第一章　降维

机器学习中常常会遇到高维数据。高维数据往往会造成数据量过大、计算困难、维数灾难等问题，因此**降维 (Dimension Reduction)** 在机器学习中是一个重要的主题. 本章从统计学习的角度讨论一些常见的降维方法，包括PCA、LDA和流形学习，以及一系列变种.

1.1　主成分分析

主成分分析 (Principal Component Analysis, PCA) 是最常见的线性降维方法之一. PCA 降维后的数据尽可能保留了原数据的信息，因此 PCA 也广泛用于数据预处理.

1.1.1　最大方差视角

设样本矩阵 $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ ，其中 $\mathbf{x}_i \in \mathbb{R}^m$ 为 m 维列向量. 降维用到的投影矩阵设为 $\mathbf{W} \in \mathbb{R}^{m\times d}$。对于任意一个数据点 $\mathbf{x} \in \mathbb{R}^m$，投影后的数据为:

$$\mathbf{z} = \mathbf{W}^T\mathbf{x} \tag{1.1.1}$$

在 PCA 中，投影后的数据应当具有最大的方差（或者称散度, scatter），因为这样可以能使数据分散开，从而尽量保留其原有的数据分布趋势；此外，我们希望投影矩阵 $\mathbf{W}$ 是标准正交的，因为这可以使投影后的数据在每一维上都是互不相关的（注意，这里并不是相互独立，我们会在后面说明这一点），便于后续的处理. 因此我们可以得出 PCA 对应的优化问题：

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{z} - \bar{\mathbf{z}}\|^2 \tag{1.1.2}$$

$$= \arg\max_{\mathbf{W}} \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}}\|^2, \quad \text{s.t. } \mathbf{W}^T\mathbf{W} = \mathbf{I}_d \tag{1.1.3}$$

其中，$\mathbf{I}_d$ 是 d 阶单位矩阵，

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \quad \bar{\mathbf{z}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{z}_i.$$

化简 (1.1.3) 可以进一步得到:

$$\begin{aligned}
\hat{\mathbf{W}} &= \arg\max_{\mathbf{W}} \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}}\|^2 \\
&= \arg\max_{\mathbf{W}} \sum_{i=1}^{n} \operatorname{tr}\left[(\mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}})(\mathbf{W}^T\mathbf{x} - \mathbf{W}^T\bar{\mathbf{x}})^T\right] \\
&= \arg\max_{\mathbf{W}} \sum_{i=1}^{n} \operatorname{tr}\left[\mathbf{W}^T(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T\mathbf{W}\right]
\end{aligned}$$

$$
\begin{aligned}
&= \arg\max_{\mathbf{W}} \operatorname{tr}\left\{\mathbf{W}^T\left[\sum_{i=1}^{n}(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T\right]\mathbf{W}\right\} \\
&= \arg\max_{\mathbf{W}} \operatorname{tr}(\mathbf{W}^T\mathbf{S}_t\mathbf{W}), \quad \text{s.t. } \mathbf{W}^T\mathbf{W}=\mathbf{I}_d
\end{aligned} \tag{1.1.4}
$$

其中，$\operatorname{tr}(\cdot)$ 是矩阵的迹，$\mathbf{S}_t$ 是 $\mathbf{X}$ 散度矩阵 (scatter matrix)，其定义为

$$
\mathbf{S}_t = \sum_{i=1}^{n}(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T.
$$

其中，$\tilde{\mathbf{X}}$ 是去中心化后的样本矩阵，即$\tilde{\mathbf{X}} = [\mathbf{x}_1-\bar{\mathbf{x}},\cdots,\mathbf{x}_n-\bar{\mathbf{x}}]$. 求解 (1.1.4) 中的优化问题可以运用**拉格朗日乘数法**，设拉格朗日函数为：

$$
L(\mathbf{W},\mathbf{\Lambda}) = \operatorname{tr}(\mathbf{W}^T\mathbf{S}_t\mathbf{W}) - \operatorname{tr}[\mathbf{\Lambda}^T(\mathbf{W}^T\mathbf{W}-\mathbf{I}_d)]. \tag{1.1.5}
$$

令其对 $\mathbf{W}$ 的导数为0得：

$$
\frac{\partial L}{\partial \mathbf{W}} = \mathbf{S}_t\mathbf{W} - \mathbf{W}\mathbf{\Lambda} = 0 \Rightarrow \mathbf{S}_t\mathbf{W} = \mathbf{W}\mathbf{\Lambda}. \tag{1.1.6}
$$

显然，(1.1.6) 对应着下面的特征方程：

$$
\mathbf{S}_t\mathbf{w} = \lambda\mathbf{w}. \tag{1.1.7}
$$

因此，$\mathbf{W}$ 应由 $\mathbf{S}_t$ 中最大的 d 个特征值对应的特征向量组成。

1.1.2 最小重构误差视角

从几何的直观意义上来说，PCA 降维后的数据能够最好地还原回原数据，即重构误差最小。对于任意一个降维后的样本 $\mathbf{z}$，其重构数据表示为

$$
\hat{\mathbf{x}} = \mathbf{W}\mathbf{z} = \mathbf{W}\mathbf{W}^T\mathbf{x}.
$$

在这里，我们用均方误差来度量 PCA 的重构误差，就有：

$$
\begin{aligned}
\hat{\mathbf{W}} &= \arg\min_{\mathbf{W}} \frac{1}{n}\sum_{i=1}^{n}\|\tilde{\mathbf{x}}_i - \mathbf{W}\mathbf{W}^T\tilde{\mathbf{x}}_i\|^2 \\
&= \arg\min_{\mathbf{W}} \|\tilde{\mathbf{X}} - \mathbf{W}\mathbf{W}^T\tilde{\mathbf{X}}\|_F^2, \quad \text{s.t. } \mathbf{W}^T\mathbf{W}=\mathbf{I}_d,
\end{aligned} \tag{1.1.8}
$$

其中，$\|\cdot\|_F$ 表示 F 范数，$\tilde{\mathbf{X}}$ 是去中心化后的样本。进一步化简 (1.1.8) 就有：

$$
\begin{aligned}
\hat{\mathbf{W}} &= \arg\min_{\mathbf{W}} \|\tilde{\mathbf{X}} - \mathbf{W}\mathbf{W}^T\tilde{\mathbf{X}}\|_F^2 \\
&= \arg\min_{\mathbf{W}} \operatorname{tr}\left[(\tilde{\mathbf{X}} - \mathbf{W}\mathbf{W}^T\tilde{\mathbf{X}})(\tilde{\mathbf{X}} - \mathbf{W}\mathbf{W}^T\tilde{\mathbf{X}})^T\right] \\
&= \arg\min_{\mathbf{W}} \operatorname{tr}(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T) - 2\operatorname{tr}(\tilde{\mathbf{X}}\mathbf{W}\mathbf{W}^T\tilde{\mathbf{X}}) + \operatorname{tr}(\mathbf{W}\mathbf{W}^T\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\mathbf{W}\mathbf{W}^T) \\
&= \arg\min_{\mathbf{W}} -\operatorname{tr}(\mathbf{W}^T\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\mathbf{W}) \\
&= \arg\max_{\mathbf{W}} \operatorname{tr}(\mathbf{W}^T\mathbf{S}_t\mathbf{W}), \quad \text{s.t. } \mathbf{W}^T\mathbf{W}=\mathbf{I}_d.
\end{aligned} \tag{1.1.9}
$$

可见 (1.1.4) 和 (1.1.9) 是相同的优化问题，因此对于 PCA，最大化投影后数据的方差和最小化重构误差是等价的。

1.1.3 概率视角

我们从概率的视角来观察之前得到的结果. 根据 (1.1.1)，我们可以得到

$$\mathbb{E}(\mathbf{z}) = \mathbb{E}(\mathbf{W}^T\mathbf{x}) = \mathbf{W}^T\boldsymbol{\mu}_{\mathbf{x}}, \tag{1.1.10}$$

$$\begin{aligned}\mathrm{Cov}(\mathbf{z}) &= \mathbb{E}[(\mathbf{z}-\mathbb{E}(\mathbf{z}))(\mathbf{z}-\mathbb{E}(\mathbf{z}))^T] \\ &= \mathbf{W}^T\mathbb{E}[(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})^T]\mathbf{W} \\ &= \mathbf{W}^T\mathrm{Cov}(\mathbf{x})\mathbf{W} = \mathbf{W}^T\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{W},\end{aligned} \tag{1.1.11}$$

其中，$\boldsymbol{\mu}_{\mathbf{x}}$, $\boldsymbol{\Sigma}_{\mathbf{x}}$ 表示 $\mathbf{x}$ 的均值和协方差矩阵. 事实上，当样本量足够时，$\mathbf{S}_t/n$ 就是一个对 $\boldsymbol{\Sigma}_{\mathbf{x}}$ 的近似. 因此对 $\mathbf{S}_t$ 进行特征分解相当于对 $\boldsymbol{\Sigma}_{\mathbf{x}}$ 特征分解. 假设 $\boldsymbol{\Sigma}_{\mathbf{x}}$ 的特征分解为：

$$\boldsymbol{\Sigma}_{\mathbf{x}} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T, \tag{1.1.12}$$

其中 $\mathbf{V}$ 是特征向量组成的矩阵，且满足 $\mathbf{V}^T\mathbf{V} = \mathbf{I}_m$，对角矩阵 $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_m)$，且 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$. 根据前两节的分析知道，$\mathbf{W} = \mathbf{V}(:, 1:d)$，代入 (1.1.11) 可以得到：

$$\begin{aligned}\mathrm{Cov}(\mathbf{z}) &= \mathbf{W}^T\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T\mathbf{W} \\ &= \begin{bmatrix}\mathbf{I}_d & \boldsymbol{O}_{m\times(m-d)}\end{bmatrix}\begin{bmatrix}\mathrm{diag}(\lambda_1,\cdots,\lambda_d) & \boldsymbol{O}_{d\times(m-d)} \\ \boldsymbol{O}_{(m-d)\times d} & \mathrm{diag}(\lambda_{d+1},\cdots,\lambda_m)\end{bmatrix}\begin{bmatrix}\mathbf{I}_d \\ \boldsymbol{O}_{m\times(m-d)}\end{bmatrix} \\ &= \mathrm{diag}(\lambda_1, \cdots, \lambda_d).\end{aligned} \tag{1.1.13}$$

其中，$\boldsymbol{O}$ 代表元素全为0的矩阵. 从上式可以看到，投影后的数据 $\mathbf{z}$ 的协方差矩阵是一个对角矩阵，这说明投影后各个方向的分量互不相关，因此 $\mathbf{z}$ 的每一维之间没有线性关系. 这意味着 PCA 投影的过程中消除了各个变量之间的线性关系. 特别的，如果原数据 $\mathbf{x}$ 服从高斯分布 $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})$，则 $\mathbf{z}$ 也服从高斯分布：

$$p(\mathbf{z}|\mathbf{W}) = \mathcal{N}(\mathbf{W}^T\boldsymbol{\mu}_{\mathbf{x}}, \mathrm{diag}(\lambda_1, \cdots, \lambda_d)). \tag{1.1.14}$$

根据高斯分布独立性与相关性等价，可知 $\mathbf{z}$ 的每一维是**相互独立**的. 因此当原数据服从高斯分布时，PCA具有最佳的表现.

1.1.4 PCA 的优化

根据之前的分析，PCA 的计算需要对一个 $m \times m$ 维的协方差矩阵进行特征分解，其时间复杂度是$O(m^3)$. 当 m 很大时，PCA 的计算耗时也是巨大的. 文献 [1] 中提出了一种优化方法. 对于特征方程 (1.1.7)，两边同时乘以 $\tilde{\mathbf{X}}^T$ 可以得到：

$$\begin{aligned}&\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\hat{\mathbf{w}} = \lambda\tilde{\mathbf{X}}^T\hat{\mathbf{w}} \\ \Leftrightarrow&\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\mathbf{a} = \lambda\mathbf{a}.\end{aligned} \tag{1.1.15}$$

其中，$\mathbf{a} = \tilde{\mathbf{X}}^T\hat{\mathbf{w}}$. 可见，$\mathbf{a}$ 就是 $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ 的特征向量. 又由于

$$\tilde{\mathbf{X}}\mathbf{a} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\hat{\mathbf{w}} = \lambda\hat{\mathbf{w}},$$

考虑到正交限制，$\hat{\mathbf{w}}^T\hat{\mathbf{w}} = 1$，因此我们可以得到：

$$\hat{\mathbf{w}} = \frac{1}{\sqrt{\lambda}}\tilde{\mathbf{X}}\mathbf{a}. \tag{1.1.16}$$

因此，我们只需要求解 $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ 的特征向量 $\mathbf{a}$，然后再使用 (1.1.16) 计算$\hat{\mathbf{w}}$ 即可. 此时的计算时间复杂度变为 $O(n^3)$，因此当样本量 $n \ll m$ 时，该算法可以得到很大的性能提升.

另一种方式是借助奇异值分解（Singular Value Decomposition, SVD）替换前面的特征分解. 假设 $\mathbf{X}_t$ 的奇异值分解有以下形式：

$$\tilde{\mathbf{X}} = \mathbf{U}\mathbf{D}\mathbf{V}^T, \tag{1.1.17}$$

这里 $\mathbf{D} = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_h)$，其中 $h = \min\{m, n\}$，且 σ_i 是第 i 大的奇异值， $\mathbf{U}, \mathbf{V}$ 分别是左、右奇异矩阵，满足 $\mathbf{U}^T\mathbf{U} = \mathbf{I}_h, \mathbf{V}^T\mathbf{V} = \mathbf{I}_h$，由此我们可以得到：

$$\mathbf{S}_t = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T = \mathbf{V}\mathbf{D}^2\mathbf{V} \tag{1.1.18}$$

对比 $\mathbf{S}_t$ 的特征分解可以知道，这里的 $\mathbf{V}$ 就是 $\mathbf{S}_t$ 的特征向量矩阵，且 $\mathbf{\Lambda} = \mathbf{D}^2$，说明 $\mathbf{S}_t$ 的特征值是其对应奇异值的平方 $\lambda_i = \sigma_i^2$. 所以我们可以通过对 $\tilde{X}$ 进行奇异值分解求出 $\mathbf{V}$，再用其前 d 列组成投影矩阵 $\mathbf{W}$. 奇异值分解往往比对应的特征分解迅速，并且我们甚至不需要计算协方差矩阵，因此这种方法也大大降低了计算时间.

1.2 线性判别分析

分类是机器学习中的一个重要任务. 尽管 PCA 能够尽量还原原始数据的面貌，但由于其没有用到**标签 (label)** 信息，所以 PCA 是一种**无监督学习**. **Fisher 判别分析 (Fisher Discriminant Analysis, FDA)** 则与之不同，它考虑降维之后不同类别的数据应尽可能分开. 由于 FDA 的学习需要用到所有样本的标签信息，因此它是一种**监督学习方法**. 在线性降维的情况下， FDA 也叫**线性判别分析 (Linear Discriminant Analysis, LDA)**

1.2.1 Fisher 判别准则和线性判别分析

设样本矩阵 $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ ，其中 $\mathbf{x} \in \mathbb{R}^m$ 为 m 维列向量. 设标签集为 $\mathcal{Y}$，数据分为 c 个类，即 $|\mathcal{Y}| = c$，样本 $\mathbf{x}_i$ 对应的标签是 $y_i \in \mathcal{Y}$. 方便起见，不妨设 $\mathcal{Y} = \{1, 2, \cdots, c\}$.

我们在上一节接触到了散度这一个词，它描述了数据的分散程度，因此我们常用数据方差来描述它. LDA 的核心思想是希望投影之后的数据每个类之间尽量分散，而每个类内的样本尽量聚集. 也就是最大化**类间散度**以及最小化**类内散度**，为了量化这两种散度，我们引入类间散度矩阵 $\mathbf{S}_b$ 和类内散度矩阵 $\mathbf{S}_w$：

$$\mathbf{S}_b = \sum_{l=1}^{c}(\boldsymbol{\mu}_l - \boldsymbol{\mu})(\boldsymbol{\mu}_l - \boldsymbol{\mu})^T, \quad \mathbf{S}_w = \sum_{l=1}^{c}\sum_{y_j=l} n_l(\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T, \tag{1.2.1}$$

其中，n_l 为第 l 个类的样本数量，且 $n_1 + n_2 + \cdots + n_c = n$. $\boldsymbol{\mu}_i$ 和 $\boldsymbol{\mu}$ 分别指第 i 个类样本的均值和样本总均值，即

$$\boldsymbol{\mu}_l = \frac{1}{n_l}\sum_{y_i=l}\mathbf{x}_i, \quad \boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i. \tag{1.2.2}$$

类比前一节 PCA 中散度的概念，我们将类间散度和类内散度定义为 $\mathrm{tr}(\mathbf{S}_b)$ 和 $\mathrm{tr}(\mathbf{S}_w)$，和 PCA 中类似，我们可以推出投影后数据的类间散度和类内散度就是 $\mathrm{tr}(\mathbf{W}^T\mathbf{S}_b\mathbf{W})$ 和 $\mathrm{tr}(\mathbf{W}^T\mathbf{S}_w\mathbf{W})$.

在量化了类间散度和类内散度后，我们再来考虑 LDA 中最大化类间散度和最小化类内散度的目标. 我们可以直接用一个优化问题来描述：

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \frac{\mathrm{tr}(\mathbf{W}^T\mathbf{S}_b\mathbf{W})}{\mathrm{tr}(\mathbf{W}^T\mathbf{S}_w\mathbf{W})}. \tag{1.2.3}$$

这就是 **Fisher 判别准则 (Fisher Discriminant Criterion)**. 直观上来看，要最大化 (1.2.3) 中的目标函数，当然是希望分子尽量大、分母尽量小，这也就迫使投影后数据的类间散度尽量大、类内散度尽量小. 同时，这是一个关于广义瑞利商 (Genralized Rayleigh Quotient) 的优化问题，有许多种求解方法，其中一种求解方法是，当 $\mathbf{S}_w$ 可逆时，我们可以作以下变形：

$$\begin{aligned}\hat{\mathbf{W}} &= \arg\max_{\mathbf{W}} \frac{\mathrm{tr}(\mathbf{W}^T\mathbf{S}_b\mathbf{W})}{\mathrm{tr}(\mathbf{W}^T\mathbf{S}_w\mathbf{W})} \\ &= \arg\max_{\mathbf{W}} \mathrm{tr}(\mathbf{W}^T\mathbf{S}_b\mathbf{W}) \quad \text{s.t. } \mathbf{W}^T\mathbf{S}_w\mathbf{W} = \mathbf{I}_d.\end{aligned} \tag{1.2.4}$$

通过拉格朗日乘数法可以推出，$\mathbf{W}$ 由以下广义特征方程确定：

$$\mathbf{S}_b\hat{\mathbf{w}} = \lambda\mathbf{S}_w\hat{\mathbf{w}}. \tag{1.2.5}$$

因此，$\mathbf{W}$ 由矩阵 $\mathbf{S}_w^{-1}\mathbf{S}_b$ 最大的 d 个特征值对应的特征向量组成. 但在通常情况下，$\mathbf{S}_w$ 都不是满秩的，因此一种常见的做法是先用 PCA 进行预处理，去除掉小特征值的成分，从而使得 $\mathbf{S}_w$ 可逆. 这种做法称作 PCA+LDA [2].

1.2.2 k近邻分类器

对于一个分类问题，降维只是辅助手段，分类才是最终的目标. 在分类这一步上，我们需要确定一个**决策函数** $\mathcal{D}(\cdot)$ **(decision function)**，对于一个新的数据点 $\mathbf{x}$，我们就可以预测它的标签为：

$$\hat{y} = \mathcal{D}(\mathbf{x}). \tag{1.2.6}$$

在这里我们先介绍两种分类模型，包括 ***k*近邻分类器**和**高斯判别分析**.

在 PCA 和 LDA，我们通过训练计算出了投影矩阵 $\mathbf{W}$，这就是模型的参数 (model parameter). 而 k 近邻分类器 (k-Nearest Neighbour Classifier, KNN) 是一种非参数方法 (non-parametric method)，言外之意就是它不需要对参数学习. 同时 KNN 只需要训练数据，不需要训练过程，因此它也是一种惰性学习 (lazy learning). 在 KNN 中还有一个参数 k，与前面需要通过训练学习的模型参数不同，它是由人工设定的，这一类参数叫做超参数 (hyper-parameter).

KNN 的思想很朴素：离谁近就和谁一个类. 我们考虑最简单的情形，当 $k = 1$ 时，对于待分类样本 $\mathbf{x}$，我们在训练样本中找到距离它最近的样本点 $\mathbf{x}_i$，这里我们可以用多种距离度量方法，其中欧氏距离比较常见：

$$d(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{x} - \mathbf{x}_i\|_2. \tag{1.2.7}$$

然后，再以 $\mathbf{x}_i$ 的标签作为 $\mathbf{x}$ 的预测标签，这就是预测的过程. 当 $k = 1$ 时， KNN 也称作最近邻分类器 (Nearest Neighbour Classifier). 然而通常我们只找到最近的那个样本点并不一定能够得到最好的预测结果，因为所参考的近邻点只有 1 个，具有一定的局部性. 因此通常我们会考虑 $k > 1$，也就是找到 k 个距离 $\mathbf{x}$ 最近的训练样本，然后确定这 k 个训练样本的标签，用其众数作为 $\mathbf{x}$ 的标签预测.

为了方便描述，我们用 $\mathsf{N}_k(\mathbf{x})$ 来表示数据 $\mathbf{x}$ 在训练样本集中的 k 个近邻点的集合. 则可以将 KNN 的决策函数表示为：

$$\mathcal{D}(\mathbf{x}) = \mathrm{mode}[\mathbf{y}(\mathsf{N}_k(\mathbf{x}))], \tag{1.2.8}$$

其中 $\mathrm{mode}(\cdot)$ 为众数，以及

$$\mathbf{y}(\mathsf{N}_k(\mathbf{x})) = \{y_i \mid \mathbf{x}_i \in \mathsf{N}_k(\mathbf{x})\}. \tag{1.2.9}$$

KNN 需要依靠计算向量之间的距离进行分类，而当样本维数较高时，KNN 的计算开销也很大，因此降维成为了一个必然的选择. 实际中， PCA 降维后 KNN 的分类效果和不降维时的效果相差不大，而 LDA 降维后 KNN 的分类效果往往能有明显提升.

1.2.3 高斯判别分析

高斯判别分析 (Gaussian Discriminant Analysis, GDA) 是基于概率的判别模型. GDA 并不复杂，但这个模型的优化思想是十分重要的.

我们首先引入独热编码 (one-hot encoding) 的概念. 我们常常用一个整数 y_i 来表示标签值，我们可以把它转化为等价的独热编码向量 $\mathbf{t}_i$ 表示：

$$\mathbf{t}_i = [t_{i1}, t_{i2}, \cdots, t_{ic}]^T, \quad t_{ij} = \begin{cases} 1, & j = y_i, \\ 0, & \text{otherwise}, \end{cases} \tag{1.2.10}$$

由于可以带来一些表示上的便利，独热编码的标签表示在多分类问题中经常出现.

高斯判别分析中用到了**高斯分布 (Gaussian Distribution)**，它是机器学习中最常见的概率分布之一. 设一个 m 维高斯分布的均值和方差为 $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$，则我们将它记为$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$，其概率密度函数为：

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{m/2}\det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{1.2.11}$$

其中，$\det(\cdot)$ 为行列式算子.

GDA 中用到了另一个分布是**多项分布 (Multinomial Distribution)**，它是一个离散型分布. 对于离散型随机变量 y ($y = 1, 2, \cdots, c$)，多项分布有参数 $\boldsymbol{\alpha} \in \mathbb{R}^c$，则其概率分布满足：

$$p(y|\boldsymbol{\alpha}) = \prod_{i=1}^{c} \alpha_i^{t_i} = \alpha_y, \quad \sum_{i=1}^{n} \alpha_i = 1, \tag{1.2.12}$$

其中 $\mathbf{t}$ 是 y 对应的独热编码向量. 注意，上面这个概率值等于 α_i，当且仅当 $t_i = 1$，或 $y = i$，所以这个概率可能的取值就是 $\boldsymbol{\alpha}$ 中的每个元素.

接下来正式提出 GDA 的模型. 考虑后验分布 $p(y|\mathbf{x})$，显然这是一个离散分布（因为 y 是离散取值的），它表示已知给定样本 $\mathbf{x}$ 时，标签 y 的分布，也正是 $\mathbf{x}$ 的标签为 y 的概率，因此一种自然的想法就是，对于未知数据 $\mathbf{x}$， $\mathbf{x}$ 的标签应该预测为使得 $p(y|\mathbf{x})$ 最大的 y. 因此，我们可以写出决策函数

$$\mathcal{D}(\mathbf{x}) = \arg\max_y p(y|\mathbf{x}). \tag{1.2.13}$$

这个决策函数可以进一步简化，注意到根据贝叶斯定理有：

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \propto p(\mathbf{x}|y)p(y). \tag{1.2.14}$$

因此我们有：

$$\mathcal{D}(\mathbf{x}) = \arg\max_y p(y|\mathbf{x}) \tag{1.2.15}$$

$$= \arg\max_y p(\mathbf{x}|y)p(y), \tag{1.2.16}$$

其中， $p(\mathbf{x}|y)$ 表示了已知类别为 y 时 $\mathbf{x}$ 的分布，在 GDA 中假设为时高斯分布， $p(y)$ 则表示了类标签 y 的分布，在 GDA 中假设为多项分布. 这些分布的参数均需要通过学习的方式获得，在这里的学习原理是**极大似然估计 (Maximum Likelihood Estimate，MLE)**，即参数的选择应该使得样本出现的概率最大化，即：

$$\hat{\Theta} = \arg\max_\Theta p(\mathbf{X}|\mathbf{y}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)p(\mathbf{y}|\boldsymbol{\alpha}), \tag{1.2.17}$$

其中， $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ 是第 i 个类的高斯分布的均值和协方差矩阵， $\boldsymbol{\alpha}$ 是 y 对应多项分布的参数， Θ 为参数集，即 $\Theta = \{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, \boldsymbol{\alpha}\}$，这里的 $\mathbf{X}, \mathbf{y}$ 为训练样本和对应标签. 根据各个样本独立同分布，有：

$$\begin{aligned}
\hat{\Theta} &= \arg\max_\Theta p(\mathbf{X}|\mathbf{y}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)p(\mathbf{y}|\boldsymbol{\alpha}) \\
&= \arg\max_\Theta \prod_{j=1}^{n} p(\mathbf{x}_j|y_j, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)p(y_j|\boldsymbol{\alpha}) \\
&= \arg\max_\Theta \prod_{j=1}^{n} \left[\frac{1}{(2\pi)^{m/2}\det(\boldsymbol{\Sigma}_{y_j})} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_{y_j})^T \boldsymbol{\Sigma}_{y_j}^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_{y_j}) \right) \alpha_{y_j} \right]
\end{aligned} \tag{1.2.18}$$

对 (1.2.18) 式取对数 (这也是 MLE 中常见的处理手段)，可以得到

$$\begin{aligned}
\hat{\Theta} &= \arg\max_\Theta \sum_{j=1}^{n} \left[-\frac{1}{2}\ln\det(\boldsymbol{\Sigma}_{y_j}) - \frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_{y_j})^T \boldsymbol{\Sigma}_{y_j}^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_{y_j}) + \ln\alpha_{y_j} \right] + \text{const} \\
&= \arg\max_\Theta -\frac{1}{2}\sum_{j=1}^{n} \left[\ln\det(\boldsymbol{\Sigma}_{y_j}) + (\mathbf{x}_j - \boldsymbol{\mu}_{y_j})^T \boldsymbol{\Sigma}_{y_j}^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_{y_j}) \right] + \sum_{i=1}^{c} n_i \ln\alpha_i.
\end{aligned} \tag{1.2.19}$$

简单起见，我们先求解 $\boldsymbol{\alpha}$. 对于其第 i 个分量 α_i 的求解，上面的目标函数就可以化简为 $n_i \ln\alpha_i$，注意到约束(1.2.12)，构造拉格朗日函数：

$$L(\alpha_i, \lambda) = n_i \ln\alpha_i + \lambda\left(1 - \sum_{j=1}^{n}\alpha_j\right). \tag{1.2.20}$$

对 α_i 求导为0，可以得到：

$$\frac{\partial L}{\partial \alpha_i} = 0 \Rightarrow \frac{n_i}{\alpha_i} - \lambda = 0 \Rightarrow \alpha_i = \frac{n_i}{\lambda}. \tag{1.2.21}$$

显然，根据 $\alpha_1 + \alpha_2 + \cdots + \alpha_c = 1$，可知 $\lambda = n$. 因此有 $\hat{\alpha}_i = n_i/n$，这也就求出了 $\hat{\boldsymbol{\alpha}}$.

下一步我们计算 $\boldsymbol{\mu}_i$，针对第 i 个类的样本，我们可以将 (1.2.19) 化简为

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_i &= \arg\max_{\boldsymbol{\mu}_i} -\frac{1}{2}\sum_{j=1}^{n} \left[\ln\det(\boldsymbol{\Sigma}_{y_j}) + (\mathbf{x}_j - \boldsymbol{\mu}_{y_j})^T \boldsymbol{\Sigma}_{y_j}^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_{y_j}) \right] + \sum_{i=1}^{c} n_i \ln\alpha_i \\
&= \arg\min_{\boldsymbol{\mu}_i} \sum_{k=1}^{n_i} \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_i) + \text{const} \\
&= \arg\min_{\boldsymbol{\mu}_i} \sum_{k=1}^{n_i} \mathbf{x}_k^T \boldsymbol{\Sigma}_i^{-1}\mathbf{x}_k - 2\boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1}\mathbf{x}_k + \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \text{const} \\
&= \arg\min_{\boldsymbol{\mu}_i} \sum_{k=1}^{n_i} -2\boldsymbol{\mu}_i^T \boldsymbol{\Sigma}_i^{-1}\mathbf{x}_k + \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i,
\end{aligned} \tag{1.2.22}$$

令 (1.2.22) 中的目标函数对 $\boldsymbol{\mu}_i$ 的导数为 0 有：

$$\sum_{k=1}^{n_i} 2\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_i) = 0 \Rightarrow \hat{\boldsymbol{\mu}}_i = \frac{1}{n_i}\sum_{k=1}^{n_i}\mathbf{x}_k, \tag{1.2.23}$$

其中需要注意，$\boldsymbol{\Sigma}_i$ 为正定矩阵，因此这里可以直接消去.

最后，我们计算第 i 个高斯分布的协方差矩阵 $\boldsymbol{\Sigma}_i$. 化简 (1.2.19) 有：

$$\begin{aligned}\hat{\boldsymbol{\Sigma}}_i &= \arg\max_{\boldsymbol{\Sigma}_i} -\sum_{j=1}^{n}\left[\ln\det(\boldsymbol{\Sigma}_{y_j}) + (\mathbf{x}_j - \boldsymbol{\mu}_{y_j})^T\boldsymbol{\Sigma}_{y_j}^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_{y_j})\right] + \sum_{i=1}^{c} n_i \ln\alpha_i \\ &= \arg\min_{\boldsymbol{\Sigma}_i}\sum_{k=1}^{n_i}\ln\det(\boldsymbol{\Sigma}_i) + (\mathbf{x}_k - \boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_i) + \text{const} \\ &= \arg\min_{\boldsymbol{\Sigma}_i}\sum_{k=1}^{n_i}\ln\det(\boldsymbol{\Sigma}_i) + \operatorname{tr}\left[(\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}\right].\end{aligned} \tag{1.2.24}$$

对 $\boldsymbol{\Sigma}_i$ 求导为 0 得到：

$$\begin{aligned}&\sum_{k=1}^{n_i}\left[\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_i^{-2}(\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^T\right] = 0 \\ \Rightarrow\ &\hat{\boldsymbol{\Sigma}}_i = \frac{1}{n_i}\sum_{i=1}^{n_i}(\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^T.\end{aligned} \tag{1.2.25}$$

事实上， $\boldsymbol{\mu}_i$ 和 $\boldsymbol{\Sigma}_i$ 的结果都是理所当然的，因为它们在统计量上恰好描述了第 i 个类数据的平均值和协方差. 至此，我们求出了 GDA 中的模型参数，用决策函数(1.2.16)就可以对新的数据进行标签预测了. 由于各个参数均是有闭式解 (closed-solution) 的，因此 GDA 模型在计算上并不复杂.

1.3 流形学习

数据在高维空间中的分布往往具有一个特定的低维结构，例如数据是3维的，但它其实分布在一条2维抛物线上，那么这个抛物线就可以说是一个**流形 (manifold)**，而且这个低维的流形嵌入到了高维空间中. 那么，如何找到低维流形的真实形状呢? 在流形学习 (Manifold Learning) 中，我们可以通过降维的方式找到这些结构，也就是找出高维空间的低维流形嵌入，挖掘出数据的真实分布，这也是降维的一个重要作用.

1.3.1 拉普拉斯特征映射

拉普拉斯特征映射 (Laplacian Eigenmap (LE) 是非线性降维的方法[3]. 设样本矩阵 $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ ，其中 $\mathbf{x}_i \in \mathbb{R}^m$ 为 m 维列向量. 向量 $\mathbf{z} \in \mathbb{R}^d$ 是降维之后的数据，我们想找到一个映射 $\phi : \mathbb{R}^m \mapsto \mathbb{R}^d$，使得 $\mathbf{z} = \phi(\mathbf{x})$. 直观上，我们希望这个映射尽可能地保留数据在高维空间中的近邻关系，这是很直观的，因为如果我们想把流形结构保留下来，自然希望原先在高维空间中相近的点在低维空间中也是临近的，远离的点在低维空间中仍然远离. 设降维后的数据为 $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n]$，因此 LE 的优化目标为

$$\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}}\sum_{i,j=1}^{n}\|\mathbf{z}_i - \mathbf{z}_j\|^2 w_{ij}. \tag{1.3.1}$$

其中，样本通过一个无向图来量化 “临近” 关系，这里 $\mathbf{W}$ 是样本的邻接矩阵. 定义 $\mathbf{W}$ 的方式有很多，例如：

$$w_{ij} = \begin{cases} 1, & \|\mathbf{x}_i - \mathbf{x}_j\| \le \epsilon, \\ 0, & \text{otherwise.} \end{cases},$$

$$w_{ij} = \begin{cases} \exp\left(\|\mathbf{x}_i - \mathbf{x}_j\|^2/t\right), & \mathbf{x}_j \in \mathsf{N}_k(\mathbf{x}_i) \vee \mathbf{x}_i \in \mathsf{N}_k(\mathbf{x}_j) \\ 0, & \text{otherwise.} \end{cases}, \quad t > 0.$$

接下来，我们对 (1.3.1) 进行化简：

$$\begin{aligned} \hat{\mathbf{Z}} &= \arg\min_{\mathbf{Z}} \sum_{i,j=1}^{n} (\mathbf{z}_i - \mathbf{z}_j)^T (\mathbf{z}_i - \mathbf{z}_j) w_{ij} \\ &= \arg\min_{\mathbf{Z}} \sum_{i,j=1}^{n} (\mathbf{z}_i^T \mathbf{z}_i - 2\mathbf{z}_i^T \mathbf{z}_j + \mathbf{z}_j^T \mathbf{z}_j) w_{ij} \\ &= \arg\min_{\mathbf{Z}} 2 \sum_{i,j=1}^{n} \mathbf{z}_i^T w_{ij} \mathbf{z}_i - 2 \sum_{i,j=1}^{n} \mathbf{z}_i^T w_{ij} \mathbf{z}_j \\ &= \arg\min_{\mathbf{Z}} \sum_{i=1}^{n} \mathbf{z}_i^T d_i \mathbf{z}_i - \sum_{i,j=1}^{n} \mathbf{z}_i^T w_{ij} \mathbf{z}_j \\ &= \arg\min_{\mathbf{Z}} \operatorname{tr}(\mathbf{Z}\mathbf{D}\mathbf{Z}^T) - \operatorname{tr}(\mathbf{Z}\mathbf{W}\mathbf{Z}^T) \\ &= \arg\min_{\mathbf{Z}} \operatorname{tr}(\mathbf{Z}\mathbf{L}\mathbf{Z}^T). \end{aligned}$$

其中，$\mathbf{D}$ 为度矩阵，是一个对角矩阵：

$$\mathbf{D} = \operatorname{diag}\left(\sum_{j=1}^{n} w_{ij} \right)_{i=1}^{n}, \tag{1.3.2}$$

$\mathbf{L}$ 为拉普拉斯矩阵，定义为 $\mathbf{L} = \mathbf{D} - \mathbf{W}$. 为了使降维后每个维度互不相关，并且消除放缩因子的影响我们需要向原问题增加一个约束 $\mathbf{Z}\mathbf{D}\mathbf{Z}^T = \mathbf{I}_d$，则原问题变为

$$\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} \operatorname{tr}(\mathbf{Z}\mathbf{L}\mathbf{Z}^T), \quad \text{s.t. } \mathbf{Z}\mathbf{D}\mathbf{Z}^T = \mathbf{I}_d. \tag{1.3.3}$$

则 $\mathbf{Z}$ 由下面广义特征方程给出，

$$\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{D}\mathbf{u}_i. \tag{1.3.4}$$

并且 $\mathbf{Z} = [\mathbf{u}_2^T, \mathbf{u}_3^T, \cdots, \mathbf{u}_{d+1}^T]^T$，且 $0 = \lambda_1 \le \lambda_2 \le \lambda_3 \le \cdots \le \lambda_{d+1}$. 这里需要注意，$\mathbf{D}^{-1}\mathbf{L}$ 的最小特征值一定是 0，且特征向量一定是 $\mathbf{1}_n$，此处 $\mathbf{1}_n$ 指的是 n 维全 1 列向量，这是一个平凡解 (trivial solution)，因此我们将其丢弃.

需要注意的是，LE 中无法直接知道映射 ϕ 是怎样的，换言之，从 $\mathbf{x}$ 到 $\mathbf{z}$ 的映射是隐式的，这也为我们带来了许多不便. 例如对于新的数据点，我们无法用 ϕ 直接将它映射到低维空间.

局部保留投影 (Locality Preserving Projection, LPP) 是 LE 的一个线性近似版本 [4]. 它假设 ϕ 是一个线性投影：

$$\mathbf{z} = \phi(\mathbf{x}) = \mathbf{P}^T \mathbf{x}. \tag{1.3.5}$$

其中，$\mathbf{P}$ 是投影矩阵. 我们可以将(1.3.1) 转化为

$$\begin{aligned}\hat{\mathbf{P}} &= \arg\min_{\mathbf{P}} \sum_{i,j=1}^{n} \|\mathbf{z}_i - \mathbf{z}_j\|^2 w_{ij} \\ &= \arg\min_{\mathbf{P}} \sum_{i,j=1}^{n} \|\mathbf{P}^T\mathbf{x}_i - \mathbf{P}^T\mathbf{x}_j\|^2 w_{ij} \\ &= \arg\min_{\mathbf{P}} \operatorname{tr}(\mathbf{P}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{P}), \quad \text{s.t. } \mathbf{P}^T\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{P} = \mathbf{I}. \end{aligned} \tag{1.3.6}$$

因此，$\mathbf{P}$ 由以下特征方程给出

$$\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{p} = \lambda\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{p}. \tag{1.3.7}$$

也就是说，投影矩阵 $\mathbf{P}$ 由 $(\mathbf{X}\mathbf{D}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{L}\mathbf{X}^T$ 最小的 d 个特征值对应的特征向量组成. LPP 是线性映射，相比 LE 更容易进行计算，在模式识别中也有许多应用[4].

1.3.2 局部线性嵌入

为了挖掘数据的局部流形结构，**局部线性嵌入 (Locally Linear Embedding, LLE)** [5] 做出了一个假设：数据点 $\mathbf{x}_i$ 能用其 k 个近邻点近似地线性表示：

$$\mathbf{x}_i \approx \sum_{j=1}^{k} w_{ij}\mathbf{x}_j^{(i)}, \text{ with } \mathbf{x}_j^{(i)} \in \mathsf{N}_k(\mathbf{x}_i), \sum_{j=1}^{k} w_{ij} = 1. \tag{1.3.8}$$

为了保持这样的局部结构，降维后的数据 $\mathbf{z}_i$ 也应该保持这样的关系. 因此 LLE 分为两大步，第一步是求解线性表示系数矩阵 $\mathbf{W}$（或者叫权重矩阵），第二步才是根据求得的 $\mathbf{W}$ 降维.

第一步：求解表示系数

根据 (1.3.8)，我们期望近似表示时的误差尽量小，因此对于每一个样本 $\mathbf{x}_i$，有

$$\hat{\mathbf{w}}_i = \arg\min_{\mathbf{w}_i} \left\| \mathbf{x}_i - \sum_{j=1}^{k} w_{ij}\mathbf{x}_j^{(i)} \right\|^2 \tag{1.3.9}$$

$$= \arg\min_{\mathbf{w}_i} \left\| \mathbf{x}_i - \mathbf{X}^{(i)}\mathbf{w}_i \right\|^2, \quad \text{s.t. } \mathbf{1}_k^T\mathbf{w}_i = 1, \tag{1.3.10}$$

其中，$\mathbf{X}^{(i)} = [\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_k^{(i)}]$，$\mathbf{1}_k$ 为元素全是1的 k 维列向量. 进一步化简(1.3.10)式，有：

$$\begin{aligned}\hat{\mathbf{w}}_i &= \arg\min_{\mathbf{w}_i} \left\| \mathbf{x}_i - \mathbf{X}^{(i)}\mathbf{w}_i \right\|^2 \\ &= \arg\min_{\mathbf{w}_i} \left\| \mathbf{x}_i\mathbf{1}_k^T\mathbf{w}_i - \mathbf{X}^{(i)}\mathbf{w}_i \right\|^2 \\ &= \arg\min_{\mathbf{w}_i} \left\| \left(\mathbf{x}_i\mathbf{1}_k^T - \mathbf{X}^{(i)}\right)\mathbf{w}_i \right\|^2 \\ &= \arg\min_{\mathbf{w}_i} \mathbf{w}_i^T \left(\mathbf{x}_i\mathbf{1}_k^T - \mathbf{X}^{(i)}\right)^T \left(\mathbf{x}_i\mathbf{1}_k^T - \mathbf{X}^{(i)}\right)\mathbf{w}_i \\ &= \arg\min_{\mathbf{w}_i} \mathbf{w}_i^T\mathbf{S}_i\mathbf{w}_i, \quad \text{s.t. } \mathbf{1}_k^T\mathbf{w}_i = 1. \end{aligned} \tag{1.3.11}$$

其中，

$$\mathbf{S}_i = \left(\mathbf{x}_i\mathbf{1}_k^T - \mathbf{X}^{(i)}\right)^T \left(\mathbf{x}_i\mathbf{1}_k^T - \mathbf{X}^{(i)}\right). \tag{1.3.12}$$

求解上面的优化问题可以使用拉格朗日乘数法，令拉格朗日函数为

$$L(\mathbf{w}_i, \lambda) = \mathbf{w}_i^T \mathbf{S}_i \mathbf{w}_i - \lambda(\mathbf{1}_k^T \mathbf{w}_i - 1). \tag{1.3.13}$$

令其对 $\mathbf{w}_i$ 的导数为 0 有：

$$\frac{\partial L}{\partial \mathbf{w}_i} = 2\mathbf{S}_i \mathbf{w}_i - \lambda \mathbf{1}_k = 0 \Rightarrow \mathbf{w}_i = \frac{1}{2}\lambda \mathbf{S}_i^{-1} \mathbf{1}_k. \tag{1.3.14}$$

注意到约束条件 $\mathbf{1}_k^T \mathbf{w}_i = 1$，因此有

$$\mathbf{1}_k^T \mathbf{w}_i = \frac{1}{2}\lambda \mathbf{1}_k^T \mathbf{S}_i^{-1} \mathbf{1}_k = 1 \Rightarrow \lambda = \frac{2}{\mathbf{1}_k^T \mathbf{S}_i \mathbf{1}_k} \tag{1.3.15}$$

从而

$$\mathbf{w}_i = \frac{\mathbf{S}_i^{-1} \mathbf{1}_k}{\mathbf{1}_k^T \mathbf{S}_i^{-1} \mathbf{1}_k} \tag{1.3.16}$$

对于每一个样本都按这种方式求解. 注意，这里求得的 $\mathbf{w}_i$ 只是一个 k 维列向量，其中的元素只考虑了 $\mathbf{x}_i$ 的 k 个近邻点的系数，事实上，对于其他的样本，其系数为 0. 为了方便表示 $\mathbf{x}_i$ 和所有样本的关系，我们将 $\mathbf{w}_i \in \mathbb{R}^k$ 扩充为 $\tilde{\mathbf{w}}_i \in \mathbb{R}^n$，其中

$$\tilde{w}_{ij} = \begin{cases} w_{ij}, & \mathbf{x}_j \in \mathsf{N}_k(\mathbf{x}_i), \\ 0, & \text{otherwise.} \end{cases} \tag{1.3.17}$$

因此扩充后对应的权重矩阵为 $\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2, \cdots, \tilde{\mathbf{w}}_n]$.

第二步：降维

根据之前的分析，我们希望降维后的数据集 $\mathbf{Z} \in \mathbb{R}^{d \times n}$ 依然保持局部线性关系，所以我们依然想要最小化低维空间种线性表示的误差，也就是说：

$$\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} \sum_{i=1}^{n} \left\| \mathbf{z}_i - \sum_{j=1}^{n} \tilde{w}_{ij} \mathbf{z}_j \right\|^2, \quad \text{s.t. } \mathbf{Z}\mathbf{1}_n = 0, \mathbf{Z}\mathbf{Z}^T = n\mathbf{I}_d, \tag{1.3.18}$$

其中，约束 $\mathbf{Z}\mathbf{1}_n = 0, \mathbf{Z}\mathbf{Z}^T = n\mathbf{I}_d$ 希望降维后的数据具有0均值以及单位协方差，也就是说，每一个维度互不相关.

$$\begin{aligned} \hat{\mathbf{Z}} &= \arg\min_{\mathbf{Z}} \sum_{i=1}^{n} \left\| \mathbf{z}_i - \sum_{j=1}^{n} \tilde{w}_{ij} \mathbf{z}_j \right\|^2 \\ &= \arg\min_{\mathbf{Z}} \left\| \mathbf{Z} - \mathbf{Z}\tilde{\mathbf{W}} \right\|_F^2 \\ &= \arg\min_{\mathbf{Z}} \left\| \mathbf{Z}(\mathbf{I}_n - \tilde{\mathbf{W}}) \right\|_F^2 \\ &= \arg\min_{\mathbf{Z}} \operatorname{tr}\left[\mathbf{Z}(\mathbf{I}_n - \tilde{\mathbf{W}})(\mathbf{I}_n - \tilde{\mathbf{W}})^T \mathbf{Z}^T \right] \\ &= \arg\min_{\mathbf{Z}} \operatorname{tr}(\mathbf{Z}\mathbf{M}\mathbf{Z}^T), \quad \text{s.t. } \mathbf{Z}\mathbf{1}_n = 0, \mathbf{Z}\mathbf{Z}^T = n\mathbf{I}_d, \end{aligned} \tag{1.3.19}$$

其中，$\mathbf{M} = (\mathbf{I}_n - \tilde{\mathbf{W}})(\mathbf{I}_n - \tilde{\mathbf{W}})^T$. 该优化问题的解由 $\mathbf{M}$ 的特征分解给出：

$$\mathbf{M}\mathbf{u} = \lambda \mathbf{u}. \tag{1.3.20}$$

注意到 $\mathbf{1}_n^T\tilde{\mathbf{W}}=\mathbf{1}_n^T$，或者说 $\tilde{\mathbf{W}}^T\mathbf{1}_n=\mathbf{1}_n$，因此

$$\begin{aligned}\mathbf{M}\mathbf{1}_n&=(\mathbf{I}-\tilde{\mathbf{W}})(\mathbf{I}-\tilde{\mathbf{W}})^T\mathbf{1}_n\\&=(\mathbf{I}-\tilde{\mathbf{W}}^T-\tilde{\mathbf{W}}-\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T)\mathbf{1}_n\\&=\mathbf{1}_n-\tilde{\mathbf{W}}^T\mathbf{1}_n-\tilde{\mathbf{W}}\mathbf{1}_n-\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T\mathbf{1}_n\\&=\mathbf{1}_n-\mathbf{1}_n-\tilde{\mathbf{W}}\mathbf{1}_n-\tilde{\mathbf{W}}\mathbf{1}_n\\&=\mathbf{0}=0\mathbf{1}.\end{aligned}\tag{1.3.21}$$

因此 $\mathbf{M}$ 具有特征值 0 以及其特征向量 $\mathbf{u}_1=\mathbf{1}_n$，与 LE 中相似，我们需要舍弃这一个特征向量，所以 $\mathbf{Z}$ 应该表示为：

$$\mathbf{Z}=[\mathbf{u}_2^T,\mathbf{u}_3^T,\cdots,\mathbf{u}_{d+1}^T]^T.\tag{1.3.22}$$

这里的巧妙之处在于，根据每个特征向量之间的单位正交性，可知

$$\mathbf{u}_1^T\mathbf{u}_i=0,\quad \forall i=2,3,\cdots,n.\tag{1.3.23}$$

所以此时我们求得的 $\mathbf{Z}$ 恰好满足约束 $\mathbf{Z}\mathbf{1}_n=0$. 因此在这里，我们将零特征值对应的特征向量舍弃可以理解为令降维后数据具有零均值.

与 LE 类似， LLE 也有一个线性版本 NPP 或 NPE (Neighborhood Preserving Projections/Embedding) [6, 7]，其假设了从 $\mathbf{x}$ 到 $\mathbf{z}$ 的映射是线性的，即 $\mathbf{z}=\mathbf{P}^T\mathbf{x}$， NPP 中确定权重 $\mathbf{W}$ 的方法与 LLE 中是一样的，只有在降维部分有所不同. 因此我们有

$$\begin{aligned}\hat{\mathbf{P}}&=\arg\min_{\mathbf{P}}\sum_{i=1}^n\left\|\mathbf{z}_i-\sum_{j=1}^n\tilde{w}_{ij}\mathbf{z}_j\right\|^2\\&=\arg\min_{\mathbf{P}}\operatorname{tr}(\mathbf{Z}\mathbf{M}\mathbf{Z}^T)&(1.3.24)\\&=\arg\min_{\mathbf{P}}\operatorname{tr}(\mathbf{P}^T\mathbf{X}\mathbf{M}\mathbf{X}^T\mathbf{P}),\quad \text{s.t. }\mathbf{Z}\mathbf{Z}^T=\mathbf{P}^T\mathbf{X}\mathbf{X}^T\mathbf{P}=\mathbf{I}_d&(1.3.25)\end{aligned}$$

此时称 $\mathbf{P}$ 是**广义正交 (generalized orthogonality)**的. 此时，$\mathbf{P}$ 则是通过解下面的广义特征方程得到.

$$\mathbf{X}\mathbf{M}\mathbf{X}^T\mathbf{p}=\lambda\mathbf{X}\mathbf{X}^T\mathbf{p}.\tag{1.3.26}$$

ONPP (Orthogonal neighborhood preserving projections) [6] 是使用正交约束的 NPP，这里的投影矩阵 $\mathbf{P}$ 是单位正交的. 与 (1.3.19) 中的推导类似，我们可以得到

$$\hat{\mathbf{P}}=\arg\min_{\mathbf{P}}\operatorname{tr}(\mathbf{P}^T\mathbf{X}\mathbf{M}\mathbf{X}^T\mathbf{P}),\quad \text{s.t. }\mathbf{P}^T\mathbf{P}=\mathbf{I}_d\tag{1.3.27}$$

其中，约束 $\mathbf{P}^T\mathbf{P}=\mathbf{I}_d$ 确保了特征之间互不相关，这个在 PCA 一节中曾说明过. 此时，$\mathbf{P}$ 由下面的特征方程给出.

$$\mathbf{X}\mathbf{M}\mathbf{X}^T\mathbf{p}=\lambda\mathbf{p}\tag{1.3.28}$$

注意到，在上面这两种方法中 $\mathbf{X}\mathbf{M}\mathbf{X}^T$ 是近似低秩的，这是因为第一步求得的 $\tilde{\mathbf{W}}$ 使得 $\mathbf{X}$ 近似在 $(\mathbf{I}-\tilde{\mathbf{W}})$ 的左零空间中，所以实际求解时仍然要舍去最小特征值对应的特征向量，防止降维后的数据可能出现某一维度为 0 或趋近于 0.

参考文献

[1] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[2] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[3] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[4] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face recognition using laplacianfaces," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 3, pp. 328–340, 2005.

[5] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[6] E. Kokiopoulou and Y. Saad, "Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2143–2156, 2007.

[7] X. He, D. Cai, S. Yan, and H.-J. Zhang, "Neighborhood preserving embedding," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1208–1213, IEEE, 2005.