2022/08/22 11:06 EPCPGrowth-st

Discovering Periodic Frequent patterns in Big Data Using EPCPGrowth Algorithm

In this tutorial, we will discuss two approaches to find Periodic Frequent patterns in big data using top algorithm.

- 1. **Basic approach:** Here, we present the steps to discover Periodic Frequent patterns using a single minimum support value
- 2. **Advanced approach:** Here, we generalize the basic approach by presenting the steps to discover Periodic Frequent patterns using multiple minimum support values.

Basic approach: Executing EPCPGrowth on a single dataset at a particular minimum support value

Step 1: Import the EPCPGrowth algorithm

```
In [1]: from PAMI.periodicCorrelatedPattern import EPCPGrowth as alg
```

Step 2: Specify the following input parameters

```
inputFile = 'temporal_T10I4D100K.csv'

minimumSupportCount=100  #Users can also specify this constraint between 0 to 1.
maxmunPeriodCount=5000
minAllConfCount=0.5
maxPerAllmaxPerConfCount=0.5
seperator='\text{\frac{4}{4}}'
```

Step 3: Execute the EPCPGrowth algorithm

```
obj = alg. EPCPGrowth(iFile=inputFile, minSup=minimumSupportCount, minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf=minAllConf
```

2022/08/22 11:06 **EPCPGrowth-st**

```
[21] 4248 219 1 1
[20] 4258 236 1 1
[19] 4309 327 1 1
[18] 4388 224 1 1
[17] 4438 185 1 1
[16] 4511 232 1 1
[15] 4559 189 1 1
[14] 4629 173 1 1
[13] 4681 201 1 1
[12] 4902 197 1 1
[11] 4973 174 1
[10] 4993 143 1 1
[9] 5057 159 1 1
[8] 5102 186 1 1
[7] 5375 204 1 1
[6] 5408 184 1 1
[5] 5835 140 1 1
[4] 5845 136 1 1
[3] 6265 112 1 1
[2] 6810 146 1 1
[1] 7057 133 1 1
[0] 7828 113 1 1
Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
```

Step 4: Storing the generated patterns

Step 4.1: Storing the generated patterns in a file

```
obj. savePatterns (outFile='periodicFrequentPatternsMinSupCount100.txt')
In [4]:
```

Step 4.2. Storing the generated patterns in a data frame

```
periodicFrequentPatternsDF= obj.getPatternsAsDataFrame()
In [5]:
```

Step 5: Getting the statistics

Step 5.1: Total number of discovered patterns

```
print('Total No of patterns: ' + str(len(periodicFrequentPatternsDF)))
In [6]:
        Total No of patterns: 0
```

Step 5.2: Runtime consumed by the mining algorithm

```
print('Runtime: ' + str(obj.getRuntime()))
In [7]:
        Runtime: 5.089999437332153
        ##### Step 5.3: Total Memory consumed by the mining algorithm
In [8]:
        print('Memory (RSS): ' + str(obj.getMemoryRSS()))
In [9]:
        print('Memory (USS): ' + str(obj.getMemoryUSS()))
        Memory (RSS): 572571648
        Memory (USS): 533786624
```