IBM Frequent Subgraph Miner

- Blurb

A data mining tool to discover all frequent substructure patterns from a set of labeled graphs.

- Contents

executable file : Linux version only
sample data : analyzed data which is created from 340 chemical compound.
ReadMe file : this file

- Overview

The input to this mining tool is a set of labeled graphs in which each vertex and each edge have a vertex label and an edge label respectively. When a set of vertices V, a set of edges E, a set of vertex labels L_V and a set of edge labels L_E are provided as

$$V=\{v_1,v_2,\ldots,v_k\},$$
$$E=\{(v_i,v_j) \mid v_i,v_j \in V,\ i \neq j\},$$
$$L_V=\{lb(v_i) \mid \forall v_i \in V\},$$
$$L_E=\{lb((v_i,v_j)) \mid \forall (v_i,v_j) \in E\}$$

respectively, graph G is expressed as $G=(V,E,L_V,L_E)$. The number of vertices, $|V|$, is called the size of graph G. Given an undirected graph G, if a path exists between any two vertices, G is called a connected graph.

Given a set GD of labeled graphs, the support sup(P) of a graph pattern P is defined as a ratio of the number of graph data including P to the total number of graph data in the dataset GD.

$$\sup(P) = \frac{|G \mid G \in GD, P \subseteq G|}{|GD|},$$

where $P \subseteq G$ means G includes P as (induced) subgraph.

When the dataset which consists of graph structured data and the minimum support are given as input, this mining tool efficiently discovers all frequent connected (induced) subgraphs that have the support greater than or equal to the minimum support value in the dataset.

You can get detail description of the algorithm from

http://domino.watson.ibm.com/library/cyberdig.nsf/1e4115aea78b6e7c85256b360066f0d4/2b6c952f98bcc3c785256bd40023f13d?OpenDocument&Highlight=0,

Inokuchi.

- System Requirements

This runs on Linux only. Requirements for CPU and main memory depend on analyzed data.

- Installation Instruction

Put executable file in bin directory.

- FAQ

Q. How to make analyzed data?

A. This executable file can analyze a set of undirected graph which do not contain self-looped vertices. The format of input file is as follows.

size(x,y)
: x: graph identifier starting with 1.
: y: size which means the number of vertices in the graph.

name(x,y)
: x: graph identifier starting with 1.
: y: name of the graph

node(x,y,z)
: x: graph identifier starting with 1.
: y: vertex identifier starting with 1.
: z: vertex labels

link(x,y1,y2,z)
: x: graph identifier starting with 1.
: y1: vertex identifier starting with 1.
: y2: vertex identifier starting with 1.
: z: edge label of an edge between vertices y1 and y2.

Q. How to get frequent patterns?

A. add –out option. All discovered patterns are output as follow when a sample data is used.

size = 4
VertexLabels = c22,c22,h3,h3

code = r,s,0,0,s,0

count = 189

It means that one of the discovered patterns has 4 vertices, 189 graphs contain this pattern, and its structure is represented by the following adjacency matrix.

$$\begin{array}{c} \\ c22 \\ c22 \\ h3 \\ h3 \end{array} \begin{array}{c} \begin{array}{cccc} c22 & c22 & h3 & h3 \end{array} \\ \begin{pmatrix} 0 & r & s & 0 \\ r & 0 & 0 & s \\ s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \end{pmatrix} \end{array},$$

Q. What is code?

A. In order to reduce the candidates of the frequent subgraphs, the code of an adjacency matrix is defined as follows. The code of an adjacency matrix X_k whose size is k x k is defined as

$$code(X_k)=x_{1,2}x_{1,3}x_{2,3}x_{1,4}\ldots x_{k-2,k}x_{k-1,k}$$

by using (i,j)-element $x_{i,j}$ of X_k.

- Executables

For example, input as follows from command line, where pte.txt is a sample data.

acgm –in pte.txt –sup 10 –subgraph –out

usage: acgm [options] [files]

[options]

-in [filename]	input file name
-sup [minimumsupport]	minimum support(%)
-subgraph (or -isubgraph)	derived pattern (subgraph or induced subgraph)
-out	output derived patterns

- Bio

Akihiro Inokuchi
Tokyo Research Laboratory, IBM Japan
Inokuchi@jp.bim.com