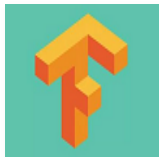


【Python数据挖掘课程】八.关联规则挖掘及Apriori实现购物推荐

原创 Eastmount 最后发布于2016-11-28 03:38:33 阅读数 25617 ☆ 收藏

编辑 展开



Python+TensorFlow人工智能

该专栏为人工智能入门专栏，采用Python3和TensorFlow实现人工智能相...



Eastmount

¥9.90

去订阅

这篇文章主要介绍三个知识点，也是我《数据挖掘与分析》课程讲课的内容。

- 1.关联规则挖掘概念及实现过程；
- 2.Apriori算法挖掘频繁项集；
- 3.Python实现关联规则挖掘及置信度、支持度计算。

前文推荐：

- 【Python数据挖掘课程】一.安装Python及爬虫入门介绍
- 【Python数据挖掘课程】二.Kmeans聚类数据分析及Anaconda介绍
- 【Python数据挖掘课程】三.Kmeans聚类代码实现、作业及优化
- 【Python数据挖掘课程】四.决策树DTC数据分析及鸢尾数据集分析
- 【Python数据挖掘课程】五.线性回归知识及预测糖尿病实例
- 【Python数据挖掘课程】六.Numpy、Pandas和Matplotlib包基础知识
- 【Python数据挖掘课程】七.PCA降维操作及subplot子图绘制

希望这篇文章对你有所帮助，尤其是刚刚接触数据挖掘以及大数据的同学，这些基础知识真的非常重要。如果文章中存在不足或错误的地方，还请海涵~

参考：

[关联规则挖掘之Apriori算法实现超市购物 - eastmount](#)

[关联规则简介与Apriori算法 - 百度文库guaidaoK](#)

一. 关联规则挖掘概念及实现过程

1.关联规则

关联规则 (Association Rules) 是反映一个事物与其他事物之间的相互依存性和关联性，如果两个或多个事物之间存在一定的关联关系，那么，其中一个事物就能通过其他事物预测到。关联规则是数据挖掘的一个重要技术，用于从大量数据中挖掘出有价值的数

据项之间的相关关系。

关联规则首先被Agrawal, Imielinski and Swami在1993年的SIGMOD会议上提出。

关联规则挖掘的最经典的例子就是沃尔玛的啤酒与尿布的故事，通过对超市购物篮数据进行分析，即顾客放入购物篮中不同商品之间的关系来分析顾客的购物习惯，发现美国妇女们经常会叮嘱丈夫下班后为孩子买尿布，30%-40%的丈夫同时会顺便购买喜爱的啤酒，超市就把尿布和啤酒放在一起销售增加销售额。有了这个发现后，超市调整了货架的设置，把尿布和啤酒摆放在一起销售，从而大大增加了销售额。

2.常见案例

前面讲述了关联规则挖掘对超市购物篮的例子，使用Apriori对数据进行频繁项集挖掘与关联规则的产生是一个非常有用的技术，其中我们众所周知的例子如：

- (1) 沃尔玛超市的尿布与啤酒
- (2) 超市的牛奶与面包
- (3) 百度文库推荐相关文档
- (4) 淘宝推荐相关书籍
- (5) 医疗推荐可能的治疗组合
- (6) 银行推荐相关联业务

这些都是商务智能和关联规则在实际生活中的运用。



3.置信度与支持度

(1) 什么是规则？

规则形如"如果...那么...(If...Then...)", 前者为条件，后者为结果。例如一个顾客，如果买了可乐，那么他也会购买果汁。

如何来度量一个规则是否够好？有两个量，置信度 (Confidence) 和支持度 (Support)，假如存在如下表的购物记录。

(2) 基本概念

关联规则挖掘是寻找给定数据集中项之间的有趣联系。如下图所示：

TID	List of item_ID's
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4

其中， $I = \{ I_1, I_2, \dots, I_m \}$ 是 m 个不同项目的集合,集合中的元素称为项目 (Item) 。

项目的集合 I 称为项目集合 (Itemset) , 长度为 k 的项集成为 k -项集 (k -Itemset) 。

设任务相关的数据 D 是数据库事务的集合, 其中每个事务 T 是项的集合, 使得 $T \subseteq I$ 。每个事务有一个标识符TID; 设 A 是一个项集, 事务 T 包含 A 当且仅当 $A \subseteq T$, 则关联规则形式为 $A \Rightarrow B$ (其中 $A \subseteq I$, $B \subseteq I$, 并且 $A \cap B = \emptyset$) , 交易集 D 中包含交易的个数记为 $|D|$ 。

在关联规则度量中有两个重要的度量值：支持度和置信度。

对于关联规则 $R: A \Rightarrow B$, 则：

支持度 (support)：是交易集中同时包含 A 和 B 的交易数与所有交易数之比。

$$\text{Support}(A \Rightarrow B) = P(A \cup B) = \text{count}(A \cup B) / |D|$$

置信度 (confidence)：是包含 A 和 B 交易数与包含 A 的交易数之比。

$$\text{Confidence}(A \Rightarrow B) = P(B|A) = \text{support}(A \cup B) / \text{support}(A)$$

(3) 支持度

支持度 (Support) 计算在所有的交易集中, 既有 A 又有 B 的概率。例如在5条记录中, 既有橙汁又有可乐的记录有2条。则此条规则的支持度为 $2/5 = 0.4$, 即：

$$\text{Support}(A \Rightarrow B) = P(AB)$$

现在这条规则可表述为, 如果一个顾客购买了橙汁, 则有50%(置信度)的可能购买可乐。而这样的情况 (即买了橙汁会再买可乐) 会有40% (支持度) 的可能发生。

顾客	项目
1	橙汁, 可乐
2	牛奶, 橙汁, 空气清洁器
3	橙汁, 洗洁精
4	橙汁, 洗洁精, 可乐
5	空气清洁器

(4) 置信度

置信度 (confidence) 表示了这条规则有多大程度上值得可信。设条件的项的集合为A,结果的集合为B。置信度计算在A中, 同时也含有B的概率 (即: if A ,then B的概率) 。即 :

$$\text{Confidence}(A \Rightarrow B) = P(B|A)$$

例如计算 “如果Orange则Coke” 的置信度。由于在含有 “橙汁” 的4条交易中, 仅有2条交易含有 “可乐” , 其置信度为0.5。

(5) 最小支持度与频繁集

发现关联规则要求项集必须满足的最小支持阈值, 称为**项集的最小支持度** (Minimum Support) , 记为supmin。支持度大于或等于supmin的项集称为频繁项集, 简称频繁集, 反之则称为非频繁集。通常k-项集如果满足supmin, 称为k-频繁集, 记作Lk。关联规则的最小置信度 (Minimum Confidence) 记为confmin, 它表示关联规则需要满足的最低可靠性。

(6) 关联规则

定义 关联规则

关联规则(Association Rule)可以表示为一个蕴含式:

$$R: X \Rightarrow Y$$

其中: $X \subset I$, $Y \subset I$, 并且 $X \cap Y = \Phi$ 。

例如: R: 牛奶 \rightarrow 面包

定义 关联规则的支持度

对于关联规则 $R: X \Rightarrow Y$ ，其中 $X \subset I$ ， $Y \subset I$ ，并且 $X \cap Y = \Phi$ 。

规则 R 的支持度(Support)是交易集中同时包含 X 和 Y 的交易数与所有交易数之比。

$$\text{support}(X \Rightarrow Y) = \frac{\text{count}(X \cup Y)}{|D|}$$

定义 关联规则的置信度

对于关联规则 $R: X \Rightarrow Y$ ，其中 $X \subset I$ ， $Y \subset I$ ，并且 $X \cap Y = \Phi$ 。

规则 R 的置信度(Confidence)是指包含 X 和 Y 的交易数与包含 X 的交易数之比

$$\text{confidence}(X \Rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

一般来说，只有支持度和置信度均较高的关联规则才是用户感兴趣的、有用的关联规则。

(7) 强关联规则

如果规则 $R: X \Rightarrow Y$ 满足 $\text{support}(X \Rightarrow Y) \geq \text{supmin}$ 且 $\text{confidence}(X \Rightarrow Y) \geq \text{confmin}$ ，称关联规则 $X \Rightarrow Y$ 为**强关联规则**，否则称关联规则 $X \Rightarrow Y$ 为**弱关联规则**。

在挖掘关联规则时，产生的关联规则要经过 supmin 和 confmin 的衡量，筛选出来的强关联规则才能用于指导商家的决策。

关联规则挖掘举例

假设最小值支持度为50%，最小置信度为50%

交易ID	购买商品
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

频繁项集	支持度
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

- 对于规则 $A \Rightarrow C$:
- 支持度 = $\text{support}(\{A, C\}) = 50\%$
- 置信度 = $\text{support}(\{A, C\}) / \text{support}(\{A\}) = 66.6\%$

规则 $A \Rightarrow C$ 满足最小支持度和最小置信度，所以它是强关联规则

二. Apriori算法挖掘频繁项集

关联规则对购物篮进行挖掘，通常采用两个步骤进行：

- 找出所有频繁项集（文章中我使用Apriori算法 \geq 最小支持度的项集）
- 由频繁项集产生强关联规则，这些规则必须大于或者等于最小支持度和最小置信度。

下面将通过超市购物的例子对关联规则挖掘Apriori算法进行分析。

Apriori算法是一种对有影响的挖掘布尔关联规则频繁项集的算法，通过算法的连接和剪枝即可挖掘频繁项集。

Apriori算法将发现关联规则的过程分为两个步骤：

- 通过迭代，检索出事务数据库中的所有频繁项集，即支持度不低于用户设定的阈值的项集；
 - 利用频繁项集构造出满足用户最小置信度的规则。
- 挖掘或识别出所有频繁项集是该算法的核心，占整个计算量的大部分。

Apriori算法的重要性质

假设项集{A,C}是频繁项集，则{A}和{C}也为频繁项集

▶ 性质1：频繁项集的子集必为频繁项集

▶ 性质2：非频繁项集的超集一定是非频繁的

假设项集{D}不是频繁项集，则{A,D}和{C,D}也不是频繁项集

补充频繁项集相关知识：

K-项集：指包含K个项的项集；

项集的出现频率：指包含项集的事务数，简称为项集的频率、支持度计数或计数；

频繁项集：如果项集的出现频率大于或等于最小支持度计数阈值，则称它为频繁项集，其中频繁K-项集的集合通常记作 L_k 。

下面直接通过例子描述该算法：如下图所示，使用Apriori算法关联规则挖掘数据集
中的频繁项集。（最小支持度计数为2）

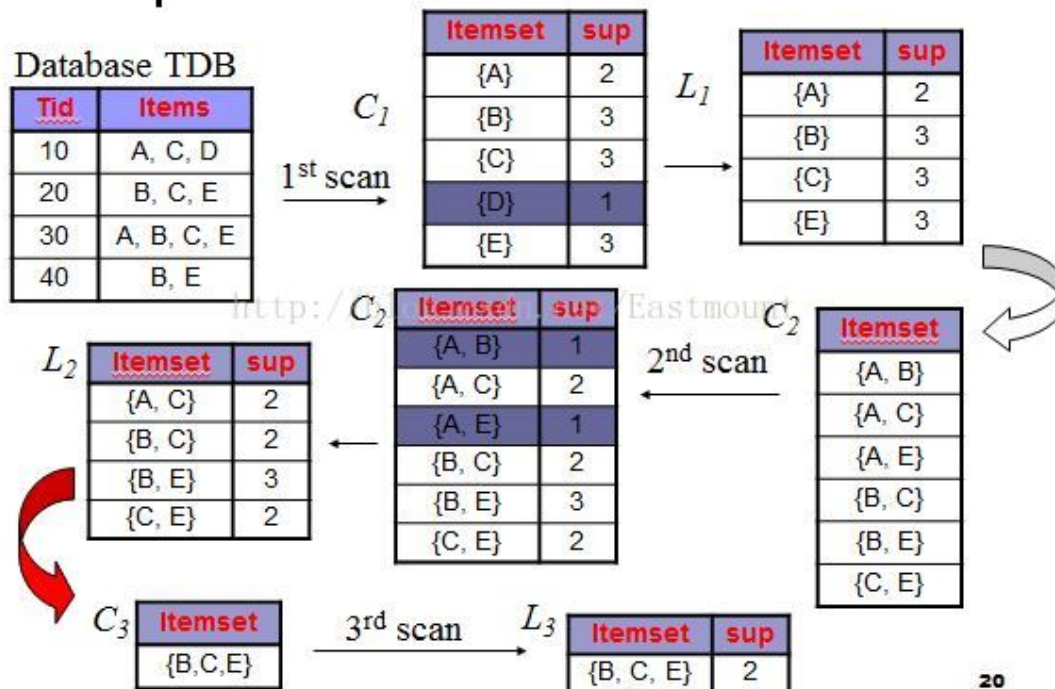
Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

具体过程如下所示：



最小支持度计数为2

Example



具体分析结果：

第一次扫描：对每个候选商品计数得 C_1 ，由于候选{D}支持度计数为1 < 最小支持度计数2，故删除{D}得频繁1-项集合 L_1 ；

第二次扫描：由 L_1 产生候选 C_2 并对候选计数得 C_2 ，比较候选支持度计数与最小支持度计数2得频繁2-项集合 L_2 ；

第三次扫描：用Apriori算法对 L_2 进行连接和剪枝产生候选3项集合 C_3 的过程如下：

1.连接：

$C_3 = L_2 \bowtie (\text{连接}) L_2 = \{\{A, C\}, \{B, C\}, \{B, E\}, \{C, E\}\} \bowtie \{\{A, C\}, \{B, C\}, \{B, E\}, \{C, E\}\} = \{\{A, B, C\}, \{A, C, E\}, \{B, C, E\}\}$

2.剪枝：

{A, B, C}的2项子集{A, B}, {A, C}和{B, C}，其中{A, B}不是2项子集 L_2 ，因此不是频繁的，从 C_3 中删除；

{A, C, E}的2项子集{A, C}, {A, E}和{C, E}，其中{A, E}不是2项子集 L_2 ，因此不是频繁的，从 C_3 中删除；

{B, C, E}的2项子集{B, C}, {B, E}和{C, E}，它的所有2项子集都是 L_2 的元素，保留 C_3 中。

经过Apriori算法对 L_2 连接和剪枝后产生候选3项集的集合为 $C_3 = \{B, C, E\}$ 。在该候选商品计数，由于等于最小支持度计数2，故得频繁3-项集合 L_3 ，同时由于4-项集中仅1个，故 C_4 为空集，算法终止。

三. 举例：频繁项集产生强关联规则

强关联规：如果规则R：X=>Y满足 $\text{support}(X \Rightarrow Y) \geq \text{supmin}$ (最小支持度，它用于衡量规则需要满足的最低重要性)且 $\text{confidence}(X \Rightarrow Y) \geq \text{confmin}$ (最小置信度，它表示关联规则需要满足的最低可靠性) 称关联规则X=>Y为强关联规则，否则称关联规则X=>Y为弱关联规则。

例子：

现有A、B、C、D、E五种商品的交易记录表，找出所有频繁项集，假设最小支持度 $\geq 50\%$ ，最小置信度 $\geq 50\%$ 。

对于关联规则R：A=>B，则：

支持度 (support)：是交易集中同时包含A和B的交易数与所有交易数之比。

$$\text{Support}(A \Rightarrow B) = P(A \cup B) = \text{count}(A \cup B) / |D|$$

置信度 (confidence)：是包含A和B交易数与包含A的交易数之比。

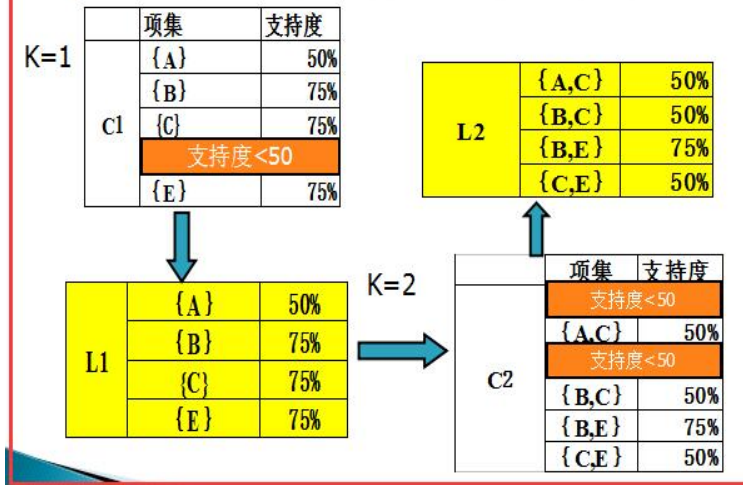
$$\text{Confidence}(A \Rightarrow B) = P(B|A) = \text{support}(A \cup B) / \text{support}(A)$$

交易号	商品代码
T1	A、C、D
T2	B、C、E
T3	A、B、C、E
T4	B、E

计算过程如下，K=1的时候项集{A}在T1、T3中出现2次，共4条交易，故支持度为 $2/4=50\%$ ，依次计算。其中项集{D}在T1出现，其支持度为 $1/4=25\%$ ，小于最小支持度50%，故去除，得到L1。

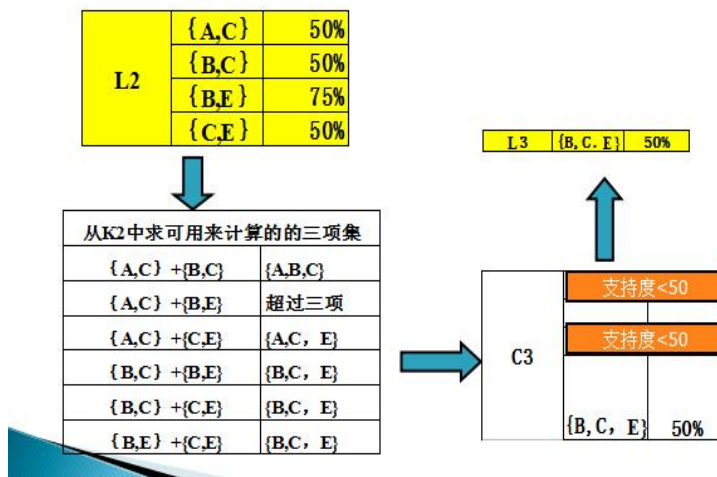
然后对L1中项集两两组合，再分别计算其支持度，其中项集{A, B}在T3中出现1次，其支持度 $=1/4=25\%$ ，小于最小支持度50%，故去除，同理得到L2项集。

Apriori算法举例_产生频繁项集



然后如下图所示，对L2中的项集进行组合，其中超过三项的进行过滤，最后计算得到L3项集{B, C, E}。

Apriori算法举例_产生频繁项集



最后对计算置信度，如下图所示。

Apriori算法举例_产生关联规则

- 对于频繁项集{B,C,E},它的非空子集有{B}、{C}、{E}、{B,C}、{B,E}、{C,E}。以下就是据此获得的关联规则及其置信度。

规则	置信度Confidence
B→CE	66.7%
C→BE	66.7%
E→BC	66.7%
CE→B	1
BE→C	66.7%
BC→E	1

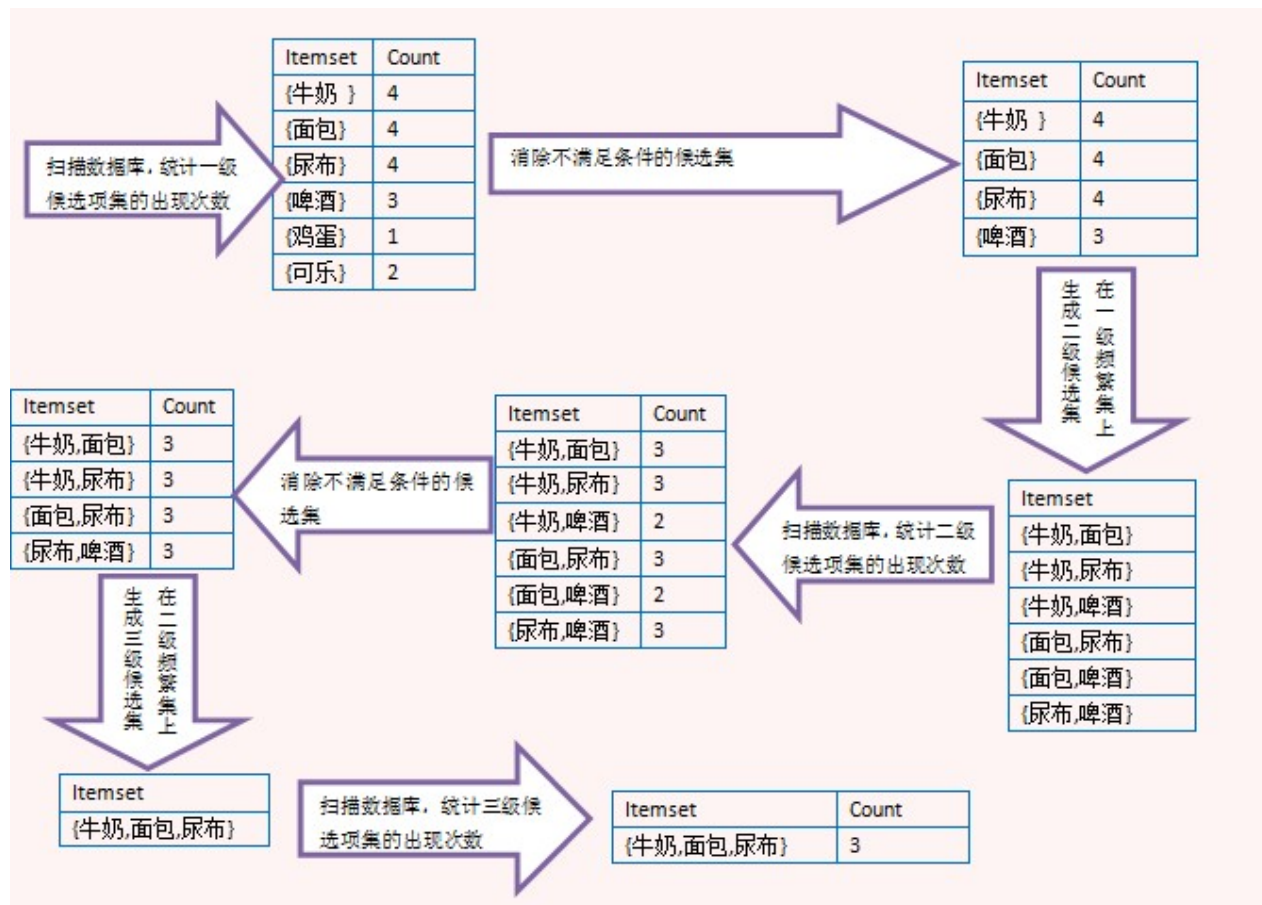
置信度≥50%(最小置信度),
都是强关联规则

Apriori算法弊端：需要多次扫描数据表。如果频繁集最多包含10个项，那么就需要扫描交易数据表10遍，这需要很大的I/O负载。同时，产生大量频繁集，若有100个项目，可能产生候选项数目。

$$C_{100}^1 + C_{100}^2 + \dots + C_{100}^{100} \approx 1.27 * 10^{30}$$

故：Jiawei Han等人在2000年提出了一种基于FP-树的关联规则挖掘算法FP_growth，它采取“分而治之”的策略，将提供频繁项目集的数据库压缩成一棵频繁模式树（FP-树）。

推荐一张图，详细分析关联规则的过程：



参考文献:

- [1]高明. 关联规则挖掘算法的研究及其应用[D].山东师范大学. 2006
- [2]李彦伟. 基于关联规则的数据挖掘方法研究[D].江南大学. 2011
- [3]肖劲橙, 林子禹, 毛超. 关联规则在零售商业的应用[J]. 计算机工程. 2004, 30(3): 189-190.
- [4]秦亮曦, 史忠植. 关联规则研究综述[J]. 广西大学学报. 2005, 30(4): 310-317.
- [5]陈志泊, 韩慧, 王建新, 孙俏, 聂耿青. 数据仓库与数据挖掘[M]. 北京: 清华大学出版社. 2009.
- [6]沈良忠. 关联规则中Apriori 算法的C#实现研究[J]. 电脑知识与技术. 2009, 5(13): 3501-3504.
- [7]赵卫东. 商务智能(第二版)[M]. 北京: 清华大学出版社. 2011.

四. Python实现关联规则挖掘及置信度、支持度计算

由于这部分代码在Sklearn中没有相关库, 自己后面会实现并替换, 目前参考空木大神的博客。地址: <http://blog.csdn.net/u010454729/article/details/49078505>

```
# -*- coding: utf-8 -*-
```

```
"""
```

```
Created on Mon Nov 28 03:29:51 2016
```

```
地址: http://blog.csdn.net/u010454729/article/details/49078505
```

```
@author: 参考CSDN u010454729
```

```
"""
```

```
# coding=utf-8
```

```
def loadDataSet():
```

```
    return [[1,3,4],[2,3,5],[1,2,3,5],[2,5]]
```

```
def createC1(dataSet):
```

```
#构建所有候选项集的集合
```

```
    C1 = []
```

```
    for transaction in dataSet:
```

```
        for item in transaction:
```

```
            if not [item] in C1:
```

```
                C1.append([item])
```

```
#C1添加的是列表, 对于每一项进行添加, {1},{3},
```

```
{4},{2},{5}
```

```
    C1.sort()
```

```
    return map(frozenset, C1)
```

```
#使用frozenset, 被“冰冻”的集合, 为后续建立字典key-value使用。
```

```
def scanD(D,Ck,minSupport):
```

```
#由候选项集生成符合最小支持度的项集L。参数分别
```

```
为数据集、候选项集列表, 最小支持度
```

```
    ssCnt = {}
```

```
    for tid in D:
```

```
#对于数据集里的每一条记录
```

```
        for can in Ck:
```

```
#每个候选项集can
```

```
            if can.issubset(tid):
```

```
#若是候选集can是作为记录的子集, 那么其值+1, 对
```

```
其计数
```

```
                if not ssCnt.has_key(can):#ssCnt[can] = ssCnt.get(can,0)+1一句可
```

```
破, 没有的时候为0, 加上1, 有的时候用get取出, 加1
```

```
                ssCnt[can] = 1
```

```
            else:
```

```
                ssCnt[can] +=1
```

```
            numItems = float(len(D))
```

```
    retList = []
```

```
    supportData = {}
```

```
    for key in ssCnt:
```

```
        support = ssCnt[key]/numItems
```

```
#除以总的记录条数, 即为其支持度
```

```
        if support >= minSupport:
```

```
            retList.insert(0,key)
```

```
#超过最小支持度的项集, 将其记录下来。
```

```
            supportData[key] = support
```

```
    return retList, supportData
```

```
def aprioriGen(Lk, k):
```

```
#创建符合置信度的项集Ck,
```

```
    retList = []
```

```
    lenLk = len(Lk)
```

```
    for i in range(lenLk):
```



```

        for j in range(i+1, lenLk):      #k=3时, [:k-2]即取[0], 对{0,1},{0,2},{1,2}
这三个项集来说, L1=0, L2=0, 将其合并得{0,1,2}, 当L1=0,L2=1不添加,
            L1 = list(Lk[i])[:k-2]      L2 = list(Lk[j])[:k-2]
            L1.sort()                    L2.sort()

            if L1==L2:
                retList.append(Lk[i]|Lk[j])
    return retList

def apriori(dataSet, minSupport = 0.5):
    C1 = createC1(dataSet)
    D = map(set,dataSet)
    L1, supportData = scanD(D,C1,minSupport)
    L = [L1]                                #L将包含满足最小支持度, 即经过筛选的所有频繁n项
集, 这里添加频繁1项集
    k = 2
    while (len(L[k-2])>0):                  #k=2开始, 由频繁1项集生成频繁2项集, 直到下一个
打的项集为空
        Ck = aprioriGen(L[k-2], k)
        Lk, supK = scanD(D, Ck, minSupport)
        supportData.update(supK)            #supportData为字典, 存放每个项集的支持度, 并
以更新的方式加入新的supK
        L.append(Lk)                        k +=1        return L,supportData
dataSet = loadDataSet()
C1 = createC1(dataSet)
print "所有候选1项集C1:\n",C1

D = map(set, dataSet)
print "数据集D:\n",D

L1, supportData0 = scanD(D,C1, 0.5)
print "符合最小支持度的频繁1项集L1:\n",L1

L, suppData = apriori(dataSet)
print "所有符合最小支持度的项集L: \n",L
print "频繁2项集: \n",aprioriGen(L[0],2)
L, suppData = apriori(dataSet, minSupport=0.7)
print "所有符合最小支持度为0.7的项集L: \n",L

```

输出结果:

所有候选1项集C1:

```
[frozenset([1]), frozenset([2]), frozenset([3]), frozenset([4]),
frozenset([5])]
```

数据集D: [set([1, 3, 4]), set([2, 3, 5]), set([1, 2, 3, 5]), set([2, 5])]

符合最小支持度的频繁1项集L1:

```
[frozenset([1]), frozenset([3]), frozenset([2]), frozenset([5])]
```

所有符合最小支持度的项集L:

```
[[frozenset([1]), frozenset([3]), frozenset([2]), frozenset([5])],
```

```
[frozenset([1, 3]), frozenset([2, 5]),
```

```
frozenset([2, 3]), frozenset([3, 5])], [frozenset([2, 3, 5])], []] 频繁2项集:
```

```
[frozenset([1, 3]), frozenset([1, 2]), frozenset([1, 5]), frozenset([2, 3]),
```

```
frozenset([3, 5]), frozenset([2, 5])]
```

所有符合最小支持度为0.7的项集L:


```
[[frozenset([3]), frozenset([2]), frozenset([5])], [frozenset([2, 5])], []]
```

最后希望这篇文章对你有所帮助，尤其是我的学生和接触数据挖掘、机器学习的博友。星期天晚上和思华在办公室写到三点半，庆幸这么多可爱的学生，自己也在成长，经历很多终究是好事，最近沉醉某些事中，希望能成真！加油~

(By:Eastmount 2016-11-28 凌晨3点半 <http://blog.csdn.net/eastmount/>)

👍 点赞 16 ☆ 收藏 ➦ 分享



Eastmount  博客专家

发布了444 篇原创文章 · 获赞 5908 · 访问量 484万+