Solving the Fragmentation Problem of Decision Trees by Discovering Boundary Emerging Patterns

Jinyan Li*

Laboratories for Information Technology
21, Heng Mui Keng Terrace, Singapore, 119613
jinyan@lit.org.sg

Limsoon Wong

Laboratories for Information Technology
21, Heng Mui Keng Terrace, Singapore, 119613
limsoon@lit.org.sg

Abstract

The single coverage constraint does not encourage a decision tree to contain many significant rules, and the fragmentation problem can cause a decision tree to contain too many minor rules. The loss of significant rules can lead to a loss in accuracy. The presence of minor rules can also decrease accuracy. We propose to use emerging patterns to solve these problems. In our approach, many globally significant rules can be discovered. Extensive experimental results on gene expression datasets show that our approach are much more accurate than single C4.5 trees, and are also better than bagged or boosted C4.5 trees.

1 Introduction

Decision trees, typically constructed by C4.5 [?], CART [?], or ID3 [?], are widely used in machine learning since their conception in the CLS (concept learning system) framework [?]. A decision tree is a tree-based compact structure consisting of classification rules learnt from training data. There are two shortcomings in these methods based on decision trees. The first is the *fragmentation problem*. At the lower levels of a decision tree, the feature selection procedure uses fewer and fewer training data. This usually leads to the generation of many locally important but globally insignificant rules [?, ?]. The second is the *single coverage constraint*. Every training instance is to be covered by exactly one rule. This usually leads to the loss of important rules. Sometimes, the number of rules contained in a decision tree is far less than the total number of rules inherent to the data, especially for gene expression datasets. Therefore, the accuracy of decision tree classifiers is overall inferior to other classifiers like neural networks and support vector machines (SVMs). Nevertheless, compared to neural networks and SVMs, decision tree classifiers have the advantage that their rules are easily comprehensible. This is particularly important in diagnosing a disease as medical doctors sometimes need to explain their decisions to colleagues and patients.

*Contact author. Laboratories for Information Technology, 21 Heng Mui Keng Terrace, Singapore 119613. Tel. +65-6874-8217. Fax. +65-6774-8056.

This paper proposes an emerging pattern approach [?] to solve these problems by discovering rules that are globally significant in a computationally efficient way. Emerging patterns (EPs) are a relatively new concept in data mining [?]. An emerging pattern is defined as a set of conjunctive conditions that most instances of a class satisfy and none of the other class satisfies, assuming that two classes are in the training data. So an EP is also a classification rule. In this study, we compare EPs with decision trees from basic ideas to advanced topics. We explain why the idea of EPs is not a reinvention of the wheel but is a generalized new concept. Furthermore, the EP approach outperforms decision trees on accuracy in general. In fact, its accuracy is competitive to that of SVMs. At the same time, the EP approach maintains the favorable comprehensibility of the rules derived from decision trees.

Microarray gene expression technology plays an important role in the modern cancer research. This technology simultaneously measures expression levels of tens of thousands of genes, generating extremely high-dimensional numeric data. Many studies have successfully tried this technology to find disease-related genes, to conduct diagnostic classifications, and to make prognostic predictions [?, ?, ?, ?]. In this study, our experiments for comparing the performance of the decision tree and EP approaches are based on gene expression profiles of four different diseases. The discovered rules can help medical doctors understand the mechanism of a disease and provide them novel insights into relevant molecular interactions.

In the next section, we describe basic ideas used in decision trees and emerging patterns, and compare the two approaches. Section 3 discusses a new EP-based classifier called PCL. Section 4 outlines some advanced topics on decision trees and emerging patterns. Section 5 uses a real medical dataset to illustrate our methodology and for comparing the performance of the two approaches. Section 6 reports our extensive experimental results on three additional gene expression datasets. In that section, we show our EP approach can outperform C4.5 (single, bagged, or boosted), and explain why single C4.5 trees and bagged or boosted trees do not work well for expression data. Finally, we conclude the paper with a brief summary.

2 Basic Ideas on Decision Trees and Emerging Patterns

Suppose we are given two sets of relational data where a fixed number of *attributes* (also called *features* or *variables*) exist. Every attribute has a range of numeric real values or a set of categorical values. An *instance* (or a sample) is defined as a set of attribute-value pairs with a cardinality equal to the number of attributes in the relational data. An instance assigned with a class label is called a *training* instance. In contrast, a *test* instance is a fresh data whose class label is to be predicted. Table 1 [?] shows 14 training instances described by four attributes, having two class labels `Play` and `Don't`. There are 9 `Play` and 5 `Don't` instances.

2.1 Decision Trees and C4.5

The construction of a decision tree is a recursive process. The process involves determining the attribute which is *most* discriminatory and then splitting the training instances into groups. Each group contains

Table 1: A small training data set consisting of 14 instances describes weather conditions suitable for or not for an outdoor game.

Outlook	Temp	Humidity	Windy	class
$\{sun., overcast, rain\}$	(cont)	(cont)	$\{false, true\}$	$\{Play, Don't\}$
sunny	75	70	true	Play
sunny	80	90	true	Don't
sunny	85	85	false	Don't
sunny	72	95	true	Don't
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't
rain	65	70	true	Don't
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Figure 1: A simple decision tree.

multi-class instances or single-class instances, as categorized by this attribute. Next, a significant attribute of each of the subsets (groups) is used to further partition them and the process is repeated recursively until all the subsets each contain data of one class only. The resulting structure is called a decision tree, each node of which is an attribute discrimination test and each branch of which is a subset of the collection of training instances satisfying the test. We use Figure 1 to show the structure of a decision tree. This tree is induced by C4.5 according to the training data set in Table 1.

The main components of the tree—*leaves* and *nodes*—have distinct meanings. A leaf indicates a class name (e.g., `edible`, `poisonous`, `play`, `don't play`) that is described in the training data. A node specifies a test to be carried out on a single attribute value with one branch and a subtree for each possible outcome of the test [?]. The test on the root node in Figure 1 is on the condition "`outlook = sunny, overcast, or rain?`" A decision tree can be used to classify an instance by starting at the root of the tree and moving through it until a leaf is encountered. The class of the instance is predicted to be that stored at the leaf. The decision tree shown in Figure 1 has 100% accuracy on the 14 training instances.

The key to constructing a decision tree is how to choose an appropriate attribute as the root node, and subsequently some other attributes as the other nodes. Therefore, for a training set, many alternative trees can be constructed by different feature selection metrics. Preferably, a tree should have a small number of leaves such that rules derived from it can be more readable and at the same time more reliable. Less leaves means that, on the average, there are more instances per leaf. This gives the derived rules a better statistical confirmation. However, finding optimal decision trees is known to be an NP-complete problem [?]. C4.5 [?] is a heuristic algorithm for inducing decision trees. C4.5 uses an entropy-based selection measure to determine which attribute is most discriminatory. This measure is also called *maximum entropy*, *maximum information gain*, or *mutual information*. Most decision trees in the literature are constructed by this method.

In general, decision trees have the following characteristics:

- Single coverage of training data. Every instance in the training data must satisfy one and only one of the rules in the decision tree. The total number of instances covered by all leaves must be equal to the number of instances in the original data.
- Divide-and-conquer splitting strategy and the fragmentation problem. In order to discover mutually exclusive parts of the original training data, C4.5 uses a top-down successive subdivision idea to recursively partition the training data. This idea is called the divide-and-conquer strategy. This strategy causes the *fragmentation problem* because towards the end of the subdivision process, the size of the underlying data becomes quite small, even though a statistical test requires a data of significant size [?, ?]. This is a problem that cannot be avoided in top-down greedy search.
- Locally reliable but globally insignificant rules. Due to the fragmentation problem, C4.5 trees always contain many "minor" rules. For example, the rule "if `outlook` = `sunny` and `humidity` $\leq$ `75` then `Play`", has only 22% (2 out of 9) coverage over all `Play` instances. Yet this rule has a 100% coverage (2 out of 2) over those `Play` instances that satisfy the condition "`outlook` = `sunny`". The pruning idea of C4.5 does not help this. These locally reliable but globally insignificant rules may mislead the tree and thus decrease the classification performance on independent test data.

Next, we discuss emerging patterns. Emerging patterns can have multiple coverage over training data. They are also discovered globally in a computationally efficient way, and thus the fragmentation problem does not occur to them.

2.2 Emerging Patterns

To define EPs, we need some basic terms. A *condition* (also called an *item*) is defined as a pair of an attribute and its a value. An example of a condition (an item) is "the expression of *gene_x* is less than 1000". We denote this condition by $gene\_x@(-\infty, 1000)$, where the attribute is *gene_x* and its value is $(-\infty, 1000)$. A *pattern* is simply a set of conditions. A pattern is said to *occur* in an instance if the instance contains this pattern. For two classes of instances, a pattern can have a very high occurrence (equivalently, frequency) in one class, but can change to a low or even zero occurrence in the other class.

We first give a very broad definition of emerging patterns to reflect the frequency change of patterns. Then we will narrow the space to focus on a special type of EP—boundary EPs.

Definition 2.1 Given two classes of data, an *emerging pattern* is a pattern whose frequency in one class is non-zero but in the other class is zero.

Usually, the class in which an EP has a non-zero frequency is called the EP's *home* class or its own class. The other class in which the EP has the zero frequency is called the EP's *counterpart* class [**?**].

Many EPs may have very low frequency (e.g. 1 or 2) in their home class. So boundary EPs are proposed to capture highly reliable rules for regulating two classes:

Definition 2.2 A *boundary* EP is an EP whose proper subsets are not EPs.

How do boundary EPs capture big differences? Clearly, if a pattern contains fewer items (conditions), then the frequency (probability) that it occurs in a class becomes larger. Removing any one item from a boundary EP thus increases its home class frequency. However, by definition of boundary EPs, its frequency in the counterpart class becomes non-zero. Therefore, boundary EPs are maximally frequent in their home class. They separate EPs from non-EPs. They also distinguish EPs with high occurrence from EPs with low occurrence. Efficient discovery of boundary EPs has been solved in our previous work [**?**, **?**].

Using the training data shown in Table 1, we discover boundary EPs between the `Play` and `Don't` classes. The two numeric attributes, *temperature* and *humidity*, need to be discretized. We discretized the two attributes into two intervals using the middle point of the ranges (cutting points 75 and 80 respectively for temperature and humidity) [1]. Table 2 reports the ranked boundary EPs and their occurrence and coverage in the `Play` and `Don't` classes. For comparison, we list the rules in the decision tree shown in Figure 1 using Table 3.

We can see that there are 4 significant EP rules (EP no. 1, 2, 3 and 8) but only two significant C4.5 rules (rule no. 1 and 4). By significant, we mean the frequency of the rule is $\geq 44\%$ (number one C4.5 rule's frequency) in the `Play` class or $\geq 60\%$ (also number one C4.5 rule's frequency) in the `Don't` class. Especially, the first EP rule $\{Humi \leq 80, windy = false\}$ with the highest coverage 5 in the `Play` class is a rule that was not discovered by C4.5. We also found that the EPs can have multiple coverages over the training data (see the last row of Table 2).

In general, the EP approach has the following characteristics:

- A cluster of trees. Each EP is a tree with only one branch. Some EPs can be integrated into a bigger tree. For example, the first and seventh EPs in Table 2 can be combined into a tree with the root

[1] A sophisticated, entropy-based discretization method discretized the two attributes into one bin each, having no cutting points in the ranges. As the attribute humidity is used in the contrustion of the tree, we request it to be discretized for rule comparison.

Table 2: By our EP approach, a total of 12 rules are derived from the weather condition toy dataset. Some important ones cannot be discovered by C4.5.

EP No.	EPs	Occurrece (Coverage) in `Play` class	Occurrence (Coverage) in `Don't` class
1	$\{Humi \leq 80, windy = false\}$	5(56%)	0
2	$\{Temp \leq 75, windy = false\}$	4(44%)	0
3	$\{outlook = overcast\}$	4(44%)	0
4	$\{outlook = rain, windy = false\}$	3(33%)	0
5	$\{outlook = sunny, Humi \leq 80\}$	2(22%)	0
6	$\{Temp > 75, Humi \leq 80\}$	2(22%)	0
7	$\{outlook = rain, Humi > 80\}$	1(11%)	0
8	$\{outlook = sunny, Humi > 80\}$	0	3 (60%)
9	$\{Temp > 75, Humi > 80\}$	0	2 (40%)
10	$\{outlook = sunny, Temp > 75\}$	0	2 (40%)
11	$\{outlook = rain, windy = true\}$	0	2 (40%)
12	$\{Temp > 75, windy = true\}$	0	1 (20%)
Total	12	21	10

Table 3: Five rules of the tree in Figure 1 in a tabular form.

Rule No.	C4.5 rules	Occurrece (Coverage) in `Play` class	Occurrence (Coverage) in `Don't` class
1	$\{outlook = overcast\}$	4(44%)	0
2	$\{outlook = rain, windy = false\}$	3(33%)	0
3	$\{outlook = sunny, Humi \leq 75\}$	2(22%)	0
4	$\{outlook = sunny, Humi > 75\}$	0	3 (60%)
5	$\{outlook = rain, windy = true\}$	0	2 (40%)
Total	5	9	5

node containing the test on "`Humi > 80?`", and with two branches. Similarly, the second and sixth EPs can be combined into a bigger tree. The remaining three EPs (i.e., EP no. 2, 3, and 4) can be also combined into another bigger tree. However, the three bigger trees cannot combined into a single tree.

- Globally siginificant rules. Boundary EPs are aimed to differentiate one class from another by using as many instances as possible. The distinction is in terms of a whole class. However, some C4.5 rules are discovered based only on a fraction of a class and a fraction of another class. Globally, those locally discovered C4.5 rules may lose their significance. However, all top-ranked EPs are all globally important rules.

- Exponential number in size. By the greedy heuristic and the single coverage constraint, C4.5 always produces decision trees with small numbers of rules. In contrast, the number of boundary EPs can increase exponentially when the number of attributes and the number of discretized values for the attributes increase. This problem can be alleviated by selecting the most important attributes for EP discovery. Due to the removal of the less important attributes, many EPs are consequently missed. However, the most important EPs are still maintained.

Top-ranked boundary EPs have very strong discriminating power for differentiating both training and test data. The next section discusses our recently proposed EP-based classifier. This classifier is named PCL (Prediction by Collective Likelihood of emerging patterns) [?]. The performance between PCL and C4.5 (including Bagged and Boosted) are extensively compared in later sections.

3 The PCL Classifier

A test instance should contain many top-ranked EPs from its own class and contain a few low-ranked EPs—preferably none—from its opposite class. However, in some rare situations, a testing sample can contain some top-ranked EPs from its counterpart class. Therefore, for more reliable prediction, it is reasonable to use multiple highly frequent EPs of the home class to avoid the confusing signals from counterpart EPs.

PCL [?] is a heuristic for aggregating the discriminating power of multiple EPs. Basically, the PCL classifier has two phases. Given two training datasets $\mathcal{D}_P$ and $\mathcal{D}_N$ and a test sample T, the first phase of PCL is to discover two groups of boundary EPs from $\mathcal{D}_P$ and $\mathcal{D}_N$. Denote the ranked EPs of $\mathcal{D}_P$ as,

$$EP\_P\_1, EP\_P\_2, \cdots, EP\_P\_i,$$

in descending order of their frequency. Similarly, denote the ranked boundary EPs of $\mathcal{D}_N$ as

$$EP\_N\_1, EP\_N\_2, \cdots, EP\_N\_j$$

also in descending order of their frequency.

Suppose the test sample T contains the following EPs of $\mathcal{D}_P$:

$$EP\_P\_i_1, EP\_P\_i_2, \cdots, EP\_P\_i_x,$$

where $i_1 < i_2 < \cdots < i_x \leq i$, and the following EPs of $\mathcal{D}_N$:

$$EP\_N\_j_1, EP\_N\_j_2, \cdots, EP\_N\_j_y,$$

where $j_1 < j_2 < \cdots < j_y \leq j$.

The next step is to calculate two scores for predicting the class label of T. Suppose we use k ($k \ll i$ and $k \ll j$) top-ranked EPs of $\mathcal{D}_P$ and $\mathcal{D}_N$. Then we define the score of T in the $\mathcal{D}_P$ class as

$$score(T)\_\mathcal{D}_P = \sum_{m=1}^{k} \frac{frequency(EP\_P\_i_m)}{frequency(EP\_P\_m)},$$

and similarly the score in the $\mathcal{D}_N$ class as

$$score(T)\_\mathcal{D}_N = \sum_{m=1}^{k} \frac{frequency(EP\_N\_j_m)}{frequency(EP\_N\_m)}.$$

If $score(T)\_\mathcal{D}_P > score(T)\_\mathcal{D}_N$, then T is predicted as the class of $\mathcal{D}_P$. Otherwise it is predicted as the class of $\mathcal{D}_N$. We use the size of $\mathcal{D}_P$ and $\mathcal{D}_N$ to break tie.

The spirit of our proposal is to measure how far the top k EPs contained in T are away from the top k EPs of a class. Assume $k = 1$, then $score(T)\_\mathcal{D}_P$ indicates whether the number one EP contained in T is far from the most frequent EP of $\mathcal{D}_P$. If the score is the maximum value 1, then the "distance" is very close, namely the most common property of $\mathcal{D}_P$ is also present in this testing sample. With smaller scores, the distance becomes further. Thus the likelihood of T belonging to the class of $\mathcal{D}_P$ becomes weaker. Using more than one top-ranked EPs, we utilize a "collective" likelihood for more reliable predictions. For example, suppose $k = 10$ and T is a positive sample, then the ideal situation is $score(T)\_\mathcal{D}_P = 10$ and $score(T)\_\mathcal{D}_N = 0$. This means that all the top 10 EPs of $\mathcal{D}_P$ are present in T but no top-ranked EPs of $\mathcal{D}_N$ is present. Then our prediction of T as a positive instance is of high confidence.

The selection of k is proportional to the size of EPs discovered. We suggest to use small numbers like 5, 10, 15 if small number of EPs are discovered (< 500) otherwise set k as 20, 25, and 30 in this work.

4 Advanced Topics on Emerging Patterns and Decision Trees

In the development of C4.5 and EPs, some advanced techniques have been proposed. For C4.5, decision committee techniques such as AdaBoost [?] and Bagging [?] have been proposed to reduce the error rate of single trees by voting the member decisions of the committee [?, ?]. For emerging patterns, a representation tool, borders, has been proposed to concisely represent EP spaces because every EP space has the nice property of convexity. How to efficiently discover boundary EPs is another interesting problem. All these advanced topics are briefly discussed in this section.

4.1 Bagging, Boosting, and C4.5

Basically, AdaBoost and Bagging both apply a base classifier (e.g., C4.5) multiple times to generate a committee of classifiers. Each member classifier play a particular role in classifying a test instance. Assume a training data of N instances be given, and a R number of repetitions or *trials* of the base classifier be applied.

By the bagging idea, for each trial $t = 1, 2, \cdots, R$, a bootstrapped training set is generated from the original instances. This new training set is the same size as the original data, but some instances may not appear in it while others appear more than once. Denote the R bootstrapped training sets as $B_1, B_2, \cdots, B_R$. For each B_t, a classifier C_t is built. A final, bagged classifier C^* is constructed by aggregating $C_1, C_2, \cdots,$ and C_R. The output of C^* is the class predicted most often by its sub-classifiers, with ties broken arbitrarily.

Similar to bagging, the boosting method also uses a committee of classifiers for classification by voting. However, the construction of the committee of classifiers by boosting is different from by bagging. Bagging builds the individual classifiers separately; but boosting builds them sequentially and each new classifier is influenced by the performance of those built previously. Those data incorrectly classified by previous models are always emphasized in the new model with an aim to let the new model become an expert for classifying those hard instances. Another difference is that boosting weights the individual classifiers' output depending on their performance, rather than giving equal weight to all the committee members. AdaBoost.M1 [**?**] is a very good example of the boosting idea.

4.2 Convexity of EP Spaces and Border-based Discovery of Boundary EPs

The number of EPs can be very large. However, an EP space—the collection of all EPs—can be concisely represented by some special member EPs. All other EPs can be directly generated from the special EPs. This property of EP spaces is called convexity [**?**]. That is, every EP space is a convex space.

The special EPs are called boundary EPs. Boundary EPs can be distinguishable using the most general EPs and the most specific EPs. The former builds a left bound and the latter forms a right bound so as to delimit a pattern space. Right boundary EPs can be easily discovered. However, efficient discovery of left boundary EPs is a challenging problem. In our previous work, several border-based algorithms have been proposed to find the most general EPs [**?**, **?**]. Experimental results show that the algorithms are much more efficient than naive algorithms. In this work, we focus on left boundary EPs.

An EP space can be decomposed into a series of plateau-spaces and a small convex space [**?**]. This series of plateau-spaces can be sorted according to their frequency. Interestingly, the one with the highest frequency is a version space [**?**, **?**] if its member EPs have the full 100% frequency in their home class. Usually, this plateau space is empty in practice. But other plateau-spaces can contain many EPs.

Figure 2: A decision tree induced by C4.5 from the prostate disease dataset consisting of 102 gene expression profiles.

5 A Comparison Using a Real Medical Dataset

In this section, we use a recently published gene expression dataset [?] for a comparison between C4.5 and PCL. The data was used to predict common clinical and pathological phenotypes relevant to the treatment of men diagnosed with prostate cancer. This dataset consists of 52 `Tumor` and 50 `Normal` instances. All instances are described by 12600 attributes, each representing the expression level of a particular gene.

Using the 102 instances, C4.5 constructs a decision tree with 5 leaves, as shown in Figure 2. We found that of the 5 rules each corresponding to a leaf, 3 rules are not globally significant. These 3 rules correspond to the 3 middle leaves and have a coverage of no more than 6 instances each. For example, the condition that "32598_at $>$ 29 and 40707_at $<= -6$" is satisfied with only 3 `Tumor` and 1 `Normal` instances. The coverage of this rule is very low.

However, using the EP approach, we discovered many significant rules. Table 4 reports 20 top-ranked boundary EPs together with their high coverage over the data. The EPs can be understood in the following way. For example, the first EP $\{9, 36\}$ in Table 4 says that if "the expression of gene 37720_at > 215 and the expression of 38028_at ≤ 12", then 73% `Tumor` instances satisfy the condition but no samples from `Normal` class satisfies. Here the reference number 9 represents 37720_at > 215 and the number 36 represents 38028_at ≤ 12.

The EP discovery process is as follows. First, use the entropy-based discretization method to discretize the 12600 attributes. Then, select 20 most important discretized attributes such that they have the lowest entropy values. Then, remove the remaining attributes from the original data. Finally, use border-based algorithms [?, ?] to discover the boundary EPs.

We found that the four features (genes) selected by C4.5 have only one gene (32598_at) in common with the top 20 genes selected by our method. Recall that the list of our features are ranked in a global manner. Therefore, it is not surprise to see the reason why we can discover many globally important rules.

The accuracy and error rate of C4.5 and PCL are reported in Table 5. All these experiments are conducted on the discretized data of the 20 most important genes, using the Weka machine learning software package with its standard settings [?]. We can see that the LOOCV (Leave one out cross-validation) and 10-fold CV accuracy of our PCL classifier is higher than C4.5, even after incorporating the Bagging or Boosting ideas. The accuracy of PCL is also very close to that of 3-NN, and much better than SVM. We note that the LOOCV accuracy of the single C4.5 tree, bagged C4.5 trees, and boosted C4.5 trees on the original data (without discretization and removal of features) is 87%, 92%, and 92% respectively with an error rate of 8:5, 3:5, and 3:5 (Tumor:Normal).

Table 4: A partial list of top-ranked boundary EPs discovered from the prostate disease dataset. The reference numbers contained in the patterns each represent a condition. For example, the reference number 9 in the first EP stands for the condition "the expression of gene 37720_at > 215", and similarly for other numbers. We did not include the meaning of all reference numbers due to page limit.

EPs	Occurrence (Coverage) in Tumor class	Occurrence (Coverage) in Normal class
$\{9, 36\}$	38(73.08%)	0
$\{9, 23\}$	38(73.08%)	0
$\{4, 9\}$	38(73.08%)	0
$\{9, 14\}$	38(73.08%)	0
$\{6, 9\}$	38(73.08%)	0
$\{6, 23, 33\}$	37(71.15%)	0
$\{4, 19, 33\}$	37(71.15%)	0
$\{6, 19, 33\}$	37(71.15%)	0
$\{9, 25\}$	37(71.15%)	0
$\{9, 19\}$	37(71.15%)	0
$\{7, 21\}$	0	36 (72.00%)
$\{7, 11\}$	0	35 (70.00%)
$\{7, 43\}$	0	35 (70.00%)
$\{7, 39\}$	0	34 (68.00%)
$\{24, 29\}$	0	34 (68.00%)
$\{11, 24\}$	0	34 (68.00%)
$\{7, 32\}$	0	34 (68.00%)
$\{7, 26\}$	0	33 (66.00%)
$\{5, 7\}$	0	32 (64.00%)
$\{1, 7\}$	0	32 (64.00%)

Table 5: The LOOCV and 10-fold CV accuracy and error rate of four classification models on the prostate disease datasets. Note that error rate $x : y$ means x number of `Tumor` instances and y number of `Normal` instances are mis-classified.

Method	Performance	PCL			C4.5			SVM	3-NN
		$k = 5, 10, 15$			Single	Bagging	Boosting		
LOOCV	Accuracy (%)	95.1	95.1	95.1	91.2	94.1	93.1	90.2	96.1
	Error rate	3:2	3:2	2:3	6:3	4:2	5:2	5:5	1:3
10-fold	Accuracy (%)	97.1	97.1	95.1	92.2	92.2	93.1	90.2	96.1
CV	Error rate	2:1	2:1	2:3	4:4	5:3	4:3	5:5	1:3

6 More Experimental Results

This section reports our experimental results using datasets from three different biomedical research teams. One is from St. Jude Children's Research Hospital, USA [?]. The project is on the subtype classification of the acute lymphoblastic leukemia (ALL) disease using gene expression profiles. Another data set is from Dana-Farber Cancer Institute and Whitehead Institute, USA [?]. The study is also about the subtype classification of the leukemia disease but with more emphasis on the distinction between the MLL subtype and other conventional subtypes. The third dataset is about the relapse study of breast cancer which was obtained by the Netherlands Cancer Institute and Rosetta Inpharmatics, USA [?]. Relapse study is to predict, at the time of initial diagnosis, whether a patient will relapse after receiving treatment.

We first report *test* error rates—the number of misclassifications on independent *test* instances. The error rates are represented in a format of $x : y$ (left class: right class), meaning that x number of instances from left class and y number of instances from right class are misclassified. Single trees constructed by C4.5 and committees of trees derived by the bagging or the boosting methods are used to obtain the test error rates for decision trees. For our PCL classifier, we first discover the 20 most discriminatory features from the training data. Then the training data are projected to these 20 features and discretized. Test instances are also projected and discretized using the same intervals derived from the training data. EPs are discovered from the discretized training data and used to predict the class labels of the test instances. For a fair comparison, we use the same training and test data for SVM and 3-NN to get their test error rates.

We then report 10-fold CV performance on training data of the classification models. As the PCL classifier, the other models are also based on the discretized top 20 features' data.

6.1 Subtype Classification of Childhood Leukemia

Childhood leukemia is a heterogeneous disease, with individual leukemia subtypes differing in their response to therapy [?]. The current existing technology to identify correct subtypes of leukemia is an imprecise and expensive process, requiring the combined expertise from many fields which are not commonly available in a single medical center. Using microarray gene expression technology and supervised classification algorithms, this problem is expected to be solved such that the cost of diagnosis is reduced and at the same time the accuracy of both diagnosis and prognosis is increased.

The whole data consists of 327 gene expression profiles of ALL samples. These profiles were obtained by hydridization on the Affymetrix U95A GeneChip containing probes for 12558 genes. That is, in our terms, the 327 instances are described by 12558 features. The data contain all the known acute lymphoblastic leukemia subtypes, including T-cell (T-ALL), E2A-PBX1, TEL-AML1, BCR-ABL, MLL, and hyperdiploid (Hyperdip>50). The data was divided into a training set of 215 instances and an independent test set of 112 instances by the doctors involved [?]. There are 28, 18, 52, 9, 14, and 42 training instances and 15, 9, 27, 6, 6, and 22 test samples respectively for T-ALL, E2A-PBX1, TEL-AML1, BCR-ABL, MLL, and Hyperdip>50.

Table 6: Error rates of four models on the 112 test instances in the problem of subtype classification of the ALL disease. By in parallel, we mean that the data contains 7 class labels, a prediction is made to be one of the 7 classes (six main subtypes plus the mini-type).

Datasets (test data size in each class)	Error Rates for Test Data							
	PCL			C4.5			SVM	3-NN
	$k = 20, 25, 30$			Single	Bagging	Boosting		
BCR-ABL vs others (6:106)	1:0	1:0	1:0	4:4	6:0	4:4	1:1	1:0
E2A-PBX1 vs others (9:103)	0:0	0:0	0:0	0:0	0:0	0:0	0:0	0:0
HyperL50 vs others (22:90)	2:2	2:2	2:2	4:7	4:2	4:7	0:3	1:4
MLL vs others (6:106)	0:0	0:0	0:0	2:2	1:0	2:2	0:0	0:0
T-ALL vs others (15:97)	0:0	0:0	0:0	0:1	0:1	0:1	0:0	0:0
TEL-AML1 vs others (27:85)	2:0	2:0	2:0	2:2	2:1	2:2	1:1	2:0
minitype vs main (27:85)	11:10	9:12	9:3	10:16	18:0	7:4	9:2	9:1
Parallel (7 classes)	6	7	8	27	20	10	26	11

There are also 52 training and 27 test samples of other miscellaneous subtypes called mini-types here.

In the original approach reported in [?], a hierarchical analysis was applied: one first constructs a model for separating T-ALL from all other subtypes; then one uses the remaining non-T-ALL samples and constructs a model for separating E2A-PBX1 from the remaining subtypes; and so on. In this paper and on this dataset, we consider the classification problem from two different perspectives. Firstly, we ask whether we can distinguish each subtype from all other subtypes using a small number of features. Secondly, we ask whether we can predict in parallel the class label for a new data.

The performance of our classifiers constructed for these perspectives is reported in Table 6. Overall, it can be seen that the classifiers can solve the problem with very low levels of mistakes. According to the table, we also see that the C4.5 single tree approach did not work well for the eight datasets (including the multi-class parallel classification). Secondly, the bagged or boosted C4.5 trees can improve the performance of the single tree approach. For our PCL classifier, its performance is overall better than these three forms of decision trees. Moreover, its accuracy is competitive to SVM and 3-NN, and sometimes better than them.

In the experiments above, C4.5 selected its most important features across the space of all the 12558 features. In the next round of experiments, we let C4.5 select its features within the top 20 features which we considered as most globally important based on entropy. We expect to see the forced feature selection can improve C4.5's accuracy. We found that:

- Single C4.5 trees, bagged trees, and boosted trees all reduced their test error rates in most cases. No case got worse. See Table 7.
- The structure of the resulting trees are quite different from previously constructed trees which are

Table 7: Reduced error rates of C4.5 trees if the feature selection is forced to be in the range of the top 20 most important features.

Datasets (test data size in each class)	Error Rates for Test Data by C4.5 with forced feature selection		
	Single	Bagging	Boosting
BCR-ABL vs others (6:106)	1:4	2:0	1:4
E2A-PBX1 vs others (9:103)	0:0	0:0	0:0
HyperL50 vs others (22:90)	4:5	4:2	1:4
MLL vs others (6:106)	1:1	0:0	1:1
T-ALL vs others (15:97)	0:1	0:1	0:1
TEL-AML1 vs others (27:85)	3:1	1:0	1:0
minitype vs main (27:85)	8:4	13:0	8:0

based on the whole feature space.

- In most cases, our PCL classifier still outperformed the decision tree approach.

Therefore, we can see that the fragmentation problem indeed misled the selection of most important features in decision trees and thus caused the generation of many minor rules. Moreover, the single coverage constraint indeed caused insufficient number of significant rules because of the top 20 features only a small fraction are contained in the decision trees. Consequently, important rules based on other top features are still missed. That's why PCL can still outperformed decision trees. However the PCL classifier cannot beat decision trees for separating the minitype from other main subtypes though we produced much more significant rules than C4.5. This may suggest us to consider other efficient method to summarize the discriminating power of EPs.

We also conducted 10-fold cross validation experiments. The results on the training data of the PCL classifier were also generally better than C4.5 tree models. (See Table 8.) Its accuracy was close to SVM and 3-NN. Overall, 3-NN performed the best.

6.2 Distinction between MLL and other conventional ALL subtypes

Armstrong's team [?] have also published their research results on the subtype classification of the leukemia disease. However, the investigation is much smaller than St. Jude's. Only three subtypes (AML, ALL, MLL) are studied. There are a total 57 training instances (20, 17, and 20 respectively for ALL, MLL, and AML) and 15 test instances (4, 3, and 8 respectively for ALL, MLL, and AML). Table 9 reports our classification performance. Once again, single C4.5 trees made several more mistakes than the other classifiers, while our PCL classifier shown its excellence.

Table 8: 10-fold cross validation results on the 215 training instances in the problem of subtype classification of the ALL disease.

Datasets (training data size in each class)	Error Rates for Training Data							
	PCL			C4.5			SVM	3-NN
	$k = 20, 25, 30$			Single	Bagging	Boosting		
BCR-ABL vs others (9:206)	0:0	0:1	0:1	3:4	4:0	3:4	1:1	0:0
E2A-PBX1 vs others (18:197)	0:0	0:0	0:0	0:0	0:0	0:0	0:0	0:0
HyperL50 vs others (42:173)	9:4	9:4	9:4	9:4	11:4	9:6	11:7	7:4
MLL vs others (14:201)	0:1	0:2	0:2	5:3	3:0	3:0	0:0	0:1
T-ALL vs others (28:187)	0:0	0:0	0:0	0:0	0:0	0:0	0:0	0:0
TEL-AML1 vs others (52:163)	1:3	2:0	1:3	3:4	2:2	3:3	1:3	0:4
minitype vs main (52:163)	5:15	4:15	4:15	17:12	12:7	11:13	10:9	6:9

Table 9: MLL distinction from the conventional ALL by four classification models.

Datasets (test data size in each class)	Error Rates for Test Data							
	PCL			C4.5			SVM	3-NN
	$k = 20, 25, 30$			Single	Bagging	Boosting		
ALL vs others (4:11)	0:0	0:0	0:0	1:2	0:0	1:2	0:0	0:0
AML vs others (8:7)	0:0	0:0	0:0	1:0	0:0	0:0	0:0	0:0
MLL vs others (3:12)	0:1	0:1	0:1	0:0	0:0	0:0	0:0	0:0
ALL vs AML (4:8)	0:0	0:0	0:0	0:0	0:1	0:0	0:0	0:0
ALL vs MLL (4:3)	0:0	0:0	0:0	0:0	0:0	0:0	0:0	0:1
AML vs MLL (8:3)	0:0	0:0	0:0	0:0	0:0	0:0	1:0	1:0

Table 10: Error rates for relapse prediction of breast cancer patient.

Dataset size (relapse vs non-relapse)	Error Rates for Test Data							
	PCL			C4.5			SVM	3-NN
	$k = 20, 25, 30$			Single	Bagging	Boosting		
12 : 7	3:3	3:3	3:3	5:0	5:2	2:4	6:2	7:3
34 : 44	10-fold Error Rates for Training Data							
	3:3	3:3	3:3	8:4	8:2	6:3	3:2	4:4

An interesting point here is about boosting in the case of "ALL vs others". Observe that the single C4.5 tree and the boosted C4.5 method made the same mistakes. We found that the single C4.5 tree correctly classified all the training data. So, the boosting method did not generate any more individual tree as its stop criteria was already met. So the boosting idea cannot help C4.5 in such a situation.

Another interesting point is about bagging in the case of "ALL vs AML". All models, including the single C4.5 tree, made no mistakes but the bagging method made one. The reason is that the committee of trees constructed by bagging are based on boostrapped original training data. In each trial, some original training instances are repeated as training data, while some other original training data are discarded. This idea sometimes misleads the system. We also stress that rules from boostrapped data do not reflect the nature of the application because boostrapped data are not the original data. So, the bagging idea may sometime help and sometimes not help classification. Moreover, the rules should be understood cautiously.

6.3 Likelihood Prediction of Reoccurrence of Breast Cancer

The relapse study of a cancer disease is important. This is because if a patient is predicted as favourable then we can make an effort to minimize the toxity of the therapy for the patient. Veer et al [?] have collected a dataset of 97 patients who suffered from breast cancer. Some of them remained free of disease after their initial diagnosis for an interval of at least 5 years (non-relapse class). However, some of them developed distant metastases within 5 years (relapse class). The study is aiming at finding important prognostic biomarkers in gene expression profiles and use them for prediction. The data is divided into a training set of 78 instances (44 non-relapse and 34 relapse) and a test set of 19 instances (7 non-relapse and 12 relapse). Every instance is described by 24500 features. The error rates on the test data and the 10-fold CV results of the classification models are reported in Table 10.

In general, the accuracy on the test data was not good. PCL made comparable number of mistakes with other models. This problem may be solved by adding more data and by using more sophisticated feature selection methods. We think this problem can not be solved by computer scientists alone. More importantly, domain knowledge are necessary to assess the success possibility of using gene expression to conduct relapse study. In particular, we note that breast cancer is a heterogeneous disease with multiple subtypes. Different subtypes are likely to relapse along different pathways and hence prediction of relapse for breast cancer should be trained on subtype-specific datasets.

6.4 Why Single C4.5 Trees Do Not Work

C4.5 can provide small-sized elegant decision trees to classify test samples. However, as shown in the previous sections, its accuracy is not excellent on test data, even though its performance on the *training* data closely approaches to the perfect level. This may be partially caused by its inferior search algorithm and the fragmentation problem, and partially caused by the complexity of gene expression data.

As shown in Section 2, the learning of C4.5 is a greedy heuristics. The decision tree is constructed using an information gain idea and using a divide-and-conquer technique to stop the search of the most discriminatory features. As gene expression profiles often contain very large number of features, there exist many (tens of) good genes to separate two classes of cells. The greedily choosing the most discriminatory genes from the training data may not be the optimal features for test data. There may well be many other equally discriminatory genes in the training data that got discarded. Thus, the accuracy of C4.5 can be sometimes greatly suffer from this problem. This problem can be partially solved by using a collection of decision trees constructed by the bagging and boosting ideas.

7 Conclusion