K-Multiple-Means: A Multiple-Means Clustering Method with

Specified K Clusters

数据挖掘与数据仓库期末大作业
陈小莹 19214946
2019-12-31

1 论文信息

条目	内容
目标	改进论文
论文题目	K-Multiple-Means: A Multiple-Means Clustering Method with Specified K Clusters
论文作者	Feiping Nie--University of Texas at Arlington Cheng-Long Wang--Northwestern Polytechnical University
论文页数	10
会议名称	the 25th ACM SIGKDD International Conference
会议年份	2019
论文算法实现工具	matlab

2 论文核心思想

由于传统的kmeans聚类方法在对非凸集聚类时效果不是很好，本论文将传统的kmeans聚类方法改进成多均值的kmeans聚类法。

核心思想：多原型聚类。多原型的聚类方法原理是将数据集分成很小的子集，然后再根据相似度矩阵迭代地将相近的子集合并成较大的子集，最终生成k个聚类。该方法可以很好地对非凸数据集进行聚类。

主要方法：本论文结合了传统的Kmeans聚类，模糊C聚类，多原型聚类，连通图聚类多种聚类方法，并且引入拉普拉斯矩阵秩约束，使得相似度矩阵正好有k个连通分量。

3 论文主要思路

3.1 传统的kmeans聚类算法

一开始从n个数据集中随机选择k个数据点作为均值点（聚心），然后分别计算每个数据点和这k个聚心的欧氏距离，将与第i个聚心 (i<=k)距离近的那些数据点划分到第i个类，最终第一次生成k个聚类。接着再对新生成的这k个聚类算平均值得到新的聚心，再根据聚心和样本点的欧氏距离重新为每个类分配数据点，多次迭代进行，直到聚类收敛为止。

3.2 模糊C均值聚类算法

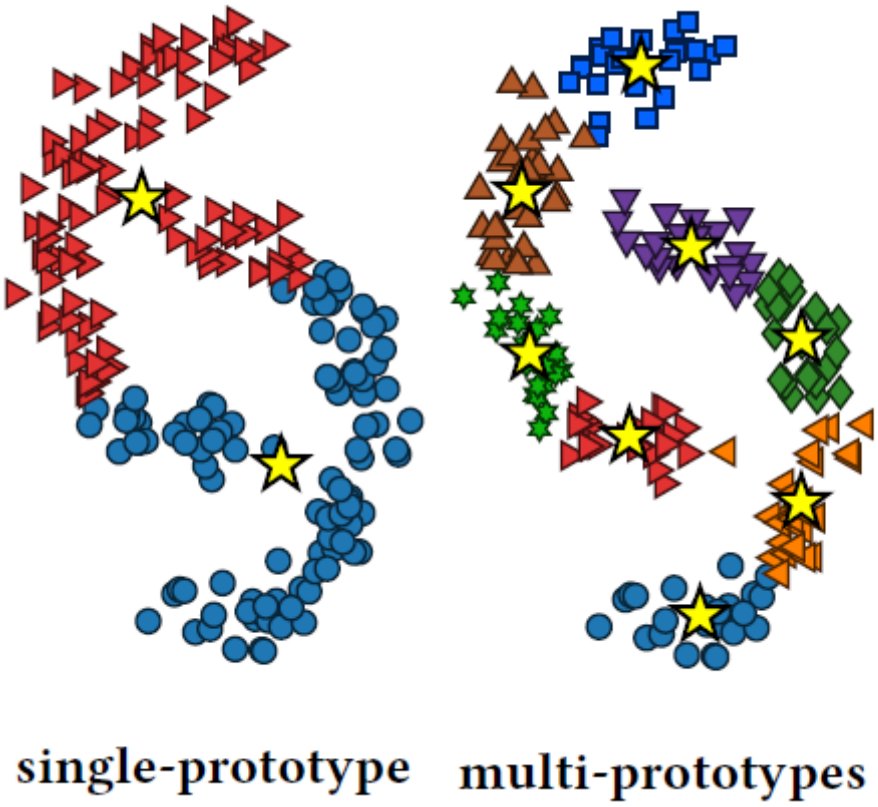
传统的kmeans聚类算法采取的是硬聚类方法，即样本点要么属于该类，要么不属于该类。软聚类则把数据以一定概率分到各类中，模糊C均值聚类就是采取这种软聚类的方法。

算法：在传统kmeans聚类方法的基础上加入权重，权值表示数据点i对聚心点l的依赖度。模糊C均值聚类算法的目标函数如下：

$$f\left(U,V\right)=\sum_{i=1}^{n}\sum_{l=1}^{k}u_r^{il}\left\|x_i-v_l\right\|_2^2 \quad (1)$$
$$U\geq 0,U1=1$$

3.3 引入多个原型点

不管是传统的kmeans聚类算法还是改进后的模糊C聚类算法，它们对球状的数据集聚类效果很好，但是对非凸数据集的聚类效果并不是很好。因此本论文提出了一种新思想，论文作者认为对于每个类，不能仅仅靠一个聚心收敛数据点，应该为每个类设置几个聚心，这些小聚心在本论文中被称作原型点。下图是每个类设单个原型点和每个类设多个原型点的聚类效果对比图：



从上面的效果对比图可以看出，为每个类设置多个原型点可以更好地为非凸数据集进行聚类。本论文设置了m=63个原型点。

3.4 引入相似度矩阵S

本论文正是从模糊C均值聚类算法中获得灵感，既然每个样本点都以一定的概率隶属于各个类，因此可以将这些隶属概率值sij提取出来，由sij组成n行m列的相似度矩阵S，矩阵中的元素代表n各个样本点对m个原型点的相似度。引入相似度矩阵后目标函数如下：

$$\begin{aligned} & min\sum_{i=1}^{n}\sum_{j=1}^{m} s_{ij}\left\|x_i-a_j\right\|_2^2+\gamma\|S\|_2^F \\ & S\geq 0, S1=1 \end{aligned} \quad (2)$$

当γ无限大时，公式（2）等价于公式（3），此时最优解是sij=1/m，即这m个原型点都以相同的概率1/m与样本点xi临近；

$$\begin{aligned} & min\sum_{j=1}^{m} s_{ij}^2 \\ & s_i^T 1=1, 0\leq s_{ij}\leq 1 \end{aligned} \quad (3)$$

当γ=0时，公式（2）等价于求公式（4）,此时最优解只有一个sij=1，其余为0，即只有一个距离最近的原型点a与样本点xi相邻，其余原型点与该样本点不相邻。

$$\begin{aligned} & min\sum_{j=1}^{m} s_{ij}\left\|x_i-a_j\right\|_2^2 \\ & s_i^T 1=1, 0\leq s_i\leq 1 \end{aligned} \quad (4)$$

γ是本论文用于控制相似度矩阵S稀疏度的控制参数。因此可以利用公式（2）为n个样本点分配邻近点，从而生成k个聚类。

3.5 引入拉普拉斯矩阵秩约束

但是实验中发现，仅仅利用公式（2）达到的聚类效果并不好。为了使得样本点能自动分配到k个类中，本论文为相似度矩阵S引入拉普拉斯矩阵的秩约束，关于拉普拉斯矩阵有以下定理：

```
Theorem 4.1. The multiplicity k of the eigenvalue 0 of the normalized
Laplacian matrix ˜LS is equal to the number of connected components in
the bipartite graph associated with S.
```

为相似度矩阵S构造拉普拉斯矩阵，通过拉普拉斯矩阵的秩约束限制相似度矩阵S只能有k个连通分量，从而实现将样本点分成k个聚类的效果。则公式（2）改为公式（5）：

$$\begin{aligned} &min\sum_{i=1}^{n}\sum_{j=1}^{m} s_{ij}\left\|x_i-a_j\right\|_2^2+\gamma\|S\|_2^F \\ &S\geq 0, S1=1, A\in R^{m\times d}, rank\left(\widetilde{L_S}\right)=(n+m)-k \end{aligned} \tag{5}$$

因为拉普拉斯矩阵的半正定性，当秩等于k时，拉普拉斯矩阵的前k个最小的特征值都为0。因此公式（5）又可以改写成公式（6）：

$$\begin{aligned} &min\sum_{i=1}^{n}\sum_{j=1}^{m} s_{ij}\left\|x_i-a_j\right\|_2^2+\gamma\|S\|_2^F+\lambda\sum_{i=1}^{k}\sigma_i\left(\widetilde{L_S}\right) \\ &S\geq 0, S1=1, A\in R^{m\times d} \end{aligned} \tag{6}$$

又因为矩阵的特征值之和等于矩阵的迹之和：

$$\begin{aligned} &\sum_{i=1}^{k}\sigma_i\left(\widetilde{L_S}\right)=minTr\left(F^T\,\widetilde{L_S}\,F\right) \\ &F\in R^{(n+m)\times k}, F^T\,F=1 \end{aligned}$$

故公式（6）又可以写成公式（7）：

$$\begin{aligned} &min\sum_{i=1}^{n}\sum_{j=1}^{m} s_{ij}\left\|x_i-a_j\right\|_2^2+\gamma\|S\|_2^F+\lambda Tr\left(F^T\,\widetilde{L_S}\,F\right) \\ &S\geq 0, S1=1, A\in R^{m\times d}, F\in R^{(n+m)\times k}, F^T\,F=I \end{aligned} \tag{7}$$

3.6 引入二分图进行双聚类

在构造相似矩阵S的过程中，不仅仅是为样本点分配邻近的原型点，同时也为原型点分配邻近的样本点。为了达到更好的聚类效果，本论文引入双聚类的方法。样本点和原型点的相似度矩阵表示成矩阵P。

$$P=\begin{bmatrix} & S \\ S^T & \end{bmatrix}$$

论文中提到的拉普拉斯矩阵采用的是标准的拉普拉斯矩阵（在下文我将其改成随机游走的拉普拉斯矩阵）：

$$\tilde{L}_S = I - D^{-\frac{1}{2}} \begin{bmatrix} & S \\ S^T & \end{bmatrix} D^{-\frac{1}{2}},$$

其中D是一个对角矩阵，pij是矩阵P的元素，dii是矩阵D对角线上的元素，P和D的关系如下：

$$d_{ii} = \sum_{j=1}^{n+m} p_{ij}$$

当矩阵S固定不变时，公式（7）等价于求公式（8）：

$$\begin{aligned} & maxTr\left(F^T\, D^{-1/2}\, P\, D^{-1/2}\, F\right) \\ & F \in R^{(n+m)\times k}, F^T\, F = 1 \end{aligned} \tag{8}$$

重新设置F和D，F和D重写成分块矩阵：

$$F = \begin{bmatrix} U \\ V \end{bmatrix},\ D = \begin{bmatrix} D_U & \\ & D_V \end{bmatrix}$$

带入公式（8）得到公式（9）：

$$\begin{aligned} & maxTr\left(U^T\, D_u^{-1/2}\, S\, D_v^v\, V\right) \\ & U^T\, U + V^T\, V = 1 \end{aligned} \tag{9}$$

为求解公式（9）本论文引用了2017年一篇论文上提到的定理，如下：

LEMMA 4.2. *Suppose* $A \in \mathbb{R}^{n\times m}, X \in \mathbb{R}^{n\times k}, Y \in \mathbb{R}^{m\times k}$. *The optimal solutions to the problem*

$$\max_{X^T X + Y^T Y = I} Tr(X^T A Y)$$

are $X = \frac{\sqrt{2}}{2} U_1, Y = \frac{\sqrt{2}}{2} V_1$ *,where* U_1, V_1 *are the leading* k *left and right singular vectors of* A*, respectively.*

根据上述定理4.2，求出相应的U和V，带入公式，化简得到公式(10):

$$\begin{aligned} & Tr\left(F^T\, \tilde{L_S}\, F\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} s_{ij} \left\| \frac{f_i}{\sqrt{d_i}} - \frac{f_{(n+j)}}{\sqrt{d_{(n+j)}}} \right\|_2^2 \\ & v_{ij} = \left\| \frac{f_i}{\sqrt{d_i}} - \frac{f_{(n+j)}}{\sqrt{d_{(n+j)}}} \right\|_2^2 \\ & d_x^{ij} = \|x_i - a_j\|_2^2 \\ & \tilde{d_{ij}} = d_x^{ij} + v_{ij} \end{aligned} \tag{10}$$

最终化简得到的目标函数如下：

$$min\left\|s_i+\frac{1}{2\gamma}\widetilde{d}_i\right\|_2^2 \quad (11)$$
$$s_{ij}\geq 0, s_i^T 1=1$$

用拉格朗日乘子法求解上述公式，构造拉格朗日方程L：

$$L(s_i,\eta,\beta_i)=\frac{1}{2}\left\|s_i+\frac{\widetilde{d}_i}{2\gamma_i}\right\|_2^2-\eta\left(s_i^T 1-1\right)-\beta_i^T\ s_i$$

求得γ和sij如下：

$$\gamma=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{k}{2}d_{th}^{i,k+1}-\frac{1}{2}\sum_{j=1}^{k}d_{th}^{ij}\right)$$

$$s_{ij}=\frac{d_{th}^{i,k+1}-d_{th}^{ij}}{kd_{th}^{i,k+1}-\sum_{k}^{j=1}d_{th}^{ij}}$$

3.7 算法过程

综上，本论文也是通过学习改进前人的聚类算法，并将各种算法有效地整合在一起，最终生成了本论文的算法。本论文提出的算法的主要过程如下：

1. 输入一个nxd矩阵X，表示有n个样本点，每个样本点有d维特征。输入聚类数k，原型点数量m。
2. 随机选取样本集中的m个点作为原型点aj，A = [a1,a2,...,am]
3. 计算样本点xi和原型点aj的距离，dij,利用dij计算相似度矩阵S。
4. 利用S生成拉普拉斯矩阵LS，再根据定理2计算出F矩阵
5. 利用f矩阵求

$$\widetilde{d_{ij}}=\|x_i-a_j\|_2^2+\lambda\left\|\frac{f_i}{\sqrt{d_i}}-\frac{f_{(n+i)}}{\sqrt{d_{(n+i)}}}\right\|_2^2$$

6. 再用5中求出的dij更新相似度矩阵S

利用以下公式：

$$min\left\|s_i+\frac{1}{2\gamma}\widetilde{d}_i\right\|_2^2 \quad （11）$$
$$s_{ij}\geq 0, s_i^T 1=1$$

7. 从 4 - 6 迭代进行，直到收敛.
8. 相似矩阵S收敛之后，将其带入：

$$a_j=\frac{\sum_{n}^{i=1}s_{ij}\,x_i}{\sum_{n}^{i=1}s_{ij}}$$

9. 利用更新的原型矩阵，带入公式（11）更新相似度矩阵S，直到收敛。
10. 收敛后，直接用相似度矩阵S构造二分图$\begin{bmatrix} & S \\ S^T & \end{bmatrix}$，直接调用matlab的

graphconncomp（）方法，找出该二分图的连通分量。求得的连通分量即是该样本点的聚类。

4 论文改进

4.1 改进一：mini-batch KMM-对大量样本集分批进行聚类

原论文对非凸数据集的聚类效果非常好，但是在处理大型数据集时时间较慢。为了加快聚类速度，我借鉴mini - batch K means的思想，将大数据集的每个类每次随机抽取部分样本，分批进行聚类，实验证明，分批处理效果快很多。实验中对3万个数据点分别用改进前和改进后的方法进行聚类，实验是取20次运行结果的平均值作为对比值：

原KMM算法	原KMM算法	mini-batch KMM算法	mini-batch KMM算法
时间	纯度	时间	纯度
17.38817	1	3.404008	0.977347

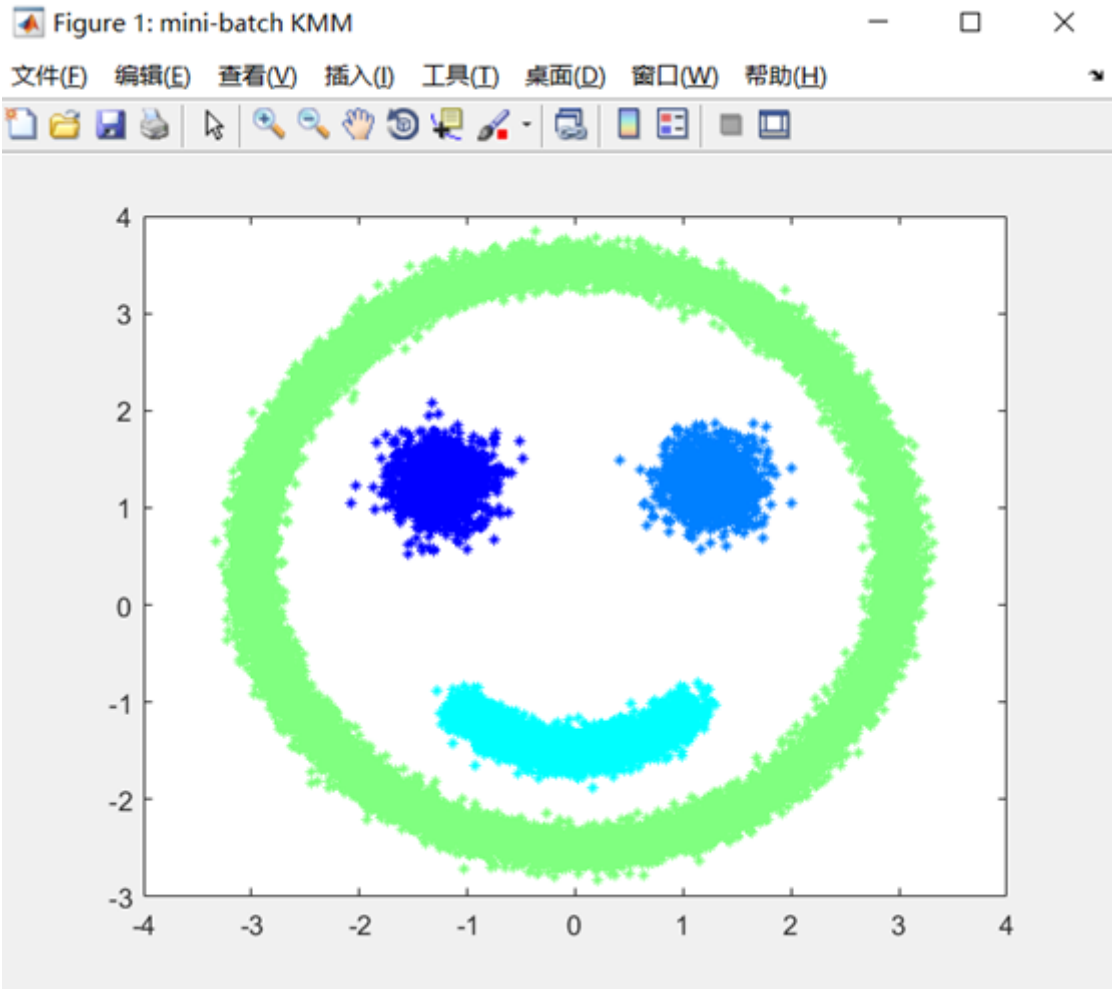
备注：这里的纯度是指用聚类算法对样本点分类的类标号和原样本点的类标号重合的比例，纯度越高聚类效果越好。

因为本论文的核心算法是找相似度矩阵的连通分量，因此我不能直接简单的分批随机抽取样本点进行聚类，这样不能保证每次抽取的样本点都能有k个连通分量。因此我采取的策略是每次从原样本集中的每个类抽取部分样本点，matlab代码如下：

```
for i = 1:30
    n1 = size(x1,1);
    n2 = size(x2,1);
    n3 = size(x3,1);
    n4 = size(x4,1);
    n11 = randperm(n1);
    n11 = n11(1:50);
    n22 = randperm(n2);
    n22 = n22(1:50);
    n33 = randperm(n3);
    n33 = n33(1:150);
    n44 = randperm(n4);
    n44 = n44(1:750);
    x11 =x1(n11,:);
    x22 =x2(n22,:);
    x33 =x3(n33,:);
    x44 =x4(n44,:);
    X = [x11;x22;x33;x44];
    x1(n11,:) = [];
    x2(n22,:) = [];
    x3(n33,:) = [];
    x4(n44,:) = [];

    [laKMM,~,~,A,~,Ah,laKMMh ]= KMM(X', c, m,k) ;
   Xey1 = X(laKMMh == 1, :);
   Xey2 = X(laKMMh == 2, :);
   Xnose = X(laKMMh == 3, :);
   Xface = X(laKMMh == 4, :);
   Xe1 = [Xe1;Xey1];
   Xe2 = [Xe2;Xey2];
   Xnos = [Xnos;Xnose];
   Xfac = [Xfac;Xface];
```

```
end
```

用分批处理的方法改进原算法之后，最终的聚类效果如下图：



根据时间和纯度的数据对比以及最终的测试效果图，可以看出改进的mini-batch KMM的算法比原算法在处理大型数据集上表现更好。

4.2 改进二：PCA KMM-对高维数据降维处理后再进行聚类

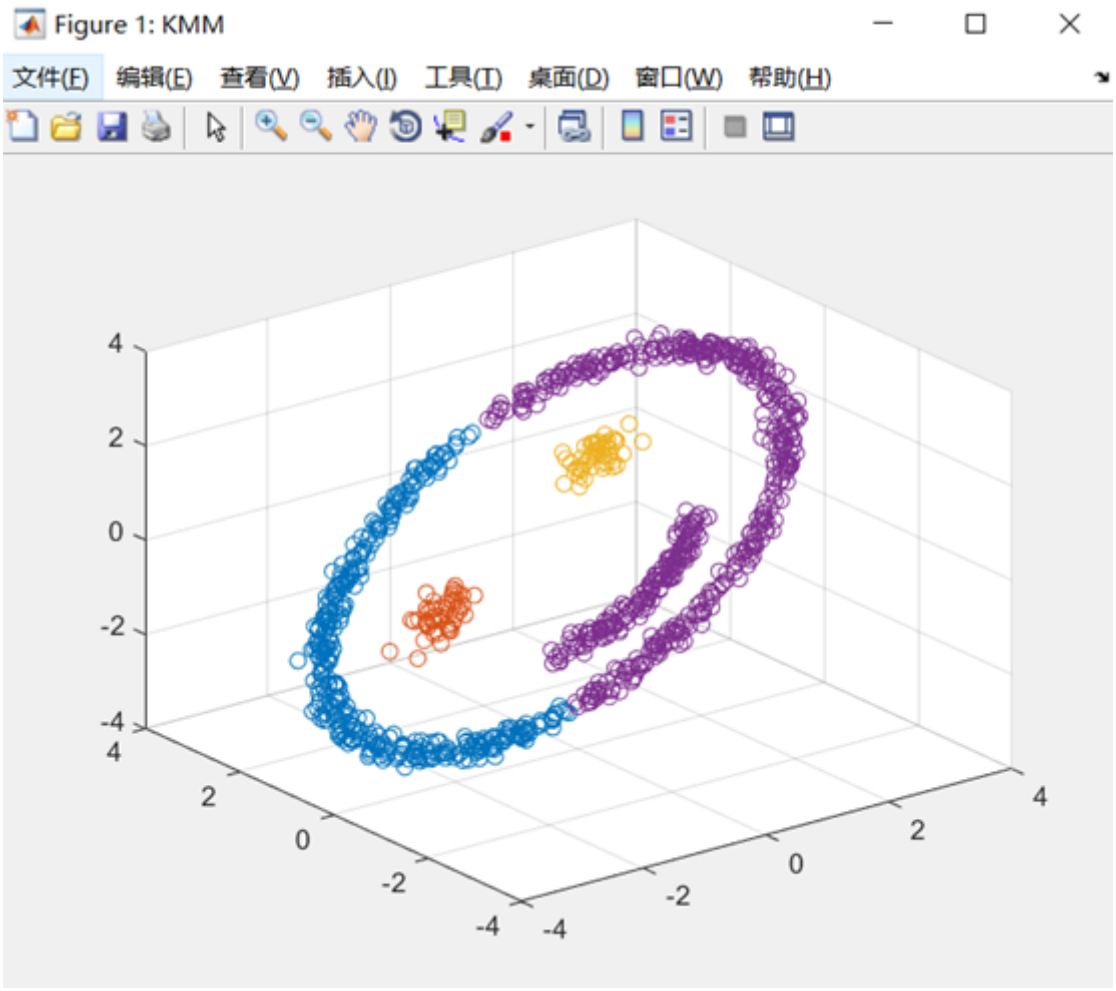
原论文的算法是对低维数据集聚类效果好，但是对于高维空间上的聚类表现略差，因此我引入了PCA主成分分析的方法，将高维数据集映射成低维，然后再聚类。实验中对1000个维度为4的数据点分别用改进前和改进后的方法进行聚类，实验对比如下：

原高维KMM算法	原高维KMM算法	PCA KMM算法	PCA KMM算法
时间	纯度	时间	纯度
0.15962	0.69775	0.130285	0.9999

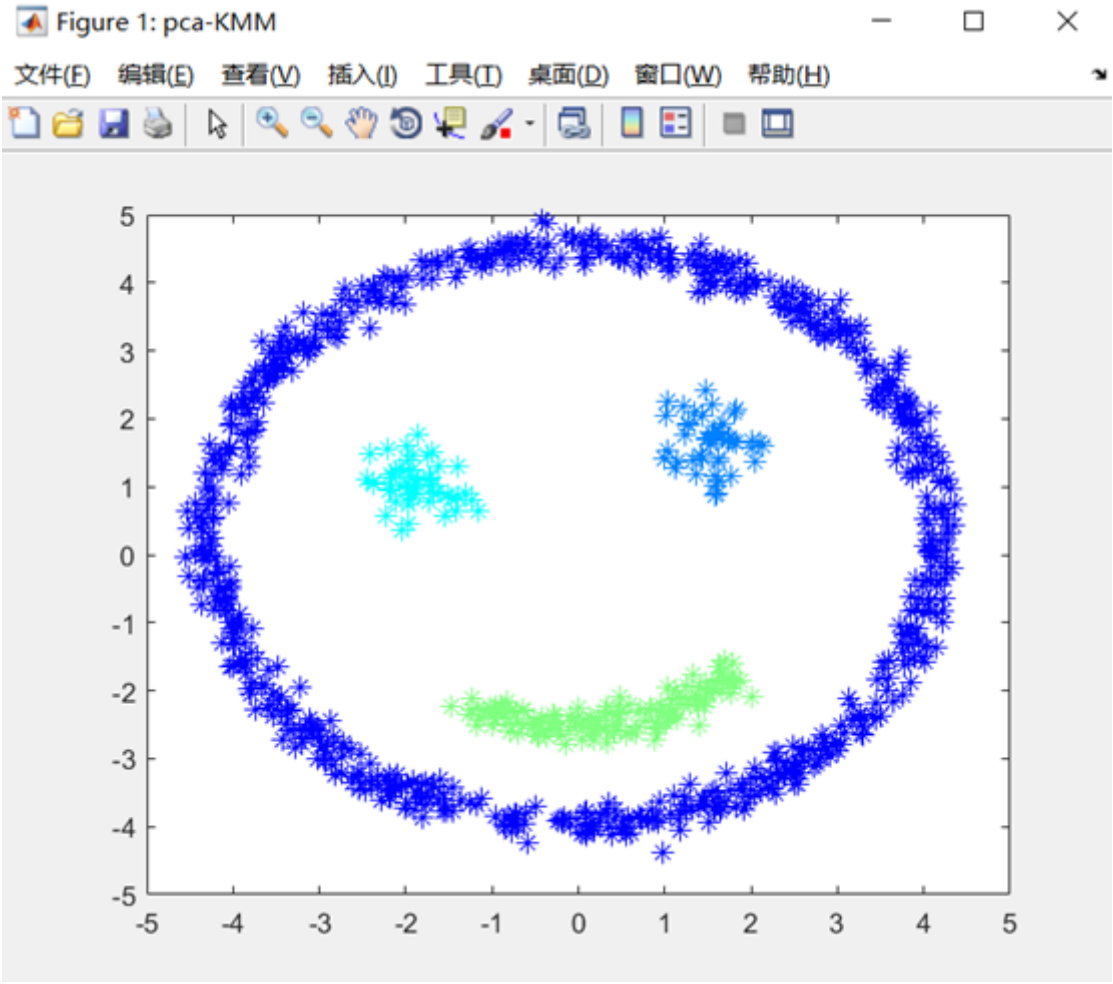
实现pca主成分分析的matlab代码如下：

```
function result = mypca(X)
%用主成分分析方法对X进行降维
%    X初始是4维的数据矩阵，先求每个维度的协方差矩阵，再求协方差矩阵的特征值
[n m] = size(X);
covX = cov(X);
[V D] = eigs(covX);
meanX = mean(X);
tempX = repmat(meanX,n,1);
score = (X-tempX)*V;
result = score;
end
```

用原论文KMM算法对具有4维特性的样本集进行聚类的效果图：



用PCA降维处理数据集之后，最终的聚类效果如下：



结合时间和纯度的数据对比以及最终的实验效果图，可以看出改进的PCA KMM算法在对高维数据集聚类中效果更好。

4.3 改进三：将标准的拉普拉斯矩阵改成随机游走的拉普拉斯矩阵

原论文利用拉普拉斯矩阵找相似度矩阵的连通分量，并巧妙的借助定理化简公式。因为论文引用的定理中表示只要是拉普拉斯矩阵就满足定理4.1.因此我改成用随机游走的拉普拉斯矩阵实现原论文算法，推理公式如下：

随机游走的拉普拉斯矩阵如下：

$$L_S = I - D^{-1} P$$

其中D是一个对角矩阵，pij是矩阵P的元素，dii是矩阵D对角线上的元素，P和D的关系如下：

$$d_{ii} = \sum_{j=1}^{n+m} p_{ij}$$

当矩阵S固定不变时，公式(7)等价于求公式（12）：

$$minTr\left(F^T \left(I - D^{-1} P\right) F\right) (12)$$

等价于

$$maxTr\left(F^T D^{-1} PF\right) (13)$$

重新设置F和D，F和D重写成分块矩阵：

$$F = \begin{bmatrix} U \\ V \end{bmatrix},\ D = \begin{bmatrix} D_U & \\ & D_V \end{bmatrix}$$

带入公式（12），等价于求公式（13）：

$$maxTr\left(U^T\ D^{-1}\ SV\right)$$
$$U^T\ U = 1, V^T\ V = 1$$

再利用论文提到的定理4.2，求出U和V，带入公式（12），推导过程如下：

$$\begin{aligned}
&\left(F^T\ L_S\ F\right) = F^T\ \left(I - D^{-1}\ P\right)F \\
&= \sum_{i=1}^{n+m} f_i^2 - \sum_{i=1}^{n+m}\sum_{j=1}^{n+m} f_i\ f_j\ d^{-1}\ p_{ij} \\
&= \sum_{i=1}^{n+m} \frac{f_i^2}{d_i} d_i - \sum_{i=1}^{n+m}\sum_{j=1}^{n+m} f_i\ f_j\ d^{-1}\ p_{ij} \\
&= \sum_{i=1}^{n+m} \frac{f_i^2}{d_i} \sum_{j=1}^{n+m} p_{ij} - \sum_{i=1}^{n+m}\sum_{j=1}^{n+m} f_i\ f_j\ d_i^{-1/2}\ d_j^{-1/2}\ p_{ij} \\
&= \frac{1}{2}\sum_{i=1}^{n+m}\sum_{j=1}^{n+m} p_{ij}\left(\frac{f_i^2}{d_i} + \frac{f_j^2}{d_j} - \frac{f_i}{\sqrt{d_i}}\frac{f_j}{\sqrt{d_j}}\right) \\
&= \frac{1}{2}\sum_{i=1}^{n+m}\sum_{j=1}^{n+m} p_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2
\end{aligned}$$

最终推导出来的公式和论文中的公式（10）是一样的：

$$Tr\left(F^T\ L_S\ F\right) = \frac{1}{2}\sum_{i=1}^{n+m}\sum_{j=1}^{n+m} p_{ij}\left\|\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right\|^2$$

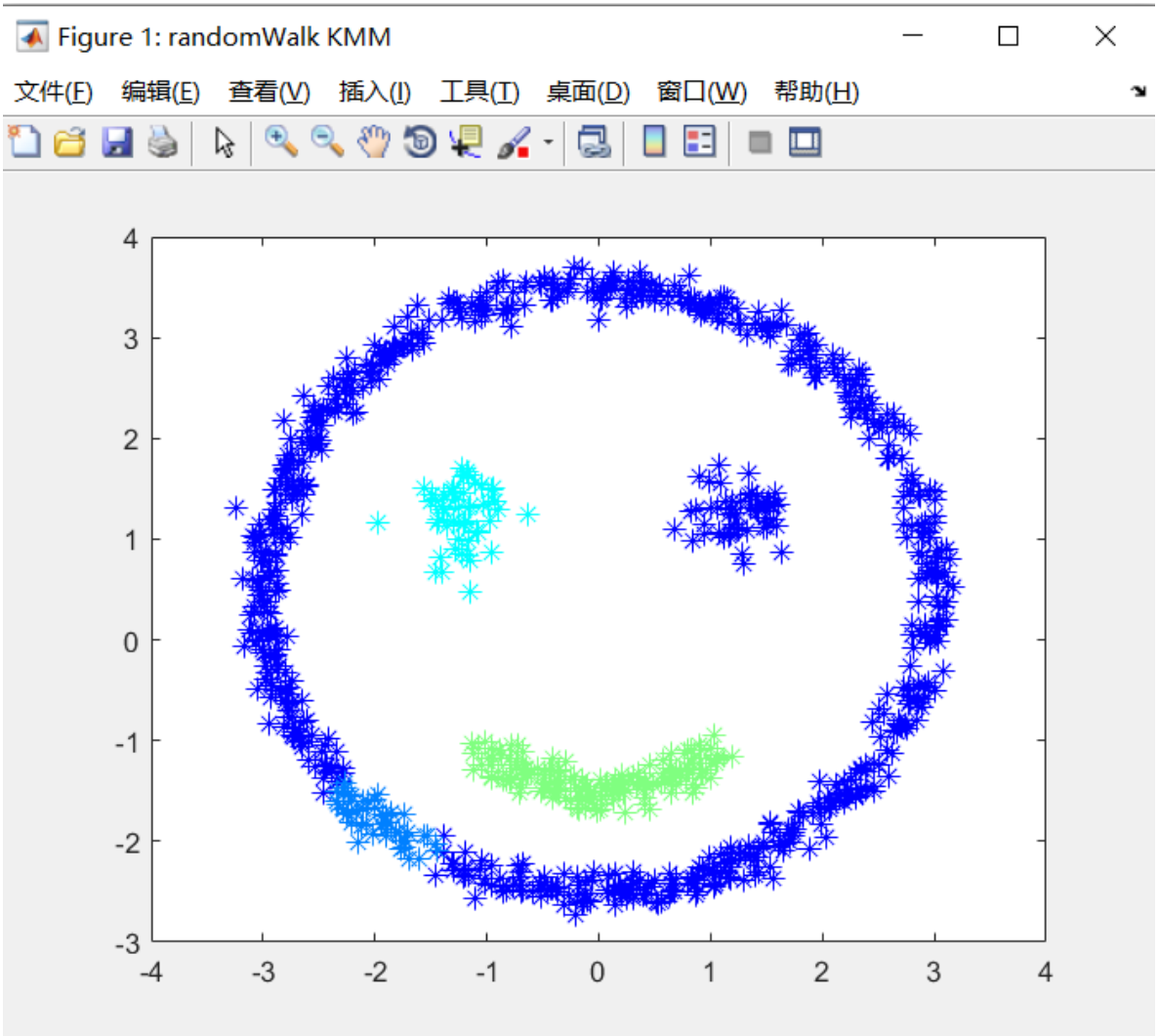
将我推导的随机游走公式换入源代码中，部分代码如下：

```
function [Z, U, V, evc, D1z, D2z] = svd2uv(Z, c)
    [n,m] = size(Z);
    ver=version;
    z1 = sum(Z,2);
    D1z = spdiags(1./sqrt(z1),0,n,n);
    D1 = spdiags(1./(z1),0,n,n);
    D = spdiags(z1,0,n,n);
    z2 = sum(Z,1);
    D2z = spdiags(1./sqrt(z2'),0,m,m);
    Z1 = D1*Z;
    LZ = full(Z1'*Z1);

    [V, evc, ~]=eig1(LZ,c+1);
    V = V(:,1:c);
    U = (Z1*V)./(ones(n,1)*sqrt(evc(1:c))');

    U = sqrt(2)/2*U; V = sqrt(2)/2*V;

end
```

实验样本集是1000个2维的数据点，实验对比如下：

原KMM算法	原KMM算法	随机游走 KMM算法	随机游走 KMM算法
时间	纯度	时间	纯度
0.060499	1	0.174537	0.9225

改成随机游走的拉普拉斯矩阵以后，聚类效果变差了，时间比原算法慢了近3倍，准确度也降低了一点。效果图如下：



改进后的randomWALK KMM算法在对数据集聚类中效果并不好，不仅时间慢了近3倍，聚类纯度也明显下降。根据论文的定理4.2，应该是所有的拉普拉斯矩阵都有“拉普拉斯矩阵中值为0的特征值的个数等于对应的二分图的连通分量”这一性质。效果不好的原因也许是我的公式推导有误，也许是随机游走的拉普拉斯矩阵效果就是不如标准的拉普拉斯矩阵好。对于这一问题我至今也没弄清楚。

5 报告总结

本报告是通过学习别人论文上的算法，并基于对原论文算法的理解做出轻微的改动。由于我能找得到原论文算法实现的matlab代码，论文和源码结合起来一起看更方便了我理解论文提出的算法。

本报告提交的代码也是在源码的基础上轻微改动而已，一开始是打算用python重写本论文的matlab源码的，但是改到一半发现，在进行数据分析时需要大量的矩阵运算，此时用matlab写代码效率更高一些，与此同时我也可以节省写代码的时间，利用这些时间更深入的理解论文算法，从而想出更有效的改进方案。但不得不说这篇顶会论文提出的算法真的很巧妙，算法简单但是很好地融合了多种算法，环环相扣，且毫不冗余。我只能在一些细枝末节上进行改动，一旦试图改动本论文的核心算法，效果就明显变差。就比如我试图将标准的拉普拉斯矩阵改成随机游走的拉普拉斯矩阵，那不管是聚类的时间还是聚类的纯度都明显变差。

总之通过这次课程项目，我阅读大量的论文和相关文献，学习到了很多经典的聚类算法以及改进后的聚类算法。