第9章　无监督学习

> 大脑有大约10^{14}个突触，我们只能活大约10^9秒。所以我们有比数据更多的参数。这启发了我们必须进行大量无监督学习的想法，因为感知输入（包括本体感受）是我们可以获得每秒10^5维约束的唯一途径。
>
> — Geoffrey Hinton, 2014 AMA on Reddit

更早的正式描述见 [Hinton et al., 1999]

无监督学习（Unsupervised Learning）是指从无标签的数据中学习出一些有用的模式。无监督学习算法一般直接从原始数据中学习，不借助于任何人工给出标签或者反馈等指导信息。如果监督学习是建立输入-输出之间的映射关系，无监督学习就是发现隐藏的数据中的有价值信息，包括有效的特征、类别、结构以及概率分布等。

典型的无监督学习问题可以分为以下几类：

无监督特征学习 无监督特征学习（Unsupervised Feature Learning）是从无标签的训练数据中挖掘有效的特征或表示。无监督特征学习一般用来进行降维、数据可视化或监督学习前期的数据预处理。

特征学习也包含很多的监督学习算法，比如线性判别分析等。

密度估计 密度估计（Density Estimation）是根据一组训练样本来估计样本空间的概率密度。密度估计可以分为参数密度估计和非参数密度估计。参数密度估计是假设数据服从某个已知概率密度函数形式的分布（比如高斯分布），然后根据训练样本去估计概率密度函数的参数。非参数密度估计是不假设数据服从某个已知分布，只利用训练样本对密度进行估计，可以进行任意形状密度的估计。非参数密度估计的方法有直方图、核密度估计等。

聚类 聚类（Clustering）是将一组样本根据一定的准则划分到不同的组（也称为集群（cluster））。一个比较通用的准则是组内的样本的相似性要高于组间样本的相似性。常见的聚类算法包括k-means算法、谱聚类等。

和监督学习一样，无监督学习方法也包含三个基本要素：模型、学习准则和优化算法。无监督学习的准则非常多，比如最大似然估计、最小重构错误等。在无监督特征学习中，经常使用的准则为最小化重构错误，同时也经常对特征进行一些约束，比如独立性、非负性或稀释性等。而在密度估计中，经常采用最大似然估计来进行学习。

本章介绍两种无监督学习问题：无监督特征学习和密度估计。

9.1 无监督特征学习

无监督特征学习是指从无标注的数据中自动学习有效的数据表示，从而能够帮助后续的机器学习模型更快速地达到更好的性能。无监督特征学习主要方法有主成分分析、稀疏编码、自编码器等。

9.1.1 主成分分析

主成份分析（Principal Component Analysis，PCA）一种最常用的数据降维方法，使得在转换后的空间中数据的方差最大。如图9.1所示的两维数据，如果将这些数据投影到一维空间中，选择数据方差最大的方向进行投影，才能最大化数据的差异性，保留更多的原始数据信息。

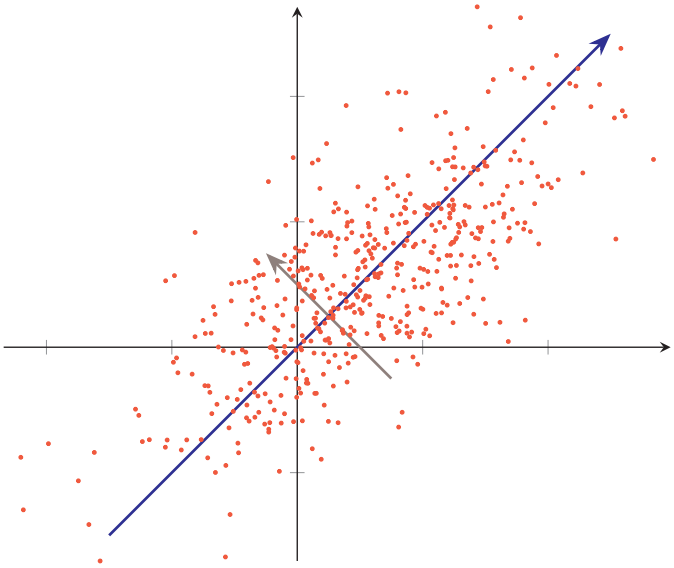


图 9.1 主成分分析

假设有一组 d 维的样本 $\mathbf{x}^{(n)} \in \mathbb{R}^d, 1 \leq n \leq N$，我们希望将其投影到一维空间中，投影向量为 $\mathbf{w} \in \mathbb{R}^d$。不失一般性，可以限制 $\mathbf{w}$ 的模为 1，即 $\mathbf{w}^{\mathrm{T}}\mathbf{w} = 1$。

每个样本点 $\mathbf{x}^{(n)}$ 投影之后的表示为

$$z^{(n)} = \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)}. \tag{9.1}$$

我们用矩阵 $X = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(N)}]$ 表示输入样本，$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(n)}$ 为原始样本的中心点，所有样本投影后的方差为

$$\sigma(X;\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} - \mathbf{w}^{\mathrm{T}}\bar{\mathbf{x}})^2 \tag{9.2}$$

$$= \frac{1}{N}(\mathbf{w}^{\mathrm{T}}X - \mathbf{w}^{\mathrm{T}}\bar{X})(\mathbf{w}^{\mathrm{T}}X - \mathbf{w}^{\mathrm{T}}\bar{X})^{\mathrm{T}} \tag{9.3}$$

$$= \mathbf{w}^{\mathrm{T}}S\mathbf{w}, \tag{9.4}$$

其中 $\bar{X} = \bar{\mathbf{x}}\mathbf{1}_d^{\mathrm{T}}$ 为 d 列 $\bar{\mathbf{x}}$ 组成的矩阵，$S = \frac{1}{N}(X - \bar{X})(X - \bar{X})^{\mathrm{T}}$ 是原始样本的协方差矩阵。

最大化投影方差 $\sigma(X;w)$ 并满足 $\mathbf{w}^{\mathrm{T}}\mathbf{w} = 1$，利用拉格朗日方法转换为无约束优化问题，

$$\max_{\mathbf{w}} \mathbf{w}^{\mathrm{T}}S\mathbf{w} + \lambda(1 - \mathbf{w}^{\mathrm{T}}\mathbf{w}), \tag{9.5}$$

其中 λ 为拉格朗日乘子。对上式求导并令导数等于 0，可得

$$S\mathbf{w} = \lambda\mathbf{w}. \tag{9.6}$$

从上式可知，$\mathbf{w}$ 是协方差矩阵 S 的特征向量，λ 为特征值。同时

$$\sigma(X;w) = \mathbf{w}^{\mathrm{T}}S\mathbf{w} = \mathbf{w}^{\mathrm{T}}\lambda\mathbf{w} = \lambda. \tag{9.7}$$

λ 也是投影后样本的方差。因此，主成分分析可以转换成一个矩阵特征值分解问题，投影向量 $\mathbf{w}$ 为矩阵 S 的最大特征对应的特征向量。

如果要通过投影矩阵 $W \in R^{d\times d'}$ 将样本投到 d' 维空间，投影矩阵满足 $W^{\mathrm{T}}W = \mathbf{I}$，只需要将 S 的特征值从大到小排列，保留前 d' 个特征向量，其对应的特征向量即使最优的投影矩阵。

$$SW = W\,\mathrm{diag}(\Lambda), \tag{9.8}$$

其中 $\Lambda = [\lambda_1, \cdots, \lambda_{d'}]$ 为 S 的前 d' 个最大的特征值。

主成分分析是一种无监督学习方法，可以作为监督学习的数据预处理方法，用来去除噪声并减少特征之间的相关性，但是它并不能保证投影后数据的类别可分性更好。提高两类可分性的方法一般为监督学习方法，比如线性判别分析（Linear Discriminant Analysis，LDA）。

参见习题9-3。

9.1.2 稀疏编码

带通滤波（bandpass filter）是指容许某个频率范围的信号通过，同时屏蔽其他频段的设备。

稀疏编码（Sparse Coding）也是一种受哺乳动物视觉系统中简单细胞感受野而启发的模型。在哺乳动物的初级视觉皮层（primary visual cortex）中，每个神经元仅对处于其感受野中特定的刺激信号做出响应，比如特定方向的边缘、条纹等特征。局部感受野可以被描述为具有空间局部性、方向性和带通性（即不同尺度下空间结构的敏感性）[Olshausen et al., 1996]。也就是说，外界信息经过编码后仅有一小部分神经元激活，即外界刺激在视觉神经系统的表示具有很高的稀疏性。编码的稀疏性在一定程度上符合生物学的低功耗特性。

在数学上，（线性）编码是指给定一组基向量 $A = [\mathbf{a}_1, \cdots, \mathbf{a}_p]$，将输入样本 $\mathbf{x} \in \mathbb{R}^d$ 表示为这些基向量的线性组合

$$\mathbf{x} = \sum_{i=1}^{p} z_i \mathbf{a}_i \tag{9.9}$$

$$= A\mathbf{z}, \tag{9.10}$$

其中基向量的系数 $\mathbf{z} = [z_1, \cdots, z_p]$ 称为输入样本 $\mathbf{x}$ 的编码（encoding），基向量 A 也称为字典（dictionary）。

编码是对 d 维空间中的样本 $\mathbf{x}$ 找到其在 p 维空间中的表示（或投影），其目标通常是编码的各个维度都是统计独立的，并且可以重构出输入样本。编码的关键是找到一组“完备”的基向量 A，比如主成分分析等。但是主成分分析得到编码通常是稠密向量，没有稀疏性。

数学小知识 | 完备性

如果 p 个基向量刚好可以支撑 p 维的欧氏空间，则这 p 个基向量是完备的。如果 p 个基向量可以支撑 d 维的欧氏空间，并且 $p > d$，则这 p 个基向量是过完备的，冗余的。

“过完备”基向量一般是指的是基向量个数远远大于其支撑空间维度。因此这些基向量一般是不具备独立、正交等性质。

为了得到稀疏的编码，我们需要找到一组“超完备”的基向量（即 $p > d$）来进行编码。在超完备基向量之间往往会存在一些冗余性，因此对于一个输入样本，会存在很多有效的编码。如果加上稀疏性限制，就可以减少解空间的大小，得到“唯一”的稀疏编码。

给定一组 N 个输入向量 $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}$，其稀疏编码的目标函数定义为：

$$L(A, Z) = \sum_{n=1}^{N} \left(\left\| \mathbf{x}^{(n)} - A\mathbf{z}^{(n)} \right\|^2 + \eta \rho(\mathbf{z}^{(n)}) \right), \tag{9.11}$$

其中 $\rho(\cdot)$ 是一个稀疏性衡量函数，η 是一个超参数，用来控制稀疏性的强度。

对于一个向量 $\mathbf{z} \in \mathbb{R}^p$，其稀疏性定义为非零元素的比例。如果一个向量只有很少的几个非零元素，就说这个向量是稀疏的。稀疏性衡量函数 $\rho(\mathbf{z})$ 是给向量 $\mathbf{z}$ 一个标量分数。$\mathbf{z}$ 越稀疏，$\rho(\mathbf{z})$ 越小。

严格的稀疏向量有时比较难以得到，因此如果一个向量只有少数几个远大于零的元素，其它元素都接近于 0，我们也称这个向量为稀疏向量。

稀疏性衡量函数有多种选择，最直接的衡量向量 $\mathbf{z}$ 稀疏性的函数是 ℓ_0 范式

$$\rho(\mathbf{z}) = \sum_{i=1}^{p} \mathbf{I}(|z_i| > 0)) \tag{9.12}$$

但 ℓ_0 范数不满足连续可导，因此很难进行优化。在实际中，稀疏性衡量函数通常使用 ℓ_1 范数

$$\rho(\mathbf{z}) = \sum_{i=1}^{p} |z_i| \tag{9.13}$$

或对数函数

$$\rho(\mathbf{z}) = \sum_{i=1}^{p} \log(1 + z_i^2) \tag{9.14}$$

或指数函数

$$\rho(\mathbf{z}) = \sum_{i=1}^{p} -\exp(-z_i^2). \tag{9.15}$$

9.1.2.1 训练方法

给定一组 N 个输入向量 $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}$，需要同时学习基向量 A 以及每个输入样本对应的稀疏编码 $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)}$。

稀疏编码的训练过程一般用交替优化的方法进行。

1) 固定基向量 A，对每个输入 $\mathbf{x}^{(n)}$，计算其对应的最优编码

$$\min_{\mathbf{x}^{(n)}} \left\| \mathbf{x}^{(n)} - A\mathbf{z}^{(n)} \right\|^2 - \eta \rho(\mathbf{z}^{(n)}), \ \forall n \in [1, N]. \tag{9.16}$$

2) 固定上一步得到的编码 $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)}$，计算其最优的基向量

$$\min_{A} \sum_{n=1}^{N} \left(\left\| \mathbf{x}^{(n)} - A\mathbf{z}^{(n)} \right\|^2 \right) + \lambda \frac{1}{2} \|A\|^2, \tag{9.17}$$

其中第二项为正则化项，λ 为正则化项系数。

9.1.2.2 稀疏编码的优点

稀疏编码的每一维都可以看作是一种特征。和基于稠密向量的分布式表示相比，稀疏编码具有更小的计算量和更好的可解释性等优点。

计算量 稀疏性带来的最大好处就是可以极大地降低计算量。

可解释性 因为稀疏编码只有少数的非零元素，相当于将一个输入样本表示为少数几个相关的特征。这样我们可以更好地描述其特征，并易于理解。

特征选择 稀疏性带来的另外一个好处是可以实现特征的自动选择，只选择和输入样本相关的最少特征，从而可以更好地表示输入样本，降低噪声并减轻过拟合。

9.1.3 自编码器

自编码器（Auto-Encoder，AE）是通过无监督的方式来学习一组数据的有效编码（或表示）。

假设有一组 d 维的样本 $\mathbf{x}^{(n)} \in \mathbb{R}^d, 1 \leq n \leq N$，自编码器将这组数据映射到特征空间得到每个样本的编码 $\mathbf{z}^{(n)} \in \mathbb{R}^p, 1 \leq n \leq N$，并且希望这组编码可以重构出原来的样本。

自编码器的结构可分为两部分：

编码器（encoder）

$$f : \mathbb{R}^d \to \mathbb{R}^p, \tag{9.18}$$

和解码器（decoder）

$$g : \mathbb{R}^p \to \mathbb{R}^d. \tag{9.19}$$

自编码器的学习目标是最小化重构错误（reconstruction errors）

$$\mathcal{L} = \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - g(f(\mathbf{x}^{(n)}))\|^2 \tag{9.20}$$

$$= \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - f \circ g(\mathbf{x}^{(n)})\|^2. \tag{9.21}$$

如果特征空间的维度 p 小于原始空间的维度 d，自编码器相当于是一种降维或特征抽取方法。如果 $p \geq d$，一定可以找到一组或多组解使得 $f \circ g$ 为单位函数（Identity Function），并使得重构错误为 0。但是，这样的解并没有太多的意义。但是如果再加上一些附加的约束，就可以得到一些有意义的解，比如编

单位函数 $I(x) = x$。

码的稀疏性、取值范围，f 和 g 的具体形式等。如果我们让编码只能取 k 个不同的值 $(k < N)$，那么自编码器就可以转换为一个 k 类的聚类问题。

最简单的自编码器是如图9.2所示的两层神经网络。输入层到隐藏层用来编码，隐藏层到输出层用来解码，层与层之间互相全连接。

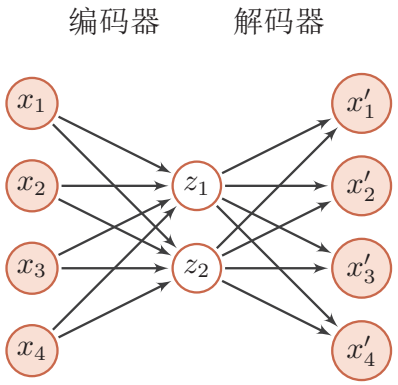


图 9.2 两层网络结构的自编码器

对于样本 $\mathbf{x}$，中间隐藏层为编码

$$\mathbf{z} = s(W^{(1)}\mathbf{x} + b^{(1)}), \tag{9.22}$$

输出为重构的数据

$$\mathbf{x}' = s(W^{(2)}\mathbf{z} + b^{(2)}), \tag{9.23}$$

其中 W, b 为网络参数，$s(\cdot)$ 为激活函数。如果令 $W^{(2)}$ 等于 $W^{(1)}$ 的转置，即 $W^{(2)} = W^{(1)\mathrm{T}}$，称为*捆绑权重*（tied weights）。

给定一组样本 $\mathbf{x}^{(n)} \in [0,1]^d, 1 \leq n \leq N$，其重构错误为

$$\mathcal{L} = \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - \mathbf{x}'^{(n)})\|^2 + \lambda \|W\|_F^2. \tag{9.24}$$

其中 λ 为正则化项系数。通过最小化重构错误，可以有效地学习网络的参数。

9.1.4 稀疏自编码器

自编码器除了可以学习低维编码之外，也学习高维的稀疏编码。假设中间隐藏层 $\mathbf{z}$ 的维度为 p 大于输入样本 $\mathbf{x}$ 的维度 d，并让 $\mathbf{z}$ 尽量稀疏，这就是*稀疏自编码器*（Sparse Auto-Encoder）。和稀疏编码一样，稀疏自编码器的优点是有很高的可解释性，并同时进行了隐式的特征选择。

通过给自编码器中隐藏层单元 $\mathbf{z}$ 加上稀疏性限制，自编码器可以学习到数据中一些有用的结构。

$$\mathcal{L} = \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - \mathbf{x}'^{(n)})\|^2 + \eta\rho(\mathbf{z}^{(n)})) + \lambda\|W\|^2, \tag{9.25}$$

其中 $\rho(\cdot)$ 为稀疏性度量函数，W 表示自编码器中的参数。

稀疏性度量函数 $\rho(\cdot)$ 除了可以选择公式 (9.13)-(9.15) 的定义外，还可以定义为一组训练样本中每一个神经元激活的频率。

给定 N 个训练样本，隐藏层第 j 个神经元平均活性值为

$$\hat{\rho}_j = \frac{1}{N}\sum_{n=1}^{N} z_j^{(n)}, \tag{9.26}$$

$\hat{\rho}_j$ 可以近似地看作是第 j 个神经元激活的概率。我们希望 $\hat{\rho}_j$ 接近于一个事先给定的值 ρ^*，比如 0.05，可以通过 KL 距离来衡量 $\hat{\rho}_j$ 和 ρ^* 的差异，即

$$\mathrm{KL}(\rho^*||\hat{\rho}_j) = \rho^* \log \frac{\rho^*}{\hat{\rho}_j} + (1-\rho^*) \log \frac{1-\rho^*}{1-\hat{\rho}_j}. \tag{9.27}$$

如果 $\hat{\rho}_j = \rho^*$，则 $\mathrm{KL}(\rho^*||\hat{\rho}_j) = 0$。

稀疏性度量函数定义为

$$\rho(\mathbf{z}^{(n)})) = \sum_{j=1}^{p} \mathrm{KL}(\rho^*||\hat{\rho}_j). \tag{9.28}$$

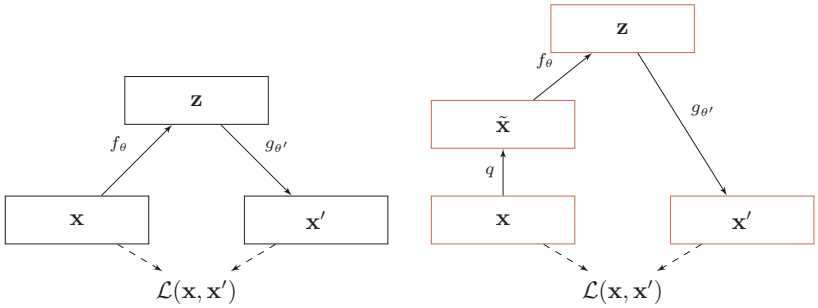
9.1.5 堆叠自编码器

对于很多数据来说，仅使用两层神经网络的自编码器还不足以获取一种好的数据表示。为了获取更好的数据表示，我们可以使用更深层的神经网络。深层神经网络作为自编码器提取的数据表示一般会更加抽象，能够更好地捕捉到数据的语义信息。在实践中经常使用逐层堆叠的方式来训练一个深层的自编码器，称为*堆叠自编码器*（Stacked Auto-Encoder，SAE）。堆叠自编码一般可以采用*逐层训练*（layer-wise training）来学习网络参数 [Bengio et al., 2007]。

9.1.6 降噪自编码器

待补充

Vincent et al. [2008]



(a) 自编码器 (b) 降噪自编码器

图 9.3 自编码器 VS 降噪自编码器。f_θ 为编码器，$g_{\theta'}$ 为解码器。$\mathcal{L}(\mathbf{x}, \mathbf{x}')$ 为重构错误。

9.2 密度估计

概率密度估计（Probabilistic Density Estimation），简称*密度估计*（Density Estimation），是基于一些观测样本来估计一个随机变量的概率密度函数。密度估计在数据建模、机器学习中使用广泛。

密度估计方法可以分为两类：参数密度估计和非参数密度估计。

9.2.1 参数密度估计

参数密度估计（Parametric Density Estimation）是根据先验知识假设随机变量服从某种分布，然后通过训练样本来估计分布的参数。

令 $\mathcal{D} = \{\mathbf{x}^{(n)}\}, 1 \leq n \leq N$ 为从某个未知分布中独立抽取的 N 个训练样本，我们假设这些样本服从一个概率分布函数 $p(\mathbf{x}|\theta)$，其对数似然函数为

$$\log p(\mathcal{D}|\theta) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}). \tag{9.29}$$

我们要估计一个参数 θ^{ML} 来使得

$$\theta^{ML} = \arg\max_{\theta} \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}|\theta). \tag{9.30}$$

这样参数估计问题就转化为最优化问题。

9.2.1.1 正态分布

假设样本 $\mathbf{x} \in \mathbb{R}^d$ 服从正态分布

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big), \tag{9.31}$$

其中 $\boldsymbol{\mu}$ 和 Σ 分别为正态分布的均值和方差。其对数似然函数为

$$\log p(\mathcal{D}|\boldsymbol{\mu},\Sigma) = -\frac{N}{2}\log\Big((2\pi)^2|\Sigma|\Big) - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}). \tag{9.32}$$

分别上式关于 $\boldsymbol{\mu},\Sigma$ 的偏导数，并令其等于 0。可得，

$$\boldsymbol{\mu}^{ML} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(n)}, \tag{9.33}$$

$$\Sigma^{ML} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}. \tag{9.34}$$

则 X 服从多项分布，其概率分布为

$$p(x_1,\ldots,x_k|\boldsymbol{\mu}) = \frac{n!}{x_1!\cdots x_k!}\mu_1^{x_1}\cdots\mu_k^{x_k}, \tag{9.35}$$

9.2.1.2 多项分布

多项分布参见第D.2.2.1节。

假设样本服从 K 个状态的多项分布。这里我们用 onehot 向量 $\mathbf{x}\in[0,1]^K$ 来表示第 k 个状态，即 $x_k=1$，其余 $x_{i,i\neq k}=0$。样本 $\mathbf{x}$ 的概率密度函数为

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^{K}\mu_k^{x_k}, \tag{9.36}$$

其中 μ_k 为第 k 个状态的概率，并满足 $\sum_{k=1}^{K}\mu_k=1$。

这里没有多项式系数。

数据集 $\mathcal{D}=\{\mathbf{x}^{(n)}\}_{n=1}^{N}$ 的对数似然函数为

$$\log p(\mathcal{D}|\boldsymbol{\mu}) = \log\sum_{n=1}^{N}\sum_{k=1}^{K}x_k^{(n)}\log(\mu_k), \tag{9.37}$$

拉格朗日乘子参考??。

多项分布的参数估计为约束优化问题。引入拉格朗日乘子 λ，将原问题转换为无约束优化问题。

$$\max_{\boldsymbol{\mu},\lambda}\quad\sum_{n=1}^{N}\sum_{k=1}^{K}x_k^{(n)}\log(\mu_k) + \lambda\Big(\sum_{k=1}^{K}\mu_k - 1\Big). \tag{9.38}$$

分别上式关于 μ_k,λ 的偏导数，并令其等于 0。可得，

$$\mu_k^{ML} = \frac{m_k}{N}, \qquad 1\leq k\leq K \tag{9.39}$$

其中 $m_k=\sum_{n=1}^{N}x_k^{(n)}$ 为数据集中取值为第 k 个状态的样本数量。

在实际应用中，参数密度估计一般存在以下问题：（1）模型选择问题，即如何选择数据分布的密度函数。实际数据的分布往往是非常复杂的，而不是简单的正态分布或多项分布。（2）维度灾难问题，即高维数据的参数估计十分困难。随着维度的增加，估计参数所需要的样本数量指数增加。在样本不足时会出现过拟合。

9.2.2 非参数密度估计

非参数密度估计（Nonparametric Density Estimation）是不假设数据服从某种分布，通过将样本空间划分为不同的区域并估计每个区域的概率来近似数据的概率密度函数。

对于高维空间中的一个随机向量$\mathbf{x}$，假设其服从一个未知分布$p(\mathbf{x})$，则$\mathbf{x}$落入空间中的小区域$\mathcal{R}$的概率为

$$P = \int_{\mathcal{R}} p(\mathbf{x}) d\mathbf{x}. \tag{9.40}$$

给定N个训练样本$\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$,落入区域$\mathcal{R}$的样本数量$K$服从二项分布

$$P_K = \binom{N}{K} P^K (1-P)^{1-K}, \tag{9.41}$$

其中K/N的期望为$\mathbb{E}[K/N] = P$，方差为$\mathrm{var}(K/N) = P(1-P)/N$。当$N$非常大时，我们可以近似认为

$$P \approx \frac{K}{N}. \tag{9.42}$$

假设区域$\mathcal{R}$足够小，其内部的概率密度是相同的，则有

$$P \approx p(\mathbf{x})V, \tag{9.43}$$

其中V为区域$\mathcal{R}$的体积。结合上述两个公式，得到

$$p(\mathbf{x}) \approx \frac{K}{NV}. \tag{9.44}$$

根据公式(9.44)，要准确地估计$p(\mathbf{x})$需要尽量使得样本数量N足够大，区域体积V尽可能地小。但在具体应用中，样本数量一般是有限的，过小的区域会导致落入该区域的样本比较少，这样估计的概率密度就不太准确。因此，实践中非参数密度估计通常使用两种方式：（1）固定区域大小V，统计落入不同区域的数量，这种方式包括直方图方法和核方法两种。（2）改变区域大小以使得落入每个区域的样本数量为K，这种方式称为K近邻方法。

9.2.2.1 直方图方法

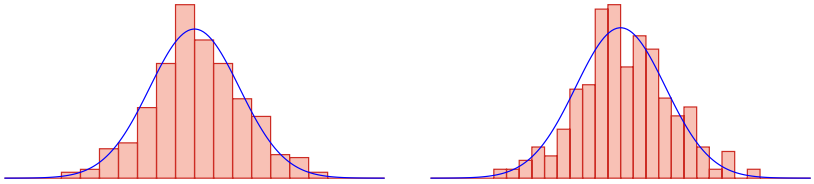
直方图方法（Histogram Method）是一种非常直观的估计连续变量密度函数的方法，可以表示为一种柱状图。

Histogram 源自希腊语 histos（竖立）和 gramma（描绘），由英国统计学家卡尔·皮尔逊于 1895 年提出。

以一维随机变量为例，首先将其取值范围分成M个连续的、不重叠的区间（bin），每个区间的宽度为Δ_m。给定N个训练样本$\mathcal{D} = \{x^{(n)}\}_{n=1}^{N}$，我们统计这些样本落入每个区间的数量$K_m$，然后将它们归一化为密度函数。

$$p_m = \frac{K_m}{N\Delta_m}, \qquad 1 \leq m \leq M \tag{9.45}$$

其中区间宽度 Δ_m 通常设为相同的值 Δ。直方图方法的关键问题是如何选取一个合适的区间宽度 Δ。如果 Δ 太小，那么落入每个区间的样本数量会比较少，其估计的区间密度也具有很大的随机性。如果 Δ 太大，其估计的密度函数变得十分平滑，很难反映出真实的数据分布。图9.4给出了直方图密度估计的例子，其中蓝线表示真实的密度函数，红色的柱状图为直方图方法估计的密度。



(a) 10 个区间（bin） (b) 30 个区间（bin）

图 9.4 直方图密度估计

直方图通常用来处理低维变量，可以非常快速地对数据的分布进行可视化，但其缺点是很难扩展到高维变量。假设一个 D 维的随机向量，如果每一维都划分为 M 个区间，那么整个空间的区间数量为 M^D 个。直方图方法需要的样本数量会随着维度 D 的增加而指数增长，从而导致维度灾难（Curse of Dimensionality）问题。

9.2.2.2 核方法

核密度估计（Kernel Density Estimation），也叫 Parzen 窗方法，是一种直方图方法的改进。

假设 $\mathcal{R}$ 为 D 维空间中的一个以点 $\mathbf{x}$ 为中心的“超立方体”，并定义核函数

$$\phi\Big(\frac{\mathbf{z}-\mathbf{x}}{h}\Big)=\begin{cases}1 & \text{当}|z_d-x_d|<\frac{h}{2}, 1\leq d\leq D\\ 0 & \text{否则}\end{cases} \tag{9.46}$$

来表示一个样本 $\mathbf{z}$ 是否落入该超立方体中，其中 h 为超立方体的边长，也称为核函数的宽度。

给定 N 个训练样本 $\mathcal{D}=\{\mathbf{x}^{(n)}\}_{n=1}^N$，落入区域 $\mathcal{R}$ 的样本数量 K 为

$$K=\sum_{n=1}^{N}\phi\Big(\frac{\mathbf{x}^{(n)}-\mathbf{x}}{h}\Big), \tag{9.47}$$

则点$\mathbf{x}$的密度估计为

$$p(\mathbf{x}) = \frac{1}{Nh^D}\sum_{n=1}^{N}\phi\Big(\frac{\mathbf{x}^{(n)}-\mathbf{x}}{h}\Big), \tag{9.48}$$

其中h^D表示区域$\mathcal{R}$的体积。

除了超立方体的核函数之外，我们还可以选择更加平滑的核函数，比如高斯核函数，

$$\phi\Big(\frac{\mathbf{z}-\mathbf{x}}{h}\Big) = \frac{1}{(2\pi)^{1/2}h}\exp\Big(-\frac{\|\mathbf{z}-\mathbf{x}\|^2}{2h^2}\Big), \tag{9.49}$$

其中h^2可以看做是高斯核函数的方差。这样点$\mathbf{x}$的密度估计为

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi)^{1/2}h}\exp\Big(-\frac{\|\mathbf{z}-\mathbf{x}\|^2}{2h^2}\Big). \tag{9.50}$$

9.2.2.3 K 近邻方法

核密度估计方法中的核宽度是固定的，因此同一个宽度可能对高密度的区域过大，而对低密度区域过小。一种更灵活的方式是设置一种可变宽度的区域，并使得落入每个区域中样本数量为固定的K。要估计点$\mathbf{x}$的密度，首先找到一个以$\mathbf{x}$为中心的球体，使得落入球体的样本数量为K，然后根据公式(9.44)，就可以计算出点$\mathbf{x}$的密度。因为落入球体的样本也是离$\mathbf{x}$最近的K个样本，所以这种方法称为*K近邻*（K-Nearest Neighbor）方法。

K近邻方法并不是一个严格的密度函数估计方法，参见习题9-5。

同样，K近邻方法中K的选择也十分关键。如果K太小，无法有效地估计密度函数，而K太大也会使得局部的密度不准确，并且增加计算开销。

K近邻方法也经常用于分类问题，称为*K近邻分类器*。当$K=1$也称为*最近邻分类器*。最近邻分类器的一个性质是，当$N\to\infty$时，其分类错误率不超过最优分类器错误率的两倍 [Cover and Hart, 1967]。

参见习题9-6。

9.3 总结和深入阅读

关于非参数密度估计的方法一般性介绍可以参考 [Duda et al., 2001] 和 [Bishop, 2007]，理论性介绍可以参考 [Devroye and Gyorfi, 1985]。

习题

习题 9-1 分析主成分分析为什么具有数据降噪能力？

习题 9-2 证明对于 N 个样本（样本维数 $d > N$）组成的数据集，主成分分析的有效投影子空间不超过 $N-1$ 维。

习题 9-3 对于一个两类分类问题，试举例分析什么样的数据分布会使得主成分分析得到的特征反而会使得分类性能下降。

习题 9-4 若数据矩阵 $X' = X - \bar{X}$，则对 X' 奇异值分解 $X' = U\Sigma V$，则 U 为主成分分析的投影矩阵。

习题 9-5 举例说明，K 近邻方法估计的密度函数不是严格的概率密度函数，其在整个空间上的积分不等于 1。

习题 9-6 对于一个 C 类的分类问题，使用 K 近邻方法估计每个类 c $(1 \leq c \leq C)$ 的密度函数 $p(\mathbf{x}|c)$，并使用贝叶斯公式计算每个类的后验概率 $p(c|\mathbf{x})$。

参考文献

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007. ISBN 9780387310732.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13 (1):21–27, 1967.

Luc Devroye and Laszlo Gyorfi. *Nonparametric Density Estimation: The L_1 View*. Wiley Series in Probability and Statistics. Wiley，New York, 1985.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification, 2nd Edition*. Wiley, 2001. ISBN 9780471056690.

Geoffrey E Hinton, Terrence Joseph Sejnowski, and Tomaso A Poggio. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

Bruno A Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.