task 3, a competition for gene/protein localization problems. In this chapter we introduce this international competition and briefly describe the problems.

1.1 The KDD Cup 2001

The KDD Cup 2001 was held in conjunction with the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. The Cup has been held annually in the last 5 years. Every year the organizers addressed some large-scale practical problems on data mining, and many people from academic and business areas are attracted to participate in this contest. After a careful evaluation, the winners are invited to give a brief talk in the KDD Conference.

The competition this year involved 3 tasks, based on two data sets. The two training datasets are available from the web, as zip files. The first dataset is a little over half a gigabyte when uncompressed and comes as a single text file, with one row per record and fields separated by

commas. The second is a little over 7 megabytes uncompressed. It includes a single text file with all the data. Participants will be judged on the following 3 tasks:

task 1: Prediction of Molecular Bioactivity for Drug Design -- Binding to Thrombin (The first dataset)

task 2: Prediction of Gene/Protein Function (The second dataset)

task 3: Prediction of Gene/Protein Localization (The second dataset)

After brief considerations, we chose task 3 for study. There are two reasons on this decision. First, task 3 is a classification problem of 15 classes, which is a typical task for our interested classification method, decision tree. Second, task 3 seems simpler than the others. For the lack of experiences on solving real world problems, it is better to begin with an easier one. Therefore, task 1 and task 2 will not be discussed in this thesis.

1.2 Overview of our study

We agree that data mining is a procedure in the knowledge discovery process and consists of an iterative sequence of the following steps:

(1) Learning the application domain:

- relevant prior knowledge and goals of application

(2) Data cleaning and preprocessing:

● Data reduction and transformation:

- Find useful features, dimensionality/variable reduction, invariant representation.

(3) Choosing the mining algorithm(s)

(4) Data mining: search for patterns of interest

(5) Pattern evaluation and knowledge presentation

- visualization, transformation, removing redundant patterns, etc.

At the beginning, without investigating the problem data, we directly used decision tree to build the predictive model. Unfortunately, our model did not achieve a satisfactory accuracy. Because contravene above-mentioned steps. On the contrary, task 3 winner, who made a good use of nearest neighbor method, did thorough observations to the training data, and figured out better

training strategies.

Afterward, we realized that data preprocessing plays an important role in data mining. According to some statistics announced by the organizers, approximately 50% of the efforts are spent on data preprocessing. The data preprocessing work will be described in detail in the following chapters. Also we cut the mining work into a couple of stages to improve our accuracy to a higher level. At the first stage, we modified the decision tree algorithm to execute the primary classification work. For the second stage, we aggregated every single gene which has interaction relationship with one gene to a flock of genes. All flocks of genes form a majority voting table for the secondary classification work.



Chapter 2

Problem description and existing approaches

2.1 Description of the problem

2.1.1 The problem

As indicated earlier, task 3 of KDD Cup 2001 is a Classification problem with 15 classes. 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. A transformed training example from dataset 2 is shown below.

table 2.1

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

The transform method will be discussed in detail in section.

Detailed knowledge of the biology should not be necessary for solving this problem. This is so much the case that we almost even anonymized all the other fields as well as the gene field. But in the end they decided instead to leave the other fields alone, since this might make the data set more interesting.

2.1.2 The dataset

Dataset 2 contains two compression files, one is Training data and the other is test data.

The Training data decompression is as follows:

- 1.Full_File.data
- 2.Full_File.names
- 3.Genes_relation.data
- 4.Genes_relation.names
- 5.Interactions_relation.data
- 6.Interactions_relation.names
- 7.TotalGenesList
- 8.TrainingGenesList
- 9.readme



The most important two files are :

- 1.Full_File.data and 5.Interactions_relation.data

The 2. Full_File.nameses and 6.Interactions_relation.names are explanation files versus

- 1.Full_File.data and 5.Interactions_relation.data

The Test data decompression are as follows:

- 1.Full_File.test
- 2.Genes_relation.test
- 3.Interactions_relation.test
- 4.readme

The full data set is in Full_File.data. But please notice that the task is quite "relational". For example, one might wish to learn a rule that says a gene G is in location "nucleus" if G interacts with another gene G1, which is in "nucleus". The organizer has made an effort to build such features into Full_File.data . (For example, for each gene they give the number of interacting genes with a given function -- these features are probably useful for predicting at least one or two of the functions). But participants may wish to construct their own additional features or to use a relational data mining algorithm. While this certainly can be done from Full_File.data, it may be easier to do this from the relational tables that they used to build Full_File.data. These are in Genes_relation.data and Interactions_relation.data. Each of the data files has a corresponding names file as well.

Dataset 2 which was prepared for the KDD Cup 2001 task 3, has three interesting features: (1) the dataset contains many missing values; (2) this dataset has a lot of attributes; 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. So our study will focus on solving these problems. We develop nonrecursive decision tree method and primary-secondary classification systems to improve prediction accuracy.

2.2 The methods for special features

2.2.1 The methods for missing values

How can we go about dealing with the missing values for this attribute? Let's look at the following methods:

1.Ignore the tuple: This is usually done when the class label is missing (assuming the mining task involves classification or description). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.

2.Fill in the missing value manually: In general, this approach is time-consuming and may not

be feasible if given a large data set with many missing values.

3. Use a global constant to fill in the missing value: Replace all missing attribute values by the same constant, such as a label like "Unknown" or $-\infty$. If missing values are replaced by, say, "Unknown," then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common—that of "Unknown." Hence, although this method is simple, it is not recommended.

4. Use the attribute mean to fill in the missing value.

5. Use the attribute mean for all samples belonging to the same class as the given tuple: For example, if classifying customers according to credit_risk, replace the missing value with the average income value for customers in the same credit risk category as that of the given tuple.

6. Use the most probable value to fill in the missing value: This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction.

7. Discard the whole record that has missing values: The simple and the most direct way is to do without the missing values in the whole data mining process.

8. Compensate the lost information of the missing attribute values by other attribute

2.2.2 The methods for the mass attributes

Data reduction

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results. Strategies for data reduction include the following:

1. Data cube aggregation, where aggregation operations are applied to the data in the

construction of a data cube.

2.Dimension reduction, where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.

3.Data compression, where encoding mechanisms are used to reduce the dataset size.

4.Numerosity reduction, where the data are replaced or estimated by alternative, smaller data representations such as parametric models (which store only the model parameters instead of the actual data), or nonparametric methods such as clustering, sampling, and the use of histograms.

5.Discretization and concept hierarchy generation, where raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction and are a powerful tool for data mining.

2.2.3 The methods for the hybrid type data

The Primary-Secondary classification system is a good strategy for the mixing type data. The different format data is tackled separately. The main dataset is treated by the primary classification method. The interaction relation data as well as the missing value genes are transacted by secondary classification method. For the Primary classification, we modify the decision tree method to the nonrecursive decision tree as the main method for classification, and the majority voting skill is used in the secondary classification. For the secondary classification, we use data aggregation to transform the interaction relation data to a majority voting table. Next chapter we will introduce the Primary-Secondary classification in detail.

Chapter 3

Primary-Secondary classification

3.1 Introduction

Because of the Full_File.data and the Interactions_relation.data are different format data and the Full_File.data is the primary data of the whole dataset, we develop the Primary-Secondary classification system to cut down the complexity of the question. At the Primary classification stage, the primary data are classified. At the Secondary classification stage, the Interactions_relation.data and the missing values are treated. In addition only depend on the Primary classification method is still not enough to reach a decent classification accuracy; and missing values and the interaction relation between genes are still the two pending problems of the Primary classification. We solve this two problems by the Secondary classification method. For the sake of a quick understanding of the Primary-Secondary classification, we present the structure of the Primary-Secondary classification system in figure 3.1.

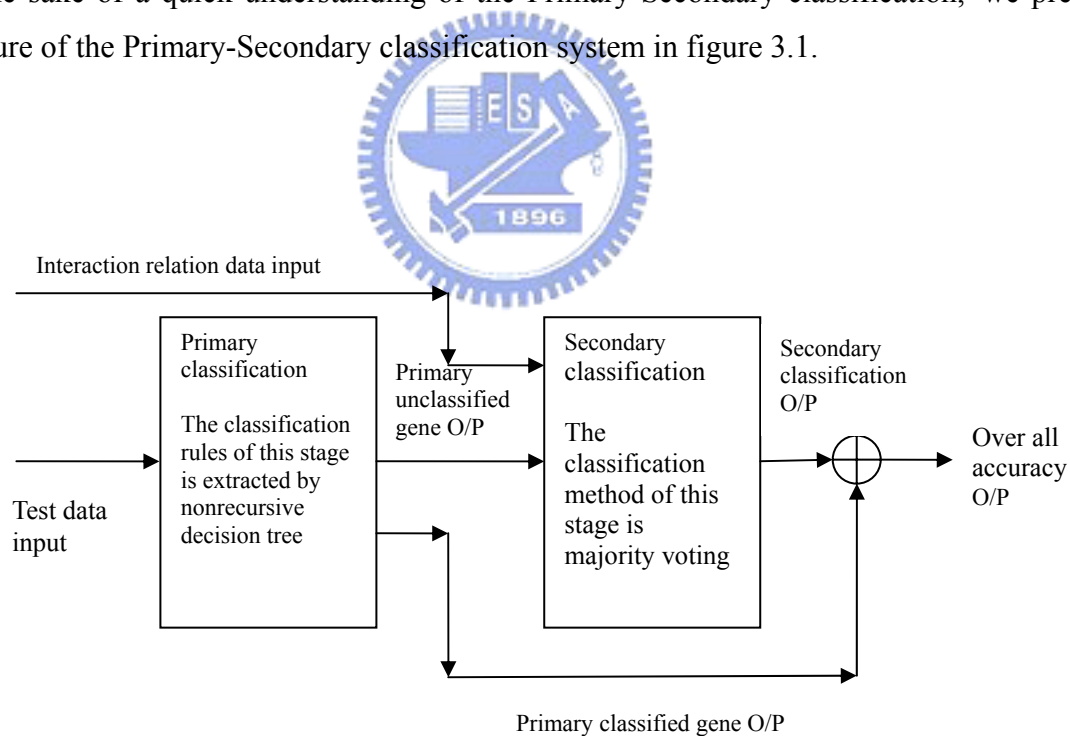


Figure 3.1

Primary classification is a nonrecursive decision tree based classifier, some of classification rules is found in this stage by feeding the training data set into the nonrecursive decision tree. The

interaction relation data and the unclassified genes of the Primary classification are the input of the Secondary classification. The classification method of this stage is majority voting. The main function of Secondary classification is to treat the missing values and interaction relation data. The integration of the Secondary classification output and Primary classification output classified genes form the overall classification accuracy.

3.2 Primary classification

3.2.1 Primary classification preprocessing

Analysis of the data

Before the primary classification work, we have known that further understanding of the whole dataset is very important. Thus we should penetrate into the analysis of the Full_File.data and the Interactions_relation.data .The following is an example of one record of the Full_File.data.



G234455,Non-Essential, No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No, Yes
s,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No, Yes,No,No,No,No,No,No,No,No,
No,N
o,No,No,No,No,No,No,No,No,No,No,No,No, Yes, No,No,No,No,No,No,No,No,No,No,
, No,
No,
No,
No,
No,
No,
No,
No,
No,
No,
No,
No,
No,
No,
No,
No,
No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,No,
No,No,No,No,No,No,No,
No,No,No,No,No,No,
No,No,No,No,No,
No,No,No,No,
No,No,No,
No,No,
No,
No,

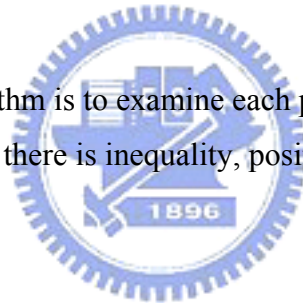
will encode the binary attribute value into text attribute value to compress the record in a lossless manner.

Data compression

We know the quantity of features was too large for most classification algorithms. For overcoming this difficulty, we have the idea of data compression modulus. We have to encode binary attribute into text attribute in order to decrease the attribute quantity (The attribute “interacting gene” will be omitted at present since it has too many attributes. The secondary classification stage has a good method for disposal.) . The basic encoding means is based on “String Searching”. The String Searching algorithm is to search for a pattern string, $pat[1..m]$, in a text string $txt[1..n]$. Usually $n \gg m$, and txt might be very long indeed, although this is not necessarily so. In the following, we introduce three typical algorithms :

Naive Search

The naive string searching algorithm is to examine each position, $i \geq 1$, in txt , trying for equality of $pat[1..m]$ with $txt[i..i+m-1]$. If there is inequality, position $i+1$ is tried, and so on.



Rabin's Algorithm

Rabin's string searching algorithm uses a "rolling hash". Hashing is an idea borrowed from hash-tables but used here without the "table". In Rabin's algorithm, the hash function is defined to compute a number in $[0..p-1]$ for a large prime number p . The integers modulo p , Z_p , behave as a mathematical field. That is, they behave just as the usual unbounded integers do with respect to $+$ and $*$, and in particular there are no "divisors of zero" i.e. no values $j \neq 0$ & $k \neq 0$ such that $j*k=0$.

At each stage of the algorithm, $hash(txt[i..i+|pat|-1])$ is compared with $hash(pat)$, e.g.

$$patHash = (((ord(pat[1])*3+ord(pat[2]))*3+...+ord(pat[m])) \bmod p$$

$$txtHash_1 = (((ord(txt[1])*3+ord(txt[2]))*3+...+ord(txt[m])) \bmod p$$

$$txtHash_i = ((txtHash_{i-1} - ord(txt[i-1])*power)*3 + ord(txt[i+m-1])) \bmod p, \text{ where } i > 1$$

$$\text{where } power = 3^{m-1}$$

and $\text{ord}(\text{ch})$ is the character code of ch

If the values are equal then a match has almost certainly been found. A character by character check must be made to be absolutely sure.

Boyer-Moore Algorithm

The Boyer Moore algorithm is always fast when having worst-case time-complexity $O(m+n)$, but on natural-language text it actually gets faster as m increases to a certain extent, e.g. Boyer and Moore suggesting $O(n/4)$ -time on average for $m=5$.

The first key idea, and it is a good one, is that if the m^{th} character $\text{ch}=\text{txt}[m]$ (numbered from '1' upwards) does not occur in pat at all, then any instance of pat in txt must start at position $m+1$ or later.

e.g. if searching for $\text{pat}=\text{'freddy'}$, in $\text{txt}=\text{'I floated lonely as a cloud'}$, the 6th character of txt is 'a' which is not in $\{d,e,f,r,y\}$ and there is no need to consider positions 1 to 6 of txt any more! In this way, we can move along txt in steps of m positions, i.e. $k=m, 2m, 3m, \dots$, provided we are lucky.

The second idea is that if the last occurrence of the character $\text{txt}[k]$ in pat is $\text{delta}_1[\text{ch}]$ positions from the right hand end of pat , then pat can slide many positions to the right before we might get a match in txt . If $\text{ch}=\text{txt}[k]$ does not occur in pat , set $\text{delta}_1[\text{ch}]$ equal to m .

e.g. $\text{delta}_1[\text{'e'}]=3$, $\text{delta}_1[\text{'f'}]=5$, $\text{delta}_1[\text{'d'}]=1$, $\text{delta}_1[\text{'r'}]=4$, $\text{delta}_1[\text{'y'}]=0$.

In general, if the last j characters match, i.e. $\text{txt}[k-j..k]=\text{pat}[m-j..m]$, but $\text{txt}[k-j-1]\neq\text{pat}[m-j-1]$, then pat can "slide" a certain distance along txt depending on where, if at all, there is an earlier instance of $\text{pat}[m-j..m]$ within pat ; e.g. consider $\text{pat}=\text{abracadabra}$, or e.g. $\text{pat}=\text{fababab}$. An array, $\text{delta}_2[1..m]$, is used to hold these precomputed distances.

The Encoding Algorithm:

Based on The String Searching algorithm we can compose a program or use some data processing tools to do the encoding job.

The encoding algorithm:

1. Load one record.
2. halt if it reaches the end of the data
3. The record will be divided into 11 segments, first segment-versus Gene, second segment versus Essential, third segment-versus Class, fourth segment versus Complex, the rest can be deduced accordingly.
4. With String Searching algorithm to find the string “Yes” of the Class segment, the sequence of the “Yes” map to the attribute value of Class.
5. With String Searching algorithm to find the string “Yes” of the Complex segment, the sequence of the “Yes” map to the attribute value of Complex.

:
:

9. With String Searching algorithm to find the string “Yes” of the Motif segment, the sequence of the attribute value map to the attribute value of Complex.

10.go to step 1

The encoded Gene G234455 is in table 3.2 :



table 3.2

Gene	Essential	Class	Complex	phenotype	Motif	Chromosome	Localization
G234455	Non-Essential	Protein Kinases (class 20)	Cytoskeleton (Complex 14)	4	no	7	cytoplasm

3.2.2 decision tree

History of ID3 [15]

In 1966, three researchers in the field of cognitive psychology by the names of Hunt, Marin, and Stone published a model of concept formation. This model is based on the idea that, in order to make discrete concepts from new, cluttered, or otherwise unstructured data, human minds break down items and concepts into subsets with common characteristics. Here is an outline of the concept formation strategy published by Hunt, Marin, and Stone:

1. Pick a criterion that appears (according to some heuristic) to be the most useful for separating a set into subsets. A simple example which dates to prehistory is the distinction between friend and foe; further separations include human/animal and male/female.
2. Separate the set according to the selected criterion.
3. Return to the first step, choosing new and increasingly precise criteria until a unique definition of the new data is reached. They called this algorithm as the Concept Learning System.

Many cognitive psychologists claim that this method is based on an innate concept of divide - and conquer. In fact, some computer scientists explain their design of divide and conquer algorithms to be intuitive. Coincidence? Perhaps not, but that is the topic for a different paper. In the late 1970's, an artificial intelligence researcher named J. R. Quinlan used the model of concept formation to develop a program which he called ID3 (for Itemized Dichotomizer 3). Quinlan's algorithm "learned" from a relatively small training set of data how to organize and process very large data sets. It used a divide and conquer strategy combined with logical dichotomization (splitting into groups) to produce impressive results in comparatively short processing times. Simultaneously, Breiman and Friedman and colleagues in statistics developed CART (Classification and Regression Tree) methods very similar to ID3. In the 1980's a variety of improvements was introduced to handle noise, continuous attributes, missing data, and improved splitting criteria. Various expert-system development tools resulted. Quinlan's updated decision-tree induction package (C4.5) released in 1993, is now commonly used.

Information Retrieval

Information retrieval (IR) is described in [17] as the science which concerns itself with the "representation, storage, organization, and accessing of information items" 2 . Artificial intelligence techniques of learning algorithms and natural language processing correlate directly to the field of IR.

Benefits and disadvantages of ID3

The IR research explained in [18] discusses the benefits of ID3 as a "greedy" technique, one which finds the optimal solution per level and never backtracks. The symbolic learning used lends itself well to the concepts of ID3 since at each decision node, an element of a window must be either desirable or undesirable; this yields a binary decision tree.

In the IR experimentation, the universe of data comprised of records, and the records were explained in terms of keywords. It is the distribution of these keywords that were used to calculate the entropy at different levels of the resultant decision tree.

Although ID3 is not usually a primary choice in algorithm selection for IR purposes, the researchers involved in this experiment found that if trained with chosen data sets intelligently, ID3 could perform as well as the more traditional chosen algorithms (relevance feedback and genetic algorithms, for example).

One of the strongest benefits of ID3 is its simplicity; when compared to other learning algorithms, ID3 is much more straightforward in its approach. Its cognition based modeling makes it relatively simple for humans to understand how ID3 works. Unfortunately, the decision trees produced by ID3, when used to process large or noisy data sets, tend to be confusing to human perception. ID3 performs very well given large and complex data sets --- much better than its predecessor, CLS. One of the ways which ID3 does this is by finding "hidden" data and relationships; it looks at problems using a simpleminded divide and conquer technique.

Another benefit of ID3 is its conservative use of system resources. The computational time involved in ID3 is linear and can be calculated as the product of the number of training objects, the number of possible attributes which describe each object, and the complexity of the final selection criteria (measured as the number of nodes in the decision tree).

A major disadvantage of ID3 is that the decision trees produced are essentially immutable --- one can not efficiently change the decision tree without rebuilding it from scratch. Using a patchwork method of updating the tree tends to yield a decision tree which is no longer optimal, thus refuting the original purpose of forming the ID3 decision tree.

decision tree [16]

Because our method is a mutation of decision tree algorithm, we should introduce the decision tree method thoroughly in the first place.

Aim:

To describe an algorithm whose input is a collection of instances and their correct classification and whose output is a decision tree that can be used to classify each instance.

Keywords table:

table 3.3

concept learning system (CLS)	A decision tree induction program - a precursor of ID3.
C4.5, C5	C4.5 & C5 are later version of the ID3 decision tree induction algorithm.
entropy	The entropy measure gives us the average amount of information in bits in some attribute of an instance. Entropy is used in the ID3 decision tree induction algorithm.
ID3	A decision tree induction algorithm, developed by Quinlan. ID3 stands for "Iterative Dichotomizer (version) 3". Later versions include C4.5 and C5.
instances	An instance is a term used in machine learning particularly with symbolic learning algorithm, to describe a single training item, usually in the form of a description of the item, along with its intended classification.

Plan:

- What is a decision tree?
- Building a decision tree

- Which attribute should we split on?
- Information theory and the splitting criterion
- ID3 example & ID3 symbolic version
- ID3 enhancements: windowing

Decision Trees

- A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision.
- For example, we might have a decision tree to help a financial institution decide whether a person should be offered a loan:

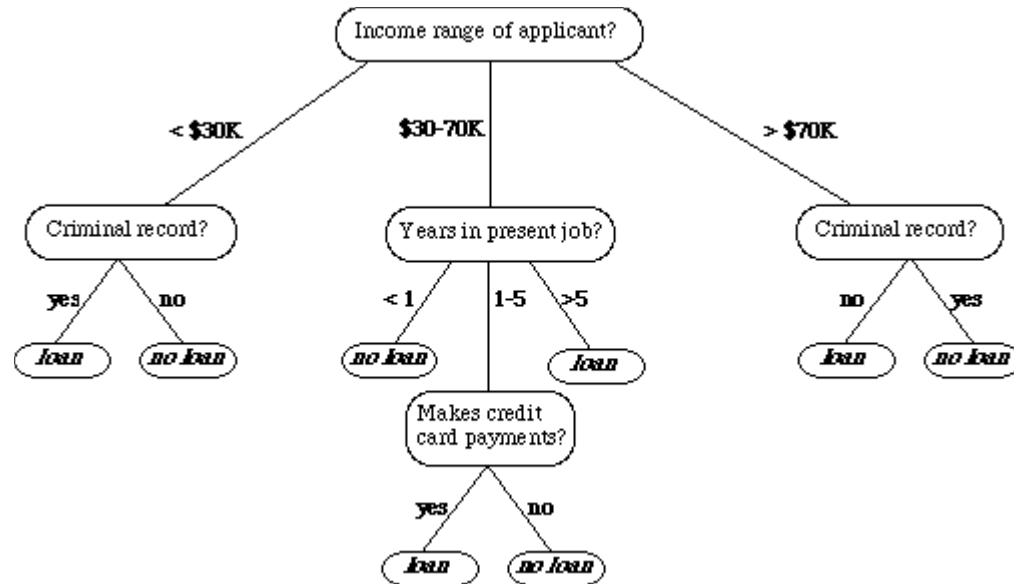


figure 3.2

- We wish to be able to induce a decision tree from a set of data instance together with the decisions or classifications for those instances.

Example Instance Data

size: small medium large

colour: red blue green

shape: brick wedge sphere pillar

% yes

medium	blue	brick
small	red	sphere
large	green	pillar
large	green	sphere

% no

small	red	wedge
large	red	wedge
large	red	pillar

- In this example, there are 7 instances, described in terms of three features or attributes (size, color, and shape), and the instances are classified into two classes – yes and no.
- We shall now describe an algorithm for inducing a decision tree from such a collection of classified instances.
- Originally termed CLS (Concept Learning System) it has been successively enhanced.
- At the highest level of enhancement that we shall describe in these notes, the system is known as ID3 - later versions include C4, C4.5 and C5.
- ID3 and its successors have been developed by Ross Quinlan.

Tree Induction Algorithm

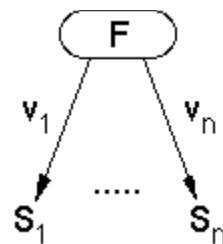


figure 3.3

- The algorithm operates over a set of training instances, C .
- If all instances in C are in class P , create a node P and stop, otherwise select a feature or attribute F and create a decision node.

- Partition the training instances in C into subsets according to the values of V.
- Apply the algorithm recursively to each of the subsets C.

Output of Tree Induction Algorithm

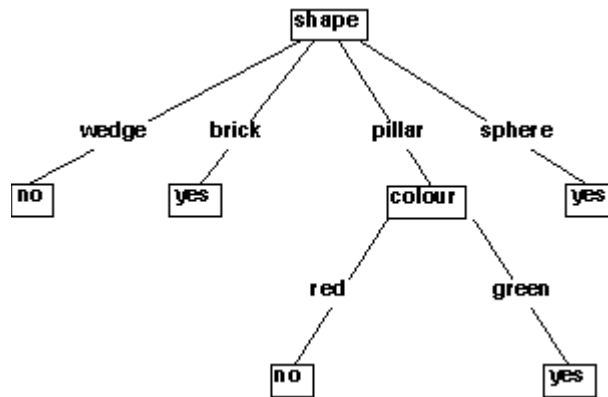


figure 3.4

This can easily be expressed as a nested if-statement

```

if (shape == wedge) return no;
if (shape == brick) return yes;
if (shape == pillar)
{
    if (color == red) return no;
    if (color == green) return yes;
}
if (shape == sphere) return yes;
  
```



Choosing Attributes and ID3

- The order in which attributes are chosen determines how complicated the tree is.
- ID3 uses information theory to determine the most informative attribute.
- A measure of the information content of a message is the inverse of the probability of receiving the message:

$$\text{information}(M) = 1/\text{probability}(M)$$

- Taking logs (base 2) makes information correspond to the number of bits required to encode a message:

$$\text{information}(M) = -\log_2(\text{probability}(M))$$

Information

- The information content of a message should be related to the degree of surprise in receiving the message.
- Messages with a high probability of arrival are not as informative as messages with low probability.
- Learning aims to predict accurately, i.e. reduce surprise.
- Probabilities are multiplied to get the probability of two or more things both/all happening. Taking logarithms of the probabilities allows information to be added instead of multiplied.



Entropy

- Different messages have different probabilities of arrival.
- Overall level of uncertainty (termed entropy) is:

$$-\sum P \log_2 P$$

- Frequency can be used as a probability estimate.
- E.g. if there are 5 +ve examples and 3 -ve examples in a node the estimated probability of +ve is $5/8 = 0.625$.

For our purposes, the entropy measure

$$-\sum_j P_j \log_2 P_j$$

gives us the average amount of information in bits in some attribute of an instance. The rationale for this is as follows: $-\log_2(p)$ is the amount of information in bits associated with an event of

probability p - for example, with an event of probability $\frac{1}{2}$, like flipping a fair coin, $\log_2(p)$ is $-\log_2(\frac{1}{2}) = 1$, so there is one bit of information. This should coincide with our intuition of what a bit means (if we have one). If there is a range of possible outcomes with associated probabilities, then to work out the average number of bits, we need to multiply the number of bits for each outcome ($-\log_2(p)$) by the probability p and sum over all the outcomes. This is where the formula comes from. Entropy is used in the ID3 decision tree induction algorithm.

Information and Learning

- We can think of learning as building many-to-one mappings between input and output.
- Learning tries to reduce the information content of the inputs by mapping them to fewer outputs.
- Hence we try to minimize entropy.
- The simplest mapping is to map everything to one output.
- We seek a trade-off between accuracy and simplicity.

Splitting Criterion

- Work out entropy based on distribution of classes.
- Trying splitting on each attribute.
- Work out expected information gain for each attribute.
- Choose best attribute.

The point of the ID3 algorithm is to decide the best attribute, out of those not already used, on which to split the training instances that are classified to a particular branch node.

The algorithm, in outline, is as follows:

1. if all the instances belong to a single class, there is nothing to do (except create a leaf node labeled with the name of that class).
2. otherwise, for each attribute that has not already been used, calculate the information gain that would be obtained by using that attribute on the particular set of instances classified to this branch node.
3. use the attribute with the greatest information gain.

This leaves the question of how to calculate the information gain associated with using a particular attribute A. Suppose that there are k classes C_1, C_2, \dots, C_k , and that of the N instances classified to this node, I_1 belong to class C_1 , I_2 belong to class C_2 , ..., and I_k belong to class C_k .

Let $p_1 = I_1/N$, $p_2 = I_2/N$, ..., and $p_k = I_k/N$.

The expected information needed to classify a given sample is given by:

$$I = -p_1 \log_2(p_1) - p_2 \log_2(p_2) \dots - p_k \log_2(p_k).$$

Now split the instances on each value of the chosen attribute A. Suppose that there are r attribute values for A, namely a_1, a_2, \dots, a_r .

For a particular value a_j , say, suppose that there are $J_{j,1}$ instances in class C_1 , $J_{j,2}$ instances in class C_2 , ..., and $J_{j,k}$ instances in class C_k , for a total of J_j instances having attribute value a_j .

Let $q_{j,1} = J_{j,1}/J_j$, $q_{j,2} = J_{j,2}/J_j$, ..., and $q_{j,k} = J_{j,k}/J_j$.

The entropy E_j associated with this attribute value a_j this position is:

$$\text{entropy } (E_j) = -q_{j,1} \log_2(q_{j,1}) - q_{j,2} \log_2(q_{j,2}) \dots - q_{j,k} \log_2(q_{j,k}).$$

Now compute:

$$\text{gain } (A) = I - E_A.$$

This is the information gain for attribute A.

Note that J_j/N is the estimated probability that an instance classified to this node will have value a_j for attribute A. Thus we are weighting the entropy estimates E_j by the estimated probability that an instance has the associated attribute value.

Summary of Splitting Criterion

Assume there are k classes C_1, \dots, C_k ($k = 2$ in our example).

to decide which attribute to split on:

- for each attribute that has not already been used
 - Calculate the information gain that results from splitting on that attribute
 - Split on the attribute that gives the greatest information gain.

to calculate the information gain from splitting N instances on attribute A:

- Calculate the entropy E of the current set of instances.
- for each value a_j of the attribute $A(j = 1, \dots, r)$
 - Suppose that there are $J_{j,1}$ instances in class C_1 , ..., $J_{j,k}$ instances in class C_k , for a total of J_j instances with $A = a_j$.
 - Let $q_{j,1} = J_{j,1}/J_j$, ..., $q_{j,k} = J_{j,k}/J_j$
 - The entropy E_j associated with $A = a_j$ is $-q_{j,1} \log_2(q_{j,1}) \dots -q_{j,k} \log_2(q_{j,k})$
- Now compute $E - (J_1/N)E_1 \dots - (J_r/N)E_r$ - this is the information gain associated with a split on attribute A .

to calculate the entropy E of the current set of instances

- Suppose that of the N instances classified to this node, I_1 belong to class C_1 , ..., I_k belong to class C_k
- Let $p_1 = I_1/N$, ..., $p_k = I_k/N$.
- Then the initial entropy E is $-p_1 \log_2(p_1) - p_2 \log_2(p_2) \dots - p_k \log_2(p_k)$.

Windowing

- ID3 can deal with very large data sets by performing induction on subsets or windows onto the data.
 1. Select a random subset of the whole set of training instances.
 2. Use the induction algorithm to form a rule to explain the current window.
 3. Scan through all of the training instances looking for exceptions to the rule.
 4. Add the exceptions to the window.
- Repeat steps 2 to 4 until there are no exceptions left.

Noisy Data

- Frequently, training data contains "noise" - i.e. examples which are misclassified, or where one or more of the attribute values is wrong.

- In such cases, one is like to end up with a part of the decision tree which considers say 100 examples, of which 99 are in class C1 and the other is apparently in class C2 (because it is misclassified).
- If there are any unused attributes, we might be able to use them to elaborate the tree to take care of this one case, but the subtree we would be building would in fact be wrong, and would likely misclassify real data.
- Thus, particularly if we know there is noise in the training data, it may be wise to "prune" the decision tree to remove nodes which, statistically speaking, seem likely to arise from noise in the training data.
- A question to consider: How fiercely should we prune?

Expected Error Pruning

- Approximate expected error assuming that we prune at a particular node.
- Approximate backed-up error from children assuming we did not prune.
- If expected error is less than backed-up error, prune.

(Static) Expected Error

- If we prune a node, it becomes a leaf labelled, C.
- What will be the expected classification error at this leaf?

$$E(S) = \frac{N - n + k - 1}{N + k}$$

(This is called the Laplace error estimate - it is based on the assumption that the distribution of probabilities that examples will belong to different classes is uniform.)

S: is the set of examples in a node

k: is the number of classes

N: examples in S

C: is the majority class in S

n: out of N examples in S belong to C

Backed-up Error

- For a non-leaf node
- Let children of Node be Node1, Node2, etc

$$\text{BackedUpError}(\text{Node}) = \sum_i P_i \times \text{Error}(\text{Node}_i)$$

- Probabilities can be estimated by relative frequencies of attribute values in sets of examples that fall into child nodes.

$$\text{Error}(\text{Node}) = \min(E(\text{Node}), \text{BackedUpError}(\text{Node}))$$

Pruning Example

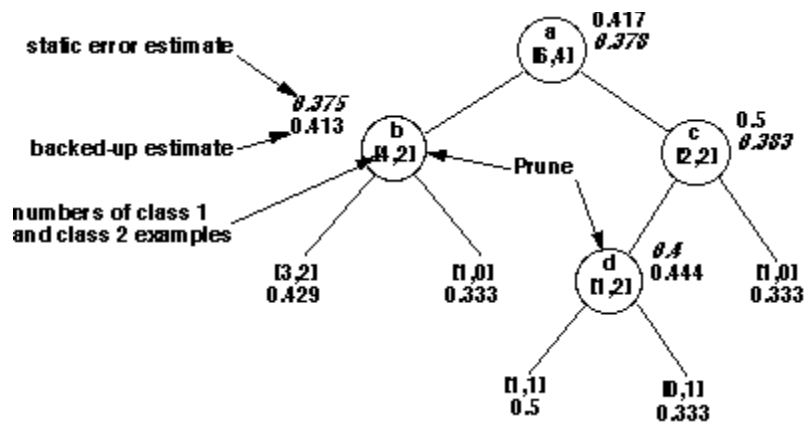


figure 3.5

Error Calculation

- Left child of b has class frequencies [3, 2]

$$E = \frac{N - n + k + 1}{N + k} = \frac{5 - 3 + 2 - 1}{5 + 2} = 0.429$$

- Right child has error of 0.333, calculated in the same way
- Static error estimate $E(b)$ is 0.375, again calculated using the Laplace error estimate formula, with $N=6$, $n=4$, and $k=2$.

- Backed-up error is:

$$\text{BackedUpError}(b) = \frac{5}{6} \times 0.429 + \frac{1}{6} \times 0.333 = 0.413$$

(5/6 and 1/6 because there are 4+2=6 examples handled by node b, of which 3+2=5 go to the left subtree and 1 to the right subtree.

- Since backed-up estimate of 0.413 is greater than static estimate of 0.375, we prune the tree and use the static error of 0.375

Summary:

The ID3 family of decision tree algorithms use information theory to decide which attribute shared by a collection of instances to split the data on next. Attributes are chosen repeatedly in this way until a complete decision tree that classifies every input is obtained. If the data is noisy, some of the original instances may be misclassified. It may be possible to prune the decision tree in order to reduce classification errors in the presence of noisy data. The speed of this learning algorithm is reasonably high, as is the speed of the resulting decision tree classification system. Generalization ability can be reasonable too.

When there is a finite number of features and a broad cross section displayed by a training set, ID3 performs well and is a wise algorithm choice. In a system with indefinite, generalized, or otherwise abstract data sets, ID3 may not perform as well as other algorithms.

3.2.2 The modified decision tree (The Proposing Approach)

Nonrecursive decision tree

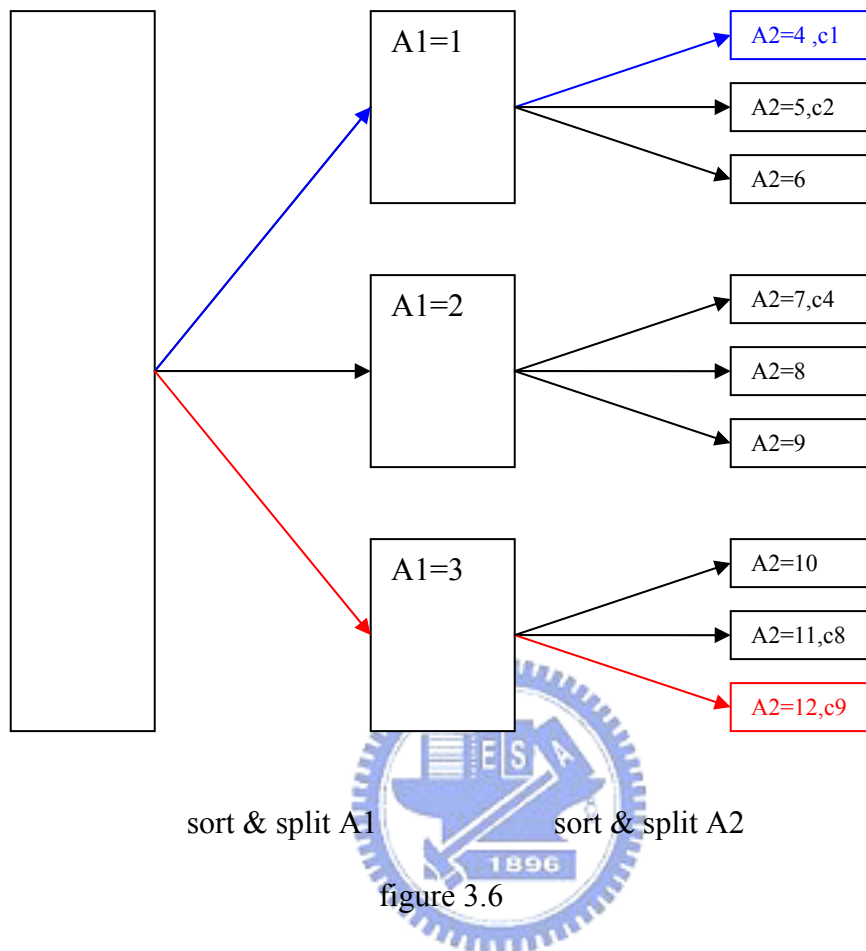
A simple word to describe our method is nonrecursive decision tree. It is not necessary to calculate the information gain of each node recursively. It only needs to be done once and can complete all partitions. The modified decision tree is a three-step process:

step1. Firstly, the information gain of each attributes are measured to decide the split priority. The

attributes with higher information gain are chosen for the goodness of split. The entropy theory is applied to calculate the training dataset's information gain of each attribute. The attribute of appropriate quantity is chosen according to the information gain, and the attribute of each selection is arranged according to the size of the information gain in the form of the data. To sum up the step 1 with one words, prioritizing attribute leads to selecting attribute assignment .

step2. This step is a series of sorting and splitting process. We start the sorting sequence by the attribute of the largest information gain. After sorting, we will incise one set of the same attribute value to observe, and from which we find out the rule of the classification. If the splitted samples are all of the same class, then we find a classification rule. If we can't completely find out all the rules, each subgroup of the second biggest information gain attribute will be sorted again, we have to repeat such action until we find out all the rules. The final classification scheme will be like hierarchical classification system. The following is a sketch of figure hint.





By sorting A1, we incise 3 sets of data, and 9 sets of data by sorting A2. With figure 1, for example, we can find 9 rules:

1. if A1=1 and A2=4 then class=c1
2. if A1=1 and A2=5 then class=c2
- :
4. if A1=2 and A2=7 then class=c4
- :
9. if A1=3 and A2=12 then class=c9

step3. The extracted rules are applied to carry on the classification to the test dataset.

3.3 Secondary classification

3.3.1 Procedure of secondary classification

The secondary classification stage include two major works:

- (1) preprocessing (data transformation)
- (2) classify by the majority voting

Data transformation

In data transformation, the data is transformed or consolidated into forms appropriate for mining. Data transformation strategies can involve the followings:

1.Smoothing, which works to remove the noise from data. Such techniques include binning, clustering, and regression.

2.Aggregation, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.

3.Generalization of the data, where low-level or "primitive" (raw) data are replaced by higher-level concepts through the use of concept hierarchies. For example, categorical attributes, like street, can be generalized to higher-level concepts, like city or country. Similarly, values for numeric attributes, like age, may be mapped to higher-level concepts, like young, middle-aged, and senior.

4.Normalization, where the attribute data is scaled so as to fall within a small specified range, such as -1.0 to 1.0, or 0.0 to 1.0.

5.Attribute construction (or feature construction), where new attributes are constructed and added from the given set of attributes to help the mining process.

The goal of data transformation is to build the majority voting table. The transformation

algorithm is to aggregate the pair relation of genes to the cluster relation. The table3.4 shown the the pair relation of some genes. The table3.5 shown the the cluster relation of some genes

table3.4

G234385	G234610
:	:
G234971	G234385
:	:
G234385	G235490
:	:
G239922	G235390
G235716	G239922
G239922	G235300
G239922	G235828
G239922	G235831
G239922	G235511
G239922	G235356
G239922	G235716

table3.5

gene	Interaction genes	Interaction gene location	predicted location
G234385	G234610,G234971,G235490	10,10,11	10
G239922	G235300,G235356,G235390,G235511,G235716,G235828,G235831	5,5,5,5,3,2,11	5

3.3.2 Secondary classification preprocessing

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 increase 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 the strength into account of each interaction. In this way, we generated binary pairs that described the interactions between genes, and we called this “the binary interaction relationship”.

With the help of binary interaction relationship we can construct the majority voting table. Actually the constructing method is the data transformation process. Build up the majority voting table from the primary classification survived test genes and the Interactions_relation.data.

The Transformation Algorithm:

1. Load one of the pending test genes represent with the “p_t_g”
2. Load one record of Interactions_relation.data.
3. halt if it reaches the end of pending test genes.
4. Apply String Searching algorithm to map the string “p_t_g” and the record of interactions_relation.data if the record hold the “p_t_g” the other gene will be taken.
5. if it comes to the end of interactions_relation.data, then we go to step 1
6. go to step 2.

3.3.3 Elect the candidate location

From the biological viewpoint, proteins interact and work together to achieve certain functions at a specific location in the cell. Thus, infer the location of one gene with binary interaction relation and the majority vote will be feasible. For example the exiting binary interaction relations of the test gene G234385 are the genes G234610,G234971 and G235490, the genes G234610,G234971 live in the location 10 and the location of the gene G235490 is 11. The location 10 is the majority, we elect the location 10 as the predicted location. Another test gene G239922 has 4 candidate locations 5,3,2 and 11, the majority location is 5, thus the test gene G239922 is classified to location 5. Select every locality of the test gene in majority voting table then we can count the accuracy for the secondary classification.

Chapter 4

What the winner did [15]

4.1 Introduction

The winners used 3 traditional approaches, such as decision trees, AdaBoost, and nearest neighbor methods. The nearest neighbor method represented the most promising approach because majority voting is used to predict multiple classes, and space reduction to handle missing values, which appeared to be straightforward, although the selection of optimal distances from the data required serious investigation. To overcome this problem, they calculated an "optimal" neighborhood. Thereby, it maximized the prediction accuracy against a test dataset (a subset of the given training dataset). Although this optimization problem is computationally intractable, they 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.



4.2 Coping with missing value

The winner developed an approach for handling missing values, thereby, allowing more accurate predictions. First, they 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, they 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.

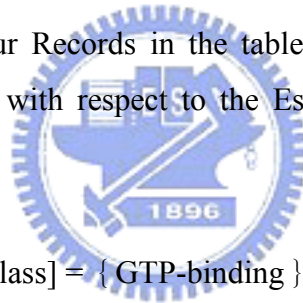
They also noted that although many attribute values for the 862 training genes were missing, at least one of the three attributes (Class, Complex, or Motif) was usually defined. Indeed, among the 381 test genes, 367 had one of these three values or interacted with other genes. Therefore, they decided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

4.3 Nearest neighbor analysis

4.3.1 Attribute agreement of records

Let R be a relation that is a set of records with which certain attributes (features) f_1, f_2, \dots, f_k are associated. They assume here that R has two special attributes. 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 . They then denote the f_i 's value in r by $r[f_i] \in D_j$. Without loss of generality, they assume that $r[f_i]$ is a set of values. In the case where the f_i 's value of r is a single value c , they formally regard it as the singleton set $\{c\}$, but they 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, $r_1[f_i] \cap r_2[f_i] \neq \emptyset$

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



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

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 they 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 4.2 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 4.1.

table 4.1

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	{ Non-Essential }	{ GTP/GDP-excange }	{ Translation }	{ PS00824,PS00825 }	1	cytoplasm
G235357	{ Non-Essential }	{ }	{ Translation }	{ PS00017,PS00190 }	7	mitochondria

table 4.2. Binary interaction relation between pairs of gene

Gene	Gene	Type	strength
G234064	G234126	Generic-Physical	0.914095071
G234064	G235065	Generic-Physical	0.751584888
G235357	G239653	Generic	0.891039915

4.3.2 Neighbors

They are now in a position to define neighborhoods among the records. They focus on G234126 in Table I and define its neighbors. They 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, they 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.

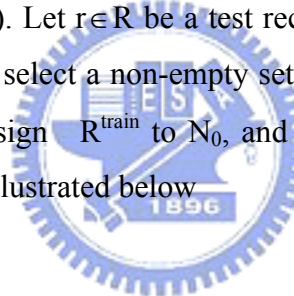
4.3.3 Nearest neighbor assignment by prioritizing attribute

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 they need to examine more than one attribute. For instance, in Table I, one can combine Class, Complex, and Motif attributes together with the binary interactions in Table 2. They 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, they perform a further selection among the neighbors. In order to achieve this, they 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. They then select Class as the secondary attribute, and they extract the genes that agree with G234126 with regard to Class, thus revealing G235065. which is located in the cytoplasm.

They now present a formal description of the method used to restrict neighbors. Suppose that they select some features from f_1, f_2, \dots, f_k , and prioritize them to yield a sequence of features $[g_1, g_2, \dots, g_m]$ that is ordered from left to right (they 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 $R^{\text{train}} \subseteq R$, such that r is not a member of R^{train} . In the initial step, they assign R^{train} to N_0 , and they continue to restrict N_i by using the priority list $[g_1, g_2, \dots, g_m]$, as illustrated below



```


$$N_0 := R^{\text{train}} ;$$

for each  $i=1,2,\dots,m$  begin
   $N_i := \{ x \in N_{i-1} \mid x \text{ and } r \text{ agree on } g_i \}$ 
if  $N_i = \emptyset$  then  $N_i := N_{i-1}$ 
end
return  $N_m ;$ 

```

In each step. they compute the elements in N_{i-1} that agree with r on the next attribute g_i , and they assign the set to N_i . The new set N_j may be empty due to many missing attribute values. In order to avoid losing all the neighbors of r when N_i is empty, they continue the calculation by re-assigning the previous non-empty N_{i-1} to N_i they 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, they call the members of N_m the nearest neighbors of r with respect to the initial training dataset R^{train} and the priority list $[g_1, g_2, \dots, g_m]$. They denote the final answer N_m as: $NN(r, R^{\text{train}}, [g_1, g_2, \dots, g_m])$.

4.3.4 Classification by nearest neighbor analysis

Let obj be an objective attribute, such as Localization, and let D_{obj} be its domain. They calculate the objective value of r , $r[obj]$ from the majority of objective values of nearest neighbors in $NN(r, R^{train}, [g_1, g_2, \dots, g_m])$ using the formula:

$\arg \max_{d \in D_{obj}} | \{ x \in NN(r, R^{train}, [g_1, \dots, g_m]) \mid x[obj] = d \} |$, which is denoted by $\text{predict}(r, R^{train}, [g_1, \dots, g_m])$.

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, $R^{test} \cap R^{train} = \emptyset$ the prediction accuracy of our classification method using the test dataset R^{test} is

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

which is referred to as accuracy $(\frac{| \{ r \in R^{test} \mid r[obj] = \text{predict}(r, R^{train}, [g_1, \dots, g_m]) \} |}{| R^{test} |})$ in the following discussion.

So far we have assumed that a priority list $[g_1, \dots, g_m]$ is provided to 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.

4.4 Computing optimal priority

In this section, the winner 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

$x \in NN(r, R^{train}, [g_1, \dots, g_m])$ 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:

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

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

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

$$\text{ub}([g_1, \dots, g_m, \dots, g_n]) \leq \text{ub}([g_1, \dots, g_m])$$

By using the procedure for computing the nearest neighborhood (shown in Figure 12), it is easy to see that the nearest neighborhood shrinks for extensions:

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

It immediately follows that if r is unpredictable when using $\text{NN}(r, R^{\text{train}}, [g_1, \dots, g_m])$, then r is unpredictable when using $\text{NN}(r, R^{\text{train}}, [[g_1, \dots, g_m, \dots, g_n]])$. 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 they find a priority list P such that

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

From Equations (2) and (3), for any extension Q' of Q $\text{accuracy}(R^{\text{test}}, R^{\text{train}}, Q') \leq \text{accuracy}(R^{\text{test}}, R^{\text{train}}, P)$

This property motivates the winner 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 $[\]$, the winner gradually expands the ensemble of candidate lists and maintains the priority list, say P_{max} which temporarily maximizes the prediction accuracy. In each step, the winner selects a frontier node in the ensemble, and the winner investigates its children (in this

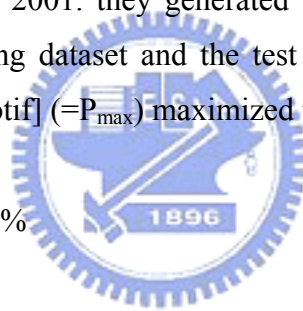
instance Q). If $ub(Q) \leq accuracy(R^{test}, R^{train}, P_{max})$, they can safely prune the subtree rooted at Q without fear of losing the optimal solution.

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.

4.5 Experiment of the winner

Let S^{train} and S^{test} denote the training data and the test data, respectively, of dataset 2, which was provided by the KDD Clip 2001. They 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} ,

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

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.

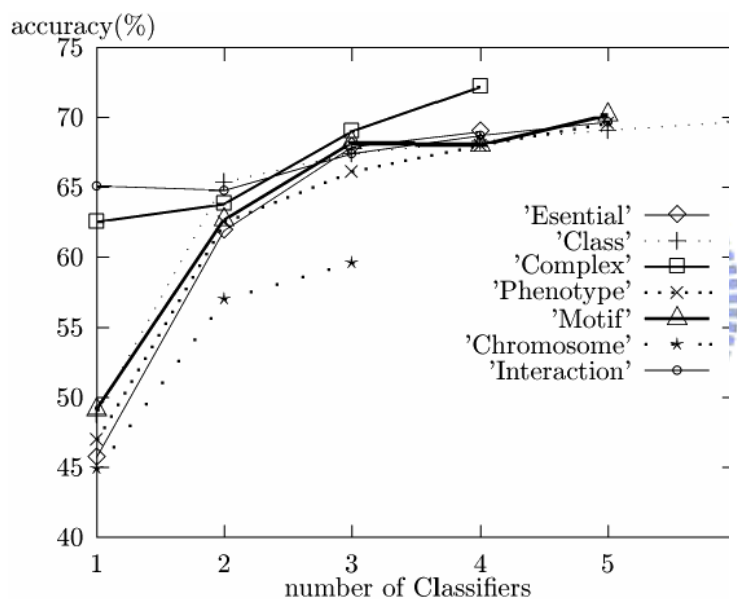
Since all the values of Localization were available, they performed additional experiments. The second column of Table 3 includes the prediction accuracy of the singleton set of each attribute. In order to show that the prediction accuracy improves when attributes are appended to each singleton set, 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 4.1 also illustrates how the prediction accuracy improves step-by-step until the optimal priority list in

Table 4.3 is generated.

table 4.3

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

Figure 4.1 Accuracy improvement by increasing the number of attributes



Chapter 5

Test result and comparison

5.1 primary classification test

In the beginning of this stage, we will strip the redundant attributes (data reduction) by feature selection methods. The best choice is entropy theory. We compute the information gain of each attribute, the complex is the highest and the class is second highest, the others have a large gap to the complex and the class. Therefore, we choose the two features as the key attributes. Sorting the complex and splitting the training data, we can extract some classification rules. For clarification, we incise a part of sorted training data and then find a rule “if complex=3 (Alpha, alpha-trehalose-phosphate synthase). The location is 2 (cytoplasm)”. The unsorted order of gene G235143, G235690 and G235741 are 103, 453 and 454. After sorting the attribute “complex”, the 3 genes are arranged together the rule shows up naturally. Eliminate the 3 training genes, the training instances left 859 records (862-3=859). Similarly the gene G235123, G235241, G235772 and G235349 make the second rule “if complex=4 then location is 11”. Discard the gene G235123, G235241, G235772 and G235349 left 855 (859-4) records. Last sorting and splitting the training dataset the undisposed training genes decreasing, the classification rules increasing. There are many of training instances can't be classified due to missing value.

Table 5.1

unsorted order	Gene	complex	class	essential	motif	Chromosome	location
451	G235245	1	0	2	189	4	10
452	G235171	2	0	2	0	3	10
103	G235143	3	0	1	0	2	2
453	G235690	3	0	2	0	13	2
454	G235741	3	0	2	213	13	2
104	G235123	4	0	1	0	2	11
105	G235241	4	0	1	0	4	11
106	G235772	4	0	1	66	14	11
455	G235349	4	0	2	0	6	11
456	G235885	5	0	2	147,148	16	2

457	G235859	6	0	2	442	15	2
460	G235463	7	0	2	0	8	5
461	G235575	7	0	2	0	11	5
458	G235200	7	0	2	0	4	15
459	G235234	7	0	2	0	4	15
788	G234763	8	21	2	0	12	2
789	G234666	8	21	2	18	11	2
790	G234772	8	21	2	125	13	2
462	G235485	9	0	2	888	9	2
773	G234594	9	20	2	107,108	10	2
774	G234659	9	20	2	63,107,108	11	2
777	G234413	10	20	2	1101	7	11
778	G234935	10	20	2	1101	15	11
779	G234535	10	20	2	107,108	9	11
783	G234936	10	20	3	107,108	15	11
775	G234522	10	20	2	107	8	13
776	G234868	10	20	2	108	14	13
852	G234572	12	4,13	1	751	10	2
853	G234553	12	4,13	1	750	9	2
652	G234824	13	5	2	0	13	2
653	G234073	13	5	2	13	1	2
654	G235044	13	5	2	17,292	16	2
745	G234185	13	18	2	0	3	11
759	G234206	13	20	1	107,108	4	11
780	G235004	13	20	2	108	16	11
781	G234652	13	20	2	107,108	11	11

. After transacting the attribute “complex”, the sorting and splitting work will shift to the attribute “class”. Then we can finish the primary classification work. All of the classification rules that can be extracted are as follows:

table 5.2 classification rules

Location	Complex	Class
1 (cell wall)	no rule	no rule

2 (cytoplasm)	Alpha, alpha-trehalose-phosphate synthase (3) Arginase (5) Arginine-specific carbamoylphosphate synthase (6) Calcineurin B (8) cAMP-dependent protein kinase (9) Chaperonine containing T-complex TRiC (TCP RING Complex) (12) Exocyst complex (17) "Fatty acid synthetase, cytoplasmic (18) " Gim complexes (19) Histone acetyltransferase complexes (24) Kel1p/Kel2p complex (28) L-aminopadipate-semialdehyde dehydrogenase (30) Nonsense-mediated mRNA decay pathway complex (33) Phosphofructo kinase (37) Prenyltransferases (38) Pyruvate kinase (42) Serine/threonine phosphoprotein phosphatase (M 49) Translation complexes (M 55)	Cyclins (5) GTP/GDP dissociation inhibitors (GDIs) (8) Protein hosphatases (21)
3(cytoskeleton)	Cytoskeleton (14) SCF (Skp1-Cdc53-F-box protein) complexes (48)	Actin related proteins (1)
4 ()	no rule	no rule
5 (ER)	Translocon (M 56)	no rule
6,7,8,9 ()	no rule	no rule
10 (mitochondria)	2-oxoglutarate dehydrogenase (1) Acetolactate synthase (2) Mitochondrial translocase complex (31) Proteases, mitochondrial (39) Respiration chain complexes (44)	(ATPases, Proteases)
11 (nucleus)	Anaphase promoting complex(APC) (4) Casein kinase (M 10) Cyclin-CDK (Cyclin-dependent kinases) complexes(M 13) Kinetochoe protein complexes (29) Nem1p-Spo7p complex (32) Nuclear pore complex(NPC) (34) Nucleosomal protein complex (35) Proteasome (40) Replication complexes (43) RNA processing complexes (M 46) RSC complex (Remodel the structure of chromatin) (47) Sister chromatid cohesion regulation complex (50) Synaptonemal complex (SC) (53) Transcription complexes/Transcriptosome (54)	Histones (11) Polymerases (M 18) Transcription factors(M 22)
12 (peroxisome)	delta3-cis-delta2-trans-enoyl-CoA isomerase (15)	no rule
13,14 ()	no rule	no rule
15 (vacuole)	H ⁺ -transporting ATPase,acuolar (22)	no rule

Applying the rules to the test data can classify 179 test genes correctly, 11 test genes are assigned to a wrong location. The accuracy of this stage is 46.98%. In the secondary classification stage, we will cope with the rest of 191 (381-179-11=191) test records that could not be predicted.

The ambiguous classification rules

Notice the classification rule table carefully some rules notating “M”. It means that these classification rules are not perfect. Because we generate these rules by way of majority voting. For example the rule “ if complex=Casein kinase (10) then location=11 ” is derived from the following table:

table 5.3

gene	complex	class	location
:	9	20	10
G234535	10	20	11
G234935	10	20	11
G234413	10	20	11
G234936	10	20	11
G234522	10	20	13
G234868	10	20	13
:	11	20	2



Most of them have high prediction accuracy, though these rules are impurity.

5.1.1 Verification of the classification rules by the Modified Windowing technique

The inspiration of this verification way comes from “Windowing”, we change the original Windowing steps as follows:

1. According to location the whole set of training instances is divided into the 15 subsets or windows.
2. If end of window then halt.

3. Use the nonrecursive decision algorithm to find the rules to the selected window.
4. Scan through all of the rest training instances looking for exceptions to the rule.
5. Add the exceptions to the window.
6. Go to next window.

Repeat steps 2 to 6 until there are no windows left. The resultant classification rules are the same as table 5.2.

5.2 Secondary classification test

In the primary classification we killed 190 (179+11=190) test genes but there were 191 test genes(381-190=191) still unclassified. This stage will treat the 191 survived test genes. We apply the data transformation approach of section 3.2.1 to transform the Interactions_relation.data to the majority voting table.

table 5.4 Part of Interactions_relation.data

G234385	G234610
:	:
G234971	G234385
:	:
G234385	G235490
:	:
G239922	G235390
G235716	G239922
G239922	G235300
G239922	G235828
G239922	G235831
G239922	G235511
G239922	G235356
G239922	G235716

Table 5.5 part of majority voting table

gene	Interaction gene	Interaction gene location	predicted location
G234385	G234610,G234971,G235490	10,10,11	10
G239922	G235300,G235356,G235390,G235511,G235716,G235828,G235831	5,5,5,5,3,2,11	5

Elect the candidate location by the majority vote:

For example, the gene G234385 may have interactions relation with the gene G234610, G234971 and G235490. We judge the gene G234385 should live in location 10 (mitochondria) as well as the gene G239922 should follow the majority voting rule and gather together with the gene G235300, G235356, G235390 and G235511. After electing the candidate locations of the 191 test genes, we can get the classification accuracy of 26.77%.

table 5.6

gene	Interaction gene	Interaction gene location	predicted location
G234385	G234610,G234971,G235490	10,10,11	10
G239922	G235300,G235356,G235390,G235511,G235716,G235828,G235831	5,5,5,5,3,2,11	5

The weird attribute values

Some training and test records have multiple attribute values in a single field. For example the attribute “complex” of the test gene G234570 has two values “ Cyclin-CDK (Cyclin-dependent kinases) complexes (13) and Transcription complexes/ Transcriptosome (54) ”. The predictive location is 11 (nucleus) . Because the gene with the unique “complex” value case and the value is 13 or 54, the gene will locate in 11 . This outcome have already presented in section 5.1.2. Or the joint location of the single “complex” value gene is 11. The other gene G235121 has two “complex” values 14 and 26. Go back to first stage after sorted the attribute “complex”, we separate all of the training genes with the “complex” value=14 from the 862 training instances to observe. The possible locations is 2,3 and 11. If we do the same work to the “complex” value=26. the probable locations is 2,5,7 and 14. The joint location of the two possibilities is 2. Choosing the joint location manually for the multiple attribute values genes is a

easy and fast way to open the small hitch.

table 5.7

gene	complex	class	essential	motif	CH	location
G234570	13,54	5	2	0	1	11
G235055	13,54	5	1	0	16	11
G235695	13,54	0	2	0	0	11
G235121	14,26	0	2	0	2	2
G235778	14,26	0	2	0	14	2

5.3 Over all test result and comparison

After putting primary and secondary classification results together, we get results the accuracy for $46.98\%+26.77\% = 73.75\%$ exceed the winner of task 3 (72.2%) and our method take fewer attributes. The task 3 winner's conclusion say "Data mining machinery automatically selects biologically meaningful four attributes." but our method only use three attributes. The task 3 winner's conclusion also say "The step of handling missing values was most elaborated and time-consuming."; our method also keeps away this problem. In the primary classification stage, the nonrecursive decision tree only classify the genes which have not critical missing values. In the secondary classification stage, the majority voting method compensate the lost information of the missing attribute values by the binary interaction relationship. That is, our method does not fritter away time to deal with missing values. Although our method must increase once data preprocessing time, the time of the increment is not so much as dealing with missing values. Therefore, apply primary-secondary classification on a blend of different type data, may acquire a good classification effect. The comparison table of Nearest neighbor、our method and decision tree is shown below

table 5.8

Method	The determination methods of attribute importance	Feature selection?	main classification method	Produce Classification rules ?
Nearest neighbor	information theory	Eliminate low information gain attributes	Calculating every weight of training data to the test data and prioritizing the weights of the test data then	no

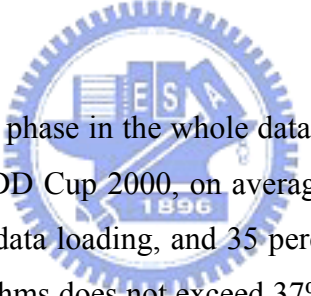
			classify the test data by majority vote.	
primary -secondary classification	information theory	Eliminate low information gain attributes	According to the information gain size then sort and split the attributes hierarchically in primary classification , in secondary classification the test data is classified by majority vote.	yes
decision tree	information theory	Don't Eliminate any attributes	Recursively find out the test node and partition training data set, until each leaf node is the same class	yes



Chapter 6

Conclusion

Since the rapid progress of biotech, data mining on Bioinformatics has attracted people from various research communities such as database, machine learning, statistics, etc. For those researchers who have novel machine learning algorithms, the most effective way to examine the performance of their algorithms as well as improve their practical experiences is to go into biotechnology problems. We can achieve this goal by directly cooperating with Genetic engineer. In addition, it is also a good idea to participate in some data. mining competitions. Currently, there are many data mining competitions hosted by academic or commercial organizations. KDD Cup 2001 is one of the most prestigious contests. Not only academic people, but many data mining companies also appear in the participants list. Through the competition, we are able to know other outstanding data miners, and acquire many experiences after overcoming the difficulties.



Data preprocessing is a vital phase in the whole data mining process. According to what the organizers have announced in KDD Cup 2000, on average participants spent 28 percent of their time on data transformation and data loading, and 35 percent on things about data investigation. The part spent on learning algorithms does not exceed 37%. While other experienced data miners devoted equally to preprocessing and learning, we had met many difficulties when training our models without understanding the data at hand.

For large and complex problems, it is a big help to divide into two processing stages. With such strategy, we may drop the complicity of the subject problem with less efforts. Also, the data investigation tasks can be easily done, which will help us to have a better understanding of the data.

A lot of participants of task 3 utilize decision tree algorithm. We think this is because decision tree method works well with text data. We have to transform binary into text in order to fit in the algorithm. Different transformation strategies may result in different models, but what strategy works best remains in doubt. When coming across a big problem, an efficient model selecting mechanism is a must. On the contrary, decision tree is a mature algorithm, which can be optimized by the pruning technique. Therefore, we conclude that we should not insist on using a

particular classification method. The method that performs well for the problem at hand is the best. Besides, the mining work is a procedure which contains many routes which are not a simplex way, we do not emphasize our suggested methods are better than any other single classification algorithm. Our favorite suggestion is another choice or concept of resolving analogous mining problem.



Reference

1. Chih-Jen Lin (2000) : Study on a practical data mining problem
2. <http://www.acm.org/sigs/sigkdd/kdd2001/>
3. <http://vschool.scu.edu.tw/old/TOJE/Biology/Contents/Cytology/細胞學首頁.htm>
4. <http://www.acm.org/sigkdd/explorations/issue3-2.htm>
5. KDD Cup 2001 Report : Jie Cheng, Christos Hatzis, Hisashi Hayashi, Mark-A. Krogel, Shinichi Morishita, David Page, and Jun Sese. SIGKDD Explorations, 3(2):47--64, January 2002.
6. Data Mining: Concepts and Techniques, J. Han and M. Kamber, Morgan Kaufmann, 2000.
7. Horowitz, Sahni, & Mehta, W.H. Freeman, Fundamentals of Data Structures in C++, 1995
8. 類神經網路與圖型識別 黃國源編著 (2000)
9. <http://www.mgt.ncu.edu.tw/~ylchen/datamining.html>
10. http://www.datamining.org.tw/software/soft_content.asp?soft_no=20
11. <http://cindy.cis.nctu.edu.tw/AI96/team10/DTRoot.htm>
13. [Quinlan 79]J. Ross Quinlan. Discovering rules from large collections of examples: a case study. In Michie, D., editor, Expert Systems in the Microelectronic Age. Edinburgh University Press, Edinburgh Scotland, 1979.
- 14.[Quinlan 93]J.Ross Quinlan. C4.5:Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

15. ID3: History, Implementation, and Applications Paul Gestwicki October 4, 1997
(<http://gunther.smeal.psu.edu/correct/7956>)
16. The Machine Learning Dictionary for COMP9414 Bill Wilson, 1998, 2003
(<http://www.cse.unsw.edu.au/~billw/mldict.html>)
17. Salton, G. and McGill, M. Introduction to Modern Information Retrieval , McGrawHill Book Company, New York, NY, 1983.
18. Chen, H, et. al. A Machine Learning Approach to Inductive Query by Exam ples: An Experiment Using Relevance Feedback, ID3, Genetic Algorithms, and Simulated Annealing. University of Arizona Technical Papers, 1995.

