数据的降维方法

为什么要降维?

维数大，信息量太大，大并不意味着好

假如有一张超大、超清人脸（10m x 10m）要我们识别

为什么要降维?

维数灾难：输入给机器的信息并不是越多越好

- 特征之间的相关性
- 数学上有一些复杂的解释

维数灾难浅述

随着维数增大，会出现一些反直觉的现象

举例：单位正方体，体积为1，其内部最远两点的距离为多少?

- 1维，=线段，最远长度为1
- 2维，=正方形，最远长度为$\sqrt{2}$
- 3维，=立方体，最远长度为$\sqrt{3}$
- ……
- n维，超正方体，最远长度为$\sqrt{n}$

反直觉：如果n接近无穷大，最远长度也接近无穷大，

- 小小的体积内，最远的距离居然有无穷大!
- n维空间的小立方体内，你从A地给B地送信，你可能永远也送不到……
- n维空间的数据，数值很接近，但是其距离可能会达到无穷大

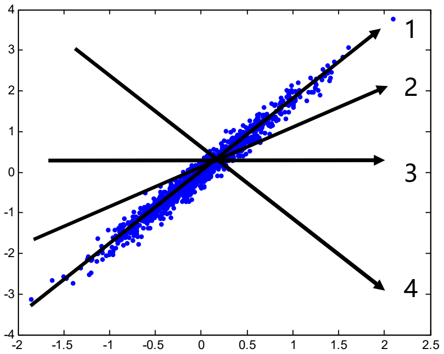
维度灾难导致两个点之间的距离变大

➢等价于让数据的分布变得**稀疏**，使得统计学习过程的鲁棒性变差

➢让模型的自由度变大，过拟合的风险急剧提高

降维的基本准则-信息量最大保持

1. 保留信息量较大的方向，去掉信息量较小的方向
   - 关键词：方向和信息量



2维降1维，保留哪个方向，
其后保留的信息量最大？
1、2、3或者4？

信息量最大的方向=数据最分散的方向

如何用数学语言描述，然后用计算机实现？

数据分散的程度 ➔ 方差、协方差

方差越大越分散！

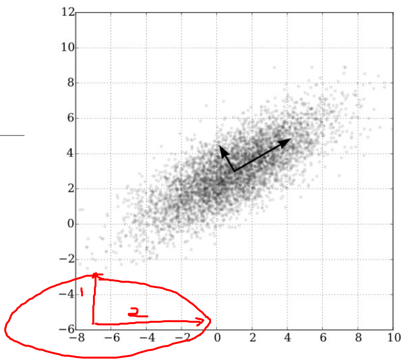
一元变量：方差，描述变量分散程度

$$\frac{\sum_{k=1}^{n}(x_k-\bar{x})(x_k-\bar{x})}{n}$$

多元变量：协方差，描述两个变量共同变化

$$Cov(i,j)=\frac{\sum_{k=1}^{n}(x_{ki}-\overline{x_i})(x_{kj}-\overline{x_j})}{n}$$

$$=Z'*Z\text{，其中}Z=X-\bar{X}$$



如何通过协方差矩阵确定投影方向

协方差矩阵的特征值和特征向量

特征值越大，描述数据越分散

特征向量的方向，描述数据分散的方向

特征值与特征向量

特征值是线性代数中的一个重要概念。在数学、物理学、化学、计算机等领域有着广泛的应用。设 A 是n阶方阵，如果存在数λ和非零n维列向量 x，使得 Ax=λx 成立，则称 λ 是A的一个特征值（characteristic value)或本征值（eigenvalue)。非零n维列向量x称为矩阵A的属于（对应于）特征值λ的特征向量或本征向量，简称A的特征向量或A的本征向量。

9

PCA：“一点数学”

假如我们有原始数据$X = \{x_1, x_2, \ldots, x_n\}$

减均值后得到$Z = \{z_1, z_2, \ldots, z_n\}$

- 均值$\bar{x} = \frac{1}{n}\sum x_i$, $z_i = x_i - \bar{x}$

找到一个方向u_1，使所有z_i在u_1方向上投影长度（方差）最大

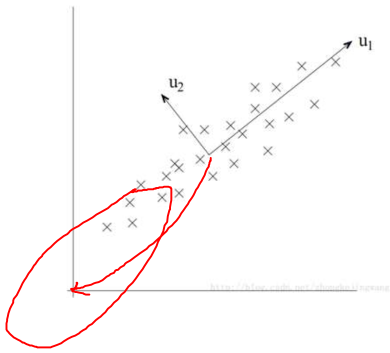
即最大化：$\frac{1}{\mathrm{n}}\sum_{i=1}^{n}(z_i^T u_1)^2$

展开：$\frac{1}{\mathrm{n}}\sum_{i=1}^{n}(z_i^T u_1)^T(z_i^T u_1) = u_1^T \frac{1}{n}\sum_{i=1}^{n} z_i z_i^T\, u_1$

还原成x：$\frac{1}{\mathrm{n}}\sum_{i=1}^{n} z_i z_i^T = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x_i} - \boldsymbol{\bar{x}})(\boldsymbol{x_i} - \boldsymbol{\bar{x}})^T$

- 最后得到了协方差矩阵Cov

目标函数：最大化$u_1^T \mathrm{Cov}\, u_1$



10

PCA： “一点数学”

max $u_1^T Cov\, u_1$ 满足u_1为单位向量，即$u_1^T u_1 = 1$

拉格朗日乘子法：

$L(u_1, \lambda) = u_1^T Cov\, u_1 + \lambda(1 - u_1^T u_1)$

对u_1求导，=0为极大值点

$\frac{\partial L}{\partial u_1} = 2Cov\, u_1 - 2\lambda u_1 = 0 \rightarrow Cov\, u_1 = \lambda u_1$

- Cov的特征值和特征向量

将等式带入目标函数

max $u_1^T Cov\, u_1 = \lambda u_1^T u_1 = \lambda$

所以特征值λ越大，数据就越分散

- **如果要计算k个方向，则保留前k大的特征值对应的特征向量**

算法：使用PCA实现降维

1、读取样本矩阵A（m行，n列，m为样本数目，n为特征数目）

- 计算均值mA（注意：mA为1行n列）

2、计算样本的协方差矩阵C（注意：协方差矩阵是n×n的方阵）

3、计算协方差矩阵的前k个特征向量V（V为n行×k列）

- matlab中的eigs函数（[V, D] = eigs(C, k)， D为特征值，k×k)。

4、降维后的数据：pcaA = (A-mA)*V（pcaA为m行k列）

基于Matlab的PCA计算

平面变成线，二维变一维

$OrF\_1 = [1 \quad 2]$ $OrF\_2 = [3 \quad 4]$ $OrF\_3 = [5 \quad 6]$ 3个点

$orF = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$ 3个点放一起，1行是1个点。这是3个2维的行向量。

$orF' = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$ 3个点放一起，1列是1个点。这是3个2维的列向量。

平面变成线，二维变一维

$$orF = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

3个点放一起，1行是1个点。这是3个2维的行向量。

$$eigF = \begin{bmatrix} -0.7 \\ -0.7 \end{bmatrix}$$

eigF就是orF对应协方差矩阵的特征向量。

$$pcaF = (orF - meanF) * eigF = \begin{bmatrix} -2 & -2 \\ 0 & 0 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} -0.7 \\ -0.7 \end{bmatrix} = \begin{bmatrix} 2.8 \\ 0 \\ -2.8 \end{bmatrix}$$

降成1维后

线变平面，一维还原为二维

$$reF = pcaF * eigF' + meanF = \begin{bmatrix} 2.8 \\ 0 \\ -2.8 \end{bmatrix} * [-0.7 \quad -0.7] + \begin{bmatrix} 3 & 4 \\ 3 & 4 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} -1.96 & -1.96 \\ 0 & 0 \\ 1.96 & 1.96 \end{bmatrix} + \begin{bmatrix} 3 & 4 \\ 3 & 4 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1.04 & 2.04 \\ 3 & 4 \\ 4.96 & 5.96 \end{bmatrix}$$

平面变成线，二维变一维

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

3个点放一起，1行是1个点。这是3个2维的行向量。

$$V = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

这是一个“任意的”向量。

$$D = (A - meanA) * V = \begin{bmatrix} -2 & -2 \\ 0 & 0 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} -8 \\ 0 \\ 8 \end{bmatrix}$$

3个降维后的点。

线变平面，一维还原为二维

$$New_A = D * V' + meanA = \begin{bmatrix} -8 \\ 0 \\ 8 \end{bmatrix} * \begin{bmatrix} 1 & 3 \end{bmatrix} + \begin{bmatrix} 3 & 4 \\ 3 & 4 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} -8 & -24 \\ 0 & 0 \\ 8 & 24 \end{bmatrix} + \begin{bmatrix} 3 & 4 \\ 3 & 4 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} -5 & -20 \\ 3 & 4 \\ 11 & 28 \end{bmatrix}$$

图片变换

$A1 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $A2 = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ 两张图片，每张图片2乘2个像素

B1 =[1 2 3 4] B2=[5 6 7 8] 图片转置

$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$ 两张图片放一起，一行是一张图片。这是2个4维的行向量。

作业：在matlab中实现上述计算

调用C = cov(f_matrix)，计算f_matrix的协方差矩阵

调用[V, D] = eigs(C, k)，计算C的前k个特征值对应的特征向量

PCA求解的加速策略

一行为一个数据　　Z=A-avg(A)

样本矩阵A是m×n矩阵，协方差矩阵$C = Z' \times Z$，其中C为n×n矩阵
- 如果n>>m，比如n=10000，m=100，那么C是一个很大的方阵，极大影响计算特征向量的效率！有没有更加高效的算法？

m很小，能否先算出m×m方阵的特征向量，再转成C的特征向量？可以
- 令$S = Z \times Z'$是m×m的方阵　　S(i,j)为数据i和j的相似度值
- $Sx = \lambda x \Rightarrow Z'Sx = \lambda Z'x \Rightarrow Z'ZZ'x = \lambda Z'x \Rightarrow C(Z'x) = \boxed{\lambda}(Z'x)$
- 算出S的前k个特征向量V，然后$Z'V$就是C的特征向量
  - 注意要把$Z'V$归一化成单位向量

$y = Z'x$

$Cy = \lambda y$

PCA的特点小结

PCA算法的主要优点有：

1）**追求数据的最佳重建效果**，仅仅需要以方差衡量信息量，不受数据集以外的因素影响。

2）各主成分之间正交，可消除原始数据成分间的相互影响的因素。

3）计算方法简单，主要运算是特征值分解，易于实现。

4）仅适用于具有高斯分布的数据

PCA算法的主要缺点有：

1）主成分各个特征维度的含义具有一定的模糊性，不如原始样本特征的解释性强。

2）方差小的非主成分也可能含有对样本差异的重要信息，因降维丢弃可能对后续数据处理有影响。

线性判别分析-LDA

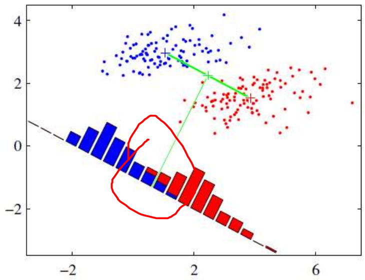
LDA是一种有监督的降维技术，数据集的每个样本是有类别输出的。

和PCA不同，PCA是不考虑样本类别输出的无监督降维技术。

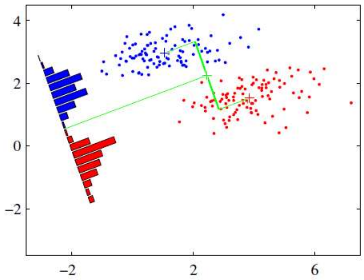
LDA的思想：**投影后类内方差最小，类间方差最大。**

将数据在低维度上进行投影，投影后希望每一种类别数据的投影点尽可能的接近，而不同类别的数据的类别中心之间的距离尽可能的大。

直观效果



PCA投影



LDA投影

瑞利商（Rayleigh quotient）与广义瑞利商（genralized Rayleigh quotient）

瑞利商定义：

$$R(A,x)=\frac{x^HAx}{x^Hx}$$

瑞利商R(A,x)的最大值等于矩阵A最大的特征值，而最小值等于矩阵A的最小的特征值，即满足：

$$\lambda_{min}\le\frac{x^HAx}{x^Hx}\le\lambda_{max}$$

当向量x是标准正交基时，即满足$x^Hx=1$时，瑞利商退化为：$R(A,x)=x^HAx$，这个形式在谱聚类和PCA中都有出现

广义瑞利商定义：

$$R(A,x)=\frac{x^HAx}{x^HBx}$$

A,B为n×n的Hermitan矩阵（推广实对称阵）。B为正定矩阵

令$x'=B^{-1/2}x$，则：

$$R(A,B,x')=\frac{x'^HB^{-1/2}AB^{-1/2}x'}{x'^Hx'}$$

R(A,B,x)的最大值为矩阵$B^{-1/2}AB^{-1/2}$的最大特征值，或者说矩阵$B^{-1}A$的最大特征值，而最小值为矩阵$B^{-1}A$的最小特征值

二类LDA

假设我们的数据集$D=\{(x_1,y_1),(x_2,y_2),...,(x_m,y_m)\}$,其中任意样本$x_i$为n维向量，$y_i\in\{0,1\}$。

我们定义$N_j(j=0,1)$为第j类样本的个数，$X_j(j=0,1)$为第j类样本的集合，而$\mu_j(j=0,1)$为第j类样本的均值向量，定义$\Sigma_j(j=0,1)$为第j类样本的协方差矩阵。

μ_j的表达式为：

$$\mu_j=1/N_j\sum_{x\in X_j}x, \qquad (j=0,1)$$

Σ_j的表达式为：

$$\Sigma_j=\sum_{x\in X_j}(x-\mu_j)(x-\mu_j)^T \qquad (j=0,1)$$

二类LDA

LDA需要

- 让不同类别的数据的类别中心之间的距离尽可能的大，即最大化$||w^T\mu_0-w^T\mu_1||^2$
- 希望同一种类别数据的投影点尽可能的接近，也就是要同类样本投影点的协方差$w^T\Sigma_0w$和$w^T\Sigma_1w$尽可能的小，即最小化$w^T\Sigma_0w+w^T\Sigma_1w$。

优化目标为：

$$\underbrace{arg\ max}_{w}\ J(w)=\frac{||w^T\mu_0-w^T\mu_1||_2^2}{w^T\Sigma_0w+w^T\Sigma_1w}=\frac{w^T(\mu_0-\mu_1)(\mu_0-\mu_1)^Tw}{w^T(\Sigma_0+\Sigma_1)w}$$

我们一般定义类内散度矩阵S_w为：

$$S_w=\Sigma_0+\Sigma_1=\sum_{x\in X_0}(x-\mu_0)(x-\mu_0)^T+\sum_{x\in X_1}(x-\mu_1)(x-\mu_1)^T$$

同时定义类间散度矩阵S_b为：

$$S_b=(\mu_0-\mu_1)(\mu_0-\mu_1)^T$$

这样我们的优化目标重写为：

$$\underbrace{arg\ max}_{w}\ J(w)=\frac{w^TS_bw}{w^TS_ww}$$

多类LDA

假设我们的数据集D={$(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)$},其中任意样本$x_i$为n维向量，$y_i\in\{1,\ldots,K\}$。定义$N_j(j=1,\ldots,K)$为第j类样本的个数，$X_j(j=1,\ldots,K)$为第j类样本的集合，而$\mu_j(j=1,\ldots,K)$为第j类样本的均值向量，定义$\Sigma_j(j=0,1)$为第j类样本的协方差矩阵。

μ_j的表达式为：

$$\mu_j=1/N_j\sum_{x\in Xj}x,\qquad (j=1,\ldots,K)$$

Σ_j的表达式为：

$$\Sigma_j=\sum_{x\in Xj}(x-\mu_j)(x-\mu_j)^T\qquad (j=1,\ldots,K)$$

在多类向低维投影中，投影到的低维空间就不是一条直线，而是一个超平面了。假设我们投影到的低维空间的维度为d，对应的基向量为$(w_1,w_2,\ldots w_d)$，基向量组成的矩阵为W, 它是一个n×d的矩阵

9 → 8

多类LDA

其中$S_b=\sum_{j=1}^{k}N_j(\mu_j-\mu)(\mu_j-\mu)^T$,$\mu$为所有样本均值向量。

$S_w=\sum_{j=1}^{k}S_{wj}=\sum_{j=1}^{k}\sum_{x\in X_j}(x-\mu_j)(x-\mu_j)^T$

常见的一个LDA多类优化目标函数定义为:

$$\underbrace{arg\ max}_{W}\ J(W)=\frac{\prod_{diag}W^TS_bW}{\prod_{diag}W^TS_wW}$$

其中$\prod_{diag}A$为A的主对角线元素的乘积，W为$n\times d$的矩阵。

$J(W)$的优化过程可以转化为:

$$J(W)=\frac{\prod_{i=1}^{d}w_i^TS_bw_i}{\prod_{i=1}^{d}w_i^TS_ww_i}=\prod_{i=1}^{d}\frac{w_i^TS_bw_i}{w_i^TS_ww_i}$$

问题：对于C类LDA而言，特征向量最多有多少维?

LDA算法流程

输入：数据集$D=\{(x_1,y_1),(x_2,y_2),\ldots,((x_m,y_m))\}$,其中任意样本$x_i$为n维向量，$y_i\in\{C_1,C_2,\ldots,C_k\}$，降维到的维度d。

输出：降维后的样本集D'

1) 计算类内散度矩阵S_w

2) 计算类间散度矩阵S_b

3) 计算矩阵$\boxed{S_w^{-1}S_b}$

4) 计算$S_w^{-1}S_b$的最大的d个特征值和对应的d个特征向量$(w_1,w_2,\ldots w_d)$,得到投影矩阵[Math Processing Error]

5) 对样本集中的每一个样本特征x_i,转化为新的样本$z_i=W^Tx_i$

6) 得到输出样本集$D'=\{(z_1,y_1),(z_2,y_2),\ldots,((z_m,y_m))\}$

与PCA形式相同

问题：如何核化LDA?

LDA算法特性

LDA算法的主要优点有：

1）在降维过程中可以使用类别的先验知识经验，而PCA无法使用类别先验知识。

2）LDA在样本分类信息依赖均值而不是方差的时候，比PCA之类的算法较优。

LDA算法的主要缺点有：

1）LDA不适合对非高斯分布样本进行降维，PCA也有这个问题。

2）LDA降维最多降到类别数k-1的维数，如果我们降维的维度大于k-1，则不能使用LDA。当然目前有一些LDA的进化版算法（kernel DA）可以绕过这个问题。

3）LDA在样本分类信息依赖方差而不是均值的时候，降维效果不好。

4）LDA可能过度拟合数据。

PCA vs. LDA

相同点：

1）两者均可以对数据进行降维。

2）两者在降维时均使用了矩阵特征分解的思想。

3）两者都假设数据符合高斯分布。

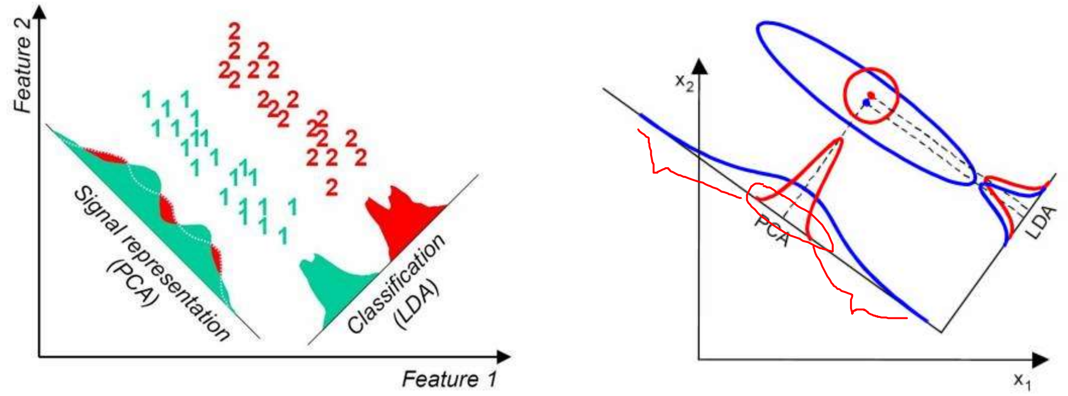
不同点：

1）LDA是有监督的降维方法，而PCA是无监督的降维方法

2）LDA降维最多降到类别数k-1的维数，而PCA没有这个限制。

3）LDA除了可以用于降维，还可以用于分类。

4）LDA选择分类性能最好的投影方向，而PCA选择样本点投影具有最大方差的方向。



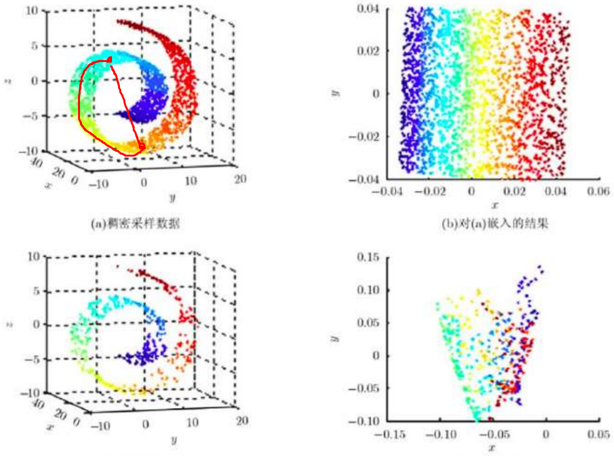
流形降维

传统的机器学习方法中，数据点和数据点之间的距离和映射函数f都是定义在欧式空间中的。

然而在实际情况中，这些数据点可能不是分布在欧式空间中的，因此传统欧式空间的度量难以用于真实世界的非线性数据，从而需要对数据的分布引入新的假设。

流形(Manifold)是局部具有欧式空间性质的空间，包括各种纬度的曲线曲面，例如球体、弯曲的平面等。流形的局部和欧式空间是同构的。

流形



(a)稠密采样数据 (b)对(a)嵌入的结果

(c)稀疏采样数据 (d)对(c)嵌入的结果

流形学习

流形学习假设所处理的数据点分布在嵌入于外维欧式空间的一个潜在的流形体上，或者说这些数据点可以构成这样一个潜在的流形体。

假设数据是均匀采样于一个高维欧氏空间中的低维流形，流形学习就是从高维采样数据中恢复低维流形结构，即找到高维空间中的低维流形，并求出相应的嵌入映射，以实现维数约简或者数据可视化。它是从观测到的现象中去寻找事物的本质，找到产生数据的内在规律。

设 $Y \subset R^d$是一个低维流形，$f: Y \rightarrow R^D$是一个光滑嵌入，其中 $D>d$. 数据集 $\{y_i\}$是随机生成的，且经过 f 映射为观察空间的数据 $\{x_i = f(y_i)\}$. 流形学习就是在给定观察样本集$\{x_i\}$的条件下重构 f 和 $\{y_i\}$.

V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction . ***Neural Information Processing Systems 15 (NIPS'2002)*****, pp. 705-712, 2003.**

局部线性嵌入（LLE）

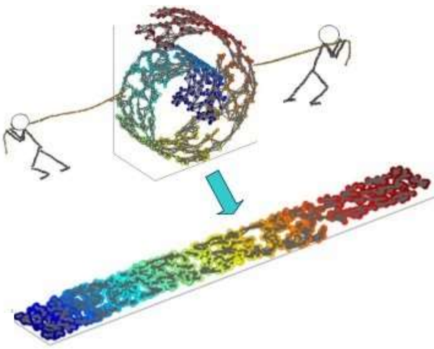
LLE算法流程

算法的主要步骤分为三步：

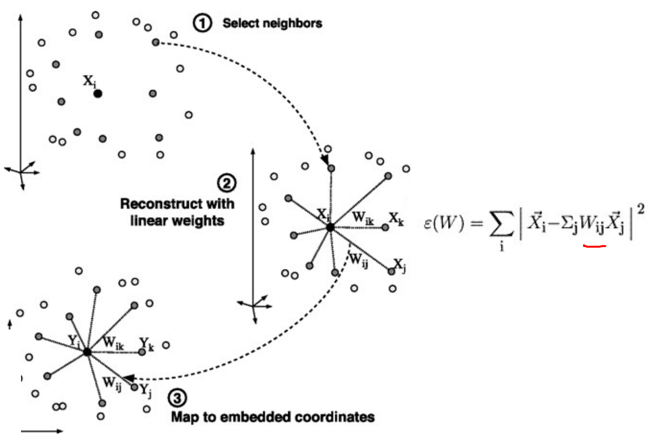
（1）寻找每个样本点x_i的k个近邻点集合X_i；

（2）由每个样本点的近邻点计算出该样本点的局部重建权值向量w_i； X=WX

（3）由所有样本点的局部重建权值向量建立映射后的坐标Y的关系; Y=WY



等价于M矩阵的特征值分解问题：$M=(I-W)^T(I-W)$,



$$\Phi(Y)=\sum_i\left|\vec{Y}_i-\Sigma_j W_{ij}\vec{Y}_j\right|^2$$

方法简称	所保持的几何属性	全局/局部关系	计算复杂度
ISOMAP	点对测地距离	全局	非常高
LLE	局部线性重构关系	局部	低
LE	局部邻域相似度	局部	低
HLLE	局部等距性	局部	高
LTSA	局部坐标表示	全局+局部	低
MVU	局部距离	全局+局部	非常高
Logmap	测地距离与方向	局部	非常低
Diffusion Maps	diffusion距离	全局	中等

作业：实现PCA降维模块

```
function [pcaA, V] = myPCA(A, k, mA)
%myPCA，主成份分析
%输入：A-样本矩阵，每行是一个样本，列是样本的维数
%       k-降至k维
%       mA-给定A的均值，行向量
%输出：pcaA-降维后的k维样本特征向量组成的矩阵，即主成分
%       V-前k个特征向量
%       在求出协方差矩阵C后，[V,D]=eigs(C,k)用于求解前k个特征向量和特征值
%       其中V为所求的特征向量
```

作业：降维后升维

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

调用myPCA将它降成k(k=1)维

- 检查特征向量是否为[-0.7, -0.7]或者[0.7, 0.7]

将1维还原成2维

作业：展示特征脸，eigenface.m

读取：$orF = ReadFace(npersons, flag = 0)$
均值：$meanF = mean(orF)$

降维：$pcaF, eigV = myPCA(orF, k, meanF)$

还原：$eigF = pcaF(i,:) * eigV' + meanF$

Matlab，向量转矩阵，然后显示图片：
eig_img = reshape(eigF, [imgrow, imgcol]);
imshow(uint8(eig_img))

原始图像向量
原始脸： orF

降维后还原的图片
特征脸： $eigF$

作业：展示特征脸，eigenface.m

降维：$pcaF = (orF - meanF) * eigV$

还原：$eigF = pcaF(1,:) * eigV' + meanF$

Matlab，向量转矩阵，然后显示图片：
eig_img = reshape(eigF, [imgrow, imgcol]);
imshow(uint8(eig_img))

原始图像向量
原始脸：orF

降维后还原的图片 (k=40)
特征脸：$eigF$