Use of Built-in Features in the Interpretation of High-dimensional Cancer Diagnosis Data

Jinyan Li **Huiqing Liu** **Limsoon Wong**

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

Abstract

Cancer diagnosis data, for example microarray gene expression profiling data and proteomic profiling data, are often described by thousands of features. To computationally make a diagnosis for new samples, these data are usually input to a *learning algorithm*, the algorithm then induces a *classifier*, the classifier then predicts a class label for any test sample. As the data is so high-dimensional, most of the resulting classifiers are very complicated particularly those based on kernel-functions such as support vector machines—the interpretation of the decision results must need all the features to be involved. In this paper, we discuss *built-in* features and use them to concisely characterize the data and to easily interpret the decisions. Built-in features are features that are used only in the classifiers, and that are only a small subset of the original features, e.g., the features in a decision tree. So, the notion of built-in features is different from input features and also from original features. As there is a significant reduction from the huge size of original features to a small number of relevant features, the complexity of the interpretation can be much eased. The use of built-in features also provides much potential for elucidating the translation between raw data and clinically useful knowledge. In this paper, we also report that the performance of classifiers using built-in features tends to remain stable even input feature space changes, but other types of classifiers fluctuate their performance. So, once again, we promote the use of classifiers that use built-in features since the algorithms can avoid the existing hard problem of selecting best number of features for learning.

Keywords: Decision trees, built-in features, interpretation, classification.

1 Introduction

Many bio-medical applications, such as diagnosis using gene expression profiles (Alon, Barkai, Notterman, Gish, Ybarra, Mack & Levine 1999), relapse studies (van't Veer, Dai, van de Vijver1, He, Hart, Mao, Peterse, van der Kooy, Marton, Witteveen, Schreiber, Kerkhoven, Roberts, Linsley, Bernards & Friend 2002), and subtype distinction of a heterogeneous disease (Yeoh, Ross, Shurtleff, Williams, Patel, Mahfouz, Behm, Raimondi, Relling, Patel, Cheng, Campana, Wilkins, Zhou, Li, Liu, Pui, Evans, Naeve, Wong & Downing 2002), are typical supervised learning problems. In these applications, a learning algorithm is usually presented with a set of training samples (known samples or history data) where each sample is described by a vector of feature values and a class label. For example, in gene expression cancer diagnosis problems the features are genes, the values are the genes' expression levels, and the class label might indicate whether or not a cell was determined to be normal or cancerous. The learning has two primary goals. One is to induce from the training data a *classifier* that can classify future (unseen) samples with high accuracy. In addition to inducing a highly accurate classifier, the other purpose of the learning is to obtain explicit, easily comprehensible, and reliable decision rules or patterns from the training data for potential clinical use.

Supervised learning has been extensively studied in machine learning and data mining, so there are many choices of learning algorithms to achieve the first goal—to induce accurate classifiers from training data. For example, neural networks (Haykin 1999) and its extension concept, support vector machines (Burges 1998, Cortez & Vapnik 1995), can approximate the training data with almost error-free (Chen & Chen 1995, Hornik 1993), and can exhibit good generalization capability on test samples. However, the structure of these classifiers are very complicated especially when the input feature space is large. This complexity is due to two reasons: (1) All the input features have to be used in these classifiers. If a dataset is described by 1000 features, then the mapping (the classifier) has 1000 variables. (2) The kernel functions are usually set as non-linear functions for the purpose of achieving good accuracy. Even they are set as linear functions, the hyperplanes (say with 1000 dimensions) are still hard to understand.

Therefore, an ideal classifier should be a classifier that is accurate and that uses only a small proportion of the original features in a systematic way. In this paper, this small proportion of the original features are called *built-in features* of this classifier. A decision tree is a good example of classifiers using built-in features. Suppose a gene expression cancer diagnosis dataset have n' number of features, using a tree induction algorithm such as C4.5 (Quinlan 1993), usually we can get a decision tree containing a very small n'' number of features. Therefore, the interpretation of the decision results needs only the n'' features rather than the n' features. From our previous studies, we found that a decision tree derived from gene expression data usually contained less than 10 features. Compared to the thousands of input features,

Contact email: jinyan@i2r.a-star.edu.sg

these n'' built-in features are only a tiny proportion. We use Figure 1 to illustrate the difference between input features and built-in features for a given learning algorithm and a training dataset having n number of original features.

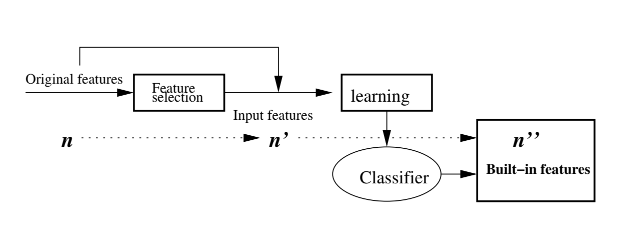

Figure 1: Feature changes before and after learning

The *input features* are directly fed to the learning algorithm. After learning, the algorithm produces a classifier. Features used in the classifier are *built-in* features. From the original features to the input features, there may have a feature-filter step, called feature selection. Common methods for feature selection include signal-to-noise (Golub, Slonim, Tamayo, Huard, Gaasenbeek, Mesirov, Coller, Loh, Downing, Caligiuri, Bloomfield & Lander 1999), t-statistics, and entropy (Fayyad & Irani 1993) etc. Without the pre-feature selection, then the original features are input features.

As seen above, decision tree classifiers can be used to achieve the second goal of learning, that is, to generate easily understandable rules using a small subset of input features. However, single decision trees are not as accurate as support vector machines in general. To achieve the both goals of learning, in this paper, we describe a new committee classifier, known as CS4 (Li & Liu 2003, Li, Liu, Ng & Wong 2003), that consists of multiple cross-supportive decision trees. This new classifier has much better performance than single decision trees. So, we can use this classifier, containing a small number of built-in features, to interpret cancer diagnosis data as accurately as kernel-based support vector machines do, meanwhile, to discover potentially useful clinical rules.

The performance of this new learning algorithm is found to be stable when input feature space changes. Suppose a dataset have n number of features. If we reduce these features to n_1' or n_2' features, using a feature selection method, then the performance of this classifier of taking the original n features as input features, is found to be similar to the performance when using the n_1' or n_2' selected features as input features. Another interesting thing is that for these 3 different senarios of input features, the number of built-in features is also stable. On the other hand, we found that support vector machines fluctuate their performance significantly, sometimes becoming better and sometime becoming worse, after the feature selections. This important observation once again suggest us to use tree classifiers for computer-aided data diagnosis because their counterparts such as SVMs and k-NN (k-nearest neighbour) are lack of a theoretical method to estimate the best number of input features, and because these counterparts are hardly understandable to non-specialists.

The remainder of the paper is organized as follows: Section 2 describes the learning algorithms k-nearest neighbour, support vector machines, and decision trees. We also distinguish two types of classifiers induced by these learning algorithms. Section 3 describes our new learning algorithms CS4 that explores the use of multiple decision trees. This committee method can produces classifiers that are highly accurate and that are also comprehensible. Section 4 studies stabilities of our new classifier CS4, k-nearest neighbour, SVM, and C4.5 (Bagging (Breiman 1996) and Boosting (Freund & Schapire 1996)) with regard to the changes happening in input feature spaces. Section 5 concludes the paper with a summary.

2 Two Categories of Classifiers

In this section, we describe two categories of classifiers. The first category include classifiers that are induced by learning algorithms such as k-nearest neighbour and support vector machines. These classifiers contain the same number of features as in the input feature space. The second category includes decision trees induced by C4.5. These classifiers contain much less numbers of features than the features in the input space.

2.1 k-Nearest Neighbour and Support Vector Machines

Viewing a dataset as a set of *points* in a high-dimensional Euclidean space, the nearest neighbour classifier uses distance as basis to classify new samples. The simple intuition is that the class label of a new sample X should agree with the majority of k nearest points of X. In the case of $k = 1$, the nearest neighbour classifier uses the class label of a training sample that is closest to a test sample as its predicted class label (Cover & Hart 1967). The distance between two points X and Y is usually defined by

$$dist(X,Y) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{\frac{1}{2}}$$

where n is the number of features.

Suppose a training dataset have q samples $\{Y_i = (y_{i1}, y_{i2}, \cdots, y_{in}), i = 1, \cdots, q\}$. To make a prediction for a test sample X, the nearest neighbour classifier first calculates the distance between X and every point Y_i in the training set. Then, the classifier compares the distances and determines a point that is closest to X. Observe that once a prediction is made, the decision result needs all the input features to help explain the distance. If $n = 2$ or 3, the predictions are quite understandable as the distance is in 2-D or 3-D Euclidean space. But if $n = 1000$ (or other large numbers), the squared-root of a sum over n dimensions does not provide any informative knowledge.

Support vector machine is a statistical learning algorithm, it is also called kernel-based learning algorithm (Burges 1998, Cortez & Vapnik 1995). Recently, SVMs have been introduced to bioinformatics and have been widely applied to gene expression data (Ben-Dor, Bruhn, Friedman, Nachman, Schummer & Yakhini 2000, Brown, Grundy, Lin, Cristianini, Sugnet, Furey, Jr & Haussler 2000, Furey, Cristianini, Duffy, Bednarski, Schummer & Haussler 2000) and other problems (Friess, Cristianini & Campbell 1998, Jaakkola, Diekhans & Haussler 2000, Zien, Raatsch, Mika, Schoelkopf, Lengauer & Mueller 2000). Consider a two-class training dataset having n number of original features $x_1, x_2, \cdots, x_n$, a classifier induced from this training data by support vector machines is a function defined by:

$$f(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i \in I} \alpha_i * K(Y_i, X) + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

where $X, Y_i \in \mathbf{R}^n$, $\alpha_i, b \in \mathbf{R}$; α_i, Y_i, I and b are parameters and X is the sample to be classified whether it is

1 (normal) or -1 (abnormal). The SVM training process determines the entire parameter set $\{\alpha_i, Y_i, I, b\}$; the resulting $Y_i, i \in I$ are a subset of the training set and are usually called *support vectors*. The kernel function K can have different forms. For example, $K = (X \cdot Y_i)^p$ implements a polynomial SVM classifier; $K = \tanh(X \cdot Y_i + \delta)$ implements a two-layer neural network.

The simplest form of the kernel functions is $K = X \cdot Y_i = \sum_{j=1}^{n} x_j * y_{ij}$, namely a n-dimensional linear function. Let us discuss the interpretation of decision results by SVM under this simplest setting. Suppose a feature selection method has selected only 10 important features as input features to the SVM learning algorithm, and the algorithm found only three support vectors (say $Y_1, Y_2, \text{and}, Y_3$) after training. If a test sample X is classified to be 1 (e.g. normal class), then the reason of making this decision is because

$$\alpha_1 \sum_{i=1}^{10} x_i y_{1i} + \alpha_2 \sum_{i=1}^{10} x_i y_{2i} + \alpha_3 \sum_{i=1}^{10} x_i y_{3i} + b \geq 0$$

In other words, this is because the test sample X's expression levels of the 10 important genes after some summation and some augumentation are larger or equal to 0. Such an explanation is not concise! We should use simpler explanations to interpret this prediction, so that the computer-aided diagnosis could be acceptable to both doctors and patients. In fact, there exist many much simpler decision rules in the data of gene expressions (Soinov, Krestyaninova & Brazma 2003, Li, Liu, Downing, Yeoh & Wong 2003). Next, we discuss how to use decision trees and rules to understand high-dimensional cancer diagnosis data.

2.2 Decision Trees—Classifiers Using Built-in Features

The induction of a decision tree by C4.5 (Quinlan 1993) from a training dataset is a recursive learning process. The process first selects a feature which is most discriminatory with regard to the entire training data. Then the process goes to use this feature to split the data into non-overlapping sub-groups so that each sub-group contains as many samples of the same class as possible. Then the process is applied to each of these sub-groups of the training data, and iteratively applied to the resulting sub-groups until all sub-groups contain pure or almost pure class of samples. As C4.5 is a greedy heuristic approach, the iterations of the recursive process can be terminated quickly. So, even a large number of features of a training dataset are input to a tree induction algorithm, the resulting classifier—the decision tree—may contain just a couple of features. So, decision trees can be used to *concisely* characterize, using classification rules, high-dimensional bio-medical data.

Each sub-group of the training data corresponds to a rule with which all samples in this sub-group satisfy. Of these rules, some are significant satisfying a large percentage of a class of samples; but some others are trivial satisfying only a small percentage of a class of samples. So, a decision tree is a set of either significant or trivial classification rules. The rules can be represented in the form of

If $cond_1$ and $cond_2 \cdots$ and $cond_m$,
then *a predictive term.*

Note that the predictive term in a rule often refers to a single class. All conditions in a rule are required to be not all true in any samples of any classes other than the one in the predictive term. In cancer and other disease diagnosis, the number m of conditions is preferred to be no more than 5 for easy understanding. Ideally, rules with m = 1, 2, or 3 are best for clinical diagnosis.

To demonstrate the typical structure of decision trees and to show where are built-in features in a tree, we present an example of decision trees that is induced by C4.5 (Quinlan 1993) from a gene expression profiling dataset. The dataset consists of 215 two-class training samples (14 MLL plus 201 OTHERS) for differentiating the MLL subtype from the other subtypes of childhood leukemia disease (Yeoh et al. 2002). The data are described by 12558 features. Here each feature is a gene, having continuous expression values. The structure of this tree is depicted in Figure 2. Observe that there are only 4 built-in features in this tree residing at the 4 non-leaf nodes, namely 34306_at, 40033_at, 33244_at, and 35604_at. Compared to the total 12558 input features, the 4 built-in features is a very tiny fraction. Each of the five leaf nodes corresponds to a rule, the rule's predictive term is the class label contained in the leaf node. As an example, the rule at the most-left side of the tree is:

If 34306_*at* $\leq$ 13683.6 and 40033_*at* $\leq$ 3691.4,
then *this sample is for* OTHERS.

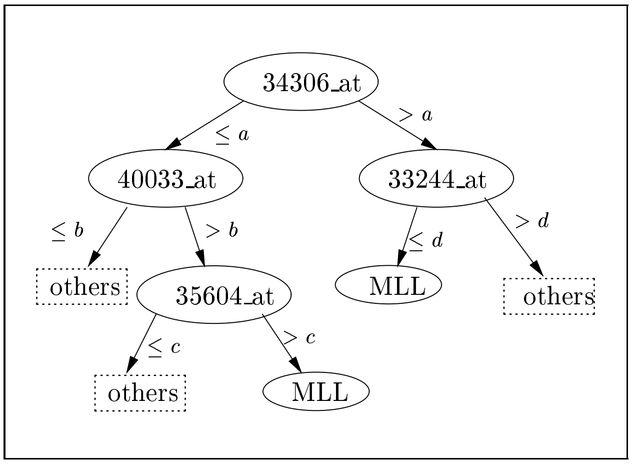

Figure 2: A decision tree induced by C4.5 from gene expression dataset for differentiating the subtype MLL against other subtypes of childhood leukemia. Here $a = 13683.6, b = 3691.4, c = 986.9, d = 846.6$.

Consequently, the interpretation of the decision results by this tree classifier needs only some of the 4 built-in features rather than the entire 12588 features. To see this clear, we decompose the tree into 5 rules and use the rules to establish a function as follows.

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

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

This example tells us that though gene expression data are high-dimensional, there indeed exist many simple rules containing only 2 or 3 features that can be used to *concisely* characterize the data. These simple rules also have much potential to be translated into clinical knowledge. In contrast, using large number of features to understand bio-medical data, as integrated in non-linear kernal based classifiers, should be avoided unless it is necessary.

3 A New Committee Classifier Using Multiple Decision Trees

From our previous studies on gene expression profiling data and proteomic profiling data (Liu, Li & Wong 2002), we have observed that single decision trees did not provide good accuracies for independent test data. For example, on the dataset discussed in the previous section, the decision tree made 4 mistakes on the 112 test samples, though it has a perfect 100% accuracy on the 215 training samples. A possible reason is that the greedy search heuristic confines the capability of the learning algorithm, only allowing the algorithm learn well on one aspect of the high-dimensional data. We found that there exist many decision trees that can characterize the data from different aspects, and that have similar accuracy. So, it is possible for us to generate multiple trees and combine these trees for more reliable predictions as done by a committee. Next, we introduce a new committee classifier, called CS4 (Li, Liu, Ng & Wong 2003, Li & Liu 2003), that explores the use of multiple cascading trees.

To discover a committee of trees, our heuristic is to use each of top-ranked features as root node to construct a decision tree. For example, to discover 10 decision trees, we use 10 top-ranked features each as the root node of a tree. We call these trees *cascading* trees (Li & Liu 2003). In this paper, the features are individually ranked by the gain-ratio method (Quinlan 1993). Our this tree construction method is reasonable as there are many outstanding features existing in high-dimensional gene expression or proteomic profiling datasets, that possess similar classification merits with little difference.

To illustrate this point, we present an example. Table 1 summarizes the training and test performance, and the numbers of built-in features used in the 10 cascading trees which are discovered from the MLL-OTHERS dataset. Note that we used the ith top-ranked feature as root node to establish the ith ($1 \leq i \leq 10$) tree. Observe that: (a) the 10 trees made similar numbers of errors on the training and test data, performing nearly equally-well; (b) the 5th, 8th, or the 9th tree made a smaller number of errors than the first tree made; (c) the rules in these trees were very simple, containing about 2, 3 or 4 features.

How to eliminate the errors made by these individual trees so that we can achieve a perfect accuracy? We use the following method (Li, Liu, Ng & Wong 2003) to combine the cascading trees for more reliable predictions. Suppose we have discovered k trees from a dataset consisting of only positive and negative training samples. Given a test sample T, each of the k trees in the committee will have a specific rule to tell us a predicted class label for this test sample. Denote the k rules from the tree committee as:

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

Here $k_1 + k_2 = k$. Each of $rule_i^{pos}$ ($1 \leq i \leq k_1$) predicts T to be in the positive class, while each of $rule_i^{neg}$ ($1 \leq i \leq k_2$) predicts T to be in the negative class. Sometimes, the k predictions can be unanimous—i.e., either $k_1 = 0$ or $k_2 = 0$. In these situations, the predictions from all the k rules agree with one another, and the final decision is obvious and seemed reliable. Oftentimes, the k decisions are mixed with either a majority of positive classes or a majority of negative classes. In these situations, we use the following formulas to calculate two classification scores based on the coverages of these rules, i.e., the percentage of a class samples that are satisfied by these rules:

$$Score^{pos}(T) = \sum_{i=1}^{k_1} coverage(rule_i^{pos}),$$

$$Score^{neg}(T) = \sum_{i=1}^{k_2} coverage(rule_i^{neg}).$$

If $Score^{pos}(T)$ is larger than $Score^{neg}(T)$, we assign the positive class to the test sample T. Otherwise, T is predicted as negative. This weighting method allows the tree committee to automatically distinguish the contributions from the trivial rules and those from the significant rules in the prediction process.

Next, we examine the performance of combined trees. When combining the first 2 trees, the committee made 3 mistakes, one less mistake than that made by the sole first tree. When combining the first 4 trees, the committee did not made any errors on the 112 test samples, eliminating all the mistakes made by the first tree. Adding more trees into the committee till the 10th tree, the expanding committee still maintained the perfect test accuracy. See Figure 3 for a graphic trend of the errors made by the 10 tree committees.

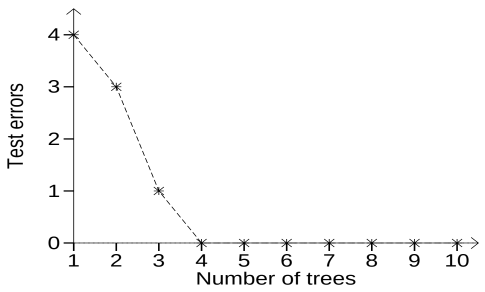

Figure 3: A decreasing trend of the errors made by the 10 tree committees.

So, the use of multiple trees can much improve the performance of single decision trees. In addition to this perfect accuracy, the interpretation of decision results made by this classifier is also easily understandable. In fact, this CS4 classifier is a set of 10 cascading decision trees, each tree is a set of rules, and each rule contains about 3 or 4 features. So, in a broad sense, CS4 is a set of rules that are organized by a cluster of decision trees. SVM also had a perfect error-free performance on this dataset, but the interpretation of this SVM classifier is much harder than the CS4 classifier because SVM needs all the 12558 input features for the interpretation. Obviously, it is better to choose CS4 for analysing bio-medical data as it can achieve the both goals of learning.

4 Performance Stability of Classifiers When Input Feature Space Changes

One way to decrease the structure complexity of classifiers induced by support vector machines is to reduce the number of input features. The concern is whether the change of input feature space will cause the performance of the resulting SVM classifiers to fluctuate.

Table 1: The training and test errors of our 10 decision trees on the MLL-others dataset that consist of 215 training and 112 test samples. The numbers of built-in features in the 10 trees are also listed.

Tree No.	1	2	3	4	5	6	7	8	9	10
Training errors	0	0	0	0	0	0	1	0	0	0
Test errors	4	3	2	3	2	6	7	2	1	6
# of features	4	4	4	4	4	4	4	4	4	6

If little impact is caused on the performance, then the learning algorithm would prefer to using smaller number of input features for easy understanding of the data. However, we found that the performance of SVM classifiers is often subject to feature selections. That is, the performance of taking all original features as input features significantly differs from those when other numbers of features are used. Also, we did not find a regular trend when input feature space changes. So, this way to smooth the complexity of SVM classifiers may deteriorate their performance. For comparison, we also examine how decision-tree-based classifiers are resistant to the changes of input feature spaces.

We used five different input feature scenarios to study the resistance problem (or stability problem). The five situations are: (a) using all the original features as input features, (b) using all *entropy-discretized features* (Li, Liu & Wong 2003) as input features, (c) using all mean-entropy discretized features (Li, Liu & Wong 2003) as input features, (d) using 30, an ad-hoc number, top-ranked features, and (e) using 20 top-ranked features. We define *entropy-discretized features* as features that are discretized into at least two intervals by an entropy-discretization method (Fayyad & Irani 1993). Similarly, mean-entropy discretized features are feature whose entropy value is *smaller* than the mean entropy value of all the entropy-discretized features. Usually, entropy-discretized features consist of only 5% - 10% of the original features for different datasets; and mean-entropy discretized features consist of only 2% - 3% of the original features for different datasets (Li, Liu & Wong 2003). So, we consider a decreasing trend of selected features for the learning algorithms. The performance of SVM did not exhibit a monotonic behavior, and not a stable trend either, as shown later.

The datasets used in this study all come from our Kent Ridge Biomedical Data Sets Repository at `http://sdmc.i2r.a-star.edu.sg/GEDatasets/Datasets.html`. Most of the datasets are described by more than 10,000 features. Table 2 gives the basic information of the datasets and numbers of original features, entropy-discretized features, and mean entropy-discretized features.

Table 3 reports the classification errors on the 15 datasets of SVM and k nearest neighbour (k-NN) when the 5 scenarios of input features are used in the 10-fold cross-validation (except the last 4 datasets in Table 3). Here an error number is the number of samples that are wrongly classified by a classifier.

From the last row (Total Errors) of Table 3, we have the following observations on SVM:

- For the most complicated case of taking all the original features as input features, the SVM classifier made the most number 99 of errors over the 15 datasets.
- For the simplest case of taking the ad-hoc top 20 features as input features, the SVM classifier made the second most 75 number of errors.
- The SVM classifier made the least number 47 of errors when it took the mean-entropy discretized features in the learning.

It can be seen that feature selection methods should be recommended in the learning of SVMs for achieving good accuracy. But, if removing too many features to reduce the complexity of the classifier, its performance may much worsen as shown in the case of only top 20 features used. Overall, the SVM's error trend exhibits a U-shape fluctuating curve over the five senarios of input features. Interestingly, the k-NN classifier also shows a similar U-shape fluctuating trend of errors when input feature spaces change. So, it is difficult for SVM and k-NN to achieve the both goals of learning—We have discussed that the interpretation of SVM classifiers is still complicated even only 10 input features are used. Unfortunately, the performance of SVM using 10 top features decreases again, compared to its performance when taking 20 top features.

Next, we report in Table 4 the error change trend of three committee classifiers that use built-in features: our CS4 classifier, Bagging (Breiman 1996) and Boosting (Freund & Schapire 1996). We used C4.5 (Quinlan 1993) as base classifier for Bagging and Boosting; their source code are available from *Weka* version 3.2 (`http://www.cs.waikato.ac.nz/~ml/weka/`) under the GNU General Public Licence. For the three committee classifiers, we set the number of base classifiers as the same number 20.

We can see that the three committee classifiers did not change much in their performance after the input features are selected by the entropy method. Nonetheless, there was still a slight decrease in total errors after the feature selections (See the last row of Table 4). The normalized variance of the performance of CS4 is as small as 0.0096. Such a trend is different from the changes occurred in SVM and k-NN when using the entropy-discretized and the mean-entropy discretized features—their total errors changed significantly. (See the last row of Table 3.) The normalized performance variance are 0.1987 and 0.1236 for k-NN and SVM respectively. So, we can largely get that classifiers using built-in features are relatively more resistant to feature selection, while SVM and k-NN are more sensitive.

CS4 made 91 and 93 errors respectively when taking the top 30 and top 20 features. It can be seen that taking these ad-hoc numbers of features did worse, though not much, performance than using all the features as input. So, we do not recommend to use these ad-hoc, small number of input features for CS4.

Next, we present a reason to explain the above stability. As discussed earlier, decision tree based classifiers usually take a small proportion of input features as their built-in features. Table 5 gives two examples, comparing the number of built-in features used in CS4 and the total number of input features. In the first example, the original ALL-AML testing data (Golub et al. 1999) have 7129 input features, but CS4 uses only 31 features of them to construct the tree committee; When using the 866 entropy-discretized features as input features, there are only 26 built-in features in CS4; When using the 350 mean-entropy discretized features as input features, there are only 25 built-in

Table 2: Dataset description. The total number of samples for each of the first 11 datasets are shown in the 3rd column. We conduct 10-fold cross-validation on these 11 datasets to get error number of a classifier. For the remaining 4 datasets, only test samples are shown. Here, “all” represents all of the original features; “entropy” represents the entropy discretized features; and “M-entropy” represents the mean-entropy discretized features.

Datasets	# classes	# samples	# of features		
			all	entropy	M-entropy
T-ALL	2	327	12558	1766	539
E2A-PBX1	2	327	12558	1103	347
TEL-AML1	2	327	12558	1180	526
BCR-ABL	2	327	12558	160	56
MLL	2	327	12558	571	201
Hyperdip>50	2	327	12558	1481	504
Ovarian cancer	2	253	15154	6137	2324
Prostate cancer	2	102	12600	1555	502
Colon tumor	2	62	2000	135	47
ALL-AML	2	72	7129	1012	344
Subtype lymphoma	2	47	4026	398	144
Stjude testing	2	112	12558	1220	412
Lung cancer testing	2	149	12533	2173	777
ALL-AML testing	2	34	7129	866	350
Armstrong testing	3	15	12582	4411	1628

Table 3: The error numbers of SVM and k-NN when using five different senarios of input features.

Datasets	SVM					k-NN				
	all	entropy	M-entropy	top-30	top-20	all	entropy	M-entropy	top-30	top-20
T-ALL	1	0	0	0	1	8	3	0	0	1
E2A-PBX1	1	1	1	0	0	1	1	1	0	0
TEL-AML1	4	3	4	7	7	14	4	4	7	8
BCR-ABL	12	8	8	8	6	15	9	10	10	9
MLL	7	5	2	2	7	9	5	4	6	8
Hyperdip>50	11	9	11	14	15	21	13	16	15	15
Ovarian	0	0	0	4	4	15	11	10	6	4
Prostate	7	8	6	6	6	18	10	8	9	9
Colon Tumor	11	9	8	8	8	19	9	11	13	12
ALL-AML	1	2	2	4	2	10	2	1	4	4
Subtype lymphoma	6	3	2	4	6	13	5	5	4	5
Stjude testing	32	1	2	6	8	20	5	2	4	8
Lung cancer	1	1	0	1	1	3	1	1	1	1
ALL-AML testing	5	1	1	5	4	10	6	2	3	3
Armstrong	0	0	0	0	0	2	2	2	1	1
Total Errors	99	51	47	69	75	178	85	77	83	88

Table 4: The error numbers of the three tree-based committee classifiers.

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

Table 5: Built-in features used in CS4 are far below the number of input features. Also the number of built-in features is stable when input space changes.

Datasets	Number of input features vs built-in features by CS4		
	original data	entropy-reduced data	M-entropy reduced data
ALL-AML testing	7129 vs 31	866 vs 26	350 vs 25
Lung-cancer	12533 vs 20	2173 vs 20	777 vs 20

features. Though the input features sharply changed from 7129 to 350, the number of built-in features was maintained in the CS4 classifier. In fact, we can prove that for any number of input features between 7129 and 350, there is no big difference on the number of built-in features in CS4. This number stability of built-in features is possibly a primary reason why CS4 is much resistant feature selection. This number stability of built-in features can be also observed in the lung-cancer dataset of Table 5. This property lets pre-feature selections become un-essential to CS4, unlike SVM that depends on feature selection heavily.

Next, we compare the accuracy of CS4 with the other classifiers. Compared to the classical Bagging and Boosting, CS4 outperformed them with only one exception case on the Hyperdip>50 data set where Boosting made 14 errors, but CS4 made 15 errors. See Table 4. Compared to k-NN and SVM, CS4 is the best if using all the original features as input features. If using selected features, CS4 is slightly better than k-NN, but slightly worse than SVM. In general, CS4 has a comparable accuracy with the non-linear classifiers.

Together with easy comprehensibility, CS4 would be a good learning and classification method for analysing high-dimensional cancer diagnosis data.

5 Conclusion

In this paper, we have discussed the use of built-in features for concisely characterizing and also for comprehensively understanding high dimensional cancer diagnosis data. Decision tree based classifiers are classifiers that use built-in features. Usually, features contained in a tree is much less than the number of input features. As seen before, this systematic reduction can be from around ten thousands of features to only a couple of features. A tree can be directly decomposed into a set of rules, each containing several features. Therefore, decision tree based classifiers have much potential to translate complex raw diagnosis data into clinical knowledge. In contrast, k-nearest neighbour and support vector machines must use the same number of features as input space to explain their decisions. It is not easy for them to concisely characterize high-dimensional biological data.

Also, we have described a new committee classifier, called CS4, that explores the use of multiple cross-supportive cascading trees. Extensive experimental results show that CS4 is an accurate classifier. Its performance is better than the classical Bagging and Boosting, and it is also comparable to the performance of k-nearest neighbour and SVMs. CS4 has been found to be not that sensitive to pre-feature selections. So, the difficult pre-feature selections become un-essential in CS4's learning. A reason causing this property is likely the stable number of built-in features used in CS4 when input space changes.

We have two directions for future work. One is to study new ways of constructing cascading trees, the other is to study the relation between CS4 and Bagging. All of these would produce more convenient methods to deal with high-dimensional cancer diagnosis data.

References

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D. & Levine, A. J. (1999), 'Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays', *Proceedings of National Academy of Sciences of the United States of American* **96**, 6745–6750.

Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M. & Yakhini, Z. (2000), 'Tissue classification with gene expression profiles', *Journal of Computational Biology* **7**, 559–584.

Breiman, L. (1996), 'Bagging predictors', *Machine Learning* **24**, 123–140.

Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Jr, M. A. & Haussler, D. (2000), 'Knowledge-based analysis of microarray gene expression data by using support vector machines', *Proc. Natl. Acad. Sci. USA* **97**, 262–267.

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

Chen, T. & Chen, H. (1995), 'Universal approximation to non-linear operators by neural networks with arbitrary activation functions and its application to dynamically systems', *IEEE Transactions on Neural Networks* **6**, 911–917.

Cortez, C. & Vapnik, V. (1995), 'Support vector networks', *Machine Learning* **20**, 273–279.

Cover, T. M. & Hart, P. E. (1967), 'Nearest neighbor pattern classification', *IEEE Transactions on Information Theory* **13**, 21–27.

Fayyad, U. & Irani, K. (1993), Multi-interval discretization of continuous-valued attributes for classification learning, *in* R. Bajcsy, ed., 'Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence', Morgan Kaufmann, pp. 1022–1029.

Freund, Y. & Schapire, R. E. (1996), Experiments with a new boosting algorithm, *in* L. Saitta, ed., 'Machine Learning: Proceedings of the Thirteenth International Conference', Morgan Kaufmann, Bari, Italy, pp. 148–156.

Friess, T.-T., Cristianini, N. & Campbell, C. (1998), The kernel adatron algorithm: A fast and simple learning procedure for support vector machines, *in* 'Proceedings of 15th International Conference on Machine Learning'.

Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M. & Haussler, D. (2000), 'Support vector machine classification and validation of cancer tissue samples using microarray expression data', *Bioinformatics* **16**, 906–914.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J., Caligiuri, M. A., Bloomfield, C. D. & Lander, E. S. (1999), 'Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring', *Science* **286**, 531–537.

Haykin, S. S. (1999), *Neural Networks: A Comprehensive Foundation*, Upper Saddle River, N.J.: Prentice Hall.

Hornik, K. (1993), 'Some new results on neural network approximation', *Neural Networks* **6**, 1069–1072.

Jaakkola, T., Diekhans, M. & Haussler, D. (2000), 'A discriminative framework for detecting remote protein homologies', *Journal of Computational Biology* **7**, 95–114.

Li, J. & Liu, H. (2003), Ensembles of cascading trees, *in* 'Proceedings of 2003 IEEE International Conference on Data Mining (ICDM 2003)', IEEE Computer Society, Melbourne, Florida, p. To appear.

Li, J., Liu, H., Downing, J. R., Yeoh, A. E.-J. & Wong, L. (2003), 'Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients', *Bioinformatics* **19**, 71 –78.

Li, J., Liu, H., Ng, S.-K. & Wong, L. (2003), 'Discovery of significant rules for classifying cancer diagnosis data', *Bioinformatics* **19**, ii93–102.

Li, J., Liu, H. & Wong, L. (2003), Mean-entropy discretized features are effective for classifying high-dimensional biomedical data, *in* 'Proceedings of the 3rd ACM SIGKDD Workshop on Data Mining in Bioinformatics', ACM Press, Washington D.C., pp. 17–24.

Liu, H., Li, J. & Wong, L. (2002), A comparative study on feature selection and classification methods using gene and protein expression profiles, *in* 'Genome Informatics 2002: Proceedings of 13th International Conference on Genome Informatics', Universal Academy Press, Tokyo, Japan, pp. 51 – 60.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.

Soinov, L. A., Krestyaninova, M. A. & Brazma, A. (2003), 'Towards reconstruction of gene networks from expression data by supervised learning', *Genome Biology* **4**, R6.1–R6.10.

van't Veer, L. J., Dai, H., van de Vijver1, M. J., He, Y. D., Hart, A. A., Mao, M., Peterse, H. L., van der Kooy, K., Marton, M. J., Witteveen, A. T., Schreiber, G. J., Kerkhoven, R. M., Roberts, C., Linsley, P. S., Bernards, R. & Friend, S. H. (2002), 'Gene expression profiling predicts clinical outcome of breast cancer', *Nature* **415**, 530–536.

Yeoh, E.-J., Ross, M. E., Shurtleff, S. A., Williams, W. K., Patel, D., Mahfouz, R., Behm, F. G., Raimondi, S. C., Relling, M. V., Patel, A., Cheng, C., Campana, D., Wilkins, D., Zhou, X., Li, J., Liu, H., Pui, C.-H., Evans, W. E., Naeve, C., Wong, L. & Downing, J. R. (2002), 'Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling', *Cancer Cell* **1**, 133–143.

Zien, A., Raatsch, G., Mika, S., Schoelkopf, B., Lengauer, T. & Mueller, K. (2000), 'Engineering support vector machine kernels that recognize translation initiation sites', *Bioinformatics* **16**, 799–807.