Classification of Micro-array Gene Expression Data using Neural Networks

David Tian and Keith Burley

Abstract— **Classification of yeast genes based on their expression levels obtained from micro array hybridization experiments is an important and challenging application domain in data mining and knowledge discovery. Over the past decade, neural networks and support vector machines (SVMs) have achieved good results for genes classification. This paper presents a methodology which uses two neural networks to classify unseen genes based on their expression levels. In order to remove some of the noise and deal with the imbalanced class distribution of the dataset, data pre-processing is firstly performed before data classification in which data cleaning, data transformation and data over-sampling using SMOTE algorithm are undertaken. Thereafter, two neural networks with different architectures are trained using Scaled Conjugate Gradient in two different ways: 1) the training-validation-testing approach and 2) 10-fold cross-validation. Experimental results show that this methodology outperforms the previous best-performing SVM for this problem and 8 other classifiers: 3 SVMs, C4.5, Bayesian network, Naive Bayes, K-NN and JRip.**

I. Introduction

The expression levels of genes can be measured through DNA micro array experiments [17]. Very often, genes of the same functional classes possess very similar expression patterns in micro array hybridization experiments. Therefore, the expression levels of genes can be used to classify them into different functional classes.

Firstly, thousands of DNA samples are attached to a glass slide to construct a micro array [17]. Each sequence corresponds to a single gene within the organism under investigation. Then, messenger RNA samples are collected from a population of cells subjected to various experimental conditions. Afterwards, the RNA samples are converted to cDNA via reverse transcription and are labelled with one of two different fluorescent dyes in the process. Therefore, a single experiment consists of hybridizing the micro array with two differently labelled cDNA samples collected at different times. Generally, one of the samples is taken from the reference or background state of the cell, whereas the other sample represents a special condition set up by the experimenter, for instance, heat shock. The level of expression of a particular gene is approximately proportional to the amount of cDNA that hybridizes with the DNA affixed to the slide. The relative levels of gene expression for any pair of conditions can be measured by measuring the ratio of each of the two dyes present at the position of each DNA sequence on the slide using laser scanning technology. The result is a series of n expression-level ratios from an experiment with n DNA samples on a single chip. The numerator of each ratio is the expression level of the gene in the condition of interest to the experimenter, while the denominator is the expression level of the gene in the reference state of the cell. Each data point produced by a DNA micro array hybridization experiment represents the ratio of expression levels of a particular gene under two different experimental conditions.

The authors are with the Department of Computing, Faculty of Arts, Computing, Engineering and Science, Sheffield Hallam University, City Campus, Howard Street, Sheffield S1 1WB UK (phone: +44 225 5555; email: d.tian@shu.ac.uk; k.burley@shu.ac.uk).

TABLE I
CLASS DISTRIBUTION

	TCA	Resp	Ribo	Prot	Hist	Non-fun
class distribution	14	27	121	35	11	2256
	1:2:11:3:1:205					

Over recent years, different classification approaches have been applied to micro array gene expression data classification [5], [11], [12], [13], [14], [15], [16]. Previous work often shows that neural networks and SVMs outperformed the other approaches used [2], [3], [4], [6], [7], [8], [9].

Some gene classification problems are very challenging in that the class distributions are very imbalanced; the datasets contain missing values and noise [17]. Brown et al. [17] applied SVMs to a dataset consisting of the expressions of 2467 genes from budding yeast S. cerevisiae measured in 79 different DNA micro array hybridization experiments. The dataset contains expression levels of 208 genes that belong to 5 functional classes (TCA, Resp, Ribo, Prot and Hist) and 2256 genes that do not belong to a functional class. The class distribution is very imbalanced (table I). The majority of the instances do not belong to any of the 5 functional classes. Each of the functional classes contains very few genes. This paper proposes a novel methodology to tackle this dataset. The methodology consists of a data pre-processing phase and two neural networks. Section II describes the dataset. The methodology is presented in section III. Section IV presents the training of two multi-layer perceptron neural networks and experimentation results. Performance comparison of the neural networks with other classification approaches are discussed in section V. Conclusions and future work are presented in section VI.

II. Dataset

The original data was obtained from a series of m hybridization experiments and represented as a $n \times m$ matrix with each of the n rows consisting of an m-element expression vector corresponding to a particular gene [17].

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

79 experiments were performed and 2467 genes were used. Hence, the dataset contains 2467 instances and 79 features. The instances are labelled with five functional classes: Tricarboxylic acid (TCA), Respiration (Resp), Ribosome (Rib), Proteasome (Pro) and Histone (Hist) [17]. The functional classes definition are taken from MYGD [17]. The tricarboxylic-acid pathway is also known as the Krebs cycle. Genes in this pathway encode enzymes that break down pyruvate (produced from glucose) by oxidation. This is a key catabolic pathway to produce energy for the cell initial in the form of NADH and is also important for producing intermediates in the biosynthesis of amino acids and other compounds. The respiration chain complexes perform oxidation-reduction reactions that capture the energy present in NADH through electron transport and the chemiosmotic synthesis of ATP. These include the NADH deydrogenase complex, cytochrome b-c complex, and cytochrome oxidase complex, all embedded in the mitochondrial membrane. The cytoplasmic ribosomal proteins are a class of proteins required to make the ribosome, an RNA-protein complex in the cytoplasm encoded by mRNA. This category does not include genes for the mitochondrial ribosome. The proteasome consists of proteins comprising a complex responsible for general degradation of proteins and other specific protein processing events. The proteasome uses ubiquitin, a small peptide that marks a protein to be degraded. Histones interact with the DNA backbone to form nucleosomes which, with other proteins, form the chromatin of the cell.

III. Methodology

The original dataset is firstly log-transformed as follows. Let X denote a gene. For each experiment, the logarithm of the ratio of $X's$ expression level in that experiment to $X's$ expression level in the reference state, is calculated. This log ratio is positive if the gene is induced according to the background and negative if it is repressed. The log-transformed dataset is then cleaned and transformed again using zero-mean normalization. The cleaned and transformed dataset is named D. The class distribution of D is very imbalanced (table I). The majority of the genes do not belong to any of the five functional classes. These genes are grouped into the sixth class: 'non-functional' (non-fun) class. The genes of the 5 functional classes are merged into a new class 'functional class' to give a binary labelled dataset $D1$. The instances of the 5 functional classes are also extracted from D to give a dataset $D2$ labelled with 5 functional classes. The ratio of functional to non-functional classes is very imbalanced: 1:11 (208:2256). Synthetic minority over-sampling technique (SMOTE) is applied to $D1$ to balance the class distribution by generating samples of the functional class. The resulting dataset is $D1'$. A multi-layer perceptron (MLP) mlp1 is trained using $D1'$. Another MLP mlp2 is trained using $D2$ (Figure 1). To classify unseen genes, mlp1 is used to recognize the genes of the functional class. Then, these genes are input to mlp2 which classifies them into 5 functional classes (Figure 2). Data pre-processing consists of data cleaning, transformation, extraction and SMOTE.

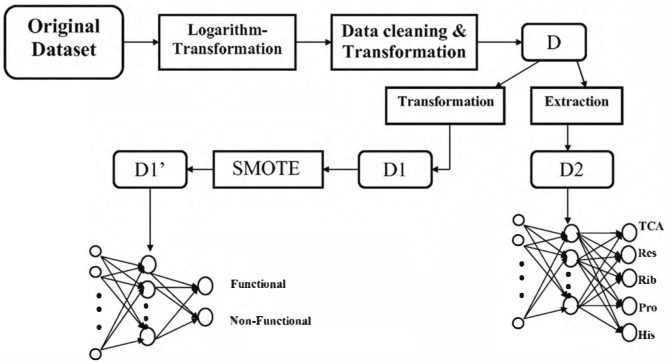

Fig. 1. Methodology of Yeast genes classification: neural networks training

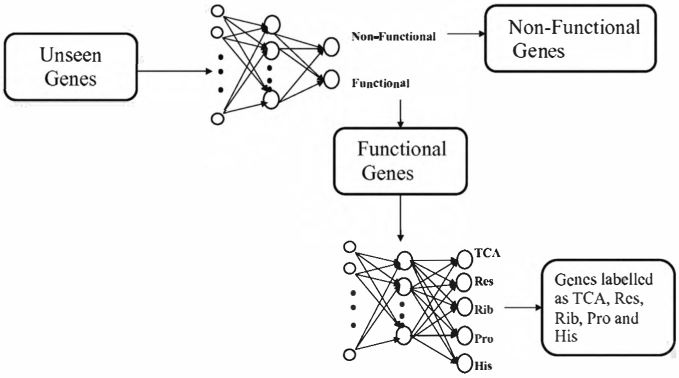

Fig. 2. Methodology of Yeast genes classification: classifying unseen genes

A. Data Cleaning and Transformation

Screening of the dataset reveals that 3 genes are labelled as both TCA and Resp. These 3 genes are considered as noise and removed from the dataset. Most of the instances i.e. genes are nearly complete. Less than 25% of the genes are complete (Fig 3). Just over 30% of the genes have 1 missing value. Just over 30% of the genes have 2 missing values. Less than 13% of the genes have 3 missing values. 5% of the genes have 4 missing values. Less than 2% of the genes have between 5 and 8 missing values. The number of missing values of each attribute is also analysed (Fig 4). Most attributes have less than 3% ($2464 \times 3\% \approx 73$) missing values. The missing values are replaced with the means of the corresponding attributes. Then, zero-mean normalization is performed on the cleaned dataset.

B. Data Transformation and Extraction for Neural Networks Training

Two MLPs are to be used to classify unseen genes. The first MLP mlp1 classifies a gene as either functional class or non-functional class. If the gene is classified as functional, the second MLP mlp2 will further classify it as one of the five functional classes. Therefore, mlp1 and mlp2 are trained using 2 different datasets: $D1$ and $D2$. $D1$ contains the genes which are labelled as functional and non-functional classes. $D2$ contains only those genes which are labelled as the 5 functional classes. $D1$ is produced by grouping together

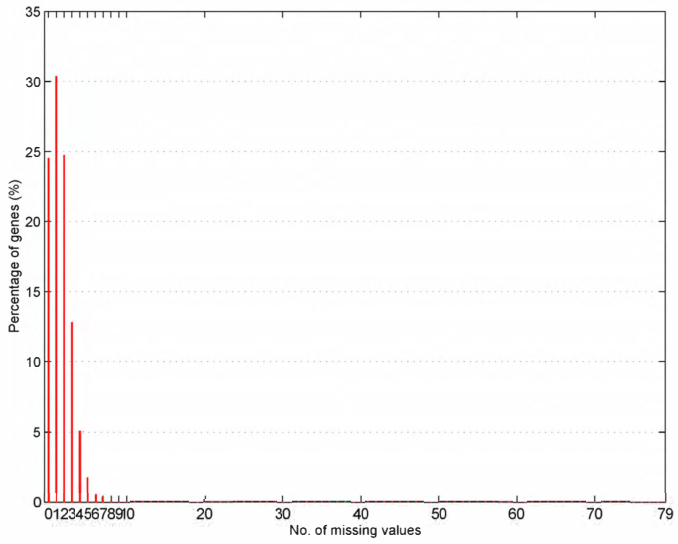

Fig. 3. Percentages of genes with different number of missing values

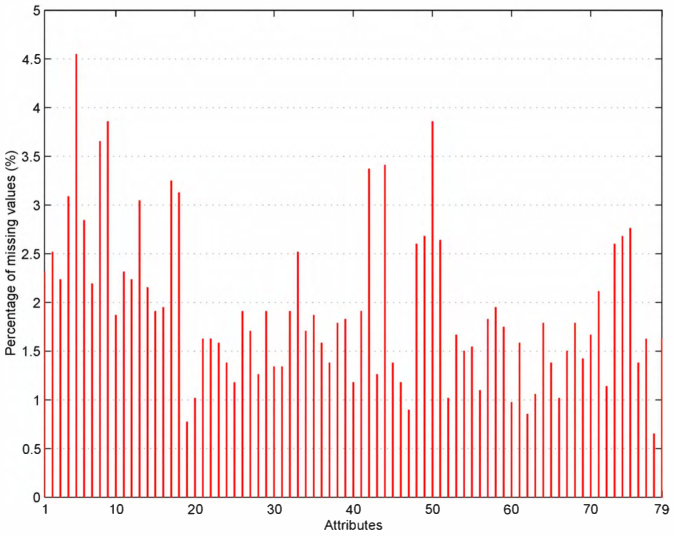

Fig. 4. Percentage of missing values of each attribute

the genes of the 5 functional classes. $D2$ is produced by extracting the instances of the 5 classes from the normalized dataset. The ratio of functional and non-functional classes is very imbalanced: 1:11 (208:2256). Synthetic Minority Over-sampling Technique (SMOTE) is applied to $D1$ to balance the class distribution by generating samples of functional classes. The resulting dataset is $D1'$.

C. Synthetic Minority Over-sampling Technique

One well-know approach to improve classification performance of classifiers learnt using imbalanced datasets is SMOTE [18]. SMOTE creates artificial data based on the feature space similarities between existing minority examples. Specifically, for subset $S_{min} \in S$ where S is a dataset; S_{min} is a minority class of S, consider the K-nearest neighbours for each example $x_i \in S_{min}$, for some specified integer K; the K-nearest neighbours are defined as the K elements of S_{min} whose Euclidian distance between itself and x_i under consideration exhibits the smallest magnitude along the n-dimensions of feature space X. To create a synthetic sample, randomly select one of the K-nearest neighbours, then multiply the corresponding feature vector difference with a random number between [0, 1], and finally, add this vector to x_i

$$x_{new} = x_i + (\hat{x}_i - x_i) \times \delta \tag{1}$$

where $x_i \in S_{min}$ is the minority instance under consideration, $\hat{x}_i$ is one of the K-nearest neighbours for x_i: $\hat{x}_i \in S_{min}$, and $\delta \in [0, 1]$ is a random number. Therefore, the resulting synthetic instance according to (1) is a point along the line segment joining x_i under consideration and the randomly selected K-nearest neighbour $\hat{x}_i$. SMOTE algorithm pseudocode is illustrated in Fig 5. SMOTE is applied recursively to $D1$ to generate synthetic samples until the resulting dataset $D1'$ is balanced (Fig 6).

D. Multi-layer Perceptrons and Scaled Conjugate Gradient

Multi-layer perceptrons model the probabilities of class membership: $p(C_k|\mathbf{x})$ where C_k is k^{th} class and $\mathbf{x}$ is a pattern. For gene expression data classification, two multi-layer perceptrons: mlp1 and mlp2 are used. mlp1 has architecture $79 \times M \times 2$ with a hidden layer of M units and 2 logistic sigmoid output units (2). One of the output units returns $p(\text{'functional class'}|\mathbf{x})$. The other returns $p(\text{'non-functional class'}|\mathbf{x})$. mlp2 has architecture $79 \times N \times 5$ with a hidden layer of N units and 5 softmax output units (3). Both mlps are trained using the Scaled Conjugate Gradient algorithm [1] and the cross entropy error function (4) where y^n is an output and t^n is a target. The cross entropy function is used because it is likely to perform better than sum-of-squares at estimating small probabilities.

$$g(a) = \frac{1}{1 + exp(-a)} \tag{2}$$

$$g(a_k) = \frac{exp(a_k)}{\sum_{k'} exp(a_{k'})} \tag{3}$$

$$E = -\sum_n \{t^n + \ln y^n + (1 - t^n) \ln(1 - y^n)\} \tag{4}$$

Regularization is a common approach for reducing overfitting of networks to training data. Regularization involves adding a penalty Ω to the error function to encourage smoother network mappings (5). Then, the network is trained according to $\widetilde{E}$.

$$\widetilde{E} = E + V\Omega \tag{5}$$

The parameter V controls the extent to which the penalty term Ω called a *regularizor* influences the form of the solution. Training is performed by minimizing the total error function $\widetilde{E}$ which requires the derivatives of Ω. A network that provides a good fit to the training data will have a small value for E, while a very smooth network will have a small value for Ω. Training a MLP is to compromise between

Algorithm: SMOTE(S_{min}[][],N,K)
S_{min}[][]: array holding the original minority class samples
T: the number of minority class samples
N: N% of minority class samples to generate
K: number of nearest neighbours
numattrs = Number of attributes
Synthetic[][]: array for synthetic samples
nnarray[]: array holding indices of nearest neighbours
1. if $N < 100$
2. then randomize the T minority class samples
3. $T = (N/100) * T$
4. $N = 100$
5. endif
6. $N = (int)(N/100)$
7. for $i \leftarrow 1$ to T
8. compute K nearest neighbours for i and save the indices in nnarray[]
14. Populate(N,i,S_{min}[][],nnarray[],Synthetic[][])
15. endfor
16. return Synthetic[][]

Populate(N,i,S_{min}[][],nnarray[],Synthetic[][])
1. newindex: a count of number of synthetic
2. samples generated, initialized to 0
3. while $N \neq 0$
4. Choose a random number between 1 and K,
5. call it nn. This step chooses one
6. of the K nearest neighbours of i.
7. for attr $\leftarrow$ 1 to numattrs
8. Compute:
9. dif = S_{min}[nnarray[nn]][attr] - S_{min}[i][attr]
10. Compute:
11. gap = random number between 0 and 1
12. Synthetic[newindex][attr]=S_{min}[i][attr]+gap*dif
13. endfor
14. newindex++
15. $N = N - 1$
16. endwhile
17. return Synthetic[][]

Fig. 5. Pseudocode of SMOTE

Procedure call_SMOTE(S_{min},S_{maj})
S_{min}: minority class samples, S_{maj}: majority class samples
$T_{min} = |S_{min}|$; T_{maj}=$|S_{maj}|$;
while($T_{maj}/T_{min} \geq 2$){
$S_{new} \leftarrow SMOTE(S_{min}, 100, 5)$;
$S_{min} \leftarrow S_{new} \cup S_{min}$;
}
$T_{new} = (T_{maj} - T_{min})/T_{min} \times 100$;
$S_{new} \leftarrow SMOTE(S_{min}, T_{new}, 5)$;
$S = S_{new} \cup S_{min}$;
return S

Fig. 6. Recursively applying SMOTE to a minority class

fitting the training data and minimizing Ω. One widely-used regularizer is *weight decay* (6) which encourages small weights. It has been shown that weight decay often leads to significant improvements in network generalization.

$$\Omega = \frac{1}{2}\sum_i \omega_i^2 \tag{6}$$

The backpropagation algorithm adjusts the weights in the steepest descent direction (negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. Scaled conjugate gradient algorithm [1] is as follows:

1) Choose weight vector w_1 and scalars $\sigma > 0$, $\lambda_1 > 0$ and $\overline{\lambda}_1 = 0$. Set $p_1 = r_1 = -E'(w_1)$, $k = 1$ and success = true.
2) If success = true then calculate second order information:

$$\sigma_k = \frac{\sigma}{|p_k|}, \tag{7}$$

$$s_k = \frac{E'(w_k + \sigma_k p_k) - E'(w_k)}{\sigma_k}, \tag{8}$$

$$\delta_k = p_k^T s_k. \tag{9}$$

3) Scale s_k:

$$s_k = s_k + (\lambda_k + \overline{\lambda}_k)p_k, \tag{10}$$

$$\delta_k = \delta_k + (\lambda_k - \overline{\lambda}_k)|p_k|^2. \tag{11}$$

4) If $\delta_k \leq 0$ then make the Hessian matrix positive definite:

$$s_k = s_k + (\lambda_k - 2(\frac{\lambda_k}{|p_k|^2})p_k, \tag{12}$$

$$\overline{\lambda}_k = 2(\lambda_k - \frac{\delta_k}{|p_k|^2}), \tag{13}$$

$$\delta_k = -\delta_k + \lambda_k|p_k|^2, \lambda_k = \overline{\lambda}_k. \tag{14}$$

5) Calculate step size:

$$\mu_k = p_k^T r_k,\ \alpha_k = \frac{\mu_k}{\delta_k}. \tag{15}$$

6) Calculate the comparison parameter:

$$\Delta_k = \frac{2\delta_k[E(w_k) - E(w_k + \alpha_k p_k)]}{\mu_k^2}.$$

7) If $\Delta_k \geq 0$ then a successful reduction in error can be made:

$$w_{k+1} = w_k + \alpha_k p_k\ , r_{k+1} = -E'(w_{k+1}), \overline{\lambda}_k = 0,$$

success = true.

a) If k mod $N = 0$ then restart algorithm: $p_{k+1} = r_{k+1}$

TABLE II

PERFORMANCE OF MLP1 WITH DIFFERENT ARCHITECTURES

Nodes	Class	Precision	Recall	Accuracy
14	Fun	1.0	0.98	98.67
	Non-fun	0.98	1.0	
13	Fun	0.99	0.97	98.3
	Non-fun	0.97	0.99	
12	Fun	1.0	0.98	98.76
	Non-fun	0.98	1.0	
10	Fun	0.99	0.98	98.67
	Non-fun	0.98	0.99	
9	Fun	0.99	0.98	98.6
	Non-fun	0.98	0.99	
8	Fun	0.98	0.97	97.78
	Non-fun	0.97	0.98	
7	Fun	0.99	0.96	97.6
	Non-fun	0.96	0.99	

else create new conjugate direction:

$$\beta_k = \frac{|r_{k+1}|^2 - r_{k+1} r_k}{\mu_k}, \quad (16)$$

$$p_{k+1} = r_{k+1} = \beta_k p_k. \quad (17)$$

b) If $\Delta_k \geq 0.75$ then reduce the scale parameter: $\lambda_k = \frac{1}{2}\lambda_k$, else a reduction in error is not possible: $\overline{\lambda}_k = \lambda_k$, success = false.

8) If $\Delta_k \leq 0.25$ then increase the scale parameter: $\lambda_k = 4\lambda_k$.

9) If the steepest descent direction $r_k \neq 0$ then set $k = k + 1$ and go to 2, else terminate and return w_{k+1} as the desired minimum.

$E(w_k)$ is an error function. p_k is a search direction. α_k is the step size. The parameter σ determines the change in the weight for the second derivative approximation. The parameter λ regulates the indefiniteness of the Hessian matrix.

IV. NEURAL NETWORKS TRAINING

mlp1 consists of 79 input units, a single hidden layer and 2 logistic outputs. $D1'$ is split into 3 datasets: a training set (50%), a validation set (25%) and a testing set (25%). Networks with different number of hidden layer nodes are trained using the training dataset and evaluated using the validation dataset (Table II). The classification performances of the networks are compared. The network with 10 nodes has the highest classification accuracy and is selected, then evaluated using the testing dataset. Precision (18) and recall (19) are used to measure the classification performance of mlp1 for each class. Classification accuracy is used to measure the classification performance of mlp1 for the dataset.

$$Precision = \frac{TP}{TP + FP} \quad (18)$$

$$Recall = \frac{TP}{TP + FN} \quad (19)$$

In order to alleviate overfitting and to obtain a smooth learned function, regularization is used to penalize large weights. The network with 10 hidden nodes is trained with different values of V and is evaluated using the validation dataset (Table III). Adding regularization leads to little improvement in classification accuracy. The learnt function without regularization is already very smooth. Making the function too smooth decreases the generalization performance of the network. The network with 10 nodes and $V = 1$ has the highest accuracy (99%) and is evaluated on the testing dataset. The classification accuracy is 98.8% (Table IV). This network will be used to classify unseen genes in future problems.

TABLE III

PERFORMANCE OF MLP1 WITH DIFFERENT REGULARIZATION

V	Class	Precision	Recall	Accuracy
10	Fun	0.99	0.98	97.9
	Non-fun	0.98	0.99	
9	Fun	0.99	0.98	98.5
	Non-fun	0.98	0.99	
8	Fun	0.99	0.97	98.3
	Non-fun	0.97	0.99	
7	Fun	0.99	0.98	98.5
	Non-fun	0.98	0.99	
6	Fun	0.99	0.98	98.6
	Non-fun	0.98	0.99	
5	Fun	0.99	0.98	98.8
	non-fun	0.98	0.99	
4	Fun	0.99	0.98	98.9
	non-fun	0.98	0.99	
3	Fun	0.99	0.98	98.5
	Non-fun	0.98	0.99	
2	Fun	0.99	0.98	98.5
	Non-fun	0.98	0.99	
1	fun	1.0	0.98	99.0
	Non-fun	0.98	1.0	

TABLE IV

PERFORMANCE OF MLP1 ON TESTING DATASET

Class	Precision	Recall	Accuracy
Fun	0.99	0.98	98.8
Non-fun	0.98	0.99	

mlp2 consists of 79 input units, a hidden layer and 5 softmax outputs. A 10-fold cross-validation is performed using $D2$ for each different number of hidden layer nodes (Table V). The network with 9 hidden layer nodes has the highest validation accuracy (96.67%). Regularization is added to the network with 9 hidden layer nodes. Then, a 10-fold CV is performed for each different value of V (Table VI). The network with $V = 4.5$ has the highest validation accuracy (97.62%) and is selected for classifying unseen genes in future problems.

V. COMPARISON WITH OTHER APPROACHES

Three SVMs: one with a RBF, one with a Polynomial and another with a Pearson VII universal function kernel, C4.5, Bayesian network, Naive Bayes, K-nearest neighbour (K-NN) and JRip are applied to datasets $D1'$ and $D2$ respectively. For $D1'$ the training-validation-testing method is used. The SVM with Pearson VII universal function kernel (SVM_Pub) achieved the highest accuracy among all the approaches used (Table VII). mlp1 has a slightly lower

TABLE V

10-FOLD CV PERFORMANCE OF MLP2 WITH DIFFERENT HIDDEN LAYER NODES

Nodes	Class	Precision	Recall	Accuracy
12	TCA	0.57	1.0	96.14
	Resp	0.96	0.84	
	Ribo	1.0	0.99	
	Pro	0.97	0.94	
	Hist	1.0	1.0	
11	TCA	0.57	1.0	96.17
	Resp	0.96	0.84	
	Ribo	1.0	0.99	
	Pro	0.97	0.94	
	Hist	1.0	1.0	
10	TCA	0.71	0.83	96.17
	Resp	0.93	0.86	
	Ribo	1.00	1.00	
	Pro	0.97	0.94	
	Hist	0.91	1.00	
9	TCA	0.71	0.91	96.67
	Resp	0.93	0.83	
	Ribo	1.0	1.0	
	Pro	0.97	0.97	
	Hist	1.00	1.00	
8	TCA	0.71	0.91	96.64
	Resp	0.93	0.86	
	Ribo	1.0	1.0	
	Pro	0.97	0.97	
	Hist	1.0	0.92	
7	TCA	0.71	0.83	96.17
	Resp	0.93	0.86	
	Ribo	1.0	0.99	
	Pro	0.97	0.97	
	Hist	0.91	1.0	

TABLE VI

10-FOLD CV PERFORMANCE OF MLP2 WITH DIFFERENT REGULARIZATION

V	Class	Precision	Recall	Accuracy
10	TCA	0.71	0.91	96.2
	Resp	0.93	0.89	
	Ribo	1.0	0.99	
	Pro	0.97	0.94	
	Hist	1.0	1.0	
6	TCA	0.71	1.0	97.12
	Resp	0.96	0.9	
	Ribo	1.0	1.0	
	Pro	0.97	0.99	
	Hist	1.0	1.0	
5.5	TCA	0.71	1.0	97.1
	Resp	0.96	0.93	
	Ribo	1.0	0.99	
	Pro	0.97	0.92	
	Hist	1.0	1.0	
5	TCA	0.79	0.92	97.12
	Resp	0.93	0.93	
	Ribo	1.0	1.0	
	Pro	0.97	0.92	
	Hist	1.00	1.00	
4.5	TCA	0.86	0.92	**97.62**
	Resp	0.93	0.89	
	Ribo	1.0	1.0	
	Pro	0.97	0.97	
	Hist	1.0	1.0	
4	TCA	0.79	0.92	97.14
	Resp	0.93	0.86	
	Ribo	1.0	1.0	
	Pro	0.97	0.97	
	Hist	1.0	1.0	
3	TCA	0.71	0.91	96.64
	Resp	0.93	0.86	
	Ribo	1.0	1.0	
	Pro	0.97	0.94	
	Hist	1.0	1.0	

performance compared with SVM_Pub and outperforms all the other approaches. There is an unique optimal solution in training a SVM. Training a SVM always finds that optimal solution. In contrast, there are multiple local optimal solutions for neural networks training. Training a neural network often arrives at a local optimal solution. In addition, SVMs are less prone to overfitting than neural networks. Pearson VII universal kernel (20) where x_i and x_j are 2 vectors, is a generic function which is capable of approximating RBF, Polynomial and linear function kernels to arbitrary accuracy by selecting appropriate values for the parameters ω and σ.

$$K(x_i, x_j) = \frac{1}{\left[1 + \left(\frac{2\sqrt{\|x_i - x_j\|^2}\sqrt{2^{(1/\omega)}-1}}{\sigma}\right)^2\right]^{\omega}} \quad (20)$$

For each of the two classes, precision and recall are very similar. This is because the class distribution of $D1'$ is balanced and the classifiers's class prediction is unbiased.

For $D2$, 10-fold CV is used. The previous work [17] applied 3-fold cross-validation and 3 SVMs: one with a RBF kernel, one with a linear kernel and another with a polynomial kernel to this dataset. Among all these classifiers, the SVM with a RBF kernel is the best-performing approach [17]. This work indicates that mlp2 outperformed the best-performing SVM of the previous work and all the other classifiers used (Table VIII).

For each of the classifiers, the classification performances of the classes are different. Rib is the majority class. For most of the classifiers, the precision and the recall of Rib are the highest among all the 5 classes. TCA is a minority class. For most of the classifiers, the precision and the recall of TCA are the lowest among all the 5 classes. Hist is the smallest minority class, the precision and recall of Hist are slightly higher than those of TCA. This is due to gene expressions of TCA contain more noise than those of TCA which lowers the classification performance of TCA. For each class, precision and recall are different because the class distribution is imbalanced and consequently, pattern classification is biased ($FP \neq FN$).

C4.5 and JRip both perform binary discretization on each of the continuous attributes in order to select the next variable in the classification model. The discretization process removes some of the important information for patterns classification. Consequently, the classification accuracy of the learnt decision tree and the induced decision rules are affected. Training of a SVM involves using a mapping function[1] $\phi : R^d \rightarrow H$ to map the original data to a very high dimensional Euclidean space; thereafter, a hyperplane which maximizes the margin, is computed in that high dimensional

[1] $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ where $K(x_i, x_j)$ is a kernel function.

TABLE VII

PERFORMANCE COMPARISON FOR DATASET $D1'$: RESULTS ON TESTING DATASET

Approach	Class	Precision	Recall	Testing Accuracy
Mlp1	Fun	0.99	0.98	98.8
	Non-fun	0.98	0.99	
SVM(RBF)	Fun	0.92	0.87	89.5
	Non-fun	0.87	0.92	
SVM(Poly)	Fun	0.96	0.96	95.8
	Non-fun	0.96	0.96	
SVM(Puk)	Fun	0.99	1.0	99.3
$\omega = 1, \sigma = 1$	Non-fun	1.0	0.99	
C4.5	Fun	0.92	0.94	93
	Non-fun	0.94	0.92	
BN	Fun	0.98	0.89	93.5
	Non-fun	0.89	0.99	
NB	Fun	0.92	0.87	89.8
	Non-fun	0.87	0.92	
K-NN	Fun	0.91	1.0	94.9
	Non-fun	1.0	0.91	
JRip	Fun	0.92	0.95	93.4
	non-fun	0.95	0.92	

space and is used to separate or shatter the instances of one class from those of another class. Therefore, projection functions critically affect the performance of SVMs in that the functions determine both the high dimensional space and the locations of the instances in that space.

Bayesian networks (BN) and naive bayes (NB) represent the conditional dependence and independence relationships between the attributes of a dataset. The root node of a BN represents the class attribute. Each other node of a BN represents an attribute of the dataset. Associated with each node is a conditional probability table which specifies the conditional distribution $p(x_i|Parents(x_i))$ for the attribute x_i given its parents $Parents(x_i)$ in the BN. Probabilities of class memberships $p(Class = C_k|\mathbf{x})$s where $\mathbf{x}$ is a pattern, are computed based on the conditional distributions of the attributes. Therefore, conditional probabilities of memberships of classes critically affect the accuracy of BNs. For continuous attributes, their probability distributions are modelled as Gaussian distributions. However, for some attributes, their distributions are non-Gaussian. This reduces the accuracies of BNs. In addition, if the structure of BNs are wrong, $p(Class = C_k|\mathbf{x})$s are wrong and the accuracies of the BNs are poor. NB assumes that all the attributes of the dataset are conditionally independent and the distribution of each continuous attribute is a Gaussian. For some attributes, these assumptions are not completely correct. Therefore, this affects the performance of NB. Neural networks are universal function approximators. They can approximate nonlinear functions to arbitrary accuracy. They are not based on the assumption that continuous attributes have Gaussian distributions.

VI. CONCLUSION

The proposed methodology is promising for heavily-imbalanced gene classification problems because it divides the classification problem into 2 simpler classification problems and then applies a neural network to each problem.

TABLE VIII

PERFORMANCE COMPARISON ON DATASET $D2$ USING 10-FOLD CV: 3-FOLD SVM IS THE BEST-PERFORMING SVM OF THE PREVIOUS WORK

Approach	Class	Precision	Recall	Accuracy
Mlp2	TCA	0.86	0.92	97.62
	Resp	0.93	0.89	
	Ribo	1.0	1.0	
	Pro	0.97	0.97	
	Hist	1.0	1.0	
3-fold-SVM	TCA	0.71	0.59	N/A
	Resp	0.76	0.87	
	Ribo	0.95	0.99	
	Pro	0.88	0.86	
	Hist	1.0	0.82	
SVM(Poly)	TCA	0.83	0.71	95.7
	Resp	0.86	0.93	
	Ribo	1	1	
	Pro	0.92	0.97	
	Hist	1	0.82	
SVM(Puk)	TCA	1	0.5	95.2
$\omega = 4\sigma = 15$	Resp	0.77	1	
	Ribo	1	1	
	Pro	0.94	0.97	
	Hist	1	0.82	
SVM(RBF)	TCA	1	0.5	95.2
	Resp	0.77	1	
	Ribo	1	1	
	Pro	0.94	0.97	
	Hist	1	0.82	
C4.5	TCA	0.67	0.57	92.8
	Resp	0.8	0.85	
	Ribo	1	1	
	Pro	0.87	0.94	
	Hist	1	0.73	
BN	TCA	0.85	0.79	95.7
	Resp	0.86	0.93	
	Ribo	1	1	
	Pro	0.92	0.94	
	Hist	1	0.82	
NB	TCA	0.67	0.86	95.2
	Resp	0.89	0.86	
	Ribo	1	1	
	Pro	0.97	0.94	
	Hist	1	0.82	
5-NN	TCA	0.8	0.57	94.2
	Resp	0.89	0.85	
	Ribo	1	1	
	Pro	0.83	1	
	Hist	1	0.82	
JRip	TCA	0.6	0.64	89.4
	Resp	0.86	0.7	
	Ribo	0.93	0.98	
	Pro	0.88	0.86	
	Hist	1	0.82	

SMOTE plays a crucial role in unbiasing the classification of genes of different classes. Even though SVMs often have better generalization performance than neural networks. The latter output probabilities of memberships of classes which can be used for further data analysis. Rejection threshold can be used to reject the classification of genes when their probabilities of memberships of classes are too small to be certain about their classes. In some applications e.g. medical screen where it is far more serious to mis-classify a tumour image as normal than to mis-classify a normal image as that of a tunour. In this case, different mis-classifications carry different penalties and the overall loss is minimized. The

probabilities of memberships of classes output by neural networks can be combined with a loss matrix to allow the minimum risk decision to be made. The probabailities of memberships of classes can also be used to estimate the probability distribution of the dataset using Baye's rule together with the prior probabilities of the attributes of the dataset. SVMs are discriminant functions outputting labels of classes. Neural networks can be more appropriate than SVMs when classification rejection is necessary. The proposed methodology will be evaluated further using imbalanced datasets from other domains e.g. fraud detection, network intrusion.

REFERENCES

[1] M. Møller, *"A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning"*, Neural Networks, 1993, vol. 6, no. 4, pp. 525-533.

[2] Z. Zainuddin and O Pauline, *"Improved Wavelet Neural Networks for Early Diagnosis of Cancer Patients Using Micro array Gene Expression Data"*, IEEE International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19 2009, pp. 3485-3492.

[3] F. Chu and L. Wang, *"Applying RBF Neural Networks to Cancer Classification Based on Gene Expressions"*, IEEE International Joint Conference on Neural Networks, July 16-21, 2006, pp. 1930-1934.

[4] J. C. Patra, Q. Zhen, E. L. Ang and A. Das, *"Neural Networks for Gene Expression Analysis and Gene Selection from DNA Micro array"*, IEEE International Joint Conference on Neural Networks, July 31 - August 4, 2005, pp. 509-514.

[5] A. Islam, K, M. Iftekharuddin and E. O. George, *"Class Specific Gene Expression Estimation and Classification in Micro array Data"*, IEEE International Joint Conference on Neural Networks, 2008, pp. 1678-1685.

[6] A Kelemen and Y. Liang, *"Bayesian Regularized Neural Network for Multiple Gene Expression Pattern Classification"*, IEEE International Joint Conference on Neural Networks, 2003, pp. 654-659.

[7] Y. Liang, E. O. George and A. Kelemen, *"Baysian Neural Network for Micoarray Data"*, IEEE International Joint Conference on Neural Networks, 2002, pp. 193-197.

[8] Z. Boger, *"Artificial Neural Networks Methods for Identification of the Most Relevant Genes from Gene Expression Array Data"*, IEEE International Joint Conference on Neural Networks, 2003, pp. 3095-3100.

[9] F. Chu and L. Wang, *"Gene Expression Data Analysis Using Support Vector Machines"*, IEEE International Joint Conference on Neural Networks, 2003, pp.2268-2271.

[10] V. P. Plagianakos, D. K. Tasoulis and M. N. Vrahatis, *"Computational Intelligence Techniques for Acute Leukemia Gene Expression Data Classification"*, IEEE International Joint Conference on Neural Networks, Montreal, Canada, july 31 - August 4, 2005, pp. 2469-2474.

[11] C. Zheng, P. Zhang, L. Zhang, X. Liu and J. Han, *"Gene Expression Data Classification Based on Non-negative Matrix Factorization"*, Altanta, Georgia, USA, June 14 - 19, 2009, pp. 3542-3547.

[12] O. Okun and H. Priisalu, *"Ensembles of K-Nearest Neighbours and Dimensionality Reduction"*, IEEE International Joint Conference on Neural Networks, 2008, pp. 2032-2039.

[13] C. yang, L. Chuang, J. Li and C. Yang, *"A Novel BPSO Approach for Gene Selection and Classification of Micoarray Data"*, IEEE International Joint Conference on Neural Networks, 2008, pp. 2147-2152.

[14] E. Pranckeviciene and R. Somorjai, *"On Classification Models of Gene Expression Micoarrays: The Simpler the Better"*, IEEE International Joint Conference on Neural Networks, Vancouver, BC, Canada, July 16-21, 2006, pp. 3572-3579.

[15] O. Okun and g. Valentini, *"Dataset Complexity Can Help to Generate Accurate Ensembles of K-Nearest Neighbours"*, IEEE International Joint Conference on Neural Networks, 2008, pp. 450-457.

[16] A. Kelemen, H. Zhou, P. Lawhead and Y. Liang, *"Naive Bayesian Classifier for Micro array Data"*, IEEE International Joint Conference on Neural Networks, 2003, pp. 1769-1773.

[17] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, M. Ares, Jr. and D. Haussler, *"Support Vector Machine Classification of Micro array Gene Expression Data"*, UCSC-CRL-99-09, June 12, 1999.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, *"SMOTE: Synthetic Minority Over-sampling Technique"*, Journal of Artificial Intelligence Research, 2002, vol 16, pp. 321-357.