Gene Selection and Cancer Classification Using a Fuzzy Neural Network

Feng Chu, Wei Xie, and Lipo Wang
School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, Nanyang Avenue, Singapore 639798
elpwang@ntu.edu.sg

Abstract – Cancer classification based on microarray gene expressions is an important problem. In this work, we use a t-test-based feature selection method to choose some important genes from thousands of genes. After that, we classify the microarray data sets with a fuzzy neural network (FNN) that we proposed earlier. This FNN combines important features of initial fuzzy model self-generation, parameter optimization, and rule-base simplification. We applied this FNN to three well-known gene expression data sets, i.e., the lymphoma data set (with 3 sub-types), small round blue cell tumor (SRBCT) data set (with 4 sub-types), and the liver cancer data set (with 2 classes, i.e., non-tumor and hepatocellular carcinoma (HCC)). Our results in all the three data sets show that the FNN can obtain 100% accuracy with a much smaller number of genes in comparison with previously published methods. In view of the smaller number of genes required by the FNN and its high accuracy, we conclude that the FNN classifier not only helps biological researchers differentiate cancers that are difficult to be classified using traditional clinical methods, but also helps biological researchers focus on a small number of important genes to find the relationships between those important genes and the development of cancers.

I. INTRODUCTION

Accurate diagnosis of different types of cancers is of great importance for doctors to choose a proper treatment. However, the similar appearances of some types of cancers are a main challenge for tradition traditional diagnostic methods. In recent years, this problem has attracted great attention in the context of Microarray gene expressions because of their ability to differentiate cancers at molecular level [1]. After the expression profiles of cancer cells are obtained, a variety of machine learning or statistical approaches can undertake the diagnosis task. Some recent approaches include neural networks [2], support vector machines (SVMs) [3], nearest shrunken centroids [4], and so on.

In 2003, Tibshirani et al. [5] successfully classified the lymphoma data set [6] with only 48 genes by using a statistical method called nearest shrunken centroids with an accuracy of 100%. For the SRBCT data set, Khan et al. [2] classified all 20 testing samples with 96 genes. They used a very simple two-layered linear neural network. In 2002, Tibshirani et al. [4] applied nearest shrunken centroids to the SRBCT data set. They obtained 100% accuracy with 43 genes. For the method of nearest shrunken centroids, it categorizes each sample to the class whose centroid is nearest to the sample. The difference between standard nearest centroids and nearest shrunken centroids is that the latter uses only some important genes rather than all the genes to calculate the centroids. In 2003, Deutsch [7] reduced the number of genes required to correctly classify the four cancer sub-types in the SRBCT data set to 12 genes using evolutionary algorithms. In the same year, Lee and Lee [8] also obtained 100% accuracy in this data set with a SVM classifier and the separability based gene importance ranking [9]. They used at least 20 genes to obtain this result. At the same time, they generated 3 principal components (PCs) from the 20 top genes. Their SVM can also obtain 100% accuracy with these 3 principal components. For the liver cancer data set, Chen et al [10] used 3180 genes (represented by 3964 cDNA) to classify HCC and the non-tumor samples.

In this paper, we use a fuzzy neural network (FNN) proposed by Frayman and Wang [11] to classify cancers based on their gene expression profiles.

This paper is organized as follows. In section II, we describe a t-test-based gene importance ranking method. In section III, we review the structure and the algorithm of the FNN proposed by Frayman and Wang [11]. In section IV, we use this FNN to classify three microarray data sets, i.e., the lymphoma data set, the SRBCT data set, and the liver cancer data set. We discuss our results and draw some conclusions in the last section.

II. GENE IMPORTANCE RANKING

We first compute the importance ranking of each gene using a feature ranking measure based on t-test. We then retain only the most important genes for classification in the next step.

The t-score (TS) of gene $i$ is defined as follows [12, 13]:

$$TS_i = \max_k \left\{ \frac{\bar{x}_{ik} \bar{x}_i}{d_k s_i} \right\} = \max_k \left\{ \frac{\bar{x}_{ik}}{d_k} \right\}$$

where

$$\bar{x}_{ik} = \sum_{j \in C_k} x_{ij} / n_k$$

and

$$\bar{x}_i = \sum_{j=1}^n x_{ij} / n$$

The t-score (TS) of gene $i$ is defined as follows [12, 13]:

$$TS_i = \max_k \left\{ \frac{\bar{x}_{ik} \bar{x}_i}{d_k s_i} \right\} = \max_k \left\{ \frac{\bar{x}_{ik}}{d_k} \right\}$$

where

$$\bar{x}_{ik} = \sum_{j \in C_k} x_{ij} / n_k$$

and

$$\bar{x}_i = \sum_{j=1}^n x_{ij} / n$$
\[
\begin{align*}
s_i^2 &= \frac{1}{n-K} \sum_{k} \sum_{j \in C_k} (x_{ij} - \overline{x}_{ik})^2 \\
d_k &= \sqrt{1/n_k + 1/n}
\end{align*}
\]

There are \( K \) classes. \( \max \{ y_k, k = 1, 2, \ldots, K \} \) is the maximum of all \( y_k, k = 1, 2, \ldots, K \). \( C_k \) refers to class \( k \) that includes \( n_k \) samples. \( x_{ij} \) is the expression value of gene \( i \) in sample \( j \). \( \overline{x}_{ik} \) is the mean expression value in class \( k \) for gene \( i \). \( n \) is the total number of samples. \( x_i \) is the general mean expression value for gene \( i \). \( d_k \) is the pooled within-class standard deviation for gene \( i \). Actually, the t-score used here is a t-statistics between a specific class and the overall centroid of all the classes [12].

III. THE FNN

In this section, we review the structure and the algorithm of the FNN proposed by Frayman and Wang [11]. The structure of the FNN is shown in Figure 1. The network consists of four layers, i.e., the input layer, the input membership function (MF) layer, the rule layer, and the output layer [11].

![Fig.1 The structure of the FNN.](image)

In databases, data fields are either numerical or categorical. The input membership function layer generates input membership functions for numerical inputs, i.e., numerical values are converted to categorical values.

Each rule node is connected to each input membership function node and each output node. Each rule node performs a product of its inputs. The input membership functions act as fuzzy weights between the input layer and the rule layer. Links between the rule layer, the output layer, and the input membership functions are adaptive during learning. In the output layer, each node receives inputs from all rule nodes connected to this output node and produces the actual output of the network.

The structure generation and learning algorithm of the FNN are as follows in Figure 2.

![Fig. 2 The flow chart of the FNN algorithm.](image)

A. FNN Initialization

Firstly, we create \( n \) nodes for the input layer and \( m \) nodes for the output layer where \( n \) and \( m \) are the number of the input variables (attributes) and the output variables (classes), respectively. The rule layer is empty, i.e., there are initially no rules in the rule base [11].

Two equally spaced triangular membership functions are added along the operating range of each input variable. In such a way, these membership functions will satisfy \( \varepsilon \)-completeness. Piecewise-linear triangular membership function is chosen for computational efficiency [14].

Then we create the initial rule base layer using the following form for rule \( i \):

Rule \( i \): IF \( x_1 \) is \( A_{1i} \) and \( \ldots \) \( x_n \) is \( A_{ni} \)
Then \( y_1 = \omega_{1i}, \ldots, y_m = \omega_{mi} \),

where \( x_j (j = 1, 2, \ldots, n) \), and \( y_l (l = 1, 2, \ldots, m) \) are the inputs and the outputs, respectively. \( \omega_{ji} \) is a real number. \( A_{qi}^q \) (\( q = x_1, x_2, \ldots, x_n \))
is the membership function of the antecedent part of rule \( i \) for node \( q \) in the input layer [15].

The membership value \( \mu_q \) of the premise of the \( i \)-th rule is calculated as a fuzzy AND, using the product operator

\[
\mu_i = A_{x_1}^i(x_1) \times A_{x_2}^i(x_2) \times \ldots \times A_{x_n}^i(x_n)
\]

The output \( y_l \) of the fuzzy inference is obtained using the weighted average [16].

\[
y_l = \frac{\sum_i \mu_i \times \omega_q^i}{\sum_i \mu_i}
\]

### B. FNN Training

The network is trained using the following general learning rule [17].

\[
y_{i}^{t}(k+1) = y_{i}^{t}(k) - \eta \frac{\partial \epsilon_t}{\partial y_{i}^{t}}
\]

The learning rules for \( \omega_q^i \) and \( A_q^i \) are:

\[
\omega_q^i(k+1) = \omega_q^i(k) - \eta \frac{\partial \epsilon_t}{\partial \omega_q^i}
\]

\[
A_q^i(k+1) = A_q^i(k) - \eta \frac{\partial \epsilon_t}{\partial A_q^i}
\]

where \( \eta \) is the learning rate. The objective is to minimize the error function

\[
\epsilon_t = \frac{1}{2} \times (y_t - y_{dl})^2
\]

where \( y_t \) is the current output, and \( y_{dl} \) is the target output.

We let the learning rate \( \eta \) vary to improve the speed of convergence, as well as the learning performance (accuracy). We update \( \eta \) according to the following two heuristic rules:

1) If the error measure undergoes five consecutive reductions, increase \( \eta \) by 5%.

2) If the error measure undergoes three consecutive combinations of one increase and one reduction, decrease \( \eta \) by 5%.

Furthermore, due to this dynamical update strategy, the initial value of \( \eta \) is usually not critical as long as it is not too large. The learning error \( \epsilon_t \) is reduced towards zero or a pre-specified small value \( \epsilon_{def} > 0 \) as the iteration number \( k \) increases.

### C. Rule Base Modification

An additional membership function is added for each input at the point of the maximum output error, following Higgins and Goodman [14]. One vertex of the additional membership function is placed at the point of the maximum output error, and it must has the membership value unity; the other two vertices lie at the centers of the two neighboring regions, respectively, and they have membership values zero. As the output of the network is not a binary 0 or 1, but a number ranged from 0 to 1, we can speed up the convergence of the network substantially by eliminating the error whose deviation from the target value is the greatest.

We then evaluate the accuracy and simplicity of rules generated above. We use a weighting parameter between accuracy and simplicity, which is the compatibility grade (CG) of each fuzzy rule. CG of rule \( j \) is calculated by the product operator as:

\[
\mu_j(x) = \mu_{j1}(x_1) \times \mu_{j2}(x_2) \times \ldots \times \mu_{jn}(x_n)
\]

when the system provides correct classification result.

Each rule whose CG falls below a pre-defined threshold are deleted. Elimination of rule nodes is rule by rule, i.e., when a rule node is deleted, its associated input membership nodes and links are deleted as well. By varying the CG threshold the user is able to specify the degree of rule base compactness. The size of the rule base can thus be kept minimal. If the classification accuracy of the FNN after the elimination of rule nodes is below the requirement, we will add another rule as described above; otherwise we stop the process.

---

**Fig. 3** The results for the lymphoma data: the upper plot is the training result and the bottom plot is the testing result.

This FNN combines the powerful features of initial fuzzy model self-generation [15], parameter optimisation [17], and rule-base simplification to achieve a better performance.
IV. EXPERIMENTAL RESULTS

A. Lymphoma Data

The lymphoma data set (http://llmpp.nih.gov/lymphoma) [6] contains 42 samples derived from diffuse large B-cell lymphoma (DLBCL), 9 samples from follicular lymphoma (FL), and 11 samples from chronic lymphocytic leukaemia (CLL). The entire data set includes the expression data of 4026 genes. In this data set, a small part of data is missing. A k-nearest neighbor algorithm was applied to fill those missing values [18].

In the first step, we ranked the entire 4026 genes according to their t-scores (TSs) in the training data set. We picked out the 100 genes with the highest TSs. Then we randomly divided the 62 samples into 2 parts, 31 samples for training, and 31 samples for testing.

In this paper, each gene is labelled after its importance rank. For example, gene 5 means the gene ranked 5. Through its ID in the microarray (for example, GENE1622X), the real name of each gene can be found on the web page of the lymphoma data set.

We used the FNN to classify the lymphoma microarray data set. We input the selected 100 genes one by one to the network according to their TS ranks starting with the gene ranked 1. That is, we first used only a single gene that is ranked 1 as the input to the network. We trained the network with the training data, and subsequently tested the network with the test data. We repeated this process with the first 2 genes, then 3 genes, and so on. We found that the FNN performed very well: it can reach 100% accuracy for both the training data and the testing data with only the first 5 genes. The results are shown in Fig. 3.

B. SRBCT Data

The SRBCT data set [2] (http://research.nhgri.nih.gov/microarray/Supplement/) contains the expression data of 2308 genes. There are totally 63 training samples and 25 testing samples provided in [2], 5 of the testing samples are not SRBCTs. The 63 training samples contain 23 Ewing family of tumors (EWS), 20 rhabdomyosarcoma (RMS), 12 neuroblastoma (NB), and 8 Burkitt lymphomas (BL). And the 20 SRBCT testing samples contain 6 EWS, 5 RMS, 6 NB, and 3 BL. In the first step, we ranked the entire 2308 genes according to their TSs [12, 13] in the training data. Then we picked out the 30 genes with the highest t-scores for classification.

We used the expression data of the gene ranked 1 to train and then test the FNN. We repeated this process with the top 2 genes, then top 3 genes, and so on. The testing error and the training error both decreased to 0 when the top 8 genes were input into the FNN. The results are shown in Fig. 4.

C. Liver Cancer Data

Fig.4 The results for the SRBCT data: the upper plot is the training result and the bottom plot is the testing result.

Fig.5 The results for the liver cancer data: the upper plot is the training result and the bottom plot is the testing result.

We used the expression data of the gene ranked 1 to train and then test the FNN. We repeated this process with the top 2 genes, then top 3 genes, and so on. The testing error and the training error both decreased to 0 when the top 8 genes were input into the FNN. The results are shown in Fig. 4.
The liver cancer data set [10] (http://genome-www.stanford.edu/hcc/) has two classes, i.e. the non-tumor liver and HCC. The data set contains 156 samples and the expression profiles of 1648 important genes. Among them, 82 are HCCs and the other 74 are non-tumor livers. In this data set, there are some missing values. We also used the k-nearest neighbor method to fill those missing values [18].

Similarly, we input the 105 genes with the highest TSs into the FNN one by one. The testing error and the training error both decreased to 0 when the top 24 genes were input into the FNN. The results are shown in Fig. 4.

IV. DISCUSSION

Our result shows that the microarray data classification problem can be solved with a much smaller number of genes. Compared to the method of nearest shrunken centroids using 48 genes, the FNN leads to 100% accuracy using only 5 genes for the lymphoma data set.

For the SRBCT data, the best known result is from the evolutionary algorithm reported by Deutsch [2]. He used 12 genes to obtain 100% accuracy. However, the FNN requires only 8 genes to obtain the same accuracy.

For the liver cancer data, Chen et al. [10] used 3180 genes (represented by 3964 cDNA) to classify HCC and the non-tumor samples. In comparison with Chen et al.’s work [10], the FNN also greatly reduced the number of genes (to 24) required to obtain an accurate result.

In view of the smaller number of genes required by the FNN and its high accuracy, we conclude that the FNN classifier not only helps biological researchers differentiate cancers that are difficult to be classified using traditional clinical methods, but also helps researchers focus on a small number of important genes to find the relationship between those important genes and the development of cancers.