Data Mining, Spring 2018

Problem Set #3: PCA, Recommender System, and Association Analysis

(Due on June 24, Sunday)

Submission Instructions

These questions require thought but do not require long answers. Please be as concise as possible. We do not do reverse engineering, so please DO NOT provide MATLAB (or other programming language) codes WITHOUT **method description**. You should also declare in the assignment that **the MATLAB (or other programming language) code was written by you, not by others either partially or fully**.

You should submit your answers as a write-up in PDF format to DataMining_2018@126.com. The email title is formatted as "hwk3_学号_姓名".

If you are in doubt, talk to me (majh8@mail.sysu.edu.cn) or our teaching assistants to understand more.

Questions

1. 主成分分析（Principal Component Analysis，PCA）

请从课程网站或此链接下载 Yale 人脸数据集进行降维。通过 MATLAB 命令 load('yale_face.mat')读取数据，包含一个4096×165矩阵。（请注意，这里的矩阵 X 是课件第 21 页定义的数据矩阵 X 的转置。）此矩阵的每一列是由一张64×64灰度人脸图像所转成的向量。例如，可以使用 imshow(reshape(X(:,1),[64 64]),[])命令显示第一张人脸图像（第一个训练样本）。

（1）试使用 MATLAB 中的 svd 函数实现 PCA 算法，即输入数据矩阵 X 和降维后的维数 k，对每一个样本进行去中心化，然后对进行去中心化后的数据矩阵 Xc 用 svd 函数[U,S,V] = svd(Xc)，输出降维的投影矩阵 Ureduce（即 U 的前 k 列），降维后的坐标表示 Z=Ureduce'*Xc，训练样本均值$\boldsymbol{\mu}$。并令 k=5，显示样本均值$\boldsymbol{\mu}$的图像和 Ureduce 的五个列向量（即协方差矩阵的前五个特征向量）所对应的图像；

答：使用 Matlab 中的 svd 函数实现 PCA 算法如下（**这里对均值归一化得到的 X_mean_norm 做 svd，而不是对协方差矩阵 sigma 做 svd**）：

```
load('D:\linjiafengyang\Code\Python\yale_face.mat')
% 均值图像
meanFace = mean(X, 2);
imshow(reshape(meanFace, [64 64]), []);
X_mean_norm = X - meanFace;
sigma = cov(X_mean_norm');
% svd函数
[U,S,V] = svd(X_mean_norm);
% 前五个特征向量
Ureduce = U(:,1:5);
for i = 1:5
    subplot(1,5,i);
    imshow(reshape(Ureduce(:,i), [64 64]), []);
end
z = Ureduce' * X_mean_norm;
```

均值图像如下：

前五个特征向量所对应的图像如下：

将 X 做均值归一化后得到 X_mean_norm，然后对 X_mean_norm 做 svd 得到的五个特征向量对应图像如下：

将 X 做均值归一化后得到 X_mean_norm，然后计算协方差矩阵，再对协方差矩阵做 svd 得到的五个特征向量对应图像如下：

可以看到对 X_mean_norm 或者协方差矩阵 sigma 做 svd 得到的特征向量基本没差别，但是对 sigma 做 svd 时花费的时间非常长，因此可以直接对均值归一化后得到的 X_mean_norm 做 svd，从而加快处理速度。

（2）试对协方差矩阵使用 MATLAB 中的 eig 函数计算特征值和特征向量，即[U,D]=eig(Xc*Xc'/m)，显示前五个最大的特征向量所对应的图像，并比较对数据矩阵使用 svd 函数的所得出的特征向量的图像与运算时间；

答：使用 Matlab 中的 eig 函数实现 PCA 算法如下（注意代码里协方差矩阵我使用的是 Matlab 自带的 cov 函数，和题目提示的 Xc*Xc'/m 的区别是分母不相同，cov 函数的计算公式是 Xc*Xc'/(m-1)，因为我在作业更新之前就做了，因此这里我就没再修改，反正结果差异不大，望 TA 体谅。）：

```
load('D:\linjiafengyang\Code\Python\PrincipalComponentAnalysis\yale_face.mat');
% 均值图像
meanFace = mean(X, 2);
% imshow(reshape(meanFace, [64 64]), []);
X_mean_norm = X - meanFace;
sigma = cov(X_mean_norm'); % 计算协方差
tic;
[V_eig,D] = eig(sigma); % V_eig为右特征向量，D为特征值
time_eig = toc;
D = diag(D); % 化为对角矩阵
V_eig = (rot90(V_eig))'; % 将特征向量矩阵从大到小排序
D = rot90(rot90(D)); % 将特征值矩阵从大到小排序
V_eig_reduce = V_eig(:,1:5); % 前五个特征向量
for i = 1:5
    subplot(1,5,i);
    imshow(reshape(V_eig_reduce(:,i), [64 64]), []);
end
Z_eig = V_eig_reduce' * X_mean_norm;
```

前五个最大的特征向量所对应的图像如下：

Svd 得到的五个特征向量如下：

	1	2	3	4	5
1	-0.0057	-0.0032	-0.0015	0.0075	-0.0124
2	-0.0049	-0.0039	-0.0012	0.0075	-0.0131
3	-0.0048	-0.0038	-7.1230e-...	0.0076	-0.0137
4	-0.0043	-0.0041	-1.1009e-...	0.0067	-0.0146
5	-0.0040	-0.0040	2.4347e-04	0.0058	-0.0155
6	-0.0030	-0.0045	4.0869e-04	0.0038	-0.0166
7	-0.0024	-0.0044	5.9662e-04	0.0017	-0.0177
8	-0.0018	-0.0044	0.0015	4.3733e-04	-0.0194
9	-0.0017	-0.0045	0.0030	-0.0017	-0.0211
10	-0.0021	-0.0047	0.0043	-0.0037	-0.0235
11	-0.0026	-0.0057	0.0060	-0.0048	-0.0258
12	-0.0020	-0.0061	0.0078	-0.0064	-0.0302
13	-0.0020	-0.0063	0.0117	-0.0059	-0.0317

Eig 得到的五个特征向量如下：

	1	2	3	4	5
1	0.0057	0.0032	-0.0015	0.0075	0.0124
2	0.0049	0.0039	-0.0012	0.0075	0.0131
3	0.0048	0.0038	-7.1230e-04	0.0076	0.0137
4	0.0043	0.0041	-1.1009e-04	0.0067	0.0146
5	0.0040	0.0040	2.4347e-04	0.0058	0.0155
6	0.0030	0.0045	4.0869e-04	0.0038	0.0166
7	0.0024	0.0044	5.9662e-04	0.0017	0.0177
8	0.0018	0.0044	0.0015	4.3733e-04	0.0194
9	0.0017	0.0045	0.0030	-0.0017	0.0211
10	0.0021	0.0047	0.0043	-0.0037	0.0235
11	0.0026	0.0057	0.0060	-0.0048	0.0258
12	0.0020	0.0061	0.0078	-0.0064	0.0302
13	0.0020	0.0063	0.0117	-0.0059	0.0317

可以发现特征向量的第一列、第二列、第五列成相反数，第三列、第四列相等；仔细对比两种方法生成的图像，也可以发现这一不同点。

运行时间：

若 svd 使用协方差矩阵进行奇异值分解，那么 svd 速度比 eig 慢很多，两者相比大致如下：（其中 time_eig 为使用 eig 函数的运行时间，time_svd 为使用 svd 函数的运行时间）

time_eig	12.6536
time_svd	50.1333

若 svd 使用均值归一化的数据矩阵进行奇异值分解，那么 svd 速度比 eig 快很多，两者相比大致如下：

time_eig	12.6536
time_svd	0.3608

（3）试计算当降维后的维数分别是 10 和 100 时，保留的方差的比例（即$1-\frac{\sum_{i=1}^{m}\left\|x^{(i)}-x_{approx}^{(i)}\right\|^2}{\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2}$

或使用（1）中的 S 计算$\sum_{i=1}^{k}S_{ii}^2/\sum_{i=1}^{m}S_{ii}^2$或使用（2）中的 D 计算$\sum_{i=1}^{k}D_{ii}/\sum_{i=1}^{m}D_{ii}$），并分别利用 10 维和 100 维坐标恢复原高维空间中的人脸图像，对前三张人脸图像，对比原图和两张恢复的图像。

答：采用如下公式计算保留的方差的比例：

$$\frac{\sum_{i=1}^{k}S_{ii}}{\sum_{i=1}^{m}S_{ii}}$$

编写程序如下（利用 svd 函数实现的 PCA 算法）：

```
X_approx = Ureduce * z;
ratio = 0;
S_diag = diag(S);
for i = 1:k
    r = S_diag(i)/sum(S_diag);
    ratio = ratio + r;
%     if (ratio >= 0.9)
%         break;
%     end
end
```

将 k 分别赋值为 10 和 100，得到保留的比例如下：

K=10 时，比例为 0.3207：

ratio	0.3207

K=100 时，比例为 0.8679：

ratio	0.8679

第一张图：（从左向右分别为原图、10 维恢复图、100 维恢复图，下同）

第二张图：

第三张图：

如果使用 eig 函数实现 PCA 算法，**将 k 分别赋值为 10 和 100，得到保留的比例如下：**

K=10 时，比例为 0.7281：

ratio	0.7281

K=100 时，比例为 0.9824：

ratio	0.9824

第一张图：（从左向右分别为原图、10 维恢复图、100 维恢复图，下同）



第二张图：



第三张图：



经过比较，可以发现维数越多，保留的方差的比例越高，图像的恢复程度越高，和原图最相似，且 eig 比 svd 效果好一点。

2. 推荐系统（Recommender System）

考虑以下的 8 个用户（A-H）对 7 部电影评级（1 到 5 级）的一个效用矩阵：

	A	B	C	D	E	F	G	H
HP1	4	4			1	1	5	
HP2	5	5		1				
HP3		4	1			1	5	4
TW	5		2	5		1	2	
SW1	1		5	4	5			1
SW2	1		5			4		
SW3		1		5		5	1	

电影的名字 HP1、HP2、HP3 分别代表《哈利波特》（Harry Potter）I、II、III，TW 代表《暮光之城》（Twilight），SW1、SW2 和 SW3 分别代表《星球大战》（Star Wars）I、II、III。

（1）试实现协同过滤算法（Collaborative filtering algorithm，课件第 19 页，不需要进行去均值操作），令正则化参数$\lambda = 0.1$，特征向量维数$n = 4$，学习率$\alpha = 0.01$，分别计算描述电影特征的7×4矩阵X和预测用户评级的8×4模型参数矩阵Θ（定义见课件第 23 页，所得结果保留小数点后 4 位），并计算预测电影评级的7×8效用矩阵即$X\Theta'$（保留小数点后 1 位）；

答：协同过滤算法大致如下：

```
def cost(X, Theta, Y, R, lamda, learning_rate, num_iterations):
    for i in range(0, num_iterations):
        # compute the cost
        error = np.multiply(np.dot(X, Theta.T) - Y, R)
        squared_error = np.power(error, 2)
        print("迭代次数 %d | 均方误差: %f" % (i+1, np.sum(squared_error)))
        J = (1 / 2) * np.sum(squared_error) + \
            ((lamda / 2) * np.sum(np.power(Theta, 2))) + \
            ((lamda / 2) * np.sum(np.power(X, 2)))
        # calculate the gradients with regularization
        X = X - learning_rate * ((error * Theta) + (lamda * X))
        Theta = Theta - learning_rate * ((error.T * X) + (lamda * Theta))
    return X, Theta
```

迭代多次基本达到收敛后：

下面结果不是固定的，因为参数随机初始化是不相同的，但最后得到的结果应该差不多。

特征矩阵 X 如下（保留小数点后四位）：

```
X为:
 [[ 1.3999  1.4757  0.1414  0.4173]
 [ 1.8985  1.5961  0.4295  0.3461]
 [ 1.2103  1.6909  0.3941  0.4404]
 [-0.4767  1.5737  1.718   0.5061]
 [-0.2611 -0.1071  0.8341  2.1552]
 [ 0.5716 -0.5346  1.002   1.6815]
 [ 0.5933 -0.4154  2.0775  1.3224]]
```

参数矩阵 Theta 如下（保留小数点后四位）：

```
Theta为:
 [[ 0.731   1.846   1.3429  0.1329]
 [ 1.4987  1.2002  0.1179  0.2806]
 [ 0.4712 -0.5099  1.2016  1.8805]
 [-0.5195  0.4988  1.9631  1.0532]
 [-0.1288  0.163   0.7865  1.9663]
 [ 1.0438 -0.7285  1.2554  1.0437]
 [ 1.6996  1.6007  0.0608  0.3997]
 [ 0.9351  1.4306  0.3612  0.5061]]
```

效用矩阵$\mathbf{X\Theta'}$如下（保留小数点后一位）：

```
效用矩阵为:
 [[4.  4.  0.9 0.7 1.  1.  4.9 3.7]
 [5.  4.9 1.2 1.  1.  1.7 5.9 4.4]
 [4.6 4.  1.  1.5 1.3 1.  5.  3.9]
 [4.9 1.5 2.  4.9 2.7 1.  2.  2.7]
 [1.  0.2 5.  4.  4.9 3.1 0.3 1. ]
 [1.  0.8 4.9 3.2 3.9 4.  0.8 1. ]
 [2.6 1.  5.5 5.  4.1 4.9 1.  1.4]]
```

（2）试计算（1）中预测的电影评级与真实评级的平方误差，即$\sum_{(i,j):\ r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2$，其中$r(i,j), y^{(i,j)}, \theta^{(j)}, x^{(i)}$的定义见课件第 7 页；讨论哪两部电影和 HP1 最相似，哪两部电影和和 SW1 最相似；

答：运行（1）中给出的代码可得均方误差大致如下：

```
迭代次数 1297 | 均方误差: 0.064398
迭代次数 1298 | 均方误差: 0.064397
迭代次数 1299 | 均方误差: 0.064396
迭代次数 1300 | 均方误差: 0.064396
```

根据（1）中得到的效用矩阵可知，**HP2、HP3 和 HP1 最相似，SW2、SW3 和 SW1 最相似**。因为每个用户对相似的电影的评级会很接近，即是说用户如果喜欢一部电影，那么他也会喜欢类似的电影；用户如果不喜欢一部电影，那么他也不会对类似的电影感兴趣。而效用矩阵很好地诠释了这一特点。

（3）试使用一个非零常数对协同过滤算法中的变量$x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$进行初始化（即更改课件第 19 页 Collaborative filtering algorithm 的第一步为$x^{(1)} = \cdots = x^{(n_m)} = \theta^{(1)} = \cdots = \theta^{(n_u)} = c\mathbf{1}$，其中$c$为非零实数，$\mathbf{1}$为所有元素都是 1 的$\mathrm{n} = 4$维列向量；使用（1）中相同的参数，分别计算描述电影特征的7×4矩阵X和预测用户评级的8×4模型参数矩阵Θ（所得结果保留小数点后 4 位），并计算预测电影评级的7×8效用矩阵即$\mathrm{X\Theta'}$（保留小数点后 1 位）和预测的电影评级与真实评级的平方误差，即$\sum_{(i,j):\ r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2$；与（1）和（2）中的结果比较，讨论此初始化方法的问题。

答：假设我们初始化 X 为所有元素都为 1 的7×4矩阵，Theta 为所有元素都为 1 的8×4矩阵，同样执行协同过滤算法，结果如下：

特征矩阵 X 如下（保留小数点后四位）：

```
X为:
 [[0.8917 0.8917 0.8917 0.8917]
 [0.8913 0.8913 0.8913 0.8913]
 [0.881  0.881  0.881  0.881 ]
 [0.8552 0.8552 0.8552 0.8552]
 [0.9573 0.9573 0.9573 0.9573]
 [1.0161 1.0161 1.0161 1.0161]
 [0.7248 0.7248 0.7248 0.7248]]
```

参数矩阵 **Theta** 如下（保留小数点后四位）：

```
Theta为:
 [[0.8311 0.8311 0.8311 0.8311]
 [1.0526 1.0526 1.0526 1.0526]
 [0.895  0.895  0.895  0.895 ]
 [1.0548 1.0548 1.0548 1.0548]
 [0.8181 0.8181 0.8181 0.8181]
 [0.664  0.664  0.664  0.664 ]
 [0.9906 0.9906 0.9906 0.9906]
 [0.6528 0.6528 0.6528 0.6528]]
```

效用矩阵$\mathbf{X\Theta'}$如下（保留小数点后一位）：

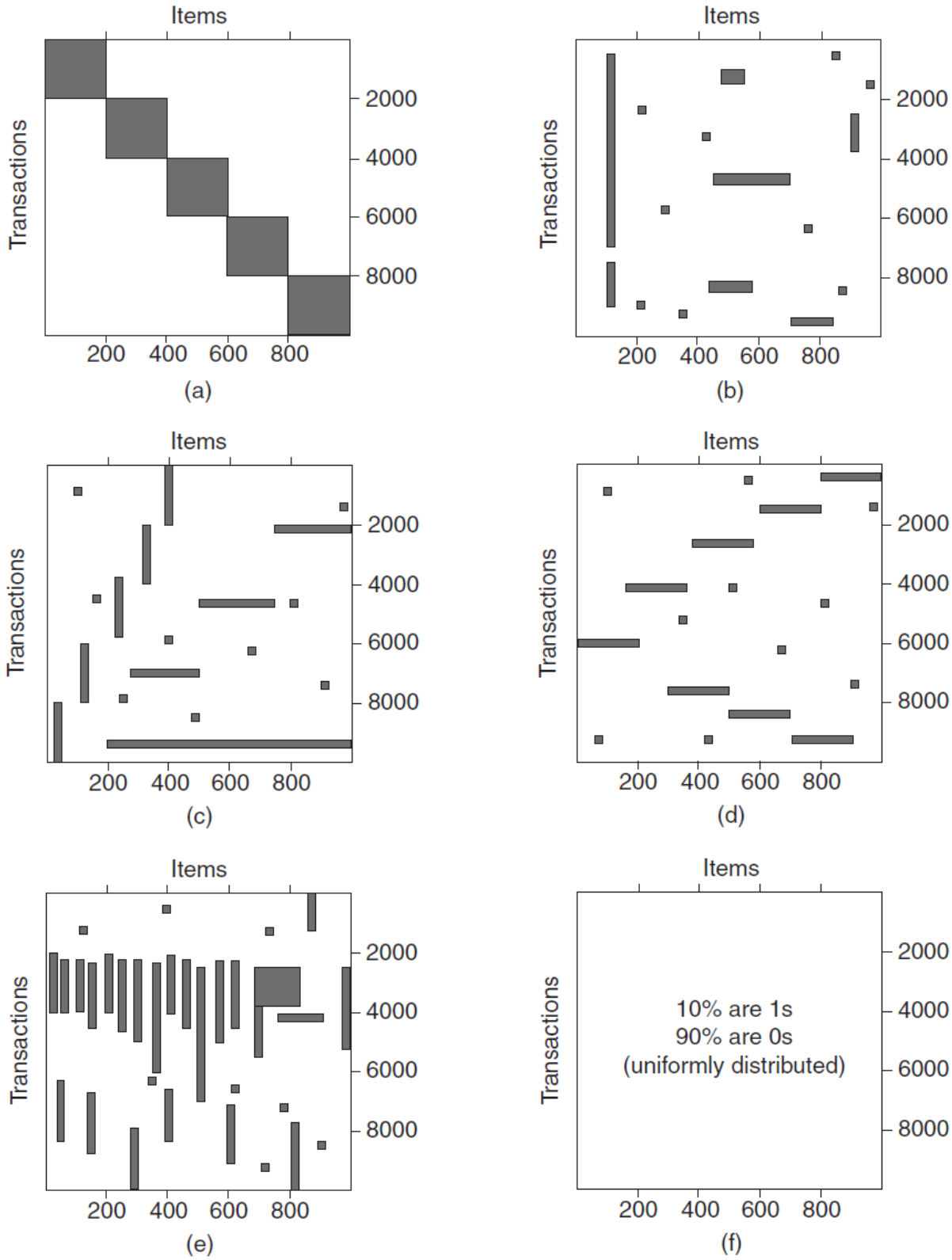
```
效用矩阵为:
 [[3.  3.8 3.2 3.8 2.9 2.4 3.5 2.3]
 [3.  3.8 3.2 3.8 2.9 2.4 3.5 2.3]
 [2.9 3.7 3.2 3.7 2.9 2.3 3.5 2.3]
 [2.8 3.6 3.1 3.6 2.8 2.3 3.4 2.2]
 [3.2 4.  3.4 4.  3.1 2.5 3.8 2.5]
 [3.4 4.3 3.6 4.3 3.3 2.7 4.  2.7]
 [2.4 3.1 2.6 3.1 2.4 1.9 2.9 1.9]]
```

均方误差大致如下：

```
迭代次数 219 | 均方误差: 88.225889
迭代次数 220 | 均方误差: 88.225889
迭代次数 221 | 均方误差: 88.225888
迭代次数 222 | 均方误差: 88.225888
迭代次数 223 | 均方误差: 88.225888
迭代次数 224 | 均方误差: 88.225888
迭代次数 225 | 均方误差: 88.225888
迭代次数 226 | 均方误差: 88.225888
```

与（1）（2）中的结果比较，可以发现将参数初始化为同一个参数会导致严重的问题，算法的效果非常差。随机初始化可以打破对称性，这和神经网络参数随机初始化的原因相似，随机初始化可以确保特征矩阵 **X** 和参数矩阵 **Theta** 在算法执行中保持不相同，而如果初始化为同一个参数在算法学习中难以达到梯度下降的优化效果，所以应避免这样的初始化。

3. 关联规则



上面六个图（a）到（f）中的每一个图包含 1000 个商品和 10000 个交易的记录。灰色位置表示存在商品交易，而白色表示不存在商品交易。我们使用 Apriori 算法提取频繁项集，并设定频繁项集的最小支持度为 10%，即 minsup=10%（即频繁项集包含在至少 1000 个交易中）。

根据上图回答以下问题：

（1）哪一个或几个数据集的频繁项集数目最多？哪一个或几个数据集的频繁项集数目最少？

答：a 数据集的频繁项集数目最多，d 数据集的频繁项集数目最少。

（2）哪一个或几个数据集的频繁项集长度最长（即包含最多商品）？

答：a 数据集的频繁项集长度最长，包含最多商品。

（3）哪一个或几个数据集的频繁项集有最高的最大支持度（highest maximum support）？

答：b 数据集的频繁项集有最高的最大支持度（highest maximum support）。

（4）哪一个或几个数据集的频繁项集有最大的支持度范围（例如频繁项集的支持度范围可以从小于 20%变化到大于 70%）？

答：b 数据集的频繁项集有最大的支持度范围。