数据降维

一、维度灾难

1. k 近邻法要求样本点比较密集。给定测试样本点 $\vec{\mathbf{x}}$，理论上希望在 $\vec{\mathbf{x}}$ 附近距离 $\delta>0$ 范围内总能找到一个训练样本 $\vec{\mathbf{x}}'$，其中 δ 是个充分小的正数。即：要求训练样本的采样密度足够大，也称作“密采样”。
   - 假设 $\delta=0.001$，且假设样本点只有一个特征，且该特征归一化后范围是 `[0,1]`，则需要 1000 个样本点平均分布在 `[0,1]` 之间。

     此时任何测试样本在其附近 0.001 距离范围内总能找到一个训练样本。
   - 假设 $\delta=0.001$，且假设样本点只有十个特征，且该特征归一化后范围是 `[0,1]`，则需要 10^{30} 个样本点平均分布 `[0,1]` 之间。

     此时任何测试样本在其附近 0.001 距离范围内总能找到一个训练样本。
2. 现实应用中特征维度经常成千上万，要满足密采样所需的样本数目是个天文数字。

   另外许多学习方法都涉及距离计算，而高维空间会给距离计算带来很大的麻烦（高维空间中计算内积都麻烦）。

   在高维情形下出现的数据样本稀疏、距离计算困难等问题是所有机器学习方法共同面临的严重障碍，称作“维度灾难”(`curse of dimensionality`)。
3. 缓解维度灾难的一个重要途径是降维(`dimension reduction`)。

   降维之所以有效的原因是：人们观测或者收集到的数据样本虽然是高维的，但是与学习任务密切相关的也许仅仅是某个低维分布，即高维空间中的一个低维“嵌入”。
4. 常见的降维方法：
   - 监督降维算法。如：线性判别分析 `Linear Discriminant Analysis:LDA` 。
   - 无监督降维算法。如：主成分分析 `PCA` 。
5. 对于降维效果的评估，通常是比较降维前后学习器的性能。如果性能有所提高，则认为降维起了作用。

   也可以将维数降至二维或者三维，然后通过可视化技术来直观地判断降维效果。
6. 对于常见的降维算法，无论是 `PCA` 还是流形学习，都是基于距离来计算重构误差。此时建议对特征进行标准化，因为距离的计算依赖于特征的量纲。如身高特征：
   - 如果采用 `m` 量纲，则取值范围通常在 `1~2` 之间。
   - 如果采用 `cm` 量纲，则取值范围通常在 `100~200` 之间。

   采用不同的量纲会导致不同的重构误差。

二、主成分分析 PCA

1. 主成分分析 `Principal Component Analysis:PCA` 是最常用的一种降维方法。

2.1 PCA 原理

2.1.1 坐标变换

1. 给定数据集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$，其中 $\vec{\mathbf{x}}_i\in\mathbb{R}^n$ 。假定样本经过了中心化，即：

$$\vec{\mathbf{x}}_i \leftarrow \vec{\mathbf{x}}_i - \frac{1}{N}\sum_{j=1}^{N}\vec{\mathbf{x}}_j$$

- $\bar{\vec{\mathbf{x}}} = \frac{1}{N}\sum_{j=1}^{N}\vec{\mathbf{x}}_j$ 称作数据集 $\mathbb{D}$ 的中心向量，它的各元素就是各个特征的均值。
- 之所以进行中心化，是因为经过中心化之后常规的线性变换就是绕原点的旋转变换，也就是坐标变换。

2. 假设坐标变换矩阵为 $\mathbf{W}' \in \mathbb{R}^{n\times n}$ ，经过变换之后样本 $\vec{\mathbf{x}}$ 的坐标为： $\vec{\mathbf{z}}' = \mathbf{W}'^{T}\vec{\mathbf{x}} = (z_1,\cdots,z_n)^T$ 。其中 $\mathbf{W}' = (\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_n)$ ， $\vec{\mathbf{w}}_i \in \mathbb{R}^n$。

   令 $\vec{\mathbf{z}} = (z_1,\cdots,z_d)^T, d<n$ ，它表示样本 $\vec{\mathbf{x}}$ 降低到 d 维度。令 $\mathbf{W} = (\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_d)$ ，则有： $\vec{\mathbf{z}} = \mathbf{W}^T\vec{\mathbf{x}}$ 。

   根据坐标变换矩阵的性质，有：

   - $||\vec{\mathbf{w}}_i||_2 = 1$ ， $i = 1,2,\cdots,d$ 。
   - $\vec{\mathbf{w}}_i\cdot\vec{\mathbf{w}}_j = 0, i\neq j$ 。
   - $\mathbf{W}'\mathbf{W}'^{T} = \mathbf{I}_{n\times n}$ ， $\mathbf{W}^T\mathbf{W} = \mathbf{I}_{d\times d}$ 。

3. 对数据集 $\mathbb{D}$ 中的样本 $\vec{\mathbf{x}}_i$ ，降维后的数据为 $\vec{\mathbf{z}}_i$ 。令：

$$\mathbf{X} = \begin{bmatrix}\vec{\mathbf{x}}_1^T\\ \vdots\\ \vec{\mathbf{x}}_N^T\end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n}\\ x_{2,1} & x_{2,2} & \cdots & x_{2,n}\\ \vdots & \vdots & \ddots & \vdots\\ x_{N,1} & x_{N,2} & \cdots & x_{N,n}\end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix}\vec{\mathbf{z}}_1^T\\ \vdots\\ \vec{\mathbf{z}}_N^T\end{bmatrix} = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,d}\\ z_{2,1} & z_{2,2} & \cdots & z_{2,d}\\ \vdots & \vdots & \ddots & \vdots\\ z_{N,1} & z_{N,2} & \cdots & z_{N,d}\end{bmatrix}$$

   即 $\mathbf{X}$ 的第 i 行就是样本 $\vec{\mathbf{x}}_i^T$ ， $\mathbf{Z}$ 的第 i 行就是降维后的数据 $\vec{\mathbf{z}}_i^T$ 。

   - 令 $\vec{\mathbf{u}}_j = (x_{1,j},\cdots,x_{N,j})^T$ ，它表示 $\mathbf{X}$ 的第 j 列，也就是原始的第 j 个特征。
   - 令 $\vec{\mathbf{v}}_j = (z_{1,j},\cdots,z_{N,j})^T$ ，它表示 $\mathbf{Z}$ 的第 j 列，也就是降维之后的第 j 个特征。

   则根据 $z_{i,j} = \vec{\mathbf{w}}_j\cdot\vec{\mathbf{x}}_i = \sum_{k=1}^{n} w_{j,k}\times x_{i,k}$，有：

$$\vec{\mathbf{v}}_j = \sum_{k=1}^{n} w_{j,k}\vec{\mathbf{u}}_j$$

   因此降维的物理意义为：通过线性组合原始特征，从而去掉一些冗余的或者不重要的特征、保留重要的特征。

2.1.2 重构误差

1. 考虑对 $\vec{\mathbf{z}}$ 进行重构，重构之后的样本为： $\hat{\vec{\mathbf{x}}} = \mathbf{W}\vec{\mathbf{z}}$ 。

   对整个数据集 $\mathbb{D}$ 所有重建样本与原始样本的误差为：

$$\sum_{i=1}^{N}||\hat{\vec{\mathbf{x}}}_i - \vec{\mathbf{x}}_i||_2^2 = \sum_{i=1}^{N}||\mathbf{W}\mathbf{W}^T\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_i||_2^2$$

2. 根据定义有：

$$\mathbf{W}\mathbf{W}^T\vec{\mathbf{x}}_i = (\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_d)\begin{bmatrix}\vec{\mathbf{w}}_1^T\\ \vec{\mathbf{w}}_2^T\\ \vdots\\ \vec{\mathbf{w}}_d^T\end{bmatrix}\vec{\mathbf{x}}_i = \sum_{j=1}^{d}\vec{\mathbf{w}}_j(\vec{\mathbf{w}}_j^T\vec{\mathbf{x}}_i)$$

   由于 $\vec{\mathbf{w}}_j^T\vec{\mathbf{x}}_i$ 是标量，所以有： $\mathbf{W}\mathbf{W}^T\vec{\mathbf{x}}_i = \sum_{j=1}^{d}(\vec{\mathbf{w}}_j^T\vec{\mathbf{x}}_i)\vec{\mathbf{w}}_j$ 。

由于标量的转置等于它本身，所以有：$\mathbf{W}\mathbf{W}^T\vec{\mathbf{x}}_i=\sum_{j=1}^d(\vec{\mathbf{x}}_i^T\vec{\mathbf{w}}_j)\vec{\mathbf{w}}_j$ 。

则有：

$$\begin{aligned}\sum_{i=1}^N||\hat{\vec{\mathbf{x}}}_i-\vec{\mathbf{x}}_i||_2^2&=\sum_{i=1}^N||\mathbf{W}\mathbf{W}^T\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_i||_2^2\\&=\sum_{i=1}^N||\sum_{j=1}^d(\vec{\mathbf{x}}_i^T\vec{\mathbf{w}}_j)\vec{\mathbf{w}}_j-\vec{\mathbf{x}}_i||_2^2\\&=\sum_{i=1}^N||\vec{\mathbf{x}}_i-\sum_{j=1}^d(\vec{\mathbf{x}}_i^T\vec{\mathbf{w}}_j)\vec{\mathbf{w}}_j||_2^2\end{aligned}$$

3. 根据 $\mathbf{X}$ 的定义，可以证明（ $||\cdot||_F$ 为矩阵的`Frobenius`范数）：

$$||\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T||_F^2=\sum_{i=1}^N||\vec{\mathbf{x}}_i-\sum_{j=1}^d(\vec{\mathbf{x}}_i^T\vec{\mathbf{w}}_j)\vec{\mathbf{w}}_j||_2^2$$

证明：

$$\begin{aligned}\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T&=\mathbf{X}-\begin{bmatrix}\vec{\mathbf{x}}_1^T\\\vdots\\\vec{\mathbf{x}}_N^T\end{bmatrix}[\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_d]\begin{bmatrix}\vec{\mathbf{w}}_1^T\\\vdots\\\vec{\mathbf{w}}_d^T\end{bmatrix}\\&=\mathbf{X}-\begin{bmatrix}\vec{\mathbf{x}}_1^T\vec{\mathbf{w}}_1&\cdots&\vec{\mathbf{x}}_1^T\vec{\mathbf{w}}_d\\\vdots&\ddots&\vdots\\\vec{\mathbf{x}}_N^T\vec{\mathbf{w}}_1&\cdots&\vec{\mathbf{x}}_N^T\vec{\mathbf{w}}_d\end{bmatrix}\begin{bmatrix}\vec{\mathbf{w}}_1^T\\\vdots\\\vec{\mathbf{w}}_d^T\end{bmatrix}\\&=\mathbf{X}-\begin{bmatrix}\sum_{k=1}^d w_{k,1}\times\vec{\mathbf{x}}_1^T\vec{\mathbf{w}}_k&\cdots&\sum_{k=1}^d w_{k,n}\times\vec{\mathbf{x}}_1^T\vec{\mathbf{w}}_k\\\vdots&\ddots&\vdots\\\sum_{k=1}^d w_{k,1}\times\vec{\mathbf{x}}_N^T\vec{\mathbf{w}}_k&\cdots&\sum_{k=1}^d w_{k,n}\times\vec{\mathbf{x}}_N^T\vec{\mathbf{w}}_k\end{bmatrix}\end{aligned}$$

则有：

$$\begin{aligned}||\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T||_F^2&=\sum_{i=1}^N\sum_{j=1}^n\left[x_{i,j}-\left(\sum_{k=1}^d w_{k,j}\times\vec{\mathbf{x}}_i^T\vec{\mathbf{w}}_k\right)\right]^2\\&=\sum_{i=1}^N||\vec{\mathbf{x}}_i-\sum_{k=1}^d(\vec{\mathbf{x}}_i^T\vec{\mathbf{w}}_k)\vec{\mathbf{w}}_k||_2^2\end{aligned}$$

将最后的下标从 k 替换为 j 即可得证。

4. `PCA`降维要求重构误差最小。现在求解最优化问题：

$$\begin{aligned}\mathbf{W}^*&=\arg\min_{\mathbf{W}}\sum_{i=1}^N||\hat{\vec{\mathbf{x}}}_i-\vec{\mathbf{x}}_i||_2^2=\arg\min_{\mathbf{W}}||\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T||_F^2\\&=\arg\min_{\mathbf{W}}tr\left[(\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T)^T(\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T)\right]\\&=\arg\min_{\mathbf{W}}tr\left[(\mathbf{X}^T-\mathbf{W}\mathbf{W}^T\mathbf{X}^T)(\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T)\right]\\&=\arg\min_{\mathbf{W}}tr\left[\mathbf{X}^T\mathbf{X}-\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T-\mathbf{W}\mathbf{W}^T\mathbf{X}^T\mathbf{X}+\mathbf{W}\mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T\right]\\&=\arg\min_{\mathbf{W}}\left[tr(\mathbf{X}^T\mathbf{X})-tr(\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)-tr(\mathbf{W}\mathbf{W}^T\mathbf{X}^T\mathbf{X})+tr(\mathbf{W}\mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)\right]\end{aligned}$$

- 因为矩阵及其转置的迹相等，因此 $tr(\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)=tr(\mathbf{W}\mathbf{W}^T\mathbf{X}^T\mathbf{X})$ 。

  由于可以在 $tr(\cdot)$ 中调整矩阵的顺序，则 $tr(\mathbf{W}\mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)=tr(\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T\mathbf{W}\mathbf{W}^T)$ 。

  考虑到：

  $$\mathbf{W}^T\mathbf{W}=\begin{bmatrix}\vec{\mathbf{w}}_1^T\\ \vdots\\ \vec{\mathbf{w}}_d^T\end{bmatrix}(\vec{\mathbf{w}}_1,\cdots,\vec{\mathbf{w}}_d)=\mathbf{I}_{d\times d}$$

  代入上式有：$tr(\mathbf{W}\mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)=tr(\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)$ 。

  于是有：

  $$\mathbf{W}^*=\arg\min_{\mathbf{W}}\left[tr(\mathbf{X}^T\mathbf{X})-tr(\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)\right]$$

- 由于 $tr(\mathbf{X}^T\mathbf{X})$ 与 $\mathbf{W}$ 无关，因此：

  $$\mathbf{W}^*=\arg\min_{\mathbf{W}}-tr(\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)=\arg\max_{\mathbf{W}}tr(\mathbf{X}^T\mathbf{X}\mathbf{W}\mathbf{W}^T)$$

- 调整矩阵顺序，则有：$\mathbf{W}^*=\arg\max_{\mathbf{W}}tr(\mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W})$ 。其约束条件为： $\mathbf{W}^T\mathbf{W}=\mathbf{I}_{d\times d}$ 。

5. PCA 最优化问题需要求解就是 $\mathbf{X}^T\mathbf{X}$ 的特征值。
   - 只需要对矩阵 $\mathbf{X}^T\mathbf{X}\in\mathbb{R}^{n\times n}$ 进行特征值分解，将求得的特征值排序： $\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_n$ 。然后取前 d 个特征值对应的单位特征向量构成坐标变换矩阵 $\mathbf{W}=(\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_d)$ 。
   - 当样本数据进行了中心化时， $\sum_{i=1}^N\vec{\mathbf{x}}_i\vec{\mathbf{x}}_i^T=\mathbf{X}^T\mathbf{X}$ 就是样本集的协方差矩阵。这也是为什么需要对样本进行中心化的一个原因。

2.2 PCA 算法

1. PCA 算法：
   - 输入：
     - 样本集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_N\}$ 。
     - 低维空间维数 d 。
   - 输出：投影矩阵 $\mathbf{W}=(\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_d)$ 。
   - 算法步骤：
     - 对所有样本进行中心化操作： $\vec{\mathbf{x}}_i\leftarrow\vec{\mathbf{x}}_i-\frac{1}{N}\sum_{j=1}^N\vec{\mathbf{x}}_j$ 。
     - 计算样本的协方差矩阵 $\mathbf{X}^T\mathbf{X}$ 。
     - 对协方差矩阵 $\mathbf{X}^T\mathbf{X}$ 做特征值分解。
     - 取最大的 d 个特征值对应的单位特征向量 $\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_d$ ，构造投影矩阵 $\mathbf{W}=(\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_d)$ 。
2. 通常低维空间维数 d 的选取有两种方法：
   - 通过交叉验证法选取较好的 d 。“比较好”指的是在降维后的学习器的性能比较好。
   - 从算法原理的角度设置一个阈值，比如 $t=95\%$ ，然后选取使得下式成立的最小的 d 的值：

     $$\frac{\sum_{i=1}^d\lambda_i}{\sum_{i=1}^n\lambda_i}\geq t$$

     其中 λ_i 从大到小排列。

2.3 性质

1. 从物理意义上看： 给定协方差矩阵 $\mathbf{X}^T\mathbf{X}$，通过坐标变换将其对角化为矩阵：

$$\Lambda=\begin{pmatrix}\lambda_1 & 0 & 0 & \cdots & 0\\ 0 & \lambda_2 & 0 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & \lambda_n\end{pmatrix}$$

   这相当于在新的坐标系中：

   - 任意一对特征之间的协方差为 0 。
   - 单个特征的方差为 $\lambda_i, i=1,2,\cdots,n$ 。

   即：数据在每个维度上尽可能分散，且任意两个维度之间不相关。

   降维的过程就是寻找这样的一个坐标变换，也就是坐标变换矩阵 $\mathbf{W}$ 。

   由于协方差矩阵 $\mathbf{X}^T\mathbf{X}$ 是对称矩阵，根据实对称矩阵的性质，这样的坐标变换矩阵一定存在。

2. `PCA` 算法中，低维空间与高维空间必然不相同。因为末尾 $n-d$ 个最小的特征值对应的特征向量被抛弃了，这就是降维导致的结果。

   - 舍弃这部分信息之后能使得样本的采样密度增大（因为维数降低了），这是缓解维度灾难的重要手段。
   - 当数据受到噪声影响时，最小特征值对应的特征向量往往与噪声有关，将它们舍弃能在一定程度上起到降噪的效果。

3. `PCA` 降低了输入数据的维度同时保留了主要信息/能量，但是这个主要信息只是针对训练集的，而且这个主要信息未必是重要信息。

   有可能舍弃了一些看似无用的信息，但是这些看似无用的信息恰好是重要信息，只是在训练集上没有很大的表现，所以 `PCA` 也可能加剧了过拟合。

4. `PCA` 中训练样本越多越好。

   - 如果训练样本太少，则训练集很有可能“偶然”近似落在同一个平面上。
   - 极端情况下，如果样本数量小于目标维度，比如样本数量为 100，目标维度为 1000 维。则这 100个样本总可以构成一个 1000 维的平面，且这样的平面有无穷多个。此时如果进行 `PCA` 降维，则前几个特征值 λ 占比非常大。

     但是如果将样本数量扩充为 10000 ，则这些样本构成一个 1000 维的平面的巧合就几乎很难成立。此时如果进行 `PCA` 降维，则前几个特征值 λ 占比就会降低。

   - 本质上是因为 N 决定了协方差矩阵 $\mathbf{X}^T\mathbf{X}$ 的秩的上界。

     当 N 较小时，$rank(\mathbf{X}^T\mathbf{X})$ 也会很小，导致大量的特征值 λ 为 0 。

5. `PCA` 不仅将数据压缩到低维，它也使得降维之后的数据各特征相互独立。

   注意： `PCA` 推导过程中，并没有要求数据中心化；但是在推导协方差矩阵时，要求数据中心化。此时：

$$Var[\vec{\mathbf{z}}]=\frac{1}{N-1}\mathbf{Z}^T\mathbf{Z}=\frac{1}{N-1}\Sigma^2$$

   其中：

   - Σ 为 $\mathbf{X}^T\mathbf{X}$ 的最大的 d 个特征值组成的对角矩阵。
   - $\mathbf{Z}$ 为降维后的样本集组成的矩阵。

6. 对于训练集、验证集、测试集，当对训练集进行 `PCA` 降维时，也需要对验证集、测试集执行同样的降维。

注意：**对验证集、测试集执行中心化操作时，中心向量必须从训练集计算而来。**不能使用验证集的中心向量，也不能用测试集的中心向量。

2.4 最大可分性

1. PCA 降维的准则有两个：
   - 最近重构性：样本集中所有点，重构后的点距离原来的点的误差之和最小（就是前面介绍的内容）。
   - 最大可分性：样本点在低维空间的投影尽可能分开。
2. 可以证明，最近重构性就等价于最大可分性。证明如下：

   对于样本点 $\vec{\mathbf{x}}_i$， 它在降维后空间中的投影是 $\vec{\mathbf{z}}_i$。 则有： $\vec{\mathbf{z}}=\mathbf{W}^T\vec{\mathbf{x}}$。

   由于样本数据进行了中心化，则投影后样本点的方差是：

   $$\sum_{i=1}^{N}\vec{\mathbf{z}}_i\vec{\mathbf{z}}_i^T=\sum_{i=1}^{N}\mathbf{W}^T\vec{\mathbf{x}}_i\vec{\mathbf{x}}_i^T\mathbf{W}$$

   根据 $\mathbf{X}$ 的定义，有：$tr(\mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W})=\sum_{i=1}^{N}\mathbf{W}^T\vec{\mathbf{x}}_i\vec{\mathbf{x}}_i^T\mathbf{W}$。则样本点的方差最大的优化目标可写作：

   $$\max_{\mathbf{W}} tr(\mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W})$$
   $$s.t.\ \mathbf{W}^T\mathbf{W}=\mathbf{I}_{d\times d}$$

   这就是前面最近重构性推导的结果。
3. LDA 也可以用于降维。对于2维空间降低到1维直线的情况下，它设法将样例投影到某一条直线上，使得同类样例的投影点尽可能接近、异类样例的投影点尽可能远离。
   - LDA 考虑的是：向类别区分最大的方向投影。如下图中的绿色投影直线。
   - PCA 考虑的是：向方差最大的方向投影。如下图中的紫色投影直线。

   因此 LDA 降维对于类别的区分效果要好的多。

    pca_lda

2.5 PCA 与 SVD

1. 酉矩阵：若 n 阶矩阵满足 $\mathbf{U}^H\mathbf{U}=\mathbf{U}\mathbf{U}^H=\mathbf{I}$，则它是酉矩阵。其中 $\mathbf{U}^H$ 为 $\mathbf{U}$ 的共轭转置。

   $\mathbf{U}$ 为酉矩阵的充要条件是： $\mathbf{U}^H=\mathbf{U}^{-1}$ 。
2. 奇异值分解：设 $\mathbf{X}$ 为 $N\times n$ 阶矩阵，且 $rank(\mathbf{X})=r$，则存在 N 阶酉矩阵 $\mathbf{V}$ 和 n 阶酉矩阵 $\mathbf{U}$ ，使得：

   $$\mathbf{V}^H\mathbf{X}\mathbf{U}=\begin{bmatrix}\Sigma & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}_{N\times n}$$

   其中

   $$\Sigma=\begin{bmatrix}\sigma_1 & 0 & 0 & \cdots & 0\\ 0 & \sigma_2 & 0 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & \sigma_r\end{bmatrix}$$

   $\sigma_1\geq\sigma_2\geq\cdots\geq\sigma_r>0$ 称作矩阵 $\mathbf{X}$ 的奇异值。
3. 根据酉矩阵的性质， $\mathbf{V}\mathbf{V}^H=\mathbf{I}_{N\times N},\mathbf{U}\mathbf{U}^H=\mathbf{I}_{n\times n}$，则有：

$$\mathbf{X}=\mathbf{V}\begin{bmatrix}\Sigma & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}_{N\times n}\mathbf{U}^H\Longrightarrow\mathbf{X}^H=\mathbf{U}\begin{bmatrix}\Sigma & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}_{n\times N}\mathbf{V}^H$$

则有 $\mathbf{X}^H\mathbf{X}=\mathbf{U}\mathbf{M}\mathbf{U}^H$， 其中 $\mathbf{M}$ 是个 n 阶对角矩阵：

$$\mathbf{M}=\begin{bmatrix}\Sigma & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}_{n\times N}\times\begin{bmatrix}\Sigma & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}_{N\times n}=\begin{bmatrix}\lambda_1 & 0 & 0 & \cdots & 0\\ 0 & \lambda_2 & 0 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & \lambda_n\end{bmatrix}_{n\times n}$$

$$\lambda_i=\sigma_i^2\quad,i=1,2,\cdots,r$$
$$\lambda_i=0\quad,i=r+1,r+2,\cdots,n$$

4. 由数据集 $\mathbb{D}$ 中样本构成的 $\mathbf{X}$ 为实矩阵，因此有 $\mathbf{X}^H=\mathbf{X}^T$ 。另外考虑到 $\mathbf{X}^T\mathbf{X}$ 为实对称矩阵，因此 $\mathbf{V}$ 也是实矩阵，因此 $\mathbf{U}^H=\mathbf{U}^T$。 则有：

   $$\mathbf{X}^T\mathbf{X}=\mathbf{U}\mathbf{M}\mathbf{U}^T$$

   - 根据 $\mathbf{U}\mathbf{U}^T=\mathbf{I}$，则有：$\mathbf{X}^T\mathbf{X}\mathbf{U}=\mathbf{U}\mathbf{M}$。
   - 根据 $\mathbf{M}$ 是个对角矩阵的性质，有：$\mathbf{U}\mathbf{M}=\mathbf{M}\mathbf{U}$，则有： $\mathbf{X}^T\mathbf{X}\mathbf{U}=\mathbf{M}\mathbf{U}$。

     则 $\lambda_i,i=1,2,\cdots,r$ 就是的 $\mathbf{X}^T\mathbf{X}$ 特征值， 其对应的单位特征向量组成正交矩阵 $\mathbf{U}$。

     因此 `SVD` 奇异值分解等价于 `PCA` 主成分分析，核心都是求解 $\mathbf{X}^T\mathbf{X}$ 的特征值以及对应的单位特征向量。

三、核化线性降维 KPCA

1. `PCA` 方法假设从高维空间到低维空间的映射是线性的，但是在不少现实任务中可能需要非线性映射才能找到合适的低维空间来降维。

   非线性降维的一种常用方法是基于核技巧对线性降维方法进行核化 `kernelized`， 如核主成分分析 `Kernelized PCA:KPCA` ，它是对 `PCA` 的一种推广。

2. 假定原始特征空间中的样本点 $\vec{\mathbf{x}}_i$ 通过映射 ϕ 映射到高维特征空间的坐标为 $\vec{\mathbf{x}}_{i,\phi}$，即 $\vec{\mathbf{x}}_{i,\phi}=\phi(\vec{\mathbf{x}}_i)$。且假设高维特征空间为 n 维的，即： $\vec{\mathbf{x}}_{i,\phi}\in\mathbb{R}^n$ 。假定要将高维特征空间中的数据投影到低维空间中，投影矩阵为 $\mathbf{W}$ 为 $n\times d$ 维矩阵。

   根据 `PCA` 推导的结果，求解方程：$\mathbf{X}_\phi^T\mathbf{X}_\phi\mathbf{W}=\lambda\mathbf{W}$ 。其中 $\mathbf{X}_\phi=(\vec{\mathbf{x}}_{1,\phi}^T,\vec{\mathbf{x}}_{2,\phi}^T,\cdots,\vec{\mathbf{x}}_{N,\phi}^T)^T$ 为 $N\times n$ 维矩阵。

   于是有：$\left(\sum_{i=1}^N\phi(\vec{\mathbf{x}}_i)\phi(\vec{\mathbf{x}}_i)^T\right)\mathbf{W}=\lambda\mathbf{W}$ 。

3. 通常并不清楚 ϕ 的解析表达式，因此并不会直接得到 $\mathbf{X}_\phi$ ，所以无法直接求解方程：$\mathbf{X}_\phi^T\mathbf{X}_\phi\mathbf{W}=\lambda\mathbf{W}$ 。

   于是引入核函数：$\kappa(\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j)=\phi(\vec{\mathbf{x}}_i)^T\phi(\vec{\mathbf{x}}_j)$ 。

   - 定义核矩阵 ：

     $$\mathbf{K}=\begin{bmatrix}\kappa(\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_1) & \kappa(\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2) & \cdots & \kappa(\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_N)\\ \kappa(\vec{\mathbf{x}}_2,\vec{\mathbf{x}}_1) & \kappa(\vec{\mathbf{x}}_2,\vec{\mathbf{x}}_2) & \cdots & \kappa(\vec{\mathbf{x}}_2,\vec{\mathbf{x}}_N)\\ \vdots & \vdots & \ddots & \vdots\\ \kappa(\vec{\mathbf{x}}_N,\vec{\mathbf{x}}_1) & \kappa(\vec{\mathbf{x}}_N,\vec{\mathbf{x}}_2) & \cdots & \kappa(\vec{\mathbf{x}}_N,\vec{\mathbf{x}}_N)\end{bmatrix}$$

     则有： $\mathbf{X}_\phi\mathbf{X}_\phi^T=\mathbf{K}$ 。

- 定义 $\vec{\alpha}_i = \frac{\vec{\mathbf{x}}_{i,\phi}^T \mathbf{W}}{\lambda}$ ，则 $\vec{\alpha}_i$ 为 $1 \times d$ 维行向量 。定义：$\mathbf{A} = (\vec{\alpha}_1, \vec{\alpha}_2, \cdots, \vec{\alpha}_N)^T$ 为 $N \times d$ 维矩阵

  则有：

$$\mathbf{W} = \frac{1}{\lambda}\left(\sum_{i=1}^{N} \vec{\mathbf{x}}_{i,\phi} \vec{\mathbf{x}}_{i,\phi}^T\right) \mathbf{W} = \sum_{i=1}^{N} \vec{\mathbf{x}}_{i,\phi} \frac{\vec{\mathbf{x}}_{i,\phi}^T \mathbf{W}}{\lambda} = \sum_{i=1}^{N} \vec{\mathbf{x}}_{i,\phi} \vec{\alpha}_i = \mathbf{X}_\phi^T \mathbf{A}$$

4. 将 $\mathbf{W} = \mathbf{X}_\phi^T \mathbf{A}$ 代入 $\mathbf{X}_\phi^T \mathbf{X}_\phi \mathbf{W} = \lambda \mathbf{W}$ ，有：$\mathbf{X}_\phi^T \mathbf{X}_\phi \mathbf{X}_\phi^T \mathbf{A} = \lambda \mathbf{X}_\phi^T \mathbf{A}$ 。
   - 两边同时左乘以 $\mathbf{X}_\phi$，再代入 $\mathbf{X}_\phi \mathbf{X}_\phi^T = \mathbf{K}$ 有：$\mathbf{K}\mathbf{K}\mathbf{A} = \lambda \mathbf{K}\mathbf{A}$ 。
   - 通常会要求核矩阵可逆，上式两边同时左乘以 $\mathbf{K}^{-1}$ ，则有：$\mathbf{K}\mathbf{A} = \lambda \mathbf{A}$ 。

     同样该问题也是一个特征值分解问题，取 $\mathbf{K}$ 最大的 d 个特征值对应的特征向量组成 $\mathbf{A}$ 即可。
5. 对于新样本 $\vec{\mathbf{x}}$， 其投影后第 j 维的坐标为：

$$z_j = \vec{\mathbf{w}}_j^T \phi(\vec{\mathbf{x}}) = \sum_{i=1}^{N} \alpha_{i,j} \phi(\vec{\mathbf{x}}_i)^T \phi(\vec{\mathbf{x}}) = \sum_{i=1}^{N} \alpha_{i,j} \kappa(\vec{\mathbf{x}}_i, \vec{\mathbf{x}})$$

   其中 $\alpha_{i,j}$ 为行向量 $\vec{\alpha}_i$ 的第 j 个分量。

   可以看到：为了获取投影后的坐标， `KPCA` 需要对所有样本求和，因此它的计算开销较大。

四、流形学习

1. 流形学习 `manifold learning` 是一类借鉴了拓扑流形概念的降维方法。
2. 流形是在局部和欧氏空间同胚的空间，它在局部具有欧氏空间的性质，能用欧氏距离进行距离计算。

   如果低维流形嵌入到高维空间中，则数据样本在高维空间的分布虽然看起来非常复杂，但是在局部上仍然具有欧氏空间的性质。
3. 当维数被降低至二维或者三维时，能对数据进行可视化展示，因此流形学习也可用于可视化。
4. 流形学习若想取得很好的效果，则必须对邻域保持样本密采样，但这恰恰是高维情形下面临的重大障碍。因此流形学习方法在实践中的降维性能往往没有预期的好。
5. 流形学习对于噪音数据非常敏感。噪音数据可能出现在两个区域连接处：
   - 如果没有出现噪音，这两个区域是断路的。
   - 如果出现噪音，这两个区域是短路的。

4.1 多维缩放

4.1.1 多维缩放原理

1. 多维缩放 `Multiple Dimensional Scaling:MDS` 要求原始空间中样本之间的距离在低维空间中得到保持。
2. 假设 N 个样本在原始空间中的距离矩阵为 $\mathbf{D} = (d_{ij})_{N \times N}$：

$$\mathbf{D} = \begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,N} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N,1} & d_{N,2} & \cdots & d_{N,N} \end{bmatrix}$$

   其中 $d_{ij} = ||\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j||$ 为样本 $\vec{\mathbf{x}}_i$ 到样本 $\vec{\mathbf{x}}_j$ 的距离。

假设原始样本是在 n 维空间，目标是获取样本在 n' 维空间且欧氏距离保持不变，其中满足 $n'<n$ 。

假设样本集在原空间的表示为 $\mathbf{X}\in\mathbb{R}^{N\times n}$ ，样本集在降维后空间的表示为 $\mathbf{Z}\in\mathbb{R}^{N\times n'}$ 。

$$\mathbf{X}=\begin{bmatrix}\vec{\mathbf{x}}_1^T\\ \vec{\mathbf{x}}_2^T\\ \vdots\\ \vec{\mathbf{x}}_N^T\end{bmatrix}=\begin{bmatrix}x_{1,1}&x_{1,2}&\cdots&x_{1,n}\\ x_{2,1}&x_{2,2}&\cdots&x_{2,n}\\ \vdots&\vdots&\ddots&\vdots\\ x_{N,1}&x_{N,2}&\cdots&x_{N,n}\end{bmatrix}$$

$$\mathbf{Z}=\begin{bmatrix}\vec{\mathbf{z}}_1^T\\ \vec{\mathbf{z}}_2^T\\ \vdots\\ \vec{\mathbf{z}}_N^T\end{bmatrix}=\begin{bmatrix}z_{1,1}&z_{1,2}&\cdots&z_{1,n'}\\ z_{2,1}&z_{2,2}&\cdots&z_{2,n'}\\ \vdots&\vdots&\ddots&\vdots\\ z_{N,1}&z_{N,2}&\cdots&z_{N,n'}\end{bmatrix}$$

所求的正是 $\mathbf{Z}$ 矩阵，同时也不知道 n'。

很明显：并不是所有的低维空间都满足线性映射之后，样本点之间的欧氏距离保持不变。

3. 令 $\mathbf{B}=\mathbf{Z}\mathbf{Z}^T\in\mathbb{R}^{N\times N}$ ，即：

$$\mathbf{B}=\begin{bmatrix}b_{1,1}&b_{1,2}&\cdots&b_{1,N}\\ b_{2,1}&b_{2,2}&\cdots&b_{2,N}\\ \vdots&\vdots&\ddots&\vdots\\ b_{N,1}&b_{N,2}&\cdots&b_{N,N}\end{bmatrix}$$

其中 $b_{i,j}=\vec{\mathbf{z}}_i\cdot\vec{\mathbf{z}}_j$ 为降维后样本的内积。

则根据降维前后样本的欧氏距离保持不变有：

$$d_{ij}^2=||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2=||\vec{\mathbf{z}}_i||^2+||\vec{\mathbf{z}}_j||^2-2\vec{\mathbf{z}}_i^T\vec{\mathbf{z}}_j=b_{i,i}+b_{j,j}-2b_{i,j}$$

假设降维后的样本集 $\mathbf{Z}$ 被中心化，即 $\sum_{i=1}^N\vec{\mathbf{z}}_i=\vec{\mathbf{0}}$，则矩阵 $\mathbf{B}$ 的每行之和均为零，每列之和均为零。即：

$$\sum_{i=1}^N b_{i,j}=0,j=1,2,\cdots,N$$

$$\sum_{j=1}^N b_{i,j}=0,i=1,2,\cdots,N$$

于是有：

$$\sum_{i=1}^N d_{ij}^2=\sum_{i=1}^N b_{i,i}+Nb_{j,j}=tr(\mathbf{B})+Nb_{j,j}$$

$$\sum_{j=1}^N d_{ij}^2=\sum_{j=1}^N b_{j,j}+Nb_{i,i}=tr(\mathbf{B})+Nb_{i,i}$$

$$\sum_{i=1}^N\sum_{j=1}^N d_{ij}^2=\sum_{i=1}^N(tr(\mathbf{B})+Nb_{i,i})=2Ntr(\mathbf{B})$$

令：

$$d_{i,\cdot}^2=\frac{1}{N}\sum_{j=1}^{N}d_{ij}^2=\frac{tr(\mathbf{B})}{N}+b_{i,i}$$

$$d_{j,\cdot}^2=\frac{1}{N}\sum_{i=1}^{N}d_{ij}^2=\frac{tr(\mathbf{B})}{N}+b_{j,j}$$

$$d_{\cdot,\cdot}^2=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}d_{ij}^2=\frac{2tr(\mathbf{B})}{N}$$

代入 $d_{ij}^2=b_{i,i}+b_{j,j}-2b_{i,j}$ ，有：

$$b_{i,j}=\frac{b_{i,i}+b_{j,j}-d_{ij}^2}{2}=\frac{d_{i,\cdot}^2+d_{j,\cdot}^2-d_{\cdot,\cdot}^2-d_{ij}^2}{2}$$

右式根据 d_{ij} 给出了 $b_{i,j}$，因此可以根据原始空间中的距离矩阵 $\mathbf{D}$ 求出在降维后空间的内积矩阵 $\mathbf{B}$。

现在的问题是已知内积矩阵 $\mathbf{B}=\mathbf{Z}\mathbf{Z}^T$ ，如何求得矩阵 $\mathbf{Z}$。

4. 对矩阵 $\mathbf{B}$ 做特征值分解，设 $\mathbf{B}=\mathbf{V}\Lambda\mathbf{V}^T$，其中

$$\Lambda=\begin{bmatrix}\lambda_1&0&0&\cdots&0\\0&\lambda_2&0&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&\lambda_N\end{bmatrix}$$

为特征值构成的对角矩阵， $\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_N$， $\mathbf{V}$ 为特征向量矩阵。

假定特征值中有 n^*个非零特征值，它们构成对角矩阵 $\Lambda^*=diag(\lambda_1,\lambda_2,\cdots,\lambda_{n^*})$。令 $\mathbf{V}^*$ 为对应的特征向量矩阵，则 $\mathbf{Z}=\mathbf{V}^*{\Lambda^*}^{1/2}$ 。

此时有 $n'=n^*$， $\mathbf{Z}\in\mathbb{R}^{N\times n^*}$ 。

5. 在现实应用中为了有效降维，往往仅需要降维后的距离与原始空间中的距离尽可能相等，而不必严格相等。

此时可以取 $n'\ll n^*\leq n$ 个最大特征值构成对角矩阵 $\tilde{\Lambda}=diag(\lambda_1,\lambda_2,\cdots,\lambda_{n'})$ 。令 $\tilde{\mathbf{V}}$ 表示对应的特征向量矩阵，则 $\mathbf{Z}=\tilde{\mathbf{V}}\tilde{\Lambda}^{1/2}\in\mathbb{R}^{N\times n'}$ 。

4.1.2 多维缩放算法

1. 多维缩放`MDS`算法：
   - 输入：
     - 距离矩阵 $\mathbf{D}\in\mathbb{R}^{N\times N}$ 。
     - 低维空间维数 n' 。
   - 输出：样本集在低维空间中的矩阵 $\mathbf{Z}$ 。
   - 算法步骤：
     - 根据下列式子计算 $d_{i,\cdot}^2,d_{j,\cdot}^2,d_{\cdot,\cdot}^2$ ：

$$d_{i,\cdot}^2=\frac{1}{N}\sum_{j=1}^{N}d_{ij}^2,\quad d_{j,\cdot}^2=\frac{1}{N}\sum_{i=1}^{N}d_{ij}^2,\quad d_{\cdot,\cdot}^2=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}d_{ij}^2$$

     - 根据下式计算矩阵 $\mathbf{B}$：$b_{i,j}=\frac{d_{i,\cdot}^2+d_{j,\cdot}^2-d_{\cdot,\cdot}^2-d_{ij}^2}{2}$ 。
     - 对矩阵 $\mathbf{B}$ 进行特征值分解。

- 取 $\tilde{\Lambda}$ 为 n' 个最大特征值所构成的对角矩阵， $\tilde{\mathbf{V}}$ 表示对应的特征向量矩阵， 计算： $\mathbf{Z}=\tilde{\mathbf{V}}\tilde{\Lambda}^{1/2}\in\mathbb{R}^{N\times n'}$ 。

4.2 等度量映射

4.2.1 算法

1. 等度量映射 `Isometric Mapping:Isomap` 的基本观点是：低维流形嵌入到高维空间后，直接在高维空间中计算直线距离具有误导性。因为在高维空间中的直线距离在低维嵌入流形上是不可达的。
2. 低维嵌入流形上，两点之间的距离是“测地线” `geodesic` 距离。
   - 计算测地线距离的方法是：利用流形在局部上与欧氏空间同胚这个性质，对每个点基于欧氏距离找出它在低维流形上的近邻点， 然后就能建立一个近邻连接图。
     - 图中近邻点之间存在链接。
     - 图中非近邻点之间不存在链接。
   - 于是计算两点之间测地线距离的问题转变为计算近邻连接图上两点之间的最短路径问题（可以通过著名的 `Dijkstra` 算法或者 `Floyd` 算法）。
   - 在得到任意两点的距离之后，就可以通过 `MDS` 算法来获得样本点在低维空间中的坐标。
3. `Isomap` 算法：
   - 输入：
     - 样本集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_N\}$
     - 近邻参数 k 。
     - 低维空间维数 n'。
   - 输出：样本集在低维空间中的矩阵 $\mathbf{Z}$。
   - 算法步骤：
     - 对每个样本点 $\vec{\mathbf{x}}_i$ ，计算它的 k 近邻。同时将 $\vec{\mathbf{x}}_i$ 与 它的 k 近邻的距离设置为欧氏距离，与其他点的距离设置为无穷大。
     - 调用最短路径算法计算任意两个样本点之间的距离，获得距离矩阵 $\mathbf{D}\in\mathbb{R}^{N\times N}$ 。
     - 调用多维缩放 `MDS` 算法，获得样本集在低维空间中的矩阵 $\mathbf{Z}$。

4.2.2 性质

1. `Isomap` 算法有个很大的问题：对于新样本，如何将其映射到低维空间?

   常用的方法是：
   - 将训练样本的高维空间坐标作为输入，低维空间坐标作为输出，训练一个回归学习器。
   - 用这个回归学习器来对新样本的低维空间进行预测。

   这仅仅是一个权宜之计，但是目前并没有更好的办法。如果将新样本添加到样本集中，重新调用 `Isomap` 算法，则会得到一个新的低维空间。
   - 一方面计算量太大（每个新样本都要调用一次 `Isomap` 算法）。
   - 另一方面每次降维后的空间都在变化，不利于降维后的训练过程。
2. 对于近邻图的构建有两种常用方案：
   - 一种方法是指定近邻点个数，比如指定距离最近的 k 个点为近邻点 。这样得到的近邻图称作 k 近邻图。
   - 另一种方法是指定距离阈值 ϵ ， 距离小于 ϵ 的点被认为是近邻点。这样得到的近邻图称作 ϵ 近邻图。

   这两种方案都有不足：

- 如果近邻范围过大，则距离很远的点也被误认作近邻，这就是“短路”问题。
- 如果近邻范围过小，则图中有些区域可能与其他区域不存在连接，这就是“断路”问题 。
  短路问题和断路问题都会给后续的最短路径计算造成误导。

4.3 局部线性嵌入 LLE

1. 与 `Isomap` 试图保持邻域内样本之间的距离不同，局部线性嵌入 `Locally Linear Embedding:LLE` 试图保持邻域内样本之间的线性关系。

   这种线性保持在二维空间中就是保持共线性，三维空间中就是保持共面性。

2. 假定样本点 $\vec{\mathbf{x}}_i$ 的坐标能够通过它的邻域样本 $\vec{\mathbf{x}}_j,\vec{\mathbf{x}}_k,\vec{\mathbf{x}}_l$ 进行线性组合而重构出来，即：
   $\vec{\mathbf{x}}_i=w_{i,j}\vec{\mathbf{x}}_j+w_{i,k}\vec{\mathbf{x}}_k+w_{i,l}\vec{\mathbf{x}}_l$ 。 `LLE` 算法希望这种关系在低维空间中得到保持。

4.3.1 重构系数求解

1. `LLE` 首先为每个样本 $\vec{\mathbf{x}}_i$ 找到其近邻点下标集合 $\mathbb{Q}_i$， 然后计算基于 $\mathbb{Q}_i$ 中的样本点对 $\vec{\mathbf{x}}_i$ 进行线性重构的系数 $\vec{\mathbf{w}}_i$ 。

   定义样本集重构误差为（ $w_{i,j}$ 为 $\vec{\mathbf{w}}_i$ 的分量）：

   $$err=\sum_{i=1}^{N}||\vec{\mathbf{x}}_i-\sum_{j\in\mathbb{Q}_i}w_{i,j}\vec{\mathbf{x}}_j||_2^2$$

   目标是样本集重构误差最小，即：

   $$\min_{\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_N}\sum_{i=1}^{N}||\vec{\mathbf{x}}_i-\sum_{j\in\mathbb{Q}_i}w_{i,j}\vec{\mathbf{x}}_j||_2^2$$

2. 这样的解有无数个，对权重增加约束，进行归一化处理。即：

   $$\sum_{j\in\mathbb{Q}_i}w_{i,j}=1,i=1,2,\cdots,N$$

   现在就是求解最优化问题：

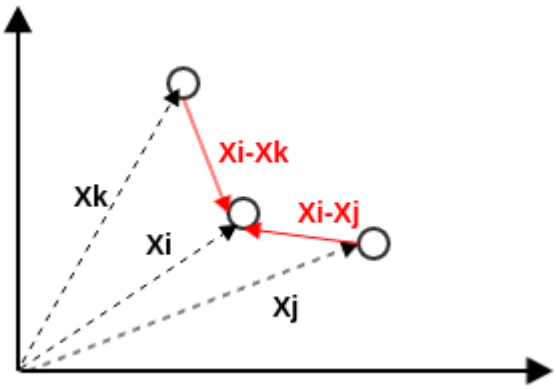
   $$\begin{aligned}&\min_{\vec{\mathbf{w}}_1,\vec{\mathbf{w}}_2,\cdots,\vec{\mathbf{w}}_N}\sum_{i=1}^{N}||\vec{\mathbf{x}}_i-\sum_{j\in\mathbb{Q}_i}w_{i,j}\vec{\mathbf{x}}_j||_2^2\\&s.t.\quad\sum_{j\in\mathbb{Q}_i}w_{i,j}=1,i=1,2,\cdots,N\end{aligned}$$

   该最优化问题有解析解。令 $C_{j,k}=(\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j)^T(\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_k)$ ，则可以解出：

   $$w_{i,j}=\frac{\sum_{k\in\mathbb{Q}_i}C_{j,k}^{-1}}{\sum_{l,s\in\mathbb{Q}_i}C_{l,s}^{-1}},j\in\mathbb{Q}_i$$

   其中：

   - $C_{j,k}$ 刻画了 $\vec{\mathbf{x}}_k$ 到 $\vec{\mathbf{x}}_i$ 的差向量，与 $\vec{\mathbf{x}}_j$ 到 $\vec{\mathbf{x}}_i$ 的差向量的内积。
   - $w_{i,j}$ 刻画了这些内积中，与 $\vec{\mathbf{x}}_j$ 相关的内积的比例。



4.3.2 低维空间保持

1. 求出了线性重构的系数 $\vec{\mathbf{w}}_i$ 之后， LLE 在低维空间中保持 $\vec{\mathbf{w}}_i$ 不变。

   设 $\vec{\mathbf{x}}_i$ 对应的低维坐标 $\vec{\mathbf{z}}_i$ ， 已知线性重构的系数 $\vec{\mathbf{w}}_i$ ， 定义样本集在低维空间中重构误差为：

   $$err' = \sum_{i=1}^{N}||\vec{\mathbf{z}}_i - \sum_{j\in\mathbb{Q}_i} w_{i,j}\vec{\mathbf{z}}_j||_2^2$$

   现在的问题是要求出 $\vec{\mathbf{z}}_i$ ， 从而使得上式最小。即求解：

   $$\min_{\vec{\mathbf{z}}_1,\vec{\mathbf{z}}_2,\cdots,\vec{\mathbf{z}}_N} \sum_{i=1}^{N}||\vec{\mathbf{z}}_i - \sum_{j\in\mathbb{Q}_i} w_{i,j}\vec{\mathbf{z}}_j||_2^2$$

2. 令 $\mathbf{Z} = (\vec{\mathbf{z}}_1^T, \vec{\mathbf{z}}_2^T, \cdots, \vec{\mathbf{z}}_N^T)^T \in \mathbb{R}^{N\times n'}$ ， 其中 n' 为低维空间的维数（ n 为原始样本所在的高维空间的维数）。令

   $$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,N} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N,1} & w_{N,2} & \cdots & w_{N,N} \end{bmatrix}$$

   定义 $\mathbf{M} = (\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$ ， 于是最优化问题可重写为： $\min_{\mathbf{Z}} tr(\mathbf{Z}^T\mathbf{M}\mathbf{Z})$ 。

   该最优化问题有无数个解。添加约束 $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_{n'\times n'}$ ， 于是最优化问题为：

   $$\min_{\mathbf{Z}} tr(\mathbf{Z}^T\mathbf{M}\mathbf{Z})$$
   $$s.t. \quad \mathbf{Z}^T\mathbf{Z} = \mathbf{I}_{n'\times n'}$$

   该最优化问题可以通过特征值分解求解：选取 $\mathbf{M}$ 最小的 n' 个特征值对应的特征向量组成的矩阵即为 $\mathbf{Z}$ 。

3. LLE 中出现了两个重构误差。
   - 第一个重构误差：为了在原始空间中求解线性重构的系数 $\vec{\mathbf{w}}_i$ 。目标是：基于 $\mathbb{Q}_i$ 中的样本点对 $\vec{\mathbf{x}}_i$ 进行线性重构，使得重构误差最小。
   - 第二个重构误差：为了求解样本集在低维空间中的表示 $\mathbf{Z}$ 。目标是：基于线性重构的系数 $\vec{\mathbf{w}}_i$ ， 将 $\mathbb{Q}_i$ 中的样本点对 $\vec{\mathbf{z}}_i$ 进行线性重构，使得重构误差最小。

4.3.2 LLE 算法

1. LLE 算法：

- 输入：
  - 样本集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_N\}$。
  - 近邻参数 k 。
  - 低维空间维数 n'。
- 输出：样本集在低维空间中的矩阵 $\mathbf{Z}$ 。
- 算法步骤：
  - 对于样本集中的每个点 $\vec{\mathbf{x}}_i,i=1,2,\cdots,N$，执行下列操作：
    - 确定 $\vec{\mathbf{x}}_i$ 的 k 近邻，获得其近邻下标集合 $\mathbb{Q}_i$ 。
    - 对于 $j\in\mathbb{Q}_i$， 根据下式计算 $w_{i,j}$：

$$w_{i,j}=\frac{\sum_{k\in Q_i}C^{-1}_{j,k}}{\sum_{l,s\in\mathbb{Q}_i}C^{-1}_{l,s}}$$
$$C_{j,k}=(\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j)^T(\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_k)$$

    - 对于 $j\notin\mathbb{Q}_i$， $w_{i,j}=0$
  - 根据 $w_{i,j}$ 构建矩阵 $\mathbf{W}$ 。
  - 计算 $\mathbf{M}=(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$ 。
  - 对 $\mathbf{M}$ 进行特征值分解，取其最小的 n' 个特征值对应的特征向量，即得到样本集在低维空间中的矩阵 $\mathbf{Z}$。

五、度量学习

1. 在机器学习中对高维数据进行降维的主要目的是：希望找出一个合适的低维空间，在这个低维空间中进行学习能比原始空间性能更好。

   每个空间对应了在样本属性上定义的一个距离度量。寻找合适的空间，本质上就是在寻找一个合适的距离度量。

2. 度量学习`metric learning`的思想就是：尝试直接学习出一个合适的距离度量。

3. 推广欧氏距离：对于两个 n 维样本 $\vec{\mathbf{x}}_i=(x_{i,1},x_{i,2},\cdots,x_{i,n})^T,\vec{\mathbf{x}}_j=(x_{j,1},x_{j,2},\cdots,x_{j,n})^T$ ，假定不同的属性的重要性不同，因此引入了权重：

$$dist^2_{ed}(\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j)=||\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j||_2^2=w_1d^2_{i,j,1}+w_2d^2_{i,j,2}+\cdots+w_nd^2_{i,j,n}$$

   其中 $d^2_{i,j,k}$ 表示 $\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j$ 在第 k 维上的距离， $w_k\geq 0$ 第 k 维距离的权重。

   定义对角矩阵 $\mathbf{W}$ 为：

$$\mathbf{W}=\begin{bmatrix}w_1&0&0&\cdots&0\\0&w_2&0&\cdots&0\\0&0&w_3&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&w_n\end{bmatrix}$$

   则 $dist^2_{ed}(\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j)=(\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j)^T\mathbf{W}(\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j)$。

   上式中的权重矩阵 $\mathbf{W}$ 可以通过学习确定。

4. 前述假设权重矩阵 $\mathbf{W}$ 是对角矩阵，这意味着坐标轴是正交的，即属性之间无关。

现实任务中可能会发生属性相关的情况，此时对应的坐标轴不再正交。于是可以将 $\mathbf{W}$ 替换成一个普通的半正定对称矩阵 $\mathbf{M}$，此时就得到了马氏距离 `Mahalanobis distance`：

$$dist_{mah}^2(\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j)=(\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j)^T\mathbf{M}(\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j)$$

其中的矩阵 $\mathbf{M}$ 也称作度量矩阵，度量学习就是对 $\mathbf{M}$ 进行学习。

为了保持距离非负而且对称，则 $\mathbf{M}$ 必须是半正定对称矩阵。即必有正交基 $\mathbf{P}$ ，使得 $\mathbf{M}=\mathbf{P}\mathbf{P}^T$ 。

5. 对 $\mathbf{M}$ 学习的目标是：将 $\mathbf{M}$ 嵌入到学习器的评价指标中去，通过优化学习器的评价指标来求得 $\mathbf{M}$ 。

   即：对 $\mathbf{M}$ 的学习无法直接提出优化目标，而是将 $\mathbf{M}$ 的学习与学习器的学习作为一个整体，然后优化学习器的优化目标。

6. 如果学习得的 $\mathbf{M}$ 是一个低秩矩阵（假设秩为 $r_M<n$），可以找到一组正交基，其中正交基的数量为 r_M ，该组正交基构成矩阵 $\mathbf{P}$ 。

   于是度量学习的结果可以衍生出一个降维矩阵 $\mathbf{P}$ ，能用于降维。降维后的低维空间就是该组正交基张成的空间。

六、概率PCA

1. 定义隐变量 $\vec{\mathbf{z}}\in\mathbb{R}^d$，它属于低维空间（也称作隐空间，即隐变量所在的空间）。假设 $\vec{\mathbf{z}}$ 的先验分布为高斯分布： $p(\vec{\mathbf{z}})=\mathcal{N}(\vec{\mathbf{0}},\mathbf{I})$ ，其均值为 $\vec{\mathbf{0}}$ ，协方差矩阵为 $\mathbf{I}$ 。

   定义观测变量 $\vec{\mathbf{x}}\in\mathbb{R}^n$ ，它属于高维空间。假设条件概率分布 $p(\vec{\mathbf{x}}\mid\vec{\mathbf{z}})$ 也是高斯分布：$p(\vec{\mathbf{x}}\mid\vec{\mathbf{z}})=\mathcal{N}(\mathbf{W}\vec{\mathbf{z}}+\vec{\mu},\sigma^2\mathbf{I})$ 。其中：均值是的 $\vec{\mathbf{z}}$ 线性函数，$\mathbf{W}\in\mathbb{R}^{n\times d}$ 为权重， $\vec{\mu}$ 为偏置；协方差矩阵为 $\sigma^2\mathbf{I}$ 。

   则 `PPCA` 模型生成观测样本的步骤为：

   - 首先以概率 $p(\vec{\mathbf{z}})$ 生成隐变量 $\vec{\mathbf{z}}$ 。
   - 然后观测样本 $\vec{\mathbf{x}}$ 由如下规则生成：$\vec{\mathbf{x}}=\mathbf{W}\vec{\mathbf{z}}+\vec{\mu}+\vec{\epsilon}$ 。

     其中 $\vec{\epsilon}$ 是一个 n 维的均值为零、协方差矩阵为 $\sigma^2\mathbf{I}$ 的高斯分布的噪声：$p(\vec{\epsilon})=\mathcal{N}(\vec{\mathbf{0}},\sigma^2\mathbf{I})$ 。

6.1 参数求解

6.1.1 解析解

1. 可以利用最大似然准则来确定参数 $\mathbf{W},\vec{\mu},\sigma^2$ 的解析解。
2. 根据边缘概率分布的定义有：

$$p(\vec{\mathbf{x}})=\int p(\vec{\mathbf{x}}\mid\vec{\mathbf{z}})d\vec{\mathbf{z}}$$

由于 $p(\vec{\mathbf{z}})$ 、$p(\vec{\mathbf{x}}\mid\vec{\mathbf{z}})$ 均为高斯分布，因此 $p(\vec{\mathbf{x}})$ 也是高斯分布。假 $\vec{\mathbf{x}}$ 的其均值为 $\vec{\mu}'$，协方差为 $\mathbf{C}$ 。则：

$$\vec{\mu}'=\mathbb{E}[\vec{\mathbf{x}}]=\mathbb{E}[\mathbf{W}\vec{\mathbf{z}}+\vec{\mu}+\vec{\epsilon}]=\vec{\mu}$$

$$\mathbf{C}=cov[\vec{\mathbf{x}}]=\mathbb{E}[(\mathbf{W}\vec{\mathbf{z}}+\vec{\mu}+\vec{\epsilon})(\mathbf{W}\vec{\mathbf{z}}+\vec{\mu}+\vec{\epsilon})^T]$$

$$=\mathbb{E}[\mathbf{W}\vec{\mathbf{z}}\vec{\mathbf{z}}^T\mathbf{W}]+\mathbb{E}[\vec{\epsilon}\vec{\epsilon}^T]+\vec{\mu}\vec{\mu}^T=\mathbf{W}\mathbf{W}^T+\sigma^2\mathbf{I}+\vec{\mu}\vec{\mu}^T$$

> 推导过程中假设 $\vec{\mathbf{z}}$ 和 $\vec{\epsilon}$ 是相互独立的随机变量。

因此 $p(\vec{\mathbf{x}})=\mathcal{N}(\vec{\mu},\mathbf{C})$ 。

3. 给定数据集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$ ，则对数似然函数为：

$$\mathcal{L}=\log p(\mathbb{D};\mathbf{W},\vec{\mu},\sigma^2)=\sum_{i=1}^{N}\log p(\vec{\mathbf{x}}_i;\mathbf{W},\vec{\mu},\sigma^2)$$
$$=-\frac{Nn}{2}\log(2\pi)-\frac{N}{2}\log|\mathbf{C}|-\frac{1}{2}\sum_{i=1}^{N}(\vec{\mathbf{x}}_i-\vec{\mu})^T\mathbf{C}^{-1}(\vec{\mathbf{x}}_i-\vec{\mu})$$

其中 $|\cdot|$ 这里表示行列式的值。

求解 $\frac{\partial\mathcal{L}}{\partial\vec{\mu}}=\vec{0}$，解得 $\vec{\mu}=\bar{\vec{\mathbf{x}}}=\frac{1}{N}\sum_{i=1}^{N}\vec{\mathbf{x}}_i$ 。

4. 对数据集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$ 进行零均值化，即：$\vec{\mathbf{x}}_i\leftarrow\vec{\mathbf{x}}_i-\vec{\mu}=\vec{\mathbf{x}}_i-\bar{\vec{\mathbf{x}}}$ 。则有：
   - $\vec{\mathbf{x}}=\mathbf{W}\vec{\mathbf{z}}+\vec{\epsilon}$ ，因此 $p(\vec{\mathbf{x}})=\mathcal{N}(\vec{\mathbf{x}};\vec{\mathbf{0}},\mathbf{C})$ 。

     其中 $\mathbf{C}=\mathbb{E}[(\mathbf{W}\vec{\mathbf{z}}+\vec{\epsilon})(\mathbf{W}\vec{\mathbf{z}}+\vec{\epsilon})^T]=\mathbf{W}\mathbf{W}^T+\sigma^2\mathbf{I}$ 。
   - 对数似然函数（忽略常数项 $-\frac{Nn}{2}\log(2\pi)$ ）：

     $$\mathcal{L}=\log p(\mathbb{D};\mathbf{W},\vec{\mu},\sigma^2)=-\frac{N}{2}\log|\mathbf{C}|-\frac{1}{2}\sum_{i=1}^{N}\vec{\mathbf{x}}_i^T\mathbf{C}^{-1}\vec{\mathbf{x}}_i=-\frac{N}{2}\left[\log|\mathbf{C}|+tr(\mathbf{C}^{-1}\mathbf{S})\right]$$

     其中 $\mathbf{S}=\frac{1}{N}\sum_{i=1}^{N}\vec{\mathbf{x}}_i\vec{\mathbf{x}}_i^T$ 为协方差矩阵。 记：

     $$\mathbf{X}=(\vec{\mathbf{x}}_1^T,\cdots,\vec{\mathbf{x}}_N^T)^T=\begin{bmatrix}\vec{\mathbf{x}}_1^T\\ \vdots\\ \vec{\mathbf{x}}_N^T\end{bmatrix}=\begin{bmatrix}x_{1,1}&x_{1,2}&\cdots&x_{1,n}\\ x_{2,1}&x_{2,2}&\cdots&x_{2,n}\\ \vdots&\vdots&\ddots&\vdots\\ x_{N,1}&x_{N,2}&\cdots&x_{N,n}\end{bmatrix}$$

     则 $\mathbf{S}=\mathbf{X}^T\mathbf{X}$ 。
5. `Tipping and Bishop(1999b)` 证明：
   - $\mathcal{L}$ 的所有驻点都可以写做：$\mathbf{W}=\mathbf{U}_d(\mathbf{\Lambda}_d-\sigma^2\mathbf{I})^{1/2}\mathbf{R}$ 。其中：
     - $\mathbf{U}_d\in\mathbb{R}^{n\times d}$ 的列由协方差矩阵 $\mathbf{S}$ 的任意 d 个特征向量组成。
     - $\mathbf{\Lambda}_d\in\mathbb{R}^{d\times d}$ 是对角矩阵，其元素是协方差矩阵 $\mathbf{S}$ 对应的 d 个特征值 λ_i 。
     - $\mathbf{R}\in\mathbb{R}^{d\times d}$ 是任意一个正交矩阵。
   - 当 d 个特征向量被选择为前 d 个最大的特征值对应的特征向量时，$\mathcal{L}$ 取得最大值。其它的所有解都是鞍点。
6. 假定协方差矩阵 $\mathbf{S}$ 的特征值从大到小排列 $\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_n$ ，对应的 n 个特征向量为 $\vec{\mathbf{u}}_1,\cdots,\vec{\mathbf{u}}_n$ 。

   则最大似然准则得到的解析解为：
   - $\mathbf{U}=(\vec{\mathbf{u}}_1,\cdots,\vec{\mathbf{u}}_d)$ ，它由前 d 个特征向量组成。$\mathbf{W}=\mathbf{U}_d(\mathbf{\Lambda}_d-\sigma^2\mathbf{I})^{1/2}\mathbf{R}$ 。
   - $\sigma^2=\frac{1}{n-d}\sum_{i=d+1}^{n}\lambda_i$ ，它就是与丢弃的维度相关连的平均方差。
7. $\mathbf{R}$ 是正交矩阵，因此它可以视作 d 维隐空间的一个旋转矩阵。

   根据 $\mathbf{C}=\mathbf{W}\mathbf{W}^T+\sigma^2\mathbf{I}=\mathbf{U}_d(\mathbf{\Lambda}_d-\sigma^2\mathbf{I})\mathbf{U}_d^T+\sigma^2\mathbf{I}$，则 $\mathbf{C}$ 与 $\mathbf{R}$ 无关。这表明： $p(\vec{\mathbf{x}})$ 在隐空间中具有旋转不变性，因此 $\mathbf{R}$ 可以选任意一个正交矩阵。
   - 这代表某种形式的统计不可区分性，因此有多个 $\mathbf{W}$ 都会产生同样的密度函数 $p(\vec{\mathbf{x}})$ 。
   - 当 $\mathbf{R}=\mathbf{I}$ 时，$\mathbf{W}=\mathbf{U}_d(\mathbf{\Lambda}_d-\sigma^2\mathbf{I})^{1/2}$ 。此时 $\mathbf{W}=(\sqrt{\lambda_1-\sigma^2}\vec{\mathbf{u}}_1,\cdots,\sqrt{\lambda_d-\sigma^2}\vec{\mathbf{u}}_d)$ ，它表示 $\mathbf{W}$ 的列是对 $\vec{\mathbf{u}}_1,\cdots,\vec{\mathbf{u}}_d$ 进行缩放，缩放比例为 $\sqrt{\lambda_i-\sigma^2}$ 。

     由于 $\vec{\mathbf{u}}_i,\vec{\mathbf{u}}_j,i\neq j$ 是正交的，因此 $\mathbf{W}$ 的列也是正交的。

- 当通过解析解直接求解时，可以直接令 $\mathbf{R}=\mathbf{I}$ 。但是当通过共轭梯度法或者 EM 算法求解时， $\mathbf{R}$ 可能是任意的，此时 $\mathbf{W}$ 的列不再是正交的。

  如果需要 $\mathbf{W}$ 是正交的，则需要恰当的后处理。

8. 对于任意一个方向向量 $\vec{\mathbf{v}}$ ，其中 $\vec{\mathbf{v}}^T\vec{\mathbf{v}}=1$ ，分布 $p(\vec{\mathbf{x}})$ 在该方向上的方差为 $\vec{\mathbf{v}}^T\mathbf{C}\vec{\mathbf{v}}$ 。
   - 如果 $\vec{\mathbf{v}}$ 与 $\{\vec{\mathbf{u}}_1,\cdots,\vec{\mathbf{u}}_d\}$ 正交，即 $\vec{\mathbf{v}}$ 是被丢弃的特征向量的某个线性组合，则有 $\vec{\mathbf{v}}^T\mathbf{U}_d=\vec{\mathbf{0}}$ 。

     因此有 $\vec{\mathbf{v}}^T\mathbf{C}\vec{\mathbf{v}}=\sigma^2$ 。
   - 如果 $\vec{\mathbf{v}}$ 就是 $\{\vec{\mathbf{u}}_1,\cdots,\vec{\mathbf{u}}_d\}$ 其中之一，即 $\vec{\mathbf{v}}=\vec{\mathbf{u}}_i,1\leq i\leq d$，则有 $\vec{\mathbf{v}}^T\mathbf{C}\vec{\mathbf{v}}=(\lambda_i-\sigma^2)+\sigma^2=\lambda_i$ 。

     可以看到：沿着特征向量 $\vec{\mathbf{u}}_i$ 方向上的方差 λ_i 由两部分组成：
     - 单位方差的分布 $p(\vec{\mathbf{z}})$ 通过线性映射 $\mathbf{W}$ 之后，在 $\vec{\mathbf{u}}_i$ 方向上的投影贡献了：$\lambda_i-\sigma^2$ 。
     - 噪声模型在所有方向上的各向同性的方差贡献了： σ^2 。

   因此： PPCA 正确的描述了数据集 $\mathbb{D}$ 沿着主轴方向（即 $\vec{\mathbf{u}}_1,\cdots,\vec{\mathbf{u}}_d$ 方向）的方差，并且用一个单一的均值 σ^2 近似了所有剩余方向上的方差。
9. 当 $d=n$ 时，不存在降维的过程。此时有 $\mathbf{U}_d=\mathbf{U}$，$\mathbf{\Lambda}_d=\mathbf{\Lambda}$ 。根据正交矩阵的性质：$\mathbf{U}\mathbf{U}^T=\mathbf{I}$ ，以及 $\mathbf{R}\mathbf{R}^T$ ，则有：$\mathbf{C}=\mathbf{U}(\mathbf{\Lambda}-\sigma^2\mathbf{I})\mathbf{U}^T+\sigma^2\mathbf{I}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T=\mathbf{S}$ 。
10. 由于计算时需要用到 $\mathbf{C}^{-1}$，这涉及到一个 $n\times n$ 矩阵的求逆。

    可以考虑简化为：$\mathbf{C}^{-1}=\sigma^{-2}\mathbf{I}-\sigma^{-2}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^T$ ，其中 $\mathbf{M}=\mathbf{W}^T\mathbf{W}+\sigma^2\mathbf{I}\in\mathbb{R}^{d\times d}$ 。计算复杂度从 $O(n^3)$ 降低到了 $O(d^3)$ 。

6.1.2 EM算法解

1. 在 PPCA 模型中，数据集 $\mathbb{D}$ 中的每个数据点 $\vec{\mathbf{x}}_i$ 都对应一个隐变量 $\vec{\mathbf{z}}_i$ ，于是可以使用 EM 算法来求解模型参数。
2. 实际上 PPCA 模型参数已经可以得到精确的解析解，看起来没必要使用 EM 算法。但是 EM 算法具有下列优势：
   - 在高维空间中，使用 EM 算法而不是直接计算样本的协方差矩阵可能具有计算上的优势。
   - EM 算法的求解步骤也可以推广到因子分析模型中，那里不存在解析解。
   - EM 算法可以为缺失值处理提供理论支撑。
3. 观测变量为 $\vec{\mathbf{x}}$，隐变量为 $\vec{\mathbf{z}}$ ，则完全数据为 $\{\mathbb{D},\mathbb{Z}\}$，其中 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$ 、 $\mathbb{Z}=\{\vec{\mathbf{z}}_1,\cdots,\vec{\mathbf{z}}_N\}$ 。其中 对数据集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$ 进行零均值化，即：$\vec{\mathbf{x}}_i\leftarrow\vec{\mathbf{x}}_i-\vec{\mu}=\vec{\mathbf{x}}_i-\bar{\vec{\mathbf{x}}}$ 。

   根据后验概率的定义以及高斯分布的性质，后验概率 $p(\vec{\mathbf{z}}\mid\vec{\mathbf{x}})=\mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^T\vec{\mathbf{x}},\sigma^2\mathbf{M}^{-1})$ 。

   完全数据的对数似然函数为：

$$\log p(\mathbb{D},\mathbb{Z};\mathbf{W},\sigma^2)=\sum_{i=1}^N\log p(\vec{\mathbf{x}}_i,\vec{\mathbf{z}}_i)=\sum_{i=1}^N[\log p(\vec{\mathbf{x}}_i\mid\vec{\mathbf{z}}_i)+\log p(\vec{\mathbf{z}}_i)]$$

   其中： $p(\vec{\mathbf{x}}\mid\vec{\mathbf{z}})=\mathcal{N}(\mathbf{W}\vec{\mathbf{z}},\sigma^2\mathbf{I})$ 、 $p(\vec{\mathbf{z}})=\mathcal{N}(\vec{\mathbf{0}},\mathbf{I})$
   - E 步：计算期望：

$$\begin{aligned}\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}\log p(\mathbb{D},\mathbb{Z};\mathbf{W},\sigma^2)=-\sum_{i=1}^N\Big[&\frac{n}{2}\log(2\pi\sigma^2)+\frac{1}{2}tr(\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i\vec{\mathbf{z}}_i^T])\\&+\frac{1}{2\sigma^2}||\vec{\mathbf{z}}_i||_2^2-\frac{1}{\sigma^2}\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i]^T\mathbf{W}^T\vec{\mathbf{x}}_i\\&+\frac{1}{2\sigma^2}tr(\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i\vec{\mathbf{z}}_i^T]\mathbf{W}^T\mathbf{W})+\frac{d}{2}\log(2\pi)\Big]\end{aligned}$$

其中：$\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}$ 表示计算期望的概率分布为后验概率分布 $p(\vec{\mathbf{z}}\mid\vec{\mathbf{x}})$ 。 假设此时迭代的参数为 $\mathbf{W}_{old}$、σ^2_{old} ，则有：

$$\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i]=\mathbf{M}_{old}^{-1}\mathbf{W}_{old}^T\vec{\mathbf{x}}_i$$
$$\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i\vec{\mathbf{z}}_i^T]=cov_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i]+\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i]\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i]^T$$
$$=\sigma^2_{old}\mathbf{M}_{old}^{-1}+(\mathbf{M}_{old}^{-1}\mathbf{W}_{old}^T\vec{\mathbf{x}}_i)(\mathbf{M}_{old}^{-1}\mathbf{W}_{old}^T\vec{\mathbf{x}}_i)^T$$

- `M` 步：求最大化：

$$\mathbf{W}_{new},\sigma^2_{new}=\arg\max_{\mathbf{W},\sigma^2}\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}\log p(\mathbb{D},\mathbb{Z};\mathbf{W},\sigma^2)$$

解得：

- $\mathbf{W}_{new}\leftarrow\left[\sum_{i=1}^N\vec{\mathbf{x}}_i\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i]^T\right]\left[\sum_{i=1}^N\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i\vec{\mathbf{z}}_i^T]\right]^{-1}$ 。
- $\sigma^2_{new}\leftarrow\frac{1}{Nn}\sum_{i=1}^N\left[||\vec{\mathbf{x}}_i||^2-2\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i]^T\mathbf{W}_{new}^T\vec{\mathbf{x}}_i+tr(\mathbb{E}_{p(\vec{\mathbf{z}}|\vec{\mathbf{x}})}[\vec{\mathbf{z}}_i\vec{\mathbf{z}}_i^T])\mathbf{W}_{new}^T\mathbf{W}_{new}\right]$

4. `EM` 算法的物理意义：
   - `E` 步涉及到数据点 $\vec{\mathbf{x}}_i$ 在隐空间上的正交投影。
   - `M` 步涉及到隐空间的重新估计，使得满足最大似然，其中投影固定。

   一个简单的类比：考虑二维空间中的一组小球，令一维隐空间用一个固定的杆表示。现在使用很多个遵守胡克定律（存储的能量正比于弹簧长度的平方）的弹簧将每个小球与杆相连。
   - `E` 步：保持杆固定，让附着的小球沿着杆上下滑动，使得能量最小。这使得每个小球独立地到达对应于它在杆上的正交投影位置。
   - `M` 步：令附着点固定，然后松开杆，让杆到达能量最小的位置。

   重复 `E` 步和 `M` 步，直到满足一个收敛准则。
5. `PPCA` 的 `EM` 算法的一个好处是大规模数据的计算效率。在高维空间中， `EM` 算法每次迭代所需的计算量都比传统的 `PCA` 要小得多。
   - `PPCA` 解析解算法中，对协方差矩阵进行特征分解的计算复杂度为 $O(n^3)$ 。如果只需要计算前 d 个特征向量和它们的特征值，则可以使用 $O(dn^2)$ 复杂度的算法。

     然后计算协方差矩阵本身需要 $O(Nn^2)$ 的计算量，因此不适合大规模数据。
   - `PPCA` 的 `EM` 算法没有显式建立协方差矩阵，其计算量最大的步骤是涉及到对数据集求和的操作，计算代价为 $O(Nnd)$ 。

     对于较大的 n，有 $d\ll n$，因此与 $O(Nn^2)$ 相比， `EM` 算法的计算量大大降低。这可以抵消 `EM` 算法需要多次迭代的缺陷。
6. `PPCA` 的 `EM` 算法可以用一种在线的形式执行，其中 $\vec{\mathbf{x}}_i$ 被读入、处理。然后在处理下一个数据 $\vec{\mathbf{x}}_j$ 时丢弃 $\vec{\mathbf{x}}_i$
   - `E` 步中需要计算的量（一个 `d` 维向量和一个 $d\times d$ 的矩阵）可以分别对每个数据点单独计算。
   - `M` 步中需要在数据点上累积求和，这个可以增量完成。

   如果 N 和 n 都很大，则这种方式很有优势。

6.2 性质

1. 概率 `PCA` （ `probabilistic PCA:PPCA` ）与传统的 `PCA` 相比，有如下优势：
   - 概率 `PCA` 能够描述数据集的主要特征，如期望、方差等。

- 概率`PCA`使用`EM`算法求解。当只需要计算几个主要的特征向量的情况下，计算效率较高，它避免了计算协方差矩阵 $\mathbf{X}^T\mathbf{X}$ 。
- 概率`PCA`可以推广到混合概率分布，并且利用`EM`算法训练。
- 概率`PCA`采用似然函数，这使得它可以与其它的概率密度模型进行比较。
- 概率`PCA`是一个生成式模型，可以用于生成新样本。

2. `PPCA`考虑的是低维空间到高维空间的映射，这与传统的`PCA`观点不同。传统`PCA`观点是高维空间到低维空间的映射。

   在`PPCA`中，如果希望对数据进行降维，即从个高维空间映射到低维空间，则需要通过贝叶斯定理进行逆映射。

   根据后验概率分布 $p(\vec{\mathbf{z}}\mid\vec{\mathbf{x}})=\mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^T\vec{\mathbf{x}},\sigma^2\mathbf{M}^{-1})$ ，任意一点 $\vec{\mathbf{x}}$ 在低维空间中的投影均值为 $\mathbb{E}[\vec{\mathbf{z}}\mid\vec{\mathbf{x}}]=\mathbf{M}^{-1}\mathbf{W}^T\vec{\mathbf{x}}$ 。

   - 如果取极限 $\sigma^2\to 0$ ，则 $\mathbb{E}[\vec{\mathbf{z}}\mid\vec{\mathbf{x}}]\to(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\vec{\mathbf{x}}$ ，这就是标准的`PCA`模型。但此时后验协方差为0，概率密度函数变得奇异。

     但是如果使用`EM`算法求解，仍然可以得到一个合法的结果。

   - 对于 $\sigma^2>0$ 的情况，低维空间中的投影会向着原点方向偏移。

3. 目前在`PCA`讨论中，假定低维空间的维度 d 是给定的，实际上必须根据应用来选择一个合适的值。

   - 如果用于数据可视化，则一般选择为 $d=2$ 。
   - 如果特征值很自然的分成了两组：一组由很小的值组成，另一组由较大的值组成，两组之间有明显的区分。则 d 选取为较大的特征值的数量。

     > 实际上这种明显的区分通常无法看到。

   - 按照传统`PCA`的方式：
     - 通过交叉验证法选取较好的 d ，这种方式依赖于后续的模型。
     - 从算法原理的角度设置一个阈值，比如 $t=95\%$ ，然后选取使得下式成立的最小的 d 的值：

       $$\frac{\sum_{i=1}^d\lambda_i}{\sum_{i=1}^n\lambda_i}\geq t$$

       这种方式需要指定阈值 t ，从而将 d 的选择转移到 t 的选择上。
   - 基于`PPCA`中的最大似然函数，使用交叉验证的方法，求解在验证集上对数似然函数最大的模型来确定维度的值。

     这种方法不依赖于后续的模型，但是缺点是计算量很大。

   - 利用贝叶斯`PCA`自动寻找合适的d 。

4. 贝叶斯`PCA`：在$\mathbf{W}$ 的每列上定义一个独立的高斯先验，每个这样的高斯分布都有一个独立的方差，由超参数 α_i 控制。因此：

   $$p(\mathbf{W};\vec{\alpha})=\prod_{i=1}^d\left(\frac{\alpha_i}{2\pi}\right)^{n/2}\exp\left(-\frac{1}{2}\alpha_i\vec{\mathbf{w}}_i^T\vec{\mathbf{w}}_i\right)$$

   其中 $\vec{\mathbf{w}}_i$ 是 $\mathbf{W}$ 的第 i 列。

   $\vec{\alpha}$ 可以通过最大化 $\log p(\mathbb{D})$ 来迭代的求解。最终某个 α_i 可能趋向于无穷大，对应的参数向量 $\vec{\mathbf{w}}_i$ 趋向于零，后验概率分布变成了原点处的 $\delta(\cdot)$ 函数。这就得到一个稀疏解。

这样低维空间的有效维度由有限的 α_i 的值确定。通过这种方式，贝叶斯方法自动的在提升数据拟合程度（使用较多的向量 $\vec{\mathbf{w}}_i$ ）和减小模型复杂度（压制某些向量 $\vec{\mathbf{w}}_i$ ）之间折中。

6.3 因子分析

1. 因子分析：寻找变量之间的公共因子。

   如：随着年龄的增长，儿童的身高、体重会发生变化，具有一定的相关性。假设存在一个生长因子同时支配这两个变量，那么因子分析就是从大量的身高、体重数据中寻找该生长因子。

2. 因子分析 `Factor Analysis:FA` 是一个线性高斯隐变量模型，它与 `PPCA` 密切相关。

   - 因子分析的定义与 `PPCA` 唯一差别是：给定隐变量 $\vec{\mathbf{z}}$ 的条件下，观测变量 $\vec{\mathbf{x}}$ 的条件概率分布的协方差矩阵是一个对角矩阵，而不是一个各向同性的协方差矩阵。

     即：$p(\vec{\mathbf{x}}\mid\vec{\mathbf{z}})=\mathcal{N}(\mathbf{W}\vec{\mathbf{z}}+\vec{\mu},\mathbf{\Psi})$ ，其中 $\mathbf{\Psi}$ 是一个 $n\times n$ 的对角矩阵。因此也可以认为 `PPCA` 是一种特殊情形的因子分析。

     如果对 $\vec{\mathbf{x}}$ 进行了零均值化，则 $p(\vec{\mathbf{x}}\mid\vec{\mathbf{z}})=\mathcal{N}(\mathbf{W}\vec{\mathbf{z}},\mathbf{\Psi})$ 。

   - 与 `PPCA` 模型相同，因子分析模型假设在给定隐变量 $\vec{\mathbf{z}}$ 的条件下，观测变量 $\vec{\mathbf{x}}$ 的各分量 $x_1,x_2,\cdots,x_n$ 是独立的。

3. 在因子分析的文献中， $\mathbf{W}$ 的列描述了观测变量之间的相关性关系，被称作因子载入 `factor loading` 。

   $\mathbf{\Psi}$ 的对角元素，表示每个变量的独立噪声方差，被称作唯一性 `uniqueness` 。

4. 观测变量的边缘概率分布为 $p(\vec{\mathbf{x}})=\mathcal{N}(\vec{\mathbf{0}},\mathbf{C})$，其中 $\mathbf{C}=\mathbf{W}\mathbf{W}^T+\mathbf{\Psi}$ 。

   与 `PPCA` 相同， `FA` 模型对于隐空间的选择具有旋转不变性。

5. 可以使用最大似然法来确定因子分析模型中的参数 $\mathbf{W}$、$\mathbf{\Psi}$ 的值。此时 $\mathbf{W},\mathbf{\Psi}$ 的最大似然解不再具有解析解，因此必须用梯度下降法或者 `EM` 算法迭代求解。

   `EM` 算法的迭代步骤为：

   - `E` 步：用旧的参数求期望：

     $$\mathbb{E}[\vec{\mathbf{z}}_i]=\mathbf{G}\mathbf{W}^T\mathbf{\Psi}^{-1}\vec{\mathbf{x}}_i$$
     $$\mathbb{E}[\vec{\mathbf{z}}_i\vec{\mathbf{z}}_i^T]=cov[\vec{\mathbf{z}}_i]+\mathbb{E}[\vec{\mathbf{z}}_i]\mathbb{E}[\vec{\mathbf{z}}_i]^T=\mathbf{G}+\mathbb{E}[\vec{\mathbf{z}}_i]\mathbb{E}[\vec{\mathbf{z}}_i]^T$$

     其中 $\mathbf{G}=(\mathbf{I}+\mathbf{W}^T\mathbf{\Psi}^{-1}\mathbf{W})^{-1}$ 。

     这里使用一个 $d\times d$ 的矩阵求逆表达式，而不是 $n\times n$ 的表达式。

   - `M` 步：求最大化来获取新的参数。

     $$\mathbf{W}_{new}\leftarrow\left[\sum_{i=1}^N\vec{\mathbf{x}}_i\mathbb{E}[\vec{\mathbf{z}}_i]^T\right]\left[\sum_{i=1}^N\mathbb{E}[\vec{\mathbf{z}}_i\vec{\mathbf{z}}_i^T]\right]^{-1}$$
     $$\mathbf{\Psi}_{new}\leftarrow\text{diag}\left[\mathbf{S}-\mathbf{W}_{new}\frac{1}{N}\sum_{i=1}^N\mathbb{E}[\vec{\mathbf{z}}_i]\vec{\mathbf{x}}_i^T\right]$$

     其中 diag 将所有非对角线上的元素全部设置为零。

七、独立成分分析

1. 独立成分分析 `ICA` 用于从混合信号中分离出原始信号。

   本质上它并不是一个降维的算法，而是一个信号分离算法。

7.1 鸡尾酒会问题

1. 假设酒会上有 n 个人，他们可以同时说话。房间里散落了 n 个声音接收器用于记录声音。酒会过后，从 n 个声音接收器中采集到一组数据：

$$\mathbb{D}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_N\}$$
$$\vec{\mathbf{x}}_i=(x_{i,1},x_{i,2},\cdots,x_{i,n})^T$$

   任务的目标是：从这 N 个时刻的采样数据中恢复出每个人说话的信号。这个过程也称作盲信号分离。

   随机变量 $\vec{\mathbf{x}}$ 表示观测随机变量，$\vec{\mathbf{x}}_i$ 是其第 i 个采样值，其物理意义为：在时刻 i 采集到的 n 个声音信号。

2. 定义：
   - 第 i 个人说话的信号为 s_i 。它是一个随机变量，其分布为 $p_s(s_i)$ 。$s_{1,1},\cdots,s_{N,1}$ 为 s_i 的 N 个时刻的采样，记作 $\vec{\mathbf{u}}_i^{(s)}$。
   - n 个人说话的信号为 $\vec{\mathbf{s}}=(s_1,s_2,\cdots,s_n)^T$ 。它是一个 n 维随机变量，分布为 $p_s(\vec{\mathbf{s}})$ 。$\vec{\mathbf{s}}_1,\cdots,\vec{\mathbf{s}}_N$ 为 $\vec{\mathbf{s}}$ 的 N 个时刻的采样。
   - 第 i 个声音接收器收到的信号为 x_i 。它是一个随机变量，其分布为 $p_x(x_i)$ 。$x_{1,1},\cdots,x_{N,1}$ 为 x_i 的 N 个时刻的采样，记作 $\vec{\mathbf{u}}_i^{(x)}$。
   - n 个声音接收器收到的信号为 $\vec{\mathbf{x}}=(x_1,x_2,\cdots,x_n)^T$ 。它是一个 n 维随机变量，分布为 $p_x(\vec{\mathbf{x}})$ 。$\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N$ 为 $\vec{\mathbf{x}}$ 的 N 个时刻的采样。
   - 定义矩阵 $\mathbf{X}$ 和矩阵 $\mathbf{S}$ 为：

$$\mathbf{X}=\begin{bmatrix}\vec{\mathbf{x}}_1^T\\ \vdots\\ \vec{\mathbf{x}}_N^T\end{bmatrix}=\begin{bmatrix}x_{1,1}&x_{1,2}&\cdots&x_{1,n}\\ x_{2,1}&x_{2,2}&\cdots&x_{2,n}\\ \vdots&\vdots&\ddots&\vdots\\ x_{N,1}&x_{N,2}&\cdots&x_{N,n}\end{bmatrix}\quad\mathbf{S}=\begin{bmatrix}\vec{\mathbf{s}}_1^T\\ \vdots\\ \vec{\mathbf{s}}_N^T\end{bmatrix}=\begin{bmatrix}s_{1,1}&s_{1,2}&\cdots&s_{1,n}\\ s_{2,1}&s_{2,2}&\cdots&s_{2,n}\\ \vdots&\vdots&\ddots&\vdots\\ s_{N,1}&s_{N,2}&\cdots&s_{N,n}\end{bmatrix}$$

     其意义为：
     - $\mathbf{X}$ 的每一行代表 $\vec{\mathbf{x}}$ 在时刻 i 的采样 $\vec{\mathbf{x}}_i$；每一列代表信号 x_j 在所有时刻的采样序列 $\vec{\mathbf{u}}_j^{(x)}$ 。
     - $\mathbf{S}$ 的每一行代表 $\vec{\mathbf{s}}$ 在时刻 i 的采样 $\vec{\mathbf{s}}_i$；每一列代表信号 s_j 在所有时刻的采样序列 $\vec{\mathbf{u}}_j^{(s)}$ 。
3. $\mathbf{A}=(a_{i,j})_{n\times n}$ 是一个未知的混合矩阵，它用于叠加 n 个人说话的信号。则有： $\mathbf{X}=\mathbf{S}\mathbf{A}^T$ 。即：$\vec{\mathbf{x}}=\mathbf{A}\vec{\mathbf{s}}$ 。

   其物理意义为：每个声音接收器采集的信号是所有人说话信号的线性叠加。

7.2 算法

1. 现在 $\mathbf{X}$ 是已知的，即信号 $\vec{\mathbf{x}}$ 是已知的。令 $\mathbf{W}=\mathbf{A}^{-1}$ ，则有：$\vec{\mathbf{s}}=\mathbf{W}\vec{\mathbf{x}}$ 。 $\mathbf{W}$ 称作分离矩阵。

   如果没有任何先验知识，则无法同时确定信号 $\vec{\mathbf{s}}$ 和 $\mathbf{W}$ 。
   - 当 $\mathbf{W}$ 的每个元素扩大 2 倍，同时信号 $\vec{\mathbf{s}}$ 放大2倍时，等式仍然成立。因此结果不是唯一的。
   - 当调整信号 $\vec{\mathbf{s}}$ 中各子信号的顺序，同时调整 $\mathbf{W}$ 中各行的顺序，等式也仍然成立。因此结果不是唯一的。
   - 信号 $\vec{\mathbf{s}}$ 不能是多维高斯分布。

     假设 $\vec{\mathbf{s}}$ 是多维高斯分布： $p(\vec{\mathbf{s}})=\mathcal{N}(\vec{\mathbf{0}},\mathbf{I})$ 。则 $\vec{\mathbf{x}}$ 也是一个多维高斯分布，均值为 $\vec{\mathbf{0}}$ ，方差为 $\mathbb{E}[\vec{\mathbf{x}}\vec{\mathbf{x}}^T]=\mathbf{A}\mathbf{A}^T$ 。

假设 $\mathbf{R}$ 为任意一个正交矩阵，令 $\mathbf{A}'=\mathbf{AR}$，则有：$\mathbf{A}'\mathbf{A}'^T=\mathbf{ARR}^T\mathbf{A}^T=\mathbf{AA}^T$ 。

这表示在给定信号 $\vec{\mathbf{s}}$ 的分布和 $\vec{\mathbf{x}}$ 的分布的情况下，参数 $\mathbf{A}$ 的值并不是唯一的，因此无法分离出每个人说话的信号 s_i 。

2. 假设每个人发出的声音信号 s_i 相互独立，则 $\vec{\mathbf{s}}$ 的概率分布为：$p_s(\vec{\mathbf{s}})=\prod_{i=1}^n p_s(s_i)$ 。

   根据 $\vec{\mathbf{s}}=\mathbf{W}\vec{\mathbf{x}}$，有：$p_x(\vec{\mathbf{x}})=p_s(\mathbf{W}\vec{\mathbf{x}})|\mathbf{W}|$ 。其中 $|\cdot|$ 为行列式。

   记：

$$\mathbf{W}=\begin{bmatrix}w_{1,1}&w_{1,2}&\cdots&w_{1,n}\\w_{2,1}&w_{2,2}&\cdots&w_{2,n}\\\vdots&\vdots&\ddots&\vdots\\w_{n,1}&w_{n,2}&\cdots&w_{n,n}\end{bmatrix}$$

   令 $\vec{\mathbf{w}}_i=(w_{i,1},w_{i,2},\cdots,w_{i,n})^T$， 即它是由 $\mathbf{W}$ 的第 i 行组成。则有：

$$p_s(\mathbf{W}\vec{\mathbf{x}})=p_s(\vec{\mathbf{w}}_1^T\vec{\mathbf{x}},\cdots,\vec{\mathbf{w}}_n^T\vec{\mathbf{x}})=\prod_{i=1}^n p_s(\vec{\mathbf{w}}_i^T\vec{\mathbf{x}})$$

   因此有：$p_x(\vec{\mathbf{x}})=|\mathbf{W}|\prod_{i=1}^n p_s(\vec{\mathbf{w}}_i^T\vec{\mathbf{x}})$ 。

3. 前面提到如果没有任何先验知识，则无法求解。这里需要假设 $p_s(s_i)$ 。

   - 首先，不能选取高斯分布。
   - 其次，考虑到概率密度函数由累计分布函数求导得到，一个方便的选择是：选择累计分布函数为 `sigmoid` 函数：

$$g(s)=\frac{1}{1+e^{-s}}$$

     则概率密度函数为：

$$p_s(s)=g'(s)=\frac{e^s}{(1+e^s)^2}$$

4. 给定采样样本集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$，则对数似然函数为：

$$\mathcal{L}=\sum_{i=1}^N\log p_x(\vec{\mathbf{x}}_i)=\sum_{i=1}^N\left(\log|\mathbf{W}|+\sum_{j=1}^n\log p_s(\vec{\mathbf{w}}_j^T\vec{\mathbf{x}}_i)\right)$$

   - 根据最大似然准则，可以采用梯度下降法求解 $\mathcal{L}$ 的最大值。

     其中：根据矩阵微积分有：$\nabla_{\mathbf{W}}|\mathbf{W}|=|\mathbf{W}|(\mathbf{W}^{-1})^T$ 。则有：

$$\nabla_{\mathbf{W}}\mathcal{L}=\begin{bmatrix}1-2g(\vec{\mathbf{w}}_1^T\vec{\mathbf{x}}_i)\\1-2g(\vec{\mathbf{w}}_2^T\vec{\mathbf{x}}_i)\\\vdots\\1-2g(\vec{\mathbf{w}}_n^T\vec{\mathbf{x}}_i)\end{bmatrix}\vec{\mathbf{x}}_i^T+(\mathbf{W}^{-1})^T$$

   - 当迭代求解出 $\mathbf{W}$ 之后，通过 $\vec{\mathbf{s}}=\mathbf{W}\vec{\mathbf{x}}$ 。 还原出原始信号。

5. 最大似然估计时，假设 $\vec{\mathbf{x}}_i$ 和 $\vec{\mathbf{x}}_j$ 之间是相互独立的。事实上对于语音信号或者其他具有时间连续性依赖性的数据（如：温度），这个假设不能成立。

- 但是当数据足够多，假设独立对于效果影响不大。
- 如果事先打乱样本，则会加快梯度下降法的收敛速度。

7.3 FastICA

1. `FastICA` 的基本思想是：使得 $s_i = \vec{\mathbf{w}}_i^T\vec{\mathbf{x}}$ 最不可能是高斯信号。
2. 度量随机变量 s 的分布为高斯分布的程度：
   - 基于峰度 `kurtosis` 的方法：$\text{kurt}[s] = \mathbb{E}[s^4] - 3(\mathbb{E}[y^2])^2$ 。
     - 对于高斯分布，其峰度为 0 ，因此如果 $\text{kurt}[s]$ 偏离 0 值越远，则它越不可能是高斯分布。
     - 实际上该指标只能刻画一个分布是不是高斯分布，而无法描述一个分布偏离高斯分布的程度。因此该方法实际效果一般。
   - 基于负熵的方法：$J[s] = H[s_{gauss}] - H[s]$。 其中 H 为随机变量的熵， s_{gauss} 是一个高斯分布，其均值、方差与非高斯分布的 s 的均值、方差相同。
     - 在信息论中可以证明：在相同方差的条件下，高斯分布的熵最大。因此可以认为 $J[s]$ 越大，s 的分布偏离高斯分布越远。
     - 由于计算 $J[s]$ 必须需要知道 s 的概率密度分布函数，实际任务中很难实现。因此通常采用近似公式 $J[s] = \{\mathbb{E}[G(s_{gauss})] - \mathbb{E}[G(s)]\}^2$ 来实现。其中 G 为非线性函数，可以为：

       $G(s) = \tanh(as)$ 、 $G(s) = s\exp(-s^2/2)$ 、$G(s) = s^3$ 。其中 $1 \leq a \leq 2$ 。

       其导数为 $G'(s)$ 。
3. 定义目标函数为 $\mathcal{J} = \sum_{i=1}^n J[s_i]$ ，采用梯度下降法求解。其迭代公式为：

$$\vec{\mathbf{w}} \leftarrow \mathbb{E}[\vec{\mathbf{x}}G(\vec{\mathbf{w}}^T\vec{\mathbf{x}})] - \mathbb{E}[G'(\vec{\mathbf{w}}^T\vec{\mathbf{x}})]\vec{\mathbf{w}}$$

$$\vec{\mathbf{w}} \leftarrow \frac{\vec{\mathbf{w}}}{||\vec{\mathbf{w}}||}$$

   - 一次 `FastICA` 算法能够估计出一个独立成分，为了估计出若干个独立成分，需要进行多次 `FastICA` 算法来得到 $\vec{\mathbf{w}}_1,\cdots,\vec{\mathbf{w}}_n$ 。
   - 为了防止这些向量收敛到同一个最大值（即：分解出同一个独立成分），当估计 $\vec{\mathbf{w}}_{i+1}$ 时，需要减去 $\vec{\mathbf{w}}_{i+1}$ 在之前得到的 $\vec{\mathbf{w}}_1,\cdots,\vec{\mathbf{w}}_i$ 上的投影。即：

$$\vec{\mathbf{w}}_{i+1} \leftarrow \mathbb{E}[\vec{\mathbf{x}}G(\vec{\mathbf{w}}_{i+1}^T\vec{\mathbf{x}})] - \mathbb{E}[G'(\vec{\mathbf{w}}_{i+1}^T\vec{\mathbf{x}})]\vec{\mathbf{w}}_{i+1}$$

$$\vec{\mathbf{w}}_{i+1} \leftarrow \vec{\mathbf{w}}_{i+1} - \sum_{k=1}^{i}(\vec{\mathbf{w}}_{i+1}^T\vec{\mathbf{w}}_k)\vec{\mathbf{w}}_k$$

$$\vec{\mathbf{w}}_{i+1} \leftarrow \frac{\vec{\mathbf{w}}_{i+1}}{||\vec{\mathbf{w}}_{i+1}||}$$

     其中下标 $i+1$ 并不是迭代步数，而是第 $i+1$ 个 $\mathbf{w}$ 。

7.4 预处理

1. `ICA` 中需要进行预处理，主要有数据中心化、白化两个步骤。
2. 数据中心化：对数据集 $\mathbb{D} = \{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$ 执行：

$$\vec{\mathbf{x}}_i \leftarrow \vec{\mathbf{x}}_i - \frac{1}{N}\sum_{j=1}^N\vec{\mathbf{x}}_j$$

$\bar{\vec{\mathbf{x}}}=\frac{1}{N}\sum_{j=1}^{N}\vec{\mathbf{x}}_j$ 称作数据集 $\mathbb{D}$ 的中心向量，它的各元素就是各个特征的均值。

该操作使得 $\mathbb{E}[\vec{\mathbf{x}}]=\vec{\mathbf{0}}$，这也意味着 $\vec{\mathbf{s}}$ 也是零均值的。

3. 白化：对 $\mathbb{D}$ 执行线性变化，使其协方差矩阵为单位矩阵 $\mathbf{I}$。即：$\mathbb{E}[\vec{\mathbf{x}}'\vec{\mathbf{x}}'^{\,T}]=\mathbf{I}$。

   $\vec{\mathbf{x}}$ 的协方差矩阵为 $\mathbf{X}^T\mathbf{X}$（经过数据中心化之后），设其特征值为 $\lambda_1,\cdots,\lambda_n$，对应的特征向量组成的矩阵为 $\mathbf{E}$，则有：$\mathbf{X}^T\mathbf{X}=\mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$，其中 $\mathbf{\Lambda}=\mathrm{diag}(\lambda_1\cdots,\lambda_n)$。

   令：$\vec{\mathbf{x}}'=\mathbf{E}\mathbf{\Lambda}^{-1/2}\mathbf{E}^T\vec{\mathbf{x}}$，则有：$\mathbb{E}[\vec{\mathbf{x}}'\vec{\mathbf{x}}'^{\,T}]=\mathbf{I}$。

   - 若 $\vec{\mathbf{x}}'$ 的协方差矩阵为单位矩阵，则根据 $\vec{\mathbf{s}}=\mathbf{W}\vec{\mathbf{x}}'$ 有：$\mathbb{E}[\vec{\mathbf{s}}\vec{\mathbf{s}}^T]=\mathbf{W}\mathbf{W}^T$。
     - 根据假设，$\vec{\mathbf{s}}$ 中各信号 $s_1,s_2,\cdots,s_n$ 是相互独立的，因此 $\vec{\mathbf{s}}$ 的协方差矩阵必须是对角矩阵。
     - 能够对 s_i 进行缩放时，相应的 $\vec{\mathbf{w}}_i$ 进行同样缩放，等式仍然成立。即：最终的解与 $\vec{\mathbf{w}}_i$ 幅度无关。因此可以选择 $\vec{\mathbf{w}}_i$ 的长度为1。

     因此有：$\mathbb{E}[\vec{\mathbf{s}}\vec{\mathbf{s}}^T]=\mathbf{W}\mathbf{W}^T=\mathbf{I}$。
   - $\mathbf{W}\mathbf{W}^T=\mathbf{I}$，即 $\vec{\mathbf{w}}_i,\vec{\mathbf{w}}_j,i\neq j$ 相互正交且长度为1。这也是`FastICA`算法中需要对 $\vec{\mathbf{w}}_{i+1}$ 进行归一化和正交化的原因。

     这使得矩阵 $\mathbf{W}$ 的参数从 n^2 个降低到 $n\times(n-1)/2$ 个，减小了算法的计算复杂度。

八、t-SNE

1. `t-SNE:t-distributed stochastic neighbor embedding` 是一种非线性降维算法，它是由`SNE`发展而来。

8.1 SNE

1. SNE 的基本思想：如果两个样本在高维相似，则它们在低维也相似。
2. SNE 主要包含两步：
   - 构建样本在高维的概率分布。
   - 在低维空间里重构这些样本的概率分布，使得这两个概率分布之间尽可能相似。
3. 在数据集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$ 中，给定一个样本 $\vec{\mathbf{x}}_i$ ，然后计算 $\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_{i-1},\vec{\mathbf{x}}_{i+1},\cdots,\vec{\mathbf{x}}_N\}$ 是 $\vec{\mathbf{x}}_i$ 的邻居的概率。

   SNE 假设：如果 $\vec{\mathbf{x}}_j$ 与 $\vec{\mathbf{x}}_i$ 越相似，则 $\vec{\mathbf{x}}_j$ 是 $\vec{\mathbf{x}}_i$ 的邻居的概率越大。
   - 相似度通常采用欧几里得距离来衡量，两个样本距离越近则它们越相似。
   - 概率 $p(\vec{\mathbf{x}}_j\mid\vec{\mathbf{x}}_i)$ 通常采用指数的形式：

$$p(\vec{\mathbf{x}}_j\mid\vec{\mathbf{x}}_i)\propto\exp\left(-||\vec{\mathbf{x}}_j-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)$$

   对 $j=1,2,\cdots,N,j\neq i$ 进行归一化有：

$$p(\vec{\mathbf{x}}_j \mid \vec{\mathbf{x}}_i)=\frac{\exp\left(-||\vec{\mathbf{x}}_j-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)}{\sum_{k\neq i}\exp\left(-||\vec{\mathbf{x}}_k-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)}$$

其中 σ_i 是与 $\vec{\mathbf{x}}_i$ 相关的、待求得参数，它用于对距离进行归一化。

- 定义 $p_{j|i}=p(\vec{\mathbf{x}}_j \mid \vec{\mathbf{x}}_i)$ 。由于挑选 $\vec{\mathbf{x}}_j$ 时排除了 $\vec{\mathbf{x}}_i$ ，因此有 $p_{i|i}=0$ 。
- 定义概率分布 $P_i=(p_{1|i},\cdots,p_{N|i})$ ，它刻画了所有其它样本是 $\vec{\mathbf{x}}_i$ 的邻居的概率分布。

4. 假设经过降维，样本 $\vec{\mathbf{x}}_i\in\mathbb{R}^n$ 在低维空间的表示为 $\vec{\mathbf{z}}_i\in\mathbb{R}^d$ ，其中 $d\leq n$ 。
   - 定义：

$$q_{j|i}=q(\vec{\mathbf{z}}_j \mid \vec{\mathbf{z}}_i)=\frac{\exp\left(-||\vec{\mathbf{z}}_j-\vec{\mathbf{z}}_i||^2\right)}{\sum_{k\neq i}\exp\left(-||\vec{\mathbf{z}}_k-\vec{\mathbf{z}}_i||^2\right)}$$

     其中 $q_{j|i}$ 表示给定一个样本 $\vec{\mathbf{z}}_i$ ，然后计算 $\{\vec{\mathbf{z}}_1,\cdots,\vec{\mathbf{z}}_{i-1},\vec{\mathbf{z}}_{i+1},\cdots,\vec{\mathbf{z}}_N\}$ 是 $\vec{\mathbf{z}}_j$ 的邻居的概率。
     - 这里选择 $\sigma^2=\frac{1}{2}$ 为固定值。
     - 同样地，有 $q_{i|i}=0$ 。
   - 定义概率分布 $Q_i=(q_{1|i},\cdots,q_{N|i})$ ，它刻画了所有其它样本是 $\vec{\mathbf{z}}_i$ 的邻居的概率分布。
5. 对于样本 $\vec{\mathbf{x}}_i$ ，如果降维的效果比较好，则有 $p_{j|i}=q_{j|i},i=1,2,\cdots,N$ 。即：降维前后不改变 $\vec{\mathbf{x}}_i$ 周围的样本分布。

   对于 $\vec{\mathbf{x}}_i$ ，定义其损失函数为分布 P_i 和 Q_i 的距离，通过 `KL` 散度来度量。

   对于全体数据集 $\mathbb{D}$ ，整体损失函数为：

$$\mathcal{L}=\sum_{i=1}^{N}KL(P_i||Q_i)=\sum_{i=1}^{N}\sum_{j=1}^{N}p_{j|i}\log\frac{p_{j|i}}{q_{j|i}}$$

6. `KL` 散度具有不对称性，因此不同的错误对应的代价是不同的。给定样本 $\vec{\mathbf{x}}_i$ ：
   - 对于高维距离较远的点 $\vec{\mathbf{x}}_j$ ，假设 $p_{j|i}=0.2$，如果在低维映射成距离比较近的点，假设 $q_{j|i}=0.8$ 。则该点的代价为 $0.2\log\frac{0.2}{0.8}=-0.277$ 。
   - 对于高维距离较近的点 $\vec{\mathbf{x}}_j$ ，假设 $p_{j|i}=0.8$，如果在低维映射成距离比较远的点，假设 $q_{j|i}=0.2$ 。则该点的代价为 $0.8\log\frac{0.8}{0.2}=1.11$ 。

   因此 `SNE` 倾向于将高维空间较远的点映射成低位空间中距离较近的点。这意味着 `SNE` 倾向于保留高维数据中的局部特征（因为远处的特征会被扭曲）。因此 `SNE` 更关注局部结构而忽视了全局结构。
7. 从 $p_{j|i}\propto\exp\left(-||\vec{\mathbf{x}}_j-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)$ 可以看到：σ_i 是与 $\vec{\mathbf{x}}_i$ 相关的、用于对距离进行归一化的参数。
   - 若 σ_i 较大，则概率 $p_{j|i}\gg 0$ 的样本 $\vec{\mathbf{x}}_j$ 更多，它们覆盖的范围更广，概率分布 P_i 的熵越大。
   - 若 σ_i 较小，则概率 $p_{j|i}\gg 0$ 的样本 $\vec{\mathbf{x}}_j$ 更少，它们覆盖的范围更窄，概率分布 P_i 的熵越小。

   定义困惑度为：$\text{Perp}(P_i)=2^{H(P_i)}$ ，其中 $H(P_i)$ 表示概率分布 P_i 的熵。
   - 困惑度刻画了 $\vec{\mathbf{x}}_i$ 附近的有效近邻点个数。
   - 通常选择困惑度为 `5~50` 之间。它表示：对于给定的 $\vec{\mathbf{x}}_i$ ，只考虑它周围最近的 `5~50` 个样本的分布。
   - 给定困惑度之后，可以用二分搜索来寻找合适的 σ_i 。

8. 当 σ_i 已经求得，可以根据数据集 $\mathbb{D}$ 以及公式 $p_{j|i}=\frac{\exp\left(-||\vec{\mathbf{x}}_j-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)}{\sum_{k\neq i}\exp\left(-||\vec{\mathbf{x}}_k-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)}$ 来求出 $p_{j|i}$ 。

   剔除 $\mathcal{L}$ 中的已知量（$\sum_i\sum_j p_{j|i}\log p_{j|i}$），则有：$\mathcal{L}=-\sum_{i=1}^N\sum_{j=1}^N p_{j|i}\log q_{j|i}$ 。

   可以通过梯度下降法求解损失函数的极小值。

   - 记 $y_{i,j}=-||\vec{\mathbf{z}}_j-\vec{\mathbf{z}}_i||^2$，则有 $q_{j|i}=\frac{\exp(y_{i,j})}{\sum_{k\neq i}\exp(y_{i,k})}$ 。

     考虑到 `softmax` 交叉熵的损失函数 $\sum y_{true}\log q$ 的梯度为 $y_{true}-q$ 。令分布 $p_{j|i}$ 为样本的真实标记 y_{true} ，则有：

$$\nabla_{y_{i,j}}\left(\sum_{j=1}^N p_{j|i}\log q_{j|i}\right)=p_{j|i}-q_{j|i}$$

$$\begin{aligned}\nabla_{\vec{\mathbf{z}}_i}\left(\sum_{j=1}^N p_{j|i}\log q_{j|i}\right)&=\nabla_{y_{i,j}}\left(\sum_{j=1}^N -p_{j|i}\log q_{j|i}\right)\times\nabla_{\vec{\mathbf{z}}_i}y_{i,j}\\&=-2(p_{j|i}-q_{j|i})\times(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)\end{aligned}$$

$$\begin{aligned}\nabla_{\vec{\mathbf{z}}_j}\left(\sum_{i=1}^N p_{j|i}\log q_{j|i}\right)&=\nabla_{y_{i,j}}\left(\sum_{i=1}^N -p_{j|i}\log q_{j|i}\right)\times\nabla_{\vec{\mathbf{z}}_j}y_{i,j}\\&=-2(p_{j|i}-q_{j|i})\times(\vec{\mathbf{z}}_j-\vec{\mathbf{z}}_i)\end{aligned}$$

     考虑梯度 $\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}$，有两部分对它产生贡献：

     - 给定 $\vec{\mathbf{x}}_i$ 时，梯度的贡献为： $-\sum_j\nabla_{\vec{\mathbf{z}}_i}\left(\sum_{j=1}^N p_{j|i}\log q_{j|i}\right)$。
     - 给定 $\vec{\mathbf{x}}_j$ 时，梯度的贡献为：$-\sum_j\nabla_{\vec{\mathbf{z}}_i}\left(\sum_{j=1}^N p_{i|j}\log q_{i|j}\right)$ 。

     因此有：

$$\begin{aligned}\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}&=-\sum_j(-2(p_{j|i}-q_{j|i})\times(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j))+(-2(p_{i|j}-q_{i|j})\times(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j))\\&=\sum_j 2(p_{j|i}-q_{j|i}+p_{i|j}-q_{i|j})(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)\end{aligned}$$

     该梯度可以用分子之间的引力和斥力进行解释：低维空间中的点 $\vec{\mathbf{z}}_i$ 的位置是由其它所有点对其作用力的合力决定。

     - 某个点 $\vec{\mathbf{z}}_\mathbf{j}$ 对 $\vec{\mathbf{z}}_i$ 的作用力的方向：沿着 $\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j$ 的方向。
     - 某个点 $\vec{\mathbf{z}}_\mathbf{j}$ 对 $\vec{\mathbf{z}}_i$ 的作用力的大小：取决于 $\vec{\mathbf{z}}_\mathbf{j}$ 和 $\vec{\mathbf{z}}_i$ 之间的距离。

   - 为了避免陷入局部最优解，可以采用如下的方法：

     - 采用基于动量的随机梯度下降法：

$$\begin{aligned}\vec{\mathbf{v}}&\leftarrow\alpha\vec{\mathbf{v}}-\epsilon\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}\\\vec{\mathbf{z}}_i&\leftarrow\vec{\mathbf{z}}_i+\vec{\mathbf{v}}\end{aligned}$$

       其中 ϵ 为学习率， $\alpha\in[0,1)$ 为权重衰减系数。

     - 每次迭代过程中引入一些高斯噪声，然后逐渐减小该噪声。

8.2 对称 SNE

1. 在 `SNE` 中使用的是条件概率分布 $p_{j|i}$ 和 $q_{j|i}$ ，它们分别表示在高维和低维下，给定第 i 个样本的情况下，第 j 个样本的分布。

而对称 SNE 中使用联合概率分布 $p_{i,j}$ 和 $q_{i,j}$ ，它们分别表示在高维和低维下，第 i 个样本和第 j 个样本的联合分布。其中：

$$p_{i,j}=\frac{\exp\left(-||\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j||^2/(2\sigma^2)\right)}{\sum_k\sum_{l,k\neq l}\exp\left(-||\vec{\mathbf{x}}_k-\vec{\mathbf{x}}_l||^2/(2\sigma^2)\right)}$$

$$q_{i,j}=\frac{\exp\left(-||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2\right)}{\sum_k\sum_{l,k\neq l}\exp\left(-||\vec{\mathbf{z}}_k-\vec{\mathbf{z}}_l||^2\right)}$$

$$p_{i,i}=0,\quad q_{i,i}=0$$

根据定义可知 $p_{i,j}$ 和 $q_{i,j}$ 都满足对称性：$p_{i,j}=p_{j,i}$ 、$q_{i,j}=q_{j,i}$ 。

2. 上述定义的 $p_{i,j}$ 存在异常值问题。

   - 当 $\vec{\mathbf{x}}_i$ 是异常值时，对所有的 $\vec{\mathbf{x}}_j,j\neq i$， 有 $||\vec{\mathbf{x}}_i-\vec{\mathbf{x}}_j||^2$ 都很大。这使得 $p_{i,j},j=1,2,\cdots,N$ 都几乎为 0 。

     这就使得 $\vec{\mathbf{x}}_i$ 的代价： $\sum_j p_{i,j}\log q_{i,j}\to 0$ 。即：无论 $\vec{\mathbf{z}}_i$ 周围的点的分布如何，它们对于代价函数的影响忽略不计。

     而在原始 SNE 中，可以保证 $\sum_{j=1}^N p_{j|i}=1$， $\vec{\mathbf{z}}_i$ 周围的点的分布会影响代价函数。

   - 为解决异常值问题，定义： $p_{i,j}=\frac{p_{j|i}+p_{i|j}}{2N}$ 。这就使得 $\sum_{j=1}^N p_{i,j}>\frac{1}{2N}$ ，从而使得 $\vec{\mathbf{z}}_i$ 周围的点的分布对代价函数有一定的贡献。

     注意：这里并没有调整 $q_{i,j}$ 的定义。

3. 对称 SNE 的目标函数为：$\mathcal{L}=\sum_{i=1}^N KL(P_i||Q_i)=-\sum_{i=1}^N\sum_{j=1}^N p_{i,j}\log q_{i,j}$ 。

   根据前面的推导有：$\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}=\sum_j 4(p_{i,j}-q_{i,j})(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)$ 。其中：

   $p_{i,j}=\frac{p_{j|i}+p_{i|j}}{2N},q_{i,j}=\frac{\exp\left(-||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2\right)}{\sum_k\sum_{l,k\neq l}\exp\left(-||\vec{\mathbf{z}}_k-\vec{\mathbf{z}}_l||^2\right)}$ 。

4. 实际上对称 SNE 的效果只是略微优于原始 SNE 的效果。

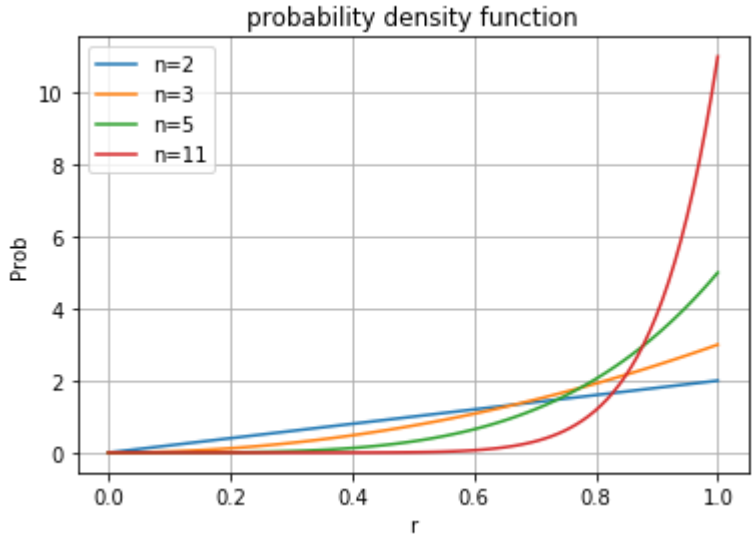
8.3 拥挤问题

1. 拥挤问题 Crowding Problem ：指的是 SNE 的可视化效果中，不同类别的簇挤在一起，无法区分开来。

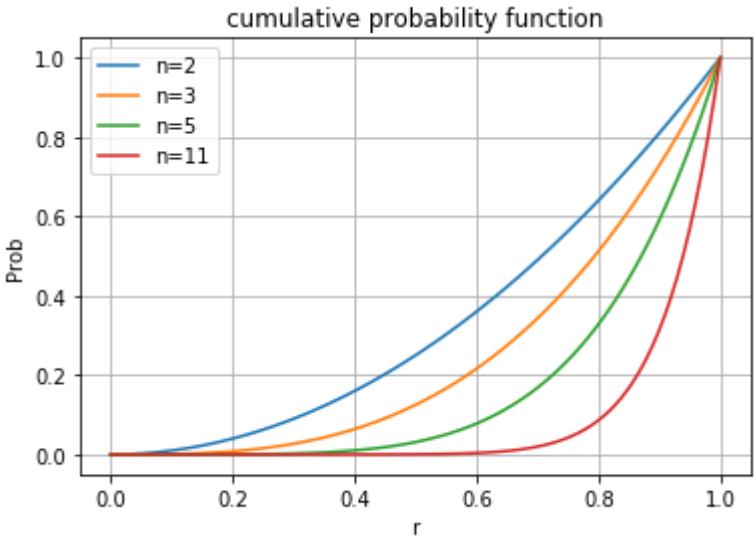
   拥挤问题本质上是由于高维空间距离分布和低维空间距离分布的差异造成的。

2. 考虑 n 维空间中一个以原点为中心、半径为 1 的超球体。在球体内部随机选取一个点，则该点距离原点的距离为 r 的概率密度分布为：

$$p(r)=\lim_{\Delta r\to 0}\frac{(r+\Delta r)^n-r^n}{\Delta r}=nr^{n-1}$$



累计概率分布为： $F(r)=\int_0^r p(r)dr=r^n$ 。



可以看到：随着空间维度的增长，采样点在原点附近的概率越低、在球体表面附近的概率越大。

如果直接将这种距离分布关系保留到低维，则就会出现拥挤问题。

8.4 t-SNE

1. `t-SNE` 通过采用不同的分布来解决拥挤问题：
   - 在高维空间下使用高斯分布将距离转换为概率分布。
   - 在低维空间下使用 `t` 分布将距离转换为概率分布。这也是 `t-SNE` 的名字的由来。
2. `t-SNE` 使用自由度为 `1` 的 `t` 分布。此时有：$q_{i,j}=\frac{(1+||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2)^{-1}}{\sum_k\sum_{l,l\neq k}(1+||\vec{\mathbf{z}}_k-\vec{\mathbf{z}}_l||^2)^{-1}}$。

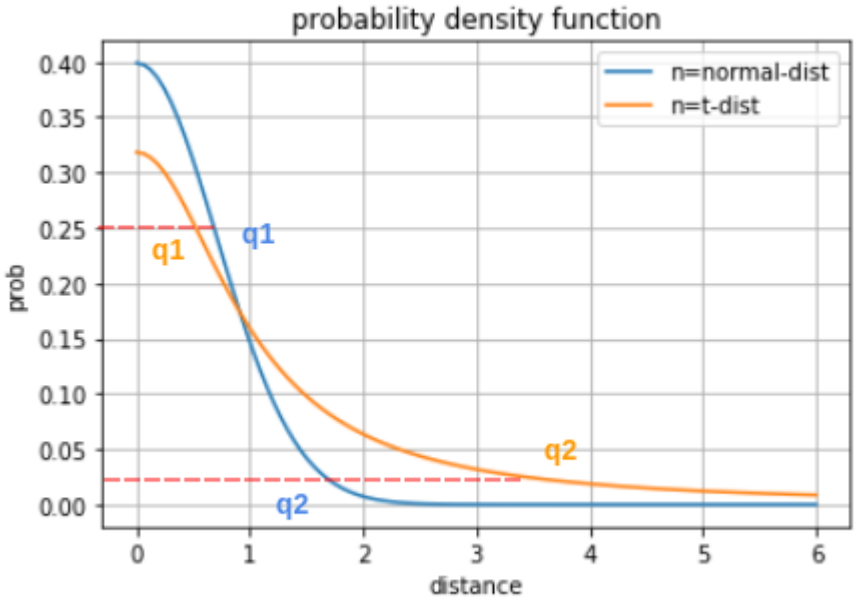
   则梯度为：

   $$\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}=\sum_j 4(p_{i,j}-q_{i,j})(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)(1+||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2)^{-1}$$

   也可以选择自由度超过 `1` 的 `t` 分布。自由度越高，越接近高斯分布。
3. `t` 分布相对于高斯分布更加偏重长尾。可以看到：

- 对于高维空间相似度较大的点（如下图中的 `q1` ），相比较于高斯分布， `t` 分布在低维空间中的距离要更近一点。
- 对于高维空间相似度较小的点（如下图中的 `q2` ），相比较于高斯分布， `t` 分布在低维空间中的距离要更远一点。

即：同一个簇内的点（距离较近）聚合的更紧密，不同簇之间的点（距离较远）更加疏远。



4. 优化过程中的技巧：
   - 提前压缩 `early compression` ：开始初始化的时候，各个点要离得近一点。这样小的距离，方便各个聚类中心的移动。可以通过引入L2正则项(距离的平方和)来实现。
   - 提前放大 `early exaggeration` ：在开始优化阶段， $p_{i,j}$ 乘以一个大于 1 的数进行扩大，来避免 $q_{i,j}$ 太小导致优化太慢的问题。比如前 50 轮迭代，$p_{i,j}$ 放大四倍。
5. `t-SNE` 的主要缺点：
   - `t-SNE` 主要用于可视化，很难用于降维。有两个原因：
     - `t-SNE` 没有显式的预测部分，所以它无法对测试样本进行直接降维。

       一个解决方案是：构建一个回归模型来建立高维到低维的映射关系，然后通过该模型来对测试样本预测其低维坐标。
     - `t-SNE` 通常用于2维或者3维的可视化。如果数据集相互独立的特征数量如果较大，则映射到 2~3 维之后信息损失严重。
   - `t-SNE` 中的距离、概率本身没有意义，它们主要用于描述样本之间的概率分布。
   - `t-SNE` 代价函数是非凸的，可能得到局部最优解。
   - `t-SNE` 计算开销较大，训练速度慢。其计算复杂度为 $O(N^2)$ 。经过优化之后可以达到 $O(N\log N)$ 。

8.5 t-SNE 改进

1. `2014` 年 `Mattern` 在论文 `Accelerating t-SNE using Tree-Based Algorithms` 中对 `t-SNE` 进行了改进，主要包括两部分：
   - 使用 `kNN` 图来表示高维空间中点的相似度。
   - 优化了梯度的求解过程。

8.5.1 kNN 图的相似度表示

1. 注意到 $p(\vec{\mathbf{x}}_j \mid \vec{\mathbf{x}}_i)$ 的表达式：

$$p(\vec{\mathbf{x}}_j \mid \vec{\mathbf{x}}_i)=\frac{\exp\left(-||\vec{\mathbf{x}}_j-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)}{\sum_{k\neq i}\exp\left(-||\vec{\mathbf{x}}_k-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)}$$

   - 每个数据点 $\vec{\mathbf{x}}_i$ 都需要计算 $\sum_{k\neq i}\exp\left(-||\vec{\mathbf{x}}_k-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)$，这一项需要计算所有其他样本点到 $\vec{\mathbf{x}}_i$ 的距离。当数据集较大时，这一项的计算量非常庞大。
   - 事实上，如果两个点相距较远，则它们互为邻居的概率非常小，因为 $\lim_{||\vec{\mathbf{x}}_j-\vec{\mathbf{x}}_i||\to\infty}\exp\left(-||\vec{\mathbf{x}}_j-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)=0$。

     因此在构建高维空间的点的相似度关系时，只需要考虑 $\vec{\mathbf{x}}_i$ 最近的若干个邻居点即可。

2. 考虑与点 $\vec{\mathbf{x}}_i$ 最近的 $\lfloor 3\mathrm{Perp}\rfloor$ 个点，其中 Perp 为点 $\vec{\mathbf{x}}_i$ 的周围点的概率分布的困惑度。记这些邻居结点的集合为 $\mathbb{N}_i$，则有：

$$p_{j|i}=p(\vec{\mathbf{x}}_j \mid \vec{\mathbf{x}}_i)=\frac{\exp\left(-||\vec{\mathbf{x}}_j-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)}{\sum_{k\in\mathbb{N}_i}\exp\left(-||\vec{\mathbf{x}}_k-\vec{\mathbf{x}}_i||^2/(2\sigma_i^2)\right)},\ j\in\mathbb{N}_i$$
$$p_{j|i}=0,\ j\neq\mathbb{N}_i$$

   这种方法会大大降低计算量。但是需要首先构建高维空间的`kNN`图，从而快速的获取 $\vec{\mathbf{x}}_i$ 最近的 $\lfloor 3\mathrm{Perp}\rfloor$ 个点。

3. `Maaten` 使用`VP树:vantage-point tree`来构建`kNN`图，可以在 $O(\mathrm{Perp}N\log N)$ 的计算复杂度内得到一个精确的 `kNN` 图。

8.5.2 梯度求解优化

1. 对 $\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}$ 进行变换。定义 $Z=\sum_k\sum_{l,l\neq k}(1+||\vec{\mathbf{z}}_k-\vec{\mathbf{z}}_l||^2)^{-1}$。则根据：

$$q_{i,j}=\frac{(1+||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2)^{-1}}{\sum_k\sum_{l,l\neq k}(1+||\vec{\mathbf{z}}_k-\vec{\mathbf{z}}_l||^2)^{-1}}$$

   有：$(1+||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2)^{-1}=q_{i,j}\times Z$。

   则有：

$$\begin{aligned}\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}&=\sum_j 4(p_{i,j}-q_{i,j})(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)(1+||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2)^{-1}\\&=\sum_j 4(p_{i,j}-q_{i,j})(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)q_{i,j}Z\\&=4\left[\sum_j p_{i,j}q_{i,j}Z(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)-\sum_j q_{i,j}^2Z(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)\right]\end{aligned}$$

   定义引力为：$F_{attr}=\sum_j p_{i,j}q_{i,j}Z(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)$，斥力为：$F_{rep}=\sum_j q_{i,j}^2Z(\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j)$。则有：

   $\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}=4(F_{attr}-F_{rep})$。

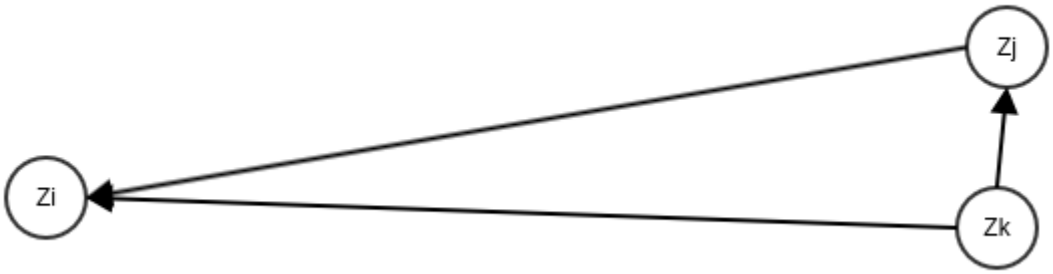
2. 引力部分的计算比较简单。

   考虑到 $q_{i,j}\times Z=(1+||\vec{\mathbf{z}}_i-\vec{\mathbf{z}}_j||^2)^{-1}$，则有：

$$F_{attr} = \sum_j p_{i,j} q_{i,j} Z(\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_j) = \sum_j p_{i,j} \frac{\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_j}{1 + ||\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_j||^2}$$

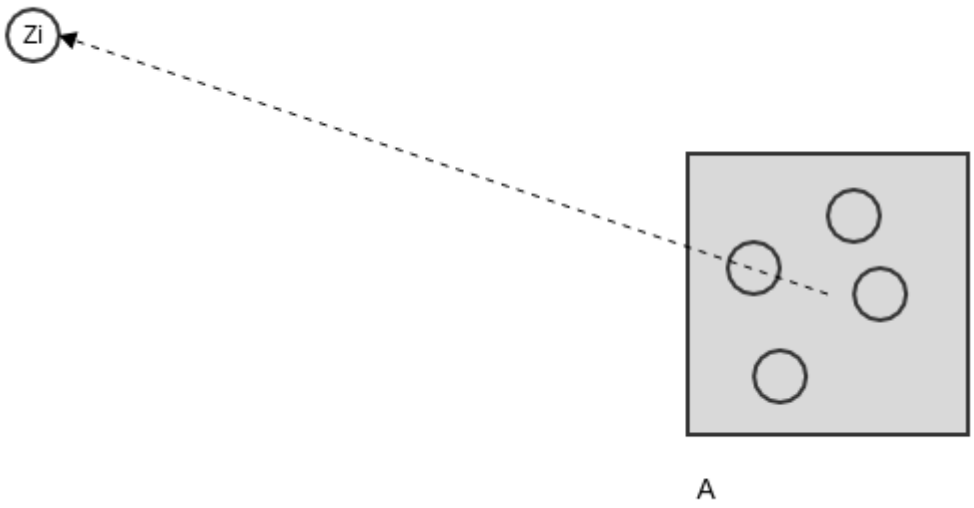
根据 $\lim_{||\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_j|| \to \infty} \frac{\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_j}{1 + ||\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_j||^2} = \vec{\mathbf{0}}$，则只可以忽略较远的结点。仅考虑与点 $\vec{\mathbf{z}}_i$ 最近的 $\lfloor 3\text{Perp} \rfloor$ 个点，则引力部分的计算复杂度为 $O(\text{Perp}N)$ 。

3. 斥力部分的计算比较复杂，但是仍然有办法进行简化。
   - 考虑下图中的三个点，其中 $||\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_j|| \simeq ||\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_k|| \gg ||\vec{\mathbf{z}}_j - \vec{\mathbf{z}}_k||$ 。此时认为点 $\vec{\mathbf{z}}_j$ 和 $\vec{\mathbf{z}}_k$ 对点 $\vec{\mathbf{z}}_i$ 的斥力是近似相等的。



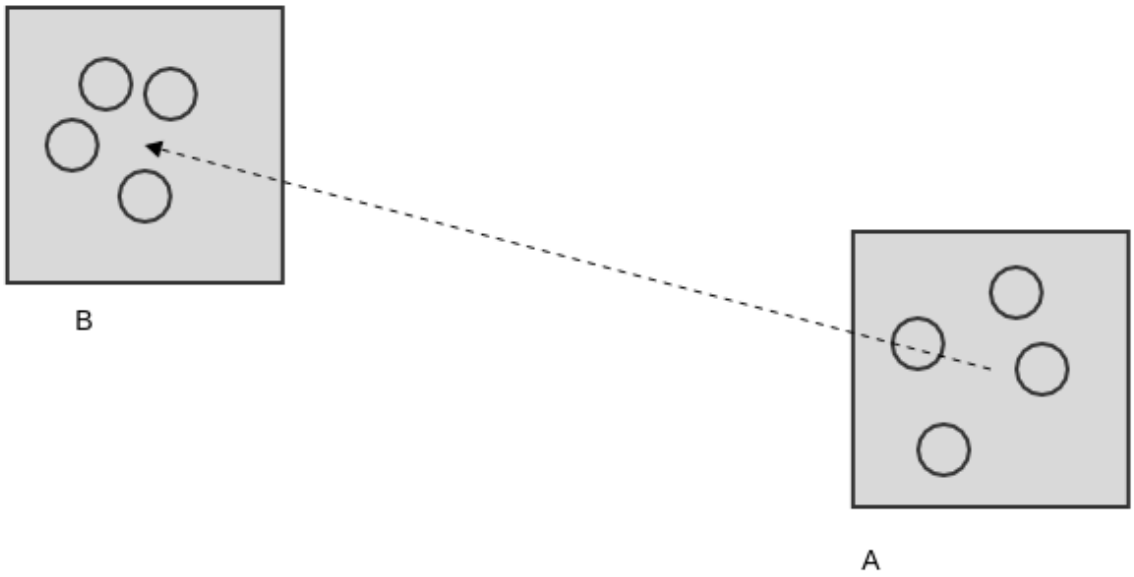
   - 事实上这种情况在低维空间中很常见，甚至某片区域中每个点对 $\vec{\mathbf{z}}_i$ 的斥力都可以用同一个值来近似，如下图所示。

     假设区域 $\mathbb{A}$ 中 4 个点对 $\vec{\mathbf{z}}_i$ 产生的斥力都是近似相等的，则可以计算这 4 个点的中心（虚拟的点）产生的斥力 $F_{\mathbb{A}_c}$，则区域 $\mathbb{A}$ 产生的总的斥力为 $4F_{\mathbb{A}_c}$ 。



   - `Matten` 使用四叉树来完成区域搜索任务，并用该区域中心点产生的斥力作为整个区域的斥力代表值。

     并非所有区域都满足该近似条件，这里使用`Barnes-Hut` 算法搜索并验证符合近似条件的`点-区域` 组合。
   - 事实上可以进一步优化，近似区域到区域之间的斥力。
     - 如下所示为区域 $\mathbb{A}$ 和区域 $\mathbb{B}$ 中，任意两个结点之间的斥力都可以用 $F_{\mathbb{A}\mathbb{B}_c}$ 来近似。其中 $F_{\mathbb{A}\mathbb{B}_c}$ 代表区域 $\mathbb{A}$ 的中心（虚拟的点）和区域 $\mathbb{B}$ 的中心（虚拟的点）产生的斥力。
     - 同样也需要判断两个区域之间的斥力是否满足近似条件。这里采用了`Dual-tree` 算法搜索并验证符合近似条件的`区域-区域` 组合 。



九、LargeVis

1. 数据可视化的本质任务是：在低维空间中保存高维数据的内在结构。即：
   - 如果高维空间中两个点相似，则低维空间中它们距离较近。
   - 如果高维空间中两个点不相似，则低维空间中它们距离较远。
2. `t-SNE` 及其变体的核心思想：
   - 从高维数据中提取数据之间的内在结构。这是通过相似度、条件概率分布来实现的。
   - 将该结构投影到低维空间。这是通过 `KL` 散度来实现的。
3. `t-SNE` 的重要缺陷：
   - 计算 `kNN` 图存在计算瓶颈。`t-SNE` 构建 `kNN` 图时采用 `Vantage-Point` 树，其性能在数据维度增加时明显恶化。
   - 当数据量变大时，可视化效率明显恶化。
   - `t-SNE` 的参数对于不同数据集非常敏感，需要花费较多的时间在调参上。
4. `LargeVis` 是一种不同的可视化技术，可以支持百万级别的数据点可视化。

   与 `t-SNE` 相比，它主要在以下方面进行了改进：
   - 通过随机投影树来构建近似 `kNN` 图。
   - 提出一个概率模型，该模型的目标函数可以通过异步随机梯度得到优化。

9.1 近似 `kNN` 图

9.1.1 随机投影树

1. 随机投影树和 `kd` 树一样，都是一种分割 n 维数据空间的数据结构。
   - `kd` 树严格按照坐标轴来划分空间，树的深度正比于空间的维度 n 。

     当 n 的数值很大（即数据空间维度很高）时， `kd` 树的深度会非常深。
   - 随机投影树划分空间的方式比较灵活，其深度为 $O\log(N)$ ，其中 N 为样本的数量。
2. 随机投影树建立过程：
   - 将数据集 $\mathbb{D}=\{\vec{\mathbf{x}}_1,\cdots,\vec{\mathbf{x}}_N\}$ 中的所有点放入根节点。

- 随机选取一个从原点出发的向量 $\vec{\mathbf{v}}_1$ ，与这个向量垂直的空间 $\mathbf{S}_1$ 将根节点划分为两部分。
  - 将 $\mathbf{S}1$ 左侧的点划分到左子树。$\mathbf{S}_1$ 左侧的点是满足 $\vec{\mathbf{x}}_i \cdot \vec{\mathbf{v}}_1 \leq 0$ 的点。
  - 将 $\mathbf{S}1$ 右侧的点划分到右子树。$\mathbf{S}_1$ 右侧的点是满足 $\vec{\mathbf{x}}_i \cdot \vec{\mathbf{v}}_1 > 0$ 的点。
- 重复划分左子树和右子树，使得每个子树包含的点的数量符合要求。

9.1.2 构建 kNN 图

1. LargeVis 通过随机投影树来构建近似 kNN 图。

   首先建立随机投影树。对于数据点 $\vec{\mathbf{x}}_i$ ，找到树中对应的叶节点 node_i。然后将叶节点 node_i 对应的子空间中的所有数据点都作为数据点 $\vec{\mathbf{x}}_i$ 的候选邻居。

2. 单棵随机投影树构建的 kNN 图精度不高。为了提高 kNN 图的精度，通常需要建立多个随机投影树 $\text{Tree}_1, \text{Tree}_2, \cdots, \text{Tree}_T$。

   对于数据点 $\vec{\mathbf{x}}_i$ ，对每颗树 Tree_t ，找到树 Tree_t 中对应的叶节点 $\text{node}_i^{(t)}$。然后将叶节点 $\text{node}_i^{(t)}$ 对应的子空间中的所有数据点都作为数据点 $\vec{\mathbf{x}}_i$ 的候选邻居。

   这种做法的缺点是：为了得到足够高的精度，需要大量的随机投影树，这导致计算量较大。

3. LargeVis 使用邻居探索技术来构建高精度的 kNN 图。

   其基本思路是：邻居的邻居也可能是我的邻居。

   - 首先建立多个随机投影树 $\text{Tree}_1, \text{Tree}_2, \cdots, \text{Tree}_T$ ，这里 T 比较小。
   - 对于数据点 $\vec{\mathbf{x}}_i$ ，在所有这些随机投影树中寻找其候选邻居。
   - 将这些候选邻居的候选邻居都作为 $\vec{\mathbf{x}}_i$ 的候选邻居。

   这种方法只需要构建少量的随机投影树，就可以得到足够高精度的 kNN 树。

9.2 LargeVis 概率模型

1. LargeVis 根据 kNN 图来定义图 $G = (V, E)$：
   - 顶点集 V ：它就是所有高维空间中的数据点的集合。
   - 边集 E ：它就是 kNN 中所有边的集合。

     其中边的权重 $w_{i,j} = p_{i,j} = \frac{p_{i|j} + p_{j|i}}{2N}$ 。
2. 将该图 G 的结构投影到低维空间保持不变。
   - 定义低维数据点 $\vec{\mathbf{z}}_i$ 和 $\vec{\mathbf{z}}_j$ 存在边的概率为：$P(e_{i,j} = 1) = f(||\vec{\mathbf{z}}_i - \vec{\mathbf{z}}_j||)$ 。

     其中：
     - $e_{i,j} = 1$ 表示点 $\vec{\mathbf{z}}_i$ 和 $\vec{\mathbf{z}}_j$ 存在边。
     - $f(\cdot)$ 为函数，可以为： $f(x) = \frac{1}{1+ax^2}$ 、或者 $f(x) = \frac{1}{1+\exp(x^2)}$ 。
   - 定义数据点 $\vec{\mathbf{z}}_i$ 和 $\vec{\mathbf{z}}_j$ 存在边、且其权重为 $w_{i,j}$ 的概率为：$P(e_{i,j} = w_{i,j}) = P(e_{i,j} = 1)^{w_{i,j}}$ 。
   - 考虑数据集 $\mathbb{D} = \{\vec{\mathbf{x}}_1, \cdots, \vec{\mathbf{x}}_N\}$ ，则对数似然函数为：

     $$\mathcal{L} = \sum_{(i,j)\in E} w_{i,j} \log P(e_{i,j} = 1) + \sum_{(i,j)\in \bar{E}} \gamma \log(1 - P(e_{i,j} = 1))$$

     其中：
     - $(i, j) \in E$ 表示 E 中的边代表的顶点对。这些顶点对也称作 正边 。
     - $\bar{E}$ 表示不存在边的那些顶点对的集合。这些顶点对也称作 负边 。
     - γ 是所有负边的权重。负边的权重是统一的。

3. 事实上在`kNN`图中，正边的数量较少，负边的数量非常庞大，因此计算 $\mathcal{L}$ 的代价较高。

   `LargeVis`利用负采样技术进行优化。

   对图 G 中的每个顶点 i，`LargeVis`仅仅从以 i 为一个端点的负边中随机采样 M 个顶点 $j_1,j_2,\cdots,j_M$ 来计算 $\mathcal{L}$。其中采样概率 $P_n(j)\propto d_j^{0.75}$，d_j 为顶点 j 的度（与它相连的边的数量）。

   则对数似然函数为：

   $$\mathcal{L}=\sum_{(i,j)\in E}w_{i,j}\left[\log P(e_{i,j}=1)+\sum_{m=1}^{M}\mathbb{E}_{j_m\sim P_n(j)}[\gamma\log(1-P(e_{i,j_m}=1))]\right]$$

   其中：j_m 表示随机采样的 M 个顶点。

4. 由于 $\nabla_{\vec{\mathbf{z}}_i}\mathcal{L}$ 中 $w_{i,j}$ 作为乘积项出现的，而网络中边的权重 $w_{i,j}$ 变化范围很大，因此梯度变化会较大。这对训练不利，也就是所谓的梯度剧增和消失问题`gradient explosion and vanishing problem`。

   为了解决这个问题，`LargeVis`对正边也采用随机采样：若正边的权重为 $w_{i,j}$，则将其转换为 $w_{i,j}$ 个权重为`1`的二元边，再随机从这些二元边中进行采样。

   - 每条边的权重都是`1`，这解决了梯度变化范围大的问题。
   - 因为权重较大的边转换得到的二元边更多，被采样的概率越大，所以保证了正确性和合理性。

5. `Largevis`使用异步随机梯度下降来进行训练。

   这在稀疏图上非常有效，因为不同线程采样的边所连接的两个节点很少有重复的，不同线程之间几乎不会产生冲突。

6. `Largevis`每一轮随机梯度下降的时间复杂度为 $O(d\times M)$，其中 M 为负样本个数，d 为低维空间的维度。

   随机梯度下降的步数和样本数量 N 成正比，因此总的时间复杂度为 $O(d\times M\times N)$，与样本数量呈线性关系。