# 《数据挖掘导论》实验 5:apriori算法实验

# 一、实验目的

（1）了解apriori算法思想

（2）掌握apriori算法的应用

# 二、实验环境

（1）Anaconda2 开发环境

（2）IDE是ipython notebook

（3）使用的库有numpy，pandas，matplotlib

# 三、实验内容

## 1.关联分析

关联分析是一种在大规模数据集中寻找有趣关系的任务。这些关系有两种形式：频繁项集或者关联规则。频繁项集是经常出现在一块的物品集合，关联规则暗示两种物品之间可能存在很强的关系。

假设拥有某个杂货店的交易清单如下表所示。

|  |  |
| --- | --- |
| 交易号码 | 商品 |
| 0 | 豆奶，莴苣 |
| 1 | 莴苣，尿布，葡萄酒，甜菜 |
| 2 | 豆奶，尿布，葡萄酒，橙汁 |
| 3 | 莴苣，豆奶，尿布，葡萄酒 |
| 4 | 莴苣，豆奶，尿布，橙汁 |

支持度可以反映频繁项集的频繁程度，一个项集的支持度被定义为数据集中包含该项集的记录所占的比例。从表中可知，在5条记录中有3条包含{豆奶，尿布}，因此{豆奶，尿布}的支持度为3/5。

可信度或者置信度是针对于一条诸如{尿布}→{葡萄酒}的关联规则来定义的。这条规则的可信度被定义为“支持度（{尿布，葡萄酒}/支持度{尿布}）”，其实就是相当于求条件概率。从表中可求得，{尿布}的支持度为3/5，{尿布，葡萄酒}的支持度为3/5，所以“{尿布}→{葡萄酒}”的可信度为3/4=0.75。

## 2.apriori

apriori算法通过剪枝思想可以快速求出频繁项集。

(1) apriori原理

假设一件商店只有4种商品：商品0，商品1，商品2和商品3。

所有商品之间的可能组合如下图所示：

3

2

1

0

23

13

12

03

02

01

123

023

013

012

0123

apriori的原理是如果某个项集是频繁的，那么它的所有子集也是频繁的。逆否命题也是成立的，即如果一个项集的子集中存在非频繁项集，那么该项集也是非频繁的。

假设上图的{2,3}是非频繁项集，那么{0,2,3}，{1,2,3}，{0，1,2,3}也是非频繁项集，在计算频繁项集时候，这几个项集就没必要计算支持度了。这个过程从图上来看就很符合剪枝的思想，在这个例子中可以把阴影部分都去掉。在求频繁项集的时候可以逐层进行，先判断单个物品是否频繁，不频繁则去掉。然后用频繁的集合两两合并求两件物品的频繁项集（第二层），接着用第二层的频繁项集两两合成第三层的项集，再去掉非频繁项集，逐层搜索去掉非频繁项集，直到最后一层。

(2) 使用apriori算法来发现频繁项集

Apriori算法的两个输入参数分别是最小支持度和数据集。该算法首先会生成所有单个物品的项集列表。接着扫描交易记录来查看哪些项集满足最小支持度要求，那些不满足最小支持度的集合会被去掉。然后，对剩下来的集进行组合以生成包含两个元素的项集。接下来重新扫描交易记录，去掉不满足最小支持度的项集。该过程重复进行直到所有项集都被去掉。

构建初始项集的伪代码如下表所示。

|  |
| --- |
| 对数据集中的每条交易记录tran  对每个候选项集can：  检查一下can是否是tran的子集：  如果是，则增加can的计数值  对每个候选项：  如果其支持度不低于最小值，则保留该项集  返回所有频繁项集列表 |

整个apriori算法的伪代码如下表所示。

|  |
| --- |
| 当集合中项的个数大于0时  构建一个k个项组成的候选项集的列表  检查数据以确认每个项集都是频繁的  保留频繁项集并构建k+1项组成的候选项集的列表 |

（3）从频繁项集中挖掘关联规则

0123→

012→3

013→2

023→1

123→0

01→23

02→13

03→12

13→02

12→03

23→01

0→123

1→023

2→013

3→012

因为P(012)≤P(12)那么P(0123)/P(12)≤P(0123)/P(012)，即P(03|12)≤P(3|012)，即如果012→3置信度小于最小置信度，那么12→03也会小于最小置信度，同理上图的所有阴影部分均小于最小置信度。不失去一般性，如果A→B的置信度小于最小置信度，那么对于一个集合C，那么C→(A∪B-C)的置信度也会小于最小置信度。

对于每一个频繁项集，将的集合大小为*m*的子集记，记。

计算项集间的关联性的过程如下表所示：

|  |
| --- |
| 对每一个频繁项集  计算的置信度，然后筛选出置信度大于等于最小置信度的关联规则，并将相应的项集记为。  对于 n 从2到k  将两两组合构成集合大小为n的，计算的置信度，然后筛选出置信度大于等于最小置信度的关联规则，并将相应的项集，记为。重复这个步骤直到*n == k*。 |

## 3.函数说明

**(1)createC1函数**

生成项的个数是1的项集列表。

createC1 (dataSet)

* 参数
* dataSet：list类型，其每个元素也是list类型
* 返回值
* 返回一个列表。列表的每个元素的类型是frozenset类型，相当于一个不可修改的列表，表示项集。

**(2)scanD函数**

计算项集的支持度，返回频繁项集合，及所有项集的支持度

scanD(D, Ck, minSupport)

* 参数
* D：list类型，数据集合
* Ck：list类型，项集列表
* 返回值
* retList：list类型，频繁项集列表
* supportData：字典，键是项集，值是项集的支持度

**(3)aprioriGen函数**

生成项的个数是k的所有项集。

aprioriGen (Lk, k)

* 参数
* Lk：list类型，项集列表。
* k：项的个数。
* 返回值
* retList：list类型，每个元素是项的个数为k的项集。

**(4)apriori函数**

apriori算法函数

apriori(dataSet, minSupport = 0.5)

* 参数
* dataSet：list类型，二维的列表。
* minSupport：最小支持度，默认值为0.5。
* 返回值
* L：list类型，第一个元素是项个数为1的项集组成的列表，第二个元素是项个数为2的项集组成的列表，以此类推。
* supportData：dict类型，键是项集，值是支持度。

**(5)generateRules函数**

generateRules生成关联规则

generateRules(L, supportData, minConf=0.7)

* 参数
* L：list类型，每个元素是一个频繁项集。
* supportData：所有项集的支持度。
* minConf：最小置信度。
* 返回值
* bigRuleList：list类型，每个元素是一个含有3个元素的列表，前两个元素是项集，第3个元素是第一个元素出现后出现第二个元素的置信度。

**(6)calcConf函数**

将H列表中存放的每个集合记做conseq，计算(freqSet -conseq) →conseq的置信度，并筛选出置信度大于等于最小置信度的关联规则。

calcConf(freqSet, H, supportData, brl, minConf=0.7)

* 参数
* freqSet：set类型，表示频繁项集。
* H：list类型，每个元素是一个项集。
* supportData：dict类型，键是项集，值是支持度。
* brl：list类型，每个元素是一个含有3个元素的列表，前两个元素是项集，第3个元素是第一个元素出现后出现第二个元素的置信度。brl是函数外对应的对象的引用。
* minConf：最小置信度。
* 返回值
* prunedH：list类型：置信度大于等于最小置信度的那些关联规则，对应的conseq存于该列表中。

**(7)rulesFromConseq函数**

计算freqSet的子集到相应的差集的置信度，并筛选出置信度大于等于最小置信度的关联规则。

rulesFromConseq(freqSet, H, supportData, brl, minConf=0.7)

* 参数
* freqSet：set类型，表示频繁项集。
* H：list类型，每个元素是一个项集。
* supportData：dict类型，键是项集，值是支持度。
* brl：list类型，每个元素是一个含有3个元素的列表，前两个元素是项集，第3个元素是第一个元素出现后出现第二个元素的置信度。brl是函数外对应的对象的引用。
* minConf：最小置信度。
* 返回值
* None

## 4.案例一，购物篮数据分析

### (1)读取数据

数据来源来自<http://fimi.ua.ac.be/data/> ，这里使用了一部分数据集。

retail.dat存储的数据以空格分隔，每一行代表一次购买的物品，如第二行表示顾客在一次交易中同时购买了32、38、39、48和1727号商品。

|  |
| --- |
| 260 4115 6897 9173 9493  32 38 39 48 1727  48 535 587 842 1016 1161 1768 2869 3273 4911 |

使用如下函数读取数据：

|  |
| --- |
| retailtSet = [line.strip().split() for line in open('retail.dat').readlines()] |

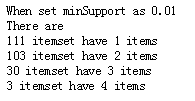
### (2)求频繁项集

|  |
| --- |
| # 支持度不小于0.01的频繁项集  L,suppData=apriori.apriori(retailtSet, minSupport=0.01) |

查看频繁项集的个数

|  |
| --- |
| print 'When set minSupport as', minSupport  print 'There are'  for alist in L:  if len(alist) > 0:  print len(alist), 'itemset', 'have', len(alist[0]), 'items' |

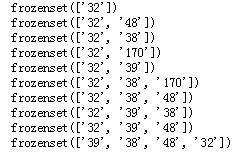
结果如下图所示：



查看包含’32’号商品的频繁项集：

|  |
| --- |
| for itemset\_list in L:  for itemset in itemset\_list:  if itemset.intersection({'32'}): print itemset |

结果如下图所示：

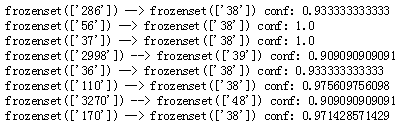


### (3)求关联规则

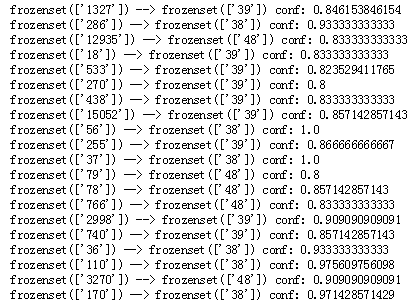
设置最小置信度为0.9，在(2)求得的频繁项集基础上，求满足条件的关联规则。

|  |
| --- |
| # 生成置信度不小于0.9的关联规则  rules=apriori.generateRules(L,suppData, minConf=0.9) |

结果如下图所示：



如果设置最小置信度为0.8时候，可以得到更多满足条件的关系，结果如下图所示：



## 5.实践任务

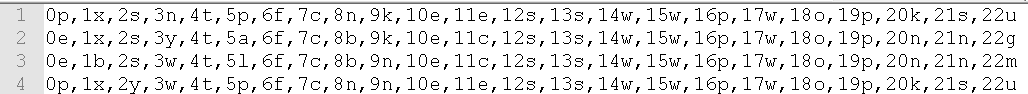
1. **双孢环柄菇毒性分析**

**a)数据来源:** [**https://archive.ics.uci.edu/ml/datasets/Mushroom**](https://archive.ics.uci.edu/ml/datasets/Mushroom)

‘agaricus-lepiota\_new.data’是整理后的双孢环柄菇数据，共有23个字段，各个字段的含义如下表所示。

|  |  |  |
| --- | --- | --- |
| 0 | classes | edible=e, poisonous=p |
| 1 | cap-shape | bell=b,conical=c,convex=x,flat=f,knobbed=k,sunken=s |
| 2 | cap-surface | fibrous=f,grooves=g,scaly=y,smooth=s |
| 3 | cap-color | brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y |
| 4 | bruises? | bruises=t,no=f |
| 5 | odor | almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s |
| 6 | gill-attachment | ttached=a,descending=d,free=f,notched=n |
| 7 | gill-spacing | lose=c,crowded=w,distant=d |
| 8 | gill-size | broad=b,narrow=n |
| 9 | gill-color | black=k,brown=n,buff=b,chocolate=h,gray=g,green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y |
| 10 | stalk-shape | enlarging=e,tapering=t |
| 11 | stalk-root | bulbous=b,club=c,cup=u,equal=e,hizomorphs=z,rooted=r,missing=? |
| 12 | stalk-surface-above-ring | fibrous=f,scaly=y,silky=k,smooth=s |
| 13 | stalk-surface-below-ring | fibrous=f,scaly=y,silky=k,smooth=s |
| 14 | stalk-color-above-ring | brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y |
| 15 | stalk-color-below-ring | brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y |
| 16 | veil-type | partial=p,universal=u |
| 17 | veil-color | brown=n,orange=o,white=w,yellow=y |
| 18 | ring-number | none=n,one=o,two=t |
| 19 | ring-type | cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z |
| 20 | spore-print-color | black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y |
| 21 | population | abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y |
| 22 | habitat | grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d |

数据的格式如下图所示，每个数据以逗号隔开，值由两部分组成，数字表示字段的索引，字母是特征值的简写，与上表对应。例如‘0p’,‘0’表示‘0’号字段，‘p’是该字段的属性值，是poisonous的简写，表示该蘑菇有毒。如果该值含有？，说明是缺失值。



**b)实验要求:**

1)设置一个最小支持度，记录下包含‘0p’(毒蘑菇)的支持度前10大的频繁项集。

2)设置两个不同的最小支持度，对于相同的最小置信度，分别找出置信度前10大且目标项集包含‘0p’（毒蘑菇）的关系。例如有关系['17w', '7c']-->['0p', '16p']，将目标项集定义为伴随出现的项集，这个例子的目标项集是['0p', '16p']。