Cancer Classification With MicroRNA Expression Patterns Found By An Information Theory Approach

Yun Zheng and Chee Keong Kwoh
Bioinformatics Research Center, School of Computer Engineering
Nanyang Technological Univerity, Singapore 639798
Email: {pg04325488, asckkwoh}@ntu.edu.sg

***Abstract*— Some non-coding small RNAs, known as microRNAs (miRNAs), have been shown to play important roles in gene regulation and various biological processes. The abnormal expression of some specific miRNA genes often results in the development of cancer. In this paper, we find discriminatory miRNA patterns for cancer classification from miRNA expression profiles with an information theory approach. Our approach evaluates subset of miRNAs by checking the mutual information between these miRNAs and the class attribute $I(\mathbf{X};Y)$ with respect to the entropy of the class attribute $H(Y)$. Then, optimal subset of miRNAs that satisfies $I(X;Y) = H(Y)$ or $H(Y) - I(X;Y) \leq \epsilon \times H(Y)$ for noisy data sets are chosen to build the classification models. The experimental results show that the expression patterns from a small set of miRNAs are very accurate in prediction. Further, the experimental results also suggest that the expression patterns of these informative miRNAs are conserved in different vertebrates during the evolution process.**

***Index Terms*— microRNAs, gene expression patterns, cancer classification, feature selection**

I. INTRODUCTION

More and more evidences show that miRNAs play important roles in gene regulation and various biological processes [1]–[3]. Some recent work has reported that the abnormal expression of some specific miRNA genes often results in the development of cancer [3]–[5].

Lu *et al.* [6] described a new bead-based flow cytometric technique to obtain miRNA expression profiles, which is used to capture the concentration levels of miRNAs in different tissues. The miRNAs show globally lower expression in cancer tissues than in normal tissues [6]. However, it is still not clear which miRNAs contribute more information to the normal/cancer distinction, since complex prediction models, like k-Nearest-Neighbors (kNN) [7] and Probabilistic Neural Networks (PNN) [8], are used in [6]. These models are black-boxes and very hard to understand.

In this paper, we aim at finding informative expression patterns of a small subset of miRNAs for cancer classification. We apply the Discrete Function Learning (DFL) algorithm [9] to the miRNA expression profiles in [6] to find the subset of miRNAs that shows strong distinction of expression levels in normal and tumor tissues.

The DFL algorithm is based on a theorem of information theory, which says that if the mutual information between a vector and the class attribute $I(\mathbf{X};Y)$ equals to the entropy of the class attribute $H(Y)$, then the class attribute is a function of the vector. The DFL algorithm efficiently finds the most discriminatory miRNA vector by checking whether its mutual information with the class attribute satisfy the theorem. We name the subset of the attributes (miRNAs) in the most discriminatory miRNA vector as the *essential attributes*, or the EAs for short. The target subset of miRNAs are supposed to contain all or most information (in terms of entropy) of the class attribute since the real data sets are often noisy. After the learning process, the DFL algorithm provides the classifiers as function tables which contain the EAs and the class attribute. To make use of the obtained function tables reasonably, the predictions are performed in the space defined by the EAs, called the *EA space*, with the 1-Nearest-Neighbor (1NN) algorithm [7]. Specifically, in predicting a new sample, the Hamming distances [10] (for binary and non-binary cases) of the EAs between the new sample and each rule of the classifier are calculated. Then, the classifier selects the class value of the rule which has the minimum Hamming distance to the new sample as the predicted class value.

In this study, we find that a small subset of miRNAs, common for different cancer types, is highly informative and discriminatory. The classifier, built from 75 human normal/tumor tissue samples, only contains these features and successfully obtains 100% accuracy when predicting 12 independent samples of mouse. This result suggests that the expression patterns of these informative miRNAs are conserved in different vertebrates.

We also compare the performances of the DFL algorithm to those from six other classification algorithms. The DFL algorithm obtains better or comparable prediction accuracies to those from the compared algorithms and to the result reported in the literature. However, it is worth mentioning that most of the compared methods, such as the Naive Bayes [11] and k-Nearest-Neighbors algorithm [7], use complex models that suffer the risk of

This paper is based on "Informative MicroRNA Expression Patterns For Cancer Classification," by Y. Zheng and C. K. Kwoh, which appeared in the Proceedings of the Data Mining for Biomedical Applications, PAKDD 2006 Workshop, BioDM 2006, vol. 3916 of Lecture Notes in Computer Science, Singapore, April 2006. © 2006 Springer.

overfitting the training data set.

The rest of the paper are organized as follows. In Section II, we review the DFL algorithm. In Section III, we introduce the data sets and display the results. In Section IV, we summarize this paper.

II. The Discrete Function Learning Algorithm

In this section, we review our method. We will first introduce some notation. We use capital letters to represent discrete random variables, such as X and Y; lower case letters to represent an instance of the random variables, such as x and y; bold capital letters, like $\mathbf{X}$, to represent a vector; and lower case bold letters, like $\mathbf{x}$, to represent an instance of $\mathbf{X}$. The cardinality of $\mathbf{X}$ is represented with $|\mathbf{X}|$.

A. Theoretic Background

The entropy of a discrete random variable X is defined in terms of probability of observing a particular value x of X as [12]:

$$H(X) = -\sum_x P(X = x) \log P(X = x).$$

The entropy is used to describe the diversity of a variable or vector. The more diverse a variable or vector is, the larger entropy they will have. Generally, vectors are more diverse than individual variables, hence have larger entropy. Hereafter, for the purpose of simplicity, we represent $P(X = x)$ with $p(x)$, $P(Y = y)$ with $p(y)$, and so on. The mutual information between a vector $\mathbf{X}$ and Y is defined as [12]:

$$\begin{aligned} I(\mathbf{X}; Y) &= H(Y) - H(Y|\mathbf{X}) = H(\mathbf{X}) - H(\mathbf{X}|Y) \\ &= H(\mathbf{X}) + H(Y) - H(\mathbf{X}, Y) \end{aligned} \quad (1)$$

Mutual information is always non-negative and can be used to measure the relation between two variable, a variable and a vector (Equation 1), or two vectors. Basically, the stronger the relation between two variables, the larger mutual information they will have. Zero mutual information means the two variables are independent or have no relation, which is formally given in Theorem 1. Proof of Theorem 1 can be found in [13].

Theorem 1: For any discrete random variables Y and Z, $I(Y; Z) \geq 0$. Moreover, $I(Y; Z) = 0$ if and only if Y and Z are independent.

The conditional mutual information $I(X; Y|Z)$ (the mutual information between X and Y given Z) [13] is defined by

$$I(X; Y|Z) = \sum_{x,y,z} p(x, y, z) \frac{p(x, y|z)}{p(x|z)p(y|z)}.$$

The chain rule for mutual information is give by Theorem 2, for which the proof is available in [13].

Theorem 2:

$$I(X_1, X_2, \ldots, X_n; Y) = \sum_{i=1}^{n} I(X_i; Y|X_{i-1}, X_{i-2}, \ldots, X_1).$$

B. Theoretic Motivation and Foundation

We restate a theorem about the relationship between the mutual information $I(\mathbf{X}; Y)$ and the number of attributes in $\mathbf{X}$.

Theorem 3: $I(\{\mathbf{X}, Z\}; Y) \geq I(\mathbf{X}; Y)$, with equality if and only if $p(y|\mathbf{x}) = p(y|\mathbf{x}, z)$ for all $(\mathbf{x}, y, z)$ with $p(\mathbf{x}, y, z) > 0$.

Proof of Theorem 3 can be found in [14]. In Theorem 3, it can be seen that $\{\mathbf{X}, Z\}$ will contain more or equal information about Y as $\mathbf{X}$ does. Intuitively, it can be illustrated in Figure 1 (a), where A is a variable and $\mathbf{U}_{s-1}$ is a vector (to be discussed in Section II-C). $H(A)$ and $H(\mathbf{U}_{s-1})$ will definitely share no less information with $H(Y)$ than $H(\mathbf{U}_{s-1})$ alone, since $H(A)$ and $H(\mathbf{U}_{s-1})$ can provide at least the part of information about Y already provided by $H(\mathbf{U}_{s-1})$ alone. To put it another way, the more variables, the more information is provided about another variable.

To measure which subset of features is optimal, we restate the following theorem, which is the theoretical foundation of our algorithm.

Theorem 4: If the mutual information between $\mathbf{X}$ and Y is equal to the entropy of Y, i.e., $I(\mathbf{X}; Y) = H(Y)$, then Y is a function of $\mathbf{X}$.

It has been proved that if $H(Y|X) = 0$, then Y is a function of X [13]. Since $I(X; Y) = H(X) - H(Y|X)$, it is immediate to obtain Theorem 4. The entropy $H(Y)$ represents the diversity of the variable Y. The mutual information $I(\mathbf{X}; Y)$ represents the relation between vector $\mathbf{X}$ and Y. From this point of view, Theorem 4 actually says that the relation between vector $\mathbf{X}$ and Y are very strong, such that there is no more diversity for Y if $\mathbf{X}$ has been known. In other words, the value of $\mathbf{X}$ can fully determine the value of Y.

C. Performing Feature Selection

In this section, we will show the advantage of using the DFL algorithm to perform feature selection. A comparison between the DFL algorithm and existing feature selection methods is given in our work [9].

The feature selection is often used as a preprocessing step before classification. The aim of feature selection is to remove the irrelevant and redundant features, so that the induction algorithms can produce better prediction accuracies with more concise models and less run times.

From Theorem 4, if a feature subset $\mathbf{U} \subseteq \mathbf{V}$ satisfies $I(\mathbf{U}; Y) = H(Y)$, then Y is a deterministic function of $\mathbf{U}$, which means that $\mathbf{U}$ is a complete and optimal feature subset. $\mathbf{U}$ is the *essential attributes* (EAs), because $\mathbf{U}$ essentially determines the value of Y [9]. But the real data sets are often noisy. Thus, the DFL algorithm estimates the optimal feature subsets with the ϵ value method to be introduced in Section II-F by finding feature subsets to satisfy $H(Y) - I(\mathbf{U}; Y) \leq \epsilon \times H(Y)$.

From Theorem 1, the irrelevant features tend to share zero or very small mutual information with the class attribute in the presence of noise. Therefore, the irrelevant features can be eliminated by choosing those features with

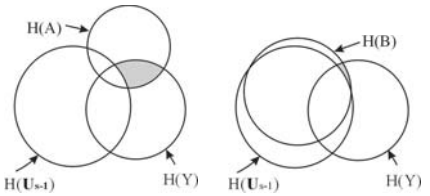

Figure 1. The advantage of using mutual information to choose the most discriminatory feature vectors. The circles represent the entropy of variables or vectors. The intersection between the circles represents the mutual information between the variables or vectors. $\mathbf{U}_{s-1}$ is the features already chosen. The shaded regions represent $I(X_i;Y|\mathbf{U}_{s-1})$, where $X_i \in \mathbf{V} \setminus \mathbf{U}_{s-1}$. (a) When $X_i = A$. A shares less mutual information with Y than B does. However, the vector $\{\mathbf{U}_{s-1}, A\}$ shares larger mutual information with Y than the vector $\{\mathbf{U}_{s-1}, B\}$ does. (b) When $X_i = B$. B shares larger mutual information with Y than A does. But B and $\mathbf{U}_{s-1}$ have a large mutual information, which means that $\mathbf{U}_{s-1}$ has contained most of the information of Y carried by B or the additional information of Y carried by B, $I(B;Y|\mathbf{U}_{s-1})$, is small.

relatively large mutual information with the class attribute in modelling process.

When choosing candidate features, our approach maximizes the mutual information between the feature subsets and the class attribute. Suppose that $\mathbf{U}_{s-1}$ has already been selected at the step $s-1$, and the DFL algorithm is trying to add a new feature $X_i \in \mathbf{V} \setminus \mathbf{U}_{s-1}$ to $\mathbf{U}_{s-1}$. Specifically, our method uses Equation 2 as a criterion to add new features to $\mathbf{U}$.

$$\begin{cases} X_{(1)} = & argmax_i I(X_i;Y), i = 1,\ldots,n \\ X_{(s)} = & argmax_i I(\mathbf{U}_{s-1}, X_i;Y), \end{cases} \quad (2)$$

where $\forall s$, $1 < s \leq k$, $\mathbf{U}_1 = \{X_{(1)}\}$, and $\mathbf{U}_s = \mathbf{U}_{s-1} \cup \{X_{(s)}\}$. From Equation 2, it is obvious that the irrelevant features have lost the opportunity to be chosen as EAs of the classifiers after the first EA, $X_{(1)}$, is chosen, since $I(X_i;Y)$ is very small if X_i is an irrelevant feature.

Next, we illustrate how to eliminate the redundant features. From Theorem 2, we have

$$I(\mathbf{U}_{s-1}, X_i;Y) = I(\mathbf{U}_{s-1};Y) + I(X_i;Y|\mathbf{U}_{s-1}). \quad (3)$$

In Equation 3, note that $I(\mathbf{U}_{s-1};Y)$ does not change when trying different $X_i \in \mathbf{V} \setminus \mathbf{U}_{s-1}$. Hence, the maximization of $I(\mathbf{U}_{s-1}, X_i;Y)$ in our method is actually maximizing $I(X_i;Y|\mathbf{U}_{s-1})$, as shown by the shaded region in Figure 1, which is the conditional mutual information of X_i and Y given the already selected features $\mathbf{U}_{s-1}$, i.e., the information of Y not captured by $\mathbf{U}_{s-1}$ but carried by X_i. As shown in Figure 1 (b), if the new feature B is a redundant feature, i.e., $I(\mathbf{U}_{s-1};B)$ is large, then the additional information of Y carried by X_i, $I(B;Y|\mathbf{U}_{s-1})$, will be small. Consequently, B is unlikely to be chosen as an EA. Hence, the redundant features are automatically eliminated by maximizing $I(\mathbf{U}_{s-1}, X_i;Y)$.

D. Training

A classification problem is to learn or approximate a function, which takes the values of attributes (except the class attribute) in a new sample as input and output a categorical value indicating its class, from a given training data set. The goal of the training process is to obtain a function which makes the output value of this function be the class value of the new sample as accurately as possible. From Theorem 4, the problem is converted to finding a subset of attributes $\mathbf{U} \subseteq \mathbf{V}$ whose mutual information with Y is equal to the entropy of Y from the training data sets.

TABLE I.
THE TRAINING DATA SET $\mathbf{T}$ OF THE EXAMPLE TO LEARN $Y = (A \cdot C) + (A \cdot D)$.

ABCD	*Y*	*ABCD*	*Y*	*ABCD*	*Y*	*ABCD*	*Y*
0000	0	0100	0	1000	0	1100	0
0001	0	0101	0	1001	1	1101	1
0010	0	0110	0	1010	1	1110	1
0011	0	0111	0	1011	1	1111	1

For n discrete variables, there are totally 2^n subsets. Clearly, it is NP-hard to examine all possible subsets exhaustively. It is often the case that there are some irrelevant and redundant features in the domain $\mathbf{V}$. Therefore, it is reasonable to reduce the searching space by considering those subsets with limited number of features. Hence, the problem can be solved in polynomial time. In the DFL algorithm, we introduce a parameter, called the expected cardinality of the EAs K, to restrict the searching space.

To efficiently find the $\mathbf{U}$, we propose the Discrete Function Learning algorithm. The main steps and analysis of the DFL algorithm are given in our early work [9], [15]. Here, we will briefly reintroduce the DFL algorithm with an example, as shown in Figure 2. The DFL algorithm has two parameters, the expected cardinality K and the ϵ value. The ϵ value will be introduced in the next section.

The K is the expected maximum number of attributes in the classifier. The DFL algorithm uses the K to prevent the exhaustive searching of all subsets of attributes by checking those subsets with less than K attributes. If K is specified as n, then the DFL algorithm will perform exhaustive search of all subsets of $\mathbf{V}$, which is useful when n is small. When trying to find the EAs from all subsets, the DFL algorithm will examine whether $I(\mathbf{X};Y) = H(Y)$. If so, the DFL algorithm will stop its searching process, and obtain the classifiers by deleting the non-essential attributes and duplicate rows in the training data sets. In the DFL algorithm, we use the following definition, called Δ supersets.

Definition 1: Let $\mathbf{X}$ be a subset of $\mathbf{V} = \{X_1, \ldots, X_n\}$, then $\Delta_i(\mathbf{X})$ of $\mathbf{X}$ are the supersets of $\mathbf{X}$ so that $\mathbf{X} \subset \Delta_i(\mathbf{X})$ and $|\Delta_i(\mathbf{X})| = |\mathbf{X}| + i$.

In this example, the set of attributes is $\mathbf{V} = \{A, B, C, D\}$ and the class attribute is determined with $Y = (A \cdot C) + (A \cdot D)$, where "$\cdot$" and "+" are logic AND and OR operation respectively. The expected cardinality K is set to 4 for this example. The training data set $\mathbf{T}$ of this example is shown in Table I.

In Figure 2, the union of all subsets with 1 variable is defined as the first layer $\mathcal{L}_1$, the union of all subsets with 2 variables is defined as the second layer $\mathcal{L}_2$, and

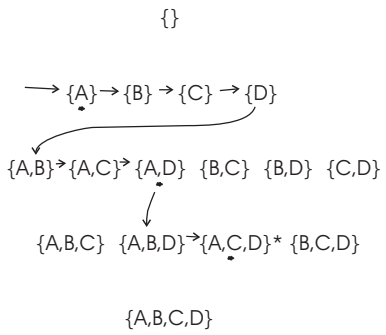

Figure 2. Search procedures of the DFL algorithm when learning $Y = (A \cdot C) + (A \cdot D)$. $\{A, C, D\}^*$ is the target combination. The combinations with a black dot under them are the subsets which share the largest mutual information with Y on their layers. Firstly, the DFL algorithm searches the first layer, then finds that $\{A\}$, with a black dot under it, shares the largest mutual information with Y among subsets on the first layer. Then, it continues to search $\Delta_1(A)$ on the second layer. Similarly, these calculations continue until the target combination $\{A, C, D\}$ is found on the third layer.

TABLE II.
THE LEARNED CLASSIFIER f OF THE EXAMPLE TO LEARN $Y = (A \cdot C) + (A \cdot D)$.

ACD	*Y*	Count	*ACD*	*Y*	Count
000	0	2	100	0	2
001	0	2	101	1	2
010	0	2	110	1	2
011	0	2	111	1	2

so on. As shown in Figure 2, the DFL algorithm searches the first layer, then it sorts all subsets according to their mutual information with Y on the first layer. It finds that $\{A\}$ shares the largest mutual information with Y among subsets on the first layer. Then, the DFL algorithm searches through $\Delta_1(A)$, ..., $\Delta_{K-1}(A)$, however it always decides the search order of Δ_{i+1} (A) bases on the calculation results of $\Delta_i(A)$. Finally, the DFL algorithm finds that the subset $\{A, C, D\}$ satisfies the requirement of Theorem 4, and will construct the classifier with these three attributes. Firstly, the B is deleted from training data set since it is a non-essential attribute. Then, the duplicate rows of $(\{A, C, D\}, Y)$ are removed from the training data set to obtain the final classifier f as shown in Table II. In the meantime, the counts of different instances of $(\{A, C, D\}, Y)$ are also stored in the classifier, which are used in the prediction process. From Table II, it can be seen that the learned classifier f is exactly the truth table of $Y = (A \cdot C) + (A \cdot D)$ along with the counts of rules. This is the reason for which we name our algorithm as the Discrete Function Learning algorithm.

The DFL algorithm will continue to search the $\Delta_1(C)$, ..., $\Delta_{K-1}(C)$, $\Delta_1(D)$, ..., $\Delta_{K-1}(D)$ and so on if it can not find the target subset in $\Delta_1(A)$, ..., $\Delta_{K-1}(A)$.

We use k to denote the actual cardinality of the EAs. After the EAs with k attributes are found in the subsets of cardinalities $\leq K$, the DFL algorithm will stop its searching. In our example, the K is 4, while the k is only 3, since there are only 3 EAs for the example.

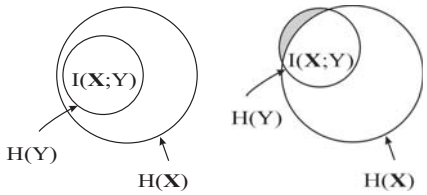

Figure 3. The Venn diagram of $H(\mathbf{X})$,$H(Y)$ and $I(\mathbf{X}, Y)$, when $Y = f(\mathbf{X})$. The circles represent the entropy of variables or vectors. The intersection between the circles represents the mutual information between the variables or vectors. The notations are the same as those in Figure 1. (a) The noiseless case, where the mutual information between $\mathbf{X}$ and Y is the entropy of Y. (b) The noisy case, where the entropy of Y is not equal to the mutual information between $\mathbf{X}$ and Y strictly. The shaded region is resulted from the noises. The ϵ value method means that if the area of the shaded region is smaller than or equal to $\epsilon \times H(Y)$, then the DFL algorithm will stop searching process, and build the classifier for Y with $\mathbf{X}$.

E. Complexity Analysis

First, we analyze the expected complexity of the DFL algorithm. Contributing to sort step in each layer of Figure 2, the DFL algorithm makes the best choice on current layer of subsets. Since there are $(n-1)$ Δ_1 supersets for a given single element subset, $(n-2)$ Δ_1 supersets for a given two element subsets, and so on. The DFL algorithm only considers $\sum_{i=0}^{k-1}(n-i) \approx k \cdot n$ subsets in the optimal case. The time complexity of the DFL algorithm is approximately $O(k \cdot n \cdot (N + \log n))$ in the optimal case, where N is the sample size and $\log n$ is for the sort step to find the subset which shares the biggest mutual information with Y in each layer of Figure 2.

Second, we analyze the worst-case complexity of the DFL algorithm. The complexity to compute the mutual information $I(\mathbf{X}, Y)$ is $O(N)$. For the example in Figure 2, $\{A, B\}$ will be visited twice from $\{A\}$ and $\{B\}$ in the worst case. $\{A, B, C\}$ will be visited from $\{A, B\}$, $\{A, C\}$ and $\{B, C\}$. Thus, $\{A, B, C\}$ will be checked for $3 \times 2 = 3!$ times in the worst case. In general, for a subset with K features, it will be checked for $K!$ times in the worst case. Hence, it takes $O((\binom{n}{1} + \binom{n}{2}2! + \ldots + \binom{n}{K}K!) \times N) = O(N \cdot n^K)$ to examine all subsets in $\mathcal{S}_K$. Another computation intensive step is the sort step. In $\mathcal{L}_1$, there is only one sort operation, which takes $O(n \log n)$ time. In $\mathcal{L}_2$, there would be n sort operations, which takes $O(n^2 \log n)$ time. Similarly, in $\mathcal{L}_K$, the sort operation will be executed for n^{K-1} times, which takes $O(n^K \log n)$ time. Therefore, the total complexity of the DFL algorithm is $O((N + \log n) \cdot n^K)$ in the worst case.

F. The ϵ Value Criterion

In Theorem 4, the exact functional relation demands the strict equality between the entropy of Y, $H(Y)$ and the mutual information of $\mathbf{X}$ and Y, $I(\mathbf{X}; Y)$. However, this equality is often ruined by noisy data, like microarray gene expression data. In these cases, we have to relax the requirement to obtain a best estimated result. As shown in Figure 3, by defining a significant factor ϵ, if the difference between $I(\mathbf{X}; Y)$ and $H(Y)$ is less than

$\epsilon \times H(Y)$, then the DFL algorithm will stop the searching process, and build the classifier for Y with $\mathbf{X}$ at the significant level ϵ.

Because the $H(Y)$ may be quite different for various classification problems, it is not appropriate to use an absolute value, like ϵ, to stop the searching process or not. Therefore, we use the relative value, $\epsilon \times H(Y)$, as the criterion to decide whether to stop the searching process or not.

The main idea of the ϵ value criterion method is to find a subset of attributes which captures not all the diversity of the class attribute $H(Y)$, but the major part of it, i.e. $(1-\epsilon) \times H(Y)$, then to build classifiers with these attributes. The features in vectors, which have strong relations with Y, are expected to be selected as EAs in the ϵ value method.

The ϵ value method is very helpful to avoid the exhaustive when dealing with noisy data sets. Because there is not subset that satisfies Theorem 4 in all subsets of $\mathbf{V}$ when the data sets are noisy. After introducing proper ϵ value, the DFL algorithm will just check the n subsets with one variable, and $n-1$ subsets with two variables, and so on. Thus, the complexity of the DFL algorithm is still $O((k \cdot n \cdot (N + \log n))$.

The ϵ value method can help to avoid over-fitting of the training data sets. For a given noisy data set, the missing part of $H(Y)$ is determined, so there exists a threshold value of ϵ with which the DFL algorithm can find the correct input variables $\mathbf{X}$ of the generation function $Y = f(\mathbf{X})$. From Theorem 3, it is known that more variables tend to contain more information about the class attribute Y. On the other hand, from Figure 3, it can be seen that some part of $H(Y)$ is not captured by the input variables $\mathbf{X}$ due to the noise. Therefore, it is likely to include more than necessary number of feature as EAs, if we continue to add variables after the threshold value of ϵ. The unnecessary input variables often incur complex models and risks of over-fitting the training data sets. By introducing the ϵ value method, the DFL algorithm will stop the searching procedure when the missing part of $H(Y)$ is smaller than or equal to $\epsilon \times H(Y)$, and avoids the inclusion of unnecessary input variables.

An example is given in Figure 4, which is generated with the LED+17 data set [16] with 3000 samples. The LED+17 data set has 23 Boolean features, 7 relevant and 16 irrelevant. We randomly choose 2000 samples as the training data set and the remaining 1000 as testing data set. From Figure 4 (b), it is seen that when ϵ is small, k is large, much larger than the actual relevant number of features, seven. Meanwhile, the prediction performance of these complex models are bad, as shown in Figure 4 (a), although using much more time as in Figure 4 (c). When choosing the ϵ_{op} of 0.31, the DFL algorithm correctly finds the seven relevant features and reaches its best performance of 72.3% in a 10-fold cross validation and 75.4% for the independent testing data set. The optimal ϵ value ϵ_{op} is automatically chosen from the training data set with the restricted learning method to be introduced in Section II-G.2.

G. Selection of Parameters

We discuss how to select the two parameters, the expected cardinality of the EAs K and the ϵ value, of the DFL algorithm in this section.

1) Selection of The Expected Cardinality K: We discuss the selection of the expected cardinality K in this section. Generally, if a data set has a large number of features, like several thousands, then K can be assigned to a small constant, like 20, since the models with large number of features will be very difficult to understand. If the number of features is small, then the K can be directly specified to the number of features n.

Another usage of K is to control model complexity. If the number of features is more important than accuracy, then a predefined K can be set. Thus, the learned model will have less than or equal to K features.

The expected cardinality K can also be used to incorporate the prior knowledge about the number of relevant features. If we have the prior knowledge about the number of relevant features, then the K can be specified as the predetermined value.

2) Selection of ϵ value: For a given noisy data set, the missing part of $H(Y)$, as demonstrated in Figure 3, is determined, i.e., there exists a specific minimum ϵ value, ϵ_m, with which the DFL algorithm can find the original model. If the ϵ value is smaller than the ϵ_m, the DFL algorithm will not find the original model. Here, we will introduce two methods to efficiently find ϵ_m.

In the first method, the ϵ_m can be found automatically by a restricted learning process. To efficiently find the ϵ_m, we restrict the maximum number of the subsets to be checked to $K \times n$. A pre-defined scope of ϵ is specified in prior. If the DFL algorithm cannot find the model for a noisy data set with the specified minimum ϵ value, then the ϵ will be increased with a step of 0.01. The restricted learning will be performed, until the DFL algorithm finds a model with a threshold value of ϵ, i.e., the ϵ_m. Since only $K \times n$ subsets are checked, the time to find ϵ_m will be $O(K \cdot n)$.

In the second method, the ϵ_m can also be found with a manual binary search method (see Supplementary Figure S1 at the supplementary website [1] of this paper). Since $\epsilon \in [0, 1)$, ϵ is specified to 0.5 in the first try. If the DFL algorithm finds a model with ϵ value of 0.5, then ϵ is specified to 0.25 in the second try. Otherwise, if the DFL algorithm cannot find a model with a long time, like 10 minutes, then the DFL algorithm can be stopped and ϵ is specified to 0.75 in the second try. The selection process is carried out until the ϵ_m value is found so that the DFL algorithm can find a model with it but cannot when $\epsilon = \epsilon_m - 0.01$. This selection process is also efficient. Since $\epsilon \in [0, 1)$, only 5 to 6 tries are needed to find the ϵ_m on the average.

[1]The supplements of this paper are available at http://www.ntu.edu.sg/home5/pg04325488/miRNA05.htm.

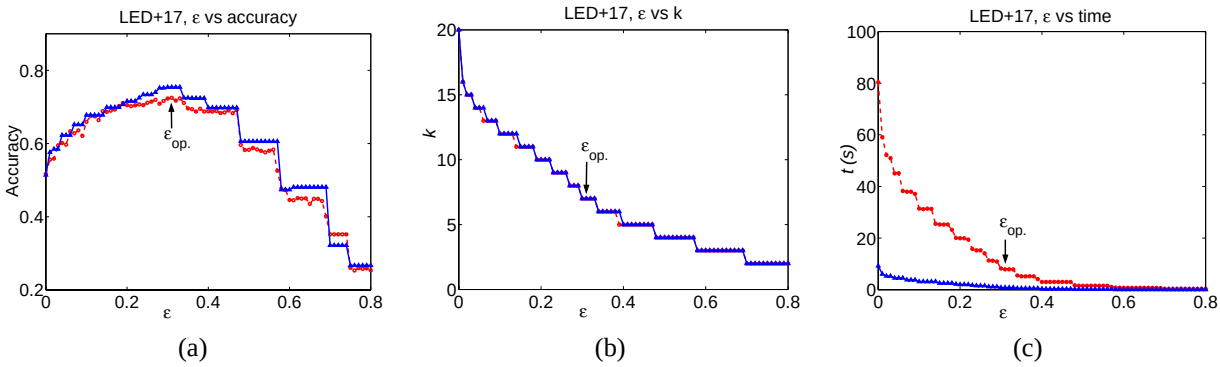

Figure 4. The performance of the DFL algorithm for different ϵ values. The figures are generated from LED+17 data sets. The training data set has 2000 samples. The curves marked with circles and triangles are for result of 10-fold cross validation and the result of an independent testing data set of 1000 samples. The ϵ_{op} pointed by an arrow is the optimal ϵ value with which the DFL algorithm reaches its highest prediction accuracy in a 10-fold cross validation for the training data set. (a) ϵ vs accuracy. (b) ϵ vs the number of selected features k. (c) ϵ vs the run time (s).

The restricted learning process can also be used to find optimal model in solving classification problems. To get optimal model, we change the ϵ value from 0 to the upper limit of the searching scope, like 0.8, with a step of 0.01. For each ϵ value, we train a model with the DFL algorithm, then validate its performance with cross validation or the testing data sets. The optimal model is the one which produces the best prediction performances. As demonstrated in Figure 4 (a), the optimal ϵ value, $\epsilon_{op} = 0.31$, is chosen from the training data set with a 10-fold cross validation. Then, this model also reaches its best performance on the testing data set, as demonstrated with the curve for the testing data set in Figure 4 (a).

H. Prediction Methods

After the DFL algorithm obtains the classifiers as function tables of the pairs $(\mathbf{u}, y)$, or called as rules, the most reasonable way to use such function tables is to check the input values $\mathbf{u}$, and find the corresponding output values y. This is due to the fact that the DFL algorithm is based on Theorem 4. As demonstrated in Section II-D, the learned model of the DFL algorithm is actually the generation function as a truth table or an estimation of it in the ϵ value method. Like the way in which people use truth tables, it is advisable to use a classification model as a truth table, or the estimation of it, with the weighted 1-Nearest-Neighbor (1NN) algorithm [9]. Therefore, we perform predictions in the space defined by the EAs $\mathbf{U}$, called the *EA space*, with the 1NN algorithm based on the Hamming distance defined as follows.

Definition 2: Let $1(a, b)$ be an indicator function, which is 0 if and only if $a = b$, otherwise is 1. The Hamming distance between two arrays $\mathbf{A} = [a_1, \ldots, a_n]$ and $\mathbf{B} = [b_1, \ldots, b_n]$ is $Dist(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{n} 1(a_i, b_i)$.

Note that the Hamming distance [10] is dedicated to binary arrays, however, we do not differentiate between binary or non-binary cases in this thesis. In the generation process of data samples, some features may change their values due to all kinds of reasons, like noise. We use the Hamming distance as a criterion to decide the class value of a new sample, since we believe that the rule with minimum Hamming distance to the EA values of a sample contains the maximum information of the sample. Thus, the class value of this rule is the best prediction for the sample.

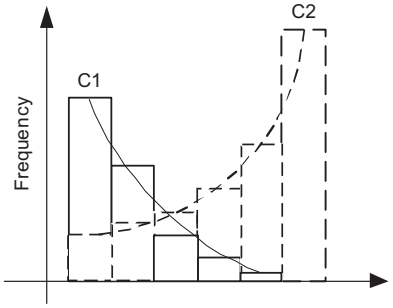

Figure 5. The noisy rules in the one-dimensional space. The rules below the two characters C1 and C2 are the genuine rules. Other rules are resulted from the noise in the data set. The vertical axis represents the frequencies of the rules in the training data sets. The rules are arranged according to their distance to the genuine rules. The solid and dashed curves are the distributions of rules for two class C1 and C2. In the real data sets, the frequencies of rules are represented by the histograms of solid and dashed lines.

In the prediction process, if a new sample is of same distance to several rules, we choose the rule with the biggest count value. The reason can be interpreted with the example shown in Figure 5. In Figure 5, it can be seen that a new sample in the region covered by two types of histograms can be of either classes. However, it is statistically more reasonable to believe the sample has the class value of a rule with higher frequency in the training data set. Hence, we call the 1NN algorithm used in our method as the weighted 1NN algorithm.

It is clear that the complexity of the weighted 1NN algorithm is mainly related to three parameters, the number of rules r in learned classifier f, the number of samples M in the testing data set and the actual cardinality of the EAs k. To compute the Hamming distance, it is sufficient to scan the two sequence with $O(k)$ time. Each sample in the testing data set is evaluated with respect to the r rules in the classifier, which takes $O(k \times r)$ steps. Thus, the total complexity of the weighted 1NN algorithm is

$O(k \cdot r \cdot M)$.

Although there exists the probability that some instances of the EAs in the testing data set are not covered by the training data set, the 1NN algorithm still gives the most reasonable predictions for such samples.

III. Results

In this section, we will first introduce the miRNA data sets. Then, we show the experimental results. We implement the DFL algorithm with the Java language version 1.4.1. All experiments are performed on an HP *AlphaServer* SC computer, with one EV68 1GHz CPU and 1GB memory, running the *Tru64* Unix operating system. All the data sets and the software of the DFL algorithm are available at the supplementary website of this paper.

A. Data Sets

In [6], 75 miRNA expression profiles from human are used to build a normal/cancer classifier. Within these 75 sample, 32 sample are normal samples from 6 different tissues: colon, kidney, prostate, uterus, lung and breast. The remaining 43 samples are tumor samples from the same 6 different tissues.

We first randomly split this data set to a training:testing ratio of 50:25, which is the D1 data set (the D1 later) in Table III. Then, in the data set D2 (the D2 later), we use the 75 samples as a training data set to build a classifier and to predict 12 testing samples from the mouse tissues. The 12 testing samples are from lung of mouse, and with the normal:tumor ratio of 5:7. In the data set D3 (the D3 later), we use 23 samples from 4 different tumor types to build a classifier and to predict an independent testing sample of 17 poorly differentiated tumors, the histological appearance of which was non-diagnostic, but for which clinical diagnosis was established by anatomical context, either directly (for example, a primary tumor arising in the colon) or indirectly (a metastasis of a previously identified primary tumor) [6].

Because the DFL algorithm is not designed for continuous data sets, we use a discretization algorithm from [17] to discretize the miRNA data sets. This method has been implemented by the *Weka*[2] software [18].

The discretization is carried out in such a way that the training data set is first discretized. Then the testing data set is discretized according to the cutting points of features (genes) determined in the training data set. After the discretization process, a substantial number of features, which are not contributing to the class distinction, are assigned with only one expression state and can be removed from the data sets. Meanwhile, the remaining discriminatory features are assigned with limited expression intervals, with the number shown in the column "D. Ftr. No." of Table III.

[2]The *Weka* software, available at http://www.cs.waikato.ac.nz/∼ml/weka/, is written with the Java language and is an open source software issued under the GNU General Public License.

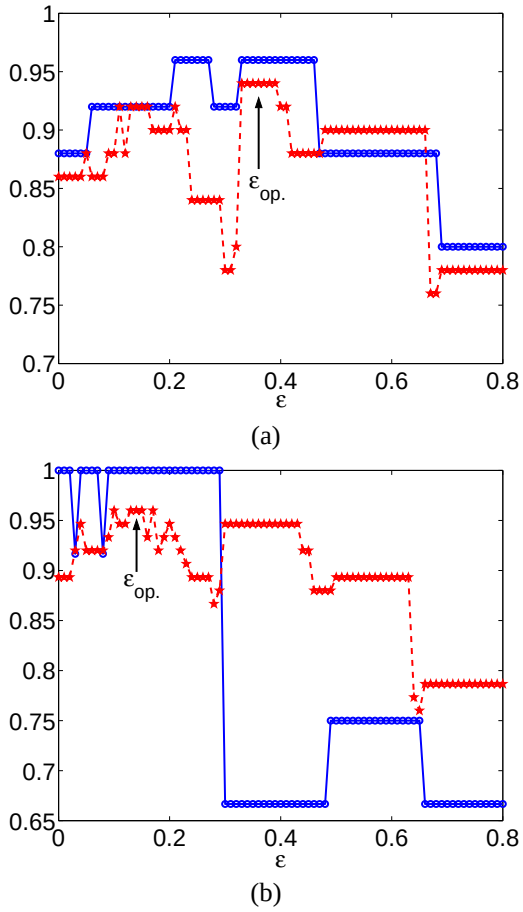

Figure 6. The ϵ vs accuracies of the DFL algorithm. The curves marked with circles and pentagrams are for the training/testing and the LOOCV respectively. In both data sets, the expected cardinality of the DFL algorithm is set to 20. The $\epsilon_{op.}$ pointed by an arrow is the optimal value that we choose. (a) For the data set D1. (b) For the data set D2.

B. Results

In all data sets, the expected cardinality of the DFL algorithm is set to 20. To obtain the optimal ϵ value, we perform the leave-one-out cross validation (LOOCV) on the training data sets for different ϵ values, from 0 to 0.8 with a step of 0.01. Then, we choose the optimal ϵ values, ϵ_{op}, with which the DFL obtains best prediction performances in the LOOCV processes, as shown in Figure 6. For the D1 and D2, we also perform the training testing validation for various ϵ values, we find that the DFL algorithm reaches its best performances for the testing data sets with the ϵ_{op} chosen in the LOOCV processes. In our implementation, this searching process to find optimal ϵ value can be done automatically. For the D3, we only perform training testing validation, since the training sample size is too small. The optimal settings and brief information of the obtained classifiers of the DFL algorithm are shown in Table III.

As shown in Figure 6, the DFL algorithm finds the optimal model with ϵ value of 0.36 and 0.14 for the data set D1 and D2 respectively. The DFL algorithm makes 1, 0 and 7 prediction errors (details available at Supplementary Table S2) for the D1, D2 and D3 data set

TABLE III.
THE MIRNA DATA SETS USED IN OUR EXPERIMENTS. THE FEATURES ARE THE EXPRESSION LEVEL OF DIFFERENT MIRNAS. THE COLUMNS FTR. NO., CLS. NO., TRAIN NO., TEST NO., D. FTR. NO., K, ϵ, k AND r ARE THE NUMBER OF FEATURES, THE NUMBER OF CLASSES, THE TRAINING SAMPLE SIZE, TESTING SAMPLE SIZE, AND THE NUMBER OF FEATURES AFTER APPLYING THE DISCRETIZATION PRE-PROCESSING METHOD [17], THE EXPECTED CARDINALITY OF THE DFL ALGORITHM, THE OPTIMAL ϵ VALUE, THE NUMBER OF FEATURES CHOSEN BY THE DFL ALGORITHM AND THE NUMBER OF RULES IN THE DFL CLASSIFIERS RESPECTIVELY.

Data Set	Ftr. No.*	Cls. No.	Train No.	Test No.	D. Ftr. No.*	Settings		Classifiers	
						K	ϵ	k	r
D1	217	2	50	25	132	20	0.36	1	4
D2	217	2	75	12	148	20	0.14	2	5
D3	217	4	23	17	42	20	0.08	3	10

* The number does not include the class attribute.

TABLE IV.
THE MIRNAS CHOSEN AS EAS BY THE DFL ALGORITHM, WITH THE SETTINGS LISTED IN TABLE III.

Data Set	miRNAs	Sequences
D1	*hsa-miR-16*	UAGCAGCACGUAAAUAUUGGCG
D2	*hsa-miR-26a*	UUCAAGUAAUCCAGGAUAGGCU
	hsa-miR-16	UAGCAGCACGUAAAUAUUGGCG
D3	*hsa-miR-7*	UGGAAGACUAGUGAUUUUGUU
	hsa-miR-130a	CAGUGCAAUGUUAAAAGGGC
	hsa-miR-135b	UAUGGCUUUUCAUUCCUAUGUG

TABLE V.
THE SUMMARY OF PREDICTION ERRORS MADE BY DIFFERENT ALGORITHMS.

Data Set	DFL	C4.5	NB	1NN	kNN*	SVM	RIP
D1 (Dis.)	1	2	3	3	3	2	2
D1 (Con.)	-	2	3	0	2	2	1
D2 (Dis.)	0	0	2	1	0	1	0
D2 (Con.)	-	4	3	1	0	1	0
D3 (Dis.)	7	10	5	4	4	6	10
D3 (Con.)	-	8	11	9	7	7	11

* The k of the kNN algorithm is set to 5.

TABLE VI.
THE COMPARISON OF MODELS FROM THE DFL, C4.5 AND RIP ALGORITHMS. THE k AND r ARE THE NUMBER OF FEATURES AND THE NUMBER OF RULES IN THE CLASSIFIERS.

Data Set	DFL		C4.5		RIP	
	k	r	k	r*	k	r
D1 (Discrete)	1	4	2	5	2	2
D1 (Continuous)	-	-	2	5	2	2
D2 (Discrete)	2	5	2	5	2	2
D2 (Continuous)	-	-	2	5	2	2
D3 (Discrete)	3	10	3	7	3	4
D3 (Continuous)	-	-	3	7	3	4

* The value shown is the number of nodes in the C4.5 tree.

TABLE VII.
THE COMPARISON OF TRAINING TIME FOR DIFFERENT CLASSIFICATION ALGORITHMS. THE VALUES ARE THE TRAINING TIMES FOR THE DISCRETIZED DATA SETS AND SHOWN IN SECOND.

	DFL	C4.5	NB	1NN	kNN	SVM	RIP
D1	0.09	0.09	0.02	0.05	0.05	0.16	0.12
D2	0.10	0.09	0.02	0.07	0.07	0.18	0.13
D3	0.09	0.09	0.02	0.03	0.03	0.32	0.02
average	0.09	0.09	0.02	0.05	0.05	0.22	0.09

respectively, as shown in Table V. With these settings, the DFL algorithm chooses the miRNAs listed in Table IV as EAs.

We use the *Weka* software (version 3.4) to evaluate the performance of other classification methods. Specifically, we compare the DFL algorithm to the C4.5 algorithm by Quinlan [19], the Naive Bayes (NB) algorithm described by Langley *et al.* [11], the 1NN and k-Nearest-Neighbors (kNN) algorithm by Aha *et al.* [7], the Support Vector Machines (SVM) algorithm by Platt [20] and the Ripper algorithm (RIP) by Cohen [21]. All these methods are implemented in the *Weka* software. The prediction errors of all compared algorithms are listed in Table V, where the values for all algorithms are the average of 10 runs.

As shown in Table V, the DFL algorithm makes only one prediction error in the D1, which is better than most other compared algorithms. In the D2, the DFL algorithm correctly classifies the 12 testing samples. In the D3, the DFL algorithm makes a slightly more prediction errors than the NB, 1NN, kNN and SVM algorithm in the discretized data sets. But the DFL algorithm is still one of the best if compared to other algorithms for the continuous D3 data set.

Accuracy is only one aspect of the quality of the classification models. To comprehensively compare classifiers, the comparisons of model complexity and training time are also important. Next, we compare the models from different algorithms in Table VI. We only compare the models of the DFL, C4.5 and RIP algorithms. The NB, 1NN, kNN and SVM algorithms build very complex models, using all features of the data sets. The complex models from these algorithms make it difficult for the users to understand which set of miRNAs is really important in contributing to the class distinctions between samples. As shown in Table VI, the complexity of models from the DFL, C4.5 and RIP algorithms are comparable. But, the DFL obtains better prediction performances than the C4.5 and RIP algorithms, as shown in Table V.

It is interesting to point out that for the D2, the RIP algorithm chooses the same features as those chosen by the DFL algorithm. These two algorithms both correctly classify all the samples in the testing data set, which suggests that the two features, *hsa-miR-26a* (EAM263) and *hsa-miR-16* (EAM115), are critical for differentiating the normal and cancer tissues.

Finally, we compare the training time of different classification algorithm in Table VII. Since all compared algorithms are implemented with the Java language and

TABLE VIII.
THE DFL CLASSIFIER FOR THE DATA SET D2.

hsa-miR-16	*hsa-miR-26a*	Class	Count
(11.5171-∞)	(11.49575-∞)	NORMAL	31
(-∞-11.5171]	(-∞-11.49575]	NORMAL	1
(-∞-11.5171]	(-∞-11.49575]	TUMOR	37
(-∞-11.5171]	(11.49575-∞)	TUMOR	3
(11.5171-∞)	(-∞-11.49575]	TUMOR	3

all experiments are performed on the same computer, the comparisons of their efficiency are meaningful. As shown in Table VII, the DFL algorithm is less efficient than the NB, 1NN and kNN algorithms, but more efficient than the SVM algorithm, and uses almost the same time the C4.5 and RIP algorithms. The training processes of the 1NN and kNN algorithms are efficient, but these two algorithms use much more time in prediction.

In summary, the DFL algorithm obtains comparable prediction performances to other compared methods, but uses more compact models.

We also compare the results of the DFL algorithm to those reported in the literature. Lu *et al.* [6] performed a top-ranking feature selection based on t-statistic, and chose 187 features. Because the top-ranking feature selection methods include many redundant features which carry similar information about the class attribute [9], the kNN classifier in [6] contained more features than the model learned by the DFL algorithm. The kNN classifier in [6] also correctly predicted the 12 testing samples in the D2. In summary, the prediction performance of the DFL algorithm is comparable to the method in [6], but the DFL classifier is more concise than the model in [6].

C. Detailed Analysis

In the D2, we find that two miRNAs, *hsa-miR-26a* (EAM263) and *hsa-miR-16* (EAM115), are very informative. In the D2, The mutual information between these two genes and the class attribute is 0.7 bits, contributing to 71% of the diversity/entropy of the class attribute, which is 0.98 bits. When considered as a vector, they capture 91% of the total diversity of the class attribute. The classifier built with the DFL algorithm for the D2 is shown in Table VIII. For instance, the first rule in Table VIII means that if the expression level of *hsa-miR-16* is larger than 11.5171 and the expression level of *hsa-miR-26a* is larger than 11.49575 in a sample, then the corresponding class value of this sample is NORMAL. And this rule has happened for 31 times out of the 75 samples in the training data set.

In Figure 7, we further investigate the expression patterns of the miRNAs chosen by the DFL algorithm for the D1 and D2. From Figure 7 (a), it is shown that the DFL algorithm makes 3 and 1 prediction errors in the LOOCV and training/testing validation of the D1 respectively. In Figure 7 (b), it is shown that the classifier in Table VIII correctly predicts the 12 samples from mouse tissues. As shown in Figure 7 (b), *hsa-miR-26a* and *hsa-miR-16* have higher expression levels in normal tissues than

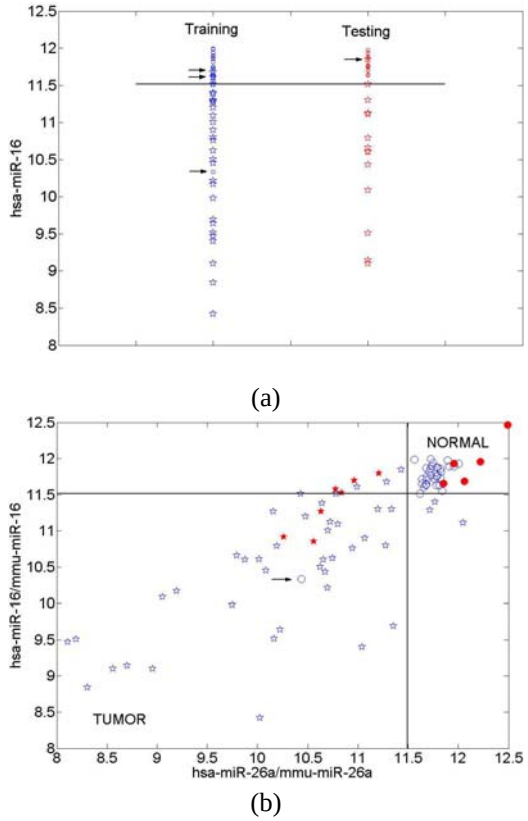

Figure 7. The expression values of the miRNAs chosen by the DFL algorithm in the D1 and D2. Normal and tumor samples are represented with circles, and pentagrams respectively. In part (b), hollow and solid samples are from training and testing data sets respectively. The black solid lines are the cutting points of the genes introduced in the discretization preprocessing. (a) The expression values of *hsa-miR-16* in the D1. The samples pointed by arrows are incorrect predictions, where the left side is for the LOOCV in the training data set and the right side is for the training/testing validation. (b) The expression values of *hsa-miR-26a* and *hsa-miR-16* in the data set D2, whose homologue genes of mouse are *mmu-miR-26a* and *mmu-miR-16*. The sample pointed by an arrow is an incorrect prediction in the LOOCV.

in tumor tissues, since most normal samples are located in the upper-right region of the space defined by the two miRNAs, *hsa-miR-26a* and *hsa-miR-16*. There is one outlier sample, pointed by an arrow, which is an incorrect prediction during the LOOCV. On the other hand, most tumor samples are located in the lower-left region of the space defined by the two miRNAs, *hsa-miR-26a* and *hsa-miR-16*, which means that the two miRNAs generally have lower expression levels in tumor tissues than in normal tissues.

IV. DISCUSSION

We have demonstrated that there exist simple and informative miRNA expression patterns, which can be used to classify normal against tumor tissues. These patterns obtain better or comparable prediction performances to the models learned from other classification methods compared in this paper and results reported in the literature.

The miRNAs in the patterns learned with the DFL algorithm generally show higher and lower expression levels in the normal and tumor tissues respectively, which is consistent with the results in the literature [6], [22].

It has been reported that the many miRNAs are phylogenetically conserved [1]. In this research, we find that the expression patterns of the two miRNAs, *hsa-miR-26a* and *hsa-miR-16*, built with human miRNA expression profiles can accurately predict the miRNA normal/tumor samples from mouse. This suggests that the expression patterns of these miRNAs are also highly conserved in vertebrates during the evolution process.

REFERENCES

[1] V. Ambros, "The functions of animal micrornas," *Nature*, vol. 431, pp. 350–5, 2004.

[2] D. P. Bartel, "Micrornas: Genomics, biogenesis, mechanism, and function," *Cell*, vol. 116, pp. 281–297, 2004.

[3] I. Alvarez-Garcia and E. A. Miska, "Microrna functions in animal development and human disease," *Development*, vol. 132, pp. 4653–62, 2005.

[4] R. I. Gregory and R. Shiekhattar, "MicroRNA Biogenesis and Cancer," *Cancer Res*, vol. 65, no. 9, pp. 3509–3512, 2005.

[5] L. He, J. Thomson, M. Hemann, E. Hernando-Monge, D. Mu, S. Goodson, S. Powers, C. Cordon-Cardo, S. Lowe, G. Hannon, and S. Hammond, "A microrna polycistron as a potential human oncogene," *Nature*, vol. 435, pp. 828–33, 2005.

[6] J. Lu, G. Getz, E. A. Miska, E. Alvarez-Saavedra, J. Lamb, D. Peck, A. Sweet-Cordero, B. L. Ebert, R. H. Mak, A. A. Ferrando, J. R. Downing, T. Jacks, H. R. Horvitz, and T. R. Golub, "Microrna expression profiles classify human cancers," *Nature*, vol. 435, pp. 834–8, 2005.

[7] D. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[8] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, pp. 109–118, 1990.

[9] Y. Zheng and C. K. Kwoh, "Identifying simple discriminatory gene vectors with an information theory approach," in *Proceedings of the 4th Computational Systems Bioinformatics Conference, CSB 2005*. IEEE Computer Society Press, 2005, pp. 12–23.

[10] R. Hamming, "Error detecting and error correcting codes," *Bell System Technical Jounral*, vol. 9, pp. 147–160, 1950.

[11] P. Langley, W. Iba, and K. Thompson, "An analysis of bayesian classifiers," in *National Conference on Artificial Intelligence*, 1992, pp. 223–228.

[12] C. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana, IL.: University of Illinois Press, 1963.

[13] T. Cover and J. Thomas, *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[14] R. McEliece, *The Theory of Information and Coding: A Mathematical Framework for Communication*, ser. Encyclopedia of Mathematics and Its Applications, G.-C. Rota, Ed. Reading, MA.: Addison-Wesley Publishing Company, 1977, vol. 3.

[15] Y. Zheng and C. Kwoh, "Dynamic algorithm for inferring qualitative models of gene regulatory networks," in *Proceedings of the 3rd Computational Systems Bioinformatics Conference, CSB 2004*. Stanford, CA: IEEE Computer Society Press, 2004, pp. 353–362.

[16] C. Blake and C. Merz, "UCI repository of machine learning databases," University of California, Irvine, Dept. of Information and Computer Sciences, Tech. Rep., 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[17] U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI-93*, Chambery, France, 1993, pp. 1022–1027.

[18] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. Witten, "Data mining in bioinformatics using Weka," *Bioinformatics*, vol. 20, no. 15, pp. 2479–2481, 2004.

[19] J. Quinlan, *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.

[20] J. Platt, *Advances in kernel methods: support vector learning*. MIT Press, 1999, ch. Fast training of support vector machines using sequential minimal optimization, pp. 185–208.

[21] W. W. Cohen, "Fast effective rule induction," in *Proc. 12th International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 115–123.

[22] P. S. Meltzer, "Cancer genomics: Small rnas with big impacts," *Nature*, vol. 435, pp. 745–6, 2005.

Yun Zheng is a research scholar of Department of Computer Science and Engineering, Washington University in St. Louis. He was a research fellow of Department of Computer Science, National University of Singapore. Currently, he is investigating how to find microRNA precursors from DNA sequences. He used to do research of learning gene regulatory networks, cancer classification based on biological data sets, learning Boolean networks/functions, and feature selection in his PhD candidature at Nanyang Technological University (NTU), Singapore. He obtained his B.Eng. from Beijing University of Aeronautics and Astronautics, China, in 1998. He can be contacted at zhengy@cse.wustl.edu.

Chee Keong Kwoh is an associate professor of computer engineering at Nanyang Technological University in Singapore. His research interests include: applying statistical learning theory in bioinformatics research, statistical learning theory, particularly in protein-protein interaction networks; probabilistic inference, numerical expert systems, particularly in the area of probabilistic reasoning; and augmented reality systems and image processing in biomedical applications. Dr Kwoh is the Deputy Director, Biomedical & Pharmaceutical Engineering (BPE) Cluster of NTU. He was the Deputy Director of Biomedical Engineering Research Center, NTU (between 1997 to 2003) and has been the director, MSc in Bioinformatics since 2002. He can be contacted at asckkwoh@ntu.edu.sg.