Finding Rule Groups to Classify High Dimensional Gene Expression Datasets

Finding Rule Groups to Classify High Dimensional Gene Expression Datasets

Jiyuan An[1] and Yi-Ping Phoebe Chen[1,2]
[1]*Faculty of Science and Technology, Deakin University*
[2]*Australian Research Council Centre in Bioinformatics*
{jiyuan, phoebe}@deakin.edu.au

Abstract

Microarray data provides quantitative information about the transcription profile of cells. To analyze microarray datasets, methodology of machine learning has increasingly attracted bioinformatics researchers. Some approaches of machine learning are widely used to classify and mine biological datasets. However, many gene expression datasets are extremely high dimensionality, traditional machine learning methods can not be applied effectively and efficiently. This paper proposes a robust algorithm to find out rule groups to classify gene expression datasets. Unlike the most classification algorithms, which select dimensions (genes) heuristically to form rules groups to identify classes such as cancerous and normal tissues, our algorithm guarantees finding out best-k dimensions (genes), which are most discriminative to classify samples in different classes, to form rule groups for the classification of expression datasets. Our experiments show that the rule groups obtained by our algorithm have higher accuracy than that of other classification approaches.

1. Introduction

Mining gene expression datasets has generated interest among many bioinformatics researchers. One of the important trends in bioinformatics is identification of genes or groups of gene to differentiate diseased tissues from normal tissues. Classification of tissues into cancerous and normal tissues using the identified genes, is one of the key problems being faced in bioinformatics. Gene expression data is usually represented as a matrix; each element in the matrix represents an appearance level of a particular gene under a particular condition.

We assume that a gene expression matrix has n rows and m columns. The rows represent samples that are divided into different classes such as cancerous tissue and normal tissue. The columns represent genes whose number is usually more than several thousands. The number of rows is much lower than that of columns as the sample used ranges from ten to several hundreds. To cope with this kind of extremely high dimensional data, traditional machine learning techniques such as decision tree and support virtual machine, can not classify effectively as they use heuristics to select significant dimensions (genes); many discriminative dimensions can be left out.

In this paper, we propose a classification method that generates rule groups to categorize samples. A rule is a conjunction of several dimensions (genes); each gene is constrained into one interval. For example, $(gene1 > 120.5) \wedge (gene2 \leq 20.3)$ is one such rule. If a sample satisfies the conjunction of a rule, it will be covered by the rule. The above rule covers samples whose expression values of *gene1* are larger than 120.5 and expression values of *gene2* are smaller than or equal to 20.3. In contrast to traditional machine learning algorithms that use heuristics our method guarantees finding out best-k genes, which are most discriminative to classify samples in different classes, to form rule groups. The value of parameter k is set to around 5. It is based on the fact that each rule should not be too long from the principle of Occam's razor [4]; otherwise, the problem of overfitting arises [5].

2. Approach

A rule group is associated with a target class as different classes have different rule groups that reflect the common characters for the classes. The samples that belong to the target class are treated as ***positive samples***, and the samples that belong to other classes are treated as ***negative samples*** throughout this paper. For the sake of consistency, we treat ***dimensions*** as columns (or genes) in gene expression matrix.

Rule groups reveal biological relationship between cellular function and group genes. In this paper, a rule

Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)
0-7695-2521-0/06 $20.00 © 2006 IEEE

has the form: $LHS \Rightarrow C$, where C represents the consequence of the rule. It is a class label such as cancerous and normal tissues. *LHS* represents the condition of the rule. It is a conjunction of items; that is, intersection of different items. Each ***item*** in the conjunction is represented as (g, i), where g is a gene (dimension) number; i is an interval where the gene expression value of g belongs to. For example, interval (-∞, 123.5] includes all real number less than 123.5. The conjunction is formed from items to represent the condition of a rule; that is, $(g_1, i_1) \cap (g_2, i_2) \cap \ldots$, where the gene g_i and interval i_i appear as a pair. The item (g_i, i_i) means that gene expression level of g_i is the range of interval i_i. Figure 1 summarizes the terminologies that are used in rule group.

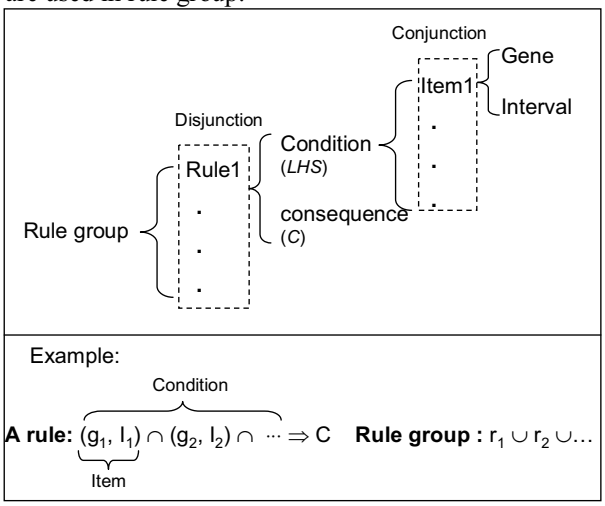
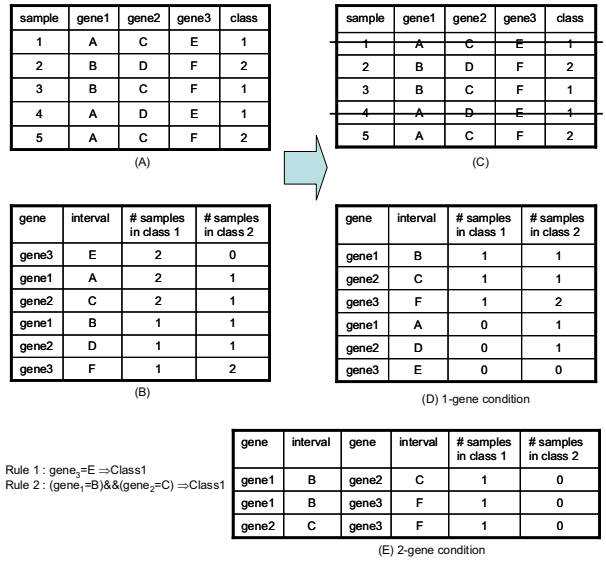


Figure 1 The relationship of notations for rule group.

A rule can be viewed as a subspace that covers the samples whose gene expression values satisfy the condition in the rule. In general, one rule can not cover all positive samples such as a cancerous tissue. So a rule group that consists of more than one rule, is needed to cover all samples of a target class. Many rule generation methods such as decision tree [5], SVM [4], and etc., have been proposed These methods select discriminative features heuristically to describe common characters of a specific class. As they can not effectively select high discriminative dimensions for high dimensional (features) datasets, a more robust algorithm has been proposed [1]. This method enumerates all possible combinations of genes and pruning power is used to remove unrelated combinations from exponentially increasing enumerated combinations. This method guarantees finding out a rule that can cover the largest number of samples of a specific class. But it is a costly process. In this paper, we propose a new robust algorithm that can deal with very high dimensional data effectively without any loss of accuracy. Section 2.1 gives an example to find a rule group for a specific class. Section 2.2 describes constraint to avoid overfitting.

2.1. An example to find rule group

According to the principal of Occam's razor, simplicity is an important criterion for evaluation of the generated rule groups: 1). the number of rules should be small. 2). the condition *LHS* should be short. From the first point, it is desirable to find subspaces that cover as more positive samples as possible. As a result, the number of rules in the rule group becomes smaller. The second point demands fewer dimensions (genes) to form the condition of a rule. Based on the two points, we propose an enumeration based algorithm to find rule groups for a specific class.



sample	gene1	gene2	gene3	class
1	A	C	E	1
2	B	D	F	2
3	B	C	F	1
4	A	D	E	1
5	A	C	F	2

(A)

sample	gene1	gene2	gene3	class
1	A	C	E	1
2	B	D	F	2
3	B	C	F	1
4	A	D	E	1
5	A	C	F	2

(C)

gene	interval	# samples in class 1	# samples in class 2
gene3	E	2	0
gene1	A	2	1
gene2	C	2	1
gene1	B	1	1
gene2	D	1	1
gene3	F	1	2

(B)

gene	interval	# samples in class 1	# samples in class 2
gene1	B	1	1
gene2	C	1	1
gene3	F	1	2
gene1	A	0	1
gene2	D	0	1
gene3	E	0	0

(D) 1-gene condition

Rule 1 : $gene_3$=E ⇒Class1
Rule 2 : ($gene_1$=B)&&($gene_2$=C) ⇒Class1

gene	interval	gene	interval	# samples in class 1	# samples in class 2
gene1	B	gene2	C	1	0
gene1	B	gene3	F	1	0
gene2	C	gene3	F	1	0

(E) 2-gene condition

Figure 2 An example to find the rule group to describe class 1.

Figure 2 shows an example to find the rule group to describe *class 1*. There are five samples out of which three are positive samples and two are negative samples. Different genes have different intervals: expression values of *gene1* have two intervals 'A' and 'B' as showing Figure 2 (A); expression values of $gene_2$ and $gene_3$ have 'C', 'D' and 'E', 'F' intervals, respectively. The original expression values are real numbers. Therefore, expression values have been discretized. It will be explained in the next section.

All items are enumerated to find out the rule group for *class1*,. The numbers of positive and negative samples covered by these items are calculated. The two numbers become primary and secondary keys to sort out these items as shown in Figure 2 (B). The rows of the table are sorted out in a **descending** order of the primary key and **ascending** order of the secondary key.

The first item ($gene_3$, 'E') becomes a rule, because it covers more positive samples than any other items, and it does not cover any negative samples. In this example, we assume that the confidence of rule is 100%; that is, every sample that satisfies the condition of a rule constantly belongs to the consequence (class) of the rule.

To find out more rules that cover the rest of positive samples, the samples covered by the first rule are removed as shown in Figure 2 (C): The samples 1 and 4 are then removed. This process is repeated until all positive samples are removed. Figure 2 (D) shows the sorted items for the next rule. In Figure 2 (D), we can not find any item that covers only positive samples. That is, we can not find 1-gene condition rule; therefore, we have to combine 2 genes to find 2-gene condition rule to describe *class1*. From Figure 2 (E), we can find the second rule: ($gene_1$='B') && ($gene_2$='C') $\Rightarrow$ *class1*. It covers one positive sample and does not cover any negative sample. After the removal of all items covered by the second rule, no positive samples are left out. The process of generating rule groups is finally terminated.

In general, if all positive samples are covered by a rule, many items are needed to form a condition of the rule. However, it may lead to overfitting problem [5].

2.2. Avoiding overfitting

The aim of finding rule groups is to classify unlabeled samples (test data) using labeled samples (training data). The training samples can be classified perfectly only if we add more items in the conjunction of condition of rule. However, accuracy may be lost, when there is a noise in the data or the training dataset is small as the generated rules may overfit the training samples.

In order to avoid the overfitting problem, our algorithm restricts the number of items in the conditions of a rule. The long conditions can not reflect the common characters of a specific class according to the principle of Occam's razor. From our experiment, the number of items for one rule should not be more than four. Otherwise, the accuracy of generated rule groups will decrease. Furthermore, we employ post pruning to reduce overfitting. A rule will be pruned away if its support is less than a threshold. In our experiment, the threshold is set to 5%.

3. Algorithm

Our algorithm enumerates all possible combinations of items to find rule group to describe a specific class. Like most rule generation algorithms, the gene expression data is discretized to symbols. The dimensionality of gene expression data is usually very high; low discriminative genes are removed in the preprocessor of our algorithm.

3.1. Discretization & dimension reduction

Most of the association rule algorithms use symbol data. Many discretization methods have been proposed such as principal of components analysis, χ^2 based algorithm and etc. Entropy based technique is considered effectively method to discretize continuous attribute values [3]. After discretization, all values in gene expression matrix are converted into symbols. Meanwhile, dimensions with small entropy gain are removed. In our work, the discretization and dimension selection are combined together to prepare data.

As regards discretization, finding cutting points in continuous attribute values is very crucial. The cutting points are usually assumed in the middle of every two contiguous attribute values [7]. The cutting point that has largest information gain is used to separate intervals needed for discretization. This process is repeated recursively until information gain is greater than the minimal description length as given Equation 1.

$$gain > \frac{\log_2(N-1)}{N} + \frac{\log_2(3^k-2) - kE + k_1E_1 + k_2E_2}{N} \qquad (1)$$

where N is the number of samples, k is the number of classes, and E is the entropy of the whole set of samples. The entropy of the samples in left and right hands are *E1* and *E2*, respectively; the number of classes in left and right hands are *k1* and *k2*, respectively. It is to be noted that the right side of Equation (1) represents minimal description length. The algorithm used to discretize continuous attribute values and select discriminatory dimensions, is shown in Figure 3. The steps involved in this algorithm are given below: 1) Attribute values in each dimension are sorted out as given in line 4. 2) Find out the best cutting point as in line 5. 3) Put the pair of dimension number and cutting point in the result *Rset*, if the Equation (1) is satisfied. 4) A dimension is separated into two parts by its best cutting point. In the left and right sides, it is repeated to find out best cutting point until the Equation 1 is not satisfied as in lines 6-9. 5) Select the best *nc* points from the best cutting points as in line 12. When the number of cutting points in one dimension from *RSet*, is more than 1, the following strategy should be considered: If the first cutting point does not get selected in the final result, the second cutting point is ignored even if its information gain is the largest. It is due to the fact that the information gain of second cutting point is based on the first cutting point.

1. **Input:** the number of cutting point: *nc*
2. **Output:** pairs of dim and cutting point: *Rset*
3. for each d∈ Dim
4. sort continuous values of d to S
5. [S_1, S_2, *cp*]← findBestCutPoint(S)
6. if gain(S_1, S_2, *cp) > Eq. 1*
7. add {*d,cp*} to *Rset* descend in entropy
8. findBestCutPoint(S_1)
9. findBestCutPoint(S_2)
10. end if
11.end for
12.select first *nc* elements in *Rset*.

Figure 3 Entropy based discretization and dimension selection.

3.2. Implementation

To find the rule group to describe a specific class, the support and confidence are decided similar to Aprior-like algorithm. For a rule ***r***: *LHS*⇒*C*. support(*r*) = support(*LHS* ∪ *C*); confidence(*r*) = support(*r*)/support(*LHS*). In Figure 2, we set the support threshold = 1; confidence threshold = 100%, which is the strictest condition. In real datasets, we usually set loose thresholds for support and confidence. In our experiments, we set the threshold of support as five percent of the total number of samples, the threshold of confidence is set to one hundred percent. Support and confidence are two parameters of the algorithm. The maximum number of items in the condition of a rule, is another important parameter. As mentioned earlier, the number of items can not be large to reflect the common character of a specific class,. In our experiment, the largest number of items of condition is set to four.

To enumerate all possible items, the concept of candidate group [2] is employed. Each candidate group consists of two parts, namely, Head and Tail. Head is denoted as h(*g*) and Tail is denoted as t(*g*), where *g* represents a candidate group. The set enumeration tree [6] is used to generate all possible items.

The Head of root candidate group is empty and its Tail consists of all items as shown in Figure 4. The branch candidate groups are grown by moving the items from Tail to Head one by one. All the leaf candidate groups have empty Tail. The Heads in all leaf candidate groups are all possible items: {d_1}, {d_2}, {d_3}, {d_1, d_2}, {d_1, d_3}, {d_2, d_3}, {d_1, d_2, d_3}. Each node has two branches: the first one is generated by removing the first item in the Tail of the node; the second one is generated by moving the first item from Tail to Head. For example, a node h(g):{d_1, d_2,…}, t(g):{d_i,d_{i+1},…} has two branches h(g_1):{d_1, d_2,…}, t(g_1):{d_{i+1},…} and h(g_2):{d_1, d_2,…, d_i}, t(g_2):{d_{i+1},…}.

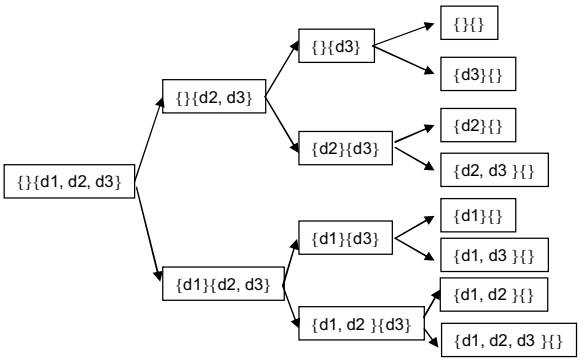


Figure 4 Enumeration of all possible items

4. Conclusion

In this paper, we propose a robust algorithm to find out rule groups that describe a specific class in high dimensional gene expression datasets. Our algorithm enumerates all possible combinations of dimensions. By introducing pruning power and constraint of the number of items, the procedures can be executed efficiently in any personal computer. The algorithm guarantees finding out best-k rules for a specific class data; the predictive accuracy is found to be better than that of the state of art methods. Future research may involve clarifying the properties of constant k in "best-k rules", and searching for a more systematic method of determining the constant k.

10. References

[1] An, J and Chen Y-P. P. Another inductive algorithm. KES'05 LNCS 3682: 37-44.

[2] Bayardo R. J. Efficiently mining long patterns from databases. SIGMOD'98 pp 85–93

[3] Fayyad, U. M. and Irani, K. B.. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. IJCAI93, pp 1022-1027, 1993.

[4] Mitchell, T. "Machine learning", McGraw Hill, 1997.

[5] Quinlan J. R. Induction of Decision Trees. Machine Learning 1(1): pp. 81-106 1986

[6] Rymon, R. Search through Systematic set Enumeration, Proceedings KR'92, pp. 268–275.

[7] Yang, Y., Pedersen J.P. A Comparative Study on Feature Selection in Text Categorization Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 1997, pp412-420.