1A2-3

Extracting Approximate Biclusters/Patterns from Time Series Medical Data Using Suffix Trees

Simona Muwazi Makoto Haraguchi

Garduate School of Information Science and Technology, Hokkaido University

Time series medical data contains many null values and is collected over a long period of time. The focus is on extracting longer decreasing/increasing patterns/biclusters that may be of interest to medical experts in analysing drug responses and therapies, as well as predicting certain disease occurences. We apply the technique of biclustering to extract new interesting patterns from this data. Given the data for each patient, we discretize it to obtain a symbolic representation using statistical methods. We then proceed to efficiently construct a compact generalized suffix tree over the entire dataset. The algorithm presented in this work extends the problem of common motif searching as applied in microarray experiments to extract approximate biclusters from within the suffix tree utilizing a form of string edit distance restricted to substitution and deletion, and the concept of valid models.

1. Introduction

Recently, mining time series data in many different fields of research, such as genetic microarray experiment analysis and speech processing, has received a lot of focus. In this work, we develop and present an algorithm directed at processing medical time series data.

Several approaches for dealing with this kind of data have been proposed, biclustering being one of the more recent. It is also known as co-clustering; i.e. the simultaneous clustering of rows and columns in a data matrix. The precise definition of a bicluster depends on the end result desired by the application, although they generally fall into broad categories; constant value biclusters, and coherent value biclusters.

In this paper, we develop and present an algorithm to extract approximate biclusters from time series medical data using suffix trees. By analyzing medical data, we may be able to discover previously unknown patterns that predict certain types or classes of diseases or analyze the efficacy of various drugs and therapies when applied to patients.

The next sections describe the data set and explain the motivation to use biclustering using suffix trees, together with the key points of the algorithm. We conclude by presenting a brief summary of experimental results.

2. The Data Set: Characteristics

The data set we use was provided by Chiba University Hospital, and consists of a set of records for patients that attended the hospital over a period of 2-5 years, during which time they were diagnosed for thrombosis and the related collagen diseases. Table 1 lists basic information on all patients, such as ID number, birth date, and etc. Table 2 provides a record of the patien's visits to the hospital together with results for any medical tests carried out.

Table 1: Basic information recorded for each patient at the hospital, regardless of eventual diagnosis.

Item	Meaning
ID	Identification of the patient
Sex	–
Birthday	–
Description Date	The first date when a patient data was recorded
First Date	The date when a patient came to the hospital
Admission	Patient was admitted to the hospital (+) or $\cdots$
Diagnosis	Disease names

Table 2: Details the test results for a patient during hospital visits.

Item	Meaning
ID	Identification of the patient
Examination Date	Date of the test
aCL IgG	anti-Cardiolipin antibody (IgG) concentration
aCL IgM	anti-Cardiolipin antibody (IgM) concentration
ANA	anti-nucleus antibody concentration
ANA Pattern	Pattern observed in the sheet of ANA examination
aCL IgA	anti-Cardiolipin antibody (IgA) concentration
Diagnosis	Disease names

For our purposes, the data can thus be viewed as an $R \times C$ matrix of data for each patient, where $R = \{T_1, \ldots, T_n\}$ represents the test items, and $C = \{t_0, \ldots, t_n\}$ represents time instances (see Figure 1).

Patient visits are highly irregular, and not all tests are carried out at every visit. The data, although time series, is therefore characterized by several null values and missing records.

3. Pattern Definition

Before proceeding any further, we explain the kind of patterns we are interested in searching for.

In this case, data is recorded over prolonged periods of

Contact: Makoto HARAGUCHI
Graduate School of Information Science and Technology, Hokkaido University
Address: N-14 W-9, Sapporo 060-0814, JAPAN
Phone : +81-11-706-7106
E-mail : mh@ist.hokudai.ac.jp

	t_0	t_1	t_2	t_3	$\cdots$
T_1	245	235	198	209	$\cdots$
T_2	10	9.7	-	11	$\cdots$
T_3	67	58	65	98	$\cdots$

Figure 1: An $R \times C$ matrix representing the data collected for one patient.

time (2-5 years), with several null values present. We are therefore interested in searching for longer patterns that display an increasing or decreasing trend with respect to a given test item. We therefore define patterns as n-combinations of test items that preserve temporal order. An $n = 1$ pattern will consist of a single test item such that

$$test_i = v_{ij}(j = 1, \ldots, m),$$

e.g. $TEST1 = Normal \rightarrow Increasing$.

Further n-combinations of test items may be defined as need arises. For example, a pairwise pattern with $n = 2$ may be defined as

$$test_k\_test_\ell = v_{kj}v_{\ell j},$$

e.g.

$$\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = \begin{pmatrix} L \\ N \end{pmatrix}$$

assuming, L and N represent a given quality, in this case `Low` and `Normal`.

4. Biclustering: Key Definitions

Given the kind of patterns we are interested in searching, biclustering is a natural choice. A bicluster in our case is equivalent to a pattern; from here on, we use the two terms interchangeably.

Definition 1 (Bicluster): *A bicluster is a sub-matrix A_{IJ} defined by $I \subseteq R$, a subset of rows, and $J \subseteq C$, a subset of columns.*

A bicluster with only one row or column can be considered trivial. Because we are dealing with time series data, preserving time information is of the utmost importance. For this reason, we define cc, or column-coherent biclusters.

Definition 2 (CC-Bicluster): *A column coherent bicluster (cc-bicluster) A_{IJ} is a bicluster such that $A_{ij} = A_{\ell j}$ for all raws $i, \ell \in I$ and columns $j \in J$.*

It is to be noted that searching for all biclusters satisfying this coherence property is generally an NP-hard problem.

However, an important complexity reduction can be made due to the time constraint; data must observe time coherence, i.e. have contiguous columns. This leads to another important definition:

Definition 3 (CCC-Bicluster): *A contiguous column coherent bicluster A_{IJ} is a subset of rows $I = \{i_1, \ldots, i_k\}$ and a subset of* contiguous *columns $J = \{r, r+1, \ldots, s-1, s\}$ such that $A_{ij} = A_{\ell j}$, for all rows $i, \ell \in I$ and columns $j \in J$.*

Lastly, we wish to define maximal biclusters. To achieve longer patterns, we must search for the largest possible biclusters.

Definition 4 (Maximal-CCC-Biclusters): *A CCC-Bicluster A_{IJ} is maximal if no other CCC-Bicluster exists that properly contains A_{IJ}, that is, if for all other CCC-Biclusters A_{LM}, $I \subseteq L \wedge J \subseteq M \Rightarrow I = L \wedge J = M$.*

5. Approximate Biclusters

The definitions above apply to perfect biclusters, in the sense that no error margin is permitted. However, the medical data set in question contains several null values. This implies that to identify longer patterns, we must introduce an error constraint. Before defining approximate biclusters, we would like to define the notion of an error neighborhood for a string;

Definition 5 (Error Neighborhood): *The Error-Neighborhood of a string S of length $|S|$, defined over the alphabet Σ with $|\Sigma|$ symbols, $N(e, S)$, is the set of strings S_i, such that: $|S| = |S_i|$ and $Hamming(S, S_i) \leq e$, where e is an integer such that $e \geq 0$.*

This means that the Hamming distance between S and S_i is no more than e, that is, we need at most e symbol substitutions to obtain S_i from S.

We can now define an approximate ccc-bicluster as follows:

Definition 6 (Approximate CCC-Bicluster): *A ccc bicluster with e errors per test item, is a ccc-bicluster A_{IJ} where all the strings S_i that define the expression pattern of each of the test items in I are in the error neighborhood of an expression pattern S that defines the approximate-ccc-Bicluster: $S_i \in N(e, S), \forall i \in I$.*

Lastly, we formally define the notion of maximal approximate biclusters;

Definition 7 (Maximal Approximate CCC-Biclusters): *An approximate-ccc-bicluster A_{IJ} is maximal if it is raw-maximal, left-maximal and right-maximal.*

This now fits our notion of a pattern; we are effectively searching for the longest possible patterns while allowing errors in order to cope with the null values in the data. We can now state the problem:

Problem Statement: Given a discretized expression matrix A and three integers $e \geq 0$, $q_r \geq 2$ and $q_c \geq 1$, where q_r is the row constraint (minimum number of rows in I_k) and q_c is the column constraint (minimum number of columns in J_k), identify and report all maximal *approximate*-ccc-biclusters

$$B_k = A_{I_k J_k}$$

such that, I_k and J_k have at least q_r raws and q_c columns, respectively.

6. Using Suffix Trees

Suffix trees are a data structure designed to efficiently process strings. While computationally expensive to construct, several string operations can be performed very efficiently once the tree is built. Due to Ukkonen's algorithm [1] in particular, it is possible to construct the suffix tree in log-linear time.

The motifs searched by SPELLER correspond to *words* over an alphabet Σ, which must occur with at most e mismatches in $2 \leq q \leq N$ distinct sequences. Since these words representing the motifs may not be present exactly in the sequences, a motif is seen as an "external" object and is termed a *model.*

SPELLER builds a generalized suffix tree T for the set of sequences S_i and after some further preprocessing, uses this tree to "spell" the valid models. Valid models verify two properties:

1. All the prefixes of a valid model are also valid models.
2. When $e > 0$, spelling a model leads to a set of nodes $v_1, \ldots, v_k$ in T for which

$$\sum_{j=1}^{k} L(v_j) \geq q,$$

where $L(v_j)$ denotes the number of leaves in the subtree rooted at v_j.

7. Extending SPELLER to Solve Our Problem

The valid models identified by the original SPELLER algorithm are already row-maximal. However, they may be non right-maximal, non left-maximal, and start at different positions in the sequences.

Under these conditions, our modified version of SPELLER identifies one row-maximal, right-maximal *approximate*-ccc-bicluster with q rows and a maximum of $|C|$ contiguous columns.

We first extend the algorithm to extract *all right-maximal approximate biclusters* by fixing the quorum constraint used to specify the number of rows/test items necessary to identify a model as valid, to the value $q = 2$.

In this context, and in order to be able to solve our problem, we adapt SPELLER to consider not only a row constraint, $2 \leq q_r \leq |R|$, but also an additional column constraint, $2 \leq q_c \leq |C|$.

It is not possible to modify SPELLER in order to check if a valid model that is right-maximal is also left-maximal. This is so since we can only guarantee that a model is/is not left-maximal once we have computed all valid models corresponding to right maximal *approximate*-ccc-biclusters.

Therefore, in order to achieve *left-maximal approximate biclusters*, we must discard valid models which are not left-maximal in the next step of the algorithm.

In summary, we modify the SPELLER [2] as follows;

1. We redefine the original concept of node-occurrence to use the tripe (v, v_{err}, p) in order to accommodate error.
2. We only store valid models that cannot be extended to the right without losing test items that is valid models which are both row-maximal and right-maximal.
3. In SPELLER the node-occurrences of a valid model can start in any position in the sequences. In the modified version of this algorithm all node-occurrences of a valid model must start in the same position (same column in the discretized matrix) in order to guarantee that they belong to an *approximate*-ccc-bicluster.

8. Algorithm

The algorithm we propose here is thus based on the following steps;

Step 1: Compute all valid models corresponding to right-maximal *approximate*-ccc-biclusters.

Step2: Delete all valid models not corresponding to left-maximal *approximate*-ccc-biclusters. Uses all valid models computed in Step 1.

Step 3: Delete all valid models that represent the same *approximate*-ccc-biclusters. Uses all valid models corresponding to maximal *approximate*-ccc-biclusters (both left and right) computed in Step 2.

Step 4: Report all maximal *approximate*-ccc-biclusters.

9. Complexity

The asymptotic complexity of the algorithm is given by

$$O(\max(|R|^2|C|^{1+e}|\Sigma|^e, |R||C|^{2+e}|\Sigma|^e)).$$

However, in the case of medical data sets, $|R| \gg |C|$, so complexity can be given by

$$O(|R|^2|C|^{1+e}|\Sigma|^e).$$

10. Results and Discussion

Since we use the suffix tree construct, we must convert our data set to symbolic form. Before any conversion is done, we remove all redundant date.

We then apply statistical tests to determine the ranges for each individual test item, and then assign a symbol from a defined alphabet to each item. We do this using $\Sigma = \{\mathtt{N}, \mathtt{L}, \mathtt{H}, \_\}$ to represent normal, low, high and null or missing values, respectively. We also assign column numbers to each item for ease of identification during generalized suffix tree construction.

	t_0	t_1	t_2	t_3	$\cdots$
T_1	N_0	N_1	H_2	H_3	$\cdots$
T_2	N_0	L_1	$-_2$	N_3	$\cdots$
T_3	L_0	L_1	N_2	H_3	$\cdots$

Figure 2: Data-preprocessing step produces the above matrix.

Table 3: A cross-section of maximal approximate biclusters

#	ID	Pattern	# time-points (1^{st} to last)	# test items
1	15	LLLLNNHHH	$t_6 - t_{14}$	2
2	22	HHHNNNHH	$t_0 - t_7$	3
3	19	LLLNLLLNNN	$t_8 - t_{17}$	3
4	35	NNNHHHHH	$t_1 - t_8$	5
5	89	HHLLLLLLN	$t_{12} - t_{20}$	3

We take an absolute starting point for the timeline for each patient, and then proceed to record values in increments of 28 calendar days. The resulting matrix for each patient is given in Figure 2.

When the algorithm is applied to the medical time series data, it extracts and reports all maximal approximate biclusters in polynomial time. A cross-section of results is presented in Table 3.

The table lists a cross-section of the maximal approximate biclusters attained when $e = 3$ and minimum row and column constraints $q_r = q_c = 2$.

11. Future Work

We may treat the maximal approximate biclusters as basic features representing each patient. We can then attempt to formulate more complex or abstract features by an implication of abstraction hierarchy among the basic features, allowing for weighting and constraints to reduce on the complexity.

Currently we are considering the extension of formal concept analysis to medical time series data, and time series data in general.

References

[1] E. Ukkonen, Online Construction of Suffix Trees, Algorithmica, 14(3), pp. 249 – 260, 1995.

[2] M. F. Sagot, Spelling Approximate Repeated or Common Motifs Using a Suffix Tree, Proc. of the 3rd Latin American Symposium on Theoretical Informatics, Springer-LNCS 1380, pp. 374 – 390, 1998.

[3] S. C. Madeira and A. L. Oliveira, A Linear Time Biclustering Algorithm for Time Series Gene Expression Data, Proc. of WABI2005, Springer-LNCS 3692, pp. 39 – 52, 2005.