

# Identifying Simple Discriminatory Gene Vectors with An Information Theory Approach

Zheng Yun and Kwoh Chee Keong

BIRC, School of Computer Engineering, Nanyang Technological University, Singapore 639798  
+65-67906613, +65-67906057, pg04325488@ntu.edu.sg, asckkwoh@ntu.edu.sg

## Abstract

*In the feature selection of cancer classification problems, many existing methods consider genes individually by choosing the top genes which have the most significant signal-to-noise statistic or correlation coefficient. However the information of the class distinction provided by such genes may overlap intensively, since their gene expression patterns are similar. The redundancy of including many genes with similar gene expression patterns results in highly complex classifiers. According to the principle of Occam's razor, simple models are preferable to complex ones, if they can produce comparable prediction performances to the complex ones. In this paper, we introduce a new method to learn accurate and low-complexity classifiers from gene expression profiles. In our method, we use mutual information to measure the relation between a set of genes, called gene vectors, and the class attribute of the samples. The gene vectors are in higher-dimensional spaces than individual genes, therefore, they are more diverse, or contain more information than individual genes. Hence, gene vectors are more preferable to individual genes in describing the class distinctions between samples since they contain more information about the class attribute. We validate our method on 3 gene expression profiles. By comparing our results with those from literature and other well-known classification methods, our method demonstrated better or comparable prediction performances to the existing methods, however, with lower-complexity models than existing methods.*

**Keywords:** Feature Selection, Mutual Information, Cancer Classification, Gene Expression

## 1 Introduction

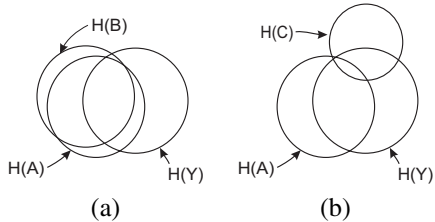
The inclusion of irrelevant, redundant and noisy attributes in the model building process phase can result in poor predictive performance and increased computation

[13]. Gene expression profiles are often noisy and contain thousands of features, many of these features are not related to the class distinctions between tissue samples [11]. Therefore, feature selection is critical for successfully classifying tissue samples based on gene expression profiles, which are of very high-dimensionality and insufficient samples. This is the well-known problem of "the curse of dimensionality".

In this paper, we construct classification models based on discriminatory gene vectors in two steps. In the first step, we use an entropy-based discretization method [7] to remove noisy genes and effectively find the most discriminatory genes [21]. In the second step, we construct simple and accurate rules from gene expression profiles with the Discrete Function Learning (DFL) algorithm [37, 36]. 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 equals to the entropy of the class attribute, then the class attribute is a function of the vector.

The mutual information [29] (Equation 1 in section 2) can be used to measure the relation between a variable and a vector. This merit makes it suitable to measure the relation between a vector of genes, which may have some kind of relations themselves, and the class attribute. As shown in Figure 1, the individual gene  $B$  shares more mutual information with the class attribute  $Y$  than gene  $C$  does, however, the combination of  $\{A, B\}$  contains less mutual information when compared with the combination of  $\{A, C\}$ . This is due to the fact that there exists a strong correlation between gene  $A$  and gene  $B$ . In gene expression profiles, such strong correlation between gene  $A$  and gene  $B$  does happen, e.g., the co-regulated genes tend to have similar expression patterns. Therefore, they have very strong correlations, or large mutual information. When one of the co-regulated genes is responsible for the class distinctions between samples, all the co-regulated genes of it may also contribute a lot to the class distinctions individually. However, from the above analysis, it is obviously neither optimal nor necessary to include all the co-regulated genes in the classifier.

In comparison, the DFL algorithm efficiently finds the most discriminatory gene vector by checking whether its



**Figure 1. The advantage of using mutual information to choose the most discriminatory gene vectors. The circles represent the entropy [29] of the genes ( $A$ ,  $B$  and  $C$ ) and class attribute ( $Y$ ). The intersections between the circles stand for the mutual information [29] between the genes, or between the genes and the class attribute. (a) Both gene  $A$  and gene  $B$  share very large mutual information with the class attribute  $Y$ . But gene  $A$  and gene  $B$  have a large mutual information which indicates similar expression pattern, i.e., strong relation between them. (b) Gene  $C$  share less mutual information with  $Y$  than gene  $B$  does. However, the tuple  $\{A, C\}$  shares larger mutual information with  $Y$  than the tuple  $\{A, B\}$  does.**

mutual information with the class attribute satisfy the theorem. We name the subset of the attributes (genes) in the most discriminatory gene vectors as the *essential attributes*, or the EAs for short. 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 [1]. Specifically, in predicting a new sample, the Hamming distances [14] (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.

Three gene expression profiles are selected to validate our method. As to be shown in section 5, the DFL algorithm achieves comparable or more competitive prediction performances than those of some other well-known classification methods with very simple and understandable rules. Our method also demonstrates comparable or more competitive prediction performance with simpler models than other methods in the literature [2, 10, 11, 19, 21, 34].

The remainder of this paper is organized as follows. First, we will review current feature selection methods and

related work of our method in section 2. Second, we will briefly introduce the DFL algorithm in section 3. Third, we describe the entropy-based discretization method [7] in section 4. Fourth, we show the experimental results for the selected data sets in section 5. Fifth, we discuss the differences between our methods and other classification methods and feature selection methods in section 6. Finally, we summarize this paper in the last section.

## 2 Background

### 2.1 Feature Selection Categorization

Feature selection methods fall into two main categories, those evaluating individual features and those evaluating feature subsets.

In the individual feature selection methods, the evaluation statistics for each feature are calculated, then a feature ranking list is provided in predefined order of the statistics. The statistics used for individual feature selection include information gain [13, 22, 34], signal-to-noise (S2N) statistic [2, 10, 11, 30], correlation coefficient (CC) [31],  $t$ -statistic [22],  $\chi^2$ -statistic [19, 22]. The main shortcoming of these individual feature selection methods lies in that a larger than necessary number of redundant top features with similar gene expression patterns are selected to build the models. Hence, such choice often brings much redundancy to the models, since the selected features carry similar information about the class attribute. According to the principle of Occam's razor, these models are not optimal although accurate, since they are often complex and suffer the risk of overfitting the data sets [34]. In addition, the large number of genes in the predictors makes it difficult to know which genes are really useful for recognizing different classes.

In the feature subset selection method, a search algorithm is often employed to find the optimal feature subsets. In evaluating a feature subset, a predefined score is calculated for the feature subset. Since the number of feature subsets grows exponentially with the number of features, heuristic searching algorithms, such as forward selection, are often employed to solve the problem. Examples of feature subset selection methods are CFS (Correlation-based Feature Selection) [12], CSE (Consistency-based Subset Evaluation) [23], the WSE (Wrapper Subset Evaluation) [16]. Most feature subset selection methods use heuristic scores to evaluate feature subset under consideration, such as CFS and CSE methods. The WSE method evaluates a subset of genes by applying a target learning algorithm to the training data set with cross validation, and selects the subset of genes which produces the highest accuracy in the cross validation process. The evaluation with cross validation makes the WSE very inefficient when meeting the high-dimensional data sets like gene expression profiles.

There is another popular way of categorizing these algorithms, called “filter” and “wrapper” methods [15], based on the different nature of the metric used to evaluate features. In the filter methods, the feature selection is performed as a preprocessing step and often independent of the classification algorithms which will be applied to the processed data sets later. The WSE method mentioned above is the wrapper method.

## 2.2 Theoretic Background

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  $|\mathbf{X}|$ . In the remainder parts of this paper, we denote the attributes except the class attribute as a set of discrete random variables  $\mathbf{V} = \{X_1, \dots, X_n\}$ , the class attribute as variable  $Y$ .

The entropy of a discrete random variable  $X$  is defined in terms of probability of observing a particular value  $x$  of  $X$  as [29]:

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

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 [29]:

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

Unlike S2N or CC, 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.

The conditional mutual information  $I(X; Y|Z)$  [4](the mutual information between  $X$  and  $Y$  given  $Z$ ) 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 2.1, for which the proof is available in [4].

### Theorem 2.1

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

## 2.3 Related Work

Some feature selection methods based on mutual information have been introduced. These methods also fall into two categories.

In the first category, features are ranked according to their mutual information with the class label. Then, the first  $k$  features [6] or the features with a bigger mutual information than a predefined threshold value [35] are chosen.

The second category is feature subset selection methods. In this category, the forward selection searching algorithm is often used to find the predefined  $k$  features. In the first iteration, the  $X_i$  which shares the largest mutual information with  $Y$  is selected to the target feature subset  $\mathbf{U}$ . Then, in the next step, the selection criterion is how much information can be added with respect to the already existing  $X_{(1)}$ . Therefore, the  $X_{(2)}$  with maximum  $I(X_i, X_{(1)}; Y) - I(X_{(1)}; Y)$  is added to  $\mathbf{U}$  [32]. Formally, the features  $X_{(1)}, \dots, X_{(k)}$  are selected with the following criteria,  $X_{(1)} = \operatorname{argmax}_i I(X_i; Y)$  and

$$X_{(l)} = \operatorname{argmax}_{X_i \in \mathbf{P}_l} \min_{X_{(j)} \in \mathbf{U}_l} (I(X_i, X_{(j)}; Y) - I(X_{(j)}; Y)) \quad (3)$$

where  $\forall l, 1 < l \leq k, i = 1, \dots, (n - l + 1), j = 1, \dots, (l - 1)$ , and  $\mathbf{P}_l$  is the feature pool by removing  $X_{(1)}, \dots, X_{(l)}$ ,  $\mathbf{P}_1 = \mathbf{V} \setminus X_{(1)}$ ,  $\mathbf{P}_{l+1} = \mathbf{P}_l \setminus X_{(l)}$ , and  $\mathbf{U}_l$  is the set of selected features  $\mathbf{U}_1 = \{X_{(1)}\}$ ,  $\mathbf{U}_{l+1} = \mathbf{U}_l \cup \{X_{(l)}\}$ .

From Theorem 2.1, we have

$$I(X_i, X_{(j)}; Y) = I(X_{(j)}; Y) + I(X_i; Y|X_{(j)}),$$

then

$$I(X_i; Y|X_{(j)}) = I(X_i, X_{(j)}; Y) - I(X_{(j)}; Y). \quad (4)$$

Therefore, Equation 3 is equivalent to maximizing conditional mutual information,  $\min_{X_{(j)} \in \mathbf{U}} I(X_i; Y|X_{(j)})$  [8] in Equation 4.

Battiti [3] introduced an algorithm to find the feature subsets. In this method, the mutual information  $I(X_i; Y)$  of a new feature  $X_i$  is penalized by a weighted sum of the  $I(X_i; X_{(j)})$ , where  $X_{(j)} \in \mathbf{U}$ . This method is similar to those in [8, 32], but not theoretically formulated.

The Markov Blanket method [17, 34] is another subset selection method based on information theory. The Markov

Blanket method tries to find a subset of features which minimize the distance between the distribution of the selected feature subsets and the distribution of all features. The backward selection algorithm is used to eliminate features which minimize the expected cross-entropy [17] until some predefined number of features have been eliminated.

For all subset selection method mentioned above, one major shortcoming is that the candidate feature is compared to all the selected features in  $\mathbf{U}$ , one-by-one. The motivation underlying Equation 3 and 4 is that  $X_i$  is good only if it carries information about  $Y$ , and if this information has not been caught by any of the  $X_{(j)}$  already picked [8]. However, it can not be known whether the existing features as a vector have captured the information carried by  $X_i$  or not. In addition, it also introduces some redundant computation when evaluating the new feature  $X_i$  with respect to the already picked features  $X_{(j)} \in \mathbf{U}$ , which will be discussed further in section 6.

### 3 Methods

#### 3.1 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 3.1**  $I(\{\mathbf{X}, Z\}; Y) \geq I(\mathbf{X}; Y)$ , 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 3.1 can be found in [24]. In Theorem 3.1, 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,  $H(A)$  and  $H(C)$  will definitely share no less information with  $H(Y)$  than  $H(A)$  alone, since  $H(A)$  and  $H(C)$  can provide at least the part of information about  $Y$  already provided by  $H(A)$  alone. To put it another way, the more variables, the more information is provided about another variable.

From Theorem 3.1, it can be deduced that individual genes can not provide more information about the class attribute than gene vectors. As demonstrated in Figure 1, it is obvious that choosing the top genes will not make sure that we find the optimal subset of genes which contains maximum mutual information with the class attribute. Therefore, it is better to find the optimal subset of genes by considering the genes as vectors.

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

**Theorem 3.2** 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  $\mathbf{X}$ .

Proof of Theorem 3.2 is given in our early work [36]. 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 3.2 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  $Y$ .

#### 3.2 Training of Classifiers

A classification problem is trying 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 which indicates the class of the sample under consideration, 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 3.2, 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  $Y$ . The  $\mathbf{U}$  is the EAs which we are trying to find from the data sets. For  $n$  discrete variables, there are totally  $2^n$  subsets. Clearly, it is NP-hard to examine all possible subsets exhaustively. However, in the cancer classification problems, only a small set of genes of the human genome are responsible for the tumor cell developmental pathway [25]. Therefore, it is reasonable to reduce the searching space by considering those subsets with limited number of genes.

The main steps and analysis of the DFL algorithm are given at the supplementary website<sup>1</sup> and in our early work [37]. Here, we will briefly introduce the DFL algorithm with an example, as shown in Figure 2. The DFL algorithm has two parameters, the expected cardinality  $k$  and the  $\epsilon$  value. The  $\epsilon$  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. 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  $\Delta$  supersets.

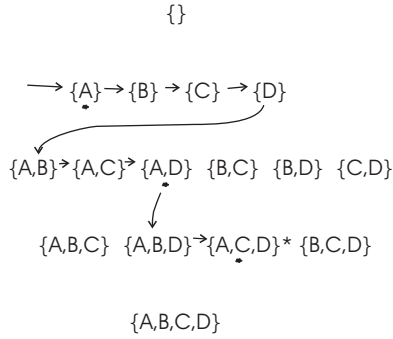
**Definition 3.1** Let  $\mathbf{X}$  be a subset of  $\mathbf{V} = \{X_1, \dots, X_n\}$ , then  $\Delta_i(\mathbf{X})$  of  $\mathbf{X}$  are the supersets of  $\mathbf{X}$  so that  $\mathbf{X} \subset \Delta_i(\mathbf{X})$  and  $|\Delta_i| = |\mathbf{X}| + i$ .

In this example, the set of attributes is  $\mathbf{V} = \{A, B, C, D\}$  and the class attribute is determined with  $Y = (A \cdot C) +$

<sup>1</sup>Supplements of this paper are available at <http://www.ntu.edu.sg/home5/pg04325488/csb2005.htm>.

**Table 1. 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



**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.**

$(A \cdot D)$ , 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 1.

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  $\Delta_1(A), \dots, \Delta_{k-1}(A)$ , however it always decides the search order of  $\Delta_{i+1}(A)$  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 3.2, 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  $\{A, C, D\} \rightarrow Y$  are

**Table 2. 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

removed from the training data set to obtain the final classifier  $f$  as shown in Table 2. In the meantime, the counts of different instances of  $\{A, C, D\} \rightarrow Y$  are also stored in the classifier, which are used in the prediction process. From Table 2, 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  $\Delta_1(C), \dots, \Delta_{k-1}(C), \Delta_1(D), \dots, \Delta_{k-1}(D)$  and so on if it can not find the target subset in  $\Delta_1(A), \dots, \Delta_{k-1}(A)$ .

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.

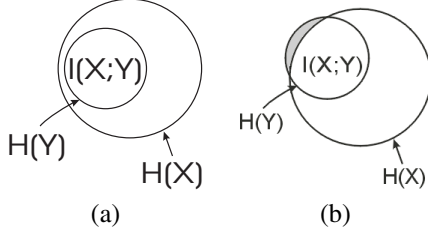
The time complexity of the DFL algorithm is approximately  $O(k^* \cdot n \cdot (N + \log n))$  on the average, where  $N$  is the sample size and  $\log n$  is for the sort step to find the subset which shares the biggest mutual information with  $Y$  in each layer of Figure 2. For detailed analysis of the complexity, see our prior work [37, 36].

In our implementation of the DFL algorithm, the  $k$  value, which can be assigned by the user, is set to a default value of 10. As to be shown in section 5, the DFL algorithm achieves good prediction performances when  $k^*$  is very small in all the experiments performed.

### 3.3 The $\epsilon$ Value Criterion

In Theorem 3.2, 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 the 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  $\epsilon$ , 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  $\epsilon$ .

Because the  $H(Y)$  may be quite different for various classification problems, it is not appropriate to use an absolute value, like  $\epsilon$ , to stop the searching process or not.



**Figure 3.** The Venn diagram of  $H(X), H(Y)$  and  $I(X, Y)$ , when  $Y = f(X)$ . (a) The noiseless case, where the mutual information between  $X$  and  $Y$  is the entropy of  $Y$ . (b) The noisy case, where the entropy of  $Y$  is not equal to the mutual information between  $X$  and  $Y$  strictly. The shaded region is resulted from the noises. The  $\epsilon$  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  $X$ .

Therefore, we use the relative value,  $\epsilon \times H(Y)$ , as the criterion to decide whether to stop the searching process or not.

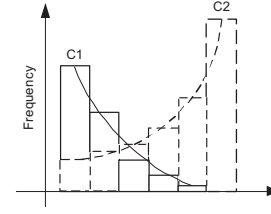
The main idea of the  $\epsilon$  value criterion method is to find a subset of attributes which captures not all the diversity of the class attribute  $H(Y)$ , but the major part of it, i.e.  $(1-\epsilon) \times H(Y)$ , then to build classifiers with these attributes. The features in vectors, which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

### 3.4 Prediction Methods

After the DFL algorithm obtaining the classifiers as function tables of the pairs  $\{u \rightarrow y\}$ , the most reasonable way to use such function tables is to check the input values  $u$ , then find the corresponding output values  $y$ . Therefore, we perform predictions in the *EA space*, with the 1NN algorithm based on the Hamming distance defined as follows.

**Definition 3.2** 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  $\mathbf{A} = [a_1, \dots, a_n]$  and  $\mathbf{B} = [b_1, \dots, b_n]$  is  $Dist(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^n 1(a_i, b_i)$ .

Note that the Hamming distance [14] is dedicated to binary arrays, however, we do not differentiate between binary or non-binary cases in this paper. 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 4.** 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 has same distance to several rules, we choose the rule with the biggest count values. The reason can be interpreted with the example shown in Figure 4. In Figure 4, 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 more reasonable to believe the sample has the class value of a rule with higher frequency in the training data set.

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

For convenience, we will express the proposed classification method as the DFL algorithm hereafter when it does not result in misunderstanding.

## 4 The Discretization Method

Gene expression data are continuous and noisy. As discussed early, to remove noisy genes, we use a widely used discretization method [7] based on entropy to discretize the expression data.

Following the notation in [5, 7], we will briefly summarize the discretization algorithm. Let partition boundary  $T$  separate set  $S$  into  $S_1$  and  $S_2$ . Let there be  $k$  classes  $C_1, \dots, C_k$ . Let  $P(C_i, S_j)$  be the proportion of examples in  $S_j$  that have class value  $C_j$ . The class entropy of a subset  $S_j$ ,  $j = 1, 2$  is defined as:

$$Ent(S_j) = - \sum_{i=1}^k P(C_j, S_j) \log P(C_j, S_j).$$

Let  $S_1$  and  $S_2$  be induced with the boundary  $T$  of attribute  $A$ , then the class information entropy of the partition is given by:

$$E(A, T; S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2).$$

For a given attribute  $A$ , the boundary  $T_{min}$  is chosen to minimize  $E(A, T; S)$  as a binary discretization boundary. This method is recursively used to the two partitions induced by  $T_{min}$ , until some stop criteria is reached, therefore creating multiple intervals on the attribute  $A$ .

The Minimum Description Length principle is used as the stop criterion of the partitioning by [7]. The recursive partitioning within a set of values  $S$  stops iff

$$Gain(A, T; S) < \frac{\log_c(N-1)}{N} + \frac{\delta(A, T; S)}{N}$$

where  $N$  is the number of instances in the set  $S$ ,  $Gain(A, T; S) = Ent(S) - E(A, T; S)$ ,  $\delta(A, T; S) = \log_2(3^k - 2) - [k \cdot Ent(S) - k_1 \cdot Ent(S_1) - k_2 \cdot Ent(S_2)]$ , and  $k_i$  is the number of class labels represented in set  $S_i$ .

After the discretization process, a substantial number of genes, which are not contributing to the class distinction, are assigned with only one expression state. Meanwhile, the remaining discriminatory genes are assigned with limited expression intervals. For example in our experiments, the *Zyxin* gene in the ALL data set is one of the genes most highly correlated with the ALL-AML class distinction [11]. In the discretization process, the expression values of the *Zyxin* gene are discretized into two intervals,  $(-\infty - 994]$  and  $(994 - \infty)$ . This method has been implemented by the *Weka* software [33, 9]. 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.

## 5 Experiments and 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. We choose 3 data sets listed in Table 3 to verify the DFL algorithm in this paper. The implementation software, data sets and their details (Table S1) are available at the supplementary website of this paper.

**Table 3. The summary of the selected data sets. The column name Att.#, C.#, Trn.#, Tst.# and Lit. are the number of attributes, the number of classes, training sample size, testing sample size and literature of the data sets.**

Data Set	Att.#	C.#	Trn.#	Tst.#	Lit.
ALL	7129	2	38	34	[11]
MLL	12582	3	57	15	[2]
DLBCL	7129	2	55	22	[30]

**Table 4. The summary of the number of genes in the selected data sets.**

Data Set	Original #	# After Discret.	# Chosen by DFL ( $k^*$ )
ALL	7129	866	1
MLL	12582	4411	2
DLBCL	7129	761	1

We will first present the discretization results. 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 genes determined in the training data set. The number of genes with more than one expression intervals, and the number of genes chosen by the DFL algorithm, i.e. the actual cardinality  $k^*$  of our classifiers, are shown in Table 4. As expected, the discretization method remove substantial amount of genes which are irrelevant to the class distinctions.

Then, we show the results of the DFL algorithm. To get optimal model, we change the  $\epsilon$  value from 0 to 0.6, with a step of 0.01. For each  $\epsilon$  value, we train a model with the DFL algorithm, then validate its performance for the testing data sets. The  $\epsilon$  vs prediction error is given in supplementary Figure S1. In our implementation of the DFL algorithm, the optimal model can be automatically chosen. As shown in Table 5, the DFL algorithm learns the optimal classifier of three rules for the ALL data set. The optimal classifiers for other data sets, prediction details, corresponding settings of the DFL algorithm (Table S2), and genes in the classifiers (Table S3), are available at the supplementary website. The incorrect predictions and the number of genes in the corresponding classifiers are given in Table 6 and Table 4 respectively. As shown in Table 6 and Table 4, the DFL algorithm finds the most discriminatory gene vectors with only a few genes, and achieves good prediction performances.

Figure 5 shows the expression values of the genes chosen by the DFL algorithm in the ALL and MLL data sets. As shown in Figure 5 (a), the classifier in Table 5 only makes two incorrect predictions in the ALL testing data set. *CST3* (Cystatin C, M27891) is one of the 50 genes most highly

**Table 5. The classifier for the ALL data set learned with the DFL algorithm.**

<i>CST3</i>	Class	Count
$(-\infty - 1419.5]$	ALL	27
$(1419.5 - \infty)$	AML	10
$(-\infty - 1419.5]$	AML	1

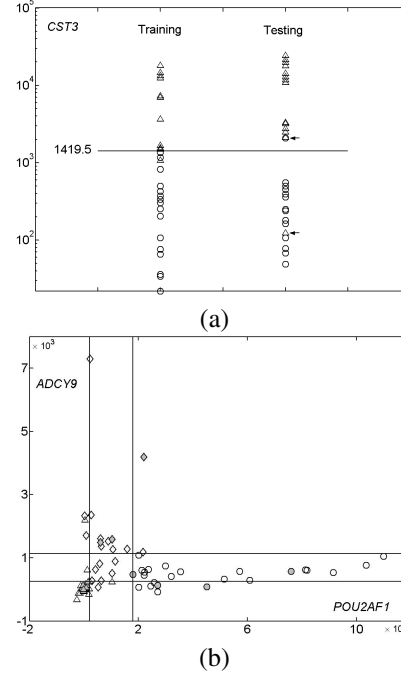
**Table 6. The comparison of prediction errors from the DFL algorithm and some well-known classification methods. The numbers shown are the incorrect predictions on discretized/continuous data sets.**

Data Set	DFL <sup>1</sup>	C4.5	NB	1NN	<i>k</i> NN <sup>2</sup>	SVM
ALL	2	3/3	5/4	8/9	6/11	6/5
MLL	0	2/3	2/0	2/3	3/2	0/0
DLBCL	1	1/4	1/4	1/4	1/2	1/1
Average	1	2/3	3/3	4/5	3/5	2/2

<sup>1</sup> The results are for the discretized data sets. <sup>2</sup> The *k* value of the *k*NN algorithm is set to 5.

correlated with the ALL-AML class distinction in the classification model of Golub *et al.* [11]. In Figure 5 (b), it can be seen that the samples in the MLL testing data set are all correctly classified in the *EA space* defined by the two genes *POU2AF1* and *ADCY9*. *POU2AF1* is one of the genes required for the appropriate B-cell development and one of the genes that are specifically expressed in MLL, ALL or AML [2]. From Figure 5 (b), it can be seen that most AML, MLL and ALL samples are located in the left, central and right regions divided by the cutting points of the *POU2AF1* expression values respectively. *ADCY9* is not as discriminative as *POU2AF1*, however, it serves as a good complement to *POU2AF1*. *POU2AF1* captures 77% diversity (entropy) of the class attribute in the MLL training data set, but the combination of *POU2AF1* and *ADCY9*, as a vector, captures 94.7% of the same measurement. For the DLBCL data set, the DFL algorithm selects *MCM7* (*CDC47* homolog) gene, which is associated with cellular proliferation and one of the genes highly correlated with the class distinctions[30]. The comparison of expression values of *MCM7* gene is given in supplementary Figure S2.

We use the *Weka* software (version 3.4) to evaluate the performances of other classification methods. Specifically, we compare the DFL algorithm with the C4.5 algorithm by Quinlan [27], the Naive Bayes (NB) algorithm by Langley *et al.* [18], the 1NN and *k*-Nearest-Neighbors (*k*NN) algorithm by Aha *et al.* [1] and the Support Vector Machines (SVM) algorithm by Platt [26]. All these methods are implemented in the *Weka* software. The comparison of the incorrect predictions from these algorithms and the DFL algorithm are shown in Table 6.



**Figure 5. The comparisons of the expression values of the genes chosen by the DFL algorithm. ALL, AML and MLL samples are represented with circles, triangles and diamonds 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 *CST3* in the ALL data set. The two samples pointed by arrows are the incorrect predictions. (b) The expression values of *POU2AF1* and *ADCY9* in the MLL data set.**

As shown in Table 6, when other methods are dealing with continuous data sets, their performances are not better than those of the DFL algorithm in most cases. For the discretized data sets, the performance of the DFL algorithm is still the best for the ALL and MLL data sets among all compared methods. For the DLBCL data set, the DFL algorithm achieves comparable performances to other methods.

In Table 7, we also compare our results with those in the literature. Golub *et al.* [11] employed a weighted-voting algorithm on the ALL data set, and made 5 prediction errors with a model of 50 genes. Furey *et al.* [10] used the SVM algorithm with 1000 selected genes to classify the ALL data set, and produced 2 to 4 prediction errors. Li *et al.* [21] made 3 prediction errors with a method called *emerging patterns* (EP) on the ALL data set. Xing *et al.* [34] found an

**Table 7. The comparison of the DFL algorithm and other methods in literature. The column names E., Al., M. and  $k^*$  stand for the number of incorrect predictions, the algorithm used, the relation measures used to do feature selection and the number of genes in the classifiers respectively. For Al. column, the WV, SVM, EP,  $k$ NN, C45, PCL and NB represent the weighted-voting, support vector machine, emerging pattern [21],  $k$ -nearest-neighbors, C4.5, Prediction by Collective Likelihoods [19] and Naive Bayes algorithm respectively. For the M. column, the S2N, E, MB and  $\chi^2$  are the *signal-to-noise* statistic [11], entropy [7], Markov Blanket [34] and  $\chi^2$ -statistic respectively. For all columns, NA stands for not available. For all data sets, the training/testing samples are the same as those in Table 3.**

Data Set	DFL	Methods in Literature				
		E.	Al.	M.	$k^*$	Literature
ALL	2	5	WV	S2N	50	[11]
		2-4	SVM	S2N	1000	[10]
		3	EP	E	1	[21]
		0	$k$ NN	MB	42	[34]
MLL	0	1	$k$ NN <sup>1</sup>	S2N	40	[2]
		3	C45	$\chi^2$	20	[19]
		1	SVM	$\chi^2$	20	[19]
		1	$k$ NN	$\chi^2$	20	[19]
		0	PCL	$\chi^2$	20	[19]
		0	NB	$\chi^2$	20	[19]
DLBCL	1	NA				

<sup>1</sup> The  $k$ NN classifier in [2] misclassified 1 sample out of 10 independent testing samples.

optimal feature subsets of about 40 genes chosen by Markov Blanket [17] and made 0 errors with the  $k$ NN classifier for the ALL data set.

Armstrong *et al.* [2] chose 40 to 250 genes, then applied the  $k$ NN algorithm to an independent testing data of 10 samples and misclassified 1 out of the 10 testing samples. For the MLL data set, Li *et al.* [19] selected 20 top-ranked gene by the  $\chi^2$  method, and made 3, 1, 1, 0 and 0 errors with the C4.5, SVM,  $k$ NN, PCL (Prediction by Collective Likelihoods) [19] and NB algorithm respectively.

We do not compare the results for the DLBCL data set with that in literature, since the evaluation data set used by us is different from that of Shipp *et al.* [30].

As shown in Table 7, the models with many top genes (methods in line 1-2,5-8) make more prediction errors than simple gene vectors of a few genes found by the DFL algorithm for the ALL and MLL data sets. For the rest cases (methods in line 3, 4, 9 and 10), the prediction performances of our approach is comparable to those in the literature.

**Table 8. The training times for discretized data sets of different classification methods. The unit is second.**

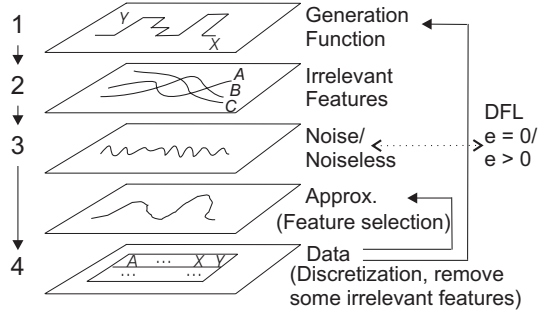
Data Set	DFL	C4.5	NB	1NN	$k$ NN	SVM
ALL	0.02	0.10	0.03	0.12	0.12	0.21
MLL	0.48	0.34	0.12	0.73	0.75	1.11
DLBCL	0.01	0.13	0.03	0.14	0.14	0.23

As mentioned early, the prediction performance is only one aspect of the classifiers, but not all. Next, we compare the model complexities of different methods. From Table 4, it can be seen the classifiers of our method are very simple, with only a few genes. The model from the C4.5 algorithm is comparable to our models (details available at the supplementary Table S4), but the performances of the C4.5 algorithm are not better than our method. The NB, 1NN,  $k$ NN and SVM algorithms build very complex models, using all genes of the data sets. The complex models from these algorithms make it difficult for the users to understand which set of genes is really important in contributing to the class distinctions between samples. When meeting multi-class data sets, such as the MLL data sets, the SVM algorithm and the NB algorithm solve the problems by building individual one-vs-all (OVA) pairwise classifiers for each class [28]. Although effective in practice, this method also makes the model even more complex than individual classifiers obtained from the SVM algorithm and the NB algorithm. In comparison, the DFL algorithm just builds one model for multi-class data sets. By comparing the number of genes ( $k^*$ ) used by different classification models in Table 4 and Table 7, it can be seen that the models in the literature are also more complex than the classifiers obtained by the DFL algorithm, with the only one exception of the model by Li *et al.* [21] for the ALL data set. For the ALL data set, although the two models use the same number of genes, our model only produces 2 prediction errors, but the model from [21] made 3 errors.

Finally, the training times of different methods are compared in Table 8. Since all compared algorithms are implemented with the Java language and all experiments are performed on the same computer, the comparisons of their efficiency are meaningful. From Table 8, it can be seen that the DFL algorithm is more efficient than other methods in most cases.

## 6 Discussion

The fundamental difference between the DFL algorithm and other classification methods lies in the underlying philosophy of the algorithms, as shown in Figure 6. What the DFL algorithm does is to estimate the classification func-



**Figure 6. The philosophy of the DFL algorithm and other classification algorithms.  $Y = f(X)$  is the generation function. The 1, 2, 3 and 4 are four steps in the production of data sets. The arrows on the left represent the production process of the data sets. In the first step, the generation function generates the original data sets. In the second and the third step, irrelevant features and noise are introduced into the data sets respectively. The arrows on the right stand for the learning philosophy of different algorithms. Other algorithms, like Multi-Layer Perceptrons and SVMs, are approximating the generation function with complex models from noisy data sets. The feature selection process is an optional step for these algorithms. However, the DFL algorithm directly estimates the generation function with low-complexity models. As indicated by the dotted arrow, when the data sets are noisy or noiseless, the DFL algorithm uses the positive or zero  $\epsilon$  values. The discretization step [7] is optional for all algorithms, and helps to remove some irrelevant features from continuous data sets.**

tions directly (based on Theorem 3.2) with low-complexity models, as demonstrated in Table 2. However, other classification methods are trying to approximate the classification functions with complex models, like what have been done by the Multi-Layer Perceptrons and the SVMs with different kernels.

The DFL algorithm can be categorized as a feature subset selection method and a filter method. However, the DFL algorithm is also different from other feature subset selection methods, like the CFS, CSE and WSE methods. Based on Theorem 3.2, the DFL algorithm can produce function tables for the training data sets, while other subset feature selection methods only generate a subset of features. Particularly, the DFL algorithm is different from existing feature

subset selection methods based on information theory in the following four aspects.

First, the stopping criterion of the DFL algorithm is different from those of existing methods. The DFL algorithm stops the searching process based on Theorem 3.2. The existing methods stop the searching process with a predefined  $k$  or threshold value of the mutual information. Hence, the feature subsets selected by existing methods may be sensitive to the  $k$  or threshold value of the mutual information.

Second, the feature subset evaluation method of the DFL algorithm is also different from those in existing methods.  $I(\mathbf{X}; Y)$  is evaluated with respect to  $H(Y)$  in the DFL algorithm. Suppose that  $\mathbf{X}$  is the already selected feature subset in  $\mathbf{U}$ , and the DFL algorithm is trying to add a new feature  $Z$  to  $\mathbf{U}$ ,  $X_{(1)} = \text{argmax}_i I(X_i; Y), i = 1, \dots, n$  and

$$X_{(l)} = \text{argmax}_Z I(\mathbf{X}, Z; Y), \quad (5)$$

where  $\forall l, 1 < l \leq k, \mathbf{U}_1 = \{X_{(1)}\}$ , and  $\mathbf{U}_{l+1} = \mathbf{U}_l \cup \{X_{(l)}\}$ . From Theorem 2.1, we have

$$I(\mathbf{X}, Z; Y) = I(\mathbf{X}; Y) + I(Z; Y|\mathbf{X}). \quad (6)$$

In Equation 6, note that  $I(\mathbf{X}; Y)$  does not change when trying different  $Z$ . Hence, the maximization of  $I(\mathbf{X}, Z; Y)$  in the DFL algorithm is actually maximizing  $I(Z; Y|\mathbf{X})$ , the conditional mutual information of  $Z$  and  $Y$  given the already selected features  $\mathbf{X}$ , i.e., the information of  $Y$  not captured by  $\mathbf{X}$  but carried by  $Z$ . Equation 6 is different from Equation 4 used in [8], where the new feature is evaluated with respect to individual features in  $\mathbf{U}$ . As intuitively shown in Figure 1, by considering the selected features as vectors, the redundancy introduced by new features to be added to  $\mathbf{U}$  is automatically eliminated.

Let us further investigate the measure,  $I(Z; Y|\mathbf{X})$ . From Equation 1, we have

$$I(Z; Y|\mathbf{X}) = H(Y|\mathbf{X}) - H(Y|Z, \mathbf{X}). \quad (7)$$

Similar to Equation 6,  $H(Y|\mathbf{X})$  does not change when trying different  $Z$ . As pointed out by Fleuret [8], the ultimate goal of feature subset selection is to find  $\{Z, \mathbf{X}\}$  which minimizes  $H(Y|Z, \mathbf{X})$ . But  $H(Y|Z, \mathbf{X})$  can not be estimated with a training set of realistic size as it requires the estimation of  $2^{k+1}$  probabilities [8]. Hence, the authors of [8, 32] proposed the estimated increase of the information content of the feature subset using Equation 3 and 4. However, from Equation 6 and 7, it can be seen that it is not necessary to compute the  $H(Y|Z, \mathbf{X})$ , as the problem can be directly solved by maximizing  $I(\mathbf{X}, Z; Y)$  as implemented in the DFL algorithm.

Furthermore, the evaluation of the feature subsets is more efficient than penalizing the new feature with respect to every selected features, as done in [3, 8, 32]. To evaluate  $I(\mathbf{X}; Y)$ ,  $O(n \cdot N)$  operations are needed when adding

each feature, and  $O(k \cdot n \cdot N)$  operations are necessary to choose  $k$  features in the DFL algorithm. However, in calculating  $I(X_i, X_{(j)}; Y) - I(X_{(j)}; Y)$  [8, 32], since there are already  $(l - 1)$  features in  $\mathbf{U}$  in the  $l$  iteration, there would be  $(l - 1) \times O(n \cdot N)$  operations in this iteration. Therefore, it needs  $\sum_{l=1}^k (l - 1) \times O(n \cdot N) \approx O(k^2 \cdot n \cdot N)$  operations to select  $k$  features, which is less efficient. The computational cost of the backward selection for the Markov Blanket is at least  $O(2^k \cdot n \cdot N)$  [17], which is even worse than the  $O(k^2 \cdot n \cdot N)$  of the forward selection in [8, 32]. In addition, the correlation matrix of all features needs to be computed in the Markov Blanket method, which costs  $O(n^2(\log n + N))$  operations.

Third, the searching method used by the DFL algorithm is also different from the forward selection searching or the backward selection searching used by methods discussed above. In the DFL algorithm, the exhaustive search of all subsets with  $\leq k$  features is guaranteed.

Fourth, the methods in [8, 32] can only deal with binary features, however, the DFL algorithm can deal with multi-value discrete features as well.

In the feature selection for cancer classification problems, we show that it is better to choose top gene vectors, or subsets of genes, not top individual genes. It is demonstrated in Figure 1 that to select the top genes individually will not make sure that we find the optimal subset of genes. From Theorem 3.1, it is known that gene vectors may contain more information about the class distinction between samples than individual genes, hence are more discriminatory than individual genes. By selecting the best gene vectors, low ranked genes can also be selected as EAs of our classifiers. As reported by Li *et al.* [20], low ranked genes are important components in building significant rules, and included in their classifiers for many data sets.

## 7 Conclusion

In this paper, we have validated the DFL algorithm on 3 benchmark gene expression profiles. We have presented that by considering the genes as vectors, the DFL algorithm can efficiently find accurate and low-complexity models on the selected data sets. Since gene vectors are more discriminatory than individual genes, the DFL algorithm avoids the redundancies of including genes with similar expression patterns in the classifiers.

In the current implementation, the DFL algorithm will stop its searching when it finds the first feature subset to satisfy  $I(\mathbf{X}; Y) = H(Y)$  or  $I(\mathbf{X}; Y) \geq (1 - \epsilon) \times H(Y)$  in the  $\epsilon$  value method. In gene expression profiles, it is possible that there exist several subsets of genes which are biologically meaningful and can give good prediction performance. In the future, the DFL algorithm can be used to find all feature vectors which capture  $H(Y)$  or at least  $(1 - \epsilon) \times H(Y)$

with less than or equal to  $k$  features, and to find the prediction performances of the classifiers built over these feature vectors, by continuing the search process after the DFL algorithm finds the first satisfactory gene subset.

In addition, the DFL algorithm is a quite general method for learning functional dependencies from data sets. In another work [36], we demonstrated that the DFL algorithm, with minor modification, can be used to find gene regulatory networks from time-series gene expression data.

## Acknowledgements

We thank Li Jinyan of Institute of Infocomm Research, Singapore, for his review on an early version of this paper.

## References

- [1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
- [2] S. A. Armstrong, J. E. Staunton, L. B. Silverman, R. Pieters, M. L. den Boer, M. D. Minden, S. E. Sallan, E. S. Lander, T. R. Golub, and S. J. Korsmeyer. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, 30:41–47, 2002.
- [3] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5:537–550, 1994.
- [4] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.
- [5] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202, 1995.
- [6] S. T. Dumais, J. C. Platt, D. Hecherman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM*, pages 148–155, 1998.
- [7] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI-93*, pages 1022–1027, Chambéry, France, 1993.
- [8] F. Fleuret. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.*, 5:1531–1555, 2004.
- [9] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. H. Witten. Data mining in bioinformatics using Weka. *Bioinformatics*, 20(15):2479–2481, 2004.
- [10] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.
- [11] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286(5439):531–537, 1999.

- [12] M. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, Waikato University, Department of Computer Science, 1999.
- [13] M. A. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, 15:1–16, 2003.
- [14] R. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 9:147–160, 1950.
- [15] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning*, pages 121–129, 1994.
- [16] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
- [17] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning*, pages 284–292, 1996.
- [18] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *National Conference on Artificial Intelligence*, pages 223–228, 1992.
- [19] J. Li, H. Liu, J. R. Downing, A. E.-J. Yeoh, and L. Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients. *Bioinformatics*, 19(1):71–78, 2003.
- [20] J. Li, H. Liu, S.-K. Ng, and L. Wong. Discovery of significant rules for classifying cancer diagnosis data. *Bioinformatics*, 19(90002):93ii–102, 2003.
- [21] J. Li and L. Wong. Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. *Bioinformatics*, 18(5):725–734, 2002.
- [22] H. Liu, J. Li, and L. Wong. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, 13:51–60, 2002.
- [23] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of the 13th International Conference on Machine Learning*, pages 319–327, 1996.
- [24] R. J. McEliece. *The Theory of Information and Coding: A Mathematical Framework for Communication*, volume 3 of *Encyclopedia of Mathematics and Its Applications*. Addison-Wesley Publishing Company, Reading, MA., 1977.
- [25] J.-P. Mira, V. Benard, J. Groffen, L. C. Sanders, and U. G. Knaus. Endogenous, hyperactive Rac3 controls proliferation of breast cancer cells by a p21-activated kinase-dependent pathway. *PNAS*, 97(1):185–189, 2000.
- [26] J. C. Platt. *Advances in kernel methods: support vector learning*, chapter Fast training of support vector machines using sequential minimal optimization, pages 185–208. MIT Press, 1999.
- [27] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.
- [28] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.-H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, T. Poggio, W. Gerald, M. Loda, E. S. Lander, and T. R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *PNAS*, 98(26):15149–15154, 2001.
- [29] C. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, IL., 1963.
- [30] M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutok, R. C. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, T. S. Ray, M. A. Koval, K. W. Last, A. Norton, T. A. Lister, J. Mesirov, D. S. Neuberg, E. S. Lander, J. C. Aster, and T. R. Golub. Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature Medicine*, 8:68–74, 2002.
- [31] L. van ’t Veer, H. Dai, M. van de Vijver, Y. He, A. Hart, M. Mao, H. Peterse, K. van der Kooy, M. Marton, A. Witteveen, G. Schreiber, R. Kerckhoven, C. Roberts, P. Linsley, R. Bernards, and S. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530 – 536, 2002.
- [32] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *ICCV*, pages 281–288, 2003.
- [33] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.
- [34] E. P. Xing, M. I. Jordan, and R. M. Karp. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 601–608. Morgan Kaufmann Publishers Inc., 2001.
- [35] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.
- [36] Y. Zheng and C. K. Kwoh. Dynamic algorithm for inferring qualitative models of gene regulatory networks. In *Proceedings of the 3rd Computational Systems Bioinformatics Conference, CSB 2004*, pages 353–362. IEEE Computer Society Press, 2004.
- [37] Y. Zheng and C. K. Kwoh. Identifying decision lists with the discrete function learning algorithm. In *Proceedings of the 2nd International Conference on Artificial Intelligence in Science And Technology, AISAT 2004*, pages 30–35, 2004.