Mining Correlated Patterns in Transactional Databases

What is correlated pattern mining?

Frequent pattern mining aims to discover all interesting patterns in a transactional database that have **support** no less than the user-specified **minimum support** (**minSup**) constraint and **all-confidence** no less than the user-specified **minimum all confidence** (***minAllConf***). The **minSup** controls the minimum number of transactions that a pattern must appear in a database. The **confidence** indicates how reliable the rule is by knowing how frequently rule head occurs among all the groups containing the rule body.

What is the transactional database?

A transactional database is a collection of transactions, where each transaction contains a transaction-identifier and a set of items.
A hypothetical transactional database containing the items ***a, b, c, d, e, f, and g*** as shown below

tid	Transactions
1	a b c g
2	b c d e
3	a b c d
4	a c d f
5	a b c d g
6	c d e f
7	a b c d
8	a e f
9	a b c d
10	b c d e

Note: Duplicate items must not exist in a transaction.

Acceptable format of transactional databases in PAMI

Each row in a transactional database must contain only items. PAMI algorithms implicitly consider the row number as the transactional-identifier to reduce storage and processing costs.

```
a b c g
b c d e
a b c d
a c d f
a b c d g
c d e f
a b c d
a e f
a b c d
b c d e
```

Understanding the statisctics of database

To understand about the database. The below code will give the detail about the transactional database.

- Total number of transactions (Database size)
- Total number of unique items in database
- Minimum lenth of transaction that existed in database
- Average length of all transactions that exists in database
- Maximum length of transaction that existed in database
- Standard deviation of transaction length
- Variance in transaction length
- Sparsity of database

The sample code

```
import PAMI.extras.dbStats.transactionalDatabaseStats as stats

obj = stats.transactionalDatabaseStats('sampleInputFile.txt', ' ')
obj.run()
obj.printStats()
```

What is the input to correlated pattern mining algorithms

Algorithms to mine the correlated patterns requires transactional database, minSup and minAllConf, (specified by user).

- Transactional database in following formats:

  > - In string format (`/Users/Likhitha/Downlaods/sampleInputFile.txt`)
  > - In URL format (`https://www.u-aizu.ac.jp/~udayrage/datasets/transactionalDatabases/tra`
  > - In DataFrame format (dataframe variable with heading `Transactions`)

- minSup should be mentioned in **count (beween 0 to length of database)** or __percentage (multiplied with length of database)
- minAllConf should be mentioned between 0 to 1
- Specify the seperator of input file.

What is the output of correlated pattern mining algorithms

The output of these algorithms is in two ways:

> - Saving the patterns in user specified output file.
> - Returns the patterns in dataframe variable.

How to run the frequent pattern algorithm in terminal

- Download the code from github.
- Navigate to PAMI folder where you downloaded the file.
- Go to correlatedPattern/basic folder
  Execute the following command on terminal.

python3 algorithmName.py `path of Sample input file` `path of output file` minSup minAllConf `seperator`

Sample command to execute the CPGrowth code in frequentPattern/basic folder

python3 `CPGrowth.py` `/Users/Donwloads/inputFile.txt` `/Users/Downloads/outputFile.txt` 3 0.4 `' '`

How to implement the code by importing PAMI package

Import the PAMI package executing: **pip3 install PAMI**

Run the below sample code by making simple changes

- Replace sampleInputFile name or path in place of iFile and sampleOutputFile name or path in place of oFile
- Specify the minimum support (like 10 or 0.1) in place of minSup
- Specify the minimum all confidenceAll (between 0 to 1) in place of minAllConf
- Specify the seperator of input file after minSup. (If no seperator is specified the default tab seperator is considered for input file)

import PAMI.correlatedPattern.basic.CPGrowth as alg
obj = alg.CPGrowth(iFile, minSup, minAllConf, ' ')
obj.startMine()
obj.savePatterns(oFile) (to store the patterns in file).
Df = obj.getPatternsAsDataFrame() (to store the patterns in dataframe)
obj.printStats()

What is the output of frequent pattern mining algorithms

Returns the pattern, support and confidence respectively with $minSup$ = 6 and $minAllConfidence$ = 0.7 from above sample database

The output in file format:

```
d : 8 : 1.0
d c : 8 : 0.8888888888888888
c : 9 : 1.0
b : 7 : 1.0
b c : 6 : 0.6666666666666666
b d : 6 : 0.75
b c d : 6 : 0.6666666666666666
a : 7 : 1.0
a c : 6 : 0.6666666666666666
```

The output in DataFrame format:

	Patterns	Support	Confidence
0	d	8	1.0
1	d c	8	0.8
2	c	9	1.0
3	b	7	1.0
4	b c	6	0.66
5	b d	6	0.75
6	b c d	6	0.66
7	a	7	1.0
8	a c	6	0.66