Data Mining CS 634

Midterm Project

Apriori Algorithm for Association rule generation

Done by Shishir Sumant

UCID: ss3688

Under Professor Dr. Jason T.L. Wang,

Department of Computer Science

New Jersey Institute of Technology

Contents

1. Software and Hardware requirements
2. Description
3. Source code
4. Screenshots

1. Software and Hardware requirements

1.1 Software:

* Language used: Python 3.7
* IDE: JetBrains PyCharm Edu 2018.2 x64
* Platform: Windows 10
* Database: CSV files stored in Windows File System

1.2 Minimum System requirements

* Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit)
* 2 GB RAM minimum
* 4 GB RAM recommended
* 1024x768 minimum screen resolution
* Python 2.4 or higher, Jython, PyPy or IronPython

2. Description

The project represents an implementation of the Apriori algorithm to generate association rules between item-sets, given the values of minimum support and confidence.

The dataset is a CSV file consisting of 20 different transactions, comprising a total of ten unique elements. On running the program, the user is prompted to enter a value for minimum support and minimum confidence, based on which the association rules are generated.

The algorithm first reads all the transactions, line by line to determine a unique 1 item candidate list. The count of each element is calculated, and it is tested whether it meets the minimum support criteria. The size of candidate itemset is then iteratively increased to generate k item frequent sets which meet the support criteria. The other item-sets are pruned. Item-sets and their corresponding support values are stored in a dictionary as a key-value pair.

Once the unique frequent item-sets are listed, a function is called to list out all the association rules. These rules are calculated based on the minimum confidence value given by the user. The confidence is calculated as follows:

Confidence(A->B)=Support\_count(A∪B)/Support\_count(A)

The program outputs the following:

1. All the transactions in the dataset
2. Number of unique items in the transactions (1 item candidates)
3. Frequent k item candidate sets with increasing size iteratively
4. Association rules

The same is implemented on 5 different datasets, 3 of which contain 10 unique items and 2 other datasets which contain a different set of 10 unique items.

Minimum support and confidence values are changed to show variation in results.

The application of this algorithm in real world is to predict the likeliness of a customer who buys a set of items, will buy another set of item/items. It helps in maximizing profits and enhancing sales.

3.Source Code

**import** csv  
  
*#function to parse the items in transaction files***def** readTrans(fPath):  
 transList = []  
 **with** open(fPath)**as** transCsv:  
 csvRead = csv.reader(transCsv)  
 **for** trans **in** csvRead:  
 eachTrans = trans[0].split(**','**)  
 transList.append(eachTrans)  
 print(**'Transactions in the set are'**,transList)  
 **return** transList  
  
*#function to find the 1 itemset candidate list***def** findCand1(allTrans):  
 Candsize1 = []  
 **for** eachTrans **in** allTrans:  
 **for** item **in** eachTrans:  
 **if not** [item] **in** Candsize1:  
 Candsize1.append([item])  
 Candsize1.sort()  
 **return** list(map(frozenset,Candsize1))  
  
*#function to check whether itemset meets minimum support condition***def** supportScan(alltrans,Candsetk,minsup,transcount):  
 kItemset = {}  
 **for** eachtrans **in** alltrans:  
 **for** itemset **in** Candsetk:  
 **if** itemset.issubset(eachtrans): *#increment count if itemset is present* **if not** itemset **in** kItemset:  
 kItemset[itemset] = 1  
 **else**:  
 kItemset[itemset] += 1  
 kfreq = []  
 ksupport = {}  
 **for** itemset **in** kItemset:  
 support = kItemset[itemset]/transcount *#calculate support* **if** support >= minsup: kfreq.insert(0,itemset)  
 ksupport[itemset] = support  
 **return** kfreq,ksupport  
  
*#function to generate candidate itemsets iteratively with increasing size***def** apriori(freqlistk, sizek):  
 candListk = []  
 kLen = len(freqlistk)  
 **for** i **in** range(kLen):  
 **for** j **in** range(i+1, kLen):  
 list1 = list(freqlistk[i])[:sizek-2]  
 list2 = list(freqlistk[j])[:sizek-2]  
 list1.sort()  
 list2.sort()  
 **if** list1==list2:  
 candListk.append(freqlistk[i] | freqlistk[j])  
 **return** candListk  
  
*#calculates the support values and composes a final list of candidate itemsets***def** itemSetk(dataSet, support,candList1,totTrans):  
 dataSetList = list(map(set, dataSet))  
 frequentList1, supDict = supportScan(dataSetList, candList1, support,totTrans)  
 totList = [frequentList1]  
 size = 2  
 **while** (len(totList[size-2]) > 0):  
 candlistk = apriori(totList[size-2], size)  
 freqlistk, supportK = supportScan(dataSetList, candlistk, support,totTrans)  
 supDict.update(supportK)  
 print(**'\nCandidate Itemset of size '**,size,**' is'**,freqlistk)  
 totList.append(freqlistk)  
 size += 1  
 **return** totList, supDict  
*#generates association rules with confidence greater than min confidence***def** getMinConfSet(freqItemSet,current,SupInfo, rules, minconf):  
 prunedCurrent = []  
 **for** itemset2 **in** current:  
 currentConf = SupInfo[freqItemSet]/SupInfo[freqItemSet-itemset2]  
 **if** (currentConf >= minconf):  
 print(freqItemSet-itemset2,**' ==> '**, itemset2,**' has confidence '**,currentConf)  
 rules.append((freqItemSet-itemset2, itemset2,currentConf))  
 prunedCurrent.append(itemset2)  
 **return** prunedCurrent  
*#creates new association rules by merging new items to existing itemset***def** generateNew(freqItemSet, current, supInfo, rules, minconf):  
 currentSetLen = len(current[0])  
 freqlen = len(freqItemSet)  
 **if**(freqlen > (currentSetLen+1)):  
 newCand = apriori(current,currentSetLen+1)  
 newCand = getMinConfSet(freqItemSet,newCand,supInfo,rules,minconf)  
 newLen = len(newCand)  
 **if**(newLen>1):  
 generateNew(freqItemSet,newCand,supInfo,rules,minconf)  
  
  
**def** association(suplist,supdata,minconf):  
 listFin = []  
 **for** i **in** range(1,len(suplist)):  
 **for** itemset1 **in** suplist[i]:  
 tempcopy = [frozenset([item]) **for** item **in** itemset1]  
 **if** (i > 1): generateNew(itemset1,tempcopy,supdata,listFin,minconf)  
 **else**: getMinConfSet(itemset1,tempcopy,supdata,listFin,minconf)  
 **return** listFin  
  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 minSupport = input(**'Enter the minimum support value in percentage'**)  
 minSupport = float(minSupport)/100  
 minConfidence = input(**'Enter the minimum confidence value in percentage'**)  
 minConfidence = float(minConfidence)/100  
 Transactions = readTrans(**'C:/Users/HP/Desktop/Transaction3.csv'**)  
 totTrans = float(len(Transactions))  
 candList1 = findCand1(Transactions) *#find the candidate 1 itemset* numItems=len(candList1)  
 print(**'\nThe number of unique items in the dataset are: '**,numItems)  
 print(**'\nThe items are as follows: '**,candList1)  
 finList,supportInfo = itemSetk(Transactions,minSupport,candList1,totTrans) *#iteratively find k- frequent itemsets* print(**'\n Item sets and corresponding support values are:'**)  
 print(supportInfo)  
 print(**'\n The list of rules are : \n'**)  
 finitems = association(finList,supportInfo,minConfidence)

4.Screenshots

![A screenshot of a cell phone

Description generated with very high confidence]()

Transaction file 1

![A screenshot of a social media post

Description generated with very high confidence]()

Result generated for file1 with support =20% and confidence =30%

![A screenshot of a cell phone

Description generated with very high confidence]()

Transaction file 2

![A screenshot of a social media post

Description generated with very high confidence]()

Result generated for file2 with support =10% and confidence =10%

![A screenshot of a social media post

Description generated with very high confidence]()

Result of file 2 continued

![A screenshot of a cell phone

Description generated with high confidence]()

Transaction file 3

![A screenshot of a social media post

Description generated with very high confidence]()

Result generated for file3 with support =5% and confidence =5%

![A screenshot of a social media post

Description generated with very high confidence]()

Result of file3 continued

![A screenshot of a cell phone

Description generated with high confidence]()

Transaction file 4

![A screenshot of a social media post

Description generated with very high confidence]()

Result of file 4 with support = 15% and confidence = 15%

![A screenshot of a cell phone

Description generated with very high confidence]()

Transaction file 5

![A screenshot of a social media post

Description generated with very high confidence]()

Result of file 5 with support = 50% and confidence = 50%