Chapter 7

Majority Voting Genetic Programming Classifier

7.1 Introduction

In the previous chapter, we have applied genetic programming to analyze microarray gene expression data and found that it suffers from over-fitting. The potential challenge for genetic programming is that it has to search two large spaces of functions and genes simultaneously to find an optimal solution. In most cases, the evolved single rules or sets of rules produce very poor test accuracies. To overcome this limitation of genetic programming, we propose a majority voting technique for the prediction of the class of a test sample. We call this method *majority voting genetic programming classifier* (MVGPC). The motivation behind this is that a group of rules can be more accurate than the best member of the group (Kuncheva and Whitaker, 2003). In its typical implementation, we evolve multiple rules in different GP runs, apply them one by one to a test sample and count their votes in favor of a particular class. Then the sample is assigned to the class that gets the highest number of votes in favor of it. However, the success of majority voting depends on the number of rules in a voting group. Here we investigate the number of rules in a majority voting group that produces the best results.

In this chapter, we apply MVGPC to four microarray data sets, including two multi-category data sets, to demonstrate the effectiveness of our proposed method. We perform different kinds of experiments on the data sets and analyze the evolved rules.

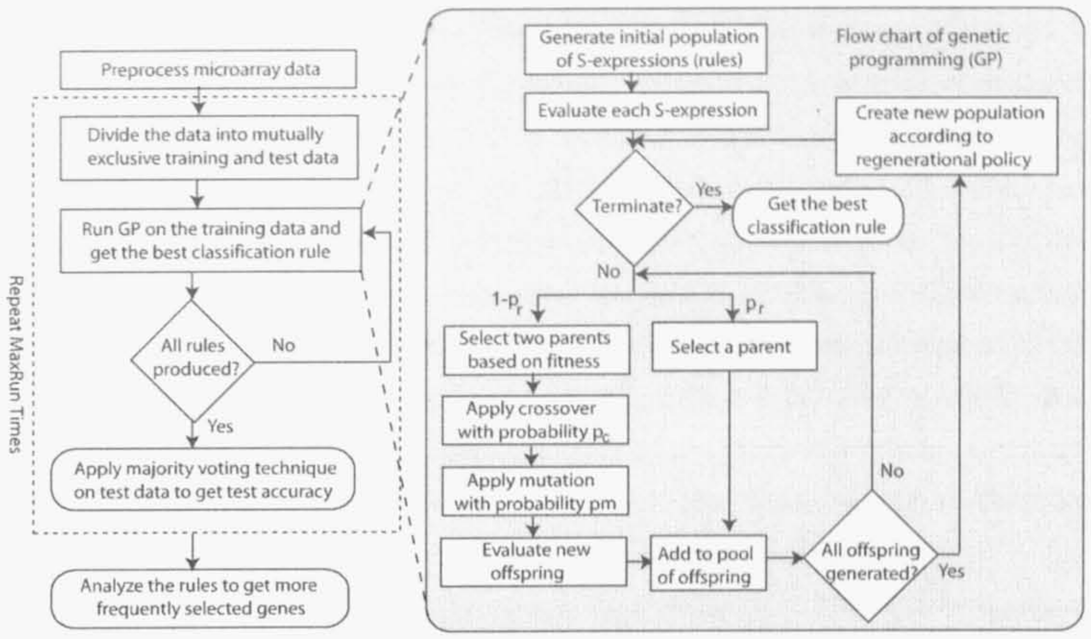

Preprocess microarray data
Divide the data into mutually exclusive training and test data
Run GP on the training data and get the best classification rule
All rules produced?
No
Yes
Apply majority voting technique on test data to get test accuracy
Analyze the rules to get more frequently selected genes
Repeat MaxRun Times
Generate initial population of S-expressions (rules)
Flow chart of genetic programming (GP)
Evaluate each S-expression
Create new population according to regenerational policy
Terminate?
Yes
Get the best classification rule
No
1-p_r
p_r
Select two parents based on fitness
Select a parent
Apply crossover with probability p_c
Apply mutation with probability pm
Evaluate new offspring
Add to pool of offspring
All offspring generated?
No
Yes

Figure 7.1: Steps in classification of gene expression data with MVGPC

7.2 Class prediction through majority voting

In this section, we describe our majority voting genetic programming classifier in detailed. The steps required for classification of gene expression data are shown in Fig. 7.1.

7.2.1 Majority voting technique

Overfitting is a major concern in classification of gene expression data using machine learning techniques. Since the number of available training samples is very small compared to huge number of genes and the number of samples per class is not evenly distributed, a single rule or a single set of rules produces very poor test accuracy. Due to the smaller number of training samples, most machine learning techniques use leave-one-out-cross-validation (LOOCV) technique (Kohavi, 1995) to calculate the training accuracy. In LOOCV, one sample from the training set is excluded, and the rest of the training samples is used to build the classifier. Then, the classifier is used to predict the class of the left out one, and this is repeated for each sample in the training set. The LOOCV estimate of accuracy is the overall number of correct classifications, divided by the number of samples in the training set. Thereafter, the final classifier is built using all the training samples, and the classes of the test samples are predicted one by one using that classifier. Note here that the data corresponding to the selected genes remain the same during learning of the classifier using the LOOCV technique. Conversely, in genetic programming, different rules may evolve in different iterations of the LOOCV technique, and therefore we cannot calculate the LOOCV accuracy of a particular rule. Instead, in most cases, the training accuracy of a rule is calculated by executing it on the whole training data in one pass. However, the single rules or sets of rules evolved in this way produce very poor test accuracy.

Instead of a single rule or set of rules, we can produce multiple rules in multiple GP runs and employ them to predict the labels of the test samples through majority voting. This majority voting technique is described here through examples. Suppose we want to predict the binary labels (A or B) of test samples through the votes of v single rules. For this task, we run GP v times to get v best rules. If the output of the S-expression of a rule is positive on a test sample Y, the vote in favor of class A is increased by one; otherwise, the vote in favor of class B is increased by one. Then the label of Y is predicted to be the class that has the higher number of votes.

However, for multi-category samples, the majority voting technique is applied in a different way. If there are c types of samples in the microarray data set, we generate a total of $v * c$ rules in $v * c$ GP runs—v rules for each type of samples. During evolution of a rule for class i, we consider all the samples of type i as positive samples, and the remaining samples as negative samples. Thus, each rule acts as a binary classifier. If the output of the S-expression of a rule for class i has positive output on a test sample Y, the positive vote in favor of class i is increased by one; otherwise, the negative vote against class i is increased by one. Then the class of the test sample Y is predicted as follows:

$$Class(Y) = \max_i \{r_1, r_2, \ldots, r_c\} \tag{7.2.1}$$

where r_i is the ratio of positive and negative votes for class i. If the number of negative votes of class i is zero, r_i is set to v. The test sample gets the label of the the class that has the highest ratio of positive and negative votes. If two or more ratios are the same, the class is determined by randomly picking one class from the classes corresponding to the ratios. If all ratios are zero, the test sample is treated as misclassified. Let us give an example (see Table 7.1). Suppose that there are four classes (A, B, C, D) of samples in a microarray data set, and the number of rules per voting group (v) is 5. If the number of positive and negative votes for the classes are $\{0, 3, 4, 4\}$ and $\{5, 2, 1, 1\}$, respectively, the predicted label of the test sample will be either C or D (should be randomly chosen).

7.2.2 Dependency of MVGPC on the performance of single rules

The performance of MVGPC is very dependent on the performance of single rules. When all the rules in the ensemble of MVGPC is very poor, MVGPC too performs poorly. Here we derive the probability that MVGPC is no better than single rules for binary classification.

Suppose that the number of samples in the test subset is m and the number of rules per voting group in MVGPC is v (v is an odd number). Let p be the probability that a single rule makes a false prediction. (For simplicity, we have assumed the false prediction rate by each rule is same).

For binary classification using majority voting, a test sample is misclassified if more than $\lfloor \frac{v}{2} \rfloor$ rules make false predictions. Therefore, the probability of a test sample being misclassified by majority voting is

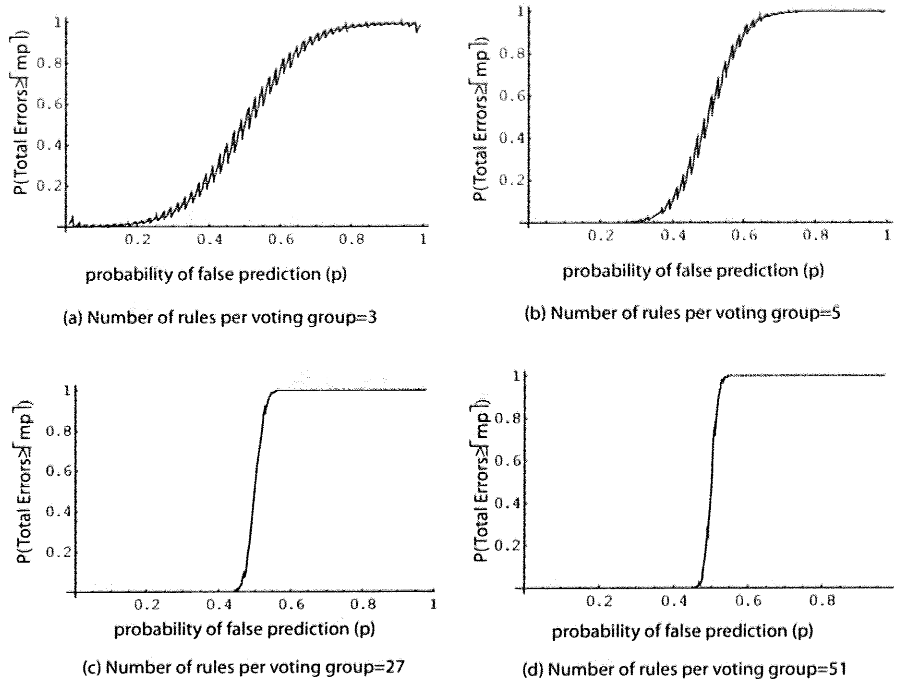
$$e = \sum_{i=\lceil \frac{v}{2} \rceil}^{v} {}_vC_i p^i (1-p)^{v-i} \ . \tag{7.2.2}$$

If $p < 0.5$, the probability of misclassification by majority voting will be less than p , i.e., majority voting will be better than a single rule.

For m test samples, the expected number of false predictions by a single rule will be $\lceil mp \rceil$. So, the probability that MVGPC will make false predictions equal

Table 7.1: Votes of different rules in the example of MVGPC

Category	***S-expression of rule***	***Output: S(Y)***	***+votes***	***-vote***
	$S_1^A = X_{34}/X_{39} - X_{55} * \sqrt{X_{45}}$	-56		√
	$S_2^A = X_{100} + X_{345} - X_{67}^2/\sqrt{X_5}$	-40		√
A	$S_3^A = X_{99} + X_{57} - X_{98}$	-100		√
	$S_4^A = X_{32}/(\sqrt{X_{65}} - X_{89})$	-89		√
	$S_5^A = -X_{90} * (X_{320}/X_{1032})$	-34		√
	$S_1^B = X_{91} - (X_{87}/\sqrt{X_{66}})$	-45		√
	$S_2^B = X_{103} + X_{567} + X_{11}^2$	456	√	
B	$S_3^B = X_{543}/(X_{87} + \sqrt{X_{23}})$	253	√	
	$S_4^B = (X_{32} - X_{95})/(X_{98} - \sqrt{X_{65}})$	-78		√
	$S_5^B = X_{768} * (X_{34}/(X_{12}/\sqrt{X_{67}}))$	87	√	
	$S_1^C = X_{768} * (X_{45} + X_{65})$	123	√	
	$S_2^C = (X_{99}/X_{54})/(X_{32} + \sqrt{X_{57}})$	16	√	
C	$S_3^C = -X_{543}/(X_{56} * (X_{43} - X_{234}))$	-86		√
	$S_4^C = X_{768} * \sqrt{(X_{76} + X_{99})}$	165	√	
	$S_5^C = X_{32}^2/\sqrt{(X_{34} + X_{87})}$	17	√	
	$S_1^D = X_{45}^2 * \sqrt{X_{99}}$	456	√	
	$S_2^D = \sqrt{X_{543}} * (X_{67} - X_{768} - X_{11})$	-249		√
D	$S_3^D = (X_{85} + X_{99}) * (X_{44} + X_{103})$	789	√	
	$S_4^D = X_{65} * X_{43}/\sqrt{X_{131}}$	357	√	
	$S_5^D = X_{32}^2 * \sqrt{X_{567}}$	521	√	



(a) Number of rules per voting group=3

(b) Number of rules per voting group=5

(c) Number of rules per voting group=27

(d) Number of rules per voting group=51

Figure 7.2: Probability that MVGPC will be worse than a single rule

to $\lceil mp \rceil$ or more, i.e., the probability that MVGPC is not better than a single rule is

$$P(\text{Total Errors} \geq \lceil mp \rceil) = \sum_{j=\lceil mp \rceil}^{m} {}_mC_j e^j (1-e)^{m-j} \tag{7.2.3}$$

For a particular v and m, there is a cut off value for p upto which MVGPC will be better than single rules, i.e., $P(\text{Total Errors} \geq \lceil mp \rceil)$ will be zero. As we increase v, MVGPC performs better for a larger cut off value of p but that value is less than 0.5. Therefore, when $p < 0.5$, the test accuracy of MVGPC with sufficient number of diverse rules in the voting group will be better than that of a single rule.

In Fig. 7.2, we have shown the graphical plots of the probability under $m = 51$ and different values of v. As we increase v, the cut off value for p increases. Interestingly, the performance of MVGPC with 27 and 51 rules per voting group are almost identical.

7.2.3 Optimal number of rules for MVGPC

The success of majority voting depends on the number of members in a voting group (ensemble size). Ensemble containing very smaller number of genetic programming rules may be not strong enough to predict cancer classes with higher accuracy; similarly, ensemble containing many rules may not be an optimal choice because some rules may be redundant and may either not contribute to higher accuracy or affect negatively. From the empirical studies, we have found that for a data set containing smaller number of training samples, the best results are obtained when the number of members in a voting group is equal to the number of training samples. However, for a data set containing many training samples, generation of all the rules may not be feasible; smaller number of rules may be sufficient. Our hypothesis about optimal number of rules is as follows:

1. Binary classification:

   - If the number of training samples is smaller:

   $$\text{Number of rules in the ensemble=Number of training samples.} \tag{7.2.4}$$

   - If the number of training samples is higher:

   $$\text{Number of rules in the ensemble=} \begin{cases} \lceil N/2 \rceil & \text{If } \lceil N/2 \rceil \text{ is odd;} \\ \lceil N/2 \rceil + 1 & \text{If } \lceil N/2 \rceil \text{ is even} \end{cases} \tag{7.2.5}$$

   where N is the number of training samples.

2. Multi-class classification:

   - If the number of training samples is smaller:

   $$\text{Number of rules per class=Number of training samples.} \tag{7.2.6}$$

   In this case, total number of rules in the ensemble is cN where c and N are the number of classes and the number of training samples.

- If the number of training samples is higher, the total number of rules in the ensemble should be approximately equal to the number of training samples.

$$\text{Number of rules per class} = \begin{cases} \lceil N/c \rceil & \text{If } \lceil N/c \rceil \text{ is odd;} \\ \lceil N/c \rceil + 1 & \text{If } \lceil N/c \rceil \text{ is even} \end{cases} \qquad (7.2.7)$$

where N and c are as above.

7.2.4 Majority voting with LOOCV rules

We have already said that we cannot calculate the LOOCV accuracy of a particular rule. However, MVGPC can be applied for the prediction of the label of a test sample using the rules of LOOCV in the following way:

- Generate N(=number of training samples) rules in N GP runs.
- In each run i, leave sample i for validation and use the remaining $(N-1)$ samples as training data. If the evolved best rule can correctly classify the left out one, add this best rule to the voting group.
- Apply majority voting on the test data using the members of the voting group.

Note here that the number of members in a voting group may be smaller than N.

7.2.5 Difference between AdaBoost and MVGPC

Boosting is a general method for improving the accuracy of any given learning algorithm (Schapire, 1999). The most widely known boosting algorithm is AdaBoost (Freund and Schapire, 1997). AdaBoost calls a given weak or base classifier repeatedly in a series of rounds and maintains a distribution or set of weights over the training set. Initially, all weights are set equally, but on each round, the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set. The final classifier is a weighted majority vote of the weak classifiers. The AdaBoost algorithm is given below.

Let $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ be the sample i where x_{ik} is the gene expression levels of gene k in it and the label of this sample be $y_i \in \{0, 1\}$. The training set S is a collection of N labeled samples, i.e., $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$. Let $p_t^{(i)}$ represents the probability that the training sample i is included in the training set TR_t at iteration t; note here that $TR_t \subseteq S$. Let C be a genetic programming classifier that returns a rule $h_t : \mathbf{x} \rightarrow \{0, 1\}$ using the samples TR_t and T be the maximum number of allowable iterations in AdaBoost. The steps in AdaBoost are as follows:

1. Set $t = 1$.
   Assign equal probability to each sample, i.e., $p_t^{(i)} = \frac{1}{N}$ for $i = 1, 2, \ldots, N$.

2. Pick N training samples with replacement using the probability distribution $\mathbf{p}_t$ to form TR_t.

3. Apply C to the samples $TR(t)$ to get a rule $h_t : X \rightarrow \{0, 1\}$.

4. Calculate error of h_t: $\varepsilon_t = \sum_{i=1}^{N} p_t^{(i)} |y_i - h_t(\mathbf{x}_i)|$.

5. If $\varepsilon_t > 0.5$, backtrack to previous iteration (set $t = t - 1$).

6. Calculate confidence level: $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$.

7. Update probability distribution:

$$p_{t+1}^{(i)} = \begin{cases} \frac{p_t^{(i)} \exp(-\alpha_t)}{Z_t} & \text{if } y_i = h_t(\mathbf{x}_i); \\ \frac{p_t^{(i)} \exp(\alpha_t)}{Z_t} & \text{if } y_i \neq h_t(\mathbf{x}_i) \end{cases}$$

   where Z_t is a normalization factor so that $\sum_{i=1}^{N} p_{t+1}^{(i)} = 1$.

8. t=t+1. If $t < T$, go to **Step** 2.

9. Predict the label of a sample $\mathbf{x}$ as follows:

$$h_f(\mathbf{x}) = \begin{cases} 1 & \text{if } \alpha_+ > \alpha_-; \\ 0 & \text{otherwise} \end{cases}$$

   where $\alpha_+ = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ and $\alpha_- = \sum_{t=1}^{T} \alpha_t (1 - h_t(\mathbf{x}))$.

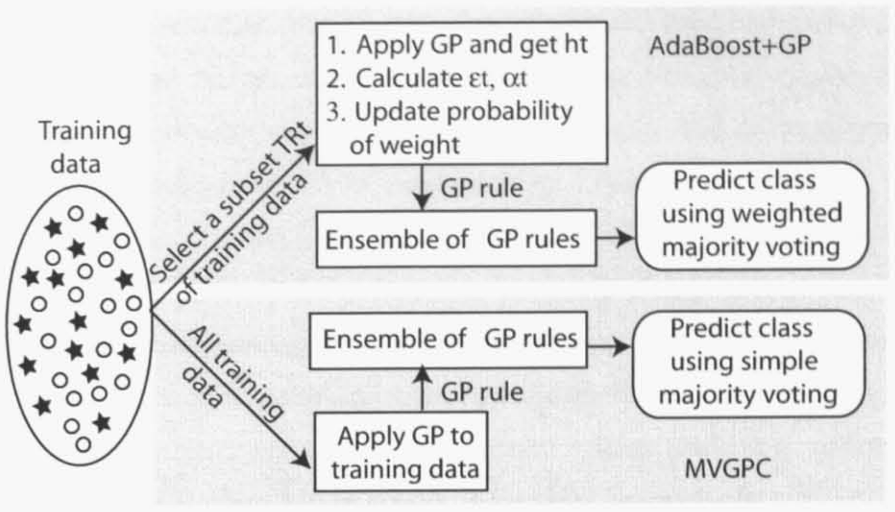

Training data
Select a subset TRt of training data
All training data
1. Apply GP and get ht
2. Calculate εt, αt
3. Update probability of weight
GP rule
Ensemble of GP rules
Predict class using weighted majority voting
AdaBoost+GP
Ensemble of GP rules
GP rule
Apply GP to training data
Predict class using simple majority voting
MVGPC

Figure 7.3: Class prediction by AdaBoost+GP and MVGPC

In our method, we treat all the training samples equally and evolve rules using all the training samples in each GP run. That is, in each GP run, we are trying to evolve a rule that will perfectly classify all the training samples whereas in each run of AdaBoost, we trying to evolve a rule that will perfectly classify a subset of training samples. That is why, we hypothesize that MVGPC would be a much stronger classifier than AdaBoost. Since all rules of the majority voting are evolved using all the training samples, their votes in prediction of the test labels are equally weighted ($\alpha_t = 1$). The difference between MVGPC and AdaBoost+GP has been shown by using Fig. 7.3.

7.3 Evaluation of MVGPC

To evaluate the performance of MVGPC, we used four microarray data sets; the data sets include brain cancer (Nutt *et al.*, 2003), prostate cancer (Singh *et al.*, 2002), breast cancer (Hedenfalk *et al.*, 2001), and lung carcinoma (Bhattacharjee *et al.*, 2001) data sets. *Unlike the fixed split of data in all experiments in Chaps. 5 and 6, the processed data set was randomly divided into two mutually exclusive training and test subsets in each experiment of MVGPC.* This is done to show that MVGPC is not biased towards any fixed split of data.

For larger data sets like the prostate cancer and the lung carcinoma data sets, the ratio of training to test size was approximately 1:1. For smaller data sets like the brain cancer and the breast cancer data sets, the ratio of training to test size was approximately 2:1 as suggested by Dudoit *et al.* (2002). However, during random split of a data set, precaution was taken so that the desired ratio was maintained for each class of samples in the training and test subsets. If precaution is not taken, samples of some classes may be absent in either training or test subset, which in turns may cause larger over-fitting. The training and test sizes of the brain cancer, prostate cancer, breast cancer and lung carcinoma data sets were (34, 16), (51, 51), (16, 6), and (103, 100), respectively.

We used the same settings of different parameters as in Table 6.1 (Chap. 6) except the maximum number of generations per run, and the fitness function. the maximum number of generations per run was 50, while the fitness of a rule was evaluated using equation (6.2.4). To describe a gene in texts, we have also used the same notation as of previous chapter (Chap. 6).

7.3.1 Test accuracies on the data sets

We performed different types of experiments on the four microarray data sets using different number of rules (v) per voting group. Each GP run in these experiments used a different random seed to create the initial population, i.e., each run was independent. Since the minimum number of members in a voting group should be 3 to make a decision, we performed 20 experiments on each data set with $v = 3$. Then for binary classification problems (brain cancer and prostate cancer data sets), we performed experiments with $v = 5$, and for multi-class classification problems (breast and lung carcinoma data sets), we performed experiments with $v = 3c$, where c is the number of classes in the data set. Finally, we performed experiments on the brain cancer, prostate cancer, breast cancer and lung carcinoma data sets with v=17, 27, 16, and 21, respectively as determined according to the formulas in Subsection 7.2.3.

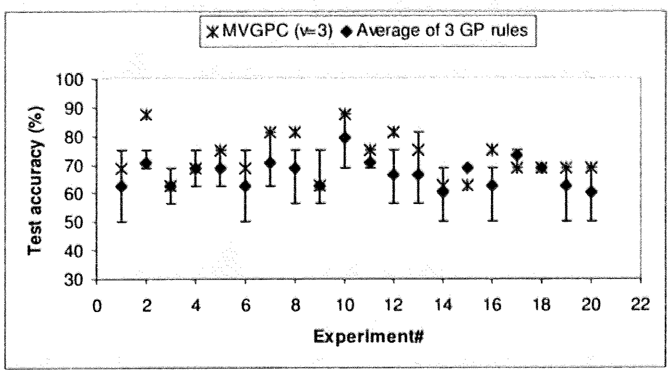
The experimental results on different data sets are presented in Figs. 7.4–7.7.

In each figure, the majority voting accuracy is presented along with the average accuracy of v single rules or sets of rules in the ensemble; the maximum and the minimum accuracies obtained by those rules or sets of rules are indicated by the corresponding error bars.

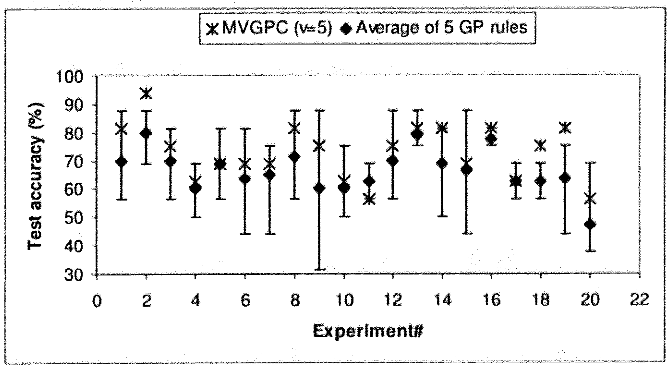
The summary of different experiments on the brain cancer data set with different number of rules per voting group is presented in Fig. 7.4. From the figure, we find that the majority voting accuracy is better than the corresponding average accuracy in most cases. However, the best average test accuracy by MVGPC was obtained with $v = 17$. In that case, out of 20 experiments, MVGPC obtained the highest 87.50% test accuracy in only 5 experiments; in the remaining experiments, it obtained either 81.25% or 75.0% test accuracy on the data set. However, neither MVGPC nor any single rule could classify all the test samples perfectly. Out of 340 (=17*20) rules, though there were many perfect training rules that could classify all the training samples correctly, only two of them could classify 15 out of 16 test samples correctly.

Though the prostate cancer data set is a binary classification problem like the brain cancer data set, the numbers of training and the test samples are greater than those in the brain cancer data set; thus making it a more difficult binary classification problem than the brain cancer data set. On this data set, we performed experiments with v=3, 5, and 27. Here the best test accuracies were obtained when the number of rules in a voting group was equal 27. In that case, the highest test accuracy obtained by majority voting was 94.12%. Out of 540 (=27*20) rules, we found 141 rules that individually could classify all the training samples correctly; however, only one of these rules could classify 49 test samples out of 51 correctly.

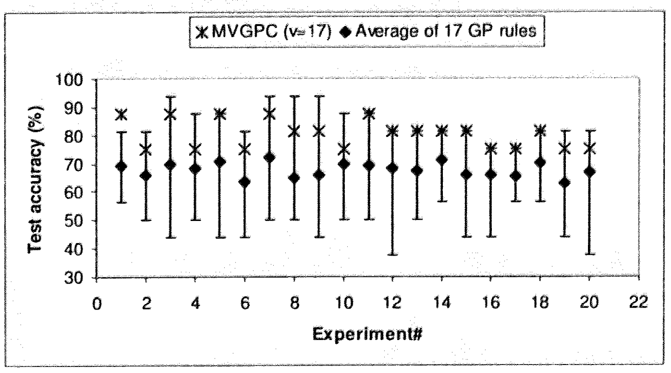
The breast cancer data set contains three classes of samples (BRAC1, BRAC2 and sporadic) and is the smallest among the data sets considered in this paper with only six test samples. On this data set, we performed three sets of 20 experiments with v=3, 9, and 16. The results are shown in Fig. 7.6. In all cases, the majority voting accuracy is much better than the average accuracy of the v sets of 3 rules. The best test accuracies were obtained when the number of rules per class was



(a) Number of voting rules=3

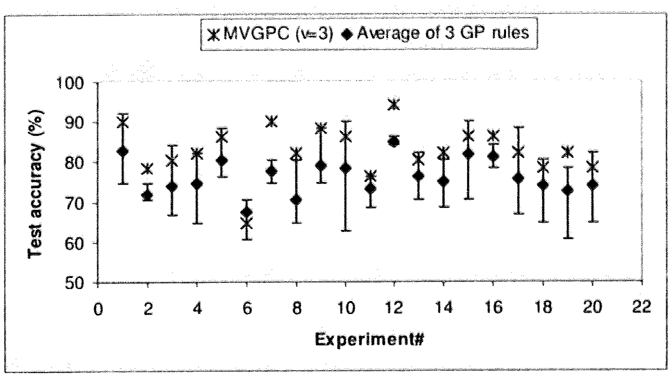


(b) Number of voting rules=5

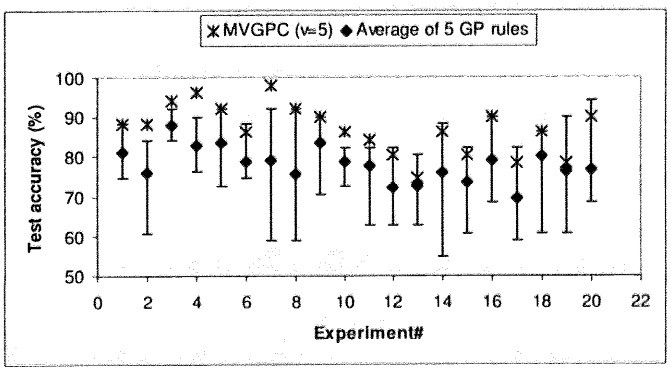


(c) Number of voting rules=17

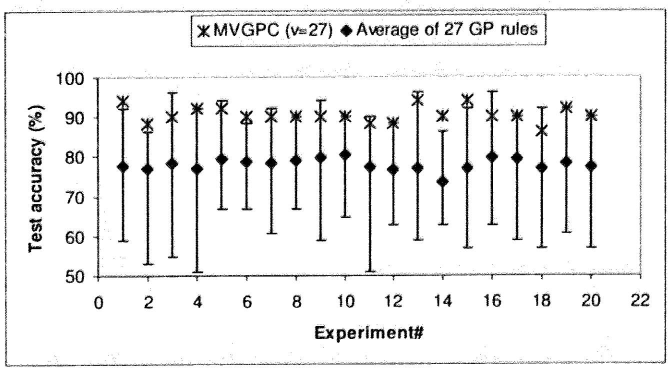
Figure 7.4: Test accuracies on brain cancer data under different conditions. For each experiment, in addition to the average accuracy, the maximum and the minimum accuracies are plotted on the graphs using error bars



(a) Number of voting rules=3



(b) Number of voting rules=5



(c) Number of voting rules=27

Figure 7.5: Test accuracies on prostate cancer data under different conditions. For each experiment, in addition to the average accuracy, the maximum and the minimum accuracies are plotted on the graphs using error bars

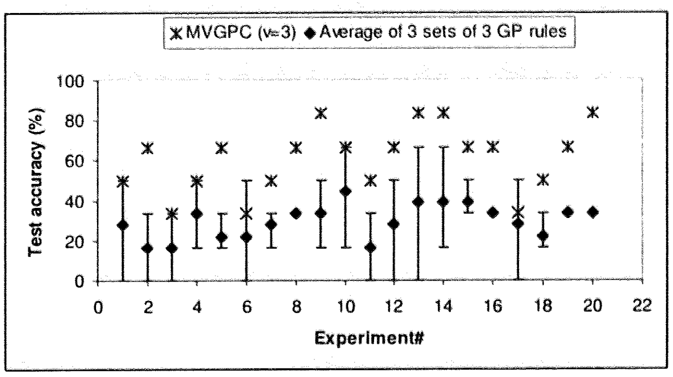
equal to the number of training samples, i.e., the number of rules per ensemble was 48 (=3*16). Out of 20 experiments, we got 100% test accuracy in 5 cases and in the remaining cases, the test accuracy was 83.33%, 66.66% or 50%. Interestingly, either or both of the two sporadic samples in the test subset were misclassified in the 15 experiments where majority voting test accuracy was below 100%. However, as individual sets of rules, we did not find any set of rules that could classify all the test samples 100% accurately.

The lung carcinoma data set contains five classes of samples, and has 103 training samples and 100 test samples making it a more difficult problem than the other three data sets. Since the minimum number of members in a voting group should be 3, we performed experiments with $v = 3$. Then we performed experiments with $v = 15$, which is the equal to the product of the number of types of samples in the data set and the minimum number of required voting members. Finally, we performed experiments with $v = 21$; in this case, the total number of voting rules per experiment is 105 (=21*5), which is approximately equal to the number of training samples in the data set. In Fig. 7.7, we have presented the experimental results on lung carcinoma data. In each experiment, the majority voting accuracy was much better than the average accuracy; it was even better than the maximum accuracy of v sets of rules. Of the three types of experiments, the best average test accuracies were obtained with $v = 21$. In this case, the average, the maximum and the minimum test accuracies of majority voting were 95.50%, 99.0% and 94.0%, respectively.

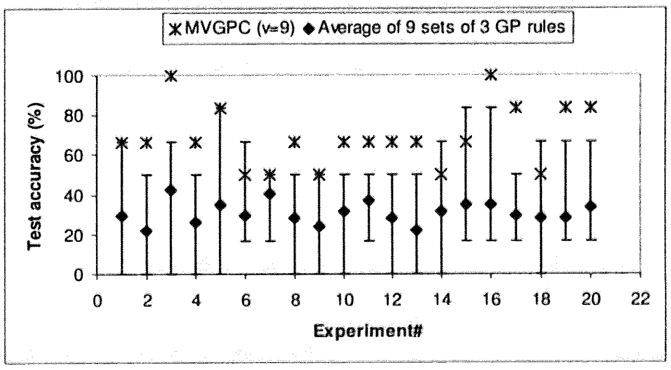
7.3.2 More frequently occurring genes

We analyzed the classification rules that produced the best results stated before to get the more frequently selected genes. In Table 7.2, we summarize those genes. In the table, the official symbol (if any) and the name of a gene are given in the format *symbol: name* under the column 'Gene description'.

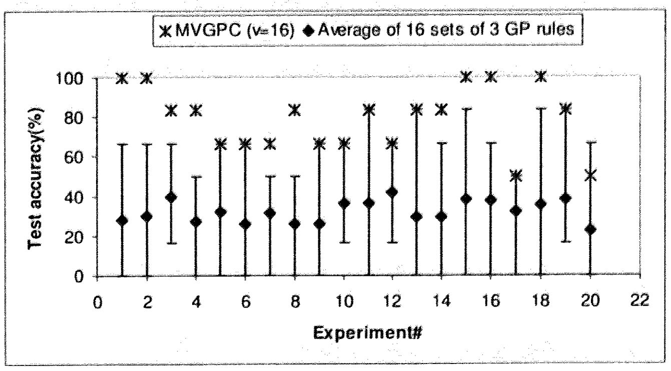
Out of 4434 genes, 2355 genes were included at least once in the 340 rules of brain cancer data. Of these, the gene 40422_at (IGFBP2) [GenBank:X16302]



(a) Number of voting rules per class=3

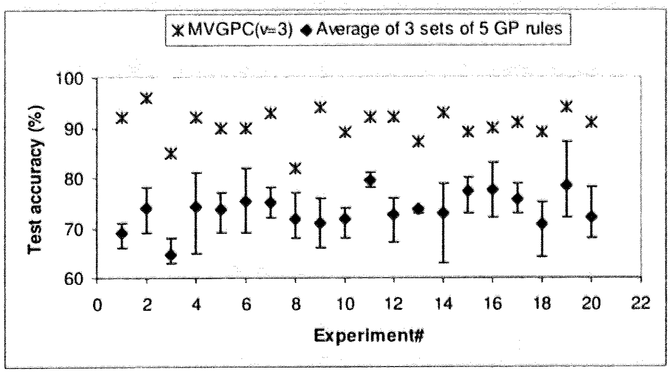


(b) Number of voting rules per class=9

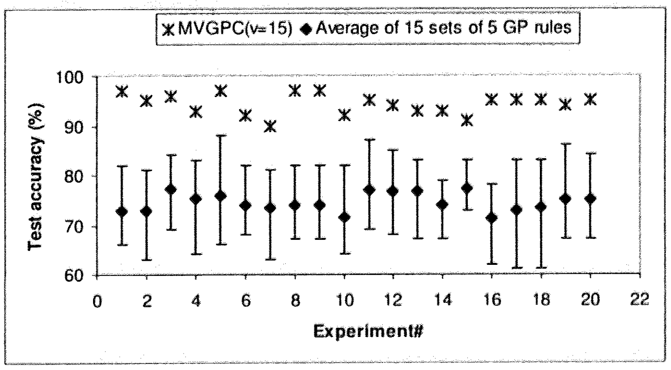


(c) Number of voting rules per class=16

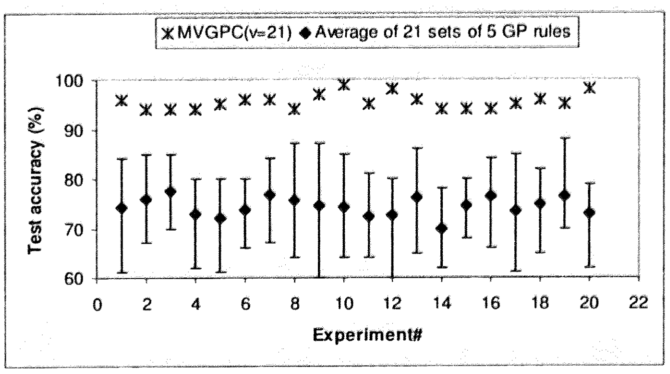
Figure 7.6: Test accuracies on breast cancer data under different conditions. For each experiment, in addition to the average accuracy, the maximum and the minimum accuracies are plotted on the graphs using error bars



(a) Number of voting rules per class=3



(b) Number of voting rules per class=15



(c) Number of voting rules per class=21

Figure 7.7: Test accuracies on lung carcinoma data under different conditions. For each experiment, in addition to the average accuracy, the maximum and the minimum accuracies are plotted on the graphs using error bars

Table 7.2: More frequently selected genes of the data sets

Data Set	Id/Feature#	Accession#	Gene description	Freq.
Brain cancer	40422_at	X16302	IGFBP2: insulin-like growth factor binding protein 2, 36kDa	33
	40840_at	M80254	PPIF: peptidylprolyl isomerase F	26
	36618_g_at	X77956	ID1: inhibitor of DNA binding 1	25
	41859_at	NM_005715	UST: uronyl-2-sulfotransferase	25
	34531_at	AF007139	Homo sapiens clone 23898 unknown mRNA, partial cds	23
	41726_at	Z35307	ECE1: endothelin converting enzyme 1	14
Prostate cancer	41468_at	M30894	Human T-cell receptor Ti rearranged gamma-chain mRNA V-J-C region, complete cds	126
	40282_s_at	M84526	CFD: complement factor D (adipsin)	107
	37639_at	X07732	HPN: hepsin (transmembrane protease, serine 1)	85
	37366_at	AL049969	Homo sapiens mRNA; cDNA DKFZp564A072	51
	32598_at	D83018	NELL2: NEL-like 2 (chicken)	49
Breast cancer	X336	NM_005749	TOB1: transducer of ERBB2, 1	18
	X1482	NM_001885	CRYAB: crystallin, alpha B	17
	X860	NM_002658	PLAU: plasminogen activator, urokinase	15
	X809	NM_001826	CKS1B: CDC28 protein kinase regulatory subunit 1B	14
	X1479	NM_053056	CCND1: cyclin D1	13
Lung carcinoma	613_at	NM_000424	KRT5: keratin 5(epidermolysis bullosa, simplex Dowling-Meara/Kobner/ Weber-Cockayne types)	142
	33904_at	NM_001306	CLDN3: claudin 3	124
	31791_at	NM_003722	TP73L: tumor protein p73-like	119
	1802_s_at	NM_001005862	ERBB2: neuroblastoma/glioblastoma derived oncogene homolog	97
	35276_at	NM_001305	CLDN4: claudin 4	89
	39990_at	NM_002202	ISL1: ISL1 transcription factor, LIM /homeodomain, (islet-1)	76
	37741_at	NM_006907	PYCR1: pyrroline-5-carboxylate reductase 1	73

was included the highest 33 times and is known to have been associated with glioma progression in part by enhancing MMP-2 gene transcription and in turn tumor cell invasion (Wang *et al.*, 2003). Among the other more frequently selected genes that are presented in the table, 41726_at (ECE1) [GenBank:Z35307] has a role in limiting Abeta accumulation in the mouse brain (Eckman *et al.*, 2003). The relationships of other frequently occurring genes with brain cancer are yet unknown.

In the case of prostate cancer data, 3096 genes out of 5966 were selected at least once in the 540 classification rules. Of the five more frequently selected genes, 37639_at (HPN) [GenBank:X07732] is known to have roles in prostate cancer progression. HPN is functionally linked to hepatocyte growth factor/MET pathway, which may contribute to prostate cancer progression (Kirchhofer *et al.*, 2005).

In the 960 rules of breast cancer data, 3038 genes out of 3226 were selected at least once. Some of these more frequently selected genes are shown in the table. The genes X1482 (CRYAB) [GenBank:NM_001885], X860 (PLAU) [GenBank:NM_002658], X809 (CKS1B) [GenBank:NM_001826] and X1479 (CCND1) [GenBank:NM_053056] are of biological interest because they are known to be associated with breast cancer.

Among the more frequently occurring genes of lung carcinoma data, 31791_at (TP73L) [GenBank:NM_003722] and 1802_s_at (ERBB2) [GenBank:NM_001005862] are known to have some roles in lung cancer. TP73L is consistently expressed in the squamous cell carcinoma in the lung, but non-consistently expressed in a subset of adenocarcinomas and large cell carcinomas (Au *et al.*, 2004); overexpression of ERBB2 is associated with recurrent non-small cell lung cancer (Onn *et al.*, 2004).

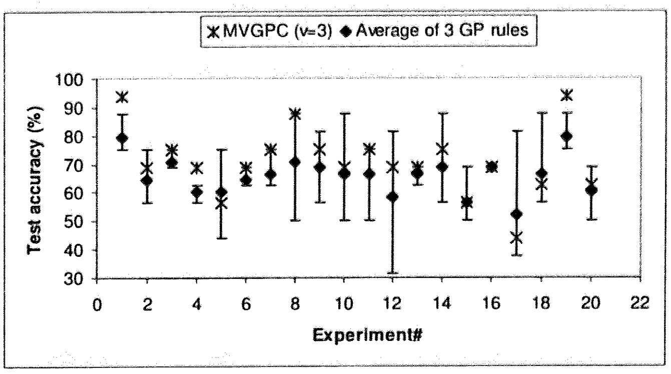
However, very few of these more frequently selected genes are included in the best single rules (or sets of rules) that individually produce the best test accuracies. There may be two reasons for this phenomenon. First, the acquired classification accuracies by these genes are reduced by the negative effects of irrelevant genes. Second, the genes involved in the best rules or sets of rules are correlated and associated with the cancer through their joint interactions; however, as single genes they are not differentially expressed in normal and/or different types of tumor samples. We need further studies in this regard to come to a concrete conclusion.

7.3.3 Effects of scaling of the values on the classification accuracy

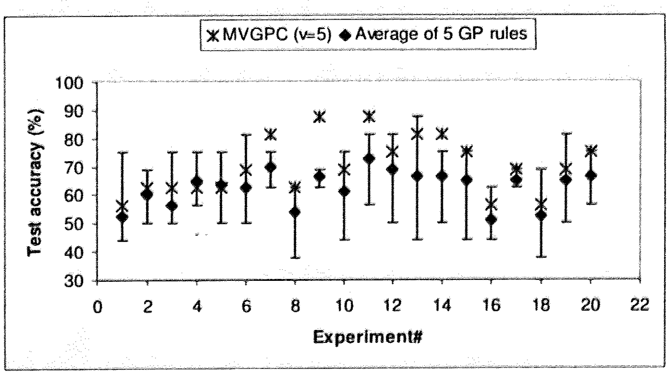
We investigated the effects of the scaling on the classification accuracy by applying MVGPC on the brain cancer and the prostate cancer data that are scaled by taking base-10 logarithm on the values. (We avoided performing experiments on the breast cancer and the lung carcinoma data because the breast cancer data are already normalized while the lung carcinoma data set contains many 0s.) In Figs. 7.8 and 7.9, we present the test accuracies on the brain cancer and the prostate cancer data. Here also the majority voting accuracy in most cases is better than the average accuracy of the rules in a voting group. Therefore, MVGPC will produce higher test accuracies than single rules in most cases; it may not matter whether the data are normalized or not. However, we did not find any statistically significant differences (at 5% level of non-parametric, two tailed t-test) between the average accuracies of MVGPC on scaled and non-scaled data.

7.3.4 Speed of convergence and computational time of MVGPC

For a given population size, the speed of convergence to the optimum fitness in a run depends on a number of factors including the training size and the complexity of the data. For two binary classification problems of same training size and same number of positive and negative samples, GP may progress to the optimum fitness at different speeds; on the contrary, for two different training sizes, GP may converge to the optimum in the same average number of generations. However, on a given data set, it is expected that the average number of generations required

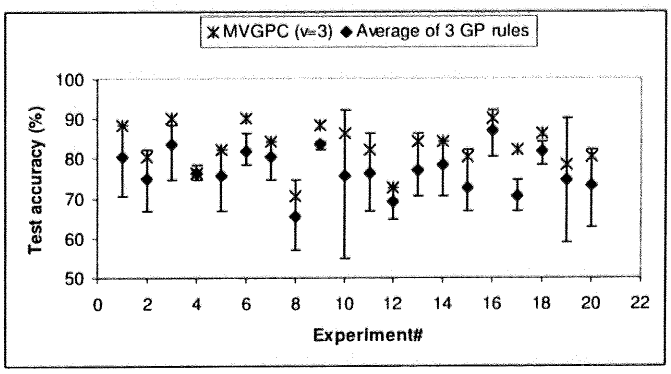


(a) Number of voting rules=3

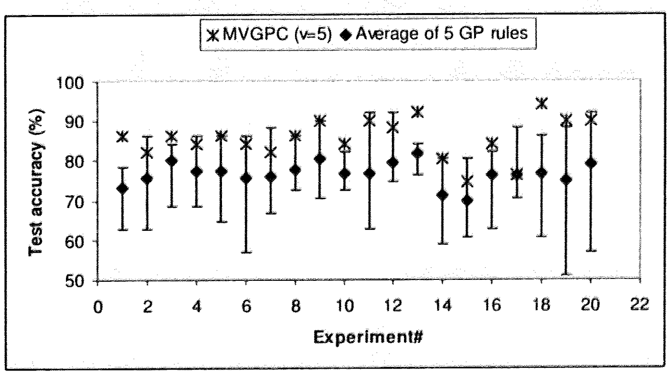


(b) Number of voting rules=5

Figure 7.8: Test accuracies on the brain cancer data that are normalized in base-10 logarithmic scale



(a) Number of voting rules=3



(b) Number of voting rules=5

Figure 7.9: Test accuracies on the prostate cancer data that are normalized in base-10 logarithmic scale

by GP to reach the optimum fitness will increase with the increasing training size. For example, the average numbers of generations required by GP to produce PTRs on prostate cancer data were 25.45 and 29.90 for training sizes of 51 and 68, respectively. During evolution of rules for multiclass classification, the speed of convergence may follow not only the training size but also the complexity of the positive samples in relative to other samples in the training subset in a run. On lung carcinoma data, GP took on the average 30.87, 9.68, 3.35, 2.42, and 5.61 generations to produce a perfect training rule for AD, SQ, SCLC, COID and NL, respectively. Interestingly, the average number of generations required by GP to produce a PTR for COID class is smaller than that required by GP to produce a PTR for SCLC class though the number of samples of COID class is higher than that of SCLC class. This suggests that the COID samples are relatively easy to separate from samples of other classes whereas samples of other classes are not easily separable from one another. In Fig 7.10, we have illustrated this by plotting the gene expression values of two typical genes 38032_at (X2744) and 39601_at (X2289) across different samples of the lung carcinoma data set. Though it is a typical case, we found many other linear GP rules containing two or three genes that could perfect classify all the COID samples, which indicates that these samples are easy to classify.

However, due to huge number of genes and different complexities of training data, GP may not reach the optimal fitness in every run of an experiment. In the worst case, the number of fitness evaluations per experiment will be $P \times G \times v \times c$ where P is the population size, G is the maximum number of generations per run, v is the number of rules per class, and c is the number of rules needed for classification of the data (for binary classification, $c = 1$). MVGPC was encoded in Java Programming Language and executed on machines each having Athlon 64 X2 4400+ processor, 2GB of RAM and running Scientific Linux OS. The average computation time (CPU time) for an experiment on the brain cancer ($v = 17$), prostate cancer ($v = 27$), breast cancer ($v = 16$), and lung carcinoma ($v = 21$) data sets were 15411.48, 43282.62, 995.80, and 84663.96 seconds, respectively. As expected, the execution time of each experiment was according to the complexity

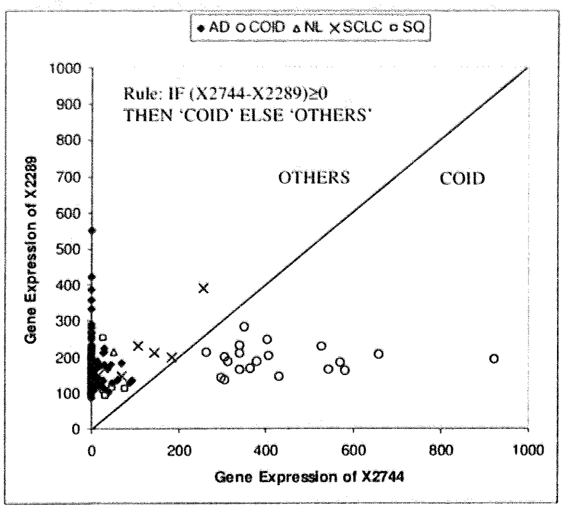


Figure 7.10: Plot of gene expressions of 38032_at (X2744) and 39601_at (X2289) across different samples of lung carcinoma

of the data set; MVGPC took the longest time to produce the 105 rules in an ensemble of lung carcinoma whereas it took the shortest time to produce the 48 rules in an ensemble of breast cancer.

However, if the dimensionality of the problem is reduced by applying a suitable technique before applying MVGPC, the computation time may be reduced drastically. In our future works, we want to investigate the performance of MVGPC on the data of the more frequently selected genes.

7.3.5 Comparative test accuracies of different methods

The subsection 7.3.1 shows how MVGPC can be used to improve the test accuracies on the four data sets. In this subsection, we present the comparative test accuracies on those data sets using different computational methods.

MVGPC is very much related with AdaBoost technique. To investigate whether MVGPC is competitive with AdaBoost or not, we performed experiments with AdaBoost using genetic programming as a weak classifier (denoted as *AdaBoost+GP*). For binary classification, simple AdaBoost was implemented while for multi-class classification, AdaBoost.M1 (Freund and Schapire, 1997) was implemented. For

Table 7.3: Comparative test accuracies on the data sets

Method/Data set	Brain	Prostate	Breast	Lung
MVGPC	80.31±5.08	90.59±2.07	79.17±16.11	95.50±1.54
AdaBoost+GP	71.88±12.74	84.31±6.36	63.33±17.61	92.25± 2.47
PTR/PSTR	67.90±10.86	79.21±6.67	32.24±18.23	75.55±5.66
kNN+RPMBGA	67.50±10.26	84.41±4.99	60.0±15.67	89.35±3.34
SVM+GA	56.25±0.0	51.37±1.97	65.0±23.51	76.80±1.51

Table 7.4: p-values in statistical tests of significance (MVGPC vs other method)

Method/Data set	Brain	Prostate	Breast	Lung
AdaBoost+GP	6.08E-03	3.90E-05	2.80E-03	1.13E-05
PTR/PSTR	2.98E-12	2.21E-36	6.22E-36	6.54E-17
kNN+RPMBGA	1.72E-06	1.01E-07	2.92E-04	6.31E-11
SVM+GA	7.71E-13	6.16E-15	2.06E-02	9.04E-14

each experiment using AdaBoost, the numbers of iterations for the brain cancer, prostate cancer, breast cancer and lung carcinoma data sets were 17, 27, 16 and 21, respectively, which are the numbers of rules per voting group that obtained the best average test accuracies using MVGPC above.

Definition 1 (Perfect training rule). For a binary classification problem, we define a perfect training rule (PTR) as the classification rule of genetic programming that can perfectly classify all the training samples in the data set and thus has a fitness of 1.0. For a multiclass data set containing c classes of samples, a collection of c PTRs—one PTR for each class of sample is defined as a perfect set of training rules (PSTR).

Next, we analyzed the test accuracies obtained by the perfect training rules (or the perfect set of training rules) of the genetic programming classifier. The numbers of PTRs/PSTRs among the majority voting rules that produced the best average test accuracies on the brain cancer, prostate cancer, breast cancer, and lung carcinoma data were 192, 141, 320, and 31, respectively.

To support the claim that MVGPC is superior to other techniques, we performed additional experiments using the kNN classifier with RPMBGA (Chap. 5) (denoted as *kNN+RPMBGA*), and SVM with GA (denoted as *SVM+GA*). To evaluate a gene subset of GA/RPMBGA, the classifier (SVM or kNN) was applied to the data of the selected genes in an individual. We chose the SVM and the kNN classifier because these methods are widely used by the bioinformatics community for prediction of cancer class using gene expression data. We chose RPMBGA and GA as gene selection algorithms because they are evolutionary computation methods like GP and are intensively used in selection of informative genes. Therefore, the comparison of MVGPC with these two techniques (kNN+RPMBGA and SVM+GA) on the data sets is convincing.

For RPMBGA and GA, each individual was encoded as a binary string of 0's and 1's, and the population size, offspring size, crossover and mutation probabilities (for GA), maximum number of generations, and maximum number of runs (=experiments in MVGPC) were the same as of MVGPC. For GA, we used uniform crossover to create offspring as the number of bits in an individual is huge. Though initial population of both methods were randomly generated, we restricted the the maximum number of genes selected in each individual of GA to 100 (=the maximum number of nodes in a tree of rule of MVGPC) because without this restriction, the fitness of GA did not improve over initial generation. The fitness of a gene subset was evaluated using the same method used in Chap. 5. For kNN with RPMBGA, we performed different experiments on the four data sets using 3, 5, and 7 nearest neighbor members in the kNN classifier. In terms of average training and test accuracies, the best results were obtained on the brain cancer, prostate cancer, breast cancer, and lung carcinoma data with k = 3, 3, 5, and 3, respectively. For SVM with GA, we used LIBSVM (Chang and Lin, 2001) implementation of SVM with C=32, and $\gamma = 0.0078125$.

In Table 7.3, the comparative test accuracies obtained by applying different methods on the four data sets are summarized. The average test accuracies of MVGPC on the data sets are better than those of other methods presented in the table. To test whether the difference in average values are statistically significant or not, we performed non-parametric t-tests (one tailed, two-sample unequal variance) on the values using Microsoft Excel. The p-values returned by the *ttest* function of Microsoft Excel are shown in Table 7.4. As the p-values are very small, the average test accuracies of MVGPC are significantly higher (at 5% level of non-parametric t-test) than those of other methods.

7.3.6 Overfitting on the data sets

Though MVGPC is much better than other methods presented in this chapter, overfitting had occurred and 100% test accuracy had not been obtained on the data sets except on the breast cancer data. The biggest overfitting can be observed on the breast cancer and the brain cancer data set, which are very typical microarray data sets. The test set of the brain data set contains non-classic gliomas raising the possibility that the test data are diverged from the training data. Even using all the data of brain cancer as the training data, we did not find any GP rule that could classify all the samples 100% accurately, which shows the difficulty of the classification of this data set. For breast cancer, the number of training samples is very small, specially the number of sporadic samples in the training subset—four training samples for two test samples. We found in our experiments that either or both of the two sporadic test samples were misclassified in the 15 cases where majority voting test accuracy was below 100%. However, by performing some experiments with 5:1 ratio of training to test size of sporadic samples (training size=17), we found the average majority voting test accuracy much improved. In the case of the prostate cancer and the lung carcinoma data, the numbers of test and training samples were almost equal. There is a high possibility that increasing the number of training samples may result in higher test accuracy.

7.4 Discussion

In a microarray data set, the number of genes is huge compared to the number of training samples and many of these genes are redundant. These redundant genes sometimes affect negatively the classification accuracy acquired by other relevant genes; sometimes they have no effect on the acquired accuracy. Moreover, the numbers of training samples of different categories are not evenly distributed; some classes have more samples than some other classes. In most cases, it has been observed that a perfect training rule having more supportive training data causes less overfitting than a perfect training rule having smaller number of training samples. It sometimes may happen that the training accuracy of a classifier is 100% but its accuracy on test data is 0%. In our experiments, we observed this phenomenon on the breast cancer data.

Usually, the perfect training rules evolved by using a machine learning approach are applied to predict the labels of test samples. In our comparative results, we have shown that these single PTRs/PSTRs produce very poor test accuracies; whereas, using an ensemble technique like MVGPC, these poor classification rules produce much better test accuracies. Though AdaBoost is widely used as an ensemble technique to improve the prediction rate of a weak classifier, we have found that MVGPC may outperform AdaBoost in classification of gene expression data.

However, the success of majority voting depends on the number of rules per voting group and on the rate of false prediction (on test samples) by single rules. If either the number of rules per voting group in majority voting is very small or the rate of false prediction by all single rules is greater than or equal to 0.5, MVGPC will be no better than single rules or sets of rules. Since the maximum number of nodes in a GP tree is restricted to very small compared to the huge number of genes in a data set, there is a high possibility that all the evolved GP rules in the ensemble of MVGPC are different from one another. Therefore, when the number of rules in the ensemble of MVGPC is sufficiently large, the probability of false prediction rate of all the rules being 0.5 or more will be very low.

Assuming that most of the rules in the majority voting have false prediction rate below 0.5, the optimal number of rules should be determined based on the training data. From our experiments, we have noticed that for a data set like breast cancer containing smaller number of training samples, the best test accuracies may be obtained when the number of members per voting group is equal to the number of training samples. However, for a data set containing many training samples, generation of all the rules may not be feasible; smaller number of rules may be sufficient. For a larger data set like lung carcinoma, our hypothesis is that the total number of rules should be approximately equal to the number of training samples. For other data sets, the number of samples per voting group of MVGPC may be determined using equations in Subsection 7.2.3. We need further experiments to verify this hypothesis.

7.5 Summary

In this chapter, we introduce the majority voting technique for the prediction of the class of a test sample by the rules of the genetic programming. By performing experiments on four public cancer data sets including two multi-category data sets, we have found that in almost all cases, the accuracy obtained with majority voting is better than the average accuracy of single rules or sets of rules. Individually those rules or sets of rules classify the test samples very poorly but as a group of rules, they classify the samples very accurately.

To support our claim that MVGPC is competitive with other methods, we did additional experiments with the state of the art methods in the classification literature and found that the test accuracies obtained by MVGPC are superior to those by other methods. We have also found that scaling does not affect the higher accuracy of the MVGPC negatively. These evidences strongly suggest that our proposed method is an appropriate method for the prediction of the labels of test samples. Moreover, some of the more frequently occurred genes in the evolved rules of MVGPC are the potential biomarkers of the types of cancers considered in this chapter.

Chapter 8

Evaluation of MVGPC on Non-Microarray Data

In the previous chapter, we have found that MVGPC obtains better test accuracy than other methods on microarray gene expression data, where the number of genes is huge compared to the number of training samples. Moreover, we performed MVGPC experiments with arithmetic functions. In this chapter, we perform experiments on non-microarray data containing larger number of samples compared to smaller number of attributes. We performed experiments on these data sets with arithmetic and/or logical functions. The objective of this chapter is to show whether MVGPC obtains competitive accuracy on non-microarray data or not.

8.1 Non-microarray data sets

We downloaded two data sets: Wisconsin breast cancer and Monk's problem data from UCI machine learning (ML) repository (Newman *et al.*, 1998) at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

8.1.1 Wisconsin breast cancer data

Wisconsin diagnostic breast cancer data set is a collection of 569 samples described by 30 numeric features. It is a binary classification problem (benign or malignant), and contains 357 begin and 212 malignant samples.

8.1.2 Monk's problem

The Monk's problem is a very simple binary classification problem consisting of artificial data of 6 attributes. There are three Monk's problem; we have used Monk-1 and Monk-3 for our experiments. The Monk-1 problem consists of 124 training samples and 432 independent test samples. The target concept for this problem is

IF ((X1=X2) OR (X5=1)) THEN 1 ELSE 0.

The Monk-3 problem has 122 training samples and 432 test samples. 5% noises are added to the class labels of the training data; the six mislabeled samples in the training subset are *data_5, data_61, data_203, data_213, data_214,* and *data_391.* The target concept is

IF ((X5 =3 AND X4 = 1) OR (X5 /= 4 and X2 /= 3)) THEN 1 ELSE 0.

8.2 Results

We performed experiments on the two data sets with the settings of different parameters as shown in Table 8.1. As functions, we considered two sets depending on the types of attributes:

- Wisconsin breast cancer data:
  - arithmetic functions: { +,-,*, /, SQR, SQRT} or
  - arithmetic and logical functions: { +,-,*, /, SQR, SQRT, AND, OR, NOT, =, <>, >, <, >=, <=};
- Monk's problem:
  - logical functions: {AND, OR, NOT, =, <>, >, <, >=, <=} or
  - arithmetic and logical functions: { +,-,*, /, SQR, SQRT, AND, OR, NOT, =, <>, >, <, >=, <=}.

Table 8.1: Values of different GP parameters

Parameter	Value
Population size	4000
Max generations	50
Runs	20
Reproduction rate	0.1
Crossover rate	0.9
Mutation rate	0.1
Max nodes in a GP tree	100

Table 8.2: Accuracies of MVGPC and single rules on Wisconsin breast cancer data

Function type	Ensem-ble size	MVGPC		Single rules	
		Training	Test	Training	Test
Arithmetic and logical	11	97.47±0.84	94.49±1.36	96.39±1.24	92.96±1.89
	31	97.63±0.83	95.04±1.13	96.26±1.28	93.42±1.68
Arithmetic	11	97.68±0.75	94.84±1.04	96.69±0.97	93.60±1.51
	31	97.72±0.72	95.37±0.95	96.58±0.97	93.74±1.66

On Wisconsin breast cancer data, we performed experiments with ensemble size 11 and 31. Since this data set is not divided into training and test subsets, we constructed those subsets by randomly splitting the data into 1:1 ratio in each experiment. The training subset contained 179 benign and 106 malignant samples. The accuracies obtained by MVGPC and single rules are shown in Table 8.2. All the accuracies obtained with MVGPC are significantly better (at 5% level of t-test) than those obtained with single rules. However, the accuracies obtained with either ensemble size and function set did not change significantly from one other. The most important six features of this data set are X8, X288, X10, X27, X7 and X14.

For Monk's problem, we performed experiments with ensemble size 3 and 11. On Monk-1 problem, both MVGPC and single rules got 100% training and 100%test accuracy in every run with either set of the functions and with either size of ensemble. The most important features of this data set found by MVGPC are X1, X2 and X5, which are consistent with the features in the target concept.

Table 8.3: Accuracies of MVGPC and single rules on Monk-3 problem

Function type	Ensemble size	MVGPC		Single rules	
		Training	Test	Training	Test
Arithmetic and logical	3	93.52± 0.37	97.16±0.26	93.84±0.87	96.72±1.66
	11	93.44± 0.0	97.22±0.0	94.02±0.85	97.11±1.44
Logical	3	93.44± 0.0	97.22±0.0	93.46±0.11	97.19±0.24
	11	93.44±0.0	97.22±0.0	93.44±0.0	97.22±0.0

On Monk-3 problem with six mislabeled classes, the performance of MVGPC and single rules are shown in Table 8.3. In the experiments, slightly better accuracies were obtained with the set of arithmetic and logical functions than with the set of logical functions. Using the set of arithmetic and logical functions, only 25 rules out of 220 (=11*20) could classify all the test samples correctly, and 116 training samples correctly labeled. Two rules that classified 119 and 118 training samples could classify respectively 406 and 410 test samples out of 432 correctly. That is to say these two rules are not perfect classifiers. In summary, the overall accuracies of MVGPC are not significantly different from the accuracies of single GP rules. The most important features in the rules evolved by using the set of logical functions are X2 and X5 while those in the rules evolved by using the set of arithmetic and logical functions are X2, X4, and X5.

8.3 Summary

In this chapter, we have shown the comparative accuracies of MVGPC and single rules on non-microarray data. The test accuracies of MVGPC and single rules were the same on Monk-1 problem. However, on noisy data like Monk-3 problem, the test accuracies of single rules were slightly better because some rules of the ensemble of MVGPC could obtain 100% training and test accuracies but the number of such rules was very small compared to the ensemble size. However, on Wisconsin breast cancer data, the accuracies of MVGPC were significantly better than those of single rules. Therefore, MVGPC will perform better than single rules in most cases on non-artificial data.

Chapter 9

Conclusions and Future Works

9.1 Conclusions

In this dissertation, we have focused on extraction of informative genes from microarray data and design of a reliable classifier for classification of patient samples with a view to developing a gene expression based cancer diagnosis and treatment system. Though different gene selection methods and classifiers have been proposed in the literature (Chaps. 3 and 4), their successes are very limited due to huge number of genes compared to smaller number of training samples and many redundant and irrelevant genes. To this end, we have proposed two methods: *random probabilistic model building genetic algorithm (RPMBGA)* (Chap. 5) and *majority voting genetic programming classifier (MVGPC)* (Chap. 7). Both methods obtain very higher accuracies on independent test data compared to other methods but RPMBGA needs a classifier whereas MVGPC itself acts a classifier and a gene selection method and is better than RPMBGA. The summary of these two methods is given below.

9.1.1 Random probabilistic model building genetic algorithm

RPMBGA, a variant of genetic algorithm, is a gene selection method. The main characteristics of RPMBGA are that it is faster than traditional gene selection method like genetic algorithm, has no crossover or mutation, and obtains compact gene subsets and higher classification accuracies compared to other competitive techniques.

Starting from the initial population with many genes selected in each individual, RPMBGA successively reduces many irrelevant genes and finally terminates with a population having very small number of genes selected in each individual. It generates subsets of more than one gene a time and preserves the interactions among the genes. These are advantageous over rank-based methods that selects one gene at a time based on single genes' capability of data separation because we have found evidence that the genes of a subset that produces the best classification accuracy using a classifier do not produce higher classification accuracy individually. Moreover, neither single genes nor gene subsets containing very higher number of genes classify patient samples perfectly. In subsets with very higher number of genes, the irrelevant genes act negatively on the classification accuracy obtained by other informative genes.

However, RPMBGA suffers from the problem that the selected genes or the acquired classification accuracy is dependent on the classifier employed to calculate the fitness of gene subsets.

9.1.2 Majority voting genetic programming classifier

MVGPC, based on genetic programming (GP) and majority voting technique, improves the classification accuracies of GP and is more reliable than RPMBGA. Though GP has advantages over other evolutionary computation methods that it acts as a classifier and a gene selection method and produces transparent classification rules, which provide an insight into the quantitative relationships among the genes in classification of samples, it produces very poor test accuracies when the accuracy of a rule is calculated by executing it on the training data in one pass. The ways the optimum ensemble size be determined, the label of a test sample be predicted using the ensemble, and the potential biomarkers be extracted from microarray data are our main contributions in MVGPC.

Accuracy improvement

MVGPC uses an ensemble of different genetic programming rules and appears to be a reliable and robust method for prediction of the label of a test sample. In its typical implementation, we evolve multiple rules in different GP runs, apply them one by one to a test sample and count their votes in favor of a particular class. Then the sample is assigned to the class that gets the highest number of votes in favor of it. Our assumption behind MVGPC is that single rules that are evolved using genetic programming are not strong enough to predict the labels of samples and a team of rules can make decision with enough confidence. Though AdaBoost is widely used as an ensemble technique to improve the prediction rate of a weak classifier, we have found that MVGPC may outperform AdaBoost+GP in classification of gene expression data. The probable reason may be that the rule or set of rules evolved with GP in each iteration of AdaBoost may not use all the training samples whereas the rules of MVGPC use all training samples and are stronger than those of AdaBoost+GP.

However, the success of majority voting depends on the number of rules per voting group and on the rate of false prediction (on test samples) by single rules. If either the number of rules per voting group in majority voting is very small or the rate of false prediction by all single rules is greater than or equal to 0.5, MVGPC will be no better than single rules or sets of rules. By restricting the maximum number of nodes in a GP tree to be very small, diversity among the rules can be preserved, and with sufficient number of diverse rules, MVGPC will perform better than single rules. However determination of optimal ensemble size from training data is not straight forward. Our hypothesis is that for smaller data sets, the ensemble size should be equal to the number of training samples; for larger binary data sets, the ensemble size should be the half of the training size while for larger multiclass data sets, the total number of rules per ensemble should be approximately equal to the number of training samples. To this end, equations in Subsection 7.2.3 (Chap. 7) may be employed to determine the near optimal ensemble size.

Biomarkers identification

For identification of potential biomarkers, we propose that classifier be first devised, which will obtain higher classification accuracy, and then the evolved rules be analyzed to determine the most frequently occurring genes, i.e. first classification, then gene selection. To get a more stable frequency distribution of selected genes, MVGPC should be repeated on the microarray data for several times. Our proposal is based on the observation that some genes are frequently always selected whatever gene selection algorithms and classifiers are used. These more frequently selected genes may be either potential biomarkers of cancers or junk genes that are highly correlated with distinction of different training and test samples but have no biological significance.

In the experiments performed according to the proposal, we have found that some of the more frequently selected genes in the rules of MVGPC are biologically significant and related with the types of cancer being studied in this dissertation while the relationships of some other more frequently selected genes with the cancers are yet unknown.

9.2 Future works

Though our reported models of gene selection and classifications have classified clinically ambiguous tumors with higher accuracy, it is not 100% perfect. Indeed, more training and test samples, and clinical trials are needed to determine how best the molecular-based diagnosis will fit the standard patient care. Besides, there are some unresolved technical issues that are discussed in this section.

9.2.1 Classification of unbalanced data

Many real-world data including biological and financial data are very unbalanced. In unbalanced data, one class has much larger number of samples than other classes. By applying traditional classifiers, one may obtain very high classification accuracies on unbalanced data but those accuracies are heavily biased by the majority class, and the sensitivity or the specificity will be very unbalanced. Therefore, simple accuracy is a useless index of performance measurement of a classifier for unbalanced data. To get meaningful results, the sensitivity and specificity information must be incorporated into the fitness function of a GP rule. The correlation based fitness function (6.2.4) defined in Chap. 6 is one that incorporates this information. Other functions that take into account sensitivity and specificity information are defined below.

F-measure:

$$F - measure = \frac{2 * recall * precision}{recall + precision} \tag{9.2.1}$$

where

$$recall = \frac{N_{tp}}{N_{tp} + N_{fn}}; \qquad precision = \frac{N_{tp}}{N_{tp} + N_{fp}}$$

with N_{tp}, N_{fn} and N_{fp} being the numbers of true positives, false negatives, and false positives, respectively. In binary classification, recall and precision are analogous to sensitivity, and positive predictive value, respectively.

Area under ROC curve(AURC): Hanley and Mcneil (1982) have shown that nonparametric Wilcoxon statistics can be used to measure the area under ROC (receiver operating characteristic) curve in a rating method. That is,

$$AURC = \frac{1}{2}(sensitivity + specificity) \tag{9.2.2}$$

where

$$sensitivity = \frac{N_{tp}}{N_{tp} + N_{fn}}; \qquad specificity = \frac{N_{tn}}{N_{tn} + N_{fp}}$$

with with N_{tp}, N_{fn}, N_{tn} and N_{fp} being the numbers of true positives, false negatives, true negatives and false positives, respectively. Note here that AURC is the arithmetic mean of sensitivity and specificity.

Geometric mean of sensitivity and specificity:

$$G - mean = \sqrt{(sensitivity * specificity)}. \tag{9.2.3}$$

Biased fitness: Biased fitness function incorporates sensitivity and specificity information but assigns higher weight to either sensitivity or specificity. One such fitness function is as follows:

$$B - fitness = \frac{1}{2} sensitivity * specificity + \frac{1}{2} sensitivity. \qquad (9.2.4)$$

The first term balances sensitivity and specificity equally while the second term biases the fitness towards sensitivity.

It is suggested that the performance of MVGPC be verified on the unbalanced data by utilizing these fitness functions during evolution of GP rules for the ensemble of MVGPC.

9.2.2 Use of logical functions instead of arithmetic functions

In MVGPC, the output of a rule is a floating-point number and the slice point for determination of class membership is zero. There are some limitations of this method. For example, suppose that a microarray data set contain two types of samples: A and B. If the output of two rules: R_1 and R_2 on a sample Y is 0.1 and 100.0, Y is predicted to be of type A in both cases; however, the confidence level of R_2 is much higher than that of R_1. Instead of real-valued functions, we can consider logical functions to evolve genetic programming rules. In lieu of values, the logical relationships among the genes would be biologically more interesting because the prediction models may sometimes be affected by the scaling of gene expression values. It is recommended that logical functions be used to determine the relationships among the potential biomarkers.

9.2.3 Determination of ensemble size dynamically

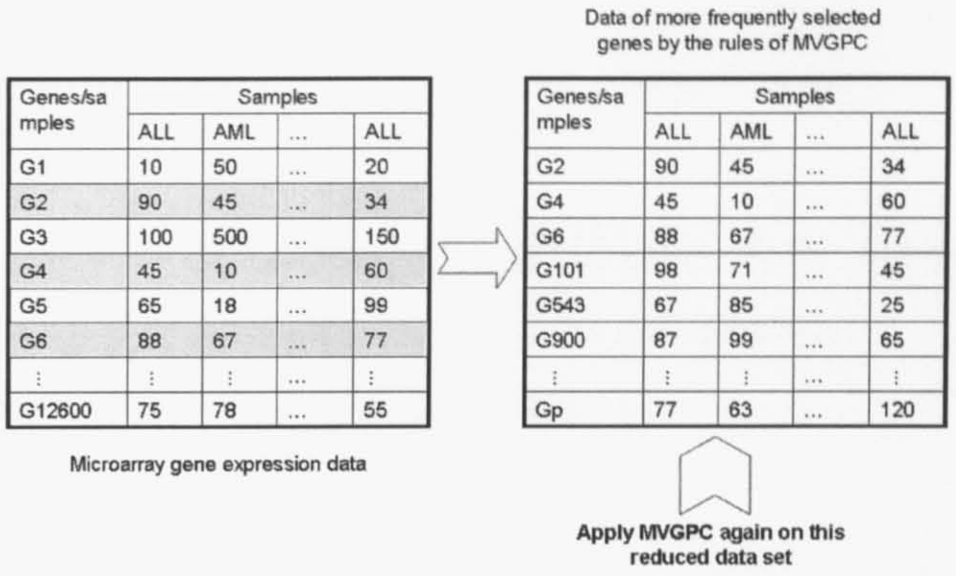
In Chap. 7, we have shown that increasing the ensemble size increases the test accuracy but generation of large number of rules may not be feasible. Moreover, determination of ensemble size based on a single training set is difficult. It would be better if we can determine the ensemble size dynamically. In this context, the available training samples should be divided into internal training and validation subsets, and the optimal number of rules in the ensemble should be determined based on the performance of an ensemble on the validation set.

9.2.4 MVGPC on the data of more frequently selected genes

In MVGPC, though we get better test accuracy through the ensemble of different rules, we are concerned about the number of genes involved in an ensemble. Moreover, in Sec. 7.3.4, we have shown that the execution time of a typical MVGPC experiment on a larger, multiclass microarray data set is very longer compared to other methods. It is suggested that performance of the ensemble of rules containing only the more frequently selected genes (see Fig. 9.1) be investigated because there is a possibility that this ensemble may produce better test accuracy. Here MVGPC will be applied in two stages. In the first stage, MVGPC will be used as a preprocessor to identify more frequently occurring genes while in the second stage, it will be applied to build more strong predictive model using only the selected genes in previous stage.

9.2.5 Comparison of MVGPC with other ensemble methods

In this work, we have found that both MVGPC and AdaBoost+GP obtain the same level of accuracies on training data but MVGPC obtains significantly better accuracies than AdaBoost+GP on test data. Besides AdaBoost, there are have been proposed a number of ensemble techniques in literature for improving the classification accuracy of weak classifiers. It is recommended that one compare MVGPC with other ensemble techniques.


Data of more frequently selected genes by the rules of MVGPC
Microarray gene expression data
Apply MVGPC again on this reduced data set

Genes/samples	Samples			
	ALL	AML	...	ALL
G1	10	50	...	20
G2	90	45	...	34
G3	100	500	...	150
G4	45	10	...	60
G5	65	18	...	99
G6	88	67	...	77
⋮	⋮	⋮	...	⋮
G12600	75	78	...	55

Genes/samples	Samples			
	ALL	AML	...	ALL
G2	90	45	...	34
G4	45	10	...	60
G6	88	67	...	77
G101	98	71	...	45
G543	67	85	...	25
G900	87	99	...	65
⋮	⋮	⋮	...	⋮
Gp	77	63	...	120

Figure 9.1: MVGPC on the data of more frequently selected genes

9.2.6 Investigation of influences of different GP parameters

In Chap. 7, we have shown that the performance of MVGPC is very much dependent on the false prediction rates by single rules of GP. And there are many parameters like population size, offspring size, crossover and mutation rates, length of an individual, etc. that influence the performance of each GP rule. It is absolutely necessary to verify the performance of MVGPC by varying the values of different GP parameters.

Appendix A

EGPC: A Powerful Tool for Data Classification and Important Features Identification

The majority voting genetic programming classifier (MVGPC) has been implemented in Java programming language and named as EGPC (ensemble of genetic programming classifiers). The software can be downloaded from the IBA Laboratory homepage (`http://www.iba.k.u-tokyo.ac.jp/english/EGPC.html`). EGPC is a very powerful, easy to use Java based tool for preprocessing and classification of data and for identification of important features from data. The main features of EGPC are that:

1. It improves the test accuracies of genetic programming rules;
2. It runs in command line interface (CLI) and graphical user interface (GUI) modes;
3. It can be used for binary and multi-class classification;
4. It can handle microarray gene expression data as well as UCI machine learning (ML) databases (with no missing values) (Newman *et al.*, 1998);
5. It can handle numeric, nominal and Boolean (converted to numbers) features;
6. It can evolve rules with arithmetic and/or logical functions;
7. It can handle training subsets constructed by fixed or random split of data; and

8. It can handle training and validation data stored in two separate files provided that they have the same number of features and those features are in same order.

Moreover, EGPC can be applied repeatedly on the data for a reduced set of features and possibly for better test accuracies; in this case, EGPC acts as a preprocessor of features.

A.1 Data format

EGPC can handle two formats of (tab, colon, semicolon, or space delimited) text files: microarray and UCI ML data files. In microarray format, the first row contains the labels of the samples and the other rows contain the gene expression values of different genes in different samples. One file: BreastCancer.txt in the examples is in this format.

In UCI ML format, the first column contains the labels of the samples and the other columns contain the values of attributes in different samples/instances. Two files: Monk.txt and WCBreast.txt in the examples are in this format.

In either form, the labels of samples must be numeric values starting from 0, i.e. for binary classification, the labels of samples will be either 0 or 1 while for multi-class classification, the labels will be from 0 to $(c-1)$ where c is the number of classes of samples in the data file.

A.2 Training and test subsets constructions

The training and the test data can be in one file or in two separate files. If the data are in two separate files, the split of whole data is fixed. While the data are in one file, the training and test subsets can be constructed randomly or by fixed split of the data. In the case of random split, the information about training subset is entered by combining the training sizes of different classes, delimited by colon (:), into a single string. Let us give an example of this. Suppose a data set consists of three classes of samples, and the number of samples of the classes is 8, 8, and

6, respectively. If the training subset consists of 50% of the samples, the string containing information about the training data would be 4:4:3. However, in the case of the fixed split of the samples stored in one file, a file containing the indexes of the training samples should be passed to the program. See BreastTrainIndex.txt for an example of the fixed split of BreastCancer.txt data. Note that the index of the first sample is 1.

A.3 Attributes/Genes

EGPC can handle numeric, nominal and Boolean attributes. If any attribute is in nominal format, convert it to numeric format. For example, if the values of a nominal attribute is Sunny, Cloudy, Rainy, and Snowing, convert these values to 0, 1, 2, and 3, respectively. The identifiers for these attributes are as follows: N: Numeric; L: Nominal; and B: Boolean. If a range of the attributes are of same type, they can be indicated by < start index >:< end index >. For example, if the first 30 attributes of a data set are numeric, indicate it by N1:30. For nominal attributes, there is one extra parameter, the maximum value of the nominal attribute, which follows the < end index > separated by colon. For example, L1:30:4. For multi-type attributes, indicate each type and range by the above notations and separate each other by colon. An example may be N1:30:L31:50:4:B51:60.

Each feature in the data file is represented by an 'X' followed by the feature number. For example, X1314 represents the feature 1314 of a data set.

A.4 Functions

EGPC can handle the following functions :{ +,-,*, /, SQR, SQRT, LN, EXP, SIN, COS, AND, OR, NOT, =, <>, >, <, >=, <=}. If all the attributes are numeric, one can use all the above functions, only arithmetic functions: {+,-,*, /, SQR, SQRT, LN, EXP, SIN, COS}, or only logical and Boolean functions: {AND, OR, NOT, =, <>,>, <,>=, <=}. Note here that since all the attributes are numeric, only AND, OR, NOT can not be used. If all the attributes are either nominal or Boolean, we recommend using logical and Boolean functions only. However, EGPC would be able to handle the combinations as nominal attributes are converted to numeric values (by the user) before running the software.

Some functions are executed in protected mode to avoid undefined results—underflow or overflow. These functions are treated as follows:

- $SQRT(Y) = 0$ if Y is negative;
- $Y/Z = 1$ if Z is 0;
- $EXP(Y) = EXP(Max(-10000, Min(Y, 20)))$; Y is bounded between [-10000,20];
- $LN(Y) = 0$ if $Y = 0$; otherwise $LN(Y) = LN(|Y|)$ and Y is bounded between [-1.0E+100, 1.0E+100];
- $SIN(Y)$ and $COS(Y)$: Y is bounded between [-10000, 10000].

A.5 Ensemble size

The number of single rules or sets of rules that participate in majority voting is indicated here as ensemble size. The minimum ensemble size is 3. For binary classification, we recommend an odd value for ensemble size.

A.6 Evolved rules (S-expressions)

The expression of a rule is called S-expression. The predicted class of a rule is determined depending on the output of the rule:

- Boolean output (an S-expression consists of logical or logical+arithmetic functions):
  - Binary classification: IF (S-expression) THEN 0 ELSE 1.
  - Multi-class classification: IF (S-expression) THEN TargetClass ELSE Other.

Real-valued output (an S-expression consists of arithmetic functions only):

  - Binary classification: IF (S-expression$\geq$0) THEN 0 ELSE 1.
  - Multi-class classification: IF (S-expression$\geq$0) THEN TargetClass ELSE Other.

Therefore, the slice point for real-valued output is 0.

A.7 Default settings of some GP parameters

The default settings of some GP parameters in EGPC are as follows:

- Maximum number of nodes in a GP tree=100;
- Maximum initial depth=6;
- Initial population generation method: ramped half-and-half;
- Maximum crossover depth=7;
- Selection method for crossover: greedy-over;
- Reproduction probability=0.1;
- Crossover probability=0.9;
- Mutation probability=0.1;
- Termination criteria: fitness=1.0 or maximum number of generations has passed; and
- Regeneration type: elitism (elite size=1).

A.8 Example files included with this software bundle

Monk's problem (Monk.txt): It has been downloaded from `http://www.ics.uci.edu/~mlearn/MLRepository.html`. It is a binary classification problem, and consists of 6 nominal attributes (values: 1-4), and total 556 samples divided into 124 training and 432 test samples. We have put the 432 independent test samples into the file MonkValid.txt that are used as a validation file for the problem. The data are in UCI ML format. It is a simple problem; the target concept is: IF ((X1 = X2) OR (X5 = 1)) THEN 1 ELSE 0.

Wisconsin breast cancer data (WCBreast.txt): It has been downloaded from `http://www.ics.uci.edu/~mlearn/MLRepository.html`. It consists of 30 numeric attributes, and a total of 569 samples. It is a binary classification problem. To use as an example, we randomly split this data set into training and test subset into 1:1 ratio.

Breast cancer data (BreastCancer.txt): It is a microarray data file consisting of the gene expressions values of 3226 preprocessed genes across 22 samples. It is a 3-class classification problem. The classes of the samples are BRAC1, BRAC2 or Sporadic. All the attributes are numeric. The training subset consists of the fixed split of this data set, and the indexes of the training samples are stored in BreastTrainIndex.txt. The numbers of training and test samples are 17 and 5, respectively. The number of BRAC1, BRAC2 and Sporadic samples in the training and test subsets are 6, 6, 5, and 2, 2, 1, respectively. The original data set is available at `http://research.nhgri.nih.gov/microarray/NEJM_Supplement/`.

Brain cancer data (BrainPre.txt): It is a microarray data file consisting of expressions values of 12625 genes across 50 samples. It is a binary classification problem. This data file is provided as an example file for data preprocessing. The original data file is available at `http://www-genome.wi.mit.edu/cancer/pub/glioma`.

A.9 Execution of the software

The EGPC software is implemented in Java programming language and available as a jar (Java Archive) executable file. Three jar files: EGPCpre.jar, EGPC-com.jar, and EGPCgui.jar are available with this software bundle. EGPCpre.jar is for preprocessing of microarray data in CLI mode; EGPCcom.jar and EGPCgui.jar are for data classification and important features' identification in CLI and GUI modes, respectively. EGPCgui.jar has also interfaces for data preprocessing in GUI mode.

Given an input file for preprocessing, EGPCpre.jar will create two output files containing the preprocessed data and the cross reference indexes of the genes. If the name of the output file is not provided, the preprocessed data will be in the file "DataOut.txt"; the file containing cross reference indexes is "CrossRefIdx.txt".

EGPCcom.jar and EGPCgui.jar will create three files containing rules, accuracy and gene frequency information in the working directory (from where the software is run). If the name of a data file is Example.txt, EGPC software will create three files named as RulesExample.txt, AccuracyExample.txt, and GeneFreqExample.txt.

A.9.1 Execution of EGPCpre.jar in CLI mode

To run the program from command prompt, type:

```
java [-Xmx<heap size>] -jar EGPCpre.jar [arguments...]
```

Command line arguments and formats:

- -Xmx<heap size>: maximum heap size; some data sets may require higher heap size. Example: -Xmx512m (m or M for mega byte).
- -f <input file>: input data file name (with path if not on the current working directory); <input file> must be provided.
- -o <output file>: output file name (with path if not on the current working directory); default: DataOut.txt.
- -p <l:h:d:f>: preprocessing parameters; l=lower threshold, h-higher threshold, d=difference, f=fold change.

- -n <normalization info>: normalization info; for log normalization type G with the base like G10 or Ge while for linear normalization type La:b where a:b is the range.
- -h <header info>: header info; G: first column contains genes IDs; S: first row contains samples IDs; GS or SG for both.

Example: java -jar EGPCpre.jar -f "DataFile/BrainPre.txt" -o BrainPro.txt -p 20:16000:100:3 -n Ge -h GS

A.9.2 Execution of EGPCcom.jar in CLI mode:

To run the program from command prompt, type:

```
java [-Xmx<heapsize>] -jar MVGPCcom.jar [arguments...]
```

Command line arguments and formats:

- -Xmx<heap size>: maximum heap size; some data sets may require higher heap size. Example: -Xmx512m (m or M for mega byte).
- -u: UCIML format; default (if it is omitted) is Microarray format.
- -d <data file>: data file name (with path if not on the current working directory); <datafile> must be provided.
- -v <validation file>: validation file name (with path if not on the current working directory); if it is not provided, the training information must be provide under the '-t' below.
- -s <sample size>: number of samples; must be provided.
- -a <attribute size>: number of attributes; must be provided.
- -A <attribute info>: attribute information; default is that all attributes are numeric. See above "Attributes/Genes" for details about attribute information.

- -t <training info>: training subset information; the training information can be either the filename (with path if not on the current working directory) containing the indexes of the training samples or the training size of each type of sample delimited by colon like 179:106.
- -c <classes>: number of classes; default is 2.
- -m <ensemble size>: ensemble size; default is 3.
- -F <functions>: functions to be used; functions are delimited by colon (:) and the default functions are "+:-:/:*:sqr:sqrt". Note here that the functions' string must be within double quotation (" ").
- -p <population size>: population size; default is 1000.
- -g <max gen>: maximum number of generations; default is 50.
- -r <max run>: number of trials or repetition; default is 20.

A.9.3 Execution of example files from command prompt

- Monk problem:
  java -jar MVGPCcom.jar -u -d "DataFile/Monk.txt" -v "DataFile/MonkValid.txt" -s 556 -a 6 -A L1:6:4 -F ">:<:=:<>:>=:<=:AND:OR:NOT" -m 3 -c 2 -r 3
- Wisconsin Breast cancer (WCBreast.txt):
  java -jar MVGPCcom.jar -u -d "DataFile/WCBreast.txt" -s 569 -t 179:106 -a 30 -A N1:30 -F "+:-:*:/:SQRT:SQR:AND:OR:NOT:>:<:=:<>:>=:<=" -m 3 -c 2 -r 3
- Breast cancer (BreastCancer.txt):
  java -jar MVGPCcom.jar -d "DataFile/BreastCancer.txt" -s 22 -a 4434 -t "DataFile/BreastTrainIndex.txt" -A N1:4434 -F "+:-:/:*:SQRT:SQR" -m 3 -c 3 -r 3

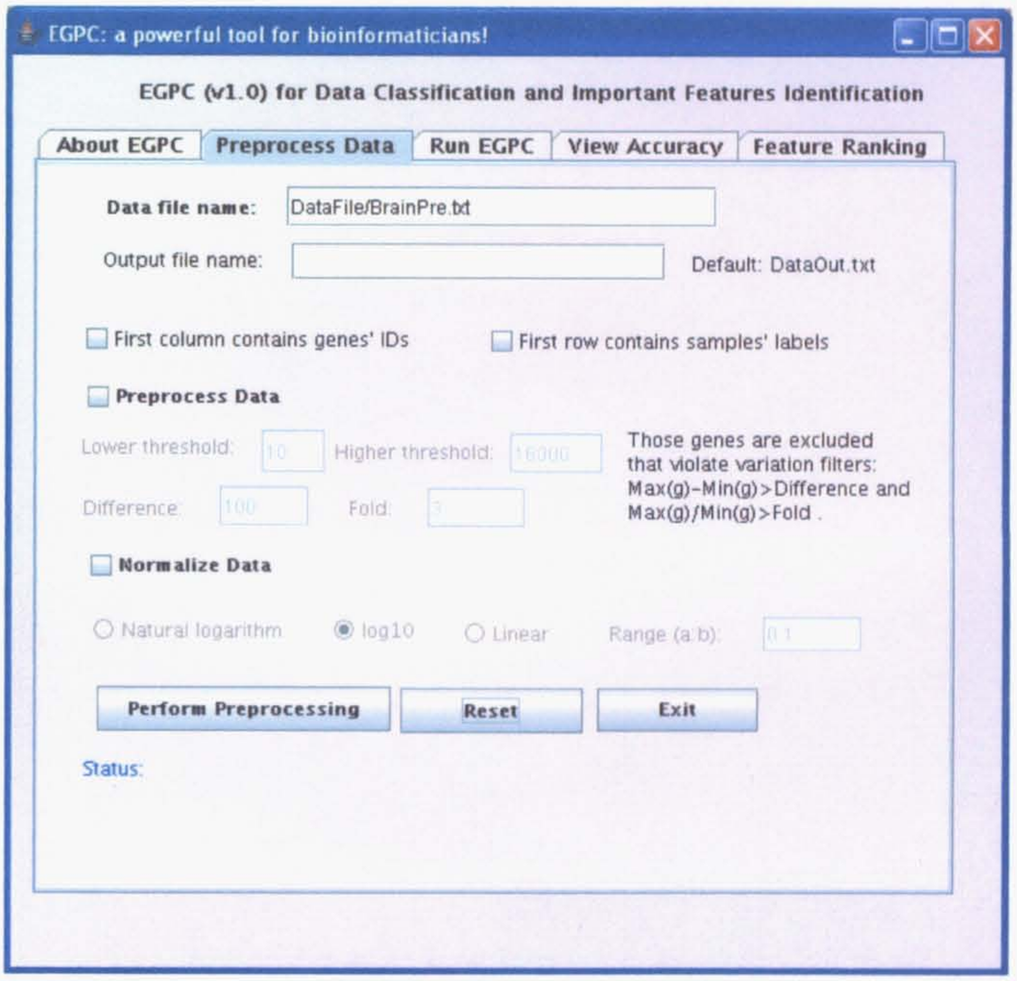
A.9.4 Execution of EGPCgui.jar in GUI mode:

Go to the command prompt and type:

```
java [-Xmx<heapsize>] -jar MVGPCgui.jar
```

Snapshots of the software in GUI view are shown in Figs. A.1–A.4. In the GUI view, the first page tells about the software. *Preprocess Data* is for preprocessing of microarray gene expression data. The data filename and values of different parameters are entered from *Run EGPC* page. While EGPC is being executed on a data file, the number of correct predictions and the accuracies by single rules or sets of rules as well as those by the EGPC can be viewed on *View Accuracy* page. The more frequently selected genes are displayed in the page *Feature Ranking*. Here also three output files for rules, accuracy and gene frequency are created.

The previously mentioned three examples can also be run in graphical user interface mode.


EGPC: a powerful tool for bioinformaticians!

EGPC (v1.0) for Data Classification and Important Features Identification

Data file name: DataFile/BrainPre.txt

Output file name: Default: DataOut.txt

☐ First column contains genes' IDs ☐ First row contains samples' labels

☐ Preprocess Data

Lower threshold: 10 Higher threshold: 16000

Difference: 100 Fold: 3

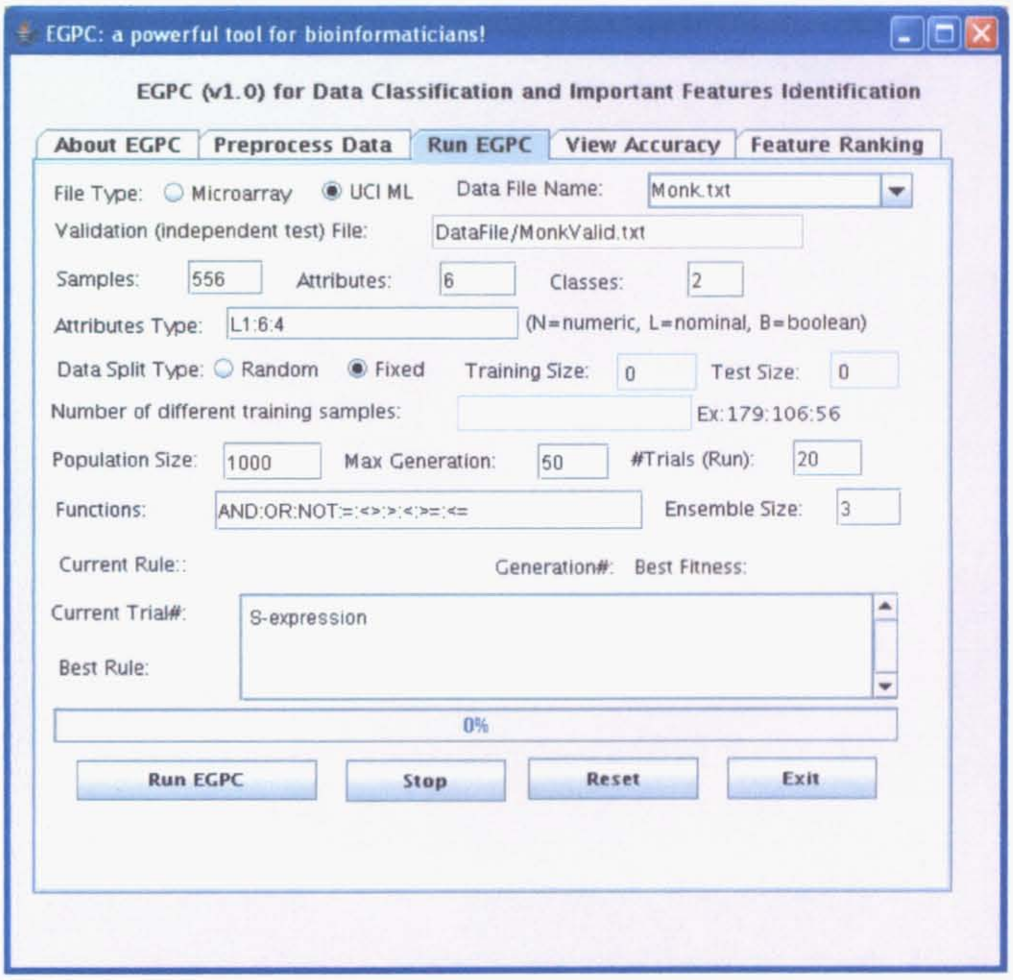
Those genes are excluded that violate variation filters: Max(g)-Min(g)>Difference and Max(g)/Min(g)>Fold .

☐ Normalize Data

○ Natural logarithm ● log10 ○ Linear Range (a,b): 0,1

Status:

Figure A.1: A screen shot of GUI of EGPC (*Preprocess Data* page)


EGPC: a powerful tool for bioinformaticians!
EGPC (v1.0) for Data Classification and Important Features Identification
About EGPC
Preprocess Data
Run EGPC
View Accuracy
Feature Ranking
Run EGPC
Stop
Reset
Exit

File Type: Microarray [✓ UCI ML] Data File Name: Monk.txt

Validation (independent test) File: DataFile/MonkValid.txt

Samples: 556 Attributes: 6 Classes: 2

Attributes Type: L1:6:4 (N=numeric, L=nominal, B=boolean)

Data Split Type: Random [✓ Fixed] Training Size: 0 Test Size: 0

Number of different training samples: Ex:179:106:56

Population Size: 1000 Max Generation: 50 #Trials (Run): 20

Functions: AND:OR:NOT:=:<>:>:<:>=:<= Ensemble Size: 3

Current Rule::

Generation#: Best Fitness:

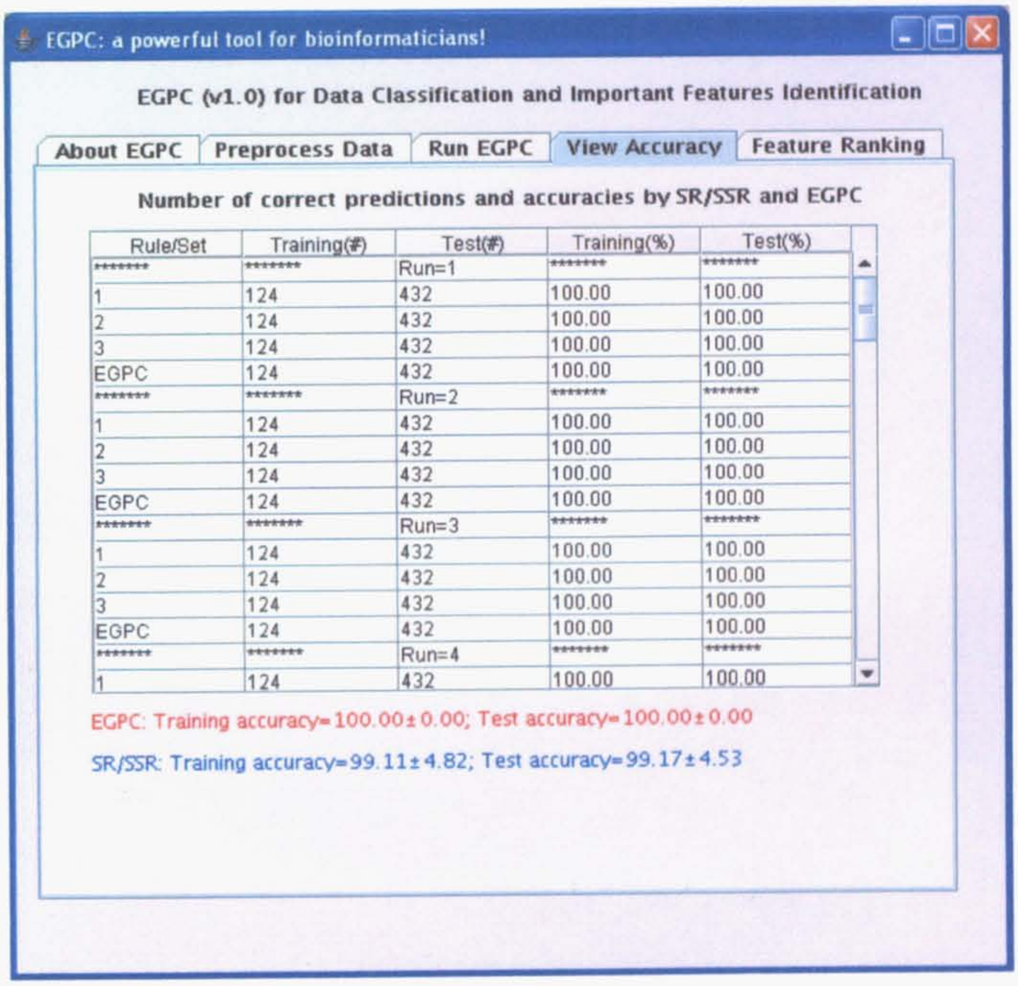
Current Trial#:

Best Rule:

S-expression

0%

Figure A.2: A screen shot of GUI of EGPC (*Run EGPC* page)



EGPC (v1.0) for Data Classification and Important Features Identification

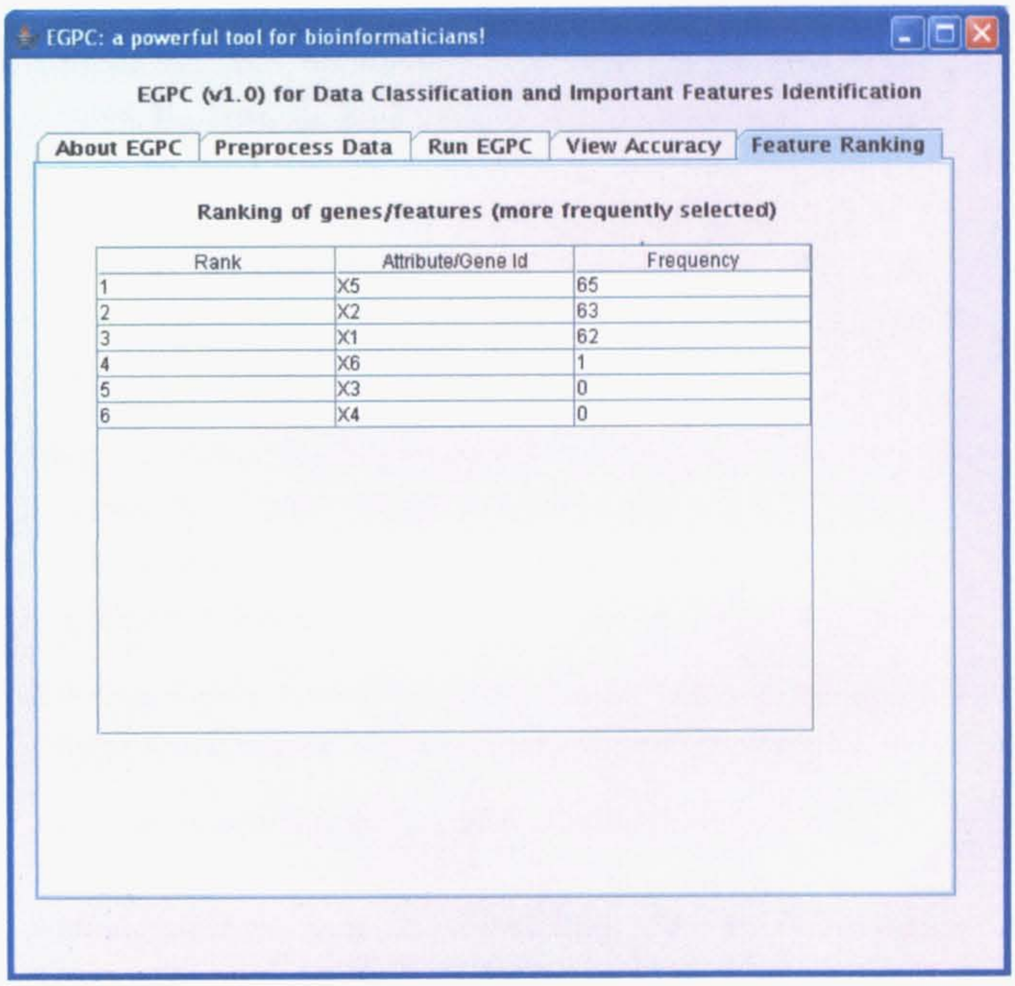
Number of correct predictions and accuracies by SR/SSR and EGPC

Rule/Set	Training(#)	Test(#)	Training(%)	Test(%)
********	********	Run=1	********	********
1	124	432	100.00	100.00
2	124	432	100.00	100.00
3	124	432	100.00	100.00
EGPC	124	432	100.00	100.00
********	********	Run=2	********	********
1	124	432	100.00	100.00
2	124	432	100.00	100.00
3	124	432	100.00	100.00
EGPC	124	432	100.00	100.00
********	********	Run=3	********	********
1	124	432	100.00	100.00
2	124	432	100.00	100.00
3	124	432	100.00	100.00
EGPC	124	432	100.00	100.00
********	********	Run=4	********	********
1	124	432	100.00	100.00

EGPC: Training accuracy=100.00±0.00; Test accuracy=100.00±0.00

SR/SSR: Training accuracy=99.11±4.82; Test accuracy=99.17±4.53

Figure A.3: A screen shot of GUI of EGPC (*View Accuracy* page)


EGPC: a powerful tool for bioinformaticians!
EGPC (v1.0) for Data Classification and Important Features Identification
About EGPC
Preprocess Data
Run EGPC
View Accuracy
Feature Ranking

Ranking of genes/features (more frequently selected)

Rank	Attribute/Gene Id	Frequency
1	X5	65
2	X2	63
3	X1	62
4	X6	1
5	X3	0
6	X4	0

Figure A.4: A screen shot of GUI of EGPC (*Feature Ranking* page)

Bibliography

Aha, D. W., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, **6**, 37–66.

Alizadeh, A., Eisen, M., Davis, R., Ma, C., Lossos, I., Rosenwald, A., Boldrick, J., Sabet, H., Tran, T., Yu, X., Powell, J., Yang, L., Marti, G., Moore, T., Hudson, J. J., Lu, L., Lewis, D., Tibshirani, R., Sherlock, G., Chan, W., Greiner, T., Weisenburger, D., Armitage, J., Warnke, R., Levy, R., Wilson, W., Grever, M., Byrd, J., Botstein, D., Brown, P., and Staudt, L. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, **403**(6781), 503–511.

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of National Academy of Science, Cell Biology*, **96**, 6745–6750.

American Cancer Society, Inc (2005). Cancer facts and figures 2005. Available at `http://www.cancer.org/downloads/STT/CAFF2005f4PWSecured.pdf`.

Anderson, D. and McNeil, G. (1992). Artificial neural networks technology. Technical Report F30602-89-C-0082, Data & Analysis Center for Software, 775 Daedalian Drive, Rome, NY 13441-4909. URL: `http://www.dacs.dtic.mil/techs/neural/neural.title.html`.

Ando, S. and Iba, H. (2004). Classification of gene expression profile using combinatory method of evolutionary computation and machine learning. *Genetic Programming and Evolvable Machines*, **5**, 145–156.

Au, N., Gown, A., Cheang, M., Huntsman, D., Yorida, E., Elliott, W. M., Flint, J., English, J., Gilks, C., and Grimes, H. (2004). p63 expression in lung carcinoma: A tissue microarray study of 408 cases. *Applied Immunohistochemistry & Molecular Morphology*, **12**(3), 240–247.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Banerjee, A., Liu, J., Yuan, Y., Gopalakrishnan, V., SL, J., Dinda, A., Gupta, N., Trevino, L., and Vishwanatha, J. (2003). Expression of biomarkers modulating prostate cancer angiogenesis: differential expression of annexin ii in prostate carcinomas from india and usa. *Molecular Cancer*, **2**, 34.

Banzhaf, W., Nordin, P., Keller, R., and Francone, F. (1998). *Genetic Programming–An Introduction*. Morgan Kaufmann, San Francisco.

Ben-Dor, A., Shamir, R., and Yakhini, Z. (1999). Clustering gene expression patterns. *Journal of Computational Biology*, **6**, 281–297.

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

Bhattacharjee, A., Richards, W., Stauton, J., Li, C., Monti, S., Vasa, P., Ladd, C., Behesti, J., Buneo, R., Gillete, M., Loda, M., Weber, G., Mark, E., Lander, E., Wong, W., Johnson, B., Golub, T., Sugarbaker, D., and Meyerson, M. (2001). Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. In *Proceedings of National Academy of Science*, volume 98, pages 13790–13795.

Boser, B., Guyon, I., and Vapnik, V. (1992). An training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh. ACM.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chien, B. C., Lin, J., and Hong, T. P. (2002). Learning discriminant functions with fuzzy attributes for classification using genetic programming. *Expert Syst. Applicat.*, **23**, 31–37.

Dallol, A., Silva, N. D., Viacava, P., Minna, J., Bieche, I., Maher, E., and Latif, F. (2002). Slit2, a human homologue of the drosophila slit2 gene, has tumor suppressor activity and is frequently inactivated in lung and breast cancers. *Cancer Research*, **62**(20), 5874–5880.

Dasarathy, B. (1991). *Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press.

Deb, K. and Reddy, A. R. (2003). Reliable classification of two-class cancer data using evolutionary algorithms. *BioSystems*, **72**, 111–129.

Deutsch, J. M. (2003). Evolutionary algorithms for finding optimal gene sets in microarray prediction. *Bioinformatics*, **19**(1), 45–52.

Devoogdt, N., Revets, H., Ghassabeh, G., and Baetselier, P. D. (2004). Secretory leukocyte protease inhibitor in cancer development. *Annals of the NY Academy of Sciences*, **1028**, 380–389.

Ding, C. (2000). Tumor tissue classification using support vector machines and K-nearest neighbor method. In *Proceedings of the first Conference on Critical Assessment of Microarray Data Analysis*.

Ding, C. H. Q. (2003). Unsupervised feature selection via two-way ordering in gene expression analysis. *Bioinformatics*, **19**(10), 1259–1266.

Driscoll, J. A., Worzel, B., and MacLean, D. (2003). Classification of gene expression data with genetic programming. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practice*, chapter 3, pages 25–42. Kluwer.

Dudoit, S., Fridlyand, J., and Speed, T. P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, **97**(457), 77–87.

Eckman, E., Watson, M., Marlow, L., Sambamurti, K., and Eckman, C. B. (2003). Alzheimer's disease beta-amyloid peptide is increased in mice deficient in endothelin-converting enzyme. *J. Biol. Chem.*, **278**(4), 2081–2084.

Efimova, E., Al-Zoubi, A., Martinez, O., Kaithamana, S., Lu, S., Arima, T., and Prabhakar, B. (2004). Ig20, in contrast to denn-sv, (madd splice variants) suppresses tumor cell survival, and enhances their susceptibility to apoptosis and cancer drugs. *Oncogene*, **23**(5), 1076–87.

Eisen, M. B., Spellman, P. T., Brown, P., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, **95**, 14863–14868.

Elkins, D., Yokomizo, A., Thibodeau, S., Schaid, D., Cunningham, J., Marks, A., Christensen, E., McDonnell, S., Slager, S., Peterson, B., Jacobsen, S., Cerhan, J., Blute, M., Tindall, D., and Liu, W. (2003). Luteinizing hormone beta polymorphism and risk of familial and sporadic prostate cancer. *The Prostate*, **56**(1), 30–36.

Entrez Gene (2006). URL: `http://www.ncbi.nlm.nih.gov/entrez/query.fcgi`.

Eshelman, L. (1991). The CHC adaptive search algorithm. In *Foundations of Genetic Algorithms I*, pages 265–283. Morgan Kauffman, San Mateo CA.

Falco, I. D., Cioppa, A. D., and Tarantino, E. (2002). Discovering interesting classification rules with genetic programming. *Applied Soft Computing*, **1**, 257–269.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1), 119–139.

Furey, T., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., and Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, **16**(10), 906–914.

Geiger, D. and Heckerman, D. (1994). Learning Gaussian networks. Technical report, Microsoft Advanced Technology Division, Microsoft Corporation, Seattle, Washington.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., and Lander, E. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**(15), 531–537.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machine. *Machine Learning*, **46**(1-3), 389–422.

Hanley, J. A. and Mcneil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic curve. *Radiology*, **143**(1), 29–36.

Hartl, D. L. and Jones, E. W. (2005). *Genetics: Analysis of Genes and Genomes*, chapter 1, pages 16–21. Jones and Bartlett Publishers, Sudbury, Massachusetts, sixth edition.

Hatziapostolou, M., Delbe, J., Katsoris, P., Polytarchou, C., Courty, J., and Papadimitriou, E. (2005). Heparin affin regulatory peptide is a key player in prostate cancer cell growth and angiogenicity. *The Prostate*, **65**(2), 151–158.

Hedenfalk, I., Duggan, D., Chen, Y., Radmacher, M., Bittner, M., Simon, R., Meltzer, P., Gusterson, B., Esteller, M., Kallioniemi, O., Wilfond, B., Borg, A., and Trent, J. (2001). Gene-expression profiles in hereditary breast cancer. *The New England Journal of Medicine*, **344**(8), 539–548.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan.

Hong, J.-H. and Cho, S. B. (2004). Lymphoma cancer classification using genetic programming with SNR features. In M. Keijzer, U.-M. O'Reilly, S. M. Lucas, E. Costa, and T. Soule, editors, *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, volume 3003 of *LNCS*, pages 78–88, Coimbra, Portugal. Springer-Verlag.

Hwang, K.-B., Cho, D.-Y., Park, S.-W., Kim, S.-D., and Zhang, B.-T. (2002). Applying machine learning techniques to analysis of gene expression data: Cancer diagnosis. In *Methods of Microarray Data Analysis*, chapter 12, pages 167–182. Kluwer Academic Publishers.

Inza, I., Sierra, B., Blanco, R., and Larrañaga, P. (2002). Gene selection by sequential wrapper approaches in microarray cancer class prediction. *Journal of Intelligent and Fuzzy Systems*, **12**(1), 25–33.

Jaakkola, S., Salmikangas, P., Nylund, S., Partanen, J., Armstrong, E., Pyrhonen, S., Lehtovirta, P., and Nevanlinna, H. (1993). Amplification of fgfr4 gene in human breast and gynecological cancers. *International Journal of Cancer*, **54**(3), 378–382.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, **32**(3), 241–254.

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer.

Keller, A., Schummer, M., Hood, L., and Ruzzo, W. L. (2000a). Bayesian classification of dna array expression data. Technical Report UW-CSE-2000-08-01, Department of Computer Science and Engineering, University of Washington, Seattle.

Keller, A., Schummer, M., Hood, L., and Ruzzo, W. (2000b). Bayesian classification of DNA array expression data. Technical Report UW-CSE-2000-08-01, Department of Computer Science and Engineering, University of Washington.

Khan, J., Wei, J., Ringer, M., Saal, L., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C., Peterson, C., and Meltzer, P. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, **7**(6), 673–679.

Kim, Y.-H., Lee, S.-Y., and Moon, B.-R. (2004). A genetic approach for gene selection on microarray expression data. In *Proccedings of the Genetic and Evolutionary Computation Conference(GECCO2004)*, pages 346–355.

Kirchhofer, D., Peek, M., Lipari, M., Billeci, K., Fan, B., and Moran, P. (2005). Hepsin activates pro-hepatocyte growth factor and is inhibited by hepatocyte growth factor activator inhibitor-1b (hai-1b) and hai-2. *FEBS Lett*, **579**(9), 1945–50.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, **97**(1-2), 273–324.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, **78**(9), 1464–1480.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press.

Kreßel, U. (1999). Pairwise classification and support vector machine. In *Advances in Kernel methods–Support Vector Machine*, pages 255–268. MIT Press, Cambridge,MA.

Kumar, D., Gokhale, P., Broustas, C., Chakravarty, D., Ahmad, I., and Kasid, U. (2004). Expression of scc-s2, an antiapoptotic molecule, correlates with enhanced proliferation and tumorigenicity of mda-mb 435 cells. *Oncogene*, **23**(2), 612–616.

Kuncheva, L. and Whitaker, C. (2003). Measures of diversity in classifier ensembles and their relationships with the ensemble accuracy. *Machine Learning*, **51**, 181–207.

Langdon, W. B. and Buxton, B. F. (2004). Genetic programming for mining DNA chip data from cancer patients. *Genetic Programming and Evolvable Machines*, **5**(3), 251–257.

Larrañaga, P. and Lozano, J. (2001). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Optimization*. Kluwer Academic Publishers, Boston, USA.

Li, G.-Z., Yang, J., Ye, C.-Z., and Geng, D.-Y. (2006). Degree prediction of malignancy in brain glioma using support vector machines. *Computers in Biology and Medicine*, **36**, 313–325.

Li, L., Umbach, D. M., Terry, P., and Taylor, J. A. (2004). Application of the GA/KNN method to SELDI proteomics data. *Bioinformatics*, **20**(10), 1638–1640.

Liu, J. and Iba, H. (2001). Selecting informative genes with parallel genetic algorithms in tissue classification. In *Genome Informatics*, pages 14–23.

Liu, J. and Iba, H. (2002). Selecting informative genes using a multiobjective evolutionary algorithm. In *Proceedings of the 2002 IEEE World Congress on Computational Intelligence(WCCI-2002)*, pages 297–302.

Liu, J. J., Cutler, G., Li, W., Pan, Z., Peng, S., Hoey, T., Chen, L., and Ling, X. B. (2005). Multiclass cancer classification and biomarker discovery using ga-based algorithms. *Bioinformatics*, **21**(11), 2691–2697.

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochemica et Biophysica Acta.*, **405**, 442–451.

Menez, J., Chansac, B. L. M., Dorothee, G., Vergnon, I., Jalil, A., Carlier, M., Chouaib, S., and Mami-Chouaib, F. (2004). Mutant alpha-actinin-4 promotes tumorigenicity and regulates cell motility of a human lung carcinoma. *Oncogene*, **23**(15), 2630–9.

Mitra, A. P., Almal, A. A., George, B., Fry, D. W., Lenehan, P. F., Pagliarulo, V., Cote, R. J., Datar, R. H., and Worzel, W. P. (2006). The use of genetic programming in the analysis of quantitative gene expression profiles for identification of nodal status in bladder cancer. *BMC Cancer*, **6**(159).

Mitsuta, K., Yokoyama, A., Kondo, K., Nakajima, M., Arita, K., and Kohno, N. (2005). Polymorphism of the muc1 mucin gene is associated with susceptibility to lung adenocarcinoma and poor prognosis. *Oncology Reports*, **14**(1), 185–189.

Mocharla, H., Mocharla, R., and Hodes, M. E. (1990). Coupled reverse transcription-polymerase chain reaction (RT-PCR) as a sensitive and rapid method for isozyme genotyping. *Gene*, **93**, 271–275.

Monti, S., Tamayo, P., Mesirov, J., and Golub, T. (2003). Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning Journal*, **52**(1-2), 91–118.

Moore, J. H., Parker, J. S., Olsen, N. J., and Aune, T. M. (2002). Symbolic discriminant analysis of microarray data in automimmune disease. *Genetic Epidemiology*, **23**, 57–69.

Mühlenbein, H. and Paaß, G. (1996). From recombination of genes to the estimation of distribution I. Binary parameters. In *Parallel Problem Solving from Nature-PPSN IV*, Lecture Notes in Computer Science (LNCS) 1411, pages 178–187. Springer-Verlag, Berlin, Germany.

Muni, D. P., Pal, N. R., and Das, J. (2004). A novel approach to design classifiers using genetic programming. *IEEE Transaction on Evolutionary Computation*, **8**(2), 183–196.

Nakanishi, H., Suda, T., Katoh, M., Watanabe, A., Igishi, T., Kodani, M., Matsumoto, S., Nakamoto, M., Shigeoka, Y., Okabe, T., Oshimura, M., and

Shimizu, E. (2004). Loss of imprinting of peg1/mest in lung cancer cell lines. *Oncology Reports*, **12**(6), 1273–1278.

National Cancer Center (2006). Cancer statistics in japan. online.

Newman, D. J., Hettich, S., Blake, C. L., and Merz, C. J. (1998). UCI repository of machine learning databases.

Nutt, C., Mani, D., Betensky, R., Tamayo, P., Cairncross, J., Ladd, C., Pohl, U., Hartmann, C., McLaughlin, M., Batchelor, T. T., Black, P., von Deimling, A., Pomeroy, S., Golub, T., and Louis, D. (2003). Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Research*, **63**(7), 1602–1607.

Ohguro, H. and Nakazawa, M. (2002). Pathological roles of recoverin in cancer-associated retinopathy. *Advances in Experimental Medicine and Biology*, **514**, 109–124.

Onn, A., Correa, A. M., Gilcrease, M., Isobe, T., Massarelli, E., Bucana, C. D., O'Reilly, M. S., Hong, W. K., Fidler, I. J., Putnam, J. B., and Herbst, R. S. (2004). Synchronous overexpression of epidermal growth factor receptor and her2-neu protein is a predictor of poor outcome in patients with stage i non-small cell lung cancer. *Clinical Cancer Research*, **10**, 136–143.

Ooi, C. H. and Tan, P. (2003). Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. *Bioinformatics*, **19**(1), 37–44.

Pan, F., Wang, B., Hu, X., and Perrizo, W. (2004). Comprehensive vertical sample-based knn/lsvm classification for gene expression analysis. *Journal of Biomedical Informatics*, **37**(4), 240–248.

Park, P., Pagano, M., and Bonnetti, M. (2001). A nonparametric scoring algorithm for identifying informative genes from microarray data. *Pacific Symposium on Bioinformatics(PSB)*, **6**, 30–41.

Paul, T. K. and Iba, H. (2003a). Linear and combinatorial optimizations by estimation of distribution algorithms. In *Proceedings of the 9th MPS Symposium on Evolutionary Computation*, pages 99–106. IPSJ. Article available at `http://www.iba.k.u-tokyo.ac.jp/english/EDA.htm`.

Paul, T. K. and Iba, H. (2003b). Reinforcement learning estimation of distribution algorithm. In *GECCO '03: Proceedings of the 2003 conference on Genetic and evolutionary computation*, Lecture Notes in Computer Science (LNCS) 2724, pages 1259–1270. Springer-Verlag.

Paul, T. K. and Iba, H. (2004a). Identification of informative genes for molecular classification using probabilistic model building genetic algorithm. In *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO2004)*, number 3102 in Lecture Notes in Computer Science, LNCS, pages 414–425. Springer-Verlag.

Paul, T. K. and Iba, H. (2004b). Selection of the most useful subset of genes for gene expression-based classification. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, pages 2076–2083, Portland, Oregon, USA.

Pavlidis, P., Weston, J., Cai, J., and Grundy, W. N. (2001). Gene functional classification from heterogeneous data. In *RECOMB*, pages 249–255.

Pelikan, M., Goldberg, D., and Lobo, F. (1999). A survey of optimizations by building and using probabilistic models. Technical Report, Illigal Report 99018, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, USA.

Pomeroy, S., Tamayo, P., Gaasenbeek, M., Sturla, L., Angelo, M., McLaughlin, M., Kim, J., Goumnerovak, L., Black, P., Lau, C., Allen, J., Zagzag, D., Oslon, J., Curran, T., Wetmore, C., Biegel, J., Poggio, T., Mukherjee, S., Rifkin, R., Califanok, A., Stolovitzkyk, G., Louis, D., Mesirov, J., Lander, E., and Golub,

T. (2002). Prediction of central nervous system ebryonal tumor outcome based on gene expression. *Nature*, **415**, 436–442.

Quinlan, J. (1993). *C4.5:Programs for Machine Learning*. Morgan Kaufman Publishers.

Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C.-H., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J., Poggio, T., Gerald, W., Loda, M., Lander, E., and Golub, T. (2001). Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Science*, **98**(26), 15149–15154.

Reece, R. J. (2005). *Analysis of Genes and Genomes*, chapter 10, pages 314–324. John Wiley and Sons, Ltd.

Rowland, J. (2003). Generalization and model selection in supervised learning with evolutionary computation. In *EvoWorkshops 2003, LNCS 2611*, pages 119–130. Springer.

Sandrini, F., Matyakhina, L., Sarlis, N., Kirschner, L., Farmakidis, C., Gimm, O., and Stratakis, C. (2002). Regulatory subunit type i-alpha of protein kinase a (prkar1a): a tumor-suppressor gene for sporadic thyroid cancer. *Genes Chromosomes Cancer*, **35**(2), 182–92.

Schapire, R. (1999). A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Article available at `http://www.boosting.org/`.

Schena, M. (2000). *DNA Microarrays*. Oxford University Press, New York, USA.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, **7**(2), 461–464.

Shalon, D., Smith, S., and Brown, P. (1996). A DNA microarray system for analyzing complex dna samples using two color fluorescent probe hybridization. *Genome Research*, **6**, 639–645.

Shen, L. and Tan, E. C. (2006). A generalized output-coding scheme with svm for multiclass microarray classification. In *Proceedings of 4th Asia-Pacific Bioinformatics Conference*, pages 179–186.

Singh, D., Febbo, P., Ross, K., Jackson, D., Manola, J., Ladd, C., Tamayo, P., Renshaw, A., D'Amico, A., Richie, J., Lander, E., Loda, M., Kantoff, P., Golub, T., and Sellers, W. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*.

Sloan, E., Stanley, K., and Anderson, R. (2004). Caveolin-1 inhibits breast cancer growth and metastasis. *Oncogene*, **23**(47), 7893–7897.

Slonim, D., Tamayo, P., Mesirov, J., Golub, T., and Lander, E. (2000). Class prediction and discovery using gene expression data. In *Proceedings of the fourth annual international conference on Computational molecular biology*, pages 263–272, Tokyo, Japan.

Smith, K. A. and Ng, A. (2003). Web page clustering using a self-organizing map of user navigation patterns. *Decision Support Systems*, **35**, 245–256.

Swift, S., Tucker, A., Vinciotti, V., Martin, N., Orengo, C., Liu, X., and Kellam, P. (2004). Consensus clustering and functional interpretation of gene-expression data. *Genome Biology*, **5**(11), R94.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S., and Golub, T. R. (1999). Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci.*, **96**, 2907–2912.

Tan, K. C., Tay, A., Lee, T. H., and Heng, C. M. (2002). Mining multiple comprehensible classification rules using genetic programming. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1302–1307.

Thomas, P. (1980). Hybridization of denatured RNA and small DNA fragments transferred to nitrocellulose. *Proc. Natl. Acad. Sci.*, **77**, 5201–5205.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York,USA.

Wang, H., Wang, H., Shen, W., Huang, H., Hu, L., Ramdas, L., Zhou, Y., Liao, W., Fuller, G., and Zhang, W. (2003). Insulin-like growth factor binding protein 2 enhances glioblastoma invasion by activating invasion-enhancing genes. *Cancer Research*, **63**(15), 4315–4321.

Whitfield, M. L., Finlay, D. R., Murray, J. I., Troyanskaya, O. G., Chi, J.-T., Pergamenschikov, A., McCalmont, T. H., Brown, P. O., Botstein, D., and Connolly, M. K. (2003). Systemic and cell type-specific gene expression patterns in scleroderma skin. *PNAS*, **100**(21), 12319–12324.

Publication List

Journal Paper:

Paul, T. K. and Iba, H. (2005). Gene selection for classification of cancers using probabilistic model building genetic algorithm. *BioSystems*, **82**(3), 208–225.

Book Chapter:

Paul, T. K. and Iba, H. (2006). Genetic programming for classifying cancer data and controlling humanoid robots. In R. L. Riolo, T. Soule, and B. Worzel, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 10. Springer, Ann Arbor.

Conference Papers (Peer Reviewed):

1. Paul, T. K. and Iba, H. (2006). Classification of scleroderma and normal biopsy data and identification of possible biomarkers of the disease. In *Proceedings of IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology 2006 (CIBCB2006)*, pages 306–311, Toronto, Ontario, Canada. IEEE Computational Intelligence Society.

2. Paul, T. K. and Iba, H. (2006). Identification of weak motifs in multiple biological sequences using genetic algorithm. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 271–278, New York, NY, USA. ACM Press.

3. Paul, T. K., Hasegawa, Y., and Iba, H. (2006). Classification of gene expression data by majority voting genetic programming classifier. In *Proceedings of the 2006 IEEE World Congress on Computational Intelligence*, pages 8690–8697, Vancouver, BC, Canada.

4. Paul, T. K. and Iba, H. (2005). Extraction of informative genes from microarray data. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 453–460, New York, NY, USA. ACM Press.

5. Paul, T. K. and Iba, H. (2004). Identification of informative genes for molecular classification using probabilistic model building genetic algorithm.

In *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO2004)*, number 3102 in Lecture Notes in Computer Science, LNCS, pages 414–425. Springer-Verlag.

6. Paul, T. K. and Iba, H. (2004). Selection of the most useful subset of genes for gene expression-based classification. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, pages 2076–2083, Portland, Oregon, USA.

7. Paul, T. K. and Iba, H. (2003). Optimization in continuous domain by real-coded estimation of distribution algorithm. In *Design and application of hybrid intelligent systems*, pages 262–271. IOS Press, Amsterdam, The Netherlands.

8. Paul, T. K. and Iba, H. (2003). Reinforcement learning estimation of distribution algorithm. In *GECCO '03: Proceedings of the 2003 conference on Genetic and evolutionary computation*, Lecture Notes in Computer Science (LNCS) 2724, pages 1259–1270. Springer-Verlag.

9. Paul, T. K. and Iba, H. (2003). Real-coded estimation of distribution algorithm. In *Proceedings of the Fifth Metaheuristics International Conference (MIC2003)*, Kyoto, Japan.

10. Paul, T. K. and Iba, H. (2003). Linear and combinatorial optimizations by estimation of distribution algorithms. In *Proceedings of the 9th MPS Symposium on Evolutionary Computation*, pages 99–106. IPSJ.

Poster in International Conference

Paul, T. K. and Iba, H. (2005). Classification of cancer data with genetic programming. *Genome Informatics*, **16**(2), PACIFICO YOKOHAMA, Japan.

Other Article

Paul, T. K. and Iba, H. (2003). From genetic algorithm to artificial life. *Computer Today*, **114**, March, 2006. (in Japanese)