Computational Biology and Chemistry 33 (2009) 108–113

Contents lists available at ScienceDirect

Computational Biology and Chemistry

journal homepage: www.elsevier.com/locate/compbiolchem

Brief Communication

Finding rule groups to classify high dimensional gene expression datasets☆

Jiyuan An[a,*], Yi-Ping Phoebe Chen[a,b,*]

[a] Faculty of Science and Technology, Deakin University, Melbourne, VIC 3125, Australia
[b] Australian Research Council Centre in Bioinformatics, Australia

ARTICLE INFO

Article history:
Received 26 September 2007
Received in revised form 24 July 2008
Accepted 24 July 2008

Keywords:
Gene expression datasets
Microarray data analysis
Classification

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 cannot 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) 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 (Alon et al., 1999; Chen et al., 2007; Knapp and Chen, 2007; Mann et al., 2007; Mramor et al., 2007). 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. Golub (Golub et al., 1999) firstly showed the better diagnostic performance of gene expression signatures in acute leukemia classification compared to other currently used diagnostic method. Many other studies (Bhattacharjee et al., 2001; Khan et al., 2001; Shipp et al., 2002) have been undertaken in almost all cancer types. Recently, a wide range of statistical and machine learning methods for microarray data analysis developed (Allison et al., 2006; Nahar et al., 2007; Asyali et al., 2006; Pham et al., 2006). Since the particularity of microarray data, i.e. high number of genes and small number of samples, it becomes a hard task to find accurate patterns to classify microarray data. This paper attempts to find rule groups for several different cancerous datasets. The results are encouraging in terms of accuracy and effectiveness.

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, cannot 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

☆ This work is partially supported by Grant DP0344488 from the Australian Research Council.

* Corresponding author at: Faculty of Science and Technology, Deakin University, Melbourne, VIC 3125, Australia.
E-mail addresses: Jiyuan@deakin.edu.au (J. An), phoebe@deakin.edu.au (Y.-P.P. Chen).

1476-9271/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compbiolchem.2008.07.031

be too long from the principle of Occam's razor (Mitchell, 1997); otherwise, the problem of overfitting will arise (Quinlan, 1986).

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 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 $(-\infty, 123.5]$ 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. Fig. 1 summarizes the terminologies that are used in 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 cannot 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 (Quinlan, 1986; An et al., 2005), SVM (Mitchell, 1997), and CN (Clark and Boswell, 1991), have been proposed These methods select discriminative features heuristically to describe common characters of a specific class. As they cannot effectively select high discriminative dimensions for high dimensional (features) datasets, a more robust algorithm has been proposed (An and Chen, 2005). 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. Describes constraint to avoid overfitting.

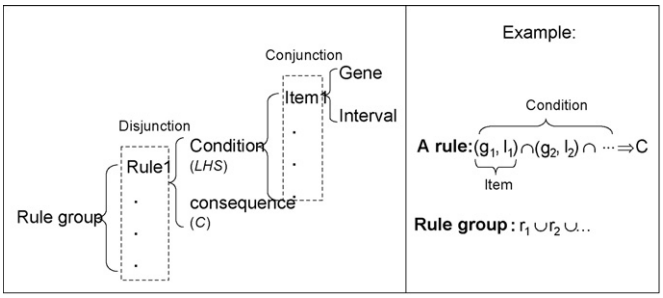

Fig. 1. The relationship of notations for rule group.

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:

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

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

(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. Only if always Fig. 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 Fig. 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 *class 1*. 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 Fig. 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 Fig. 2(C): the samples 1 and 4 are then removed. This process is repeated until all positive samples are removed. Fig. 2(D) shows the sorted items for the next rule. In Fig. 2(D), we cannot find any item that covers only positive samples. That is, we cannot find 1-gene condition rule; therefore, we have to combine 2 genes to find 2-gene condition rule to describe *class 1*. From Fig. 2(E), we can find the second rule: ($gene_1$ = 'B') *&&* ($gene_2$ = 'C') ⇒ *class 1*. 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 (Quinlan, 1986; Mitchell, 1997).

3. Methods

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 and 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, etc. Entropy-based technique is considered effectively method to discretize continuous attribute values (Fayyad and Irani, 1993). 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 (Fayyad and Irani, 1993; Yang and Pedersen, 1997). 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 in the following equation:

```
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.    [S1, S2, cp]← findBestCutPoint(S)
6.    if gain(S1, S2, cp) > Eq. 1
7.       add {d,cp} to Rset descend in entropy
8.       findBestCutPoint(S1)
9.       findBestCutPoint(S2)
10.   end if
11.end for
12.select first nc elements in Rset.
```

Fig. 3. Entropy-based discretization and dimension selection.

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

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 Eq. (1) represents minimal description length. The algorithm used to discretize continuous attribute values and select discriminatory dimensions, is shown in Fig. 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 Eq. (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 Eq. (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.

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: $LHS \Rightarrow C$. support (r) = support($LHS \cup C$); confidence (r) = support (r)/support (LHS). In Fig. 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 cannot 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 (Bayardo, 1998) is employed. Each candidate group consists of two

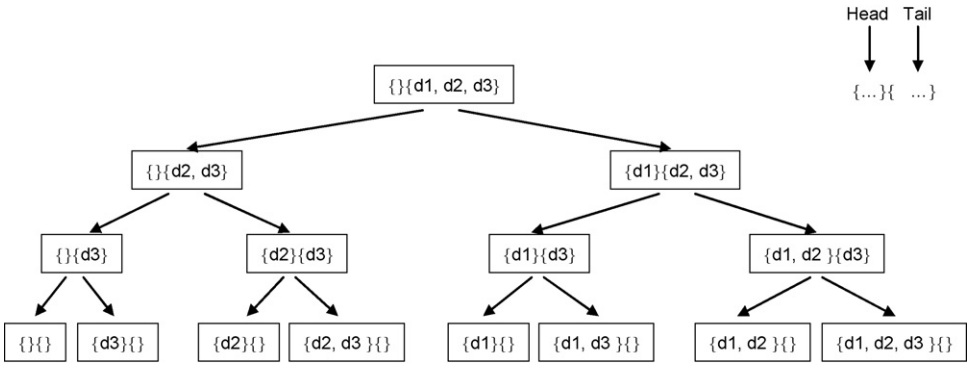

Fig. 4. Enumeration of all possible items.

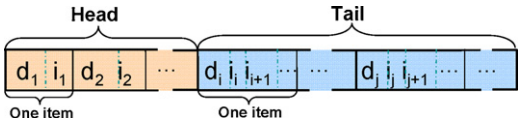

Fig. 5. The data structure of candidate group.

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 (Rymon, 1992) is used to generate all possible items. Fig. 4 shows an example to enumerate all possible items. For the sack of simplicity, we only list the dimensions: d_1, d_2 and d_3. The intervals are omitted.

The Head of root candidate group is empty and its Tail consists of all items as shown in Fig. 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 $h(g)$:$\{d_1, d_2,\ldots\}$, $t(g)$:$\{d_i,d_{i+1},\ldots\}$ has two branches $h(g_1)$:$\{d_1, d_2,\ldots\}$, $t(g_1)$:$\{d_{i+1},\ldots\}$ and $h(g_2)$:$\{d_1, d_2,\ldots, d_i\}$, $t(g_2)$:$\{d_{i+1},\ldots\}$. To represent intervals of a candidate group, a new data structure is needed as shown (Fig. 5). In, d_1 is a number and represents a dimension; i_1 is an interval of dimension d_1. In the Head portion, each dimension is followed by an interval. On the other hand, for the Tail portion, a dimension may be followed more than one interval. For example, an item "3 A C" means *gene3* = 'A' or 'C'. Fig. 6 illustrates the main algorithm to generate a rule group. The threshold of support, Tsd_S represents the minimum number of positive samples that are covered by a rule. The threshold of confidence, Tsd_c is the maximum confidence of a rule. The maximum positive coverage *maxCover* is initialized zero. When a rule is found, *maxCover* becomes the number of positive samples that are covered by the rule. The candidate group is initialized as C in line 8–9. We use a very simple gene dataset as shown in Fig. 2 to illustrate initialization. The gene dataset has only three dimensions, namely, *gene1*, *gene2* and *gene3*. Each dimension has two intervals such as 'A' and 'B'. The Head and Tail are initialized as {} and {{1 A B}}, {2 C D}, {3 E F}}. The {1 A B} means *dimension1* or *gene1* that has two intervals: 'A' and 'B'.

The items are added to Head to accommodate a dimension from Tail as shown in lines from 11 to 18 of Fig. 6. First, the new generated candidate groups *CForCheck* are checked whether the combination of Head with every item in Tail covers more than *maxCover* positive samples. Second, candidate groups are sorted out in the descending order of the number of positive samples that are covered by the Head of candidate groups as shown in Fig. 8(A). Third, if the

```
1  Input: threshold of support & confidence: Tsd_s, Tsd_c
2       Maximum number of items in LHS: maxLHS
3  Output: rule group RuleSet;
4  Global variables: F, C //frequent Pattern & candidates sorted by # of Samples of specific class  and other class
5                    maxCover
6  while dataset ≠ empty
7    maxCover = 0
8    h(g)={ }
9    t(g) = {all dimensions and their intervals}
10   CForCheck=g
11   while (True) do
12     for each g in CForCheck
13       chkFrequency(&g)
14     endfor
15     sortbyMaxPosSupport(CForCheck)
16     if (candidateGroup = empty) break
17     CForCheck=GenSubNode()
18   end while
19   rule = F.last
20   ruleSupport = removeRecordMatchRule(rule)
21   if ruleSupport ≥Tsd_s
22     RuleSet = RuleSet ∪ rule
23   endif
24 end while
```

Fig. 6. Main algorithm.

```
1  chkFrequency(g)
2  h(g_new) = h(g)
3  for each d ∈ t(g)
4    for each i ∈ intervals of d
5      posCover, negCover = positive and negative coverage of {h(g) ∪ {d, i}}
6      if (# of head < maxLHS) && (posCover > maxCover)
7        if posCover / (posCover + negCover) ≥ Tsd_s
8          F = F ∪ {h(g) ∪ {d, i}}
9          maxCover = h(g) ∪ {d, i} positive coverage
10       endif
11       t(g_new) = t(g_new) ∪ {d, i}
12     endif
13   end for
14 end for
15 g = g_new
```

Fig. 7. Check candidate groups.

candidate is empty, the rule is found. Fourth, the candidate group that has the largest positive coverage is chosen to expand its Head. It is expanded into two candidate groups as shown in Fig. 8(B). For example, a candidate group with $h(g)$ = {{1 'A'}} and $t(g)$ = {{2 'C'}, {3 'E'}} is expanded to $h(g1)$ = {{1, 'A'}}, $t(g1)$ = {{3 'E'}} and $h(g2)$ = {{1, 'A'}, {2 'C'}}, $t(g2)$ = {{3 'E'}}. The rule candidates are saved in a variable *F*. The last candidate is a rule. To find the next rule, the samples covered by the first rule are removed. If the rule satisfies support threshold, it is put into a rule group. The function called chkFrequncy can be found in line 13 of Fig. 7. It checks for the conditions as to whether the combination of Head with every item in Tail covers more that *maxCover* positive samples as detailed in Fig. 8. An item in Tail can be pruned, if its combination with Head does not cover more than *maxCover* positive samples. As a result, the most of nodes in Fig. 4 can be pruned out.

In the function *sortbyMaxPosSupport* as shown in Fig. 8(A), if a node or candidate group has empty Tail, it is removed. The combinations of items that satisfy thresholds of support and confidence have been put in F as candidates of rules. To find the rule that covers more positive samples fast, all candidate groups are sorted out in the descending order of the number of positive samples and in the ascending order of the number of negative samples. The candidate group that has biggest number of positive samples is selected to expand in the function called *GenSubNode* of Fig. 8(B). The expanded candidate groups are returned.

4. Experiment

We test our algorithm with widely used five gene datasets: ALL-AML leukemia (ALL), breast cancer (BC), colon tumor (CT), lung cancer (LC) and prostate cancer (PC). The rows of the datasets represent clinical samples; the columns represent the gene expression values, which is real data illustrating gene expression level of a specific gene for a sample. There are two classes of samples in these datasets. The datasets can be found at http://sdmc.i2r.a-star.edu.sg/rp.

Table 1
Gene expression datasets

Dataset	# samples	# genes	Class 1		Class 2	
			Label	#	Label	#
ALL	72	7129	ALL	47	AML	25
BC	97	24481	Relapse	46	Non-relapse	51
CT	62	2000	Negative	40	Positive	22
LC	181	12533	ADCA	150	Mesothelioma	31
PC	136	12600	Tumor	77	Normal	59

Table 2
Comparison with decision tree and support virtual machine

	Target class	Decision tree (%)	SVM (%)	Our method (%)
ALL	ALL	86.11	92.22	100.0
BC	Relapse	58.76	65.98	93.00
CT	Negative	75.80	83.87	91.43
LC	ADCA	90.60	99.45	100.0
PC	Tumor	78.68	91.18	87.14
Average		74.77	86.54	94.31

Table 1 shows the information of these five datasets: the number of samples (# samples), the number of genes (# genes), and two classes labels and the number of samples in the two classes. All experiments presented here use the class 1 as consequence in rule group. The minimum support is set to 5 percent of the number of samples. For example, ALL dataset has 2 ($47 \times 5\%$) samples minimum support. Confidence is set to percentage of total number of samples. The maximum number of items for a condition is set to 6. In the preprocessor, all gene expression values are discretized into symbols: '0' and '1'; the number of dimensions of these five datasets has been reduced to 30 by using entropy-based algorithm.

Our method is compared with decision tree and support virtual machine, which are considered effective methods to deal with high dimensional datasets. Table 2 illustrates the percentages of correctly predicted test data for three methods. We employ the tenfold cross-validation to evaluate their accuracy. The details of rule groups for these five datasets can be in http://www.deakin.edu.au/~phoebe/CBACAnChen/microarray_classification.html. Our method has high accuracy as compared to other two methods.

Table 3 shows the statistical data of rule groups for these five datasets. The number of rules per rule group reflects the complex of the datasets. The more the number of rules for each rule group is, the more the difficulty the dataset is described by rules. It can be used to explain that the prostate cancer (PC) dataset has low accuracy. An important element for rule group is average support

```
1  sortbyMaxPosSupport(CForCheck)
2    for each g in CForCheck
3      if tail(g) = {}
4        delete g from CForCheck
5      endif
6    end for
7    insert CForCheck into C sorted by primary key # of
         positive cover (ascending) and second key # of
         negative cover (descending)
```

(A)

```
1  GenSubNode()
2  deltaC = empty
3  g = last item in C
4  {d,i1,i2,..} ← 1st item in t(g)
5  t(g') ← t(g) - {d,i1,i2,..} //remove the 1st item
6  h(g) ∪ t(g') → last item in C
7  for each i in i1, i2,...
8      h(g')=h(g) ∪ {d, i} and
9          t(g') = {d'|d' ∈ t(g) and d' follows d in t(g)}
10     C=C ∪ g'
11     deltaC=deltaC ∪ g'
12 end for
13 return deltaC
```

(B)

Fig. 8. Generation of sub-node and sort of candidate group.

Table 3
The average support

	Average # rules/per rule group	Average # items/per condition	Average support/per rule (%)
ALL	2	1.5	79.47
BC	4.7	3.7	32.38
CT	3.4	2.8	46.40
LC	1	4	100
PC	6.2	3.35	18.02

per rule. It is desirable that every rule has large coverage. A rule that has a large support reflects the common character of the dataset. For example, the lung cancer (LC) dataset has 100% support for each rule. It means every generated rule reflects the common character for a specific class.

5. 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*.

References

Allison, D.B., et al., 2006. Microarray data analysis: from disarray to consolidation and consensus. Nat. Rev. Genet., 55–65.

Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., Levine, A.J., 1999. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc. Natl Acad. Sci. U.S.A. 96, 6745–6750.

An, J., Chen, Y.P.P., Chen, H., 2005. DDR: an index method for large time series datasets. Inform. Syst. 30, 333–348.

An, J., Chen, Y.P.P., 2005. Another inductive algorithm. Lect. Notes Artif. Intellig. 3682, 37–44.

Asyali, M.H., et al., 2006. Gene expression profile classification: a review. Curr. Bioinformatics 1, 55–73.

Bayardo, R.J., 1998. Efficiently mining long patterns from databases. 17th ACM SIGMOD International Conference on Management of Data, 85–93.

Bhattacharjee, A., et al., 2001. Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. Proc. Natl Acad. Sci. U.S.A. 98, 13790–13795.

Chen, Q., Chen, Y.P.P., Zhang, C., 2007. Detecting inconsistency in biological molecular databases using ontology. Data Min. Knowl. Discov. 15 (2), 275–296.

Clark, P., Boswell, R., 1991. Rule induction with CN2: some recent improvements. Mach. Learn. EWSL-91, 151–163.

Fayyad, U.M., Irani, K.B., 1993. Multi-interval discretization of continuous-valued attributes for classification learning. IJCAI 93, 1022–1027.

Golub, T.R., et al., 1999. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286, 531–537.

Khan, J., et al., 2001. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nat. Med. 7, 673–679.

Knapp, K., Chen, Y.P.P., 2007. An evaluation of contemporary hidden markov model genefinders with a predicted exon taxonomy. Nucleic Acids Research 35, 317–324.

Mann, S., Li, J., Chen, Y.P.P., 2007. A pHMM-ANN based discriminative approach to promoter identification in prokaryote genomic contexts. Nucleic Acids Res. 35 (e12), 1–7.

Mitchell, T., 1997. Machine Learning. McGraw Hill.

Mramor, M., Leban, G., Demsar, J., Zupan, B., 2007. Visualization-based cancer microarray data classification analysis. Bioinformatics 23 (16), 2147–2154.

Nahar, J., Chen, Y.P.P., Ali, S., 2007. Kernel based naive bayes classifier for breast cancer prediction. J. Biol. Syst. 15 (1), 17–25.

Pham, T.D., et al., 2006. Analysis of microarray gene expression data. Curr. Bioinformatics 1, 37–53.

Quinlan, J.R., 1986. Induction of decision trees. Mach. Learn. 1 (1), 81–106.

Rymon, R., 1992. Search through systematic set enumeration. Proceedings KR'92, 268–275.

Shipp, M.A., et al., 2002. Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. Nat. Med. 8, 68–74.

Yang, Y., Pedersen, J.P., 1997. A comparative study on feature selection in text categorization. Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 412–420.