10 chapter 降维

降维是一种非监督学习任务。有时候虽然原始输入是高维数据，但这些输入特征之间有冗余，其本质维度可能很低。降维是将高维数据变成低维表示，同时数据中蕴含的信息尽量保持不变。

本章我们将学习两大类降维技术：基于重构的降维和基于拓扑结构保持的降维。前者我们主要讨论主成分分析和自编码器。后者亦被称为流形学习，我们将学习等距特征映射、局部线性映射、拉普拉斯特征映射和基于 T 分布的随机邻域嵌入。最后我们通过案例，学习 Scikit-Learn 中降维技术的 API。

莎士比亚曾说过：“Brevity is the soul of wit”，中文中我们也有“言简意赅”的说法。在机器学习中，我们用降维来实现言简意赅。降维可去除原始数据中的冗余信息和噪声，留下和任务更相关的简约数据对模型学习更有利。另外降维也意味着我们可以用更少的存储、计算量更少，从而模型的训练更快。

降维可用于对监督学习的特征提取，也常用于数据可视化。降维是非监督学习，因为降维过程中不用样本的标签。当降维是一个大的监督学习任务的一部分时，我们也可以利用监督信

息来指导降维过程，如选择最佳的降维超参数，或降维和监督学习任务一起进行。

降维技术可以主要分为两类：(1) 尽可能多地保留原数据的信息。从重构的角度，使得重构残差最小，本章讨论的主成分分析和自编码器属于这一大类；(2) 尽可能多地保留原始数据的拓扑结构。流形学习属于这一范畴，本章我们将学习等距特征映射、局部线性映射、拉普拉斯特征映射和 T 分布的随机邻域嵌入。

10.1 主成分分析

主成分分析（Principal Component Analysis，PCA）算法是是一种简单的线性降维技术。线性降维是指通过线性变换，即通过矩阵乘法，将高维空间线性映射到一个低维空间。

假设原始数据为D维的$\boldsymbol{x}$，降维后的数据为D'维的$\boldsymbol{z}$，$D' < D$。给定N个原始数据构成矩阵$\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N) \in \mathbb{R}^{D\times N}$，降维后数据构成的矩阵为$\boldsymbol{Z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_N) \in \mathbb{R}^{D'\times N}$，变换矩阵为$\boldsymbol{W} = (\boldsymbol{w_1}, \boldsymbol{w}_2 \dots \boldsymbol{w}_{D'}) \in \mathbb{R}^{D\times D'}$，则

$$\boldsymbol{Z} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}。 \tag{10-1}$$

变换矩阵$\boldsymbol{W}$的每一列为一个投影方向，将原始的D维数据变成 1 维，由于我们只考虑方向，令$\boldsymbol{w}_j$为单位向量，即$\|\boldsymbol{w}_j\|_2 = 1$。在 PCA 中，我们假设$D'$个投影方向中投影方向两两正交，即当$j \neq k$时，$\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{w}_k = 0$。这两个约束合并写成矩阵形式为：

$$\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \boldsymbol{I}, \tag{10-2}$$

其中$\boldsymbol{I}$表示单位矩阵（对角线上元素为 1，其余元素为 0）。

由于$\boldsymbol{W}$为正交变换，我们可以根据数据的低维表示$\boldsymbol{z}$重构原始信号：

$$\hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} + \boldsymbol{W}\boldsymbol{z}, \tag{10-3}$$

其中**$\bar{\boldsymbol{x}}$为训练样本的均值**。为了方便，通常我们假设样本均值$\bar{\mathbf{x}} = \mathbf{0}$（可以通过对数据进行中心化得到）。假设降维后我们希望尽可能保持原始数据的信息，即重构误差最小，因此最小化目标函数：

$$\begin{aligned}
\sum_{i=1}^{N}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 &= \sum_{i=1}^{N}(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^T(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i) \\
&= \sum_{i=1}^{N}\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i - \boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_i^{\mathrm{T}}\boldsymbol{x}_i + \hat{\boldsymbol{x}}_i^{\mathrm{T}}\hat{\boldsymbol{x}}_i\right) \\
&= \sum_{i=1}^{N}\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i - 2\hat{\boldsymbol{x}}_i^{\mathrm{T}}\boldsymbol{x}_i + \hat{\boldsymbol{x}}_i^{\mathrm{T}}\hat{\boldsymbol{x}}_i\right) \\
&= \sum_{i=1}^{N}\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i - 2(\boldsymbol{W}\boldsymbol{z}_i)^{\mathrm{T}}\boldsymbol{x}_i + (\boldsymbol{W}\boldsymbol{z}_i)^{\mathrm{T}}\boldsymbol{W}\boldsymbol{z}_i\right) \\
&= \sum_{i=1}^{N}\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i - 2\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x}_i + \boldsymbol{z}_i^{\mathrm{T}}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W})\boldsymbol{z}_i\right)
\end{aligned} \tag{10-4}$$

$$= \sum_{i=1}^{N} \left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_i - 2\boldsymbol{z}_i^{\mathrm{T}} \boldsymbol{W}^{\mathrm{T}} \boldsymbol{x}_i + \boldsymbol{z}_i^{\mathrm{T}} \boldsymbol{z}_i \right)$$

$$= \sum_{i=1}^{N} \left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_i - 2\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} \boldsymbol{x}_i + \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} \boldsymbol{x}_i \right)$$

$$= \sum_{i=1}^{N} \left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_i - \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} \boldsymbol{x}_i \right)$$

$$= \sum_{i=1}^{N} \left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_i - \boldsymbol{z}_i^{\mathrm{T}} \boldsymbol{z}_i \right)。$$

上述式子中第一项为与$\boldsymbol{W}$无关的常数项，因此最小化重构误差$\sum_{i=1}^{N} \| \boldsymbol{x}_i - \hat{\boldsymbol{x}}_i \|_2^2$等价于最大化降维后各维方差之和$\sum_{i=1}^{N} \boldsymbol{z}_i^{\mathrm{T}} \boldsymbol{z}_i$。

在信号处理中，我们认为信号具有较大的方差，噪声有较小的方差，因此保留那些降维后方差比较大的方向，舍弃那些降维后方差较小的方向，可保留信号中的信息，去掉信号中的噪声。这也是主成分名字的含义，即找出数据里最主要的成分，用数据里最主要的成分来代替原始数据。如图 10-1 所示，第一个主成分是 45°的红色直线，原始数据在红色直线上投影后散得很开（方差大），所以这一个主成分将保留更多信息。第二个主成分为垂直于红线（第一个主成分）的灰色虚线（135°）。当数据点投影到第二个主成分上时，它们挤在样本均值（空心点）附近，方差非常小。所以如果我们要将原始的 2 维数据降至 1 维，应该选红色线，即第一主成分。

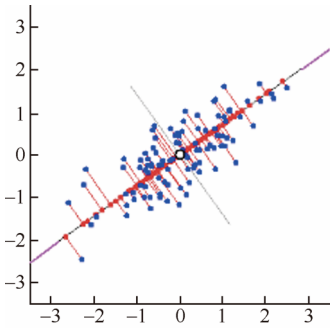


图 10-1　PCA 示例。

降维后数据的协方差矩阵为$\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{W}$，其中$\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$为原始数据的协方差矩阵。最大化降维后各维方差之和$\sum_{i=1}^{N} \boldsymbol{z}_i^{\mathrm{T}} \mathbf{z}_i$，等价最大化该降维后的协方差矩阵的对角线元素之和，即协方差矩阵的迹（Trace），记为$tr(\boldsymbol{W}^{T}\boldsymbol{S}\boldsymbol{W})$。

所以 PCA 的优化目标为：

$$\begin{aligned} &\max tr(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{W}), \\ &s.t.\ \boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \boldsymbol{I}, \end{aligned} \tag{10-5}$$

其中约束条件表示各投影方向正交，且为单位向量。

令协方差矩阵$\boldsymbol{S}$的特征值为$\lambda_1 \geqslant \lambda_2 \ldots \geqslant \lambda_D$，对应的单位特征向量分别为$\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_D$，则将$\boldsymbol{w}_j$组成投影矩阵$\boldsymbol{W}$的第$j$列，该$\boldsymbol{W}$为上述优化问题的解。下面我们详细证明。

假设将数据降至 1 维，这样矩阵$\boldsymbol{W}$成为一个列向量$\boldsymbol{w}_1$，约束条件为$\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{w}_1 = 1$，目标函数$J = tr(\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_1) = \boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_1$，拉格朗日函数$L$为

$$L = \boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_1 - \lambda_1(\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{w}_1 - 1)。$$

拉格朗日函数L对$\boldsymbol{w}_1$求偏导并等于 0，得到：

$$\frac{\partial L}{\partial \boldsymbol{w}_1} = 2\boldsymbol{S}\boldsymbol{w}_1 - 2\lambda_1\boldsymbol{w}_1 = \boldsymbol{0}。$$

所以

$$\boldsymbol{S}\boldsymbol{w}_1 = \lambda_1\boldsymbol{w}_1, \tag{10-7}$$

即λ_1为协方差矩阵$\boldsymbol{S}$的特征值，$\boldsymbol{w}_1$是对应的特征向量。

此时目标函数为$J = tr(\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_1) = \boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_1 = \boldsymbol{w}_1^{\mathrm{T}}\lambda_1\boldsymbol{w}_1 = \lambda_1$。要使得目标函数值最大，$\lambda_1$为协方差矩阵$\boldsymbol{S}$最大的特征值，$\boldsymbol{w}_1$为对应的特征向量。

接着求第二个投影方向$\boldsymbol{w}_2$。约束条件为$\boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{w}_2 = 1$，$\boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{w}_1 = 0$，目标函数为$J = tr(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{W}) = \boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_1 + \boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_2$，拉格朗日函数$L$为

$$L = \boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_1 + \boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_2 - \lambda_2(\boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{w}_2 - 1) - \lambda_{2,1}\boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{w}_1。$$

拉格朗日函数L对$\boldsymbol{w}_2$求偏导并等于 0，得到：

$$\frac{\partial L}{\partial \boldsymbol{w}_2} = 2\boldsymbol{S}\boldsymbol{w}_2 - 2\lambda_2\boldsymbol{w}_2 - \lambda_{2,1}\boldsymbol{w}_1 = \boldsymbol{0}。 \tag{10-8}$$

将式（10-8）两边同乘以$\boldsymbol{w}_1^{\mathrm{T}}$，得到

$$2\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_2 - 2\lambda_2\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{w}_2 - \lambda_{2,1}\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{w}_1 = 0。$$

此等式左边前两项等于 0，所以第三项也必须等于 0，而$\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{w} = 1$，**所以**$\lambda_{2,1} = 0$。因此式（10-8）变成

$$\boldsymbol{S}\boldsymbol{w}_2 = \lambda_2\boldsymbol{w}_2,$$

即λ_2为协方差矩阵$\boldsymbol{S}$的特征值，$\boldsymbol{w}_2$是对应的特征向量。

此时目标函数为$J = tr(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{W}) = \boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_1 + \boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{S}\boldsymbol{w}_2 = \lambda_1 + \lambda_2$。要使得目标函数值最大，在给定$\lambda_1$为协方差矩阵$\boldsymbol{S}$最大的特征值的条件下，$\lambda_2$为$\boldsymbol{S}$的第二大特征值，$\boldsymbol{w}_2$为对应的特征向量。

依次类推，λ_d为协方差矩阵$\boldsymbol{S}$的第d大特征值，$\boldsymbol{w}_d$为对应的特征向量。如果想降至D'维，则取$\boldsymbol{S}$的前D'个特征值对应的特征向量，构成投影矩阵$\boldsymbol{W}$。

算法 10-1：PCA

输入：N个D维向量$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$，

降维后的维度D'

过程：

1．对所有样本进行中心化：$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{x}_k$；

2．计算输入样本的协方差矩阵$\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$；

3．对协方差矩阵$\boldsymbol{S}$做特征值分解；

输出：前D'个最大特征值对应的特征向量$\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{D'}$，构成变换矩阵$\boldsymbol{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{D'})$。

当输入数据维度D很大时，协方差矩阵会非常大$(D\times D)$，对协方差矩阵$\boldsymbol{S}$做特征值分解计算量大。此时可以通过对$\boldsymbol{X}$进行奇异值分解（Singular value decomposition, SVD）实现得到$\boldsymbol{S}$的特征向量。$\boldsymbol{X}$的奇异值分解为：

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}},$$

其中$\boldsymbol{U}$的列即为协方差矩阵$\boldsymbol{S}$的特征向量，从而可以得到 PCA 投影矩阵为$\boldsymbol{W} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{D'})$。

PCA 比较简单，除了要降维后的特征维度D'（要保留的主成分的数目），没有其他要设置的参数。相比其他非线性降维技术，PCA 速度快，但性能有时差强人意，所以 PCA 也常作为其

他一些降维技术（如 T-NSE）的预处理步骤。

通过核技巧引入核函数，可将 PCA 扩展到核化的 PCA（Kernel PCA），可实现非线性降维。

10.2 自编码器

自编码器（Auto-Encoders，AE）是一种采用神经网络实现学习一个恒等函数（Identity Function），用于重构原始数据。如图 10-2 所示，自编码器由两个部分组成：

- **编码器网络**：将原始数据编码为低维隐向量；
- **解码器网络**：从隐编码中重构出原始数据。

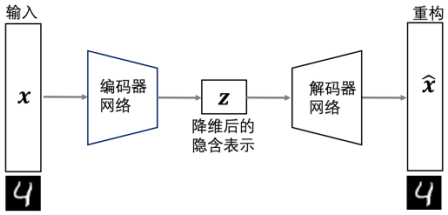


图 10-2　自编码器结构示意图。

与 PCA 相比，自编码器相当于用一个编码器神经网络表示的函数$g(\boldsymbol{x})$代替 PCA 中的$\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x}$，用解码器神经网络$f(\boldsymbol{z})$代替 PCA 中的$\boldsymbol{W}\boldsymbol{z}$。因此自编码器的目标函数为：

$$J(g,f)=\sum_{i=1}^{N}||\boldsymbol{x}_i-\hat{\boldsymbol{x}}_i||_2^2+R(g)。\tag{10-9}$$

目标函数中第一项同 PCA，表示训练数据集上的重构残差平方和；第二项为正则项，用于控制编码器的复杂度。正则项可以为隐含层响应神经元的数目（稀疏自编码器）、神经网络连接权重的 L2 正则或 L1 正则。正则项尤其在编码器网络中隐含层结点数目超过输入数据维度时是必要的。

降噪自编码器（Denoising Autoencoder，DAE）对输入数据$\boldsymbol{x}$添加噪声，加噪后的数据$\tilde{\boldsymbol{x}}$做为自编码器的编码器的输入，解码器还是尽可能重构原始未被损坏的数据$\boldsymbol{x}$。噪声可以是纯高斯噪声，也可以是随机丢弃输入层的某个特征。通过对加入噪声的数据重构不含噪声的数据，自编码器强迫网络学习到更加鲁棒的性特征。DAE 的过程如图 10-3 所示。

根据原始数据的特点，实现编码器和解码器的神经网络可为全连接神经网络、卷积神经网络或循环神经网络。

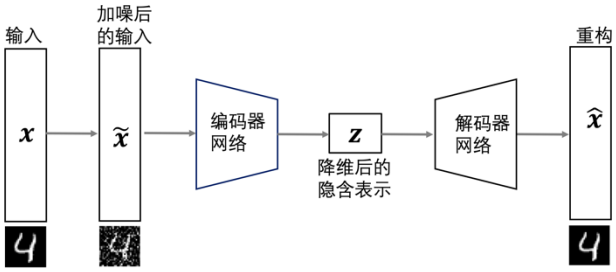


图 10-3　降噪自编码器结构示意图。

1．全连接自编码器

图 10-4 给出了一个全连接神经网络实现自编码器的例子。输入图像为28×28 = 784，经过 4 个隐含层，隐含层结点数目分别为 1000、500、250 和 30，得到原始数据降维后的隐含表示为 30 维。这 30 维向量再经过解码器部分，重构 784 维信号。通常解码器和编码器对应层的权重对称（矩阵转置），这样模型的参数少一倍，不过这种对称参数约束不是必须的。

自编码器训练时，通常采用逐层训练的方式确定模型初始参数，最后再在初始值的基础上进行参数细调，被称为栈式自编码器（Stack Autoencoder）。如在如图 10-4 所示的模型中，首先训练 784→1000→784 的自编码器；而后已经固定已经训练好的参数和 1000 维的结果，再训练第二个自编码器：1000→500→1000；而后固定已经训练好的参数和训练的中间层结果，训练第三个自编码器：500→250→500，固定参数和中间隐层的结果。此时，前 3 层的参数已经训练完毕，此时，最后一层接一个回归器，将整体网络使用反向传播进行训练，对参数进行微调。

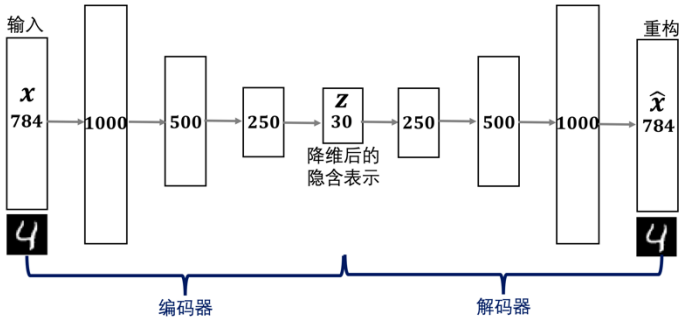


图 10-4　全连接自编器示例。

2．卷积自编码器（Convolutional Auto Encoder, CAE）

如果输入数据是图像，采用卷积神经网络作为编码器和解码器更高效。通常编码器由二维卷积层和池化层（下采样层）堆叠而成，起到降维作用。而解码器由反卷积层（其实也是卷积层）和反池化层（上采样层）堆叠组成，重构原始输入。

- 反卷积

反卷积其实也是卷积。为了理解这一点，我们先复习一下卷积的数学操作。

卷积操作可写成矩阵相乘，如对4×4的输入$\boldsymbol{X}$和3×3的卷积核$\boldsymbol{C}$，

$$\boldsymbol{X}=\begin{bmatrix}x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4}\\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4}\\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4}\\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4}\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}c_{1,1} & c_{1,2} & c_{1,3}\\ c_{2,1} & c_{2,2} & c_{2,3}\\ c_{3,1} & c_{3,2} & c_{3,3}\end{bmatrix},$$

卷积结果$\boldsymbol{y}$为：

$$\boldsymbol{y}=\begin{bmatrix}y_1\\ y_2\\ y_3\\ y_4\end{bmatrix}$$

$$=\begin{bmatrix}c_{1,1} & c_{1,2} & c_{1,3} & 0 & c_{2,1} & c_{2,2} & c_{2,3} & 0 & c_{3,1} & c_{3,2} & c_{3,3} & 0 & 0 & 0 & 0 & 0\\ 0 & c_{1,1} & c_{1,2} & c_{1,3} & 0 & c_{2,1} & c_{2,2} & c_{2,3} & 0 & c_{3,1} & c_{3,2} & c_{3,3} & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & c_{1,1} & c_{1,2} & c_{1,3} & 0 & c_{2,1} & c_{2,2} & c_{2,3} & 0 & c_{3,1} & c_{3,2} & c_{3,3} & 0\\ 0 & 0 & 0 & 0 & 0 & c_{1,1} & c_{1,2} & c_{1,3} & 0 & c_{2,1} & c_{2,2} & c_{2,3} & 0 & c_{3,1} & c_{3,2} & c_{3,3}\end{bmatrix}。$$

4×1的向量$\boldsymbol{y}$可以变形为2×2的矩阵。

反卷积的操作相当于上述$\boldsymbol{y}$左乘一个矩阵$\boldsymbol{C}'^{\mathrm{T}}$，得到16×1的向量，再变形为4×4的矩阵，和输入信号维度一致。因此反卷积也被称为**转置的卷积（Transposed Convolution）**。

$$\begin{bmatrix}x_{1,1}\\ x_{1,2}\\ x_{1,3}\\ x_{1,4}\\ x_{2,1}\\ x_{2,2}\\ x_{2,3}\\ x_{2,4}\\ x_{3,1}\\ x_{3,2}\\ x_{3,3}\\ x_{3,4}\\ x_{4,1}\\ x_{4,2}\\ x_{4,3}\\ x_{4,4}\end{bmatrix}=\begin{bmatrix}c'_{1,1} & 0 & 0 & 0\\ c'_{1,2} & c'_{1,1} & 0 & 0\\ c'_{1,3} & c'_{1,2} & 0 & 0\\ 0 & c'_{1,3} & 0 & 0\\ c'_{2,1} & 0 & c'_{1,1} & 0\\ c'_{2,2} & c'_{2,1} & c'_{1,2} & c'_{1,1}\\ c'_{2,3} & c'_{2,2} & c'_{1,3} & c'_{1,2}\\ 0 & c'_{2,3} & 0 & c'_{1,3}\\ c'_{3,1} & 0 & c'_{2,1} & 0\\ c'_{3,2} & c'_{3,1} & c'_{2,2} & c'_{2,1}\\ c'_{3,3} & c'_{3,2} & c'_{2,3} & c'_{2,3}\\ 0 & c'_{3,3} & c'_{2,3} & 0\\ 0 & 0 & c'_{3,1} & 0\\ 0 & 0 & c'_{3,2} & c'_{3,1}\\ 0 & 0 & c'_{3,3} & c'_{3,2}\\ 0 & 0 & 0 & c'_{3,3}\end{bmatrix}\begin{bmatrix}y_1\\ y_2\\ y_3\\ y_4\end{bmatrix}。$$

- 反池化

池化的作用是降维，常见的池化有最大池化和平均池化。池化层不需要训练参数。平均池化对应的反池化操作为反平均池化，将分辨率扩大，复制均值多份即可。反最大池化可以记录池化时最大值的位置，然后扩大分辨率后最大值位置复制最大值，其他位置置 0；或者每个位置都复制最大值。

注意：在 PyTorch 框架下，卷积自编码器的解码器既可以通过转置卷积实现，也可以先上采样（升维，没有参数），然后再普通卷积实现。

3．LSTM 自编码器

如果输入的是序列，可以使用 LSTM 编码器将输入序列转换成包含整个序列信息的单个向量，然后重复该向量T次，（T为输出序列中的时间步数），并运行一个 LSTM 解码器将该恒定序列转换成目标序列。LSTM 自动编码器也被称为 Seq2Seq（Sequence to Sequence），表

示可以从序列到序列。Seq2Seq 结构存在两个问题，一是将整个序列的信息压缩为一个低维向量，会造成信息损失；二是如果序列过长，长程依赖很难被完全学习，从而使得准确率下降。因此，人们研究 Seq2Seq 结构与注意力机制（Attention Mechanism）相结合，在此不再详述。

对自编码器结构稍加改造，可得到可以产生新样本的生成式模型，如变分自编码器（Variational AutoEncoders，VAE）、对抗自编码器（Adversarial autoencoder，AAE）等。

10.3 多维缩放

多维缩放（Multiple Dimensional Scaling，MDS）的目标是在降维的过程中保持数据之间的距离，即高维空间中的距离关系与低维空间中距离关系保持不变。我们用矩阵$\boldsymbol{D} \in \mathbb{R}^{N\times N}$表示$N$个样本的两两距离，并且假设在低维空间中的距离是欧氏距离，降维后的数据表示为$\boldsymbol{z}_i$，则有

$$d_{i,j}^2 = ||\boldsymbol{z}_i - \boldsymbol{z}_j||_2^2 = ||\boldsymbol{z}_i||_2^2 + ||\boldsymbol{z}_j||_2^2 - 2\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j。 \tag{10-10}$$

不失一般性，我们假设低维空间中的实例点是中心化的，即

$$\sum_{i=1}^{N} \boldsymbol{z}_i = 0。 \tag{10-11}$$

对上面式子两边求和，有

$$\begin{aligned}\sum_{i=1}^{N} d_{i,j}^2 &= \sum_{i=1}^{N} \|\boldsymbol{z}_i\|_2^2 + N||\boldsymbol{z}_j||^2 + 2\boldsymbol{z}_j\left(\sum_{i=1}^{N}\boldsymbol{z}_i\right)\\ &= \sum_{i=1}^{N} \|\boldsymbol{z}_i\|_2^2 + N||\boldsymbol{z}_j||^2 + 2\boldsymbol{z}_j\times 0\\ &= \sum_{i=1}^{N} \|\boldsymbol{z}_i\|_2^2 + N||\boldsymbol{z}_j||^2。\end{aligned} \tag{10-12}$$

类似可得到：

$$\sum_{j=1}^{N} d_{i,j}^2 = \sum_{j=1}^{N} \|\boldsymbol{z}_j\|_2^2 + N||\boldsymbol{z}_i||^2。 \tag{10-13}$$

所以：

$$\begin{aligned}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}^2 &= \sum_{i=1}^{N}\left(N\|\boldsymbol{z}_i\|_2^2 + \sum_{j=1}^{N}\|\boldsymbol{z}_j\|_2^2\right)\\ &= N\sum_{i=1}^{N}\|\boldsymbol{z}_i\|_2^2 + N\sum_{i=j}^{N}\|\boldsymbol{z}_j\|_2^2\\ &= 2N\sum_{i=1}^{N}\|\boldsymbol{z}_i\|_2^2。\end{aligned} \tag{10-14}$$

对上面公式（10-12）、（10-13）两边同除以N，公式（10-14）两边同除以N^2，得到

$$\frac{1}{N}\sum_{i=1}^{N} d_{i,j}^2 = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{z}_i\|_2^2 + \|\boldsymbol{z}_j\|^2,$$

$$\frac{1}{N}\sum_{j=1}^{N} d_{i,j}^2 = \frac{1}{N}\sum_{j=1}^{N} \|\boldsymbol{z}_j\|_2^2 + \|\boldsymbol{z}_i\|^2, \tag{10-15}$$

$$\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}^2 = \frac{2}{N}\sum_{i=1}^{N} \|\boldsymbol{z}_i\|_2^2 。$$

定义内积矩阵$\boldsymbol{B} = \boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}} \in \mathbb{R}^{N\times N}$，即$b_{i,j} = \boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j$，则

$$\begin{aligned} d_{i,j}^2 &= \|\boldsymbol{z}_i\|_2^2 + \|\boldsymbol{z}_j\|_2^2 - 2\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j \\ &= b_{i,i} + b_{j,j} - 2b_{i,j}。 \end{aligned} \tag{10-16}$$

从而得到用$\boldsymbol{D}$表示$\boldsymbol{B}$：

$$\begin{aligned} b_{i,j} &= -\frac{1}{2}(d_{i,j}^2 - b_{i,i} - b_{j,j}) \\ &= -\frac{1}{2}\left(d_{i,j}^2 - \left(\frac{1}{N}\sum_{j=1}^{N} d_{i,j}^2 - \frac{1}{N}\sum_{j=1}^{N}\|\boldsymbol{z}_j\|_2^2\right) - \left(\frac{1}{N}\sum_{i=1}^{N} d_{i,j}^2 - \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{z}_i\|_2^2\right)\right) \\ &= -\frac{1}{2}\left(d_{i,j}^2 - \frac{1}{N}\sum_{j=1}^{N} d_{i,j}^2 - \frac{1}{N}\sum_{i=1}^{N} d_{i,j}^2 + \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{z}_i\|_2^2 + \frac{1}{N}\sum_{j=1}^{N}\|\boldsymbol{z}_j\|_2^2\right) \\ &= -\frac{1}{2}\left(d_{i,j}^2 - \frac{1}{N}\sum_{j=1}^{N} d_{i,j}^2 - \frac{1}{N}\sum_{i=1}^{N} d_{i,j}^2 + \frac{2}{N}\sum_{i=1}^{N}\|\boldsymbol{z}_i\|_2^2\right) \\ &= -\frac{1}{2}\left(d_{i,j}^2 - \frac{1}{N}\sum_{j=1}^{N} d_{i,j}^2 - \frac{1}{N}\sum_{i=1}^{N} d_{i,j}^2 + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}^2\right)。 \end{aligned}$$

用矩阵表示为$\boldsymbol{B} = -\frac{1}{2}\boldsymbol{J}\boldsymbol{D}\boldsymbol{J}$，其中

$$\boldsymbol{J} = \boldsymbol{I} - \frac{1}{N}\boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}, \boldsymbol{1} = (1,1,\ldots,1)。$$

$\boldsymbol{D}\boldsymbol{J}$为从$\boldsymbol{D}$中的每个元素里减去列均值，$\boldsymbol{J}\boldsymbol{D}\boldsymbol{J}$作用就是在此基础上再减去行均值，因此中心化矩阵的作用就是把元素的中心平移到坐标原点。注意该中心化过程相当于平移，并不会改变低维度点之间的距离。

对矩阵$\boldsymbol{B}$做特征分解，得到

$$\boldsymbol{B} = \boldsymbol{V}\Lambda\boldsymbol{V}^{\mathrm{T}}, \tag{10-17}$$

其中$\boldsymbol{\Lambda}$是由$\boldsymbol{B}$的特征值生成的对角矩阵，$\boldsymbol{V}$是特征向量作为列的矩阵。

如果我们希望降数据降至D'维空间，那么选择前$\boldsymbol{D'}$个最大特征值及对应的特征向量，得到$\boldsymbol{\Lambda}_{D'}$和$\boldsymbol{V}_{D'}$，则降维后的特征表示为

$$\boldsymbol{Z} = \boldsymbol{V}_{D'}\boldsymbol{\Lambda}_{D'}^{1/2}。 \tag{10-18}$$

10.4 等度量映射

MDS 需要输入高维空间中样本点之间的距离矩阵$\boldsymbol{D}$。如果高维空间中样本点之间的距离用欧氏距离表示，则 MDS 为线性降维；如果矩阵是根据流形得到的测地距离，则 MDS 为非线性降维。

流形（Manifold）是一种拓扑结构，如图 10-5 所示的三维空间中曲面就是一种流形（注意不可闭合）。流形学习理论认为，在高维空间直接计算距离是不合适的。如图 10-5 中，我们要计算 A 中两点的距离，不是如图 A 所示的两点之间的直线距离（蓝色虚线长度），而是如图 B 所示的测地距离（红色实曲线长度）。

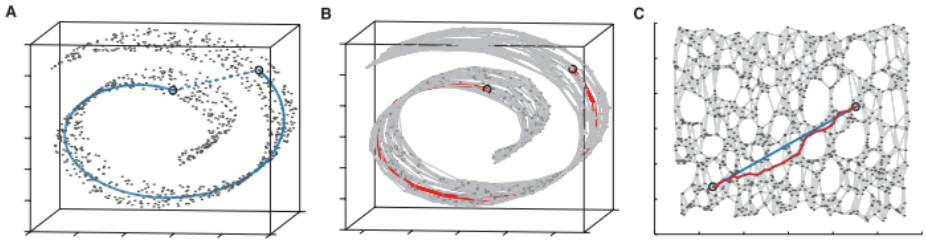


图 10-5　Isomap 示例[1]。虚线为欧氏距离，实线为测地距离，第三幅图像为 Isomap 根据测地距离，降维后的嵌入空间。

流形空间在局部上是欧氏空间的同胚空间，可以在局部进行欧氏距离的计算，再在整个流形结构上累加，获得流形曲面上的测地距离。对测地距离再进行多维缩放，得到等度量映射算法（Isometric Mapping，IsoMap）。

具体实现时，IsoMap 对于每个样本点，基于欧氏距离构造邻接图。邻接图中每个样本为一个图中一个结点，对每个样本点，选取该点的K个近邻近点，在该样本点和其K近邻之间建立边，边的权重为两点之间的欧氏距离，样本点与非邻近点的距离定义为无穷远（没有边）。对不直接相连的两个样本点，它们之间的测地距离为邻接图上的最短路径，可采用图的最短路径算法（如 Dijkstra 算法、Floyd 等算法）获得所有流形曲面上点与点之间的距离。

10.5 局部线性嵌入

局部线性嵌入（Locally Linear Embedding，LLE）假设在高维空间中，可以样本的邻近样本的线性加权可以重构样本，即$\boldsymbol{x}_i = \sum_{j\in\mathcal{N}_i} w_{j,i}\,\boldsymbol{x}_j$，其中$\mathcal{N}_i$表示$\boldsymbol{x}_i$的邻域上点的集合。LLE 同时假设低维空间里面，这种线性重构关系能够保持，即

$$\boldsymbol{z}_i = \sum_{j\in\mathcal{N}_i} w_{j,i}\,\boldsymbol{z}_j。 \tag{10-19}$$

因此 LLE 的第一步是在高维空间中，计算每两个样本点之间的距离，确定每个样本的K近邻$\mathcal{N}_i$。注意当样本数N或样本维度D很大时，可能需要采用高效的K近邻近似算法得到K近邻（如 K-D 树）。样本及其K近邻表示了原始高维空间中的数据结构。

LLE 的第二步是在样本K近邻近的基础上，求出每个样本点的局部线性组合权重$\boldsymbol{w}_i = (w_{1,i}, w_{2,i}, \ldots w_{K,i})^{\mathrm{T}}$，即：

$$\min \sum_{i=1}^{N} \left\| \boldsymbol{x}_i - \sum_{j \in \mathcal{N}_i} w_{j,i}\, \boldsymbol{x}_j \right\|_2^2, \quad \text{s.t.} \sum_{j \in \mathcal{N}_i} w_{j,i} = 1。 \tag{10-20}$$

对第i个样本点，其K近邻近的组合系数可独立求解，因此单个样本点的目标函数为：

$$\begin{aligned} J(\boldsymbol{w}_i) &= \left\| \boldsymbol{x}_i - \sum_{j \in \mathcal{N}_i} w_{j,i}\, \boldsymbol{x}_j \right\|_2^2 \\ &= \left\| \sum_{j \in \mathcal{N}_i} w_{j,i}\, \boldsymbol{x}_i - \sum_{j \in \mathcal{N}_i} w_{j,i}\, \boldsymbol{x}_j \right\|_2^2 \\ &= \left\| \sum_{j \in \mathcal{N}_i} w_{j,i}\, (\boldsymbol{x}_i - \boldsymbol{x}_j) \right\|_2^2 。 \end{aligned} \tag{10-21}$$

上述问题其实就是一个最小二乘问题，我们可以直接将最小二乘的解代入，再将权重进行归一（和为 1）。

也可以写成矩阵形式。令$\boldsymbol{X}_i = \underbrace{[\boldsymbol{x}_i, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_i]}_{K}$，$\boldsymbol{N}_i = \underbrace{[\boldsymbol{x}_j]}_{j \in \mathcal{N}_i}$均为$D \times K$的矩阵，$\boldsymbol{S}_i = (\boldsymbol{X}_i - \boldsymbol{N}_i)^{\mathrm{T}}(\boldsymbol{X}_i - \boldsymbol{N}_i)$为局部协方差矩阵，则

$$J(\boldsymbol{w}_i) = \boldsymbol{w}_i^{\mathrm{T}}(\boldsymbol{X}_i - \boldsymbol{N}_i)^{\mathrm{T}}(\boldsymbol{X}_i - \boldsymbol{N}_i)\boldsymbol{w}_i = \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{S}_i\boldsymbol{w}_i。$$

再加上约束$\sum_{j \in \mathcal{N}_i} w_{ij} = 1$，用朗格朗日乘子法求解，得到朗格朗日函数为：

$$L(\boldsymbol{w}_i) = J(\boldsymbol{w}_i) + \lambda(\boldsymbol{w}_i^{\mathrm{T}}\mathbf{1} - 1),$$

其中$\mathbf{1} = (1,1,\ldots,1)$表示$K$维全 1 的向量。

拉格朗日函数对权重求导，并令其为 0，得到：

$$2\boldsymbol{S}_i\boldsymbol{w}_i + \lambda\, \mathbf{1} = \mathbf{0},$$

从而，

$$\boldsymbol{w}_i = \frac{1}{2}\lambda \boldsymbol{S}_i^{-1}\mathbf{1},$$

其中$-\frac{1}{2}\lambda$为常数。我们可以通过对$\boldsymbol{w}_i$根据约束$\boldsymbol{w}_i^{\mathrm{T}}\mathbf{1} = 1$对$\boldsymbol{w}_i$做归一化，得到最终的$\boldsymbol{w}_i$：

$$\boldsymbol{w}_i = -\frac{\boldsymbol{S}_i^{-1}\mathbf{1}}{\mathbf{1}\boldsymbol{S}_i^{-1}\mathbf{1}}。 \tag{10-22}$$

上面式子中分母$\mathbf{1}\boldsymbol{S}_i^{-1}\mathbf{1}$其实是对$\boldsymbol{S}_i^{-1}$的所有元素求和，而其分子$\boldsymbol{S}_i^{-1}\mathbf{1}$是对$\boldsymbol{S}_i^{-1}$的每行求和后得到的列向量。

LLE 的第三步是根据每个样本点的重构权重，计算样本的低维表示，即

$$min \sum_{i=1}^{N} \left\| \boldsymbol{z}_i - \sum_{j \in \mathcal{N}_i} w_{ij}\, \boldsymbol{z}_j \right\|_2^2 。 \tag{10-23}$$

为了防止得到无意义的解$\boldsymbol{z}_i = \mathbf{0}$，我们对低维表示增加标准化约束（$\mathbf{0}$均值、单位矩阵的

协方差矩阵）：

$$\sum_{i=1}^{N} \boldsymbol{z}_i = \mathbf{0}, \qquad \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{z}_i \boldsymbol{z}_i^{\mathrm{T}} = \mathbf{I}。 \tag{10-24}$$

我们将重构权重矩阵写成一个大的$N\times N$的稀疏矩阵$\boldsymbol{W}$：

$$W_{j,i} = \begin{cases} w_{j,i} & j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases},$$

这样目标函数为：

$$\begin{aligned} J(\boldsymbol{Z}) &= \sum_{i=1}^{N} ||\boldsymbol{z}_i - \sum_{j=1}^{N} w_{j,i}\, \boldsymbol{z}_j||_2^2 \\ &= \sum_{i=1}^{N} ||\boldsymbol{Z}\boldsymbol{i}_i - \boldsymbol{Z}\boldsymbol{w}_i||_2^2 \\ &= \sum_{i=1}^{N} ||\boldsymbol{Z}(\boldsymbol{i}_i - \boldsymbol{w}_i)||_2^2 = tr(\boldsymbol{Z}(\boldsymbol{I}-\boldsymbol{W})(\boldsymbol{I}-\boldsymbol{W})^T\boldsymbol{Z}^T), \end{aligned} \tag{10-25}$$

其中$\boldsymbol{i}_i$表示矩阵$\mathbf{I}$的第i列，$\boldsymbol{w}_i$表示矩阵$\boldsymbol{W}$的第i列，矩阵的迹$tr(\boldsymbol{Z}(\boldsymbol{I}-\boldsymbol{W})(\boldsymbol{I}-\boldsymbol{W})^{\mathrm{T}}\boldsymbol{Z}^{\mathrm{T}})$为矩阵的对角线元素之和。

令$\boldsymbol{M} = \boldsymbol{I} - \boldsymbol{W}$，则优化问题变为：

$$\begin{aligned} &\min tr(\boldsymbol{Z}\boldsymbol{M}\boldsymbol{M}^{\mathrm{T}}\boldsymbol{Z}^{\mathrm{T}}), \\ &\text{s.t. } \boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}} = N\boldsymbol{I}。 \end{aligned} \tag{10-26}$$

该问题形式同 PCA 中的优化问题（10-5）类似，因此解$\boldsymbol{Z}$的列为是$\boldsymbol{M}$的特征向量，只是这里求矩阵的迹的最小值。所以为了将数据降到D'维，我们只需要取$\boldsymbol{M}$的最小的个非零D'特征值对应的特征向量。一般最小的特征值为 0（对应的特征向量为全 1），我们将其舍弃，取从第 2 小到第D'小的特征值对应的特征向量。

算法 10-2：LLE

输入：样本集$\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$，最近邻数$K$，降维到的维数$D'$

输出：低维样本集矩阵$\boldsymbol{Z}$

1．for $i = 1 \quad to \quad N$，以欧氏距离作为度量，计算和$\boldsymbol{x}_i$最近的的K个最近邻$\mathcal{N}_i$；

2．for $i = 1 \quad to \quad N$，计算局部协方差矩阵$\boldsymbol{S}_i = (\boldsymbol{X}_i - \boldsymbol{N}_i)^{\mathrm{T}}(\boldsymbol{X}_i - \boldsymbol{N}_i)$，其中$\boldsymbol{X}_i = \underbrace{[\boldsymbol{x}_i, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_i]}_{K}$，$\boldsymbol{N}_i = \underbrace{[\boldsymbol{x}_j]}_{j\in\mathcal{N}_i}$，并求出对应的权重系数向量：$\boldsymbol{w}_i = -\frac{\boldsymbol{S}_i^{-1}\mathbf{1}}{\mathbf{1}\boldsymbol{S}_i^{-1}\mathbf{1}}$；

3．由权重系数向量组成权重系数矩阵$\boldsymbol{W}$，计算矩阵$\boldsymbol{M} = (\boldsymbol{I}-\boldsymbol{W})(\boldsymbol{I}-\boldsymbol{W})^{\mathrm{T}}$；

4．计算矩阵$\boldsymbol{M}$的前$D'+1$个最小的特征值，并计算这$D'+1$个特征值对应的特征向量：$\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{D'+1}$；

5．由第 2 个特征向量到第$D'+1$个特征向量所张成的矩阵即为输出低维样本集矩阵$\boldsymbol{Z} = (\boldsymbol{z}_2, \boldsymbol{z}_3, \ldots, \boldsymbol{z}_{D'+1})$。

LLE 的超参数有样本的近邻数K。当K取值较小时，近邻个数太少，可能不能很好地反映数据的拓扑结构。当K取值太大，近邻个数太多，邻域过大，不再满足局部线性假设。Scikit-Learn

中默认值为 5。

10.6 拉普拉斯特征映射

拉普拉斯特征映射（Laplace Eigenmaps，LE）看问题的角度和 LLE 有些相似，也是保持流形的局部结构。如果在原始高维空间中两个样本点$\boldsymbol{x}_i$和$\boldsymbol{x}_j$相似，那么在降维后的目标空间中，二者应该尽量接近。

拉普拉斯特征映射用图$\mathcal{G}=(\mathcal{V},\mathcal{E})$来构建数据之间的关系。图的结点集合$\mathcal{V}$为所有样本点，如果两个样本点相似，在图中用以一条边连接这两个点。这里相似可用K近邻表示，即首先计算每个样本点的K个邻居，每个点与其最近的K个邻居连上边，边的权重为两个样本点之间的相似度（亦可为其他合理的相似度度量）：

$$w_{i,j}=exp\left(-\frac{||\boldsymbol{x}_i-\boldsymbol{x}_j||_2^2}{\sigma^2}\right)。\tag{10-27}$$

我们的目标是让相似的数据样例在降维后的目标子空间里仍旧尽量接近，故目标函数为：

$$min\sum_{i,j}||\boldsymbol{z}_i-\boldsymbol{z}_j||_2^2 w_{i,j}。\tag{10-28}$$

对上述目标函数进行变换：

$$\begin{aligned}
\sum_{i=1}^{N}\sum_{j=1}^{N}||\boldsymbol{z}_i-\boldsymbol{z}_j||_2^2 w_{i,j} &= \sum_{i=1}^{N}\sum_{j=1}^{N}(\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_i-2\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j+\boldsymbol{z}_j^{\mathrm{T}}\boldsymbol{z}_j)w_{i,j}\\
&=\sum_{i=1}^{N}\left(\sum_{j=1}^{N}w_{i,j}\right)\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_i+\sum_{j=1}^{N}\left(\sum_{i=1}^{N}w_{i,j}\right)\boldsymbol{z}_j^{\mathrm{T}}\boldsymbol{z}_j-2\sum_{i=1}^{N}\sum_{j=1}^{N}\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j w_{i,j}\\
&=2\sum_{i=1}^{N}d_{i,i}\,\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_i-2\sum_{i=1}^{N}\sum_{j=1}^{N}\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j w_{i,j}\\
&=2\sum_{i=1}^{N}(\sqrt{d_{i,i}}\boldsymbol{z}_i)^{\mathrm{T}}(\sqrt{d_{i,i}}\boldsymbol{z}_i)-2\sum_{i=1}^{N}\boldsymbol{z}_i^{\mathrm{T}}\left(\sum_{j=1}^{N}\boldsymbol{z}_j\,w_{i,j}\right)\\
&=2tr(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Z})-2\sum_{i=1}^{N}\boldsymbol{z}_i^{\mathrm{T}}(\boldsymbol{Z}\boldsymbol{W})_i\\
&=2tr(\boldsymbol{Z}\boldsymbol{D}\boldsymbol{Z})-2tr(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{Z})\\
&=2tr[\boldsymbol{Z}^{\mathrm{T}}(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{Z}]\\
&=2tr(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{Z}),
\end{aligned}$$

其中$\boldsymbol{W}$为图的邻接关系矩阵，对角矩阵$\boldsymbol{D}$是图的度矩阵：$d_{i,i}=\sum_{j=1}^{N}w_{i,j}$，$\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{W}$为图的**拉普拉斯矩阵**。

变换后的拉普拉斯特征映射优化的目标函数如下：

$$\begin{aligned}
&\min tr(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{Z}),\\
&s.t.\ \ \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Z}=\boldsymbol{I},
\end{aligned}\tag{10-29}$$

其中限制条件$\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Z}=\boldsymbol{I}$是为了保证最优解有意义（否则$\boldsymbol{z}_i=\boldsymbol{0}$是最优解）和去掉嵌入向量中的缩放因子。

该问题形式同 LLE 的优化问题（10-26）类似。因为$\boldsymbol{D}$是对角阵，采用类似 PCA 单独求解$\boldsymbol{Z}$的列向量时，乘以$\boldsymbol{D}$相当于乘以对应的标量$d_{i,i}$，所以$\boldsymbol{Z}$的列向量也是矩阵$\boldsymbol{L}$的从第 2 小到第D'小的特征值对应的特征向量。

10.7 基于 T 分布的随机邻域嵌入

基于 T 分布的随机邻域嵌入（T-distributed stochastic neighbor embedding, T-SNE）是一种非线性降维算法，非常适用于高维数据降维到 2 维或者 3 维进行可视化。

T-SNE 的基本思想是在高维空间相似的数据点，映射到低维空间也相似。T-SNE 中，相似度用条件概率来表示。在高维空间中，给定$\boldsymbol{x}_i$的条件下，$\boldsymbol{x}_j$的条件概率$p_{j|i}$与$\boldsymbol{x}_j$和$\boldsymbol{x}_i$之间的欧氏距离有关：

$$\begin{aligned} p_{j|i}&=\frac{s_{i,j}}{\sum_{j'\neq i}s_{i,j'})},\\ s_{i,j}&=\exp\left(\frac{-\left\|\boldsymbol{x}_i-\boldsymbol{x}_j\right\|_2^2}{2\sigma_i^2}\right), \end{aligned} \tag{10-30}$$

其中$p_{j|i}$表示给定中心为$\boldsymbol{x}_i$，方差为σ_i^2的高斯分布下，$\boldsymbol{x}_j$和$\boldsymbol{x}_i$有多接近。这里参数σ_i对不同的点$\boldsymbol{x}_i$取值不一样，目的是稠密区域的方差比稀疏区域的方差大，文献[?]描述可以通过二分查找寻找最佳的σ_i。此外设置$p_{\boldsymbol{x}|\boldsymbol{x}}=0$，因为我们关注的是两两之间的相似度。

在此基础上，我们进一步定义对称的联合分布表示样本之间的相似性：

$$p_{i,j}=\frac{p_{j|i}+p_{i|j}}{2N}。 \tag{10-31}$$

在低维空间中，在给定$\boldsymbol{z}_i$和$\boldsymbol{z}_j$之间的相似度用自由度为 1 的T分布表示为：

$$\begin{aligned} q_{i,j}&=\frac{s'_{i,j}}{\sum_{j'\neq i}s'_{i,j'})},\\ s'_{i,j}&=\frac{1}{1+\left\|\boldsymbol{z}_i-\boldsymbol{z}_j\right\|_2^2}。 \end{aligned} \tag{10-32}$$

同样，设定$q_{i,i}=0$。

事实上，早期的 SNE 算法是用更简单的高斯分布表示降维后的条件概率，即$s'_{i,j}=\exp\left(-\|\boldsymbol{z}_i-\boldsymbol{z}_j\|_2^2\right)$。从下面的图 10-6 中我们可以看出为什么$T$分布比高斯分布更合适。

图 10-6 中横轴表示两个样本定之间距离，纵轴表示它们的相似度/概率。可以看到，对于较大相似度的点，T分布在低维空间中的距离需要稍小一点；而对于低相似度的点，T分布在低维空间中的距离需要更远。这恰好满足了我们的需求，即距离较近点的更紧密，距离较远的更加疏远。

如果降维的效果比较好，数据的局部结构保留完整，那么$p_{i,j}=q_{i,j}$，因此我们优化两个分布之间的距离-KL 散度（Kullback-Leibler Divergences），那么目标函数如下：

$$C=\sum_i KL(P_i||Q_i)=\sum_i\sum_j p_{i,j}\log\frac{p_{i,j}}{q_{i,j}}, \tag{10-33}$$

其中P_i表示了给定点$\boldsymbol{x}_i$下，其他所有数据点的条件概率分布。需要注意的是 KL 散度具有不对称性，即在低维映射中不同的距离对应的惩罚权重是不同的。距离较远的两个点来表达距离较近的两个点会产生更大的费用，而用较近的两个点来表达较远的两个点产生的费用相对较小。例如：用较小的$q_{i,j}=0.2$来建模较大的$p_{i,j}=0.8$，费用$C=p\log(p/q)=1.11$；用较大的$q_{i,j}=0.8$来建模较大的$p_{i,j}=0.2$，费用$C=-0.277$。因此，T-SNE 会倾向于保留数据中的局部特征。

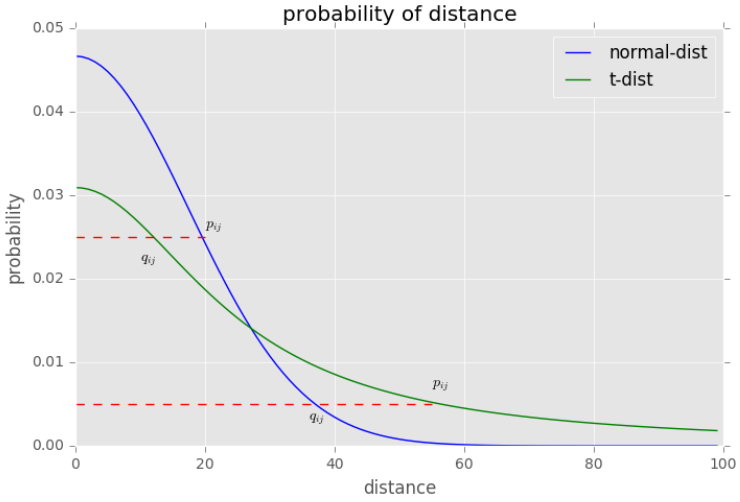


图 10-6　T-SNE 中的 T 分布与 SNE 中的高斯分布对比[4]。

对目标函数求极小值可通过梯度下降法来求解。目标函数的梯度为：

$$\frac{\partial C}{\partial \boldsymbol{z}_i}=4\sum_j(p_{i,j}-q_{i,j})(1+\|\boldsymbol{z}_i-\boldsymbol{z}_j\|^2)^{-1}(\boldsymbol{z}_i-\boldsymbol{z}_j)。\tag{10-34}$$

上述梯度计算需要对所有的j求和，计算量大，可用 Barnes-Hut [3]近似算法快速求解。

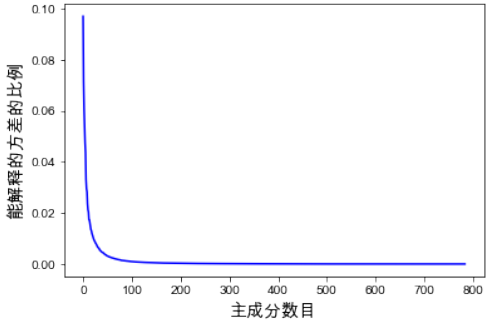
T-NSE 计算慢，工程上可以先对高维原始输入先用 PCA 等快速降维技术先降到一定维度（亦可保持全局结构），然后再用 T-NSE 进一步降维。

10.8 降维应用案例分析——MNIST 数据集

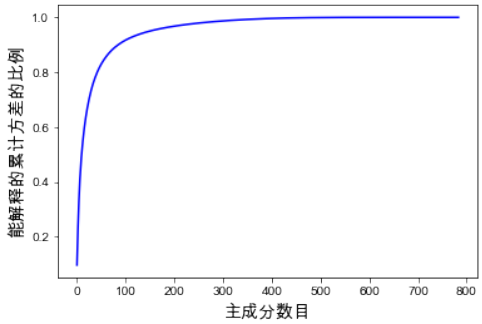
我们在 MNIST 数据集进行降维练习。MNIST 数据集简介请见 5.3.3。由于数据集本身带有标签，我们可以很方便通过可视化来简单比较各种降维技术。注意为了可视化，我们将数据降至 2 维。如果是为了更好重构原始数据或识别性能更好，降维后的维度D'需要仔细选取。由于非线性降维技术的计算复杂度高，我们从训练集中随机选择了 20%（共 8400 个样本）参与降维实验。

我们在 Scikit-Learn 框架下实现了 PCA、MDS、LLE、LE 和 T-NSE 降维。由于数字的图像中大部分为黑色色的背景，且偏白色的数字笔划大多连续且灰度值相似，所以 MNIST 数据集中，样本的各个维度之间的相关性很强，体现在 PCA 中，只有前面少数几个成分的方差很大，后续成分的方差迅速减小。图 10-7 给出了各个主成分能解释的方差比例λ_d，以及到该主成分为止，累计能解释的方差$\sum_{j=1}^{d}\lambda_j$。从信号重构的角度，可以选择能重构 85%方差的主成分数目为 59；如果要重构 95%的方差，需要个 151 个主成分。

各方法降维后的结果如图 10-8 所示，图中不同的颜色表示不同的数字。从图中我们可以发现，PCA、MDS、LLE 效果并不是太好，不同数字在二维空间中混在一起；LE 的效果稍好，T-NSE 的效果最好，但速度最慢。我们尝试先用 PCA 对原始数据将维（init='pca'），然后再用 T-NSE 降至 2 维，速度稍快，可视化效果和用 784 维原始数据做 T-NSE 相当。从图中可以看出，数字 4（紫色）和数字 9（灰色）是最容易混淆的。

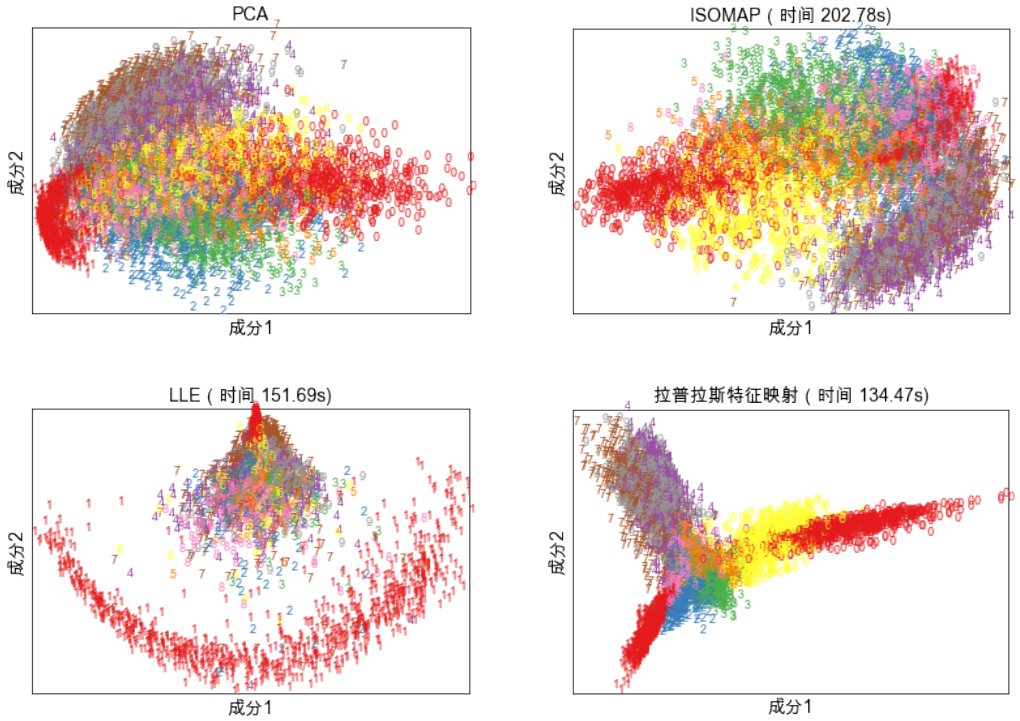


（a）主成分能解释的方差



（b）主成分能解释的累计方差

图 10-7　MNIST 数据集 PCA 结果



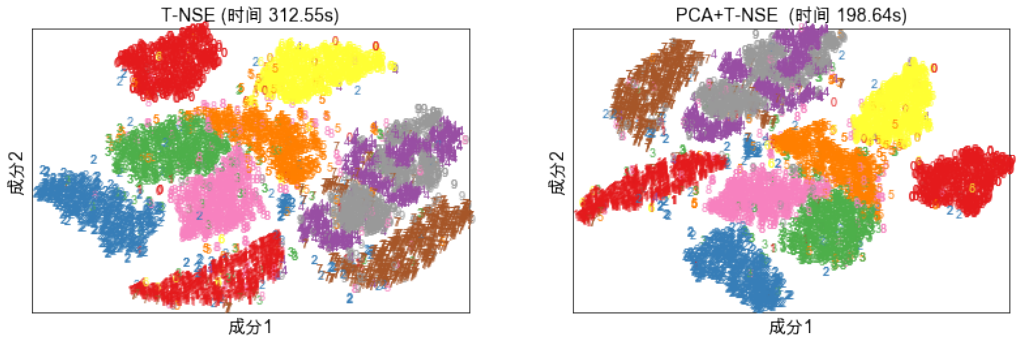


图 10-8　各种降维技术在 MNIST 数据集上的结果

我们在 PyTorch 框架下，分别采用基于全连接神经网络的自编码器和基于卷积网络的自编码器的实现降维。

全连接神经网络的自编码器的网络参数如下所示：

```
#编码器
self.encoder = nn.Sequential(
          nn.Linear(28*28, 128), #全连接层
          nn.ReLU(True),          #激活层
          nn.Linear(128, 64),    #全连接层
          nn.ReLU(True),         #激活层
          nn.Linear(64, 32),     #全连接层
       )
#解码器
self.decoder = nn.Sequential(
          nn.Linear(32, 64),
          nn.ReLU(True),
          nn.Linear(64, 128),
          nn.ReLU(True),
          nn.Linear(128, 28*28),
          nn.Sigmoid(),       # 结果在[0, 1]
       )
```

卷积神经网络的自编码器的网络参数如下所示：

```
#编码器
self.encoder = nn.Sequential(
          #卷积层：通道：1-->16，核大小 3x3： 28*28 -> 28*28
          nn.Conv2d(1, 16, 3,  padding=1) ,
          nn.ReLU(),  #激活层
          nn.MaxPool2d(2, 2),   #池化层，16*28*28-->16*14*14
          #卷积层：通道 16 --> 4，核大小 3x3：14*14 -> 14*14
          nn.Conv2d(16, 4, 3, padding=1),
```

```
            nn.ReLU(),
            nn.MaxPool2d(2, 2),   #池化层, 4*14*14-->4*7*7
        )
#解码器
self.decoder = nn.Sequential(
            nn.Linear(32, 64),
            nn.ReLU(True),

            nn.Linear(64, 128),
            nn.ReLU(True),

            nn.Linear(128, 28*28),
            nn.Sigmoid(),       # 结果在[0, 1]
          )
```

图 10-9 中给出训练 50 轮后基于全连接神经网络的自编码器和基于卷积神经网络的自编码器的结果，其中第一行是原始输入，第二行是自编码器的重构结果。读者可以尝试改变网络结构、迭代次数、优化方法和参数初始化方式等，体会网络超参数对降维和重构的影响。

(a)　　(b)

图 10-9　自编码器在 MINST 数据集上的结果。(a) 基于 DNN 的自编码器的重构结果；(b) 基于 CNN 的自编码器的重构结果。

10.9 小结

PCA 和自编码器通过重构误差最小将数据降到低维，可以保持全局信息。IsoMap 通过保持所有样本点对之间的者测地距离来计算降维后的表示，数据的全局结构也得以保持。而邻域嵌入技术，如 LLE、拉普拉斯映射、T-NSE 通过保持局部的几何特性来恢复全局的非线性结构，属于局部嵌入方法。在这些局部嵌入方法中讨论较远的点之间的关系没有意义。

10.10 练习

1. 基于 Scikit-Learn 自带的人脸数据集：
   lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4),
   运用本章所学的降维计算，并在降维后的前 2 维空间上显示样本（不同人脸用不同颜色表

示），观察降维后不同的人脸是否能区分开。

2. 对第 1 题中的数据集，对原始人脸图像 PCA 降维，然后用 RBF 核的 SVM 进行分类。请用 5 折交叉验证寻找最优超参数（主成分维数、惩罚参数C、已经 RBF 核函数参数γ）。