CONSTRAINED CLUSTERING OF GENE EXPRESSION PROFILES

Ivica Slavkov¹, Sašo Džeroski², Jan Struyf³, Suzana Loškovska¹
1. Faculty of Electrical Engineering, University of "Ss. Cyril and Methodius", Skopje, Macedonia
2. Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
3. Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium

ABSTRACT

In this paper a querying environment for analysis of patient clinical data is presented. The data consists of two parts: patients' pathological data and data about corresponding gene expression levels. The querying environment includes a generic algorithm for constructing decision trees, as well as algorithms for discretizing gene expression levels and for searching frequent patterns (itemsets). The algorithms are accessed by means of a query language. The language can be used to simulate various data mining algorithms, such as the one developed by Morishita et al. for Itemset Constrained Clustering.

1 INTRODUCTION

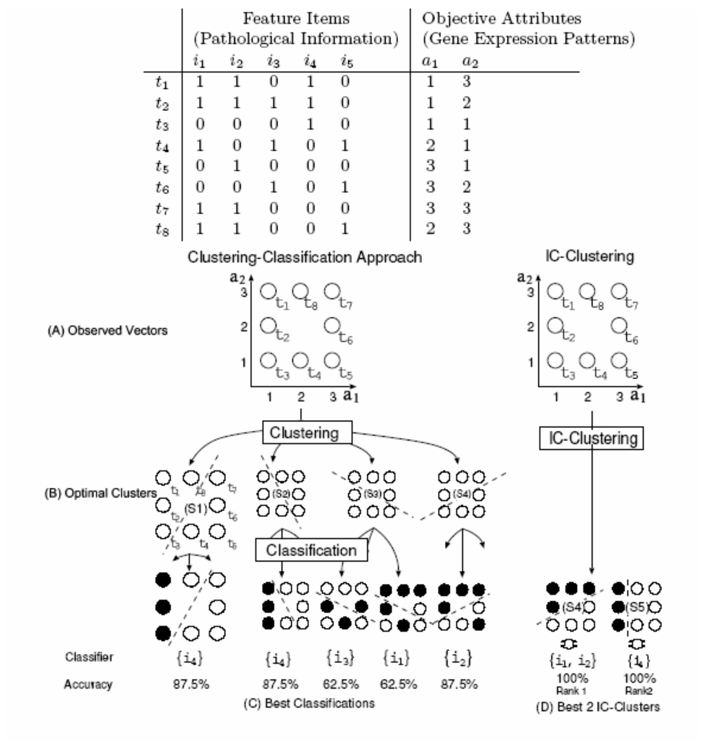
In recent years a lot of data recording gene expression levels (microarray data) has been generated at a very high throughput. One of the challenges faced by molecular biologists is to discover knowledge from this data. This includes diagnosis of disease, classifying disease and gaining information that can suggest possible treatments. Various data mining techniques have been developed for finding useful gene expression profiles (discovering bi-sets, formal concept mining [5]) and for discovering relations between the patients pathological information and microarray data. Of particular interest to this paper is the technique of itemset constrained clustering of gene expression levels based on patients' pathological features.

2 ITEMSET CONSTRAINED CLUSTERING

A classical approach towards relating the two parts of data (pathological and gene expression) would be a two step clustering-classification process (Figure 1). First, the clustering is performed according to the similarity of gene expression profiles and then a classifier is constructed for each cluster. The classifier is actually one of the pathological features (attributes), i.e., it predicts a tuple to be in the cluster if it includes the particular feature, and usually does not have 100% accuracy. Itemset constrained clustering [3] takes this one step further by allowing *n*-itemsets of the pathological data to act as classifiers, but only if they have 100% accuracy. It allows only clusters which can be expressed by item sets. The IC-clustering algorithm itself, is an association rule mining and clustering algorithm at the same time. The index which is maximized in this algorithm is *interclass variance*, which is defined as follows:

Let n be $|D|$ (the number of tuples) and $x(I)$ be $|D_I|$ (the number of tuples including itemset I) for a given database $D = \{t_1, t_2, ..., t_n\}$. Let $s_i = \sum_{t \in D} t[a_i]$, and $y_i(I) = \sum_{t \in D_I} t[a_i]$, then the *interclass variance* is defined as:

$$\mathrm{var}(D, I) = x(I)\sum_{i=1}^{m}\left(\frac{y_i(I)}{x(I)} - \frac{s_i}{n}\right)^2 + (n - x(I))\sum_{i=1}^{m}\left(\frac{s_i - y_i(I)}{n - x(I)} - \frac{s_i}{n}\right)^2$$



	Feature Items (Pathological Information)					Objective Attributes (Gene Expression Patterns)	
	i_1	i_2	i_3	i_4	i_5	a_1	a_2
t_1	1	1	0	1	0	1	3
t_2	1	1	1	1	0	1	2
t_3	0	0	0	1	0	1	1
t_4	1	0	1	0	1	2	1
t_5	0	1	0	0	0	3	1
t_6	0	0	1	0	1	3	2
t_7	1	1	0	0	0	3	3
t_8	1	1	0	0	1	2	3

Figure 1: A comparison between the clustering-classification and itemset-constrained clustering approach.

In short, the IC-clustering algorithm can be described as follows.

IC-clustering algorithm:

- **Input:** minimum cluster size (C), maximum number of tuples in which an item *I* is contained (S)
- **1**. Search for a feature itemset that splits the tuples into two clusters
- **2**. Compute the interclass variance between the clusters
- **Output:** List of the top *N* itemsets sorted by interclass variance

3 QUERY LANGUAGE

The purpose of developing the query language was to allow a simple and easy-to-use approach towards mining gene expression data. On one hand it allows mining for local patterns (i.e., finding frequent itemsets) and also for the creation of global models of the data like predictive clustering trees [1]. It also has various discretization techniques for microarray data implemented (Mid Range, X%Max, Max-X%Max), as described in [6]. For each query the rule is that it always has as input a data table specified, and as a result it also returns a table, but analysed or processed in some way. Also, the idea was to relate the query language to inductive databases [2] by allowing compositionality of the queries: the output of one query can be used as an input to another.

3.1 Query Language Syntax

The rules for generating the query language predicates are given in BNF[1] form. They can be divided in a number of groups according to their function.
The first group would be the one that allows basic content selection. It consists of the following three clauses:

<create-clause>

Used for creating new tables from rows and columns previously selected or processed in some way. It has the following notation:

<create-clause>::= **create** <table-name> **using** <extend-clause>|<select feature-clause>

example: **create** "newPatients" **using select from** "patients" **where** "tumor==yes"

<extend-clause>

Useful for extending existing tables by adding new attributes (columns) which are somehow acquired.

[1] Backus Naur Form.

<extend-clause>::= extend <table-name> **with** <table-name**>**

example: **extend** "pathological" **with** "genes"

<select feature-clause>

Selecting a specific part of the data is often required. With this clause, a selection can be made by specifying a set of names of attributes, by certain attribute values or by simply selecting which rows and columns are needed.

<select-clause>::= select from <table-name> **where** (<attnames="names"> | <att-names=att-values> | <rows&columns>)

example: **select from** "genes" **where** "attnames=={gene1, gene5, gene9 }"

The clauses through which basic data mining tasks can be achieved are:

<discretize-clause>

Gene expression level data often needs to be discretized before various data mining techniques for pattern discovery can be used. Many discretization techniques have been proposed [6] and several are implemented in our query language.

<discretize-clause>::= discretize (* | < att- names > **|** <rows&columns>**) in** <table-name> **[using <**method name>**]**

example: **discretize * in** "geneNumeric"

<frequent items-clause>

Discovering frequent itemsets (patterns) from Boolean data is very useful for extracting information, especially for discovering genes that are co-regulated. The following clause uses ACminer [7] to extract these frequent itemsets with specified support.

<frequent items-clause>::= **frequent itemsets in** (<table-name>) **with support** <real>

example: **frequent itemsets in** "pathological" **with support** 0.65

<external-clause>

For greater flexibility in using the query language, the external clause can be used. The idea is to allow an interface with external programs which can be used to further process the data or represent it in some other way.

<external-clause>::= call external <program name & location> **[using** <parameters>**]**

example: **call external** "c:\program\program.exe" **using** " -i "file" "

<convert-clause>

This clause provides some basic conversion utilities. Usually data represented in binary form is not suitable as input to the different miners, so it must be transformed to the required format.

<convert-clause>::= convert (bin | miner | freqitems) <table-name> **to (miner|bin|features)**

example: **convert bin** "patientsBin" **to miner**

<predict-clause>

This is the last clause, and it is used for interfacing with the Clus system (Section 3.2). Clus is used for constructing predictive clustering trees [1], with support for certain constraints, which are specified in the <parameters> part of the predict clause.

<predict-clause>::=predict from <table-name> **using** <parameters>

example: **predict from "**patients" **using** clusterN=50

3.2 Brief Description of Clus

Clus is a generic system for constructing decision trees. It can be used to create classification trees for predicting symbolic attributes and regression trees for predicting numerical values. In some cases it is useful to predict several attributes at the same time, so multi-objective decision trees can also be constructed. Clus can generate so-called Predictive Clustering Trees (PCTs) [1], using a beam search algorithm. PCTs are decision trees that are used for hierarchical clustering purposes. Clus also allows for generating trees with user-constraints, which can be supplied to Clus by means of our query language.

4 SIMULATING IC-CLUSTERING

The IC-clustering algorithm can be easily simulated without its dedicated implementation with the help of the query language described above. It is an association rule mining and clustering algorithm, so it can be simulated by the following steps:

- Find frequent itemsets using the patients pathological data
- Create a modified patient record using the previously generated frequent itemsets
- Create PCT stubs using the beam search algorithm of Clus

The first step is finding frequent itemsets in the patient records Boolean data, with the user specified support. The ability to specify the support of the frequent itemsets simulates the parameter *C* (minimum cluster size) in the IC-clustering algorithm. The second step of creating the so-called modified patient records is actually using the frequent itemsets as patient features, i.e., we add a Boolean attribute for each of the item sets, which takes the value true if and only if the record contains the item set. To this modified patient record we "append" the gene expression level data. By using Clus we generate PCT stubs, i.e., decision trees with only one test node. The test is selected from the modified pathological data and imposes a binary split (clustering) on the samples. The interclass variance is calculated for each stub and if the corresponding clustering is suitable it is put in the current search beam of width N.

5 RESULTS

The first scenario for analysing gene expression data with the query language that we considered was the previously described simulation of IC-clustering. The queries which are used to perform this simulation are:

- **create** *"freqpatients"* **using frequent items in** (**convert bin** *"patients"* **to miner**) **with support** 0.*1*
- **create** *"features"* **using** (**convert freqitems** *"freqpatients"* (**convert bin** *"patients"* **to miner**) **to freqfeatures**)
- **extend** *"features"* **using** *"expression"*
- **predict** *"features"* **with** *clusterN=100 targetAtt=535-2527*

The first query creates a table that is generated using the frequent itemset clause. The support is set to 0.1 to match the one that is used in [3]. Then with a few conversions the frequent itemsets are used to modify the patients' pathological data and create the "features" table. In the third query the gene expression levels are merged with the modified patients' pathological data. In the end the PCTs are created using Clus.
The dataset that was used is the same as in [3]. It contains patient records of 213 patients. The gene expression level data considers 1993 genes. The data has been previously preprocessed and missing values were substituted (predicted) in some way. The patients' pathological data contains the features: age >/< 65, sex M/F, tumor/except tumor, chirossis/ no chirossis, abnormal / normal liver, abnormal/normal liver function.
The results that were acquired are presented in the following table:

Rank	Constraint	Cluster size	Interclass variance
1	{tumor}	107	3126.9
2	{except tumor, normal liver function}	88	2534.7
3	{except tumor, HBV-}	88	2397.3
4	{tumor, man}	86	2181.5
5	{except tumor, HBV-, normal liver function}	74	2098.9
6	{except tumor, man}	83	2037.87
7	{except tumor, no cirrhosis}	68	1979.74
…	…	…	…
17	{tumor, not over 65 years old}	55	1587.7

The results in this table are identical to those in [3][2].

The second scenario concerns prediction of patients diagnosis from gene expression levels. The microarray data was first discretized using the Max-X%Max method, for a threshold of X=50% which encodes the over expressed genes with 1 and the others with 0. Then frequent itemsets were found with a support greater than 30%. Similarly as in the previous scenario they were used as features and a modified gene expression dataset was created. They were combined with the patient pathological data. Prediction was made with Clus concerning which set of genes are over expressed in patients that have a tumor.

The queries that were used are:

- **create** "*freqGenes*" **using (frequent itemsets in (discretize * in** *"genes"* **using** *"Max-XMax X=50%"*) **with support 0.3)**
- **create** *"featuresGenes"* **using** (**convert freqitems** *"freqGenes"* (**convert bin** *"patients"* **to miner**) **to freqfeatures**)
- **extend** "*featuresGenes"* **using** "*patient"*
- **predict** "*featuresGenes*" **with** *clusterN=100 targetAtt=1*

The results are presented in the table:

Rank	OverExpressed Gene	Cluster Size	Error of prediction
1	{GS11588, GS2496}	78	0.286
2	{GS2496}	79	0.291
3	{GS1659}	77	0.309
4	{GS1859}	71	0.319
5	{GS72419}	76	0.342
6	{GS12398}	73	0.347
…	…	…	…

[2] Unfortunately, the table in [3] contains two errors concerning the size of the clusters (Jun Sese, personal communication).

6 CONCLUSION AND FURTHER WORK

The purpose of the query language that was developed was to create an environment that supports basic data mining tasks, and also a compositionality of queries thus relating the language to the concept of inductive databases. With the previously described scenarios we aimed to demonstrate that the query language facilitates gene expression analysis and also allows for the simulation of some data mining algorithms.

Our further work would be first to validate the usefulness of the query language by testing it with other patient record/gene expression datasets and by using it in new scenarios. Also introducing new query constructs in the language is considered, like feature selection, mining for bi-sets and others.

References

[1] H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In Proceedings of the 15th International Conference on Machine Learning, pages 55–63, 1998.

[2] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of the ACM, 39(11):58–64, 1996.

[3] Sese Jun, Yukinori Kurokawa, Kikuya Kato, Morito Monden and Shinichi Morishita. Constrained Clusters of Gene Expression Profiles with Pathological Features. *Bioinformatics vol. 20 issue 17 Oxford University Press 2004.*

[4] Ruggero G. Pensa and Jean-François Boulicaut. Boolean Property Encoding for Local Set Pattern Discovery: An Application to Gene Expression Data Analysis. *International Seminar, Dagstuhl Castle, Germany, April 12-16, 2004, Revised Selected Papers p.115-134.*

[5] Jérémy Besson, Céline Robardet, Jean-François Boulicaut and Sophie Rome. Constraint-based concept mining and its application to microarray data analysis. *Intelligent Data Analysis 9(1): 59-82 (2005).*

[6] Ruggero G. Pensa, Claire Leschi, Jérémy Besson and Jean-François Boulicaut. Assessment of discretization techniques for relevant pattern discovery from gene expression data. *In proceedings of BIOKDD 2004: 24-30.*

[7] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering Frequent Closed Itemsets for Association Rules. *Proceeding of the 7th International Conference on Database Theory 1999 p.398-416.*