第十一讲降维与度量学习

韩丽颖，周瀚旭，甘璧丞
李澳，钱麒澎，黄大庆

2020 年 6 月 24 日

Summary

在本章中，我们将进一步介绍分类问题的常见学习方法，探讨如何对数据降维，以及对度量的学习。第一节中我们会学习 k 近邻学习，是一种十分简单易理解的分类法；第二节中我们着重介绍多维伸缩学习法，可以通过对数据有效地降维，避免维数灾难的情况；第三节中我们介绍了处理具有线性关系数据降维的主成分分析法；第四节基于主成分分析法在处理非线性问题的局限，介绍了核主成分分析法；第五节介绍了流形学习及其一种方法 Isomap 算法；第六节从实例出发介绍了度量学习的基本思想；第七节选择了课上几个较有代表性的问题进行了回答。

目录

Chapter 1

降维与度量学习

1.1 k 近邻学习 (kNN)

k 近邻，即 k -Nearest Neighbor(kNN) 学习，是机器学习中一种常用的监督学习方法。其工作机制如下：

1. 给定数据集 $S = \{(x_i, y_i)\}_{i=1}^n$，其中 x_i 为数据值，y_i 为标签，共有 n 组 (x_i, y_i) 样本；
2. 确定一种度量 d；
3. 选取一个测试样本 x_t 并对 $i \in \{1, 2, \cdots, n\}$ 计算度量 $d(x_i, x_t)$；
4. 选出距离 x_t 最近的 k 个点，即 k 个“邻居”；
5. 若在分类任务中：使用投票的方式，讲这 k 个样本中出现最多的类别标记作为预测结果；若在回归任务中：使用取平均的方法，将这 k 个样本的实值输出标记的平均值作为预测结果。（同时还可基于距离远近进行加权平均或加权投票，距离越近的样本权重越大。）

1.1.1 k 近邻学习的特点

1. 无需训练 与很多其他的机器学习方法相比, k 近邻学习有一个明显的不同之处，在于它不需要提前训练，我们可以等到收到测试集后再开始应用 k 近邻学习，因此该学习方法也被归类为“懒惰学习”(lazy learning)。

2. 超参数 k (Hyper-parameter k) 由于 k 并不是通过前期训练获得的参数，我们将 k 称为超参数。图 1.1为 k 近邻分类器的一个示意图。由此可见 k 是一个重要参数, 并且当 k 取不同

值时，分类的结果会有显著不同。在对 k 值选择时，我们通常采取交叉验证 (Cross Validation) 的方法，即从训练集中选取一部分作为验证集，接着采取不同的 k 值，比较 k 近邻分类器在验证集上的的错误率。由此我们可以获得验证集错误率和 k 值的关系，记作 $error(k)$，最后选择 $error(k)$ 取最小值时的 k_{min} 作为超参数 k 的值。

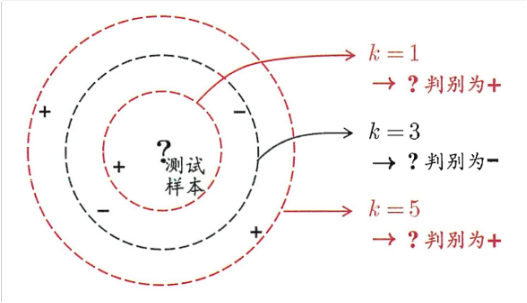


图 1.1: k 近邻分类器在 k 取不同值时的示意图，测试样本在 $k = 1$, $k = 5$ 时被判别为正例, $k = 3$ 时被判别为反例 (参考周志华《机器学习》).

3. 高时间复杂度 所谓有得必有失，k 近邻学习机制简单的代价便是有很大的计算成本。若我们有 n 个样本，每计算一次两点间的度量需要花费 $O(s)$, 那么总的时间复杂度为 $O(n \cdot s)$。

1.1.2 最近邻分类器结果分析

假设我们选择了恰当的度量 d，能够恰当地找出 k 个近邻, 我们接下来对简单情况“最近邻分类器”(1NN, 即 $k = 1$) 在二分类问题上的性能做一个简单讨论。

假设样本是独立同分布 (i.i.d.)，给定测试样本 $\boldsymbol{x}$，若其最近邻样本为 $\boldsymbol{z}$，则该分类器出错的概率为 $\boldsymbol{x}$ 与 $\boldsymbol{z}$ 类别标记不同的概率，即

$$P(err) = 1 - \sum_{c \in \mathcal{Y}} P(c|\boldsymbol{x}) P(c|\boldsymbol{z}). \tag{1.1}$$

此时我们假设 $\boldsymbol{x}$ 与 $\boldsymbol{z}$ 足够近，因此 $P(c|\boldsymbol{x}) \simeq P(c|\boldsymbol{z})$，我们可以得到

$$P(err) \simeq 1 - \sum_{c \in \mathcal{Y}} P^2(c|\boldsymbol{x}). \tag{1.2}$$

令 $c^* = \arg\max_{c\in\mathcal{Y}} P(c|\boldsymbol{x})$ 表示贝叶斯最优分类器的结果，可得

$$\begin{aligned} P(err) &= 1 - \sum_{c\in\mathcal{Y}} P(c|\boldsymbol{x})P(c|\boldsymbol{z}) \\ &\simeq 1 - \sum_{c\in\mathcal{Y}} P^2(c|\boldsymbol{x}) \\ &\leqslant 1 - P^2\left(c^*|\boldsymbol{x}\right) \\ &= \left(1 + P\left(c^*|\boldsymbol{x}\right)\right)\left(1 - P\left(c^*|\boldsymbol{x}\right)\right) \\ &\leqslant 2 \times \left(1 - P\left(c^*|\boldsymbol{x}\right)\right). \end{aligned} \tag{1.3}$$

由此可以得出结论，k 近邻学习中“最近邻分类器”(k = 1) 的泛化错误率小于贝叶斯最优分类器错误率的两倍。对于这样一种简单的学习方法来说，是相当不错的泛化错误率。

1.2 多维伸缩 (MDS)

1.2.1 维数灾难

从上一节中可以看出，为了使 k 近邻学习达到更好的分类效果，选取的数据不能过于稀疏，而是要足够密集（足够近）。假设我们的采样密度是 $\delta = 0.1$，那么在一维空间 $[0,1]$ 中我们需要采 10 个样本，在二维空间 $[0,1]^2$ 中我们需要采 100 个样本，一直到 n 维空间 $[0,1]^n$ 我们就需要 10^n 个样本。因此当数据维度很高时，采样点的个数就会指数型地增长，也就出现了维数灾难的现象。

那么我们应该如何解决这样的问题呢？我们利用了现实数据的特征，即我们观测到的数据样本虽然是高维的，但是其本征维度 (Intrinsic Dimension) 很低。图 1.2是一个简单的例子，这里圆环的数据虽然是在二维的，但如果我们选定环上一点作为起始点，记环上的点为从起始点出发，延逆时针方向经过的距离，就可以用一维的数据来表示环上的点，这就是一种低维“嵌入”。

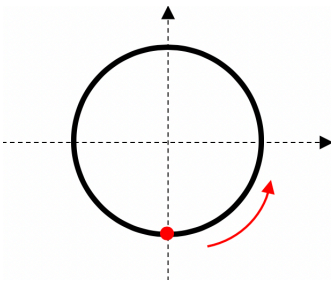


图 1.2: 二维空间中的圆环，本征维度只有一维。

因此，由于数据往往是高维空间中的一个低维“嵌入”，我们可以利用其本征维度 (Intrinsic Dimension) 通常较低的特性，将原始高维属性空间转变为一个低维空间，来避免维数灾难的发生。

1.2.2 多维伸缩学习方法

缓解维数灾难的一个重要途径是降维 (dimension reduction), 我们需要通过某种数学变换将原始高维属性空间转变为一个低维子空间 (subspace)，使得在子空间中的样本密度提高。我们要求原始空间样本之间的距离在低位空间中能够继续保持。这里我们介绍一种典型的降维方法，多维伸缩 (Multidimensional Scaling).

假设在原始空间中，有 m 个样本 $\{x_i\}_{i=1}^m$，$x_i \in \mathbb{R}^d$, 其距离矩阵定义为 $\mathbf{D} \in \mathbb{R}^{m\times m}$. 我们的目标是获得样本在 d' 维空间的表示 $\{z_i\}_{i=1}^m$, $z_i \in \mathbb{R}^{d'}$, 且满足 $d' \leq d$. 其中任意两个样本在 d' 维空间中的距离等于原始空间中的距离, 即

$$
\begin{aligned}
D_{ij} &= dist(x_i, x_j) = (x_i - x_j)^T(x_i - x_j) \\
D'_{ij} &= dist(z_i, z_j) = (z_i - z_j)^T(z_i - z_j) \\
D'_{ij} &= D_{ij}
\end{aligned}
$$

令 $\mathbf{B} = \mathbf{Z}^{\mathrm{T}}\mathbf{Z} \in \mathbb{R}^{m\times m}$ 为降维后的内积矩阵，$b_{ij} = z_i^T z_j$, 因此我们得到

$$
dist^2(x_i, x_j) = dist_{ij}^2 = (z_i - z_j)^T(z_i - z_j) = b_{ii} + b_{jj} - 2b_{ij} \tag{1.4}
$$

为了方便计算，将降维后的样本 $\mathbf{Z}$ 标准化，即 $\sum_{i=1}^m \boldsymbol{z}_i = \mathbf{0}$, 因此矩阵 $\mathbf{B}$ 的各行和各列的和均为零，即 $\sum_{i=1}^m b_{ij} = \sum_{j=1}^m b_{ij} = 0$. 再结合等式 (1.4)，易得

$$
\sum_{i=1}^m dist_{ij}^2 = \mathrm{tr}(\mathbf{B}) + mb_{jj} \tag{1.5}
$$

$$
\sum_{j=1}^m dist_{ij}^2 = \mathrm{tr}(\mathbf{B}) + mb_{ii} \tag{1.6}
$$

$$
\sum_{i=1}^m \sum_{j=1}^m dist_{ij}^2 = 2m\,\mathrm{tr}(\mathbf{B}) \tag{1.7}
$$

联立等式 (1.4), (1.5), (1.6), (1.7) 解得

$$
b_{ij} = -\frac{1}{2}\left(dist_{ij}^2 - \frac{1}{m}\sum_i dist_{ij}^2 - \frac{1}{m}\sum_j dist_{ij}^2 + \frac{1}{m^2}\sum_{i,j} dist_{ij}^2\right) \tag{1.8}
$$

因此我们得到了内积矩阵 $\mathbf{B}$.

对 $\mathbf{B}$ 做特征分解，即 $\mathbf{B}=\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathrm{T}}$, 又因为 $\mathbf{B}=\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$, $\mathbf{B}$ 可以表示为

$$\mathbf{Z}=(\mathbf{\Lambda})^{\frac{1}{2}}(\mathbf{V})^{\mathrm{T}}\in\mathbb{R}^{d\times m},$$

其中 $\mathbf{\Lambda}=\operatorname{diag}(\lambda_1,\lambda_2,\ldots,\lambda_m)$ 为特征值构成的对角矩阵，满足 $\lambda_1\geq\lambda_2\geq\ldots\geq\lambda_m$，$\mathbf{V}$ 为特征向量矩阵. 若这 m 个特征值中有 d' 个非零特征值，只保留这些特征值后，它们构成了 $\mathbf{\Lambda}^*=\operatorname{diag}(\lambda_1,\lambda_2,\ldots,\lambda_{d'})$,

$$\mathbf{Z}=(\mathbf{\Lambda}^*)^{\frac{1}{2}}(\mathbf{V}^*)^{\mathrm{T}}\in\mathbb{R}^{d\times d'}$$

在实际应用中，为了更加有效地对数据进行降维，我们可以将特征值对角矩阵中较小的特征值都舍弃，以实现 $d'\ll m$.

图1.3给出了 MDS 算法的描述。

输入: 距离矩阵 $\mathbf{D}\in\mathbb{R}^{m\times m}$, 其元素 $dist_{ij}$ 为样本 $\boldsymbol{x}_i$ 到 $\boldsymbol{x}_j$ 的距离;
低维空间维数 d'.
过程:
1: 根据式(10.7)~(10.9)计算 $dist_{i\cdot}^2$, $dist_{\cdot j}^2$, $dist_{\cdot\cdot}^2$;
2: 根据式(10.10)计算矩阵 $\mathbf{B}$;
3: 对矩阵 $\mathbf{B}$ 做特征值分解;
4: 取 $\tilde{\mathbf{\Lambda}}$ 为 d' 个最大特征值所构成的对角矩阵, $\tilde{\mathbf{V}}$ 为相应的特征向量矩阵.
输出: 矩阵 $\tilde{\mathbf{V}}\tilde{\mathbf{\Lambda}}^{1/2}\in\mathbb{R}^{m\times d'}$, 每行是一个样本的低维坐标

图 1.3: MDS 算法 (参考周志华《机器学习》).

1.2.3 多维伸缩的优势

MDS 学习法通常选取较小的 d', 可以有效降维缓解数据维度灾难，也因此有更小的计算量。与 k 近邻学习不同，在拿到训练集后，我们可以用上述方法学习数据从原始空间到降维后新空间的映射函数, $f:\mathbf{X}\to\mathbf{Z}$，该训练可以用普通的 DNN 网络实现。

1.3 主成分分析 (PCA)

在之前讲述的降维方法 MDS 中，降维后的数据 Z 与初始数据 X 之间的关系非常不明确，以致我们需要用新的学习器去学习这种关系。假设 Z 与 X 之间满足某种线性关系，如 $Z=WX$，那么对于新的数据点 x', 就能通过 $z'=wx'$ 得到降维后的数据而不需要许多的中间

步骤。这种线性方法当然有一定的局限性，而我们使用这个方法时如何做线性变换是依据在投影方向方差尽量大，而在其他方向上方差尽量小。而这个思路就是接下来细讲的主成分分析。

假定初始数据 $X \in R^{d\times n}$，其中 d 是维度，n 为样本数目。令 $w \in R^d$，则 $w^T X \in R^{1\times n}$ 则是 n 个数据点在这个方向上的投影 (从内积角度理解)。假定 $E(w^T x_i) = 0$(数据样本进行了中心化)，那么方差的表达式为：

$$\frac{1}{n}\sum_{i=1}^{n}(w^T x_i)^2 = \frac{1}{n}\sum_{i=1}^{n} w^T x_i x_i^T w = \frac{1}{n} w^T X X^T w \tag{1.9}$$

这里利用了 $\sum\limits_{i=1}^{n} x_i x_i^T = XX^T$, 这个表达式可以通过 X 的每一列为一个数据点 X^T 的每一行为一个数据点通过矩阵乘法得到。则现在目标为 $w^\star = \arg\max_{||w||_2=1}\ w^T X X^T w$, 令 $\Sigma = XX^T \in S_+^d$ 为对称半正定矩阵，则 $w^\star = \arg\max_{||w||_2=1}\ w^T \Sigma w$。下面对协方差矩阵 Σ 做特征分解以及将 w 用特征向量表示为：

$$\Sigma = \sum \lambda_i v_i v_i^T = V \wedge V^T,\quad w = \sum \alpha_i v_i \tag{1.10}$$

其中 $||w||_2 = \sum \alpha_i^2 \le 1$。则有：

$$\begin{aligned} w^T \Sigma w &= \sum_i \alpha_i v_i^T \sum_j \lambda_j v_j v_j^T \sum_k \alpha_k v_k \\ &= \sum_{i,j,k} \alpha_i \alpha_k \lambda_j (v_i^T v_j)(v_j^T v_k) \\ &= \sum_i \alpha_i^2 \lambda_i \le \lambda_{max} \sum_i \alpha_i^2 \end{aligned} \tag{1.11}$$

取等条件为 $w = v_t, t = argmax_t\ \lambda_t$。在 $w = v_t$ 时，新的数据点为 $Z_t = v_t^T X$ ，方差为：

$$Z_t Z_t^T = v_t^T X X^T v_t = v_t^T \Sigma v_t = \lambda_t v_t^T v_t = \lambda_t \tag{1.12}$$

更一般地考虑 X 在每个特征方向上的投影，则 $Z = V^T X$，方差为：

$$Cov(Z) = ZZ^T = V^T X X^T V = V^T \Sigma V = V V^T \wedge V^T V = \wedge \tag{1.13}$$

方差为对角阵说明 Z 的各个分量不想关，便于处理。下面给两点注解：

- 考虑与 MDS 的差异，PCA 是通过协方差矩阵 Σ 找到一个投影矩阵 V，则降维后的数据 $Z = V^T X$, 但 MDS 需要新的学习器找到 X 与 Z 之间的关系。
- 在 PCA 中对称半正定矩阵 Σ 是很关键的，$\Sigma = \sum x_i x_i^T = XX^T$。

显然 PCA 也有一定的局限性，在一些非线性问题中，如果使用 PCA 降维会损失大量的信息。那么对于这种问题，则需考虑下述的核主成分分析 (Kernelized PCA)。

1.4 核主成分分析

非线性降维的一种常用的方法，是基于核技巧对线性降维方法进行“核化”。下面展示核主成分分析法。

基于前述的 PCA 相关内容，由 $\Sigma v_k = \lambda_k v_k, \Sigma = \frac{1}{m}\sum_{i=1}^{m} x_i x_i^T$(m 为样本数据数量) 可得：

$$v_k = \frac{1}{\lambda_k v_k}\sum_{i=1}^{m} x_i(x_i^T v_k) = \sum_{i=1}^{m}\alpha_i^k x_i \tag{1.14}$$

其中 $\alpha_i^k = \frac{x_i^T v_k}{m\lambda_k}$ 可以理解为常数，则特征方向可以用数据点的组合表示，这一点也可以通过 $rank(\Sigma) \leq min\{m, d\}$ 来理解。

令 $K_{ij} = x_i^T x_j, K = (K_{ij}), \alpha^k = (\alpha_1^k, \ldots, \alpha_m^k)^T$，则 $\forall k$，有：

$$\lambda_k x_j^T v_k = x_j^T \sum_i \alpha_i^k x_i = \lambda_k \sum_i \alpha_i^k x_j^T x_i = \lambda_k \sum_i \alpha_i^k K_{ji} = \lambda_k [K\alpha^k]_j \tag{1.15}$$

$$\begin{aligned} x_j^T(\lambda_k v_k) &= x_j^T(\frac{1}{m}\sum_{i=1}^{m} x_i x_i^T v_k) \\ &= x_j^T \frac{1}{m}\sum_{i=1}^{m} x_i x_i^T \sum_l \alpha_l^k x_l \\ &= \sum_{i,l}^{m} \frac{1}{m}(x_j^T x_i)(x_i^T x_l)\alpha_l^k \\ &= \frac{1}{m}\sum_{i,l}^{m} K_{ji}K_{il}\alpha_l^k \\ &= [\frac{1}{m}K^2\alpha^k]_j \end{aligned} \tag{1.16}$$

所以比较上述两式可得：

$$\begin{aligned} \frac{1}{m}K^2\alpha^k &= \lambda_k K\alpha_k \\ K[K\alpha^k - m\lambda_k\alpha^k] &= 0 \end{aligned} \tag{1.17}$$

再令 $\tilde{\lambda}_k = m\lambda_k$，并假设 K 是满秩的，则有：

$$K\alpha^k = \tilde{\lambda}_k\alpha^k \tag{1.18}$$

从上述表达式中可以看出，我们只需要求解 K 的特征值及特征向量也可以得到投影方向 V。那么对于一些非线性的问题，我们可以将 x 投影到高维空间，即 $x \mapsto \phi(x)$。令 $k(x_i, x_j) =$

$\phi(x_i)^T\phi(x_j)$ 为核函数，$K_{ij} = k(x_i, x_j)$ 为核矩阵，同样求解上述问题即可，当取出 K 最大的 d' 个特征值对应的特征向量 $(v_1, \ldots, v_{d'})$ 后，对新的样本 x，则有其投影后的第 j($j = 1, 2, \ldots, d'$) 维坐标为:

$$z_j = v_j^T\phi(x) = \sum_{i=1}^{m} \alpha_i^j \phi(x_i)^T\phi(x) = \sum_{i=1}^{m} \alpha_i^j k(x_i, x) \tag{1.19}$$

这种 Kernelized PCA 方法在实际应用中较少，主要原因有 Kernal 函数不易选取以及从上述表达式能看出计算一个坐标需要对所有的样本求和，计算开销较大。

下面就 PCA 与数据学习谈一点理解，在 PCA 所得到主成分 (或者是数据的切线方向) 上数据是比较丰富的，但是在法向上数据是很少 (或者说很集中) 的也就很不稳定。所以如果要攻击一个学习算法的话从法向攻击就会比切线攻击有效得多。

1.5 流形学习

流形学习 (mainfold learning) 是一类借鉴了拓扑流形概念的降维方法，“流形”是在局部与欧式空间同胚的空间，这样就能利用欧式距离来进行距离计算，给降维带来了很大的启发。我们考虑这样的实际问题，在如下图中低维流形嵌入到高维空间的距离是哪一条线。显然我们更倾向于选择流形上的线，因为高维直线距离在低维嵌入流形中是不可达的。

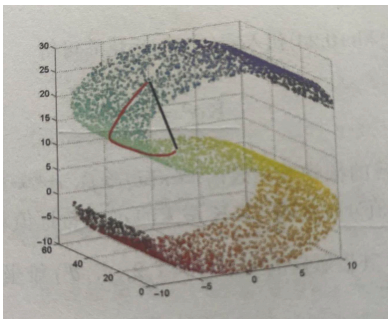


图 1.4: 测地线距离与高维直线距离

上述便是是等度量映射 (Isometric Mapping) 这种算法的基本出发点。这种算法的思想为在近邻连接图上考虑，基于已知的样本集 $D = \{x_i\}_{i=1}^m$，近邻参数 k 以及低维空间的维度 d'，首先对每个 x_i 确定其 k 近邻，然后将 x_i 与 k 近邻点的距离设置为欧式距离而其他点为无穷大；接着调用最短路径算法求 $dist(x_i, x_j)(\forall i, j)$；然后对此进行 MDS 降维，最后输出数据点的 MDS 表示。具体的过程步骤如下图 (摘自周志华老师《机器学习》)：

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

过程：

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

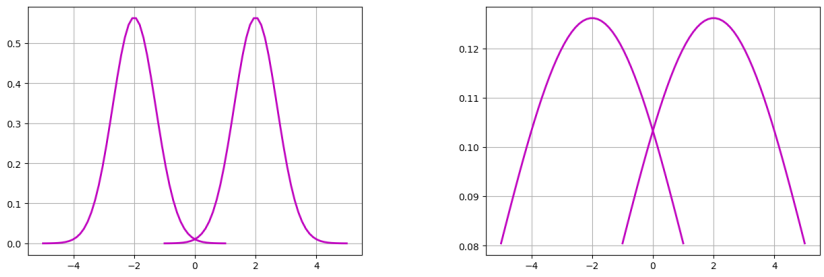
输出：样本集 D 在低维空间的投影 $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m\}$.

图 1.5: Isomap 算法

一般的近邻图的构建有两种，一是指定近邻点的个数为 k 个，称为 k 近邻图；另一种是指定距离阈值 ϵ, 距离小于 ϵ 的点被称为近邻点，称为 ϵ 近邻图。但第二种近邻图会遇到一些问题，比如 ϵ 过大会使距离比较远的点被误认为近邻点，出现“短路问题”；而 ϵ 过小可能会导致有些点无近邻点，致使有些区域无连接，出现“断路”问题。

1.6 度量学习

考虑如下的实际问题，有两个不同分布如下图：



(a) 标准差 0.5　　(b) 标准差 10

图 1.6: 不同方差的正态分布

在每个图中两个正态分布的均值差相等，那么以 d 作为两个分布的距离就显然不合适，因为

很明显第一张图的两个分布容易分类一些。而对于一般的数据点 x_i, x_j，如果仅仅考虑距离为 $(x_i - x_j)^T(x_i - x_j)$ 的话就是将每个维度都视作一样的，而有时我们需要就具体的维度特征考虑。首先考虑各个维度是独立的话，我们可以将方差除掉，即考虑距离为 $(x_i - x_j)^T\Sigma^{-1}(x_i - x_j)$。这可以考虑为新的数据点 $Z = \Sigma^{-\frac{1}{2}}X$，则有这些数据点的方差为 $ZZ^T = I$, 距离为 $Z^TZ = X^T\Sigma^{-1}X$。这个距离也被称之为马氏距离。

更一般的我们可以考虑 M 距离为 $||x||_M = x^TMx$，其中为了保证距离非负对称性，M 为一半正定对称矩阵。度量学习本质上就是学习度量矩阵 M，该如何学习 M 呢？比如在一个样本集中，我们希望相似样本之间的距离较小，不相似的样本之间的距离较大，于是可以做如下的优化问题以获得适当的距离矩阵 M：

$$\min_M \sum_{(x_i,x_j)\in\mathcal{M}} ||x_i - x_j||_M^2 \ st \sum_{(x_i,x_k)\in\mathcal{C}} ||x_i - x_k||_M^2 \geq 1. \tag{1.20}$$

其中 $(x_i, x_j) \in \mathcal{M}$ 表示两点相似，而 $(x_i, x_k) \in \mathcal{C}$ 表示两点不相似，$M \geq 0$ 为半正定的。

1.7 课堂常见问题

1. **Q:** MDS 是如何处理新添加的数据的？/用神经网络学习原始空间到降维后的空间合理吗？

   A: 在 MDS 中，我们可以用 DNN 在训练集上学习从原始空间 $\mathbf{X}$ 到新空间 $\mathbf{Z}$ 的降维变换函数，即 $T : \mathbf{X} \to \mathbf{Z}$. 这样即便我们在测试集上获得了新的数据，也可以用训练好的函数去降维。

2. **Q:** KNN(k=1) 的时候为什么假设 $\boldsymbol{x}$ 和 $\boldsymbol{z}$ 足够近说明了 $P(c|\boldsymbol{x}) \simeq P(c|\boldsymbol{z})$？

   A: 首先我们假设了样本独立同分布，若对任意 $\boldsymbol{x}$ 总能找到足够近的 $\boldsymbol{z}$，其数学含义即对任意 $\boldsymbol{x}$ 和任意小正数 δ, 在 $\boldsymbol{x}$ 附近 δ 距离范围内总能找到一个训练样本; 换言之, 对任意测试样本 $\boldsymbol{x}$, 总能在任意近的范围内找到式训练样本 $\boldsymbol{z}$ 满足 $P(c|\boldsymbol{x}) \simeq P(c|\boldsymbol{z})$。

3. **Q:** 如何理解在 PCA 与数据学习中攻击算法在法线上更有效呢？

   A: 所谓攻击算法应该是进行某些操作后得到的算法与原算法有差异。而因为在 PCA 所得到主成分 (或者是数据的切线方向) 上数据是比较丰富的，但是在法向上数据是很少 (或者说很集中) 的也就很不稳定，所以从法向攻击就会比切线攻击有效得多。举个例子比如说你在切线上新增一个点不会影响 PCA 算法所找到的主成分方向，但如果在法线方向增加了一个点，就很大可能会影响到之前所找到的投影方向。