lab4_homework

2018 年 10 月 4 日

1 lab4 实验报告

姓名: 李广泓学号:16369031 学院: 资讯管理学院专业: 信息管理与信息系统

1.1 实验目的

利用 Python 实现 Apriori 算法的应用.

1.2 实验环境

- windows 10 64 位
- anaconda 3
- jupyter notebook

1.3 实验步骤

1.3.1 利用 apriori 算法挖掘频繁集

```
In [1]: # 导入包
        import apriori

In [2]: dataSet = apriori.loadDataSet()
        dataSet

Out[2]: [[1, 2, 3, 4, 6], [2, 3, 4, 5, 6], [1, 2, 3, 5, 6], [1, 2, 4, 5, 6]]

In [3]: C1 = apriori.createC1(dataSet)

In [4]: print(C1)

[frozenset({1}), frozenset({2}), frozenset({3}), frozenset({4}), frozenset({5}), frozenset({6}
```

```
In [5]: D = list(map(set, dataSet))

In [6]: print(D)

[{1, 2, 3, 4, 6}, {2, 3, 4, 5, 6}, {1, 2, 3, 5, 6}, {1, 2, 4, 5, 6}]


In [7]: L1, supportData0 = apriori.scanD(D, C1, 0.5)

In [8]: print(L1)

[frozenset({5}), frozenset({6}), frozenset({4}), frozenset({3}), frozenset({2}), frozenset({1}


In [9]: L, supportData = apriori.apriori(dataSet)

In [10]: L[1]

Out[10]: [frozenset({1, 5}),
          frozenset({2, 5}),
          frozenset({3, 5}),
          frozenset({4, 5}),
          frozenset({5, 6}),
          frozenset({1, 2}),
          frozenset({1, 3}),
          frozenset({2, 3}),
          frozenset({1, 4}),
          frozenset({2, 4}),
          frozenset({3, 4}),
          frozenset({1, 6}),
          frozenset({2, 6}),
          frozenset({3, 6}),
          frozenset({4, 6})]

In [11]: supportData

Out[11]: {frozenset({1}): 0.75,
          frozenset({2}): 1.0,
          frozenset({3}): 0.75,
          frozenset({4}): 0.75,
```

```
frozenset({6}): 1.0,
frozenset({5}): 0.75,
frozenset({4, 6}): 0.75,
frozenset({3, 6}): 0.75,
frozenset({2, 6}): 1.0,
frozenset({1, 6}): 0.75,
frozenset({3, 4}): 0.5,
frozenset({2, 4}): 0.75,
frozenset({1, 4}): 0.5,
frozenset({2, 3}): 0.75,
frozenset({1, 3}): 0.5,
frozenset({1, 2}): 0.75,
frozenset({5, 6}): 0.75,
frozenset({4, 5}): 0.5,
frozenset({3, 5}): 0.5,
frozenset({2, 5}): 0.75,
frozenset({1, 5}): 0.5,
frozenset({1, 2, 3}): 0.5,
frozenset({1, 2, 4}): 0.5,
frozenset({1, 2, 6}): 0.75,
frozenset({1, 3, 4}): 0.25,
frozenset({1, 3, 6}): 0.5,
frozenset({2, 3, 4}): 0.5,
frozenset({2, 3, 6}): 0.75,
frozenset({1, 4, 6}): 0.5,
frozenset({2, 4, 6}): 0.75,
frozenset({3, 4, 6}): 0.5,
frozenset({2, 3, 5}): 0.5,
frozenset({2, 4, 5}): 0.5,
frozenset({2, 5, 6}): 0.75,
frozenset({3, 4, 5}): 0.25,
frozenset({3, 5, 6}): 0.5,
frozenset({4, 5, 6}): 0.5,
frozenset({1, 2, 5}): 0.5,
frozenset({1, 3, 5}): 0.25,
frozenset({1, 5, 6}): 0.5,
frozenset({1, 4, 5}): 0.25,
```

```
        frozenset({2, 3, 4, 6}): 0.5,
        frozenset({1, 2, 4, 6}): 0.5,
        frozenset({1, 2, 3, 6}): 0.5,
        frozenset({1, 2, 3, 4}): 0.25,
        frozenset({2, 4, 5, 6}): 0.5,
        frozenset({2, 3, 5, 6}): 0.5,
        frozenset({2, 3, 4, 5}): 0.25,
        frozenset({1, 2, 5, 6}): 0.5,
        frozenset({1, 2, 3, 5}): 0.25,
        frozenset({1, 2, 4, 5}): 0.25}
```

1.3.2 从频繁项集中挖掘关联规则

```
In [12]: rules = apriori.generateRules(L, supportData, minConf=0.7)
```

```
frozenset({5}) --> frozenset({2}) conf: 1.0
frozenset({2}) --> frozenset({5}) conf: 0.75
frozenset({6}) --> frozenset({5}) conf: 0.75
frozenset({5}) --> frozenset({6}) conf: 1.0
frozenset({2}) --> frozenset({1}) conf: 0.75
frozenset({1}) --> frozenset({2}) conf: 1.0
frozenset({3}) --> frozenset({2}) conf: 1.0
frozenset({2}) --> frozenset({3}) conf: 0.75
frozenset({4}) --> frozenset({2}) conf: 1.0
frozenset({2}) --> frozenset({4}) conf: 0.75
frozenset({6}) --> frozenset({1}) conf: 0.75
frozenset({1}) --> frozenset({6}) conf: 1.0
frozenset({6}) --> frozenset({2}) conf: 1.0
frozenset({2}) --> frozenset({6}) conf: 1.0
frozenset({6}) --> frozenset({3}) conf: 0.75
frozenset({3}) --> frozenset({6}) conf: 1.0
frozenset({6}) --> frozenset({4}) conf: 0.75
frozenset({4}) --> frozenset({6}) conf: 1.0
frozenset({1, 5}) --> frozenset({6}) conf: 1.0
frozenset({1, 5}) --> frozenset({2}) conf: 1.0
frozenset({4, 5}) --> frozenset({6}) conf: 1.0
frozenset({3, 5}) --> frozenset({6}) conf: 1.0
frozenset({5, 6}) --> frozenset({2}) conf: 1.0
```

```
frozenset({2, 6}) --> frozenset({5}) conf: 0.75
frozenset({2, 5}) --> frozenset({6}) conf: 1.0
frozenset({6}) --> frozenset({2, 5}) conf: 0.75
frozenset({5}) --> frozenset({2, 6}) conf: 1.0
frozenset({2}) --> frozenset({5, 6}) conf: 0.75
frozenset({4, 5}) --> frozenset({2}) conf: 1.0
frozenset({3, 5}) --> frozenset({2}) conf: 1.0
frozenset({3, 4}) --> frozenset({6}) conf: 1.0
frozenset({4, 6}) --> frozenset({2}) conf: 1.0
frozenset({2, 6}) --> frozenset({4}) conf: 0.75
frozenset({2, 4}) --> frozenset({6}) conf: 1.0
frozenset({6}) --> frozenset({2, 4}) conf: 0.75
frozenset({4}) --> frozenset({2, 6}) conf: 1.0
frozenset({2}) --> frozenset({4, 6}) conf: 0.75
frozenset({1, 4}) --> frozenset({6}) conf: 1.0
frozenset({3, 6}) --> frozenset({2}) conf: 1.0
frozenset({2, 6}) --> frozenset({3}) conf: 0.75
frozenset({2, 3}) --> frozenset({6}) conf: 1.0
frozenset({6}) --> frozenset({2, 3}) conf: 0.75
frozenset({3}) --> frozenset({2, 6}) conf: 1.0
frozenset({2}) --> frozenset({3, 6}) conf: 0.75
frozenset({3, 4}) --> frozenset({2}) conf: 1.0
frozenset({1, 3}) --> frozenset({6}) conf: 1.0
frozenset({2, 6}) --> frozenset({1}) conf: 0.75
frozenset({1, 6}) --> frozenset({2}) conf: 1.0
frozenset({1, 2}) --> frozenset({6}) conf: 1.0
frozenset({6}) --> frozenset({1, 2}) conf: 0.75
frozenset({2}) --> frozenset({1, 6}) conf: 0.75
frozenset({1}) --> frozenset({2, 6}) conf: 1.0
frozenset({1, 4}) --> frozenset({2}) conf: 1.0
frozenset({1, 3}) --> frozenset({2}) conf: 1.0
frozenset({1, 5, 6}) --> frozenset({2}) conf: 1.0
frozenset({1, 2, 5}) --> frozenset({6}) conf: 1.0
frozenset({1, 5}) --> frozenset({2, 6}) conf: 1.0
frozenset({3, 5, 6}) --> frozenset({2}) conf: 1.0
frozenset({2, 3, 5}) --> frozenset({6}) conf: 1.0
frozenset({3, 5}) --> frozenset({2, 6}) conf: 1.0
```

```
frozenset({4, 5, 6}) --> frozenset({2}) conf: 1.0
frozenset({2, 4, 5}) --> frozenset({6}) conf: 1.0
frozenset({4, 5}) --> frozenset({2, 6}) conf: 1.0
frozenset({1, 3, 6}) --> frozenset({2}) conf: 1.0
frozenset({1, 2, 3}) --> frozenset({6}) conf: 1.0
frozenset({1, 3}) --> frozenset({2, 6}) conf: 1.0
frozenset({1, 4, 6}) --> frozenset({2}) conf: 1.0
frozenset({1, 2, 4}) --> frozenset({6}) conf: 1.0
frozenset({1, 4}) --> frozenset({2, 6}) conf: 1.0
frozenset({3, 4, 6}) --> frozenset({2}) conf: 1.0
frozenset({2, 3, 4}) --> frozenset({6}) conf: 1.0
frozenset({3, 4}) --> frozenset({2, 6}) conf: 1.0
```

```
In [13]: # rules = apriori.generateRules(L, supportData, minConf=0.5)
```

1.3.3 利用 apriori 算法发现毒蘑菇的相似特征

```
In [14]: parsedDat = [line.split() for line in open('mushroom.dat').readlines()]
```

```
In [15]: parsedDat[1]
```

```
Out[15]: ['2',
          '3',
          '9',
          '14',
          '23',
          '26',
          '34',
          '36',
          '39',
          '40',
          '52',
          '55',
          '59',
          '63',
          '67',
          '76',
          '85',
```

```
          '86',
          '90',
          '93',
          '99',
          '108',
          '114']
```

```
In [16]: L, supportData = apriori.apriori(parsedDat, 0.4)
```

```
In [17]: L[3][0:2]
```

```
Out[17]: [frozenset({'1', '24', '34', '85'}), frozenset({'1', '24', '34', '86'})]
```

```
In [18]: for item in L[1]:
             if item.intersection('2'):
                 print(item)
```

```
frozenset({'28', '2'})
frozenset({'2', '34'})
frozenset({'2', '59'})
frozenset({'2', '63'})
frozenset({'2', '85'})
frozenset({'2', '86'})
frozenset({'90', '2'})
frozenset({'2', '39'})
```

```
In [19]: for item in L[2]:
             if item.intersection('2'):
                 print(item)
```

```
frozenset({'28', '2', '85'})
frozenset({'90', '2', '86'})
frozenset({'90', '2', '85'})
frozenset({'90', '2', '59'})
frozenset({'90', '2', '34'})
frozenset({'90', '2', '39'})
frozenset({'2', '39', '86'})
frozenset({'2', '39', '85'})
frozenset({'2', '85', '86'})
```

```
frozenset({'2', '63', '85'})
frozenset({'2', '39', '59'})
frozenset({'2', '59', '86'})
frozenset({'85', '2', '59'})
frozenset({'2', '39', '34'})
frozenset({'2', '34', '86'})
frozenset({'85', '2', '34'})
frozenset({'59', '2', '34'})
```

```
In [20]: # rules = apriori.generateRules(L, supportData, minConf=1.0)
```

1.4 实验感想

在本次实验中, 我第一次使用了 apriori 算法进行实验, 但是在实验进行的途中出现了异常的除 0 错误, 因此我就去阅读了源文件的代码. 在排除错误的同时我也对 apriori 算法的过程加深了理解, 对应每行代码上我都加上了注释. 最后我发现出除 0 错误是因为在 python3 中,map 函数不会像 python2 中一样返回一个列表, 而是会返回一个一次性的迭代器, 因此在进行除数操作的时候由于迭代器只能使用一次的原因会显示为除 0 错误. 解决问题很简单, 只要显式的将 map 结果转为列表即可.

1.5 操作习题

挖掘毒蘑菇的相似特征是, 给定不同支持度, 查看结果

```
In [21]: L, supportData = apriori.apriori(parsedDat, 0.8)
         L[3]
```

```
Out[21]: [frozenset({'34', '36', '85', '86'}), frozenset({'34', '85', '86', '90'})]
```

挖掘毒蘑菇的相似特征时, 给定不同置信度, 查看结果

```
In [22]: rules = apriori.generateRules(L, supportData, minConf=1.0)
```

```
frozenset({'34'}) --> frozenset({'85'}) conf: 1.0
frozenset({'36'}) --> frozenset({'85'}) conf: 1.0
frozenset({'86'}) --> frozenset({'85'}) conf: 1.0
frozenset({'90'}) --> frozenset({'85'}) conf: 1.0
frozenset({'90', '86'}) --> frozenset({'85'}) conf: 1.0
frozenset({'90', '86'}) --> frozenset({'34'}) conf: 1.0
```

```
frozenset({'90', '34'}) --> frozenset({'85'}) conf: 1.0
frozenset({'36', '86'}) --> frozenset({'85'}) conf: 1.0
frozenset({'34', '86'}) --> frozenset({'85'}) conf: 1.0
frozenset({'36', '34'}) --> frozenset({'86'}) conf: 1.0
frozenset({'36', '34'}) --> frozenset({'85'}) conf: 1.0
frozenset({'36', '34', '86'}) --> frozenset({'85'}) conf: 1.0
frozenset({'34', '36', '85'}) --> frozenset({'86'}) conf: 1.0
frozenset({'36', '34'}) --> frozenset({'85', '86'}) conf: 1.0
frozenset({'90', '34', '86'}) --> frozenset({'85'}) conf: 1.0
frozenset({'90', '85', '86'}) --> frozenset({'34'}) conf: 1.0
frozenset({'90', '86'}) --> frozenset({'34', '85'}) conf: 1.0
```

实例操作说明 map(), issubset(), set(), forzenset() 的功能和用法

```
In [23]: # map
         def f(x):
             return x * x
         map(f, [1, 2, 3, 4, 5])
```

```
Out[23]: <map at 0x1e102c09940>
```

```
In [24]: # convert map to list
         list(map(f, [1, 2, 3, 4, 5]))
```

```
Out[24]: [1, 4, 9, 16, 25]
```

```
In [25]: # issubset and set
         set1 = set([1, 2, 3, 4, 5]) # convert others to set
         set2 = set([1, 2, 3])
         set2.issubset(set1) # test whether every element is set2 is in set1
```

```
Out[25]: True
```

```
In [26]: # frozenset
         a = frozenset([1, 2, 3, 4])
         a
```

```
Out[26]: frozenset({1, 2, 3, 4})
```

```
In [27]: # map and frozenset
         list(map(frozenset, [[1, 2], [2, 3], [3, 4]]))
```

```
Out[27]: [frozenset({1, 2}), frozenset({2, 3}), frozenset({3, 4})]
```