

《神经网络与深度学习》



无监督学习

<https://nndl.github.io/>

内容

▶ 无监督学习

▶ 无监督特征学习

- ▶ 主成分分析

- ▶ 稀疏编码

- ▶ 自编码器

- ▶ 稀疏自编码器

- ▶ 降噪自编码器

▶ 概率密度估计

- ▶ 参数密度估计

- ▶ 非参数密度估计

 - 核方法

 - K 近邻方法

无监督学习 (Unsupervised Learning)

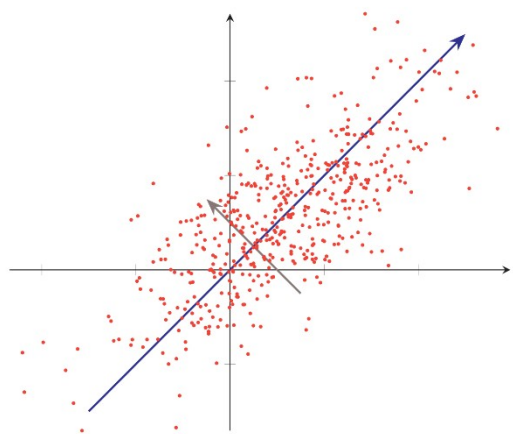
▶ 监督学习

- ▶ 建立映射关系 $f: x \rightarrow y$

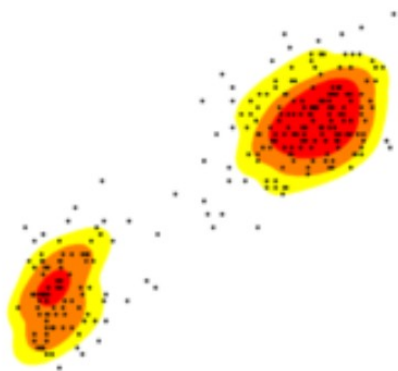
▶ 无监督学习

- ▶ 指从无标签的数据中学习出一些有用的模式。
- ▶ 聚类：建立映射关系 $f: x \rightarrow y$
 - ▶ 不借助于任何人工给出标签或者反馈等指导信息
- ▶ 特征学习
- ▶ 密度估计 $p(x)$

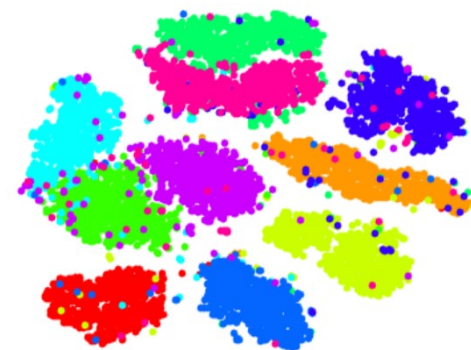
典型的无监督学习问题



无监督特征学习



密度估计



聚类

为什么要无监督学习？

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



无监督特征学习

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。

▶ 样本点 $\mathbf{x}^{(n)}$ 投影之后的表示为

$$z^{(n)} = \mathbf{w}^\top \mathbf{x}^{(n)}$$

▶ 所有样本投影后的方差为

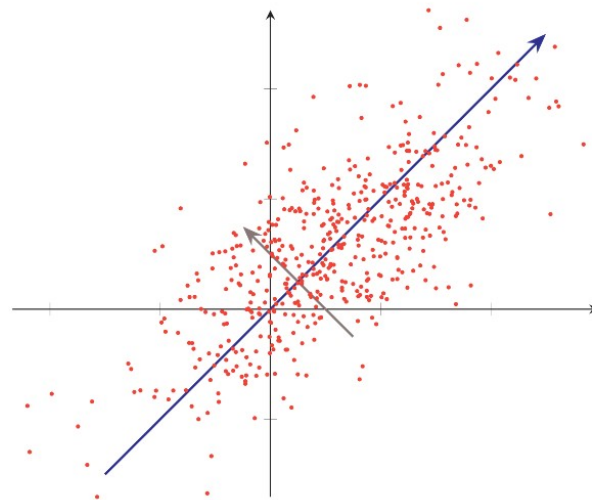
$$\begin{aligned}\sigma(\mathbf{X}; \mathbf{w}) &= \frac{1}{N} \sum_{n=1}^N (\mathbf{w}^\top \mathbf{x}^{(n)} - \mathbf{w}^\top \bar{\mathbf{x}})^2 \\ &= \frac{1}{N} (\mathbf{w}^\top \mathbf{X} - \mathbf{w}^\top \bar{\mathbf{X}})(\mathbf{w}^\top \mathbf{X} - \mathbf{w}^\top \bar{\mathbf{X}})^\top \\ &= \mathbf{w}^\top \Sigma \mathbf{w},\end{aligned}$$

▶ 目标函数

$$\max_{\mathbf{w}} \mathbf{w}^\top \Sigma \mathbf{w} + \lambda(1 - \mathbf{w}^\top \mathbf{w})$$

▶ 对目标函数求导并令导数等于 0，可得

$$\Sigma \mathbf{w} = \lambda \mathbf{w}$$



(线性) 编码

- 给定一组基向量 $A = [\mathbf{a}_1, \dots, \mathbf{a}_M]$, 将输入样本 $\mathbf{x}$ 表示为这些基向量的线性组合

$$\mathbf{x} = \sum_{m=1}^M z_m \mathbf{a}_m$$
$$= \mathbf{A}\mathbf{z},$$

字典

(dictionary)

编码

(encoding)

$$\begin{bmatrix} \text{green} \\ \text{blue} \\ \text{yellow} \\ \text{orange} \\ \text{blue} \\ \text{green} \\ \text{light blue} \\ \text{light blue} \end{bmatrix} = \begin{bmatrix} \text{grid of colored squares} \end{bmatrix} \begin{bmatrix} ? \\ ? \\ \cdot \\ \cdot \\ \cdot \\ ? \\ ? \end{bmatrix}$$

$\mathbf{x} \qquad \qquad \mathbf{A} \qquad \qquad \mathbf{z}$

完备性

数学小知识 | 完备性

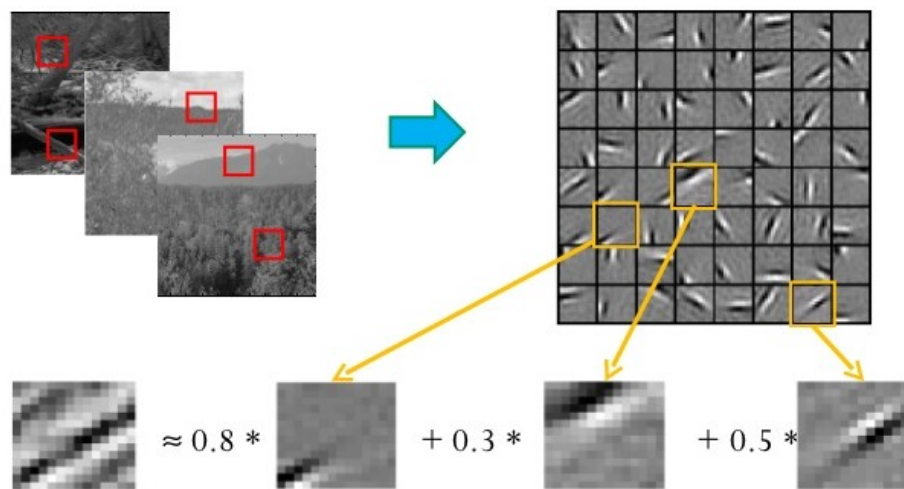
如果 M 个基向量刚好可以支撑 M 维的欧氏空间, 则这 M 个基向量是完备的. 如果 M 个基向量可以支撑 D 维的欧氏空间, 并且 $M > D$, 则这 M 个基向量是过完备的 (overcomplete)、冗余的.

“过完备”基向量是指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不具备独立、正交等性质.

► 稀疏编码

► 找到一组“过完备”的基向量 (即 $M > D$) 来进行编码。

Sparse coding illustration



$[a_1, \dots, a_{64}] = [0, 0, \dots, 0, \mathbf{0.8}, 0, \dots, 0, \mathbf{0.3}, 0, \dots, 0, \mathbf{0.5}, 0]$
(feature representation)

Slide credit: Andrew Ng

Compact & easily interpretable

稀疏编码 (Sparse Coding)

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

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

► $\rho(\cdot)$ 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。

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

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

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

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

训练过程

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

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

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

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

稀疏编码的优点

► 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。

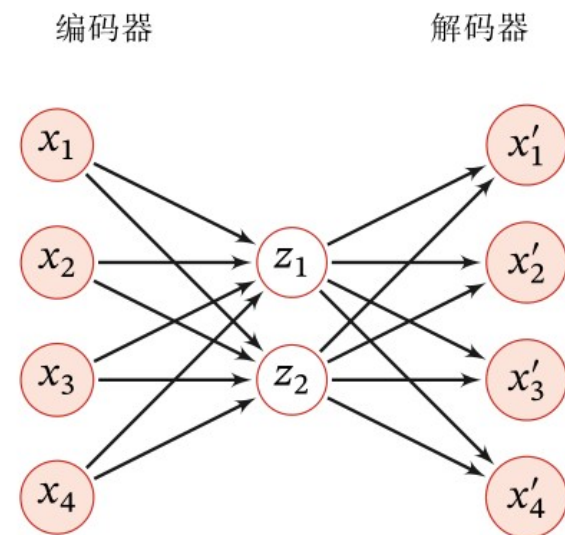
自编码器 (Auto-Encoder)

- ▶ 编码器 (Encoder $f: \mathbb{R}^D \rightarrow \mathbb{R}^M$
- ▶ 解码器 (Decoder $g: \mathbb{R}^M \rightarrow \mathbb{R}^D$

- ▶ 目标函数：重构错误

$$\begin{aligned}\mathcal{L} &= \sum_{n=1}^N \|\mathbf{x}^{(n)} - g(f(\mathbf{x}^{(n)}))\|^2 \\ &= \sum_{n=1}^N \|\mathbf{x}^{(n)} - f \circ g(\mathbf{x}^{(n)})\|^2.\end{aligned}$$

- ▶ 两层网络结构的自编码器



稀疏自编码器

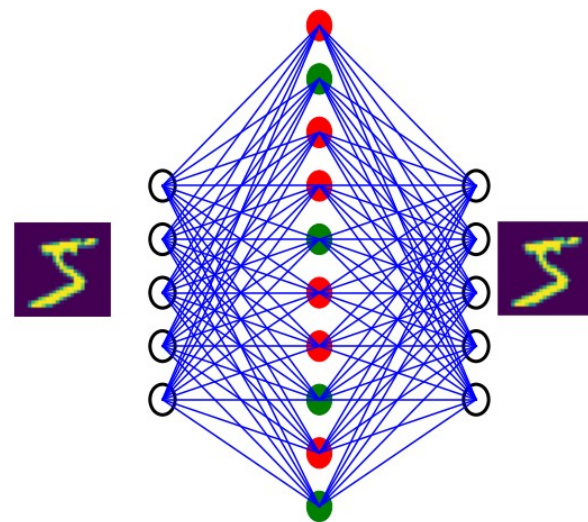
▶ 通过给自编码器中隐藏层单元 z 加上稀疏性限制，自编码器可以学习到数据中一些有用的结构。

▶ 目标函数

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

▶ $\mathbf{W}$ 表示自编码器中的参数

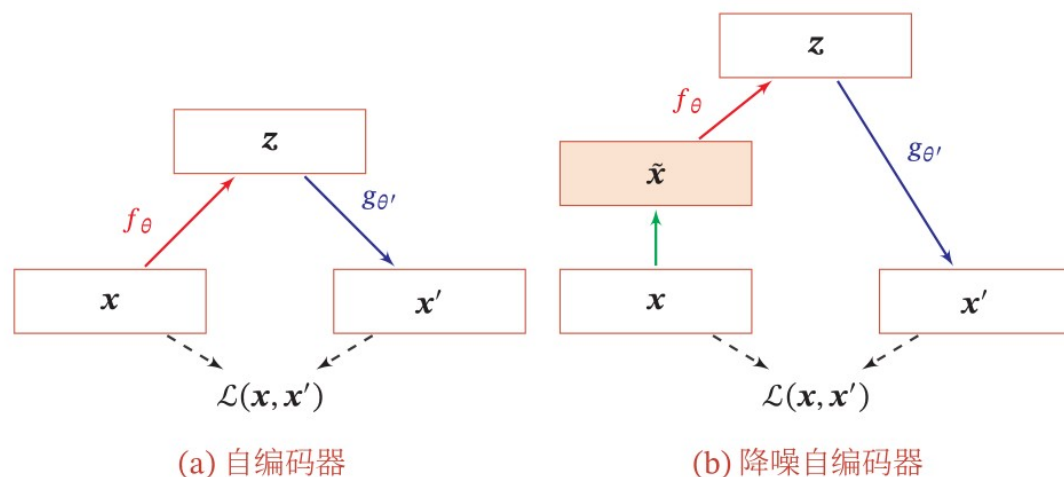
▶ 和稀疏编码一样，稀疏自编码器的优点是有很高的可解释性，并同时进行了隐式的特征选择。



降噪自编码器

通过引入噪声来增加编码鲁棒性的自编码器

- 对于一个向量 x ，我们首先根据一个比例 μ 随机将 x 的一些维度的值设置为0，得到一个被损坏的向量 $\tilde{x}$ 。
- 然后将被损坏的向量 $\tilde{x}$ 输入给自编码器得到编码 z ，并重构出原始的 x 。





概率密度估计

概率密度估计

▶ 参数密度估计 (Parametric Density Estimation)

- ▶ 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。
- ▶ 估计方法：最大似然估计

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

▶ 非参数密度估计 (Nonparametric Density Estimation)

- ▶ 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。

参数密度估计

► 正态分布

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

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right),$$

其中 $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$ 分别为正态分布的均值和方差.

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

$$\log p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{N}{2} \log\left((2\pi)^D |\boldsymbol{\Sigma}|\right) - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}^{(n)} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}^{(n)} - \boldsymbol{\mu}).$$

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

$$\begin{aligned}\boldsymbol{\mu}^{ML} &= \frac{1}{N} \sum_{n=1}^N \mathbf{x}^{(n)}, \\ \boldsymbol{\Sigma}^{ML} &= \frac{1}{N} \sum_{n=1}^N (\mathbf{x}^{(n)} - \boldsymbol{\mu}^{ML})(\mathbf{x}^{(n)} - \boldsymbol{\mu}^{ML})^\top.\end{aligned}$$

参数密度估计

► 多项分布

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

$$\log p(\mathcal{D}|\boldsymbol{\mu}) = \sum_{n=1}^N \sum_{k=1}^K x_k^{(n)} \log(\mu_k). \quad (9.34)$$

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.

$$\max_{\boldsymbol{\mu}, \lambda} \sum_{n=1}^N \sum_{k=1}^K x_k^{(n)} \log(\mu_k) + \lambda \left(\sum_{k=1}^K \mu_k - 1 \right). \quad (9.35)$$

分别求上式关于 μ_k, λ 的偏导数, 并令其等于 0. 可得,

$$\mu_k^{ML} = \frac{m_k}{N}, \quad 1 \leq k \leq K \quad (9.36)$$

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.

参数密度估计一般存在以下问题

▶ 模型选择问题

- ▶ 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。

▶ 不可观测变量问题

- ▶ 即我们用来训练的样本只包含部分的可观测变量，还有一些非常关键的变量是无法观测的，这导致我们很难准确估计数据的真实分布。

▶ 维度灾难问题

- ▶ 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。

非参数密度估计

- ▶ 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

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

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

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

- ▶ 当 N 非常大时，我们可以近似认为

$$P \approx \frac{K}{N}$$

- ▶ 假设区域 $\mathcal{R}$ 足够小，其内部的概率密度是相同的，则有

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

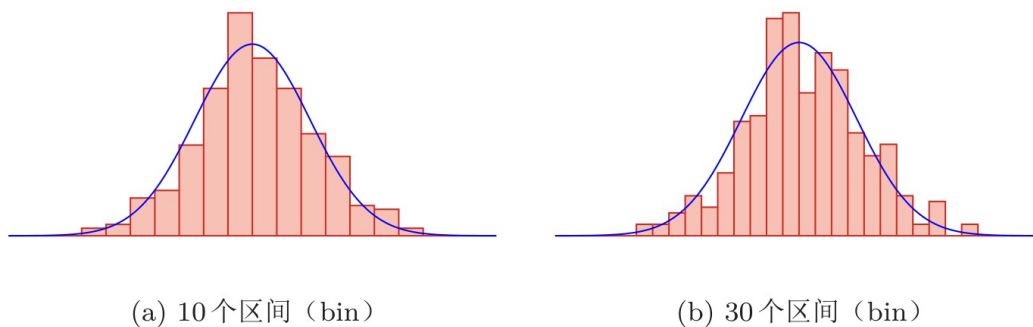
- ▶ 结合上述两个公式，得到

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

直方图方法（Histogram Method）

- 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。

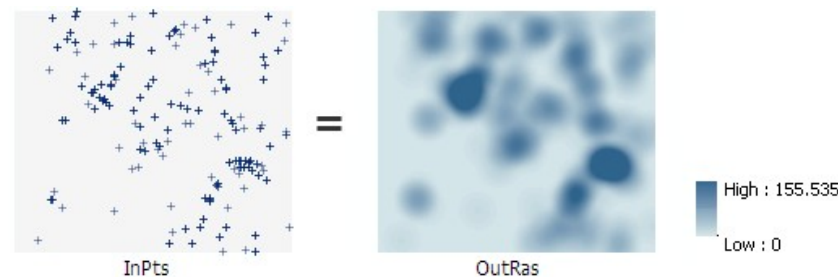
以一维随机变量为例，首先将其取值范围分成 M 个连续的、不重叠的区间（bin），每个区间的宽度为 Δ_m 。给定 N 个训练样本 $\mathcal{D} = \{x^{(n)}\}_{n=1}^N$ ，我们统计这些样本落入每个区间的数量 K_m ，然后将它们归一化为密度函数。



核密度估计 (Kernel Density Estimation)

►核密度估计是一种直方图方法的改进。

►也叫Parzen窗方法



►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”，并定义核函数来表示一个样本 $\mathbf{z}$ 是否落入该超立方体中

$$\phi\left(\frac{\mathbf{z} - \mathbf{x}}{h}\right) = \begin{cases} 1 & \text{if } |z_i - x_i| < \frac{H}{2}, 1 \leq i \leq D \\ 0 & \text{else} \end{cases}$$

►点 $\mathbf{x}$ 的密度估计为
$$p(\mathbf{x}) = \frac{K}{NH^D} = \frac{1}{NH^D} \sum_{n=1}^N \phi\left(\frac{\mathbf{x}^{(n)} - \mathbf{x}}{H}\right)$$

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 x 的密度，首先找到一个以 x 为中心的球体，使得落入球体的样本数量为 K ，就可以计算出点 x 的密度。

思考题

▶ 估计以下三种密度函数的区别: $p(x)$, $p(y|x)$ and $p(x|y)$?

- ▶ 生成模型 Generative model
- ▶ 判别式模型 Discriminative model

▶ 最大似然函数的本质是什么?

- ▶ 真实数据分布 $p_r(x)$
- ▶ 模型数据分布 $p_\theta(x)$

非参数密度估计

- ▶ 非参数密度估计 (直方图方法除外) 需要保留整个训练集。
- ▶ 而参数密度估计不需要保留整个训练集，因此在存储和计算上更加高效。

知识点

▶ 无监督特征学习

- ▶ 稀疏编码
- ▶ 自编码器
- ▶ 稀疏自编码器
- ▶ 降噪自编码器

▶ 概率密度估计

- ▶ 参数密度估计
- ▶ 非参数密度估计
 - ▶ 核方法
 - ▶ K 近邻方法

谢 谢