# 实验报告四

学号：20191060239

姓名：李品鑫

实验名称：关联规则挖掘算法实现及应用

【实验目标】：

实现Apriori算法产生频繁项集和强规则，并在Groceries数据集上进行应用。

【实验任务】：

1. 阐述Apriori算法产生频繁项集和强规则的原理及过程。

2. 编程实现Apriori算法（包括频繁项集产生和强规则生成）。

3. 设置支持度阈值min\_sup=0.01，置信度阈值min\_conf=0.3，在Groceries数据集上进行关联规则挖掘，并对挖掘结果进行分析（可列出全部或部分结果，挑选部分结果进行分析，可从挖掘到的频繁项集和强规则的含义等进行分析）。注：若因电脑性能不足，在上述阈值设置下不能跑出结果，可适当更改支持度阈值设置。

4. 改变支持度阈值和置信度阈值，观察Apriori算法生成的频繁项集数量和强规则数量，以及算法生成频繁项集所用时间和产生强规则所用时间。

5. 实验总结与思考。

【实验要求】：

（1）独立完成实验，拒绝抄袭。撰写实验报告，需包含实验目的，实验原理及过程，实验结果展示等。

（2）对实验任务中的1和2，可根据个人能力选做其中一个或两个都做，任务1的难度低于任务2。（评分会考虑任务难度，鼓励有余力的同学两个算法都实现）。

（3）可根据自己能力借助合适的工具（如sklearn库）完成实验，但需清楚算法原理。鼓励自己动手编程实现整个算法，评分时，自己动手编程实现>借助工具。

【实验过程】：

1. 阐述Apriori算法产生频繁项集和强规则的原理及过程。

Apriori算法的基本思想是通过对数据库的多次扫描来计算项集的支持度，发现所有的频繁项集从而生成关联规则。基本步骤如下

1. 首先，扫描一次事务集合，找出频繁1-项集集合L1

2. 基于L1，产生所有可能频繁的2-项集，即候选2-项集集合C2（连接）

3. 基于L1，优化C2（剪枝），基于Apriori性质过滤掉一些候选

4. 基于C2，再扫描一次事务集合，找出频繁2-项集集合L2。

依次类推，直至不能找到频繁项集为止。最后，在所有频繁项集中产生强关联规则。

2. 编程实现Apriori算法（包括频繁项集产生和强规则生成）。

该Apriori算法封装在一个matlab函数中，函数输入有两个，分别为布尔矩阵data、商品名列表item。布尔矩阵data中行表示事务，列表示项。当某一事务包含某一项时，对应列即为1，反之为0。即data(i, j)=1时，表示第i个事务中包含第j项。输出为频繁项集以及强关联规则。

代码Apriori.m如下：

function Apriori(data, item)

min\_sup=input("ÇëÊäÈë×îÐ¡Ö§³Ö¶È\n"); % ×îÐ¡Ö§³Ö¶È(Î´³ýÒÔn)

min\_con=input("ÇëÊäÈë×îÐ¡ÖÃÐÅ¶È\n"); % ×îÐ¡ÖÃÐÅ¶È(ÒÑ³ýÒÔn)

[n,m]=size(data);

for i=1:n

x{i}=find(data(i,:)==1); % ÇóÃ¿ÐÐ¹ºÂòÉÌÆ·µÄ±àºÅ

end

k=0;

while 1

k=k+1;

L{k}={};

if k==1

C{k}=(1:m)';

else

[nL,mL]=size(L{k-1});

cnt=0;

for i=1:nL

for j=i+1:nL

tmp=union(L{k-1}(i,:),L{k-1}(j,:)); % Á½¼¯ºÏ²¢¼¯

if length(tmp)==k

cnt=cnt+1;

C{k}(cnt,1:k)=tmp;

end

end

end

C{k}=unique(C{k},'rows'); % È¥µôÖØ¸´µÄÐÐ

end

[nC,mC]=size(C{k}); % ºòÑ¡¼¯´óÐ¡

for i=1:nC

cnt=0;

for j=1:n

if all(ismember(C{k}(i,:),x{j}),2)==1 % allº¯ÊýÅÐ¶ÏÏòÁ¿ÊÇ·ñÈ«Îª1£¬²ÎÊý2±íÊ¾°´ÐÐÅÐ¶Ï

cnt=cnt+1;

end

end

C\_sup{k}(i,1)=cnt; % Ã¿ÐÐ´æºòÑ¡¼¯¶ÔÓ¦µÄÖ§³Ö¶È

end

L{k}=C{k}(C\_sup{k}>=(9835\*min\_sup),:);

if isempty(L{k}) % Õâ´ÎÃ»ÓÐÕÒ³öÆµ·±Ïî¼¯

break;

end

if size(L{k},1)==1 % Æµ·±Ïî¼¯ÐÐÊýÎª1£¬ÏÂÒ»´ÎÎÞ·¨Éú³ÉºòÑ¡¼¯£¬Ö±½Ó½áÊø

k=k+1;

C{k}={};

L{k}={};

break

end

end

fprintf("\n");

for i=1:k

fprintf("µÚ%dÂÖµÄºòÑ¡¼¯Îª:",i); C{i}

fprintf("µÚ%dÂÖµÄÆµ·±¼¯Îª:",i); L{i}

end

fprintf("µÚ%dÂÖ½áÊø£¬×î´óÆµ·±Ïî¼¯Îª:",k); L{k-1}

[nL,mL]=size(L{k-1});

rule\_count=0;

for p=1:nL % µÚp¸öÆµ·±¼¯

L\_last=L{k-1}(p,:); % Ö®ºó½«L\_last·Ö³É×óÓÒÁ½¸ö²¿·Ö£¬±íÊ¾¹æÔòµÄÇ°¼þºÍºó¼þ

cnt\_ab=0;

for i=1:n

if all(ismember(L\_last,x{i}),2)==1 % allº¯ÊýÅÐ¶ÏÏòÁ¿ÊÇ·ñÈ«Îª1£¬²ÎÊý2±íÊ¾°´ÐÐÅÐ¶Ï

cnt\_ab=cnt\_ab+1;

end

end

len=floor(length(L\_last)/2);

for i=1:len

s=nchoosek(L\_last,i); % Ñ¡i¸öÊýµÄËùÓÐ×éºÏ

[ns,ms]=size(s);

for j=1:ns

a=s(j,:);

b=setdiff(L\_last,a);

[na,ma]=size(a);

[nb,mb]=size(b);

cnt\_a=0;

for i=1:na

for j=1:n

if all(ismember(a,x{j}),2)==1 % allº¯ÊýÅÐ¶ÏÏòÁ¿ÊÇ·ñÈ«Îª1£¬²ÎÊý2±íÊ¾°´ÐÐÅÐ¶Ï

cnt\_a=cnt\_a+1;

end

end

end

pab=cnt\_ab/cnt\_a;

if pab>=min\_con % ¹ØÁª¹æÔòa->bµÄÖÃÐÅ¶È´óÓÚµÈÓÚ×îÐ¡ÖÃÐÅ¶È£¬ÊÇÇ¿¹ØÁª¹æÔò

rule\_count=rule\_count+1;

rule(rule\_count,1:ma)=a;

rule(rule\_count,ma+1:ma+mb)=b;

rule(rule\_count,ma+mb+1)=ma; % µ¹ÊýµÚ¶þÁÐ¼ÇÂ¼·Ö¸îÎ»ÖÃ(·Ö³É¹æÔòµÄÇ°¼þ¡¢ºó¼þ)

rule(rule\_count,ma+mb+2)=pab; % µ¹ÊýµÚÒ»ÁÐ¼ÇÂ¼ÖÃÐÅ¶È

end

cnt\_b=0;

for i=1:na

for j=1:n

if all(ismember(b,x{j}),2)==1 % allº¯ÊýÅÐ¶ÏÏòÁ¿ÊÇ·ñÈ«Îª1£¬²ÎÊý2±íÊ¾°´ÐÐÅÐ¶Ï

cnt\_b=cnt\_b+1;

end

end

end

pba=cnt\_ab/cnt\_b;

if pba>=min\_con % ¹ØÁª¹æÔòb->aµÄÖÃÐÅ¶È´óÓÚµÈÓÚ×îÐ¡ÖÃÐÅ¶È£¬ÊÇÇ¿¹ØÁª¹æÔò

rule\_count=rule\_count+1;

rule(rule\_count,1:mb)=b;

rule(rule\_count,mb+1:mb+ma)=a;

rule(rule\_count,mb+ma+1)=mb; % µ¹ÊýµÚ¶þÁÐ¼ÇÂ¼·Ö¸îÎ»ÖÃ(·Ö³É¹æÔòµÄÇ°¼þ¡¢ºó¼þ)

rule(rule\_count,mb+ma+2)=pba; % µ¹ÊýµÚÒ»ÁÐ¼ÇÂ¼ÖÃÐÅ¶È

end

end

end

pba=cnt\_ab/cnt\_b;

if pba>=min\_con % ¹ØÁª¹æÔòb->aµÄÖÃÐÅ¶È´óÓÚµÈÓÚ×îÐ¡ÖÃÐÅ¶È£¬ÊÇÇ¿¹ØÁª¹æÔò

rule\_count=rule\_count+1;

rule(rule\_count,1:mb)=b;

rule(rule\_count,mb+1:mb+ma)=a;

rule(rule\_count,mb+ma+1)=mb; % µ¹ÊýµÚ¶þÁÐ¼ÇÂ¼·Ö¸îÎ»ÖÃ(·Ö³É¹æÔòµÄÇ°¼þ¡¢ºó¼þ)

rule(rule\_count,mb+ma+2)=pba; % µ¹ÊýµÚÒ»ÁÐ¼ÇÂ¼ÖÃÐÅ¶È

end

end

end

end

fprintf("µ±×îÐ¡Ö§³Ö¶ÈÎª%d£¬×îÐ¡ÖÃÐÅ¶ÈÎª%.2fÊ±£¬Éú³ÉµÄÇ¿¹ØÁª¹æÔò£º\n",min\_sup,min\_con);

fprintf("Ç¿¹ØÁª¹æÔò\t\tÖÃÐÅ¶È\n");

[nr,mr]=size(rule);

for i=1:nr

pos=rule(i,mr-1); % ¶Ï¿ªÎ»ÖÃ£¬1:posÎª¹æÔòÇ°¼þ£¬pos+1:mr-2Îª¹æÔòºó¼þ

for j=1:pos

if j==pos

fprintf("%s",item{rule(i,j)});

else

fprintf("%s¡Ä",item{rule(i,j)});

end

end

fprintf(" => ");

for j=pos+1:mr-2

if j==mr-2

fprintf("%s",item{rule(i,j)});

else

fprintf("%s",item{rule(i,j)});

end

end

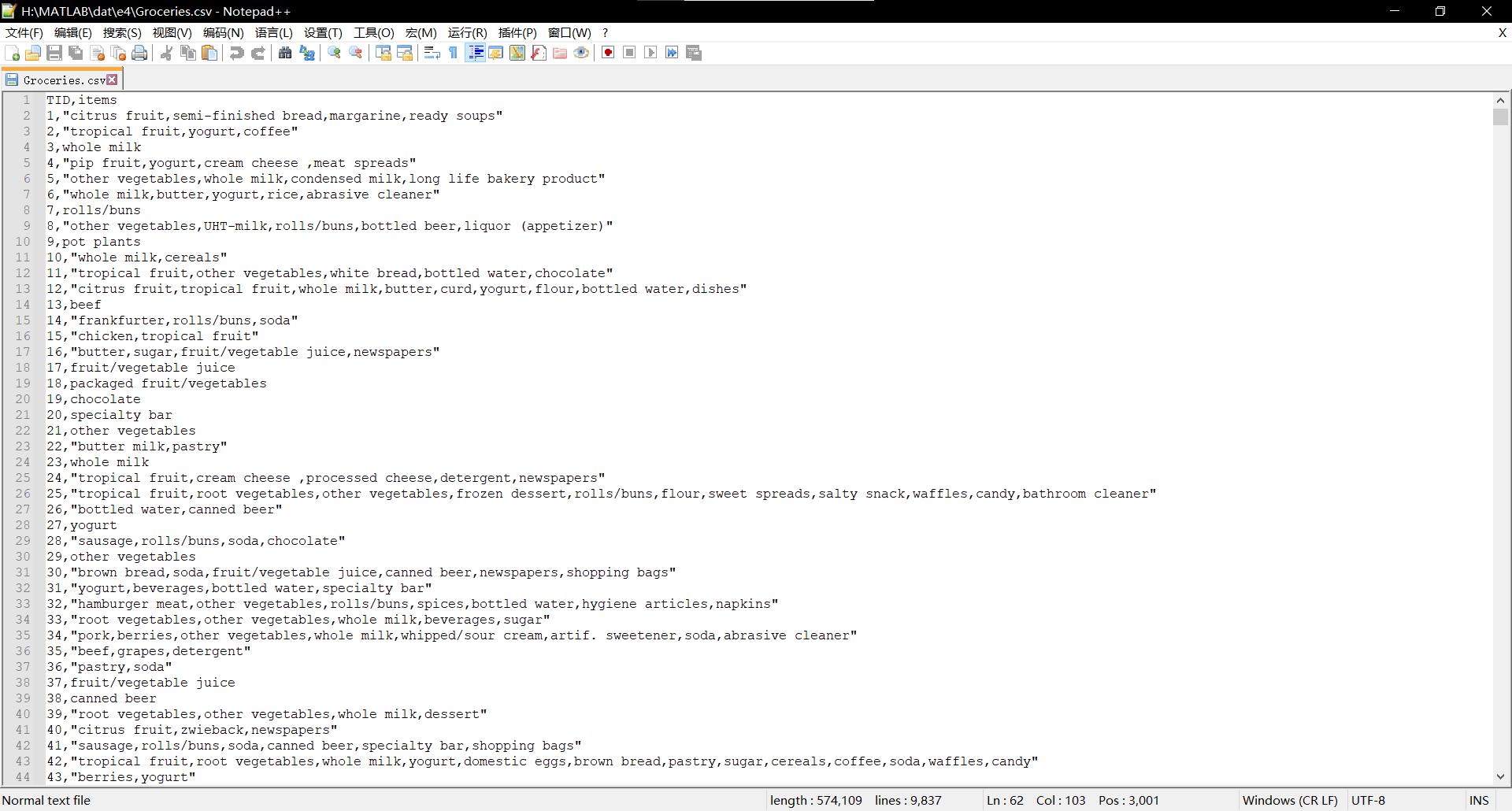
fprintf("\t\t%f\n",rule(i,mr));

end

end

3. 设置支持度阈值min\_sup=0.01，置信度阈值min\_conf=0.3，在Groceries数据集上进行关联规则挖掘，并对挖掘结果进行分析。

在对Groceries.csv数据集进行关联挖掘之前我们需要先对其数据集的结构进行观察和处理。利用Notepad++工具打开Groceries.csv文件，

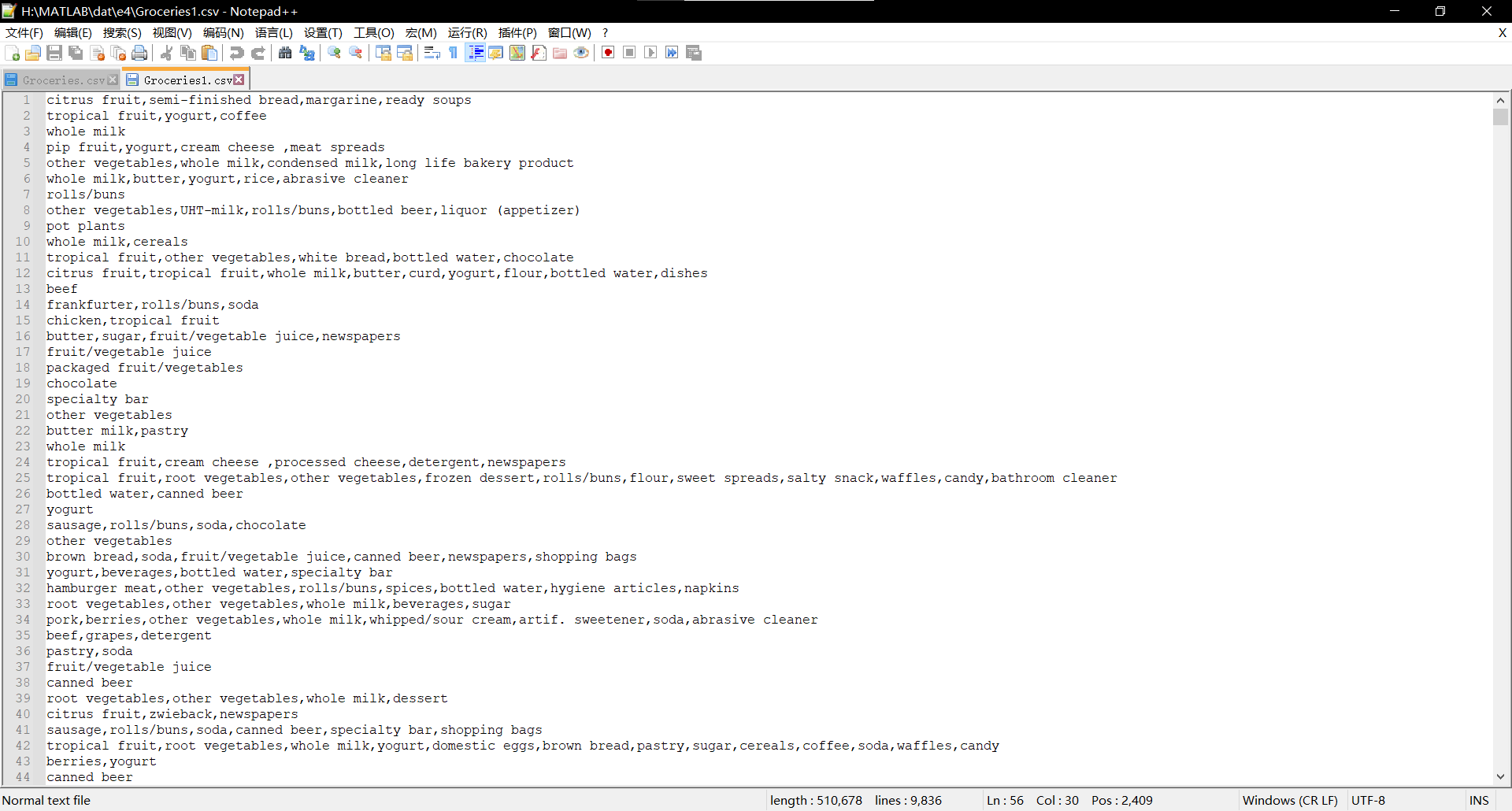


可以发现，该csv文件中第一行为”TID”与”item”标识，分别标记第一列为编号，第二列为所购买的物品。但是观察第二列发现第二列中数据样式并不标准、统一。例如以下两行数据

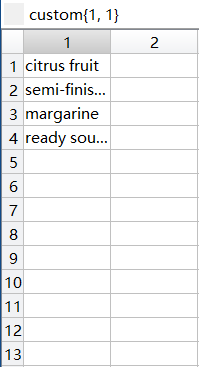
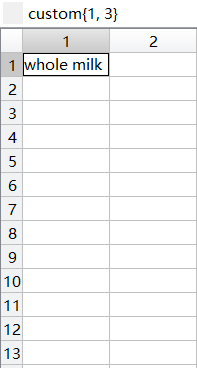
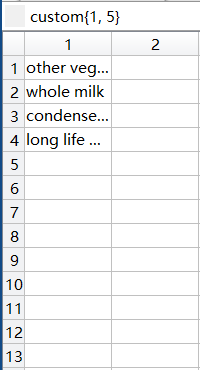
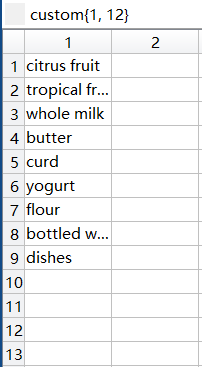


可以发现14号中多了一对引号（" "）将数据括起来，而13号没有。这样不统一的格式会对matlab中数据的读取与处理产生影响。并且csv文件是以逗号（,）作为分隔符的，但在14行中，逗号也用来作为多个商品间的分隔，这将会导致使用matlab的"csvread" "textscan "等函数读取csv文件时产生矩阵维度的混乱，进而报错。

因此，需要先使用Notepad++对其进行一些格式的处理，再送交matlab进行读取。处理后的数据集命名为Groceries1.csv，部分内容如下：



其中的变化是删去了第一行的编号以及引号（" "），这样便于matlab统一以按行与逗号分隔的方式读取数据。同时为避免读取数据时产生矩阵维度的混乱，采用了按行读取后存入一个数组中，再将该数组存入另一个数组之中的方式。在该算法中具体体现为以一个个商品数组存放顾客购买的商品数据，再采用了一个元胞数组custom存储不同顾客所对应的数组，而custom数组中的索引代表的则是顾客的编号。例如读取的1号、3号、5号、12号顾客数据对应的索引与商品列表如下：

代码main.m如下：

close all;

clear all;

clc;

filename = 'Groceries1.csv';

fileID = fopen(filename);

X = textscan(fileID,'%s','Delimiter',',');

item=unique(X{1,1});

fclose(fileID);

X = importdata(filename);

data=[];custom={};

for i =(1:length(X))

custom(1,i)=textscan(X{i,1},'%s','Delimiter',',');

end

for i =(1:length(custom))

for j=(1:length(custom{1,i}))

for k=(1:length(item))

if strcmp(custom{1,i}{j,1},item{k,1})

data(i,k)=1;

end

end

end

end

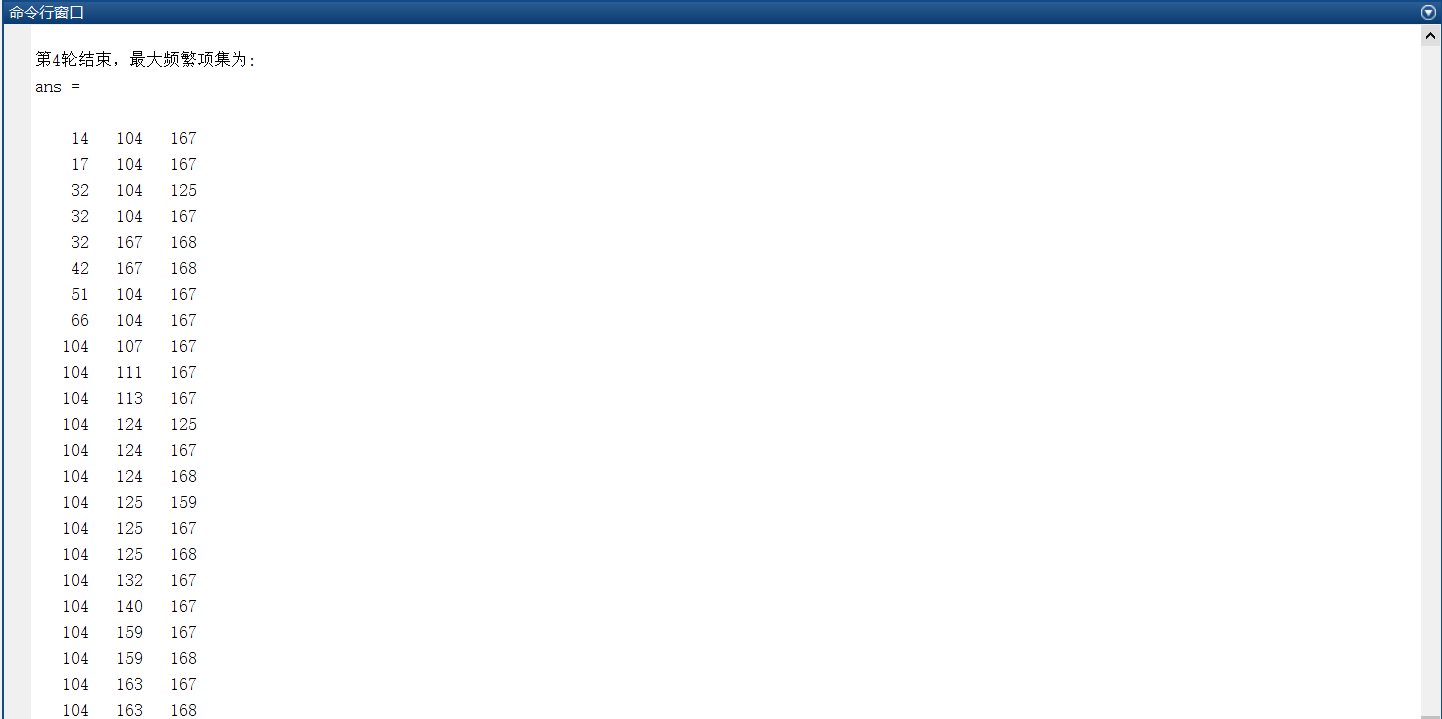
Apriori(data,item);

代码流程十分简单，仅对预处理后的数据集Groceries1.csv进行读取后调用先前编写好的Apriori函数，将读取的data与item送入其中进行处理。

输入最小支持度0.01与最小置信度0.3：

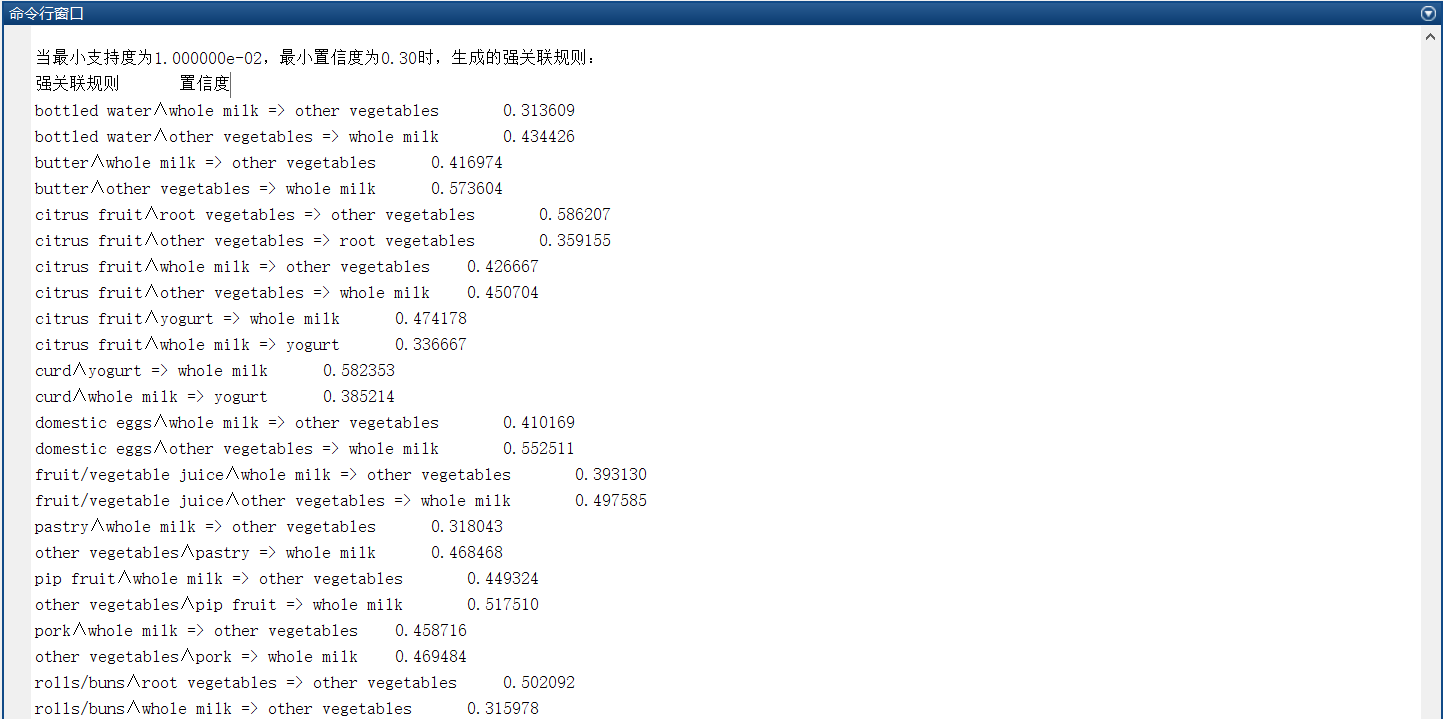


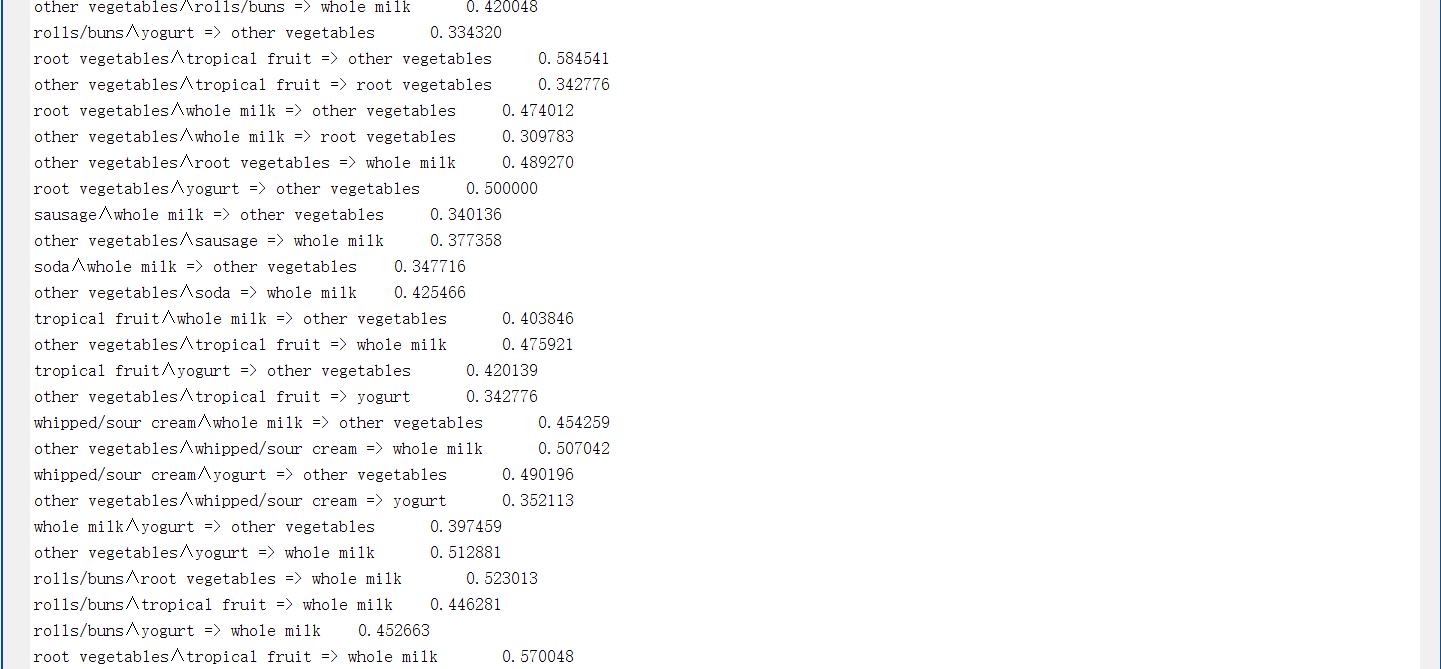
运行结果：  
（1） 输出的部分频繁项集（输出量过大故不作过多累赘展示）：





（2）最终挖掘出来的强关联规则

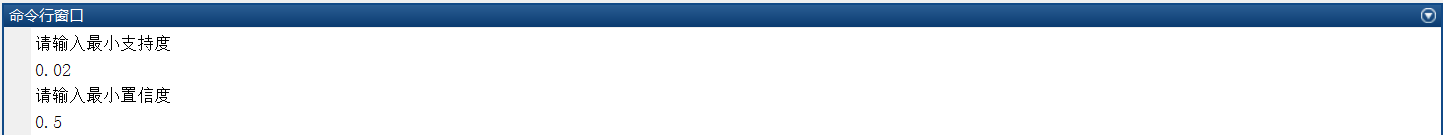




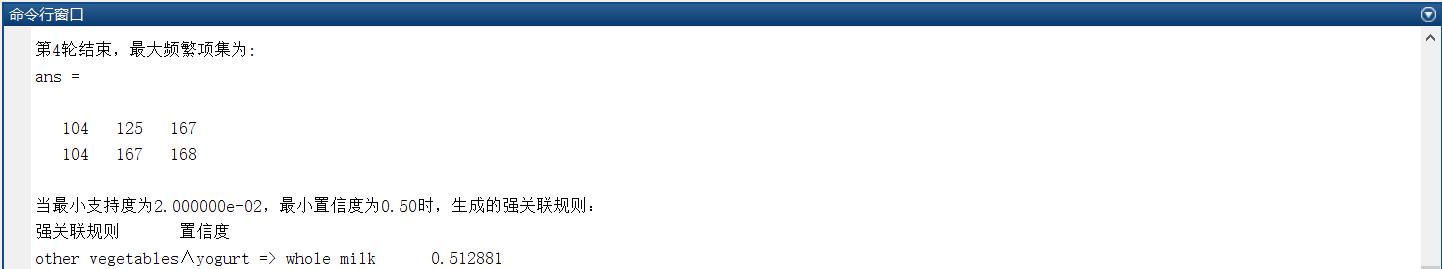


4. 改变支持度阈值和置信度阈值，观察Apriori算法生成的频繁项集数量和强规则数量，以及算法生成频繁项集所用时间和产生强规则所用时间。

根据Apriori算法原理，最小支持度为项出现次数占总项集的比例，最小置信度则为关联规则所需满足的最小的“概率”。由此可以推断，随着最小支持度与置信度的增大，对于项集的要求也就更为苛刻，满足该要求的频繁项集也越少。为了验证此猜想，将对最小支持度与置信度分别增大为0.02、0.5后再次运行代码。



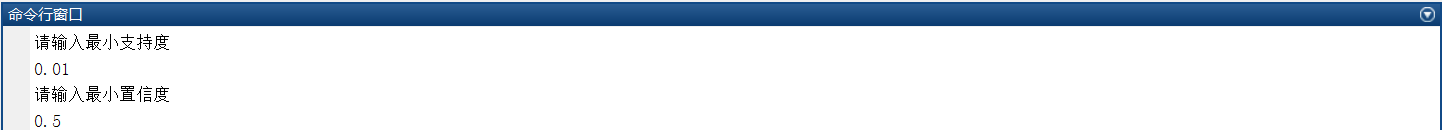
运行结果：



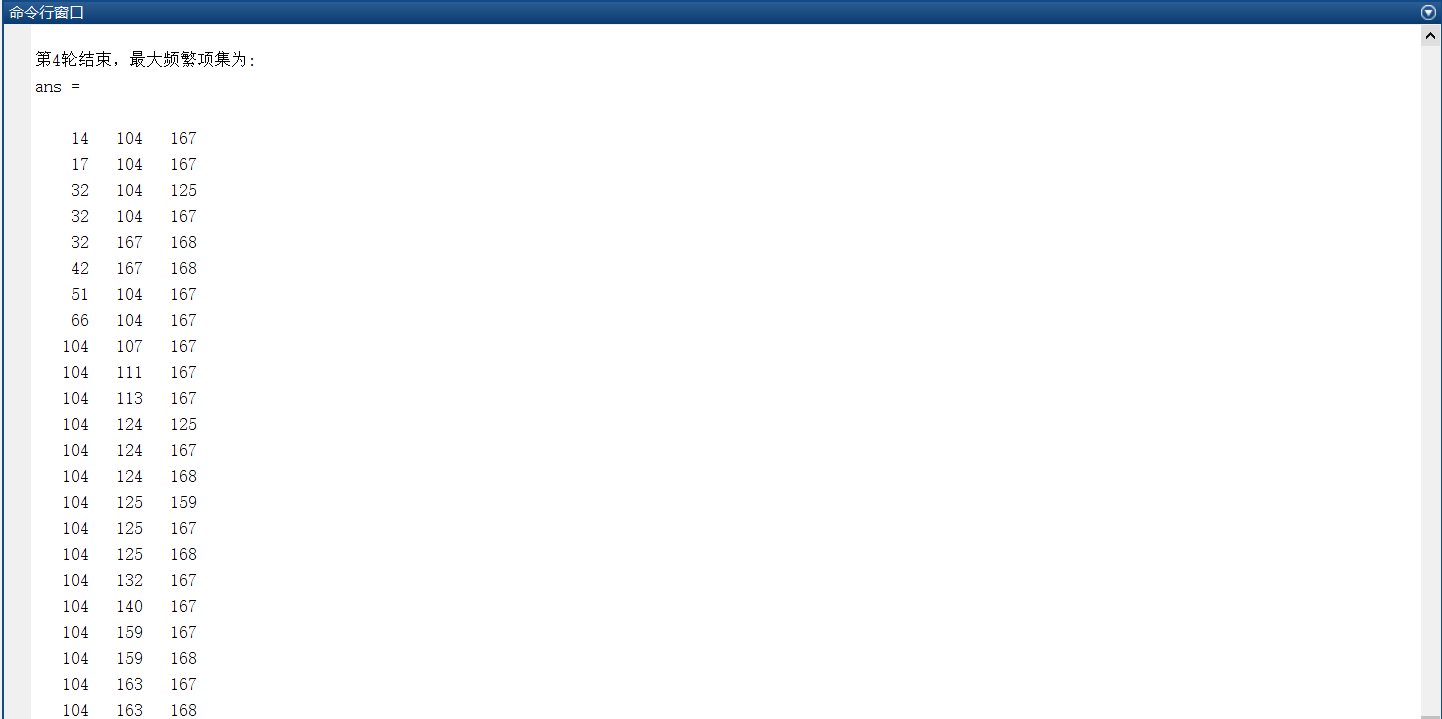
第4轮产生的最大频繁项集只剩两个，最终产生的强关联规则也只剩一条，且该规则置信度为0.512881，高于设定的阈值0.5。

为验证最小支持度与置信度各自的影响，分别进行只增大支持度和只增大置信度的实验：

（1）支持度不变，置信度增大



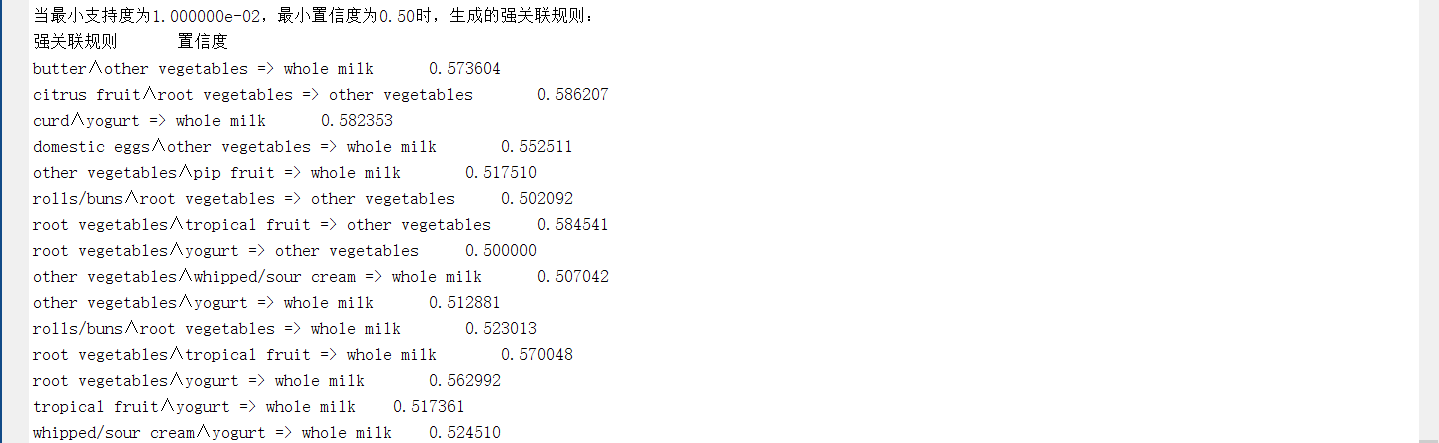
第四轮后的最大频繁项集





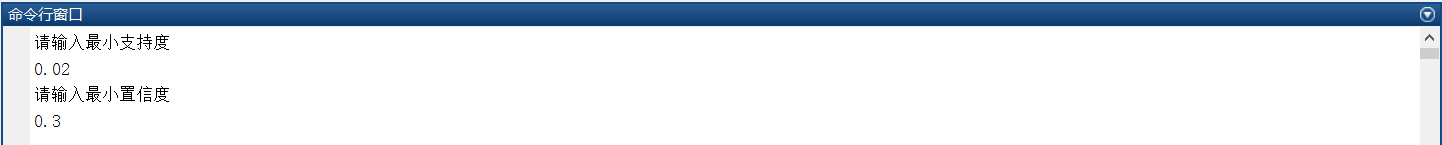
对比可以看出当支持度不变时，最大频繁项集也无变化

生成的强关联规则如下

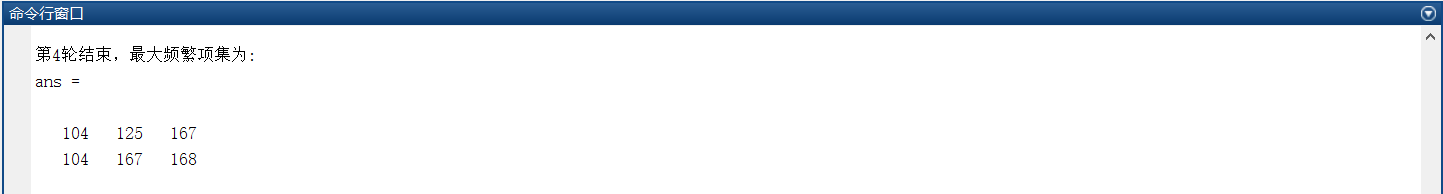


最小置信度变大，关联规则明显减少

（2）置信度不变，支持度增大



生成的最大频繁项集如下



生成的强关联规则如下



可以发现最小支持度的增大对最大频繁项集与强关联规则都产生了巨大的影响，两者都明显减少。

通过以上实验我们可以得知置信度的增大只对生产的强关联规则有影响，不影响最大频繁项集。而支持度的增大则会对两者都产生影响。这一现象是由于Apriori算法运行的原理：以最小支持度来筛选频繁项集，再根据最小置信度从频繁项集中选择出强关联规则。因此，当变动仅限于最小置信度时，并不会影响频繁项集的选择；而变动为最小支持度时，即使最小置信度不变，但由于支持度增大导致频繁项集的减少，部分能满足置信度的强关联规则由于它所属的频繁项集未能满足支持度要求而被淘汰，所能产生的强关联规则也随之变少。

5. 实验总结与思考。

该次实验相较于先前的实验，过程更复杂，难度更高，挑战更大。同时也不再是使用单一的matlab软件，而是使用多个软件，发挥他们不同的作用和长处，完成不同的实验部分。同时在实验中采用了多次重复实验，相互对比来得出结论的方式，结合理论推理与实验结果来更深刻理解算法的原理与运作流程。