来源: https://blog.csdn.net/u011826404/article/details/72123031

上篇主要介绍了几种常用的聚类算法，首先从距离度量与性能评估出发，列举了常见的距离计算公式与聚类评价指标，接着分别讨论了K-Means、LVQ、高斯混合聚类、密度聚类以及层次聚类算法。K-Means与LVQ都试图以类簇中心作为原型指导聚类，其中K-Means通过EM算法不断迭代直至收敛，LVQ使用真实类标辅助聚类；高斯混合聚类采用高斯分布来描述类簇原型；密度聚类则是将一个核心对象所有密度可达的样本形成类簇，直到所有核心对象都遍历完；最后层次聚类是一种自底向上的树形聚类方法，不断合并最相近的两个小类簇。本篇将讨论机器学习常用的方法–降维与度量学习。

11、降维与度量学习

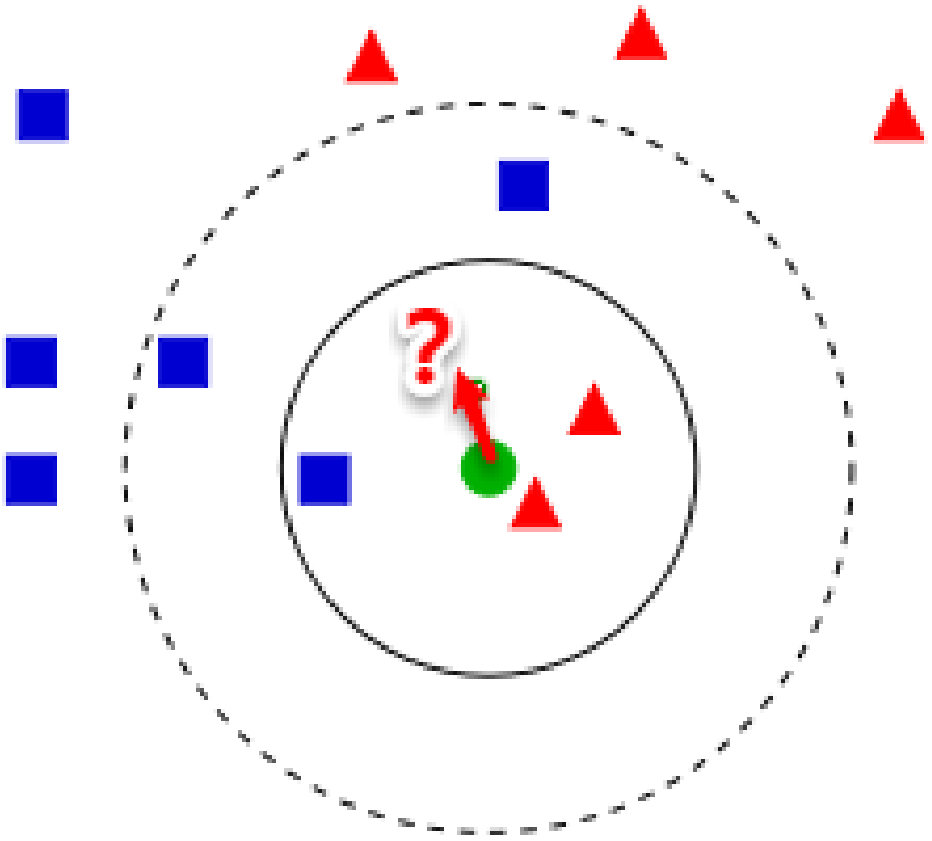
样本的特征数称为**维数**（dimensionality），当维数非常大时，也就是现在所说的“**维数灾难**”，具体表现在：在高维情形下，**数据样本将变得十分稀疏**，因为此时要满足训练样本为“**密采样**”的总体样本数目是一个触不可及的天文数字，谓可远观而不可亵玩焉...**训练样本的稀疏使得其代表总体分布的能力大大减弱，从而消减了学习器的泛化能力**；同时当维数很高时，**计算距离也变得十分复杂**，甚至连计算内积都不再容易，这也是为什么支持向量机（SVM）使用核函数**“低维计算，高维表现”**的原因。

缓解维数灾难的一个重要途径就是**降维，即通过某种数学变换将原始高维空间转变到一个低维的子空间**。在这个子空间中，样本的密度将大幅提高，同时距离计算也变得容易。这时也许会有疑问，这样降维之后不是会丢失原始数据的一部分信息吗？这是因为在很多实际的问题中，虽然训练数据是高维的，但是与学习任务相关也许仅仅是其中的一个低维子空间，也称为一个**低维嵌入**，例如：数据属性中存在噪声属性、相似属性或冗余属性等，**对高维数据进行降维能在一定程度上达到提炼低维优质属性或降噪的效果。**

11.1 K近邻学习

k近邻算法简称**kNN（k-Nearest Neighbor）**，是一种经典的监督学习方法，同时也实力担当入选数据挖掘十大算法。其工作机制十分简单粗暴：给定某个测试样本，kNN基于某种**距离度量**在训练集中找出与其距离最近的k个带有真实标

记的训练样本，然后给基于这k个邻居的真实标记来进行预测，类似于前面集成学习中所讲到的基学习器结合策略：分类任务采用投票法，回归任务则采用平均法。接下来本篇主要就kNN分类进行讨论。



从上图【来自Wiki】中我们可以看到，图中有两种类型的样本，一类是蓝色正方形，另一类是红色三角形。而那个绿色圆形是我们待分类的样本。基于kNN算法的思路，我们很容易得到以下结论：

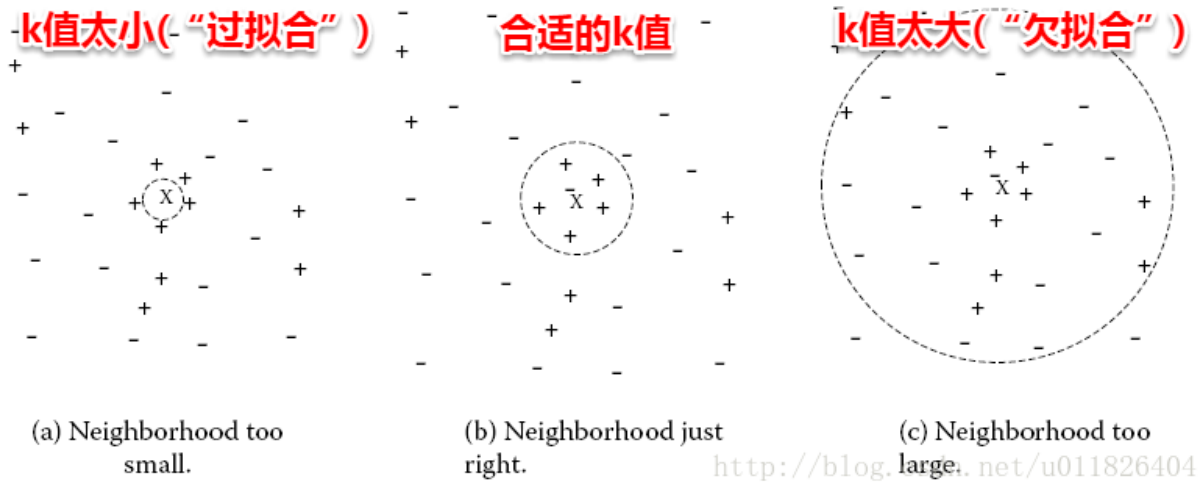
> 如果K=3，那么离绿色点最近的有2个红色三角形和1个蓝色的正方形，这3个点投票，于是绿色的这个待分类点属于红色的三角形。

> 如果K=5，那么离绿色点最近的有2个红色三角形和3个蓝色的正方形，这5个点投票，于是绿色的这个待分类点属于蓝色的正方形。

可以发现：**kNN虽然是一种监督学习方法，但是它却没有显式的训练过程**，而是当有新样本需要预测时，才来计算出最近的k个邻居，因此**kNN是一种典型的**

懒惰学习方法，再来回想一下朴素贝叶斯的流程，训练的过程就是参数估计，因此朴素贝叶斯也可以懒惰式学习，此类技术在**训练阶段开销为零**，待收到测试样本后再进行计算。相应地我们称那些一有训练数据立马开工的算法为“**急切学习**”，可见前面我们学习的大部分算法都归属于急切学习。

很容易看出：**kNN算法的核心在于k值的选取以及距离的度量。**k值选取太小，模型很容易受到噪声数据的干扰，例如：极端地取k=1，若待分类样本正好与一个噪声数据距离最近，就导致了分类错误；若k值太大， 则在更大的邻域内进行投票，此时模型的预测能力大大减弱，例如：极端取k=训练样本数，就相当于模型根本没有学习，所有测试样本的预测结果都是一样的。**一般地我们都通过交叉验证法来选取一个适当的k值。**



(a) Neighborhood too small.　(b) Neighborhood just right.　(c) Neighborhood too large.

对于距离度量，**不同的度量方法得到的k个近邻不尽相同，从而对最终的投票结果产生了影响**，因此选择一个合适的距离度量方法也十分重要。在上一篇聚类算法中，在度量样本相似性时介绍了常用的几种距离计算方法，包括**闵可夫斯基距离，曼哈顿距离，VDM**等。在实际应用中，**kNN的距离度量函数一般根据样本的特性来选择合适的距离度量，同时应对数据进行去量纲/归一化处理来消除大量纲属性的强权政治影响。**

11.2 MDS算法

不管是使用核函数升维还是对数据降维，我们都希望**原始空间样本点之间的距离在新空间中基本保持不变**，这样才不会使得原始空间样本之间的关系及总体分布发生较大的改变。**“多维缩放”（MDS）**正是基于这样的思想，**MDS要求原始空间样本之间的距离在降维后的低维空间中得以保持。**

假定m个样本在原始空间中任意两两样本之间的距离矩阵为D∈R(m*m)，我们的目标便是获得样本在低维空间中的表示Z∈R(d’ *m，d’ < d)，且任意两个样本在低维空间中的欧式距离等于原始空间中的距离，即||zi-zj||=Dist(ij)。因此接下来我们要做的就是根据已有的距离矩阵D来求解出降维后的坐标矩阵Z。

$$令\ \mathbf{B}=\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\in\mathbb{R}^{m\times m},\ 其中\ \mathbf{B}\ 为降维后样本的内积矩阵,\ b_{ij}=\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j$$

高维距离 = 低维欧氏距离

$$\begin{aligned} dist_{ij}^2 &= \|\boldsymbol{z}_i\|^2+\|\boldsymbol{z}_j\|^2-2\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j \\ &= b_{ii}+b_{jj}-2b_{ij} \end{aligned} \quad (1)$$

令降维后的样本坐标矩阵Z被中心化，**中心化是指将每个样本向量减去整个样本集的均值向量，故所有样本向量求和得到一个零向量**。这样易知：矩阵B的每一列以及每一列求和均为0，因为提取公因子后都有一项为所有样本向量的和向量。

$$B=\begin{bmatrix} z_1 \\ \dots \\ z_m \end{bmatrix} * \begin{bmatrix} z_1 & \dots & z_m \end{bmatrix} = \begin{bmatrix} z_1z_1 & z_1z_2 & \dots & z_1z_m \\ z_2z_1 & z_2z_2 & \dots & z_2z_m \\ \dots & \dots & \dots & \dots \\ z_mz_1 & z_mz_2 & \dots & z_mz_m \end{bmatrix}$$

和为零向量

和为零向量

根据上面矩阵B的特征，我们很容易得到等式（2）、（3）以及（4）：

$$\sum_{i=1}^{m} dist_{ij}^2 = \mathrm{tr}(\mathbf{B})+mb_{jj} \quad (2) \quad *1/m$$

$$\sum_{j=1}^{m} dist_{ij}^2 = \mathrm{tr}(\mathbf{B})+mb_{ii} \quad (3) \quad *1/m$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}^2 = 2m\,\mathrm{tr}(\mathbf{B}) \quad (4) \quad *1/(m^2)$$

这时根据(1)–(4)式我们便可以计算出bij，即**bij=(1)-(2)*(1/m)-(3)*(1/m)+(4)*(1/(m^2))**，再逐一地计算每个b(ij)，就得到了降维后低维空间中的内积矩阵B(B=Z’ *Z)，只需对B进行特征值分解便可以得到Z。MDS的算法流程如下图所示：

输入：距离矩阵 $\mathbf{D} \in \mathbb{R}^{m \times m}$，其元素 $dist_{ij}$ 为样本 $\boldsymbol{x}_i$ 到 $\boldsymbol{x}_j$ 的距离；
低维空间维数 d'.

过程：

1: 根据式(10.7)~(10.9)计算 $dist_{i\cdot}^2$, $dist_{\cdot j}^2$, $dist_{\cdot\cdot}^2$;

2: 根据式(10.10)计算矩阵 $\mathbf{B}$; 低维内积矩阵

3: 对矩阵 $\mathbf{B}$ 做特征值分解; 特征值分解求解

4: 取 $\mathbf{\Lambda}$ 为 d' 个最大特征值所构成的对角矩阵，$\tilde{\mathbf{V}}$ 为相应的特征向量矩阵.

输出：矩阵 $\tilde{\mathbf{V}}\tilde{\mathbf{\Lambda}}^{1/2} \in \mathbb{R}^{m \times d'}$，每行是一个样本的低维坐标 并没有得到投影向量

11.3 主成分分析（PCA）

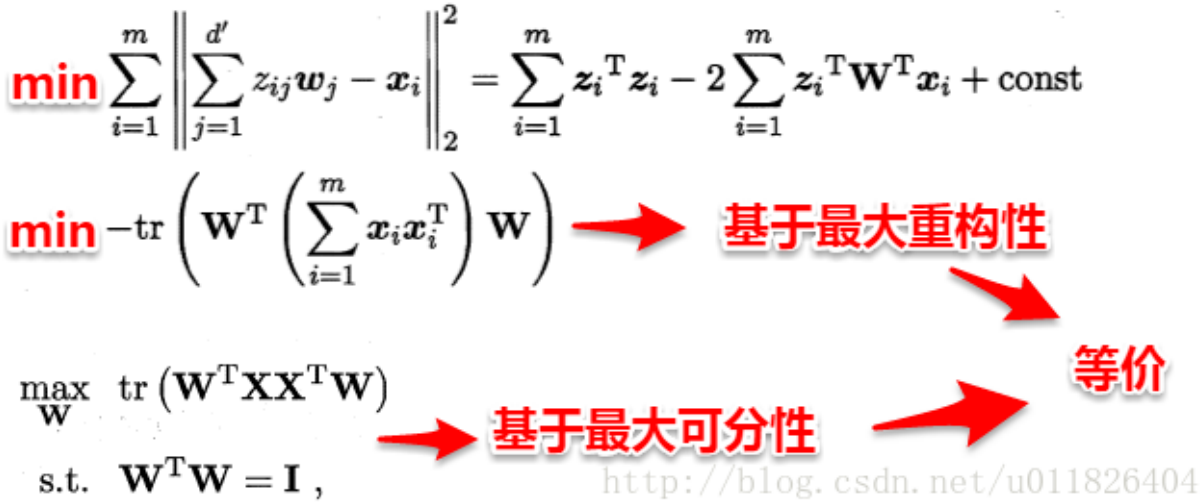
不同于MDS采用距离保持的方法，**主成分分析（PCA）直接通过一个线性变换，将原始空间中的样本投影到新的低维空间中**。简单来理解这一过程便是：**PCA采用一组新的基来表示样本点，其中每一个基向量都是原来基向量的线性组合，通过使用尽可能少的新基向量来表出样本，从而达到降维的目的。**

假设使用d’个新基向量来表示原来样本，实质上是将样本投影到一个由d’个基向量确定的一个**超平面**上（**即舍弃了一些维度**），要用一个超平面对空间中所有高维样本进行恰当的表达，最理想的情形是：**若这些样本点都能在超平面上表出且这些表出在超平面上都能够很好地分散开来**。但是一般使用较原空间低一些维度的超平面来做到这两点十分不容易，因此我们退一步海阔天空，要求这个超平面应具有如下两个性质：

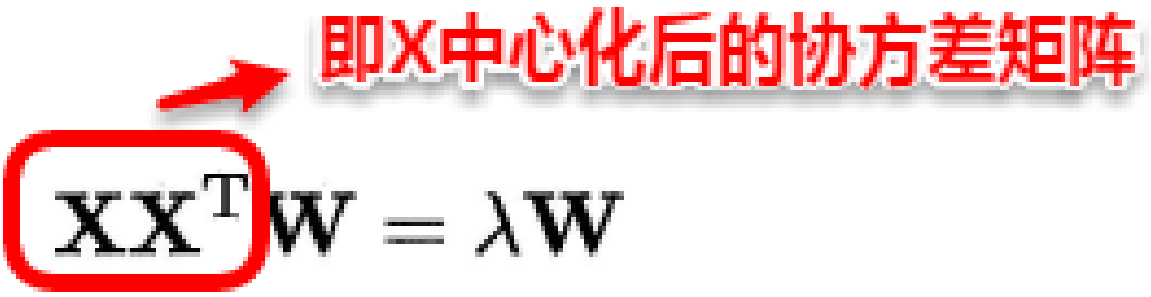
> **最近重构性**：样本点到超平面的距离足够近，即尽可能在超平面附近；

> **最大可分性**：样本点在超平面上的投影尽可能地分散开来，即投影后的坐标具有区分性。

这里十分神奇的是：**最近重构性与最大可分性虽然从不同的出发点来定义优化问题中的目标函数，但最终这两种特性得到了完全相同的优化问题**：

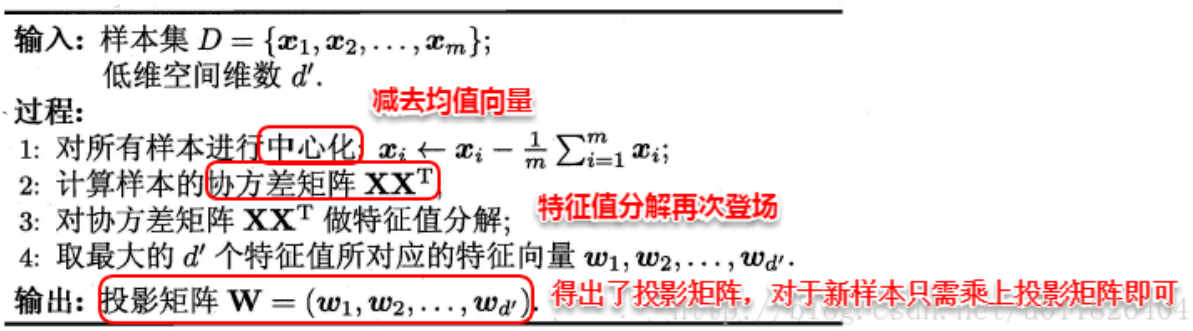


接着使用拉格朗日乘子法求解上面的优化问题，得到：



$$\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{W} = \lambda\mathbf{W}$$

因此只需对协方差矩阵进行特征值分解即可求解出W，PCA算法的整个流程如下图所示：



另一篇博客给出更通俗更详细的理解：[主成分分析解析（基于最大方差理论）]

11.4 核化线性降维

说起机器学习你中有我/我中有你/水乳相融...在这里能够得到很好的体现。正如SVM在处理非线性可分时，通过引入核函数将样本投影到高维特征空间，接着在高维空间再对样本点使用超平面划分。这里也是相同的问题：若我们的样本数据点本身就不是线性分布，那还如何使用一个超平面去近似表出呢？因此也就引

入了核函数，**即先将样本映射到高维空间，再在高维空间中使用线性降维的方法。**下面主要介绍**核化主成分分析（KPCA）**的思想。

若核函数的形式已知，即我们知道如何将低维的坐标变换为高维坐标，这时我们只需先将数据映射到高维特征空间，再在高维空间中运用PCA即可。但是一般情况下，我们并不知道核函数具体的映射规则，例如：Sigmoid、高斯核等，我们只知道如何计算高维空间中的样本内积，这时就引出了KPCA的一个重要创新之处：**即空间中的任一向量，都可以由该空间中的所有样本线性表示。**证明过程也十分简单：

$$\left(\sum_{i=1}^{m} z_i z_i^{\mathrm{T}}\right) \mathbf{W} = \lambda \mathbf{W}$$

zi为样本点在高维特征空间中的坐标向量
W为高维特征空间中的一个新基向量

$$\mathbf{W} = \frac{1}{\lambda}\left(\sum_{i=1}^{m} z_i z_i^{\mathrm{T}}\right)\mathbf{W} = \sum_{i=1}^{m} z_i \frac{z_i^{\mathrm{T}}\mathbf{W}}{\lambda}$$

计算出来是一个常数

$$= \sum_{i=1}^{m} z_i \alpha_i \text{ ，证毕}$$

这样我们便可以将高维特征空间中的投影向量wi使用所有高维样本点线性表出，接着代入PCA的求解问题，得到：

$$\Phi(X)\Phi(X)^T w_i = \lambda_i w_i$$

空间中的任一向量，都可以由该空间中的所有样本线性表示

$$w_i = \sum_{k=1}^{N} \alpha_i \Phi(x_i) = \Phi(X)\alpha$$

$$\Phi(X)\Phi(X)^T \Phi(X)\alpha = \lambda_i \Phi(X)\alpha$$

核函数矩阵

$$\Phi(X)^T \Phi(X)\Phi(X)^T \Phi(X)\alpha = \lambda_i \Phi(X)^T \Phi(X)\alpha$$

$$K^2\alpha = \lambda_i K\alpha$$

$$K\alpha = \lambda_i \alpha$$

特征值分解无处不在~

化简到最后一步，发现结果十分的美妙，只需对核矩阵K进行特征分解，便可以得出投影向量wi对应的系数向量α，因此选取特征值前d’大对应的特征向量便是d’个系数向量。这时对于需要降维的样本点，只需按照以下步骤便可以求出其降维后的坐标。可以看出：KPCA在计算降维后的坐标表示时，需要与所有样本点计算核函数值并求和，因此该算法的计算开销十分大。

$$\hat{x}_{new}$$

$$= w_i^T x_{new}$$ 新样本点在wi维度上的投影坐标 共有d'个投影向量w

$$= (\sum_{i=1}^{N} \Phi(x_i)\alpha_i)^T \Phi(x_{new})$$

$$= (\Phi(X)\alpha)^T \Phi(x_{new})$$

$$= \alpha^T \Phi(X)^T \Phi(x_{new})$$

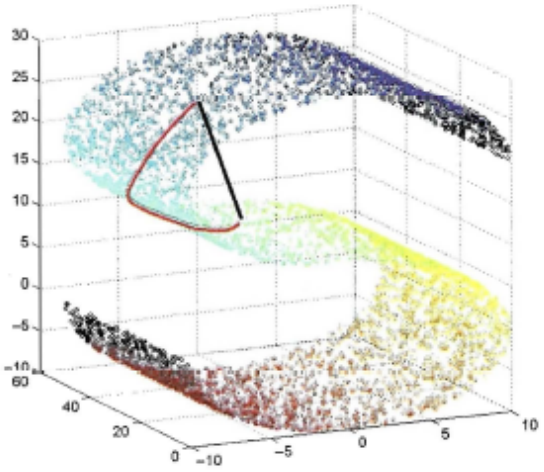
$$= [\alpha_1, ..., \alpha_N][k(x_1, x_{new}), ..., k(x_N, x_{new})]^T$$

11.5 流形学习

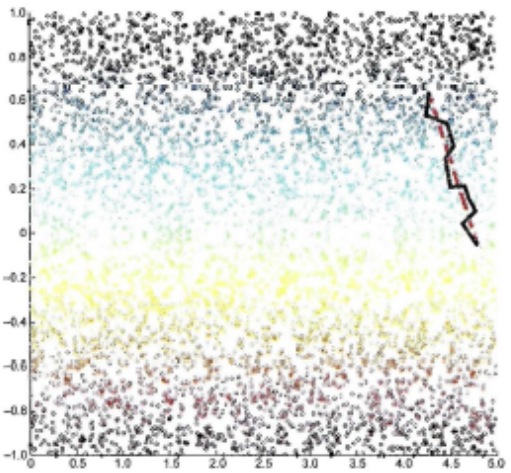
流形学习（manifold learning）是一种借助拓扑流形概念的降维方法，流形是指在局部与欧式空间同胚的空间，即在局部与欧式空间具有相同的性质，能用欧氏距离计算样本之间的距离。这样即使高维空间的分布十分复杂，但是在局部上依然满足欧式空间的性质，基于流形学习的降维正是这种**“邻域保持”**的思想。其中**等度量映射（Isomap）试图在降维前后保持邻域内样本之间的距离，而局部线性嵌入（LLE）则是保持邻域内样本之间的线性关系**，下面将分别对这两种著名的流行学习方法进行介绍。

11.5.1 等度量映射（Isomap）

等度量映射的基本出发点是：高维空间中的直线距离具有误导性，因为有时高维空间中的直线距离在低维空间中是不可达的。**因此利用流形在局部上与欧式空间同胚的性质，可以使用近邻距离来逼近测地线距离**，即对于一个样本点，它与近邻内的样本点之间是可达的，且距离使用欧式距离计算，这样整个样本空间就形成了一张近邻图，高维空间中两个样本之间的距离就转为最短路径问题。可采用著名的**Dijkstra算法**或**Floyd算法**计算最短距离，得到高维空间中任意两点之间的距离后便可以使用MDS算法来其计算低维空间中的坐标。



(a) 测地线距离与高维直线距离



(b) 测地线距离与近邻距离

从MDS算法的描述中我们可以知道：MDS先求出了低维空间的内积矩阵B，接着使用特征值分解计算出了样本在低维空间中的坐标，但是并没有给出通用的投影向量w，因此对于需要降维的新样本无从下手，书中给出的权宜之计是利用已知高/低维坐标的样本作为训练集学习出一个“投影器”，便可以用高维坐标预测出低维坐标。Isomap算法流程如下图：

输入： 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
近邻参数 k;
低维空间维数 d'.

整个样本集形成一张可达图

过程：

1: **for** $i = 1, 2, \ldots, m$ **do**
2: 确定 $\boldsymbol{x}_i$ 的 k 近邻;
3: $\boldsymbol{x}_i$ 与 k 近邻点之间的距离设置为欧氏距离, 与其他点的距离设置为无穷大
4: **end for**
5: 调用最短路径算法计算任意两样本点之间的距离 $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$;
6: 将 $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ 作为 MDS 算法的输入;
7: **return** MDS 算法的输出

输出： 样本集 D 在低维空间的投影 $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m\}$.

对于近邻图的构建，常用的有两种方法：**一种是指定近邻点个数**，像kNN一样选取k个最近的邻居；**另一种是指定邻域半径**，距离小于该阈值的被认为是它的近邻点。但两种方法均会出现下面的问题：

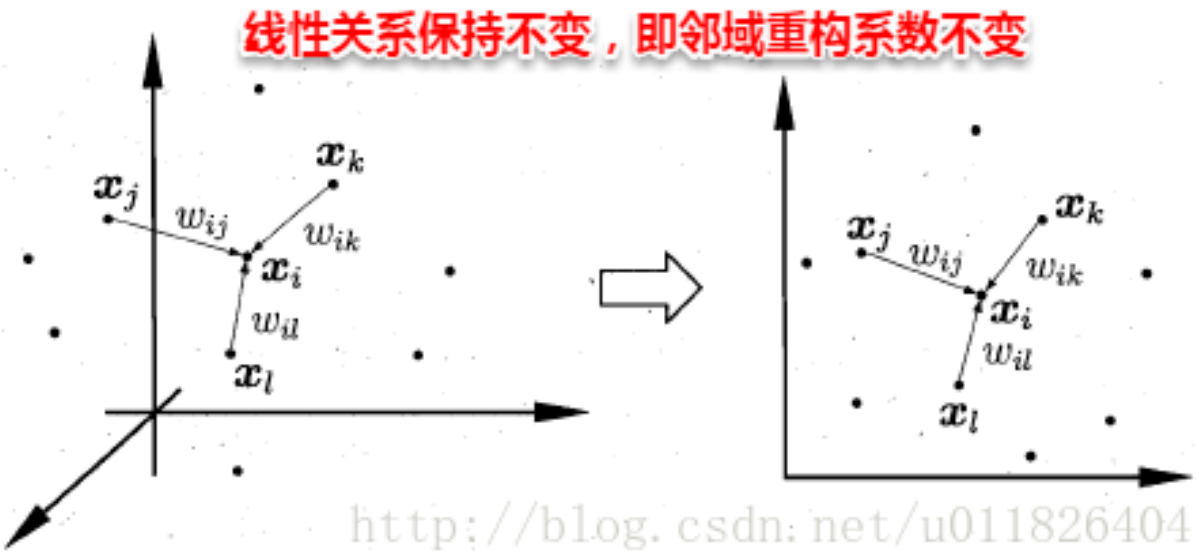
> 若**邻域范围指定过大，则会造成“短路问题”**，即本身距离很远却成了近邻，将距离近的那些样本扼杀在摇篮。

若**邻域范围指定过小，则会造成“断路问题”**，即有些样本点无法可达了，整个世界村被划分为互不可达的小部落。

11.5.2 局部线性嵌入(LLE)

不同于Isomap算法去保持邻域距离，LLE算法试图去保持邻域内的线性关系，假定样本xi的坐标可以通过它的邻域样本线性表出：

$$\boldsymbol{x}_i = w_{ij}\boldsymbol{x}_j + w_{ik}\boldsymbol{x}_k + w_{il}\boldsymbol{x}_l$$



LLE算法分为两步走，**首先第一步根据近邻关系计算出所有样本的邻域重构系数w**：

$$\min_{w_1,w_2,\dots,w_m} \sum_{i=1}^{m} \left\| \boldsymbol{x}_i - \sum_{j\in Q_i} w_{ij}\boldsymbol{x}_j \right\|_2^2$$

$$\text{s.t.} \sum_{j\in Q_i} w_{ij} = 1,$$

其中 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 均为已知, 令 $C_{jk} = (\boldsymbol{x}_i - \boldsymbol{x}_j)^{\mathrm{T}}(\boldsymbol{x}_i - \boldsymbol{x}_k)$, w_{ij} 有闭式解

$$w_{ij} = \frac{\sum\limits_{k\in Q_i} C_{jk}^{-1}}{\sum\limits_{l,s\in Q_i} C_{ls}^{-1}} .$$

接着根据邻域重构系数不变，去求解低维坐标：

$$\min_{\boldsymbol{z}_1,\boldsymbol{z}_2,\dots,\boldsymbol{z}_m} \sum_{i=1}^{m} \left\| \boldsymbol{z}_i - \sum_{j\in Q_i} w_{ij}\boldsymbol{z}_j \right\|_2^2$$

$$令\ \mathbf{Z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_m) \in \mathbb{R}^{d'\times m},\ (\mathbf{W})_{ij} = w_{ij}$$

$$\mathbf{M} = (\mathbf{I} - \mathbf{W})^{\mathrm{T}}(\mathbf{I} - \mathbf{W})$$

这样利用矩阵M，优化问题可以重写为：

$$\min_{\mathbf{Z}} \ \mathrm{tr}(\mathbf{Z}\mathbf{M}\mathbf{Z}^{\mathrm{T}})$$

特征值分解又来了~

$$\text{s.t.}\ \ \mathbf{Z}\mathbf{Z}^{\mathrm{T}} = \mathbf{I}\ .$$

M特征值分解后最小的d’ 个特征值对应的特征向量组成Z，LLE算法的具体流程如下图所示：

输入：样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$;
近邻参数 k;
低维空间维数 d'.

过程：

1: **for** $i=1,2,\ldots,m$ **do**
2: 确定 $\boldsymbol{x}_i$ 的 k 近邻;
3: 从式(10.27)求得 $w_{ij}, j\in Q_i$;
4: 对于 $j\notin Q_i$, 令 $w_{ij}=0$;
5: **end for**
6: 从式(10.30)得到 $\mathbf{M}$;
7: 对 $\mathbf{M}$ 进行特征值分解;
8: **return** $\mathbf{M}$ 的最小 d' 个特征值对应的特征向量

输出：样本集 D 在低维空间的投影 $Z=\{\boldsymbol{z}_1,\boldsymbol{z}_2,\ldots,\boldsymbol{z}_m\}$.

局部(邻域)线性关系保持不变

和Isomap一样只得到了低维坐标

11.6 度量学习

本篇一开始就提到维数灾难，即在高维空间进行机器学习任务遇到样本稀疏、距离难计算等诸多的问题，因此前面讨论的降维方法都试图将原空间投影到一个合适的低维空间中，接着在低维空间进行学习任务从而产生较好的性能。事实上，不管高维空间还是低维空间都潜在对应着一个距离度量，那可不可以直接学习出一个距离度量来等效降维呢？例如：**咋们就按照降维后的方式来进行距离的计算，这便是度量学习的初衷。**

首先要学习出距离度量必须先定义一个合适的距离度量形式。对两个样本xi与xj，它们之间的平方欧式距离为：

$$\mathrm{dist}_{\mathrm{ed}}^2(\boldsymbol{x}_i,\boldsymbol{x}_j)=||\boldsymbol{x}_i-\boldsymbol{x}_j||_2^2=dist_{ij,1}^2+dist_{ij,2}^2+\ldots+dist_{ij,d}^2$$

若各个属性重要程度不一样即都有一个权重，则得到加权的平方欧式距离：

$$\mathrm{dist}_{\mathrm{wed}}^2(\boldsymbol{x}_i,\boldsymbol{x}_j)=||\boldsymbol{x}_i-\boldsymbol{x}_j||_2^2=w_1\cdot dist_{ij,1}^2+w_2\cdot dist_{ij,2}^2+\ldots+w_d\cdot dist_{ij,d}^2$$

$$=(\boldsymbol{x}_i-\boldsymbol{x}_j)^{\mathrm{T}}\mathbf{W}(\boldsymbol{x}_i-\boldsymbol{x}_j)$$ ，其中 $\mathbf{W}=\mathrm{diag}(\boldsymbol{w})$ 是一个对角矩阵, $(\mathbf{W})_{ii}=w_i$.

此时各个属性之间都是相互独立无关的，但现实中往往会存在属性之间有关联的情形，例如：身高和体重，一般人越高，体重也会重一些，他们之间存在较大的相关性。这样计算距离就不能分属性单独计算，于是就引入经典的**马氏距离(Mahalanobis distance)**:

$$\mathrm{dist}_{\mathrm{mah}}^2(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\boldsymbol{x}_i-\boldsymbol{x}_j)^{\mathrm{T}}\mathbf{M}(\boldsymbol{x}_i-\boldsymbol{x}_j)=\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_{\mathbf{M}}^2$$

通用马氏距离

标准的马氏距离中M是协方差矩阵的逆，马氏距离是一种考虑属性之间相关性且尺度无关（即无须去量纲）的距离度量。

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T\Sigma^{-1}(\vec{x}-\vec{y})}$$ 标准马氏距离

矩阵M也称为“度量矩阵”，为保证距离度量的非负性与对称性，M必须为(半)正定对称矩阵，这样就为度量学习定义好了距离度量的形式，换句话说：**度量学习便是对度量矩阵进行学习。**现在来回想一下前面我们接触的机器学习不难发现：**机器学习算法几乎都是在优化目标函数，从而求解目标函数中的参数。**同样对于度量学习，也需要设置一个优化目标，书中简要介绍了错误率和相似性两种优化目标，此处限于篇幅不进行展开。

在此，降维和度量学习就介绍完毕。**降维是将原高维空间嵌入到一个合适的低维子空间中，接着在低维空间中进行学习任务；度量学习则是试图去学习出一个距离度量来等效降维的效果**，两者都是为了解决维数灾难带来的诸多问题。也许大家最后心存疑惑，那kNN呢，为什么一开头就说了kNN算法，但是好像和后面没有半毛钱关系？正是因为在降维算法中，低维子空间的维数d’通常都由人为指定，因此我们需要使用一些低开销的学习器来选取合适的d’，**kNN这家伙懒到家了根本无心学习，在训练阶段开销为零，测试阶段也只是遍历计算了距离，因此拿kNN来进行交叉验证就十分有优势了~同时降维后样本密度增大同时距离计算变易，更为kNN来展示它独特的十八般手艺提供了用武之地。**