频繁项集挖掘算法-Apriori 算法实现

交叉研 20　　谭泽人　　2020211335

2020 年 10 月 24 日

1　引言

关联规则是在给定数据项集上频繁出现的两个项集之间的关系。两个项集具有一定的关联规则需要具备两个条件，首先是他们都是频繁出现的，即他们的出现次数需要满足一个认为设定的参数，在本实验中为 `minSup`。其次是，他们的出现需要具有一定的联系，在本实验中用 confidence 即置信度来衡量两个项集出现的关系。通过挖掘关联规则，我们可以找到两个项集之间出现之间的关系，通过关联规则我们可以通过一个项集的出现预测另一个项集出现的概率。它在现实生活中有需要应用，例如购物篮数据分析，网页预取，交叉购物，个性化网站，网络入侵检测。商家可以通过挖掘顾客购物的关联规则来设计商店的促销规则，从而获取更高的利润。例如在 Walmart 超市，通过对购物票据的挖掘，发现成年男性在购买啤酒的同时会同时购买尿不湿，超市通过将啤酒和尿不湿捆在一起做活动——购买尿不湿送啤酒，促进了更多人购买尿不湿，从而获得更高的盈利。因此关联规则挖掘在现实生活中十分重要。

关联规则挖掘首先被 Agrawal *et al.* 提出 [2]。在这之后有很多的算法被提出，例如 Apriori 算法 [1]，FP-Growth [3]，CHARM 算法 [5]，CLOSET+ 算法 [4] 等。Apriori 算法作为较为基础的算法，是学习关联股则挖掘算法的基础，因此本实验将实现该算法，并且评测其性能。

2　算法概述

在 Apriori 算法中，我们需要先指定一个度量一个项集的出现是否频繁的参数 minimum support，出现次数大于 minimum support 的项集是频繁的。Apriori 算法首先扫描一遍数据，找到只包含一个项并且频繁的项的集合。之后递归的生成包含 k 个项的频繁项集，直到没有更大的项集可以生成。在此过程中所生成的项集必须是频繁的，即出现的次数大于 minimum support。Apriori 算法的伪代码如下：

Algorithm 1: Apriori

$L_1 \leftarrow$ the set of frequent items;
for $k = 1; L_k \neq \emptyset; k++$ **do**
  $C_{k+1} \leftarrow \emptyset$;
  for $i, j \in L_k$ **do**
    if $|i \cup j| = k + 1$ **then**
      $C_{k+1} \leftarrow C_{k+1} \cup \{i \cup j\}$;
    end
  end
  for *every transaction* t *in database* **do**
    for $c \in C_{k+1}$ **do**
      if $c \subset t$ **then**
        the frequency of c increases 1;
      end
    end
  end
  for $c \in C_{k+1}$ **do**
    if *frequency of* $c \geq$ *minSup* **then**
      $L_{k+1} \leftarrow L_{k+1} \cup c$;
    end
  end
end
return $\cup_k L_k$;

3 算法实现

3.1 数据处理

我使用的是老师提供的 `groceries.csv` 文件，其每一行包含一条交易数据，每条交易数据由一个交易的 item 数量和各个 item 组成：

我先读取数据，利用 python 中的 `set` 数据类型保存数据，生成包含所有交易数据的一个 Python generator：

```
def readDataFromFile(filename):
    '''
    parameter(s)::
        filename: the file name that is going to be read.
    return::
        a generator contains all transactions
    '''
    with open(filename,'r') as file:
        for line in file:
            line = line.rstrip(',\n')
```

	Item(s)	Item 1	Item 2	Item 3	Item 4
0	4	citrus fruit	semi-finished bread	margarine	ready soups
1	3	tropical fruit	yogurt	coffee	NaN
2	1	whole milk	NaN	NaN	NaN
3	4	pip fruit	yogurt	cream cheese	meat spreads
4	4	other vegetables	whole milk	condensed milk	long life bakery product

图 1: 数据样例

```
            record = frozenset(line.split(',')[1:])
            yield record
```

并从中利用 python set 类型保存所有交易的 item，用 python list 保存所有的交易：

```
def getItemSetTransactionList(transactionGenerator):
    '''
    parameter(s)::
        transactionGenerator: a generator contains all transactions
    return::
        itemSet: the set of items exist in all transactions
        transactionList: a python list contains all transactions
    '''
    itemSet = set()
    transactionList = list()
    for record in transactionGenerator:
        transaction = frozenset(record)
        transactionList.append(transaction)
        for item in transaction:
            itemSet.add(frozenset([item]))

    return itemSet, transactionList
```

Apriori 算法的主体如下：

```
def Apriori(transactionGenerator, minSup, minConf):
    '''
    parameter(s)::
        transactionGenerator: a generator contains all transaction
        minSup: minimum support
        minConf: minimum confidence
    return::
        LSetDict: a dictionary consists of patterns of all length that satisfy the minimum support
     constraint
```

```
        freqPattern: a python list contains all frequent pattern
        associationRule: a python list contains all association rule satisfy minimum support and minimum
     confidence constraints
    '''
    itemSet, transactionList = getItemSetTransactionList(transactionGenerator)
    freqDict = defaultdict(int)
    LOneCandidate = minSupFilter(itemSet,transactionList,minSup,freqDict)
    k = 2
    LSetDict = dict()
    currentLSet = LOneCandidate
    while (currentLSet != set()):
        LSetDict[k-1] = currentLSet
        tempSet = currentLSet
        currentLSet = generateLkSet(tempSet,k)
        currentCSet = minSupFilter(currentLSet,transactionList,minSup,freqDict)
        currentLSet = currentCSet
        k += 1

    freqPattern = list()
    for key, lset in LSetDict.items():
        freqPattern.extend([(set(item),freqDict[item]) for item in lset])
    associationRule = list()
    for key, lset in list(LSetDict.items())[1:]:
        for lseq in lset:
            properSubsets = [frozenset(x) for x in properSubset(lseq)]
            for item in properSubsets:
                if (len(lseq.difference(item))>0):
                    conf = freqDict[lseq]/freqDict[item]
                    if conf >= minConf:
                        associationRule.append([set(item),set(lseq.difference(item)),conf])
    return LSetDict, freqPattern,associationRule
```

先生成只包含一个 item 的满足 minSup 要求的频繁项集 `LOneCandidate`，之后递归的生成包含 k 个 item 的频繁项集并保存在 `LSetDict[k]` 中。`freqPattern` 包含所有的频繁项集及其相应的 support。`associationRule` 包含了所有的关联规则以及相应的 confidence。

运行结果示例：

```
>>> path = 'groceries.csv'
>>> file = readDataFromFile(path)
>>> minSup = 100
>>> minConf = 0.5
>>> t1 = datetime.now()
>>> LSetDict, freqPattern, associationRule = Apriori(file,minSup,minConf)
>>> t2 = datetime.now()
>>> print('Total running time of Apriori with minSup %d, minConf %.2f is: %.4f s' %(minSup,minConf,(t2-
    t1).seconds))
Total running time of Apriori with minSup 100, minConf 0.50 is: 7.0000 s
```

4 不同参数下算法的结果

我的算法中设置了两个参数 minSup 和 minConf，分别代表 support 的最小阈值和 confidence 的最小阈值。我在不同的参数设定下分别运行算法。实验环境为 windows 10 环境下 laptop CPU@1.60GHz。

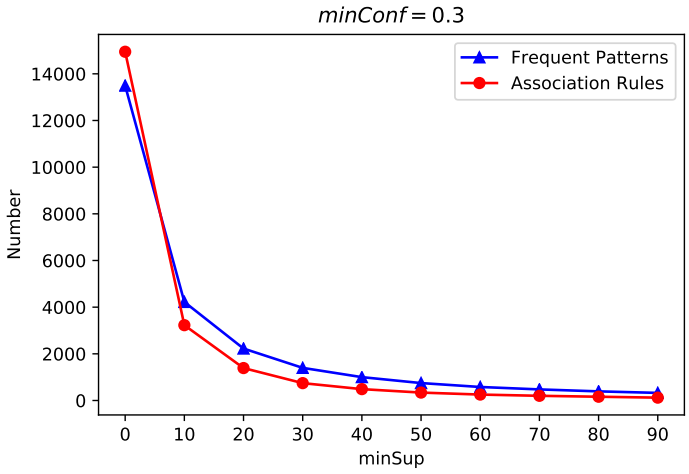

图 2: 不同 minSup 下的结果

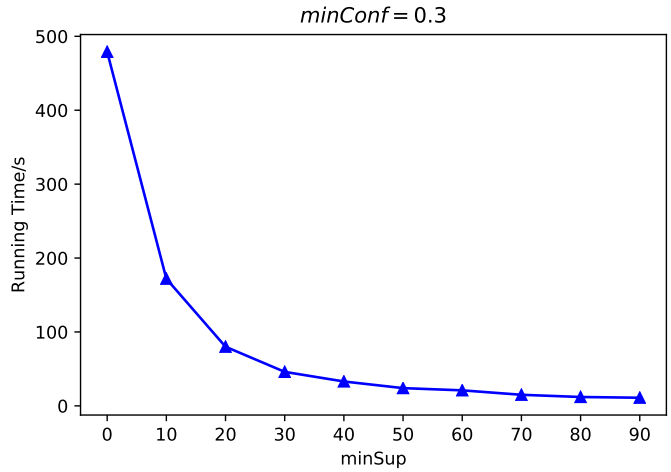

图 3: 不同 minSup 下的算法运行时间

4.1 不同 minSup 下的结果

随着 minSup 的增大，将会有更多的项集因为出现的次数不够多而被剪枝，从而最终得到的频繁项集的大小以及关联规则的大小都将减小，如图 2 所示。

并且因为剪枝增加，算法运行时间减小。

4.2 不同 minConf 下的结果

minConf 只影响关联规则的挖掘，随着 minConf 增大，关联规则被剪枝越多，最终满足约束的关联规则数量减少，但是频繁项集的数量不受影响，如图 4 所示。

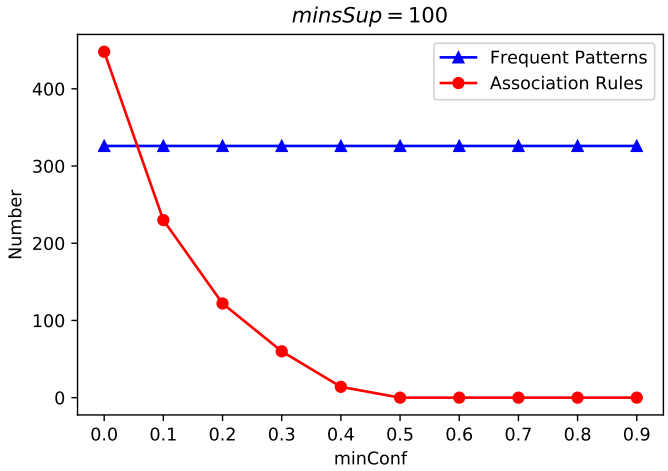

图 4: 不同 minConf 下的结果

5 实验代码

提交的结果中包含一个 `Apriori.py` 和一个 `Apriori.ipynb` 文件，ipynb 文件中包含了算法的实现和实验的结果，比较详细，可读性较.py 文件更强。如果需要运行.py 文件，请在命令行输入以下命令：

```
$ python -f  file.csv -s minSup -c minConf -of freqPath -oa assoPath
```

其中，`-f` 用于指定 csv 文件，`-s` 用于指定 minSup，`-c` 用于指定 minConf，`-of` 和`-oa` 分别用于指定 freqPatterns 和 associationRules 存储的路径。默认的 minSup 为 10，默认的 minConf 为 0.3，freqPath 和 assoPath 分别默认为 freqPattern.csv 和 associationRules.csv。

参考文献

[1] Rakesh Agarwal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, 1994.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[3] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2):1–12, 2000.

[4] J Wang, J Han, and J Pei. Closet+: Scalable and space-saving closed itemset mining. *Submitted for publication*, 2003.

[5] Mohammed J Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining*, pages 457–473. SIAM, 2002.