参赛密码 _____

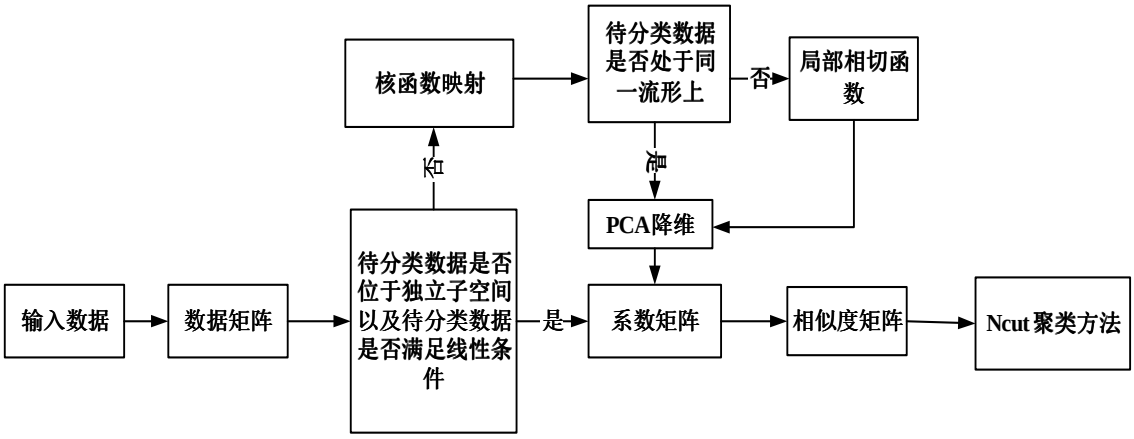
（由组委会填写）

第十二届“中关村青联杯”全国研究生数学建模竞赛

题 目 数据的多流形结构分析

摘 要：

本文以稀疏子空间聚类以及低秩子空间聚类等基本谱聚类算法为基础，通过运用核映射算法，融合与数据本身结构相关的局部切线空间函数以及主成分分析算法建立了可以应对独立子空间聚类、非独立子空间聚类、非线性聚类、混合多流体聚类问题以及多种含有大数据量的实际问题，包括处理运动分割、人脸识别、工件识别等情况中的多种类型数据分类的聚类算法，并且引入 Map-Reduce 并行处理方法优化了算法的计算效率，算法整体处理步骤如下图所示。



整个处理方法具有“通用性”，能够解决相关类的绝大部分问题。文章中问题二、问题三（a）以及问题四等可视实验结果验证了算法的有效性。

本文的算法基础在问题一的模型求解中给出。基于参考文献中的基础谱算法，本文提出了基于 SSC 与 LRR 相结合的正则项 $R(Z)=\|Z\|_1+\lambda\|Z\|_*$，同时考虑了数据

可能存在误差的情况，提出了新的保真项$F(E)=\beta\|E\|_{2,1}$，并且通过增广拉格朗日-交替极小化求解方法得到模型的表示系数，还通过附加相似度权值计算了改进方法的相似度矩阵，然后利用 Ncut 算法得到了聚类结果，最后计算得出了整个处理算法的计算复杂度$NM(t_1+dt_2)+k^2t_3)+O((N+k)N^2)$。

问题二中出现的非线性数据分类，非独立子空间分类以及混合流体分类等情况。本文首先采用了核映射算法，即通过一个非线性变换式将输入模式空间R中的数据映射到高维特征空间F中，解决了无法在低维空间处理非线性和非独立子空间的问题，这里采用的是高斯核函数$k(x,y)=\exp\left(-\|x-y\|^2/2p^2\right)$。其次，考虑到聚类的主旨在于将数据集分类，即确保类内结构的相似性尽可能大，类间相似性尽可能小。我们将结合数据的部分原始信息和可靠的几何信息来构造局部切线空间函数，融合原有的相似度函数，改进相似度矩阵得到新的融合函数$S'_{ij}=f(p_{ij},q_{ij})$，从而解决当数据来自混合流体时的一些列问题，进而得到正确的聚类。实验结果表明，该聚类算法能够很好地解决上述问题。

问题三中主要涉及到算法在实际场景中的运用，针对问题（a）将采用原有算法，首先将其核映射到高维空间，其次根据前文所述算法计算其相似度矩阵，从而获得聚类结果。针对问题三中的人脸识别以及运动分割等问题，本文提出了利用主成分分析方法进行降维处理，在保留原始图像数据所需要的大部分有用信息的前提下，用一个低维的子空间图像数据来描述人脸以及运动图像，减少了算法的运算量。

针对问题四中的大数据，前文的算法已经能够解决该类问题。但是考虑到数据处理的实时性和算法的效率，我们结合 Map-Reduce 算法，提出了用于本文的并行处理方法，大大缩减了运行时间。最后通过对问题四场景中的数据聚类进行实验验证，结论证明本文所提出的算法能够很好地解决该类问题。

关键词：谱聚类，稀疏，核映射，局部切线

1. 问题的重述

几何结构分析是进行数据处理的重要基础,特别是在对于高维数据的相关性分析和聚类分析等基本问题上结构分析格外重要。

为了挖掘数据集的低维线性子空间结构，我们常用数据降维方法处理数据，这类方法以假设数据集采样于一个线性的欧氏空间为前提。但是，往往在实际问题中很多数据具备更加复杂的结构。

针对单一子空间结构假设的后续讨论主要分为两个方面，首先是从线性到非线性的扩展，主要的代表性工作包括流形（局部具有欧氏空间性质的空间定义为流形，而欧氏空间就是流形最简单的实例）学习等。

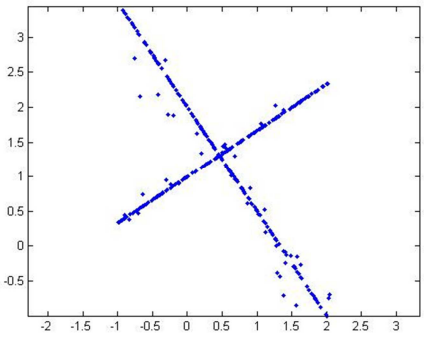
其次是流形或子空间从一个扩展到多个的问题，即考虑处理的数据集采样于多个欧氏空间的混合。子空间聚类（又称为子空间分割，假设数据分布于若干个低维子空间的并集）是将数据按某种分类准则划分到其所属的子空间的过程。通过子空间聚类，可以将来自同一子空间中的数据归为一类，再由同类数据可以提取相应子空间的相关性质。子空间聚类的求解方法包括代数方法、迭代方法、统计学方法以及基于谱聚类的方法。在众多算法中，基于谱聚类的方法在近几年较为流行，通常情况下使用这类方法一般都能得到正确的分类结果，其中代表性的谱聚类子空间分割方法包括低秩表示和稀疏表示等。

假设数据的结构为混合多流形，因为多数境况下数据来自混合子空间。虽然也有些实际问题的数据并不符合混合子空间结构的假设，但这种境况处理相对简单。此外，混合流形不全是子空间的情况，数据往往具有更复杂的结构，分析这种数据具有更大的挑战性。

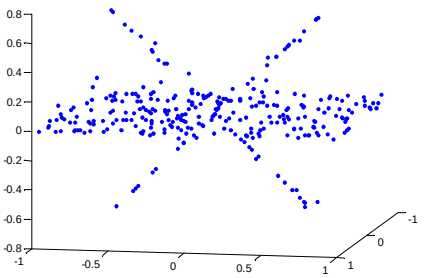
本文在几何结构分析问题中假设数据分布在多个维数不等的流形上，其特殊情况是数据分布在多个线性子空间上。下面对问题进行简要重述：

1.附件一中 1.mat 中有一组高维数据（.mat 所存矩阵的每列为一个数据点，以下各题均如此），数据结构未知，需要使用合适的方法将该组数据分成两类。

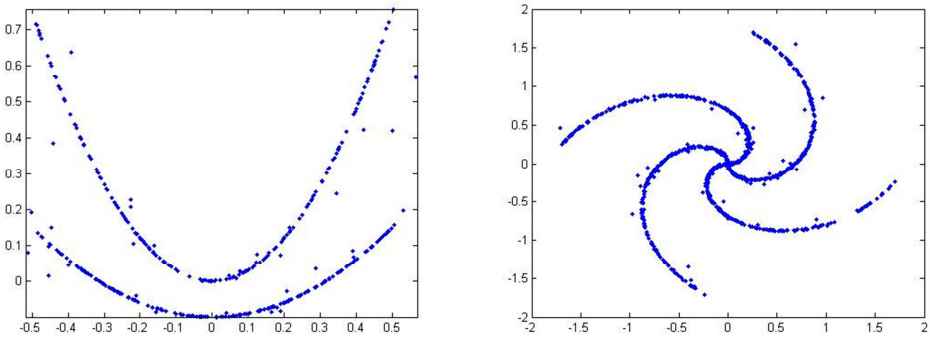
2.图 1(a)为两条交点不在原点且互相垂直的两条直线，将其分为两类；图 1(b)为一个平面和两条直线，需要按要求将其分为三类。图 1(c)为两条不相交的二次曲线，按要求将其分为两类。图 1(d)为两条相交的螺旋线，结构相对复杂需按要求将其分为两类。



(a)



(b)



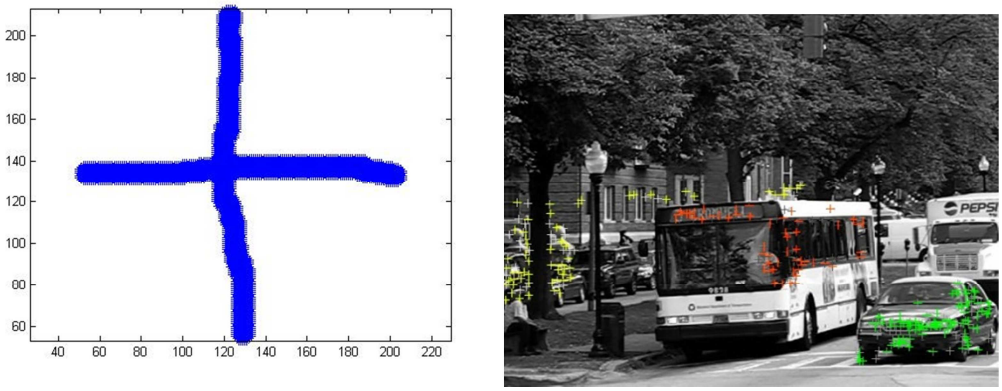
(c) (d)

图 1

3. 解决以下三个实际应用中的子空间聚类问题：

（a）如图 2（a）所示，使用适当的方法将图 2（a）中十字上的点分成两类。

（b）图 2（b）显示了物体运动视频中的一帧，有三个不同运动的特征点轨迹被提取出来保存在了 3b.mat 文件中，使用适当方法将这些特征点轨迹分成三类。



(a) (b)

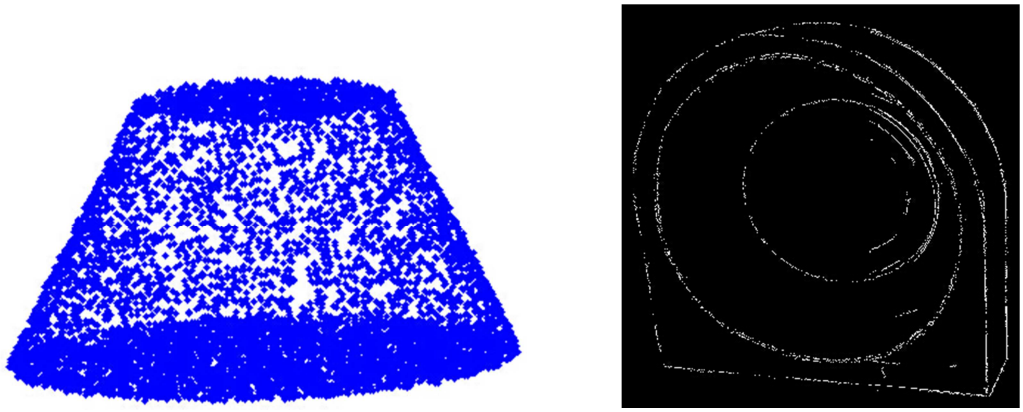
图 2

（c）3c.mat 中的数据为两个人在不同光照下的人脸图像共 20 幅（X 变量的每一列为拉成向量的一幅人脸图像），需要将这 20 幅图像分成两类。

4.解答如下两个实际应用中的多流形聚类问题

图 3(a)分别显示了圆台的点云，将点按照其所在的面分开(即圆台按照圆台的顶、底、侧面分成三类)。

图 3(b)是机器工件外部边缘轮廓的图像，将轮廓线中不同的直线和圆弧分类，类数自定。



(a) (b)

图 3

2. 模型的假设

1、假设本文中题目数据的噪声分布都处于分析中的理想状态；
2、人脸数据、运动序列都来自于单独的子空间；
3、在计算算法运行时间的时候假设计算机每次计算的时间均相同；

3. 符号说明

符号	符号说明
d_i	数据点
R^M	M 维空间
D	干净数据
$R^{M\times N}$	$M\times N$ 维空间
S	相似度矩阵
Z_{ij}	表示系数
$\|\ \|_1$	ℓ^1 范数
$\|\ \|_F$	ℓ^2 范数
$\|\ \|_{p,q}$	混合范数，$\|\ \|_{p,q}=\left(\Sigma_j\left(\Sigma_i\left\|X_{ij}\right\|^p\right)^{1/p}\right)^{1/q}$

$\|\ \|_*$	核范数
$\sigma(Z)$	矩阵 Z 的奇异值构成的向量
C_Z	系数矩阵 Z 的约束集合
λ	正则化参数
$F(E)$	数据项或者保真项
$R(Z)$	正则项或者惩罚项
Y,W,P,Q	拉格朗日乘子矩阵
μ	罚参数
Z^*	最优近似解
G	无向图
V	无向图顶点
E	无向图顶点之间的边
$D_{l\times l}$	度矩阵
$R \to F$	R 到 F 的映射
$K(x,y)$	核函数
$\varphi(x)$	变换函数
p_{ij}	相似结构体
Θ_i	数切线空间
o	可调参数
$Knn(x)$	与 x 相似的聚类
$N(d_i)$	数据 d_i 在该子空间的最近邻域
$\sum_x$	局部样本协方差矩阵
θ_m	模型参数

μ_m	数据的鲁棒均值
y	潜变量
ε_m	噪声
α_i	加权系数
ϕ_i	基向量
s_I	抽样因子
$rank()$	矩阵的秩
SSC	稀疏子空间聚类
LRR	低秩子空间聚类
SVD	奇异值分解
PCA	主成分分析
Ncut	规范化分割法

4. 问题的分析

根据题目要求本文主要讨论子空间聚类相关问题。在过去的二十年中，解决子空间聚类问题的方法层出不穷，根据算法的原理可以将这些方法分为四大类，即统计算法，迭代算法，代数算法以及基于谱聚类框架的方法。在本文当中，我们在谱聚类算法的基本框架上进行了进一步研究，包括稀疏子空间聚类（Sparse Subspace Clustering，SSC），低秩表示（Low Rank Representation，LRR）及在此基础之上建立的推广模型。

问题一中要求对来自于两个独立子空间的高维数据进行聚类。本文主要采用谱聚类方法，该类算法基于相似矩阵进行聚类。由于谱聚类算法的聚类分割效果的好坏几乎完全依赖于相似矩阵的质量，因此，相似矩阵的好坏直接影响聚类效果。所以，这里最主要的过程就是建立给定高维数据的空间表示模型，寻求其在低维空间的表示系数，然后根据表示系数矩阵构造出相似度矩阵，最后对相似度矩阵运用谱聚类方法，如规范化割（Normalized cut，Ncut）获得数据的聚类结果。稀疏子空间聚类的基本框架如图 4 所示。这也是本文解决题目问题的基础。

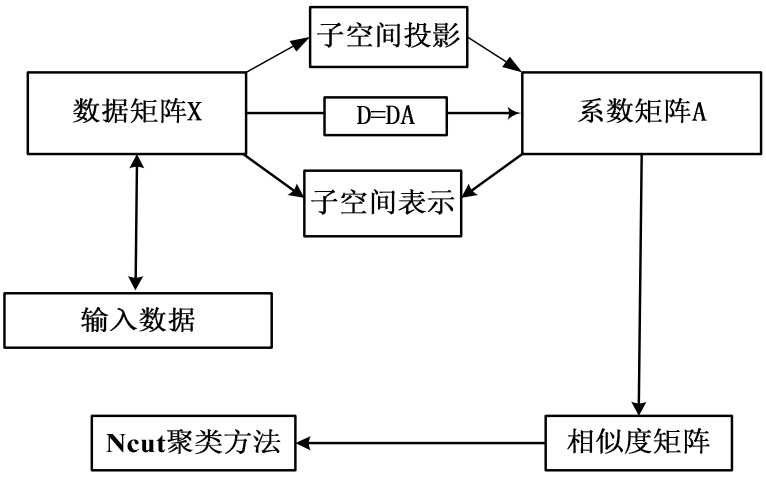


图 4　稀疏子空间聚类的基本框架

对于问题二中的问题，其中，图 1（a）中的数据来自于二维空间中两条相互垂直的直线，显然，这些数据虽然是来自两个仿射空间，但是两者相互垂直，因此可以按照问题一中的求解方式进行解答。然而图（b）中为一个平面和两条直线，这是一个不满足独立子空间关系的情况。图（c）中为两条曲线，在二维空间中它们不是非线性的结构。图（d）中两条螺纹曲线之间拥有显著交叉点，需要进行分类的数据来源于不同的两个流形结构。因此如果仍然采用原有的谱聚类方法，对于后面三个问题的聚类结果将无法满足题目要求。在前文中，基于本文所提出的改进聚类算法，能够得到较好的子空间分割概率。但是要基于一个基本的前提，即所分类的数据来自于独立子空间这个条件。然而，在本题目问题（b）、问题（c）和问题（d）中都不是完全满足该条件的，因此为了成功对以上情况进行聚类，本文将对上述算法进行改进。

本文主要采用两种方法解决上述问题。第一种方法是针对在目前的空间内不存在非独立可分的线性子空间的情况。此时，可行的解决方案是向高维空间转化，使其变得线性可分，具体方法是通过核映射的思想改进模型的构建，从而使不处于独立子空间的数据在特征空间内变得线性可分。

第二种方法是针对问题中分类的数据来自于混合流形的情况，此时不同流形体类间存在明显交叉，那么相似度矩阵就会因为极差的两两相似度而发生错误分类的情况。传统的谱聚类方法在属于不同类的相似值相对较低时才有效。因此传统方法对于这种交叉情况并不适合。对于此问题，我们的基础想法是结合采样数据的部分原始信息和可靠的几何信息来构造合适的相似度矩阵，进而得到正确的聚类。具体来讲，虽然整体上看图（d）数据位于或者靠近于多个平滑非线性流形，但局部上每个数据点和它的邻近点都位于该流形的一条线性路径上。此外，在每个点处的切线空间为非线性流形的局部集合结构提供了很好的低维线性近似。显而易见的是，在不同流形的交叉区域内，同一个流形上的数据点拥有相似的局部切线空间，而不同流形上的点数据的局部切线空间是不相似的。因此，此类局部几何信息可以用于帮助构造相似矩阵。对于相对较远的点数据，我们很难只用局部几何信息判断它们是否属于同一个流形，因此我们关注局部区域。直观上讲，对于位于相同局部区域的两个点，如果两点彼此接近切两点拥有相似的局部切线空间，那么它们就很有可能位于同一个流形。

针对题三中的问题，其中图 2（a）要求将一群二维空间的坐标点分为横竖两类，图 2（b）要求将三个不同运动的特征点轨迹分成三类，图 2（c）中要求将在不同光照下的两个人的 20 幅图像分成两类。按照规定聚类的空间，显然，图 2（a）中的数据在二维空间中不满足相互独立的子空间这一要求，因此我们需要通过核函数将其映射到高维空间，然后按照改进的聚类算法完成坐标点的分类。对于问题（b）所给出的运动序列数据，已经有文献证明每个运动序列都是一个单独的子空间，从而我们可以按照现有的算法进行聚类。同样，对于问题（c）也有文献证明，人脸的图像信息在一个较高维的空间中不是随机或散乱分布的，而是有一定规律性的。这个规律就是：同一个人的不同图像在这个高维的空间中彼此之间的距离比较近，而对于完全不相干的图像，在空间中数据的间隔就比较远。因此，也可以按照现有方法进行计算，但是运动序列数据以及人脸图像数据一般都比较多，相应地需要处理的像素点也就很多，导致直接计算量非常大，占用时间长。因此，本文先对这些数据进行变换操作处理，从而用一个低维的子空间图像数据来描述人脸的图像，在提高信号处理效率的同时又能在保留原始图像数据所需要的大部分信息，且能完成有效聚类。

问题四中给出的数据，其具体表现为混合流体结构的子空间聚类问题。前文中所给的方法已经能够解决类似问题。但相对于其他问题而言，问题四中数据规模很大，前文所给出的聚类算法在处理大规模数据时运行时间非常长，所以无论是从系统资源还是从实时性效率的角度看，都不是最合适的解决方案。为解决上述问题，本文提出一种先抽样再并行得到聚类结果的计算方法。

5. 模型建立以及求解

5.1 问题一的模型建立以及求解

根据题目中的要求，考虑对多个子空间数据聚类的建模问题。假定现在获取的数据当中有 N 个 M 维干净的数据组 $d_i \in R^M$ ， $i=1,\cdots,N(N>M)$ ，其中 $i=1,\cdots,N(N>M)$ 。由这些数据构造一个矩阵 $D=[d_1,d_2,\cdots,d_N]\in R^{M\times N}$ 。假设这些数据来自于 $n(n\geq 1)$ 个相互独立的线性或仿射子空间的并 $S=\bigcup_{i=1}^{n} S_i$ 。那么聚类问题就等效为寻求数据指标集 $\{1,2,\cdots,N\}$ 的一个划分 $C=\{C_1,C_2,\cdots,C_n\}$ ，使得 C_i 对应于子空间 $S_i\ (i=1,\cdots,n)$ 中的数据下标，且 $\sum_{i=1}^{n} C_i = N$ 。

5.1.1 模型的建立

对于前文中所说的每一个数据 $d_i \in S_i$ ，都可以由其所在子空间 S_i 当中的其他向量，通过线性组合来表示：

$$d_i = \sum_{j\neq i} Z_{ij} d_j \qquad (1)$$

此时对其表示系数 Z_{ij} 施加一定的约束，那么对于所有数据及其表示系数都可以按照一定方式的矩阵形式表示。因此，如果我们能够获取其表示系数，就可以得到数据分类进而为数据的正确聚类提供支持。以此为基础，稀疏子空间聚类和

低秩子空间聚类算法是对表示系数 Z_{ij} 进行约束再进一步求解的。下面对这两种基础算法进行介绍。

➢ 稀疏子空间聚类（SSC）

SSC 方法的基本思想是通过对表示系数进行稀疏约束，从而使数据尽可能被同一子空间的其它数据线性表示。此外，在求解数据 $d_i \in R^M$ 的稀疏表示时，为了避免平凡解，字典中需要排除 $d_i \in R^M$ ，等价于强制 $d_i \in R^M$ 的表示系数 $Z=(Z_1,Z_2,...Z_i,....)$ 的第 i 个分量为 0，那么模型可以表示为：

$$\begin{cases} \min \|Z\|_0 \\ s.t.\ \ D=DZ, diag(Z)=0 \end{cases} \tag{2}$$

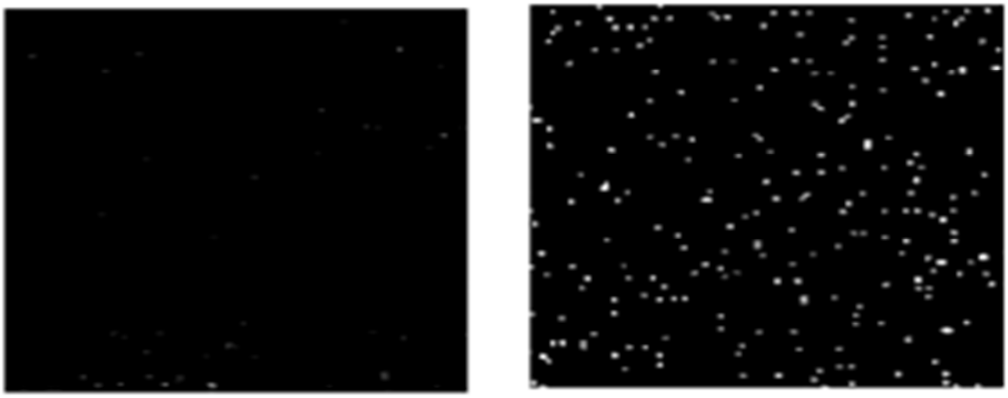
其中 $\|Z\|_0$ 是矩阵 Z 的 ℓ^0 范数，代表矩阵 Z 中非零元素的个数，即寻找到一个最稀疏的解 Z 。然而上述优化问题是一个 NP 难的问题，通常不易求解，但若其存在较稀疏的解，那么上述式子（2）等价于如下优化问题：

$$\begin{cases} \min \|Z\|_1 \\ s.t.\ \ D=DZ, diag(Z)=0 \end{cases} \tag{3}$$

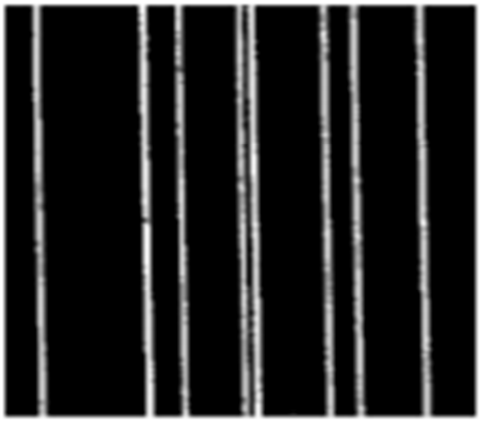
其中 $\|Z\|_1$ 是矩阵 Z 的 ℓ^1 范数，$\|Z\|_1=\sum_{i=1}^{n}\sum_{j=1}^{n}|Z_{ij}|$ 。上述模型中考虑的是理想情况，现对整个方程做出调整，假设存在着误差，那么对于考虑噪声的情况，SSC 将上述模型推广为：

$$\begin{cases} \min\limits_{Z,E} \|Z\|_1+\lambda\|E\|_F \\ s.t.\ \ D=DZ+E, diag(Z)=0 \end{cases} \tag{4}$$

其中 λ 是参数，用来平衡两项的权重。原来的 SSC 使用 ℓ^2 范数（矩阵形式下是 Frobenius 范数）$\|E\|_F$ 来刻画噪声，这适用于大部分数据都受到较小程度污染的稠密噪声。对于少部分数据受到较大程度的污染的随机污染噪声，可以使用 ℓ^1 范数 $\|E\|_1$ 进行刻画。还有一种是 $\|E\|_{2,1}$ ，可以刻画数据存在的外点，即有少部分数据根本就不来自采样子空间的情况。下图给出了这三种噪声的示例：



（a）轻微的扰动　　（b）随机的噪声



（c）采样的子空间外的噪声

图 5　噪声示意图

➢ 低秩子空间聚类（LRR）

与稀疏表示不同，LRR 表示利用同类数据的相关性，其基本思路是将广鲁棒的主成分分析推广到多个子空间。具体地，LRR 尝试从表示系数矩阵的全局出发，利用表示系数矩阵的低秩约束，从而尽可能地将数据用同一子空间的数据线性表示，其具体形式表示为：

$$\begin{cases}\min rank(Z)\\ s.t.\ \ D=DZ,\end{cases} \tag{5}$$

其中，$rank(Z)$表示矩阵Z的秩。类似于ℓ^0最优化问题，秩优化问题同样也是一个 NP 难的问题，那么我们也可以采用和 SSC 一样的技巧，使用矩阵的核范数（Nuclear Norm）来代替矩阵的秩进行优化：

$$\begin{cases}\min \|Z\|_*\\ s.t.\ \ D=DZ,\end{cases} \tag{6}$$

其中$\|Z\|_*$是矩阵Z的核范数，定义为矩阵Z的奇异值的和，那么上述模型具体可变为：

$$\|Z\|_* = \sum_{i=1}^{n}\sigma_i(Z) = \|\sigma(Z)\|_1 \tag{7}$$

其中，$\sigma(Z)\in R^n$表示矩阵Z的奇异值构成的向量。直观地将秩函数用矩阵的核范数代替，在本质上其实是用$\|\sigma(Z)\|_1$代替$\|\sigma(Z)\|_0$，因为$\|\sigma(Z)\|_0 = rank(Z)$。同样模型中考虑的是理想情况，现对整个方程做出调整，假设存在着误差，进一步考虑噪声对数据的影响，LRR 将上述式子（7）中的问题推广为：

$$\begin{cases}\min \|Z\|_* + \lambda\|E\|_{2,1}\\ s.t.\ \ D=DZ+E,\end{cases} \tag{8}$$

其中参数λ用于平衡两者的权重。LRR 使用$\ell^{2,1}$范数$\|E\|_{2,1}$来刻画噪声，可以

检测到数据当中的外点或缺失。

➢ 改进方法

影响 SSC 以及 LRR 的同类的子空间聚类办法的性能主要有两个方面。一方面，模型对各种噪声、数据缺损、奇异数据的鲁棒性取决于数据项；另一方面,子空间表示系数矩阵的结构取决于正则项。综合以上分析，该类模型均可以统一描述为如下的优化问题：

$$\begin{cases}\min J(Z)=F(E)+\lambda R(Z)\\ s.t.\quad Z\in C_Z\end{cases} \tag{9}$$

式中，λ 为正则化参数，C_Z 表示系数矩阵 Z 的约束集合，$F(E)$ 称为数据项或者保真项，反映了数据表示 DZ 与数据 X 之间的逼近程度，根据数据噪声的不同分布，$F(E)$ 采用不同的矩阵范数来度量误差；$R(Z)$ 称为正则项或者惩罚项。上述聚类方法为了使 Z 具有理想的结构，处理表示系数矩阵 Z 采用了不同的正则项。然而，SSC 算法与 LRR 算法都存在着明显的缺点。

虽然 SSC 保证不同类间样本没有连接，获得的表示系数相似度矩阵较为“理想”。但是，事实上这并不能保证能够获得准确的子空间分割结果。其原因是稀疏表示的解可能“太稀疏”，同类样本聚集性不够，从而使重构图由许多不连通子图构成，同类样本聚集性大大降低，这个问题在同类样本相关性较强时尤为明显。下面的例子可以直观地看出“太稀疏”构图的问题。构造两个子空间 S_1 和 S_2，令 $X=[X_1,X_2]$，$X_1\in S_1, X_2\in S_2$，其中：

$$X_1=\begin{bmatrix}1&1&1&1\\2&2&2&2\end{bmatrix}, X_2=\begin{bmatrix}2&2&2&2\\3&3&3&3\end{bmatrix} \tag{10}$$

且假设 $\dim(S_1)=\dim(S_2)=1$，因此易知 S_1 和 S_2 是独立的。那么问题存在如下的最优解：

$$Z^*=\begin{bmatrix}Z_1^* & 0\\ 0 & Z_2^*\end{bmatrix} \tag{11}$$

其中：

$$Z_1^*=Z_2^*=\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \tag{12}$$

S_1 和 S_2 是独立的。解矩阵 Z^* 是块对角矩阵，保证了不同类间样本不相似。但与此同时，类内连接 Z_1^* 和 Z_2^* 也非常稀疏，导致 8 个样本构得的图是实际上是 4 个不连通的子图。而理想地，两个子空间的分割问题应由 2 个不连通子图构成。因此，若以此 Z^* 作为相似度矩阵来完成谱聚类，聚类的结果很可能是错误的。这主要是因为稀疏表示只有子集选择的能力，而缺乏聚集的能力。此外，稀疏表示的解是不稳定的，比如当两个样本的相关性很强时，稀疏表示只会随机选择其中

一个，而并不在乎选择了哪一个，因此其缺乏聚集相关性强的样本的能力。实际当数据采样较充分时，其往往具有很强的相关性，此时稀疏表示聚类的结果就会大打折扣。

同样，要单靠利用低秩性质低秩表示完成子空间聚类，是远远不够的。同样举例说明，同样构造两个子空间 S_1 和 S_2，令 $X=[X_1,X_2]$，$X_1\in S_1, X_2\in S_2$，其中：

$$X_1=\begin{bmatrix}1 & 2\\ 0 & 0\end{bmatrix}, X_2=\begin{bmatrix}0 & 0\\ 1 & 2\end{bmatrix} \tag{13}$$

且假设 $\dim(S_1)=\dim(S_2)=1$，因此易知 S_1 和 S_2 是正交的。那么问题存在如下的最优解：

$$Z^*=\begin{bmatrix}0.5 & 1 & 1 & 2\\ 0.25 & 0.5 & -0.5 & -1\\ 1 & 2 & 0.5 & 1\\ -0.5 & -1 & 0.25 & 0.5\end{bmatrix} \tag{14}$$

由上可知，虽然利用核范数代替秩函数最优化后取得的解仍是最低秩的，但是这并不意味着 LRR 的优越性仅仅来自于低秩，也很有可能与核范数最优化带来的解的其余性质(如聚集性)有关。

总而言之，单独依靠两种算法的其中一种都不能很好地得出聚类结构。本文依据上述原理，采用融合 SSC 和 LRR 两种方法，进行聚类。具体准则函数：

$$\begin{cases}\min\limits_{Z,E}\|Z\|_1+\lambda\|Z\|_*+\beta\|E\|_{2,1}\\ \quad s.t.\quad D=DZ+E\end{cases} \tag{15}$$

$\|\bullet\|_1$ 为 l_1 范数，即矩阵所有元素的绝对值之和；$\|\bullet\|_*$ 为核范数，即矩阵所有奇异值之和；$\|\bullet\|_{2,1}$ 为矩阵每一列的 l_2 范数之和．第 1 项为 SSC 约束，能够获取每个数据点的局部结构；第 2 项为 LRR 低秩约束，能够获取原始数据的全局结构；第 3 项用于约束野点和噪声．参数 λ 和 β 的作用是平衡各项。

上式以原始数据 D 作为为字典，对处理效果会产生一定的影响．一般都会存在噪声的影响，因此修改上式，获得最终模型：

$$\begin{cases}\min\limits_{Z,E}\|Z\|_1+\lambda\|Z\|_*+\beta\|E\|_{2,1},\\ \quad s.t.\quad D=(D-E)Z+E,\end{cases} \tag{16}$$

5.1.2 模型的求解

本文采用增广拉格朗日-交替极小化求解模型，通过采用分裂技术求解公式(16)，设 $J=Z$，$S=Z, F=E$ 式等价于:

$$\begin{cases}\min\limits_{Z,E,J,S,F}\|Z\|_1+\lambda\|J\|_*+\beta\|F\|_{2,1},\\ s.t.\quad D=(D-E)Z+E, J=Z, S=Z, F=E,\end{cases} \tag{17}$$

利用增广拉格朗日方法，有：

$$\max_{Y,W,P,Q}\min_{Z,E,J,S,F}\{\|Z\|_1+\lambda\|J\|_*+\beta\|F\|_{2,1}+tr\left(Y^T\left(D-\left(D-E\right)S-E\right)\right)$$
$$+tr\left(W^T\left(Z-S\right)\right)+tr\left(P^T\left(Z-J\right)\right)+tr\left(Q^T\left(E-F\right)\right)+ \quad (18)$$
$$\frac{\mu}{2}\left(\left\|D-\left(D-E\right)Z-E\right\|_F^2+\left\|Z-S\right\|_F^2+\left\|Z-J\right\|_F^2\right)+\left\|E-F\right\|_F^2\}$$

其中，Y,W,P,Q 为拉格朗日乘子矩阵，μ 为罚参数。利用交替极小化方法求解公式(18)，具体算法如下：

表 1　交替极小化解答步骤

输入	原始数据 D，参数 λ,β
输出	系数矩阵 Z
初始化	$Z_0=0$，$E_0=0$，$J_0=0$，$S_0=0$，$F_0=0$，$\mu_0=10^{-6}$，$\rho=1.1$，$\mu_{\max}=10^{10}$ $Y_0=0$，$W_0=0$，$P_0=0$，$Q_0=0$，$k=0$
步骤一	对 J 求极小，得: $J_{k+1}=\arg\min_J\|J\|_*+\frac{1}{2\left(\lambda/\mu_k\right)}\left\|J-\left(Z_k+\frac{P_i}{\mu_k}\right)\right\|_F^2$；
步骤二	对 S 求极小，得 $S_{k+1}=\left(\left(E_k-D\right)^T\left(E_k-D\right)+I\right)^{-1}\left(Z_k+\left(D-E_k\right)^T\left(D-E_k\right)+\frac{W_k+\left(D-E_k\right)^TY_k}{\mu_k}\right)$
步骤三	对 Z 求极小，得 $Z_{k+1}=\arg\|Z\|_1+\frac{1}{2\left(1/2\mu_k\right)}\left\|Z-\left(\frac{S_{n+1}+J_{n+1}}{2}+\frac{W_k+P_k}{2\mu_k}\right)\right\|_F^2$
步骤四	对 F 求极小，得 $F_{k+1}=\arg\min_F\frac{\beta}{\mu_k}\|F\|_{2,1}+\frac{1}{2}\left\|F-\left(E_k+\frac{Q_k}{\mu_k}\right)\right\|_F^2$；
步骤五	对 E 求极小，得 $E_{k+1}=\left(S\left(S_{k+1}-I\right)\left(S_{k+1}-I\right)^T+F_{k+1}-\frac{Y_k\left(S_{k+1}-I\right)^T+Q_k}{\mu_k}\right)\left(\left(S_{k+1}-I\right)\left(S_{k+1}-I\right)^T+I\right)$
步骤六	$Y_{k+1}=Y_k+\mu_k\left(D-\left(D-E_{k+1}\right)S_{k+1}-E_{k+1}\right)$, $W_{k+1}=W_k+\mu_k\left(Z_{k+1}-S_{k+1}\right)$, $P_{k+1}=P_k+\mu_k\left(Z_{k+1}-J_{k+1}\right)$, $Q_{k+1}=Q_k+\mu_k\left(E_{k+1}-F_{k+1}\right)$,

步骤七	$\mu_{k+1}=\min\{\rho\mu_k,\mu_{\max}\},\ k+1\leftarrow k;$
验证收敛条件	$\frac{\|D-(D-E)S-E\|_F}{\|D\|_F}<1\times10^{-6},\frac{\|Z-J\|_F}{\|D\|_F}<1\times10^{-6}$ $\frac{\|Z-S\|_F}{\|D\|_F}<1\times10^{-6},\frac{\|E-F\|_F}{\|D\|_F}<1\times10^{-6}.$

本文运用 CVX_MATLAB 工具包通过迭代求出最优近似解 Z^*，则相似度矩阵为:

$$S_{i,j}=\left(\left|Z_{i,j}^{*}\right|+\left|Z_{j,i}^{*}\right|\right)/2 \tag{19}$$

由于在实际应用中 LRR 得到的相似度矩阵稀疏性较差，SSC 对噪声敏感，且对于图像分割来讲，特征数据不一定准确满足子空间聚类的假设．所以，本文据此给出一个加权稀疏子空间表示模型，引入权重使得数据尽可能地由最相似的数据线性表示。

利用高斯相似度函数：

$$w_{ij}=\exp\left(-\frac{\left\|d_i-d_j\right\|_2^2}{\sigma^2}\right) \tag{20}$$

式中，d_i 和 d_j 是 2 个数据样本，对 l_1 范数加权，建立如下加权稀疏表示模型

$$\sum_{j\neq i}\frac{1}{w_{ji}}\left|s_{ji}\right| \tag{21}$$

该模型中，相似度越大则权重越小，相似度越小则权重越大，从而加权稀疏约束有利于使得数据尽可能被最相似的数据线性表示，与不相似的数据无关。构造相似度矩阵，利用成熟的谱聚类算法，如规范化割（Ncut）就可以得到最终聚类结果。

Ncut 算法是将图像的各个像素视为无向图 G 的顶点 V，顶点之间的边 E 视为像素之间的特征相似度，得到的无向图记为 $G=(V，E)$。再将 V 划分成两个区域的两划分问题，Ncut 最优化的图划分目标函数如公式所示。

$$\mathrm{NCut}(A,B)=\frac{cut(A,B)}{assoc(A,V)}+\frac{cut(A,B)}{assoc(B,V)} \tag{22}$$

其中：

$$\begin{cases} cut(A,B)=\sum_{u\in A,v\in B} w(u,v) \\ assoc(A,V)=\sum_{u\in A,t\in V} w(u,t) \\ assoc(B,V)=\sum_{u\in B,t\in V} w(u,t) \end{cases} \quad (23)$$

Ncut 算法分为二路划分和多路划分两种，二路划分在进行分割时仅仅使用拉普拉斯矩阵第二小特征值对应的特征向量。本文主要使用二路划分方法。基本步骤如下：

步骤一：根据无向图 $G=(V,E)$，生成对应的相似矩阵 $W_{l\times l}$，l 为图像中像素点的个数。W 中的每个元素 W_{ij} 为原图像第 i 个像素点和第 j 个像素点的相似度；

步骤二：计算度矩阵 $D_{l\times l}$，D 是对角阵，对角线上的每个元素 $d_i=\sum_j w_{ij}$；

步骤三：求出等式 $(D-W)x=\lambda Dx$ 的第二小特征值 λ_2 对应的特征向量 x_2。将向量 x_2 划分成两类，可以用多种方法进行划分，其中以 x_2 的中值为界进行划分较为简单；

步骤四：通过计算 Ncut 的目标函数，判断划分 x_2 得到的两类是否需要继续划分，如果目标函数小于某个阈值，则转到步骤三继续进行递归分割。

5.1.4 第一问的结果

为了验证本文模型及算法对聚类的有效性，选用 1.bat 数据文件作为测试对象。按照上述算法步骤，得出的聚类结果如下：

表 2　问题 1 中的分类分布（红色为 1 类，蓝色为 2 类）

序号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
分类	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
序号	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
分类	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
序号	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
序号	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
序号	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
序号	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
分类	1	2	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1
序号	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
序号	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
分类	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
序号	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180

分类	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
序号	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
分类	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

模型总结：

上述模型成功地将高维数据按照相似性分成了两类。而且模型不需要太多的先验信息。接着进一步分析本文算法的计算复杂度。本文算法的复杂度有三个部分组成：计算相似矩阵 W 以及对 W 运用谱聚类算法。第一部分该过程的计算复杂度为: $O(NM(t_1+dt_2))$。其中 t_1 和 t_2 分别为聚类算法达到收敛所需的迭代时间以及相似性加权矩阵的计算时间， N 为数据数量， M 为子空间个数。第二部分对 W 使用谱聚类算法将数据规划为一个 k 维相嵌矩阵，然后使用二路划分算法把数据分为 k 类。解广义特征向量问题的复杂度为 $O((N+k)N^2)$，二路划分算法迭代 t_3 次处理 k 维相嵌矩阵的复杂度为 $O(Nk^2t_3)$。因此，SMMC 算法总的时间复杂度为：

$$NM(t_1+dt_2)+k^2t_3)+O((N+k)N^2) \quad (24)$$

5.2 问题二的模型建立以及求解

5.2.1 基于核映射的模型改进

➢ 核函数的基本理论：

核函数方法的基本原理是：当样本集非线性可分时，使用一个非线性变换式把输入模式空间 R 中的数据映射到高维特征空间 F 中，即 $R\to F$，然后在 F 中构造新的分类函数。值得注意的是，我们只需要利用核函数 $K(x,y)=\varphi(x)\varphi(y)$ 替代内积运算即可，而不用明确知道非线性变换的具体方式。通常情况下，变换函数 $\varphi(x)$ 比核函数 $K(x,y)$ 更为复杂，简单的核函数往往和复杂的映射是相对应的。只要选择能够满足 Mercer 条件的核函数，就可以大大降低非线性变换的计算量。由核函数方法的基本原理可知设计新的核方法的两个关键问题：一是如何选择和设计核函数以及优化核函数中的相关参数，二是采用何种学习算法并对学习算法进行改进，根据采用学习算法的不同可以构成出不同的核方法，将核函数引入到聚类方法中就可以构成核聚类方法。

一个函数必须满足 Mercer 条件才能能够成为核函数。目前可以通过多种方法构造核函数，已经有文献证明了，对于给定的样本，其核函数必然存在，核函数的选择是否合适是影响分类效果的关键因素。但是，至今仍没有一个统一的方法能够解决核及其参数确定的问题。根据核函数的现有发展理论，在实际应用中被广泛使用的主要是以下几种核函数：

1）多项式核函数

$$K(x,y)=(xy+1)^d, d=1,2,\cdots \quad (25)$$

多项式核函数是典型的全局性非线性映射函数，应用比较广泛，其复杂程度是由其多项式的阶数 d 确定的。在参数 $d=1$ 的情况下，多项式核函数会退化为一个线性函数，随着 d 的增大，多项式的空间复杂程度呈指数级的增长。当 d 很大时，如果 $(xy+1)>1$，则核函数的值会趋于 $+\infty$，如果 $(xy+1)<1$，则核函数的值会趋于

0，这导致数据处理面临很大的困难，在实际应用中 d 的值通常必须控制在一定范围之内。由于多项式核函数是典型的全局性核函数，因此它允许距离较远的样本点对核函数的值有较大的影响。

2）高斯核函数

$$K(x,y)=\exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \tag{26}$$

其中 σ 为尺度参数，其大小直接影响高斯核函数的性能好坏。

(1)若高斯核函数中尺度参数 σ 趋于 0，则拉格朗日乘子向量的所有分量都大于 0，即全部样本点都是支持向量的。

(2)当 σ 趋于无穷时，判别函数退化为常函数，此时其推广能力为零，并且无法对新样本的正确分类，会把所有样本点判为同一类。

根据以上叙述，只要选择合适的 $\sigma>0$ 于，高斯核函数能够对任意给定的训练集正确分类。现有证明说明了高斯核函数是典型的局部性核函数，距离较远的样本对核函数的值影响较小。

3）Sigmond 核函数：

$$K(x,y)=\tanh\left[b(x,y)-c\right] \tag{27}$$

其中 b，c 为常数。Sigmoid 核函数中只有参数 b 和 c 的某些值满足 Mercer 条件。一般地，当满足 $b<0$ 和 $c<0$ 时，Sigmoid 核函数较适合作为核函数，并且 b 取值较小时与 σ 取值较小时的高斯核函数性能相当。相对于其他核函数，Sigmoid 核函数并没有特别的优势，而且参数选择较为复杂和困难，所以一般不采用 Sigmoid 核函数。

根据分析，本文选择高斯核函数。一方面本文在文献中证明了高斯核函数的良好性能，另一方面高斯核函数的核值范围为(0,1)，这将使得计算过程比较简单。

下面通过简单的列子，说明核函数把低维空间映射到高维空间的过程。下图位于第一、二象限内。我们关注红色的门，以及“北京四合院”这几个字下面的紫色的字母。将红色门上的点看成是“+”数据，紫色字母上的点看成是“−”数据，它们的横、纵坐标是两个特征。显然，在这个二维空间内，“+”“−”两类数据不是线性可分的。

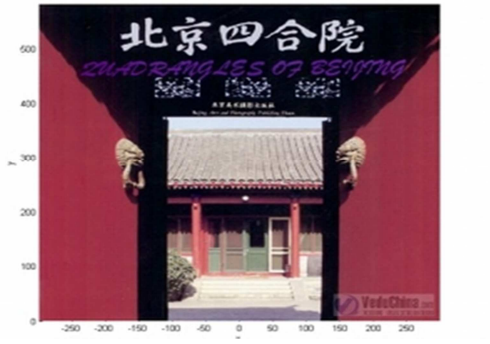


图 6　核函数示意图

现在我们考虑核函数：

$$K\left(v_1,v_2\right)=\left\langle v_1,v_2\right\rangle^2 \tag{28}$$

其中，$v_1=\left(x_1,y_1\right),v_2=\left(x_2,y_2\right)$是二维空间中的两个点。这个核函数对应着一个二维空间到三维空间的映射，它的表达式是：

$$P\left(x,y\right)=\left(x^2,\sqrt{2}xy,y^2\right) \tag{29}$$

可以验证：

$$\begin{aligned}&\left\langle P\left(v_1\right),P\left(v_2\right)\right\rangle=\left\langle\left({x_1}^2,\sqrt{2}x_1y_1,{y_1}^2\right),\left({x_2}^2,\sqrt{2}x_2y_2,{y_2}^2\right)\right\rangle\\&={x_1}^2{x_2}^2,2x_1y_1x_2y_2,{y_1}^2{y_2}^2\\&={x_1}^2{x_2}^2,2x_1y_1x_2y_2,{y_1}^2{y_2}^2\\&=\left\langle v_1,v_2\right\rangle^2\\&=K\left(v_1,v_2\right)\end{aligned} \tag{30}$$

在 P 这个映射下，原来二维空间中的图在三维空间中的像是这个样子：

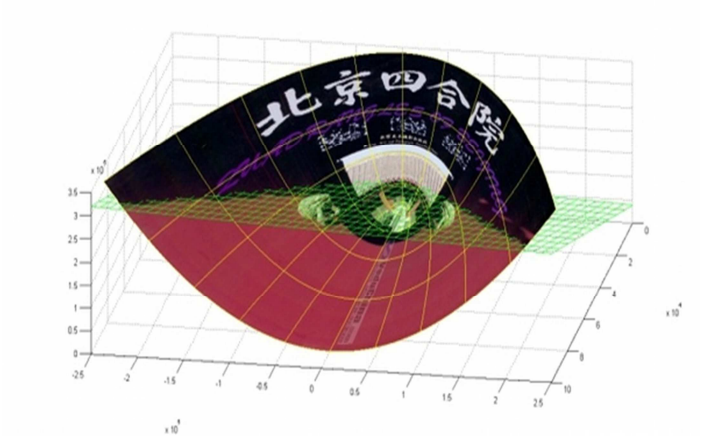


图 7　核函数示意图

其中，前后轴为 x 轴，左右轴为 y 轴，上下轴为 z 轴。有图可以看出，绿色平面可以完美地分割红色和紫色，也就是说，两类数据在三维空间中变为线性可分的了。而将三维空间中的这个判决边界，再映射回二维空间中是如下图这样的：

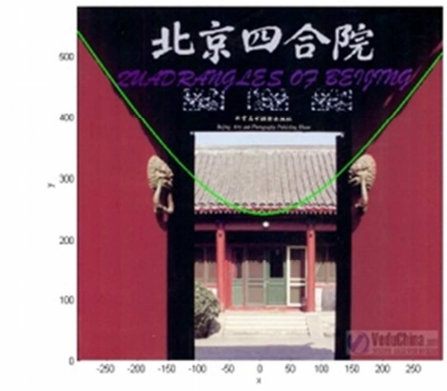


图 8　核函数示意图

这是一条双曲线，是非线性的。如上面的例子所说，核函数的作用就是隐含着一个从低维空间到高维空间的映射，而这个映射可以把低维空间中线性不可分的两类点变成线性可分的。这样，就可以期待原来并不线性可分的两类点变成线性可分了。事实中使用的核函数往往比这个例子复杂得多。它们对应的映射并不一定能够显式地表达出来；它们映射到的高维空间的维数也比所举例子（三维）高得多，甚至是无穷维的。其实问题二中的（c）是与这个类似的。

- 基于核映射子空间聚类算法

在l_1图的构建中加入核映射的思想，可以使不独立的子空间在更高维的空间维度中变成独立的。由此，我们提出一种基于核映射升维的子空间谱聚类方法。该算法的基本思想是：首先将数据集合通过核映射至特征空间得到新的样本集合，若新的样本维数过大，可以采用降维方法进行处理，然后在特征空间内对数据通过前文的谱聚类的方法达到子空间聚类的目的，具体步骤表述如下:

表 3　基于核低秩编码的子空间聚类算法

输入：	数据矩阵 D，参数 λ，类别数 k
输出：	分割结果
步骤一：	对样本矩阵 D 进行列归一化，使得每列样本的 l_2 范数为 1
步骤二：	将样本矩阵 D 通过核函数映射至特征空间，得到映射后的样本 D_1
步骤三：	对 D_1 进行降维处理，得到降维后的样本矩阵 D_2
步骤四：	按下式求解得到系数矩阵 Z $\begin{cases}\min\limits_{Z,E}\|Z\|_1+\lambda\|Z\|_*+\beta\|E\|_{2,1},\\ s.t.D=(D-E)Z+E,\end{cases}$
步骤五：	利用 Z 构造相似度矩阵
步骤六：	采用 Ncut 方法进行聚类

上表步骤二中样本矩阵 X 通过核函数映射至特征空间时，采用的是高斯核函数:

$$k(x,y)=\exp\left(-\|x-y\|^2/2p^2\right) \tag{31}$$

其中 $p \in R, x, y$ 分别为样本集合 D 中的任意样本。

5.2.2 基于局部相关切向函数的模型改进

考虑两组在混合流体的原始数据 d_i 和 d_j 之间的情况，这主要是针对相似函数的改进。首先为它们的局部相关切线空间定义一个新的函数（称作相似结构体 p_{ij}），假设原有的相似度函数为 q_{ij}。然后，混合两个相似函数得到最终的相似值：

$$S'_{ij} = f(p_{ij}, q_{ij}) \tag{32}$$

其中 f 是一个合适的融合函数。值得注意的是，为了得到相似矩阵的预期性质， f 应该是原有相似度的单调递增函数，同时还是两个切线空间相似点的单增函数。现在，我们给出 p ， q 以及当前算法中使用的 f 函数的正确公式。假设数据 d_i（ $i = 1, \cdots, N$ ）处的切线空间为 Θ_i，那么 d_i 和 d_j 局部切线空间的相似性结构体可以定义为：

$$p_{ij} = p(\Theta_i, \Theta_j) = (\prod_{l=1}^{d} \cos(\theta_l))^o \tag{33}$$

在上式（33）中， $o \in N^+$ 为可调参数，为固定值。 $0 \le \theta_1 \le, \cdots, \le \theta_d \le \pi/2$ 是两个切线空间 Θ_i 和 Θ_j 之间的一系列主要角度，递归定义如下：

$$\cos(\theta_1) = \max_{\substack{u_1 \in \Theta_i, v_1 \in \Theta_j \\ \|u_1\|=\|v_1\|=1}} u_1^T v_1 \tag{34}$$

并且：

$$\cos(\theta_l) = \max_{\substack{u_l \in \Theta_i, v_l \in \Theta_j \\ \|u_l\|=\|v_l\|=1}} u_l^T v_l, \quad l = 2, \cdots, d \tag{35}$$

其中 $u_l^T u_l = 0$， $v_l^T v_l = 0$， $i = 1, \cdots, l-1$。为了方便叙述以及简单推导，这里假设局部相似一个极端的定义，即两种数据在原来聚类算法中只有相似与不相似：

$$q_{ij} = \begin{cases} 1 & if\ d_i \in Knn(d_j)\ or\ d_j \in Knn(d_i) \\ 0 & otherwise \end{cases} \tag{36}$$

其中 $Knn(x)$ 代表与 x 相似的聚类。最后，将以上两个函数简单相乘融合得到相似值：

$$s_{ij} = p_{ij} q_{ij} = \begin{cases} (\prod_{l=1}^{d} \cos(\theta_l))^o & d_i \in Knn(d_j) \\ & d_j \in Knn(d_i) \\ 0 & otherwise \end{cases} \tag{37}$$

容易验证公式（37）能够达到预期的目标，一方面，基于原有的聚类算法，当点相距较远时候，他们的相似值较小。另一方面，属于不同类子空间或流形时其值相对较低。这是因为，一对来自不同流形的点相距较远，它们的相似值为 0。当它们靠近不同流形的交叉区域时，它们将拥有不同的局部切线空间，当参数 o 足够大时它们也将拥有一个相对较小的相似值。因此，我们可以预期将该相似矩阵应用于谱聚类会得到更好的结果。上述公式还有一个未解决的问题，就是如何

有效近似每个采样点的局部切线空间。在下一部分，我们将详细讨论这个问题。

5.2.3 模型的求解

一般而言，切线空间可以由给定的采样点的局部邻域获得。准确地说，给定一个数据 d_i 和它在该子空间的 n 个最近邻域 $N(d_i)=\{d_1,\cdots,d_n\}$，$d_i$ 附近的局部几何信息可以通过它的局部样本协方差矩阵 $\sum_x$ 获得。其中协方差矩阵定义如下：

$$\sum_x = 1/n\sum_{i=1}^{n}(d^i-u_x)(d^i-u_x)^T \tag{38}$$

其中 $u_x = 1/n\sum_{i=1}^{n} d_i$。然后，$d_i$ 点处的局部切线空间 Θ_x 通过 $\sum_x$ 的维度的左歧义向量近似而来，维度 M 的左歧义向量对应于 M 个最大奇异值。于是，设 $\sum_x$ 的 SVD 为：

$$\sum_x = \begin{bmatrix} U_M & \tilde{U}_M \end{bmatrix} \begin{bmatrix} \sum_M & 0 \\ 0 & \tilde{\sum}_M \end{bmatrix} \begin{bmatrix} V_M & \tilde{V}_M \end{bmatrix}^T \tag{39}$$

其中 $[U_M\ \tilde{U}_M]\in R^{M'\times M'}$ 为正交矩阵，并且 $U_d \in R^{M'\times M}$。其中 M' 为所有子空间的并。于是，我们有：

$$\Theta_x = span(U_M) \tag{40}$$

不幸的是，根据公式（40），当两组数据点 d_i 和 d_j 相距很近时，即使它们来自不同的流形结构，但是它们的局部切线空间 Θ_x 和 Θ_y 将会非常相似。这是因为它们基于欧氏距离的局部邻域 $N(x)$ 和 $N(y)$ 发生了严重的重叠，导致产生相似的相似点局部协方差矩阵 $\sum_x$ 和 $\sum_y$。

因此，这种传统的局部切线空间定义无法有效处理混合非线性模型。此时，我们给出一种有效并且高效的方法对每个点的局部切线空间进行近似。我们的基础思想：对于整体非线性流形可以通过获取一系列局部线性流形进行有效的近似，其次，通过主成分分析可以成功得到交叉的线性流形。此外由同分析器近似而来的点通常拥有相似的局部切线空间，那么其也可由局部分析器的主子空间近似得到。因此，我们可以训练多个局部分析器来近似潜在的流形，然后，给定点的局部切线空间就可以由它的相关局部分析器的主子空间决定。

本文中，我们训练 Num 个混合概率主成分分析（MPPCA），其中每个分析器的特征是模型参数 $\theta_m=\{\mu_m,V_m,\sigma_m^2\}$，$m=1,\cdots,Num$，其中 $\mu_m\in R^M$，$V_m\in R^{M'\times M}$，并且 σ_m^2 是一个标量。应该注意的是，Num 和所有的局部线性的流形都有关，因为这些局部线性流形用于近似得到所有的线性或非线性流形组成基础数据集。在第 m 个分析器中，一个 M' 维观测数据 d_i 和一个相应的 M 维向量有关，并且其潜变量为 y，那么可以得到：

$$d_i = V_m y + \mu_m + \varepsilon_m \tag{41}$$

其中，μ_m是数据的一个鲁棒均值，并且潜变量 y 和噪声ε_m都是高斯函数，$y \sim \mathrm{N}(0, I)$，$\varepsilon_m \sim \mathrm{N}(0, \sigma_m^2 I)$。于是，$d_i$的边缘分布可以表示如下：

$$p(d \mid m) = (2\pi)^{-D/2} \mid C_m \mid^{-1/2} \exp\{-\frac{1}{2}(d-\mu_m)^T C_m^{-1}(d-\mu_m)\} \tag{42}$$

其中模型协方差为：

$$C_m = \sigma_m^2 I + V_m V_m^T \tag{43}$$

我们可以获得所有的模型参数μ_m，V_m和σ_m^2。其中，通过 EM 算法得到最优化观测数据$D = \{d_{ii}, i = 1, \cdots, N\}$的对数似然函数：

$$\hat{\lambda} = \sum_{i=1}^{N} \ln\{\sum_{m=1}^{Num} \pi_m p(d_i \mid m)\} \tag{44}$$

其中，π_m是混合比例，定义$\pi_m \geq 0$并且$\sum_{m=1}^{Num} \pi_m = 1$。具体 EM 学习方法的主要步骤为：

E-步骤：利用当前参数$\theta_m = \{\mu_m, V_m, \sigma_m^2\}$，计算：

$$R_{im} = \frac{\pi_m p(d_i \mid m)}{\sum_{m=1}^{M} \pi_m p(d_i \mid m)} \tag{45}$$

$$\pi_m^{new} = \frac{1}{N} \sum_{i=1}^{N} R_{im} \tag{46}$$

$$\mu_m^{new} = \frac{\sum_{i=1}^{N} R_{im} d_i}{\sum_{i=1}^{N} R_{im}} \tag{47}$$

M-步骤：重新估计参数V_m和σ_m^2：

$$V_m^{new} = S_m V_m (\sigma_m^2 I + T_m^{-1} V_m^T S_m V_m)^{-1} \tag{48}$$

$$(\sigma_m^2)^{new} = \frac{1}{d} tr[S_m - S_m V_m T_m^{-1} (V_m^{new})^T] \tag{49}$$

其中：

$$S_m = \frac{1}{\pi_m^{new} N} \sum_{i=1}^{N} R_{im} (d_i - \mu_m^{new})(d_i - \mu_m^{new})^T \tag{50}$$

$$T_m = \sigma_m^2 I + V_m^T V_m \tag{51}$$

此处注意，我们使用一般的聚类算法初始化 EM。最后，采样点d_i被划分到第j个局部分析器的概率：

$$p(x_i \mid j) = \max_m \; p(d_i \mid m) \tag{52}$$

而d_i的局部切线空间表达式如下：

$$\Theta_i = span(V_j)\ ) \tag{53}$$

我们可以估计用 M 个局部线性分析器来近似基础流行的重构误差：

$$error(M)=\sum_{i=1}^{M}\sum_{l=1}^{N_j}(d_l^j-\mu_j)^T(I-V_jV_j^T)(d_l^j-\mu_j) \quad (54)$$

其中 d_l^j（$l=1,\cdots,N_j$）就是被划分到第 j 个局部分析器的 N_j（$\sum_{j=1}^{M}N_j=N$）个点。具体步骤如下：

表 4　改进散发聚类步骤

输入：	数据矩阵 D，类数 k，流形的维数 d，混合模型数 M，邻域数 K，旋转因子 σ。
输出：	将数据分为互不相连的 k 个类
步骤一：	使用 MPPCA 方法训练 M 个 d 维局部线性流形来近似基础流形；
步骤二：	确定每个点的局部切线空间；
步骤三：	通过公式计算两个局部切线空间之间的成对相似值；
步骤四：	通过公式（37）计算成对的相似矩阵 $W\in R^{N\times N}$；
步骤五：	获得对角矩阵 E，$E_{ii}=\sum_j s_{ij}$；
步骤六：	提取 $(E-W)u=\lambda Eu$ 的第一个 k 广义特征向量 $u_1,\cdots,u_k$；
步骤七：	应用改进的聚类算法对在 R^k 空间 U 矩阵的行向量进行聚类。

5.2.4 第二问的结果：

基于上述步骤输入给定的数据，求得问题二中各个图像的聚类情况，如图所示：

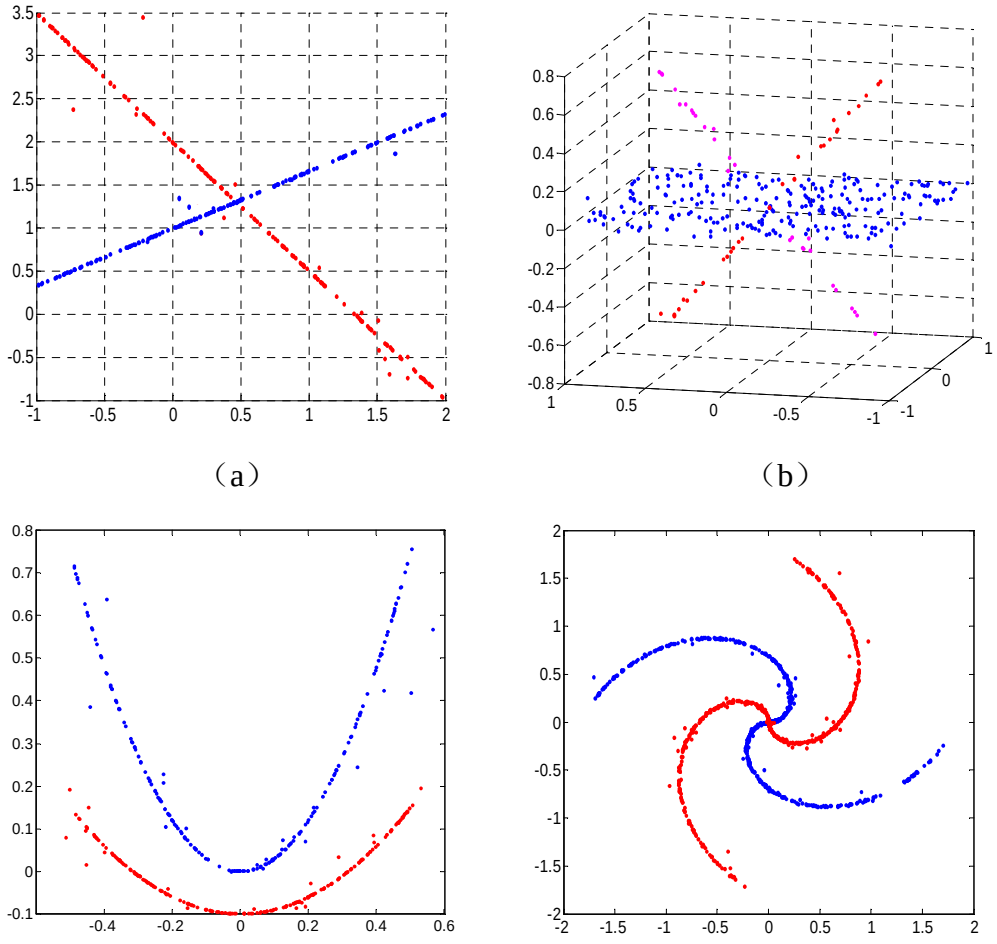


图 9　问题二解答结果

模型总结：

相对于前文来说，改进算法的计算复杂度主要增加了核函数的映射以及估计每个点的局部切线空间这两个内容。其中，核函数的映射增加了 $O(N^2t_4)$ 时间。算任意两个局部切线空间的相似值的复杂度为 $O(N^2M'M^2)$，其中 M' 为流体数量。从上述图中可以清晰地看出，尽管由于采用了核映射升维处理以及局部相关切线的因子，导致原来的算法负责度增加了 $O(N^2t_4+N^2M'M^2)$。导致运行速度减缓。但是由上图观测得知，我们的算法能够很好将上述各种情况进行聚类，准确度较高，误差率较低。

5.3问题三的模型建立以及求解

5.3.1模型的建立

- 基于 PCA 的降维处理方法：

PCA 是模式识别领域中最为有效的一种线性映射的方法，该方法通过提取原始空间数据中的主要特征，再由线性变换将提取的高维空间的特征映射到低维特

征空间中，减少数据的冗余，但优化保留了原始特征空间中的绝大部分有用特征信息，从而解决提取的特征维数过高的问题，即用尽可能少的信息来表征原有信息，达到数据压缩的目的。通常情况下，主成分拥有以下几个特点：

1、投影后的空间中的特征个数远远小于原有数据的个数，大大减少了运算量，提高了信号处理效率和识别速度。

2、通过投影变换提取的主要成分，保留了足够多的有用信息来进行有效聚类。

3、因为主成分之间相互独立且互不相关，利用这种独立、不相关性，能够降低数据的冗余特征。

PCA 方法是一种常用的正交变换，它的基础是 K-L 变换，变换的具体过程如下：

设有 *n* 维随机变量：

$$X=\sum_{i=1}^{n}\alpha_i\phi_i \tag{55}$$

其中，α_i为加权系数，ϕ_i为基向量，X 的另一种表达形式为：

$$X=(\phi_1,\phi_2,\cdots,\phi_n)(\alpha_1,\alpha_2,\cdots,\alpha_n)^T=\Phi\alpha \tag{56}$$

其中，$\Phi=(\phi_1,\phi_2,\cdots,\phi_n)$，$\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_n)^T$。基向量为正交向量，表达式为：

$$\Phi_i^T\Phi_j=\begin{cases}1,i=j\\0,i\neq j\end{cases} \tag{57}$$

由于Φ由多个正交向量构成，因此为正交矩阵，可表示为：

$$\Phi_i^T\Phi_j=I \tag{58}$$

将上式两边左乘，得到

$$\alpha=\Phi^T X \tag{59}$$

通过正交向量表示即为：

$$\alpha_i=\Phi_i^T X \tag{60}$$

设

$$R=E\left[X^T\quad X\right] \tag{61}$$

将式(56)代入式(61)中，得到

$$R=E\left[X^T\quad X\right]=E\left[\Phi\alpha\alpha^T\Phi^T\right]=\Phi E\left[\alpha\alpha^T\right]\Phi^T \tag{62}$$

令：

$$E\left[\alpha_j\alpha_k\right]=\begin{cases}\lambda_j,j=k\\0,j\neq k\end{cases} \tag{63}$$

其矩阵表达形式为：

$$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n \end{bmatrix} = \Lambda \tag{64}$$

那么

$$R = \Phi\Lambda\Phi^T \tag{65}$$

将式两边同时右乘Φ得：

$$R\Phi = \Phi\Lambda\Phi^T\Phi \tag{66}$$

由于Φ是正交矩阵，公式（66）可简化为：

$$R\Phi = \Phi\Lambda \tag{67}$$

则

$$R\Phi_j = \lambda_j\Phi_j \left(j = 1,2,\cdots,n\right) \tag{68}$$

可见，对于Φ_j来说，对应的特征值为R，并且这个特征值为λ_j的自相关实对称矩阵，所以，不同特征向量之间正交。

综上所述，计算步骤如下：

1、对于向量X，首先求取自相关矩阵$R = E\left[X^T \quad X\right]$，把协方差矩阵$\Sigma = E\left[\left(x-\mu\right)\left(x-\mu\right)^T\right]$作为 K-L 的产生矩阵，其中$\mu$为均值向量。

2、求取特征值λ_j和特征向量Φ_j。

3、系数$\alpha = \Phi^T X$。

整个变换过程的目的就是保留那些包含信息较多的特征向量，去掉一些包含信息较少的特征向量，因此，在原空间的基础上改变原来特征向量之间的相关性，达到空间降维的目的。

5.3.2模型的求解

表　5 在加入 PCA 方法用于数据聚类的具体步骤

步骤一	读入数据，生成数据矩阵
步骤二	计算 K-L 变换的生成矩阵： 生成矩阵表示为： $\Sigma = \frac{1}{M}\sum_{i=0}^{M-1}\left[\left(x-\mu\right)\left(x-\mu\right)^T\right]$ 其中μ是均值，M是训练集的个数

步骤三	计算特征值和特征向量： 用来表示人脸的特征向量被定义为脸部特征，脸部特征空间就是由所有的脸部特征组合而成的。在脸部特征空间中，将特征值降序排列，并将相应的特征向量也进行调整，大的特征值对应的特征向量称为主成分，用于描述人脸的轮廓，而较小的特征值对应的特征向量用于表示人脸的具体细节。
步骤四	将图像投影到特征空间中：把训练集和测试集两个集合中的所有的人脸图像映射到特征子空间中，特征子空间中的每一个坐标点代表一幅图像，可作为人脸识别的依据。
步骤五	求解系数矩阵：相似度函数，后采用 NCUT 方法进行聚类

5.3.3具体结果

针对上文所述算法，将给定的数据按照上述步骤进行求解，求得问题三中各个情况的聚类结果，如下所示：

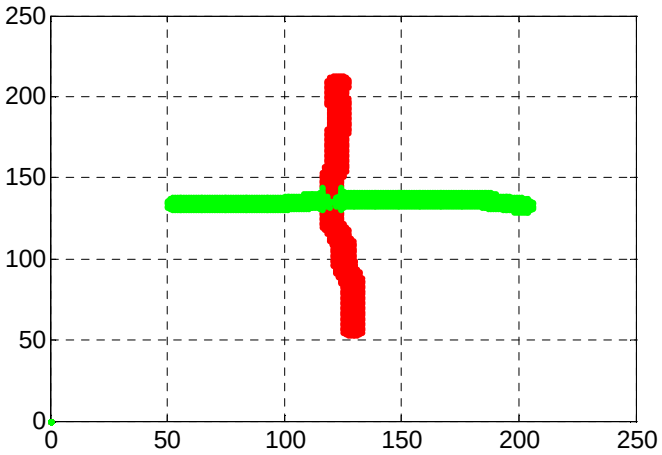


图 10　问题三（a）中结果

表 5 题目 3c 人脸目标分类分布（红色为 1 类，蓝色为 2 类）

序号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
分类	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2

表 6　题目 3 运动目标分类分布（红色为 1 类，蓝色为 2 类，紫色为 3 类）

序号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
序号	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
序号	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
序号	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
序号	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
序号	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
序号	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	
分类	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	
序号	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	
分类	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
序号	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	
分类	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
序号	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	
分类	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
序号	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	
分类	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
序号	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	
分类	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
序号	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	
分类	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
序号	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297				
分类	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3				

模型结论：

本次模型虽然增加了 PCA 处理步骤，表面上增加了计算复杂度$O(N^2(t_5+t_6))$，其中t_5和t_6分别为计算 K-L 变换以及特征值所需要。但是其降低了目标数据维度。因此其提升的时间为：

$$O(N^2(t_5+t_6))-(O(\text{原有算法在}M\text{维上的时间})-O(\text{原有算法在新}M_1\text{维上的时间})) \quad (69)$$

其中，$O(\text{原有算法在}M\text{维上的时间})$参见前文问题二的模型结论。

5. 4 问题四的模型建立以及求解

5.4.1 问题四模型的建立

首先对现有的大数据处理方法（主要是 Map-Reduce 编程模型）进行介绍。具体而言该算法的主要思路就是首先将大数据集分解成千上百个小数据集，每个小数据集分别由集群中的 1 个节点并行执行 Map 计算任务并生成中间结果，然后这些中间结果多节点并行执行 Reduce 计算任务，形成最终结果。Map-Reduce 执行过程如图 11 所示。

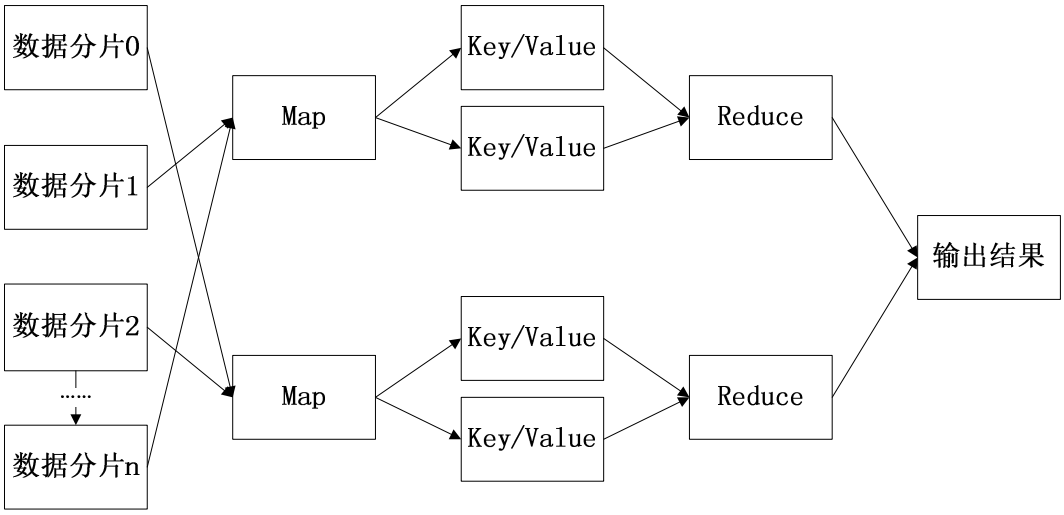


图 11　Map-Reduce 执行过程

5.4.2 本文改进算法

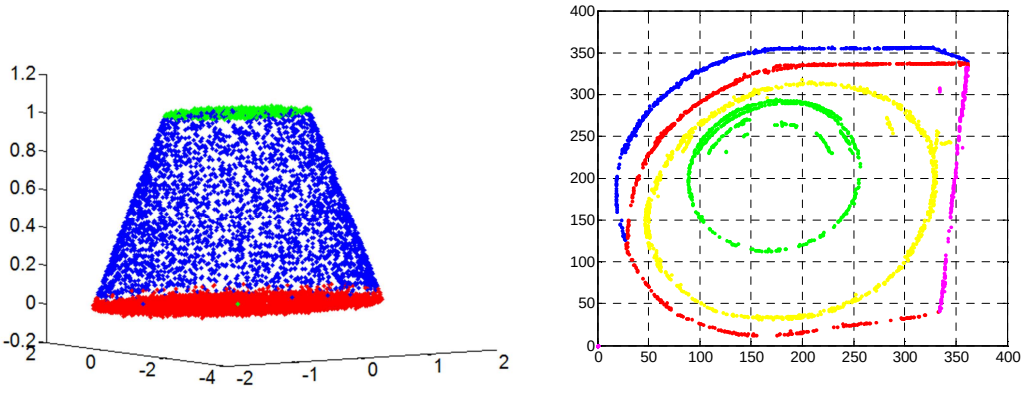
改进算法的思路如下：

设数据集 $D=\{d_1,d_2,\cdots,d_n\}$，且 $d_i \in R^d$ 其中 n 为样本个数，d 为样本维度，k 为聚类个数，m 为迭代次数。本文算法是首先结合随机抽样方法从样本集 D 中抽取一个规模较小的工作集 D^1，设 $|D^1|=|D|/s_I$，其中 s_I 为抽样因子，一般取值在 5-100 之间（即抽样数据是原始数据 $1\%-20\%$），取值视原始数据量而定。然后，采用前文的稀疏聚类算法，对数据进行聚类，并计算器聚类中心 C_1。再以 C_1 作为据的聚类中心 C。由于稀疏聚类之间的计算相互独立的，所以，可以使用 Map Reduce 框架实现计算的并行化，提高计算的效率。分别再次抽取当中的数据，计算新的聚类中心 C_2。依次抽取所有数据，直到执行结束，返回所有聚类中心结果。之后对这些聚类中心，采用谱聚类方法，再次归为三类。具体求解步骤：

表 7　改进算法解步骤

步骤一	从数据集 $D=\{d_1,d_2,\cdots,d_n\}$ 中随机抽取 n/s 个样本数据构成抽样样本 D_1。
步骤二	采用前文的聚类方法，将 D_1 中数据分成规定的类别，并计算出其聚类中心。
步骤三	得到了聚类结果之后，对这些聚类中心，再次采用谱聚类方法，合并相同的，归为四类，得出聚类结果。

5.4.3 模型结果



（a） （b）

图 12　问题四计算结果

模型结论：

本次模型采用了并处理思路，通过抽样将数据分成了多块同时处理。简单来说，就是将同一组数据在多个计算平台上同时处理，大大缩减了处理时间。通过这个流程可以分析出整个程序的时间复杂度为：

$$O\left(\text{原有时间}/\left(M*N\right)\right) \tag{70}$$

其中，M 是执行作业的 Map 个数，N 是集群中执行该任务的结点数。而且，由上图可以得知，我们的算法并没有改变聚类效果。其中，图 12（a）中，虽然有少量的杂波，但是总地来说正确地将来自三类不同流形的数据分开。图 12（b）中，我们按照算法很好地将工件分成了五块。这里考虑五块的主要原因在于：首先是区分处于不同子空间的数据，其次是区分子空间的相交数据。

6. 模型的总结与讨论

本文围绕谱聚类基础算法，探讨了背后的原理，并在此基础之上提出改进，创造性地提出了 SSC-LRR 合并谱聚类算法，并引入了核映射、局部切线以及 PCA 降维技术等概念进行了算法优化，重点研究了其在独立子空间聚类、非独立子空间聚类、非线性空间聚类、混合多流体聚类以及实际生活中运动分割、人脸识别以及工件识别等方面的应用，最后还提出了减少运算复杂度的并行处理算法。

- 全文的主要工作和创新点归纳如下：

1、研究了基于谱聚类框架的子空间聚类算法，主要是 SSC 与 LRR 类算法，并总结了基于这两种基本算法而引伸出的其他相关类算法。指出并证明了稀疏表示以及低秩表示在数据聚类问题上的不足之处。此外，我们还由此提出了两者结合的算法，并添加了相似矩阵权值用于最终的算法。

2、针对问题二中出现的非独立子空间结构、非线性结构以及混合多流体结构，本文探讨了核映射以及局部切线理论，最终提出了一种结合核映射，并且融合局部切线理论的改进模型，这是对聚类的一种推广，不仅能够解决简单的独立子空间聚类，同时也能解决其他相关问题。

3、针对问题三中人脸识别以及运动分割问题，本文提出了利用 PCA 方法进行降维处理，减少了运算量。

4、针对问题四中的大数据，本文又根据 map-Reduce 算法，提出了使用于本文的并行处理方法，大大地缩减了运行时间。

- 模型的不足以及工作展望

1、我们提出的加权的聚类算法权重设置采用测试数据之间的距离来计算，这种权重设置方式的缺点在于其在确定权重时只与测试数据有关。如果实际噪声较大，处理方式则不妥；

2、应该更加充分挖掘数据的先验信息并由此设计正则项；

3、本文提出的方法，包括提出的相关优化算法，其时间的复杂度较高，原因在于我们需要对所有数据求解稀疏表示，而且采用了迭代算法，计算量太大，一般应用于处理低维空间。在人脸识别、运动分割或者大数据聚类等诸多应用中，采用稀疏表示技术的一个关键在于所使用的字典。在较大规模的学习问题中，一个大规模的字典一般会导致求解效率的下降，而且字典中的原子并非越多越好。因此，如何构造更好的字典以应对不同的学习问题也是稀疏表示技术的关键所在。所以这些方法离实际应用仍有较大距离，更高效的稀疏表示优化算法仍然是研究的重点。

参考文献

[1] Y. Wang, Y. Jiang, Y. Wu, and Z. Zhou. Spectral clustering on multiple manifolds. IEEE Transactions on Neural Networks, 22(7):1149–1161, 2011.

[2] R. Vidal. Subspace clustering. IEEE Signal Processing Magazine, 28(2):52–68, 2011.

[3] B. Cheng, G. Liu, J. Wang, Z. Huang, and S. Yan, Multi-task low rank affinity pursuit for image segmentation, ICCV, 2011.

[4] C. Lang, G. Liu, J. Yu, and S. Yan, Saliency detection by multitask sparsity pursuit, IEEE Transactions on Image Processing, 21(3): 1327–1338, 2012.

[5] 霍焰焰，基于 PCA 和 LBP 改进算法的人脸识别研究[D]，黑龙江哈尔滨：哈尔滨理工大学，2015:31-33。

[6] 杜立翠，基于映射投影空间的一类分类算法研究[D]，河北秦皇岛：燕山大学，2014:29-33。

[7] 靳正芬，求解矩阵核范数极小化问题的交替方向法[D]，河南开封：河南大学，2012:1-5。

[8] 曹姝晗，求解约束 l1-范数极小化问题的交替方向法[D]，河南开封：河南大学，2015:9-19。

[9] 李霞，徐树维，子空间聚类改进算法研究综述[J]，计算机仿真，2010，27:174-177.

[10] 李涛，王卫卫，翟栋等，图像分割的加权稀疏子空间聚类方法[J]，系统工程与电子技术，2014。

附录：

1、主程序

```
clc；
clear all；
close all
load 1
Cst=10；
OptM=2
Lambda=0.001
Xp=data
CMat = SparseCoefRecovery(Xp,Cst,OptM,lambda);
[CMatC,sc,OutlierIndx,Fail] = OutlierDetection(CMat,s);
if (Fail == 0)
    CKSym = BuildAdjacency(CMatC,K);
    Grps = SpectralClustering(CKSym,n);
    Grps = bestMap(sc,Grps);
    Missrate = sum(sc(:) ~= Grps(:)) / length(sc);
    save Lasso_001.mat CMat CKSym Missrate Fail
else
    save Lasso_001.mat CMat Fail
end
```

2、谱聚类功能函数（CVX 功能包在附件中）

```
function [groups, kerNS] = SpectralClustering(CKSym,n)
warning off;
N = size(CKSym,1);
MAXiter = 1000; % Maximum number of iterations for KMeans
REPlic = 20; % Number of replications for KMeans
DN = diag( 1./sqrt(sum(CKSym)+eps) );
LapN = speye(N) - DN * CKSym * DN;
[~,~,vN] = svd(LapN);
kerN = vN(:,N-n+1:N);
normN = sum(kerN .^2, 2) .^.5;
kerNS = bsxfun(@rdivide, kerN, normN + eps);
groups                                                                    =
kmeans(kerNS,n,'maxiter',MAXiter,'replicates',REPlic,'EmptyAction','singleton');
function CMat = SparseCoefRecovery(Xp,cst,Opt,lambda)
if (nargin < 2)
    cst = 0;
end
if (nargin < 3)
    Opt = 'Lasso';
end
if (nargin < 4)
    lambda = 0.001;
end
D = size(Xp,1);
N = size(Xp,2);
```

```
for i = 1:N
    y = Xp(:,i);
    if i == 1
        Y = Xp(:,i+1:end);
    elseif ( (i > 1) && (i < N) )
        Y = [Xp(:,1:i-1) Xp(:,i+1:N)];
    else
        Y = Xp(:,1:N-1);
    end

    % L1 optimization using CVX
    if cst == 1
        if ( strcmp(Opt , 'Lasso') )
            cvx_begin;
            cvx_precision high
            variable c(N-1,1);
            minimize( norm(c,1) + lambda * norm(Y * c    - y) );
            subject to
            sum(c) == 1;
            cvx_end;
        elseif ( strcmp(Opt , 'L1Perfect') )
            cvx_begin;
            cvx_precision high
            variable c(N-1,1);
            minimize( norm(c,1) );
            subject to
            Y * c    == y;
            sum(c) == 1;
            cvx_end;
        elseif ( strcmp(Opt , 'L1Noisy') )
            cvx_begin;
            cvx_precision high
            variable c(N-1,1);
            minimize( norm(c,1) );
            subject to
            norm( Y * c    - y ) <= lambda;
            sum(c) == 1;
            cvx_end;
        elseif ( strcmp(Opt , 'L1ED') )
            cvx_begin;
            cvx_precision high
            variable c(N-1+D,1);
            minimize( norm(c,1) );
            subject to
            [Y eye(D)] * c    == y;
            sum(c(1:N-1)) == 1;
            cvx_end;
```

```
        end
    else
        if ( strcmp(Opt , 'Lasso') )
            cvx_begin;
            cvx_precision high
            variable c(N-1,1);
            minimize( norm(c,1) + lambda * norm(Y * c    - y) );
            cvx_end;
        elseif ( strcmp(Opt , 'L1Perfect') )
            cvx_begin;
            cvx_precision high
            variable c(N-1,1);
            minimize( norm(c,1) );
            subject to
            Y * c    == y;
            cvx_end;
        elseif ( strcmp(Opt , 'L1Noisy') )
            cvx_begin;
            cvx_precision high
            variable c(N-1,1);
            minimize( norm(c,1) );
            subject to
            norm( Y * c    - y ) <= lambda;
            cvx_end;
        elseif ( strcmp(Opt , 'L1ED') )
            cvx_begin;
            cvx_precision high
            variable c(N-1+D,1);
            minimize( norm(c,1) );
            subject to
            [Y eye(D)] * c    == y;
            cvx_end;
        end
    end
    if i == 1
        CMat(1,1) = 0;
        CMat(2:N,1) = c(1:N-1);
    elseif ( (i > 1) && (i < N) )
        CMat(1:i-1,i) = c(1:i-1);
        CMat(i,i) = 0;
        CMat(i+1:N,i) = c(i:N-1);
    else
        CMat(1:N-1,N) = c(1:N-1);
        CMat(N,N) = 0;
    end
end
```

3、计算程序

```
function [CMatC,sc,OutlierIndx,Fail] = OutlierDetection(CMat,s)
```

```
n = max(s);
N = size(CMat,2);
NanIndx = [];
FailCnt = 0;
Fail = 0;
for i = 1:N
        c = CMat(:,i);
        if( sum(isnan(c)) >= 1 )
                NanIndx = [NanIndx ; i];
                FailCnt = FailCnt + 1;
        end
end
sc = s;
sc(NanIndx) = [];
CMatC = CMat;
CMatC(NanIndx,:) = [];
CMatC(:,NanIndx) = [];
OutlierIndx = NanIndx;

if ( FailCnt > N - n )
        CMatC = [];
        sc = [];
        Fail = 1;
End
function [C,T]=hungarian(A)
[m,n]=size(A);
if (m~=n)
        error('HUNGARIAN: Cost matrix must be square!');
end
orig=A;
A=hminired(A);
  [A,C,U]=hminiass(A);
while (U(n+1))
        % Start with no path, no unchecked zeros, and no unexplored rows.
        LR=zeros(1,n);
        LC=zeros(1,n);
        CH=zeros(1,n);
        RH=[zeros(1,n) -1];
        SLC=[];
        r=U(n+1);
             LR(r)=-1;
        SLR=r;
        while (1)
                if (A(r,n+1)~=0)
                        l=-A(r,n+1);
                        if (A(r,l)~=0 & RH(r)==0)
                                RH(r)=RH(n+1);
                                RH(n+1)=r;
```

```
                CH(r)=-A(r,l);
            end
        else
            if (RH(n+1)<=0)
                % Reduce matrix.
                [A,CH,RH]=hmreduce(A,CH,RH,LC,LR,SLC,SLR);
            end
            r=RH(n+1);
            l=CH(r);
            CH(r)=-A(r,l);
            if (A(r,l)==0)
                RH(n+1)=RH(r);
                RH(r)=0;
            end
        end
        while (LC(l)~=0)
            if (RH(r)==0)
                if (RH(n+1)<=0)
                    [A,CH,RH]=hmreduce(A,CH,RH,LC,LR,SLC,SLR);
                end
                r=RH(n+1);
            end
            l=CH(r);
            CH(r)=-A(r,l);
            if(A(r,l)==0)
                RH(n+1)=RH(r);
                RH(r)=0;
            end
        end
        if (C(l)==0)
            [A,C,U]=hmflip(A,C,LC,LR,U,l,r);
            break;
        else
            LC(l)=r;
            SLC=[SLC l];
            r=C(l);
            LR(r)=l;
            SLR=[SLR r];
        end
    end
end
T=sum(orig(logical(sparse(C,1:size(orig,2),1))));
function A=hminired(A)
[m,n]=size(A);
colMin=min(A);
A=A-colMin(ones(n,1),:);
rowMin=min(A')';
A=A-rowMin(:,ones(1,n));
```

```
 [i,j]=find(A==0);
A(1,n+1)=0;
for k=1:n
      cols=j(k==i)';
      A(k,[n+1 cols])=[-cols 0];
end
function [A,C,U]=hminiass(A)
[n,np1]=size(A);
C=zeros(1,n);
U=zeros(1,n+1);
LZ=zeros(1,n);
NZ=zeros(1,n);
for i=1:n
    lj=n+1;
    j=-A(i,lj);
    while (C(j)~=0)
        lj=j;
        j=-A(i,lj);
        if (j==0)
            break;
        end
    end

    if (j~=0)
        C(j)=i;
        A(i,lj)=A(i,j);
        NZ(i)=-A(i,j);
        LZ(i)=lj;
        A(i,j)=0;
    else
        lj=n+1;
        j=-A(i,lj);
        while (j~=0)
            r=C(j);
            lm=LZ(r);
            m=NZ(r);
            while (m~=0)
                if (C(m)==0)
                    break;
                end
                lm=m;
                m=-A(r,lm);
            end

            if (m==0)
                lj=j;
                j=-A(i,lj);
            else
```

```
                    A(r,lm)=-j;
                    A(r,j)=A(r,m);
                    NZ(r)=-A(r,m);
                    LZ(r)=j;
                    A(r,m)=0;
                    C(m)=r;
                    A(i,lj)=A(i,j);
                    NZ(i)=-A(i,j);
                    LZ(i)=lj;
                    A(i,j)=0;
                    C(j)=i;
                    break;
                end
            end
        end
    end
r=zeros(1,n);
rows=C(C~=0);
r(rows)=rows;
empty=find(r==0);
U=zeros(1,n+1);
U([n+1 empty])=[empty 0];
function [A,C,U]=hmflip(A,C,LC,LR,U,l,r)
n=size(A,1);
while (1)
    C(l)=r;
    m=find(A(r,:)==-l);
    A(r,m)=A(r,l);
    A(r,l)=0;
    if (LR(r)<0)
        ...remove row from unassigned row list and return.
        U(n+1)=U(r);
        U(r)=0;
        return;
    else
        l=LR(r);
        A(r,l)=A(r,n+1);
        A(r,n+1)=-l;
        r=LC(l);
    end
end
function [A,CH,RH]=hmreduce(A,CH,RH,LC,LR,SLC,SLR)
n=size(A,1);
coveredRows=LR==0;
coveredCols=LC~=0;
r=find(~coveredRows);
c=find(~coveredCols);.
m=min(min(A(r,c)));
```

```
A(r,c)=A(r,c)-m;
for j=c
	for i=SLR
		if (A(i,j)==0)
			if (RH(i)==0)
				RH(i)=RH(n+1);
				RH(n+1)=i;
				CH(i)=j;
			end
			row=A(i,:);
			colsInList=-row(row<0);
			if (length(colsInList)==0)
				l=n+1;
			else
				l=colsInList(row(colsInList)==0);
			end
			A(i,l)=-j;
		end
	end
end
r=find(coveredRows);
c=find(coveredCols);
[i,j]=find(A(r,c)<=0);
i=r(i);
j=c(j);
for k=1:length(i)
	lj=find(A(i(k),:)==-j(k));
	A(i(k),lj)=A(i(k),j(k));
	A(i(k),j(k))=0;
end
A(r,c)=A(r,c)+m;
function Xp = DataProjection(X,r,type)
if r == 0
Xp = X;
else
	if (nargin < 3)
		type = 'NormalProj';
	end
	D = size(X,1);
	if ( strcmp(type , 'PCA') )
		[U,S,V] = svd(X',0);
		Xp = U(:,1:r)';
	elseif ( strcmp(type , 'NormalProj') )
		np = normrnd(0,1/sqrt(r),r*D,1);
		PrN = reshape(np,r,D);
		Xp = PrN * X;
	elseif( strcmp(type , 'BernoulliProj') )
		bp = rand(r*D,1);
```

```
            Bp = 1/sqrt(r) .* (bp >= .5) - 1/sqrt(r) .* (bp < .5);
            PrB = reshape(Bp,r,D);
            Xp = PrB * X;
        end
end
function CKSym = BuildAdjacency(CMat,K)
N = size(CMat,1);
CAbs = abs(CMat);
for i = 1:N
        c = CAbs(:,i);
        [PSrt,PInd] = sort(c,'descend');
        CAbs(:,i) = CAbs(:,i) ./ abs( c(PInd(1)) );
end
CSym = CAbs + CAbs';
if (K ~= 0)
        [Srt,Ind] = sort( CSym,1,'descend' );
        CK = zeros(N,N);
        for i = 1:N
            for j = 1:K
                CK( Ind(j,i),i ) = CSym( Ind(j,i),i ) ./ CSym( Ind(1,i),i );
            end
        end
        CKSym = CK + CK';
else
        CKSym = CSym;
End
```

4、交替算法

```
clear;
clc;
x0=ones(20,1);y0=ones(20,1);z0=ones(20,1);
M=20*eye(20,20)-ones(20,20);
c=1:20;
q=c';
%初始化  A=B'*B+C+D,
n=20;
B=rand(n)-0.5;
C=zeros(n,n);
D=zeros(n,n);
for i=1:n
        for j=i+1:n
            C(i,j)=rand(1)-0.5;
            C(j,i)=-C(i,j);
        end
        D(i,i)=0.3*rand(1);
        b(i,1)=10*rand(1);
end
A=B'*B+C+D;
%初始化变量
```

```
epsilon=10^-7;gama=1.6;k=-1;eu=1;beta=1;
aszb=1;dz=1;%判断 beta 修改的两个量
x=x0;y=y0;z=z0;
%Step1:判断是否继迭代
while(eu>epsilon&k<=1500)
    z1=z;
%Step2:修正 beta(k+1)
k=k+1;
if aszb<0.1*dz
    beta=beta*0.5;
else if aszb>10*dz
    beta=beta*2;
    else
        beta=beta;
    end
end
%Step3:求 x(k+1),求解变分不等式子问题，利用外梯度算法
x=x-(M+A'*A)*x+q-A'*(y+b);
zg=y-beta*(A*x-b);
if norm(zg)>1
    zg0=zg/norm(zg);
else
    zg0=zg;
end
z=1/beta*(zg0-zg);
%Step5:求 y(k+1)
y=y-gama*beta*(A*x-z-b);
%Step6:计算修正 beta 判断条件的两个变量
axzb=norm(A*x-z-b);dz=norm(z-z1);
eu=axzb^2+dz^2;
 if mod(k,100)==0
        fprintf('km=%4d    epsm=%9.3e      \n',k,eu);
    end
end
fprintf('km=%4d    epsm=%9.3e      \n',k,eu);
```