第 3 章

CHAPTER 3

概率密度函数的估计

贝叶斯决策前提条件是已知各类的先验概率和类条件概率，但实际中所得到的只是样本集，需要由样本集计算所需的概率密度函数，即需要进行概率密度函数的估计。先验概率的估计可以由各类样本在总体样本集中所占的比例进行估计。本章主要介绍类条件概率密度估计的方法，即已知 $\omega_{j,j=1\sim c}$ 类中的若干样本 $\boldsymbol{x}_{i,i=1\sim N}$，采用某种规则估计出样本所属类的概率密度函数 $p(\boldsymbol{x}|\omega_j)$ 的方法。

3.1 基本概念

1. 概率密度函数的估计方法

概率密度函数的估计方法可以分为两大类，即参数估计(Parametric Estimation)和非参数估计(Nonparametric Estimation)。

如果已知类条件总体概率密度函数形式，未知其中部分或全部参数，用样本估计这些参数，称为参数估计。如例 2-12 中，已知样本集服从正态分布，未知正态分布的均值向量和协方差矩阵，根据样本集确定 $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$，即参数估计。本章学习最大似然估计、最大后验估计和贝叶斯估计方法。

非参数估计指的是概率密度函数形式也未知，需要求函数本身。本章主要学习直方图方法、Parzen 窗法及 k_N 近邻密度估计法。

2. 参数估计的基本概念

参数估计是统计推断的基本问题之一，回顾一下涉及的几个基本概念。

(1) 统计量。假如概率密度函数的形式已知，但表征函数的参数 θ 未知，则可将 θ 的估计值构造成样本的某种函数，这种函数称为统计量。

(2) 参数空间：参数 θ 的取值范围称为参数空间，表示为 Θ。

(3) 点估计、估计量和估计值：构造一个样本的函数 $\hat{\theta}$，将其观察值作为未知参数 θ 的估计，称为点估计；$\hat{\theta}$ 称为 θ 的估计量；由具体样本 $(\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N)$ 作为自变量计算出来的 $\hat{\theta}(\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N)$ 值称为 θ 的估计值。

(4) 区间估计：通过从总体中抽取的样本，根据一定的正确度与精确度的要求，构造出适当的区间，作为未知参数的真值所在范围的估计。这个区间称为置信区间。置信区间是一种常用的区间估计方法，是以统计量的置信上限和下限为上下界构成的区间。

本章要学习的参数估计属于点估计问题，点估计有不同的方法，同一参数用不同的估计方法求出的估计量可能不相同，需要合适的标准评估估计量。常用的标准有无偏性、有效性、一致性等，此处不再赘述，可以参看数理统计类资料。

3.2 参数估计

本节主要学习参数估计方法，包括最大似然估计（Maximum Likelihood Estimation，MLE）、最大后验（Maximum a Posteriori，MAP）估计、贝叶斯估计（Bayesian Estimation）方法。

3.2.1 最大似然估计

最大似然估计是通过定义似然函数，对似然函数求最大值，进而确定未知参数的估计值的方法。

1. 前提假设

最大似然估计方法的使用，需要满足下列基本假设：

（1）待估计参数 θ 是确定（非随机）而未知的量。

（2）样本集分成 c 类 $\mathcal{X}_1,\mathcal{X}_2,\cdots,\mathcal{X}_c,\mathcal{X}_j$ 的样本是从概率密度为 $p(\boldsymbol{x}|\omega_j)$ 的总体中独立抽取出来的，即满足独立同分布条件。

（3）类条件概率密度 $p(\boldsymbol{x}|\omega_j)$ 具有某种确定的函数形式，其参数 θ_j 未知，可以把 $p(\boldsymbol{x}|\omega_j)$ 表达为 $p(\boldsymbol{x}|\omega_j,\theta_j)$ 或 $p(\boldsymbol{x}|\theta_j)$，表示与 θ_j 有关。

（4）不同类别的参数在函数上独立，可以分别对每一类单独处理。

在这些假设的前提下，参数估计的问题描述为：分别从概率密度为 $p(\boldsymbol{x}|\omega_j)$ 的总体中独立抽取，构成 c 个样本集 $\mathcal{X}_1,\mathcal{X}_2,\cdots,\mathcal{X}_c$，利用 $\mathcal{X}_j$ 的样本估计 $p(\boldsymbol{x}|\omega_j)$ 的参数 θ_j。

2. 似然函数

样本集 $\mathcal{X}$ 包含 N 个样本，即 $\mathcal{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$，各样本相互独立，获得样本集的概率 $l(\theta)$ 即各个样本的联合概率，定义 $l(\theta)$ 为样本集 $\mathcal{X}$ 的似然函数（Likelihood Function），如式（3-1）所示。

$$l(\theta)=p(\mathcal{X}\mid\theta)=p\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\mid\theta\}=\prod_{i=1}^{N}p(\boldsymbol{x}_i\mid\theta) \tag{3-1}$$

由于各样本为相互独立抽取，所以 N 个随机变量 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N$ 的联合概率密度 $p\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N|\theta\}$ 等于各自概率密度的乘积。

独立抽取时，样本集中的样本最有可能来源于概率密度最大的情况。似然函数定义为联合概率密度，样本独立抽取时为概率密度的乘积，所以，已知一组样本，最有可能来自似然函数最大所对应的情况。因此，可以利用似然函数作参数估计。令 $l(\theta)$ 为样本集的似然函数，如果 $\hat{\theta}$ 是参数空间中使 $l(\theta)$ 最大化的值，那么 $\hat{\theta}$ 为 θ 的最大似然估计量，如式（3-2）所示。

$$\hat{\theta}=\arg\ \max l(\theta) \tag{3-2}$$

至此，估计问题转换为求极值的问题。

为便于分析，可以采用对数函数将连乘转换为累加，定义对数似然函数为

$$h(\theta)=\ln l(\theta)=\sum_{i=1}^{N}\ln p(\boldsymbol{x}_i\mid\theta) \tag{3-3}$$

由于对数函数的单调性，使对数似然函数 $h(\theta)$ 最大的 $\hat{\theta}$ 也必然使得似然函数 $l(\theta)$ 最大。

3. 求解估计量

通过定义似然函数，将估计问题转换为求极值的问题。根据未知量 θ 的情况，最大似然估计量的求解可以分为如下几种情况。

1）未知参数为一元情况

只有一个未知参数，最大似然估计量是式(3-4)或式(3-5)的解。

$$\frac{\mathrm{d}l(\theta)}{\mathrm{d}\theta}=0 \tag{3-4}$$

或

$$\frac{\mathrm{d}h(\theta)}{\mathrm{d}\theta}=0 \tag{3-5}$$

【例 3-1】 设样本服从指数分布，密度函数为 $p(x)=\begin{cases}\lambda\mathrm{e}^{-\lambda x}, & x\geqslant 0\\ 0, & x<0\end{cases}$，求 λ 的最大似然估计量。

解： 设样本集 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$，定义似然函数为

$$l(\theta)=\prod_{i=1}^{N}\lambda\mathrm{e}^{-\lambda x_i}$$

λ 是未知参数。定义对数似然函数为

$$h(\theta)=\ln l(\theta)=\sum_{i=1}^{N}\ln(\lambda\mathrm{e}^{-\lambda x_i})=\sum_{i=1}^{N}(\ln\lambda-\lambda x_i)=N\ln\lambda-\sum_{i=1}^{N}\lambda x_i$$

因为

$$\theta=\lambda$$

所以

$$\frac{\mathrm{d}h}{\mathrm{d}\lambda}=\frac{N}{\lambda}-\sum_{i=1}^{N}x_i=0$$

所以

$$\hat{\lambda}=\frac{N}{\sum_{i=1}^{N}x_i}=\frac{1}{\bar{x}}$$

λ 的最大似然估计量为 $1/\bar{x}$，即为样本均值的倒数。

2）未知参数为多元情况

$\boldsymbol{\theta}=\begin{bmatrix}\theta_1 & \theta_2 & \cdots & \theta_s\end{bmatrix}^{\mathrm{T}}$ 是由多个未知参数组成的向量，似然函数对 $\boldsymbol{\theta}$ 各分量分别求导，组成方程组，求最值，即解

$$\frac{\partial l(\boldsymbol{\theta})}{\partial\theta_1}=0,\frac{\partial l(\boldsymbol{\theta})}{\partial\theta_2}=0,\cdots,\frac{\partial l(\boldsymbol{\theta})}{\partial\theta_s}=0 \tag{3-6}$$

或

$$\frac{\partial h(\boldsymbol{\theta})}{\partial \theta_1}=0,\frac{\partial h(\boldsymbol{\theta})}{\partial \theta_2}=0,\cdots,\frac{\partial h(\boldsymbol{\theta})}{\partial \theta_s}=0 \tag{3-7}$$

【例 3-2】 设 x 服从正态分布 $N(\mu,\sigma^2)$，其中参数 μ、σ^2 未知，求它们的最大似然估计量。

解：设样本集 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$，定义似然函数为

$$l(\boldsymbol{\theta})=\prod_{i=1}^{N}p(x_i\mid\boldsymbol{\theta})$$

定义对数似然函数并简化为

$$\begin{aligned}h(\boldsymbol{\theta})&=\ln l(\boldsymbol{\theta})=\sum_{i=1}^{N}\ln p(x_i\mid\boldsymbol{\theta})=\sum_{i=1}^{N}\ln\left\{\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x_i-\mu)^2}{2\sigma^2}\right]\right\}\\&=\sum_{i=1}^{N}\left[-\frac{1}{2}\ln 2\pi-\frac{1}{2}\ln\sigma^2-\frac{(x_i-\mu)^2}{2\sigma^2}\right]\\&=-\frac{N}{2}\ln 2\pi-\frac{N}{2}\ln\sigma^2-\sum_{i=1}^{N}\frac{(x_i-\mu)^2}{2\sigma^2}\end{aligned}$$

因为

$$\boldsymbol{\theta}=[\mu,\sigma^2]^{\mathrm{T}}$$

所以

$$\begin{cases}\dfrac{\partial h}{\partial\mu}=-\displaystyle\sum_{i=1}^{N}-\frac{2(x_i-\mu)}{2\sigma^2}=\sum_{i=1}^{N}\frac{x_i-\mu}{\sigma^2}=0\\[2ex]\dfrac{\partial h}{\partial\sigma^2}=-\dfrac{N}{2\sigma^2}+\dfrac{1}{2(\sigma^2)^2}\displaystyle\sum_{i=1}^{N}(x_i-\mu)^2=0\end{cases}$$

$$\begin{cases}\hat{\mu}=\dfrac{1}{N}\displaystyle\sum_{i=1}^{N}x_i=\bar{x}\\[2ex]\hat{\sigma}^2=\dfrac{1}{N}\displaystyle\sum_{i=1}^{N}(x_i-\bar{x})^2\end{cases}$$

$\hat{\mu}$ 和 $\hat{\sigma}^2$ 是 μ、σ^2 的最大似然估计量。

3）特殊情况

如果导数方程无解，可以根据具体数据来定。

【例 3-3】 设 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$ 是来自 $p(\mathcal{X}|\theta)$ 的随机样本，当 $0\leqslant x\leqslant\theta$ 时，$p(x|\theta)=1/\theta$；否则为 0。证明 θ 的最大似然估计量是 $\max\limits_i x_i$。

解：定义似然函数为

$$l(\theta)=\prod_{i=1}^{N}p(x_i\mid\theta)=\prod_{i=1}^{N}\frac{1}{\theta}$$

对数似然函数为

$$h(\theta)=\ln l(\theta)=-N\ln\theta$$

令导数为零得

$$\frac{\mathrm{d}h}{\mathrm{d}\theta}=-N\cdot\frac{1}{\theta}=0$$

方程的解为无穷大，但实际问题中 $\theta\neq\infty$。已知有 N 个随机样本，且 $0\leqslant x\leqslant\theta$ 时，$p(x|\theta)=$

$1/\theta$，所以，取 $\hat{\theta}$ 为样本中的最大值，即 θ 的最大似然估计量是 $\max_i x_i$。

三个例题分别实现了对指数分布、正态分布和均匀分布参数的最大似然估计，其他不同分布参数估计的过程类似，即为：①生成数据集；②定义似然函数或对数似然函数；③对函数求最大值得参数的最大似然估计量。

4. 仿真实现

SciPy 库的 stats 模块中，基类 rv_discrete 用于构造离散随机变量的特定分布，派生出 bernoulli、binom、poisson、randint 等子类，每个子类具有大量方法，其中，rvs 方法用于生成给定分布的随机变量，pmf、logpmf、cdf 方法分别计算给定分布在给定点的概率质量函数值、概率质量函数对数值、累积分布函数值；基类 rv_continuous 用于构造连续随机变量的特定分布，派生出 beta、exponnorm、lognorm、norm、uniform 等子类，每个子类具有 rvs、pdf、logpdf、cdf、fit 等方法，fit 方法采用最大似然估计方法估计数据分布的相关参数（也可以采用矩估计的方法）。

【例 3-4】 利用 norm 类方法生成正态分布数据，并采用 fit 方法对参数进行估计。

程序如下：

```
import numpy as np
from scipy.stats import norm
x = norm.rvs(0, 1, size=1000)                              #生成单变量的正态分布数据
mu1, sigma1 = np.mean(x), np.std(x)    #采用 NumPy 库函数估计参数,标准差的估计采用了 MLE 方法
mu2, sigma2 = norm.fit(x)                                  #采用 SciPy 库函数估计参数
print("正态分布均值为:%.4f" % mu2)
print("正态分布标准差为:%.4f" % sigma2)
```

运行程序，将输出以下内容：

```
正态分布均值为: -0.0316
正态分布标准差为: 0.9764
```

【例 3-5】 scikit-learn 中的 diabetes 数据集中有 442 名病人的各 10 个体检数据，包括年龄、性别、BMI 体重指数、血压以及 6 个血检指标，读取该数据集，对 BMI 体重指数进行正态分布拟合，并绘制概率密度函数曲线。

程序如下：

```
from scipy.stats import norm
from sklearn.datasets import load_diabetes
from matplotlib import pyplot as plt
db_data = load_diabetes()                                  #导入数据集
bmi_order = db_data.feature_names.index('bmi')             #查看 BMI 数据是第几项
X = db_data.data[:, bmi_order]                             #获取 BMI 数据
mu, sigma = norm.fit(X)                                    #采用 SciPy 库函数估计参数
p = norm.pdf(X, mu, sigma)                                 #计算概率密度函数值
print("均值为:%.4f" % mu, "标准差为:%.4f" % sigma)
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.plot(X, p, 'k+')                                       #绘制概率密度函数曲线
plt.xlabel('x')
plt.ylabel('p(x)')
plt.show()
```

运行程序，生成 BMI 数据的正态概率密度函数如图 3-1 所示，并输出以下内容：

均值为：－0.0000，标准差为：0.0476

diabetes 数据集中的每项数据都进行过标准化，每列数据均值为 0 且平方和为 1，估计结果和实际情况一致。

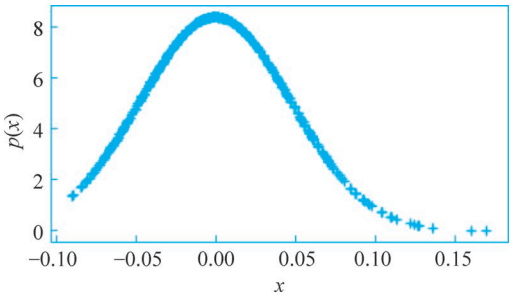


图 3-1 BMI 数据对应的正态概率密度函数曲线

3.2.2 最大后验估计

在最大似然估计中，认为参数 θ 是确定的量，假如把 θ 看作随机变量，在估计中，需要考虑 θ 本身服从的分布。因此，利用贝叶斯公式计算 θ 的后验概率，最大后验概率对应的参数值为参数的估计值，这种方法即是最大后验估计。

1. 基本原理

第 5 集
微课视频

设参数 θ 是先验概率为 $P(\theta)$的随机变量，其取值和样本集有关。设样本集为 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$，各样本相互独立，使 $P(\theta|\mathcal{X})$取最大值的 $\hat{\theta}$ 为 θ 的最大后验估计。通过贝叶斯公式进行 $P(\theta|\mathcal{X})$的计算，如式(3-8)所示。

$$P(\theta \mid \mathcal{X})=\frac{P(\theta)p(\mathcal{X} \mid \theta)}{p(\mathcal{X})}=\frac{P(\theta)p(\mathcal{X} \mid \theta)}{\int P(\theta)p(\mathcal{X} \mid \theta)\mathrm{d}\theta} \tag{3-8}$$

式中，$p(\mathcal{X}|\theta)$是样本集 $\mathcal{X}$ 的似然函数，如式(3-1)所示。

由于 $p(\mathcal{X})$相对于 θ 独立，最大后验估计即是求解如下问题：

$$\arg\max_{\theta} P(\theta)l(\theta) \tag{3-9}$$

或

$$\arg\max_{\theta} [\ln P(\theta)+\ln l(\theta)] \tag{3-10}$$

假设 θ 本身服从均匀分布，即对于所有的 θ，$P(\theta)$是常量，比较式(3-9)和式(3-2)，两者一致，即最大后验估计和最大似然估计结果一致；一般情况下，两种估计会得到不同的结果。

【例 3-6】 设 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$是来自总体分布为 $N(\mu,\sigma^2)$的样本集，已知 μ 服从 $N(\mu_0,\sigma_0^2)$分布，假设 σ^2 已知，用最大后验估计的方法求 μ 的估计量 $\hat{\mu}$。

解： 确定 μ 的先验分布 $P(\mu)$。已知 μ 服从 $N(\mu_0,\sigma_0^2)$分布，得

$$P(\mu)=\frac{1}{\sqrt{2\pi}\sigma_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right]$$

求样本联合分布 $p(\mathcal{X}|\mu)$，即似然函数 $l(\mu)$。样本集 $\mathcal{X}$ 总体分布为 $N(\mu,\sigma^2)$，即

$p(x_i|\mu)\sim N(\mu,\sigma^2)$。

所以

$$l(\mu)=\prod_{i=1}^{N}p(x_i|\mu)=\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2\right]$$

定义准则函数 $J(\mu)$ 为

$$\begin{aligned}J(\mu)&=\ln P(\mu)+\ln l(\mu)\\&=-\ln\sqrt{2\pi}\sigma_0-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2-N\ln\sqrt{2\pi}\sigma-\sum_{i=1}^{N}\frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2\end{aligned}$$

求导数并令其等于零

$$\frac{\mathrm{d}J}{\mathrm{d}\mu}=-\frac{1}{\sigma_0}\left(\frac{\mu-\mu_0}{\sigma_0}\right)+\sum_{i=1}^{N}\frac{1}{\sigma}\left(\frac{x_i-\mu}{\sigma}\right)=0$$

简化为

$$\frac{\mu_0}{\sigma_0^2}+\frac{1}{\sigma^2}\sum_{i=1}^{N}x_i-\left(\frac{1}{\sigma_0^2}+\frac{N}{\sigma^2}\right)\mu=0$$

得 μ 的最大后验估计量为

$$\hat{\mu}=\frac{\sigma^2}{\sigma^2+\sigma_0^2N}\mu_0+\frac{\sigma_0^2}{\sigma^2+\sigma_0^2N}\sum_{i=1}^{N}x_i$$

2. 仿真实现

【例 3-7】 设 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$ 是来自总体分布为对数正态分布的样本集，对数正态分布的概率密度函数为 $p(x)=\frac{1}{\sigma x\sqrt{2\pi}}\exp\left[-\frac{(\ln x-\mu)^2}{2\sigma^2}\right]$，$(x>0)$。已知 μ 服从 $N(0,3)$ 分布，$\sigma=2$，用最大后验估计的方法求 μ 的估计量 $\hat{\mu}$。

分析： 设定均值 μ，标准差 $\sigma=2$，生成对数正态分布样本；设定 μ 的取值范围为 $[-5,5]$，计算准则 $J(\mu)=\ln P(\mu)+\ln l(\mu)$ 的取值，并取最大值对应的 μ 作为估计值。

仿真函数：

题目中 $\ln x\sim N(\mu,\sigma^2)$，$x$ 服从对数正态分布，x 的期望和方差分别为 $e^{\mu+\sigma^2/2}$ 和 $(e^{\sigma^2}-1)e^{2\mu+\sigma^2}$。SciPy 库 stats 模块的 lognorm 类封装了对数正态分布的相关方法，下面简要介绍主要的方法。

(1) pdf(x, s, loc=0, scale=1)：计算 x 处对数正态分布概率密度函数取值。s 是正态分布的标准差，loc 和 scale 是分布平移和缩放参数，lognorm.pdf(x, s, loc, scale)相当于 lognorm.pdf((x−loc)/scale, s)/scale。logpdf、cdf、logcdf 函数和 pdf 函数参数一致，分别用于计算 x 处的对数概率密度函数值、累积分布函数值和对数累积分布函数值。

(2) rvs(s, loc=0, scale=1, size=1, random_state=None)：生成服从对数正态分布的随机数。如果 $\ln x\sim N(\mu,\sigma^2)$，那么 x 服从对数正态分布，设置参数 s 为 σ、scale 为 $\exp(\mu)$。

(3) median(s, loc=0, scale=1)、mean(s, loc=0, scale=1)、var(s, loc=0, scale=1)、std(s, loc=0, scale=1)、stats(s, loc=0, scale=1, moments='mv')：分别计算对数正态分布的中值、均值、方差、标准差，以及同时计算均值、方差、偏度和峭度。

程序如下：

```
from scipy.stats import lognorm
import numpy as np
from matplotlib import pyplot as plt

#计算 lnl(μ)
def lg_dist(x, miu = 0, sigma = 0.1):
    z = (np.log(x) - miu) / sigma
    lg_pdf = np.sum( - np.log(sigma * x) - .5 * np.log(2 * np.pi) - .5 * (z ** 2))
    return lg_pdf

#计算 lnP(μ)
def prior_dist(x, miu = 0, sigma = 0.1):
    z = (x - miu) / sigma
    lp_pdf = - np.log(sigma) - .5 * np.log(2 * np.pi) - .5 * (z ** 2)
    return lp_pdf

#以下为设定 μ 先验分布参数、对数正态分布参数并生成对数正态分布样本 500 个
mu0, sigma0 = 0, 3
mu, sd = 2, 2
training = lognorm.rvs(s = sd, loc = 0, scale = np.exp(mu), size = 500, random_state = 2)
#以下是在 μ 的取值范围内计算准则 J(μ)的值,并找 J(μ)的最大值对应的 μ
number = 1000
mu_value = np.linspace( - 5, 5, number)
Poster = np.zeros([number, 1])
for i in range(number):
    l_pdf = lg_dist(training, mu_value[i], sd)     #当前均值下对数正态分布的对数似然函数
    Poster[i] = l_pdf + prior_dist(mu_value[i], mu0, sigma0) #计算 J(μ)
pos = np.argmax(Poster)
print("估计的均值为：%.2f" % mu_value[pos])
plt.rcParams['font.sans - serif'] = ['Times New Roman']
plt.plot(mu_value, Poster, 'k')                                    #绘制 J(μ)函数图形
plt.xlabel('μ')
plt.ylabel('J(μ)')
plt.show()
```

运行程序，绘制的 $J(\mu)$（准则）函数图形如图 3-2 所示，并输出：

```
估计的均值为：1.97
```

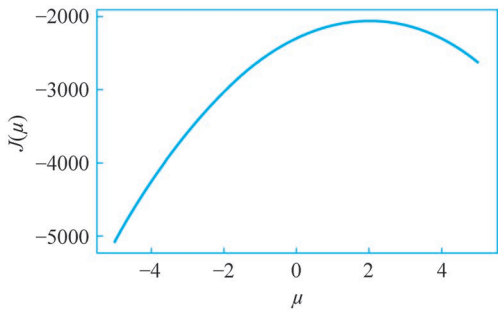


图 3-2　$J(\mu)$函数图形

3.2.3　贝叶斯估计

参数估计寻找最优的参数值，可以看作在连续的参数空间 Θ 对参数的取值进行决策的问题，因此，采用决策的思路进行估计。根据样本集 $\mathcal{X}$，找出一个估计量，用来估计 $\mathcal{X}$ 所属总体分布的某个真实参数，使带来的贝叶斯风险最小，这种方法称为贝叶斯估计。

1. 基本原理

样本独立抽取的样本集为 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$，用估计量 $\hat{\theta}$ 近似代替真实参数 θ，带来的损失为 $\lambda(\theta,\hat{\theta})$，参数 θ 的先验分布为 $P(\theta)$，定义 $R(\hat{\theta}|\mathcal{X})$ 为给定 $\mathcal{X}$ 条件下估计量 $\hat{\theta}$ 的条件风险。

$$R(\hat{\theta}\mid\mathcal{X})=\int_{\Theta}\lambda(\theta,\hat{\theta})P(\theta\mid\mathcal{X})\mathrm{d}\theta \tag{3-11}$$

如果 θ 的估计量 $\hat{\theta}^*$ 使条件风险 $R(\hat{\theta}|\mathcal{X})$ 最小，称 $\hat{\theta}^*$ 是关于 θ 的贝叶斯估计量，即

$$\hat{\theta}^*=\arg\min_{\hat{\theta}}R(\hat{\theta}\mid\mathcal{X}) \tag{3-12}$$

条件风险的计算需要定义损失函数 $\lambda(\theta,\hat{\theta})$，可以定义为不同的形式，最常用的是平方误差损失函数，即

$$\lambda(\theta,\hat{\theta})=(\theta-\hat{\theta})^2 \tag{3-13}$$

可以证明，如果采用平方误差损失函数，在给定样本集 $\mathcal{X}$ 条件下，θ 的贝叶斯估计量为

$$\hat{\theta}^*=E[\theta\mid\mathcal{X}]=\int_{\Theta}\theta P(\theta\mid\mathcal{X})\mathrm{d}\theta \tag{3-14}$$

整理贝叶斯估计的步骤如下。

(1) 确定 θ 的先验分布 $P(\theta)$。

(2) 根据式(3-1)，由样本集 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$ 求出样本联合分布 $p(\mathcal{X}|\theta)$。

(3) 利用贝叶斯公式(3-8)，求 θ 的后验分布 $P(\theta|\mathcal{X})$。

(4) 结合损失函数计算贝叶斯估计量。

【例 3-8】 设 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$ 是来自总体分布为 $N(\mu,\sigma^2)$ 的样本集，已知 μ 服从 $N(\mu_0,\sigma_0^2)$ 分布，采用平方误差损失函数，用贝叶斯估计的方法求 μ 的估计量 $\hat{\mu}$。

解： (1) 确定 μ 的先验分布 $P(\mu)$。

$$P(\mu)=\frac{1}{\sqrt{2\pi}\sigma_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right]$$

(2) 求样本联合分布 $p(\mathcal{X}|\mu)$。

$$p(\mathcal{X}\mid\mu)=\prod_{i=1}^{N}p(x_i\mid\mu)=\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2\right]$$

(3) 求 μ 的后验分布 $P(\mu|\mathcal{X})$。

$$P(\mu\mid\mathcal{X})=\frac{p(\mathcal{X}\mid\mu)\cdot P(\mu)}{\int p(\mathcal{X}\mid\mu)\cdot P(\mu)\mathrm{d}\mu}=\alpha\prod_{i=1}^{N}p(x_i\mid\mu)\cdot P(\mu)$$

$$=\alpha\cdot\frac{1}{\sqrt{2\pi}\sigma_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right]\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2\right]$$

$$=\alpha' \cdot \exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right]\prod_{i=1}^{N}\exp\left[-\frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2\right]$$

$$=\alpha' \cdot \exp\left\{-\frac{1}{2}\left[\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2+\sum_{i=1}^{N}\left(\frac{x_i-\mu}{\sigma}\right)^2\right]\right\}$$

$$=\alpha'' \cdot \exp\left\{-\frac{1}{2}\left[\left(\frac{N}{\sigma^2}+\frac{1}{\sigma_0^2}\right)\mu^2-2\left(\frac{1}{\sigma^2}\cdot\sum_{i=1}^{N}x_i+\frac{\mu_0}{\sigma_0^2}\right)\mu\right]\right\}$$

$P(\mu|\mathcal{X})$是μ的二次函数的指数函数，所以仍然是一个正态密度函数，把$P(\mu|\mathcal{X})$写成$N(\mu_N,\sigma_N^2)$的形式为

$$P(\mu\mid\mathcal{X})=\frac{1}{\sqrt{2\pi}\sigma_N}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_N}{\sigma_N}\right)^2\right]$$

应用待定系数法，令两式对应的系数相等，即

$$\begin{cases}\dfrac{1}{\sigma_N^2}=\dfrac{N}{\sigma^2}+\dfrac{1}{\sigma_0^2}\\[2ex]\dfrac{\mu_N}{\sigma_N^2}=\dfrac{N}{\sigma^2}\bar{x}+\dfrac{\mu_0}{\sigma_0^2}\end{cases}\rightarrow\begin{cases}\mu_N=\dfrac{N\sigma_0^2}{N\sigma_0^2+\sigma^2}\bar{x}+\dfrac{\sigma^2}{N\sigma_0^2+\sigma^2}\mu_0\\[2ex]\sigma_N^2=\dfrac{\sigma_0^2\sigma^2}{N\sigma_0^2+\sigma^2}\end{cases}$$

其中，

$$\bar{x}=\frac{1}{N}\sum_{k=1}^{N}x_k$$

(4) 求贝叶斯估计量。由式(3-14)知$\hat{\mu}=\int\mu P(\mu\mid\mathcal{X})\mathrm{d}\mu$，即给定样本集下的$\mu$的条件期望；而$P(\mu|\mathcal{X})\sim N(\mu_N,\sigma_N^2)$，所以，$\mu$的贝叶斯估计量$\hat{\mu}=\dfrac{N\sigma_0^2}{N\sigma_0^2+\sigma^2}\bar{x}+\dfrac{\sigma^2}{N\sigma_0^2+\sigma^2}\mu_0$。

2. 推论分析

需要注意，进行参数估计的根本目的是估计样本的概率密度函数$p(x|\mathcal{X})$，由于概率密度函数形式已知，才转换为估计概率密度函数参数的问题。因此，在贝叶斯估计中，在得到参数的后验概率后，可以不再去估计参数，直接根据$P(\theta|\mathcal{X})$得到样本的概率密度函数。

$$p(x\mid\mathcal{X})=\int_{\Theta}p(x\mid\theta)P(\theta\mid\mathcal{X})\mathrm{d}\theta \tag{3-15}$$

其中，$P(\theta|\mathcal{X})$如式(3-8)所示。

式(3-15)的含义是：要估计的概率密度$p(x|\mathcal{X})$是所有可能的参数取值下样本概率密度$p(x|\theta)$的加权平均，加权是在观测样本下估计出的随机变量θ的后验概率$P(\theta|\mathcal{X})$。

假设$P(\theta|\mathcal{X})$已知，由式(3-15)可知，$p(x|\mathcal{X})$是关于θ的$p(x|\theta)$的期望，有

$$p(x\mid\mathcal{X})=E_{\Theta}[p(x\mid\theta)] \tag{3-16}$$

如果采样点θ_m足够多，$m=1,2,\cdots,M$，可以用式(3-17)代替式(3-16)

$$p(x\mid\mathcal{X})\approx\frac{1}{M}\sum_{m=1}^{M}p(x\mid\theta_m) \tag{3-17}$$

如何生成一系列的采样点θ_m呢？如果$P(\theta|\mathcal{X})$已知，可以根据这个概率密度函数进行抽样生成。但$P(\theta|\mathcal{X})$的计算一般比较困难，可以使用马尔可夫链蒙特卡洛(Markov Chain Monte Carlo，MCMC)理论的方法，在不计算贝叶斯公式(3-8)的分母$p(\mathcal{X})$的情况下，对参

数的后验概率 $P(\theta|\mathcal{X})$ 进行随机抽样，生成采样点序列 θ_m，进而实现概率密度函数的估计。Gibbs 抽样和 Metropolis-Hastings 算法是两种最流行的方法，本书不详细介绍这些方法，有兴趣的读者可以查阅相关资料。

3. 贝叶斯学习

为了反映样本数目，重新标记样本集：$\mathcal{X}^N=\{x_1,x_2,\cdots,x_N\}$，$\theta$ 的贝叶斯估计量为

$$\hat{\theta}=\int_{\Theta}\theta P(\theta\mid\mathcal{X}^N)\,\mathrm{d}\theta \tag{3-18}$$

其中，$P(\theta|\mathcal{X}^N)$ 为 θ 的后验分布为

$$P(\theta\mid\mathcal{X}^N)=\frac{p(\mathcal{X}^N\mid\theta)\cdot P(\theta)}{\int p(\mathcal{X}^N\mid\theta)\cdot P(\theta)\mathrm{d}\theta} \tag{3-19}$$

样本独立抽取，当 $N>1$ 时，有

$$p(\mathcal{X}^N\mid\theta)=p(x_N\mid\theta)\,p(\mathcal{X}^{N-1}\mid\theta) \tag{3-20}$$

将式(3-20)代入式(3-19)得递推公式

$$P(\theta\mid\mathcal{X}^N)=\frac{p(x_N\mid\theta)P(\theta\mid\mathcal{X}^{N-1})}{\int p(x_N\mid\theta)P(\theta\mid\mathcal{X}^{N-1})\,\mathrm{d}\theta} \tag{3-21}$$

把先验概率记作 $P(\theta|\mathcal{X}^0)=P(\theta)$，表示在没有样本情况下的概率密度估计。随着样本数的增加，得到一系列对概率密度函数参数的估计，即

$$P(\theta),P(\theta\mid x_1),P(\theta\mid x_1,x_2),\cdots,P(\theta\mid x_1,x_2,\cdots,x_N),\cdots \tag{3-22}$$

称作递推的贝叶斯估计。随着样本数的增加，式(3-22)的后验概率序列逐渐尖锐，逐步趋于以 θ 的真实值为中心的一个尖峰，当样本无穷多时收敛在参数真实值上的脉冲函数，这一过程称作贝叶斯学习。

【例 3-9】 设 $\mathcal{X}=\{x_1,x_2,\cdots,x_N\}$ 是来自总体分布为 $N(\mu,\sigma^2)$ 的样本集，已知 μ 服从 $N(0,9)$ 分布，$\sigma=2$，用贝叶斯学习的方法求 μ 的估计量 $\hat{\mu}$。

程序如下：

```
from scipy.stats import norm
import numpy as np
from matplotlib import pyplot as plt
N, mu0, sigma0 = 300, 0, 3                          # 样本数,μ的先验分布参数
mu_value = np.linspace(-10, 10, 200)                # μ的取值范围
prior = norm.pdf(mu_value, mu0, sigma0)             # μ的先验概率密度函数
plt.plot(mu_value, prior, 'b-', label='N=0')        # 绘制先验分布曲线
mu, sd = 2, 2                                       # 样本服从的正态分布参数
training = norm.rvs(mu, sd, N, random_state=None)   # 生成样本集
prev = prior
c = ['g:', 'm--', 'c-.', 'r-']                      # 不同曲线的标记
id_fmt = 0
for i in range(N):
    n_pdf = norm.pdf(training[i], mu_value, sd)     # 当前样本对应的 p(x_i|μ)
    numerator = n_pdf * prev                        # 计算 p(X^i|μ)=p(x_i|μ)p(X^{i-1}|μ)P(μ)
    prev = numerator
    total_P = np.sum(numerator)                     # 贝叶斯公式的分母
    poster = numerator / total_P                    # 计算 P(μ|X^i)
```

```
        if i == 9 or i == 49 or i == 99 or i == 299:          #绘制后验概率曲线
            plt.plot(mu_value, poster, c[id_fmt], label = 'N = %d' % (i + 1))
            id_fmt = id_fmt + 1
    pos = np.where(poster == poster.max())                    #确定 P(μ|𝒳^N)的最大值
    plt.rcParams['font.sans-serif'] = ['Times New Roman']
    plt.axvline(x = mu_value[pos], linestyle = ':', color = 'r', label = 'μ')    #绘制估计量竖直线
    plt.xticks(mu_value[pos], np.around(mu_value[pos], 2))
    plt.xlabel('μ')
    plt.legend()
    plt.show()
```

运行程序，绘制的一系列概率密度函数如图 3-3 所示。可以看出，随着样本数 N 的增大，θ 的后验概率密度函数逐渐尖锐。后验概率序列最后一个的尖峰在 $\mu=2.06$ 处出现，即估计的 μ 为 2.06。受计算机计算能力限制，程序中 N 取值有限。

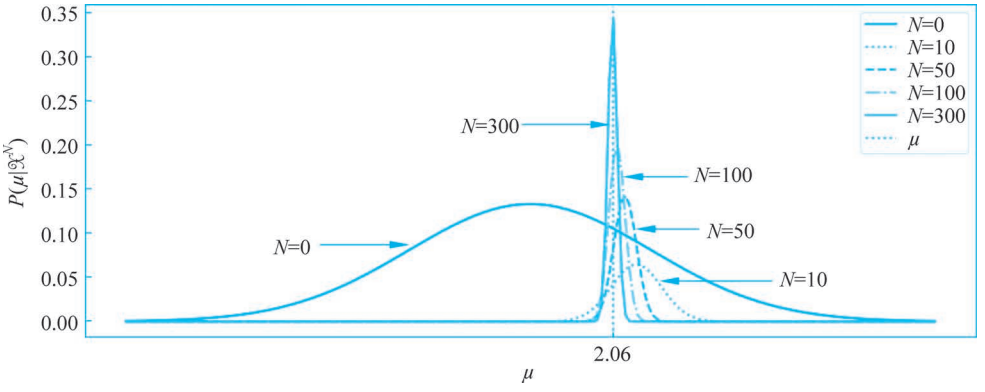


图 3-3 一系列的后验概率估计图

3.3 非参数估计

很多情况下对样本的分布并没有充分的了解，无法事先给出密度函数的形式，而且有些样本分布的情况也很难用简单的函数描述。在这种情况下，就需要进行非参数估计，直接用样本估计出整个函数，即直接计算 $p(\boldsymbol{x})$ 的值。

3.3.1 直方图方法

1. 基本原理

以一维为例，解释直方图(Histogram)方法估计概率密度函数的含义。将数据 x 的取值范围分为若干个长度相等的区间，统计每个区间内的样本数，以样本数占总数的比例作为该区间内的概率密度值，即直方图方法。例如，图像的灰度直方图，就是统计图像中一维灰度数据的概率密度。

如果推广到 n 维，采用直方图方法估计概率密度函数的做法如下。

(1) 把 n 维样本的每个分量在其取值范围内分成 m 个等间隔的小窗，则这种分割会得到 m^n 个小舱，每个小舱的体积记作 V。

(2) 统计落入每个小舱内的样本数目 k。

(3) 把每个小舱内的概率密度看作常数，并用 k/NV 作为其估计值(N 为样本数)，即

小区域范围内概率密度函数为

$$\hat{p}(\boldsymbol{x})=\frac{k}{NV} \tag{3-23}$$

2. 仿真实现

NumPy 库提供了统计直方图的函数，主要的几个函数如下：

(1) hist，bin_edges＝histogram(a，bins＝10，range＝None，density＝None，weights＝None)：对数据 a 统计一维直方图，histogram 函数主要参数见表 3-1。

表 3-1　histogram 函数主要参数

参数名称	取值及含义
bins	设置为整数时指直方图中柱的数目，数据区间等分；设置为向量时各分量代表各柱边界；设置为字符串时指定确定最优柱宽和数目的方法
range	指定柱的范围，范围外的将被忽略，默认情况下为(a. min()，a. max())
density	取 False 时，返回的 hist 为各柱内样本数目；取 True 时，返回的 hist 为各柱对应的概率密度值
weights	数据权系数
hist	返回的直方图数据，可取柱内样本数目或概率密度值，取决于 density 参数
bin_edges	各柱的左边界以及最后一个柱的右边界

(2) H，xedges，yedges＝histogram2d(x，y，bins＝10，range＝None，density＝None，weights＝None)：统计二维直方图。参数 x 和 y 分别是待统计数据的第一维和第二维坐标数组。其他参数同 histogram 函数的参数含义一致，但是是二维情况。

(3) H，edges＝histogramdd(sample，bins＝10，range＝None，density＝None，weights＝None)：统计多维直方图。返回的 edges 是各维柱的边界列表。

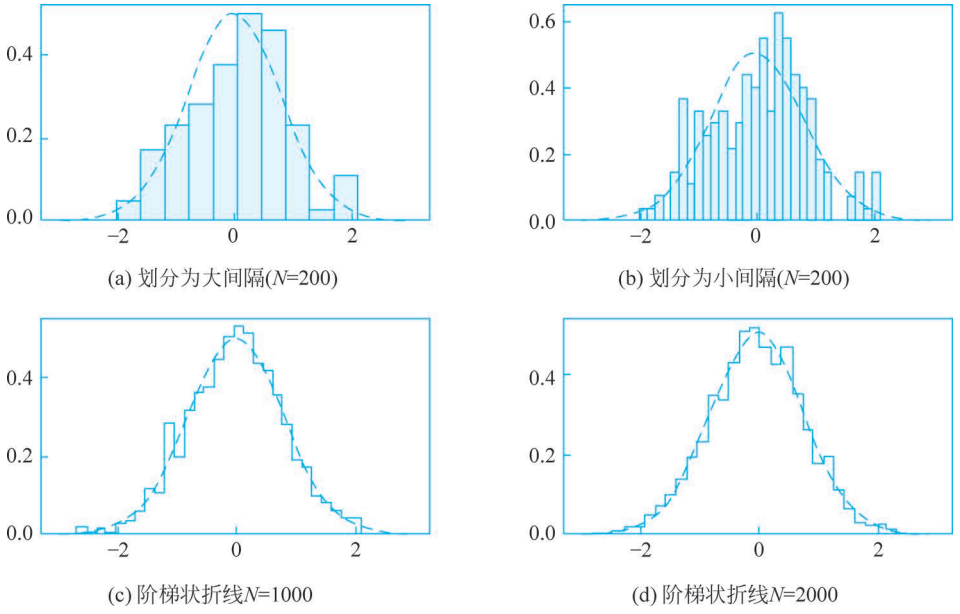
【例 3-10】　设定参数，生成正态分布数据集，利用直方图方法估计概率密度函数并绘制函数曲线。

程序如下：

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm
N, mu, sd = 200, 0, 0.8                                    #指定样本数、正态分布参数
x = np.random.default_rng().normal(mu, sd, N)              #生成数据集
x_values = np.linspace(-3, 3, 600)
px = norm.pdf(x_values, mu, sd)                            #计算概率密度函数
count1, bin1 = np.histogram(x, bins=10, density=True)      #设定 10 个区间进行估计，大间隔
count2, bin2 = np.histogram(x, bins=30, density=True)      #设定 30 个区间进行估计，小间隔
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.subplot(121)
plt.plot(x_values, px, 'k--')                              #绘制概率密度函数曲线
plt.hist(bin1[:-1], bin1, weights=count1, edgecolor='k')   #绘制第一幅直方图
plt.subplot(122)
plt.plot(x_values, px, 'k--')
plt.hist(bin2[:-1], bin2, weights=count2, edgecolor='k')   #绘制第二幅直方图
#plt.stairs(count2, bin2)                                  #绘制阶梯状折线图
plt.show()
```

运行程序，绘制的概率密度函数曲线如图 3-4(a)和图 3-4(b)所示；修改样本数为

1000,30 个柱,绘制阶梯状折线,如图 3-4(c)所示;样本数为 2000 时,绘制的阶梯状折线如图 3-4(d)所示。同等间隔,随着样本数的增多,估计的效果逐渐提高。所以,估计概率密度函数本身需要充足的样本。



(a) 划分为大间隔(N=200)　(b) 划分为小间隔(N=200)

(c) 阶梯状折线N=1000　(d) 阶梯状折线N=2000

图 3-4　利用直方图方法估计概率密度函数

3. 方法分析

直方图方法估计的效果与小舱的选择密切相关。当小舱较大时,每个小舱内设概率密度为常数,估计出的概率密度函数粗糙,如图 3-4(a)所示;当小舱较小时,有些小舱内可能会没有样本或者样本很少,导致估计出的概率密度函数不连续,如图 3-4(b)所示。所以,小舱的大小要合理选择。

小舱的选择应与总样本数相适应。直方图方法在每个小舱内用常量表示概率密度函数值,当样本数 $N\to\infty$时,近似值收敛于真实值的条件是:随样本数的增加,小舱体积应该尽可能小;同时必须保证小舱内有充分多的样本,但每个小舱内的样本数又必须是总样本数中很小的一部分,用公式表示为

$$\lim_{N\to\infty} V_N = 0 \quad \lim_{N\to\infty} k_N = \infty \quad \lim_{N\to\infty} \frac{k_N}{N} = 0 \tag{3-24}$$

V_N、k_N 表示小舱体积 V 和小舱内样本数 k 与总样本数 N 有关。

小舱内的样本数与样本分布有关。在有限样本数目下,如果小舱体积相同,样本密度大的地方小舱内有较多的样本,密度小的地方小舱内样本很少甚至没有,将导致密度的估计在样本密度不同的地方表现不一致。所以,最好能够根据样本分布情况调整小舱体积。

小舱数目随着样本维数的增加急剧增多,实际中很难有足够的样本用于估计。例如,如果样本为 100 维,每维分为 4 个间隔,将有 $4^{100}\approx 1.6\times 10^{60}$ 个小舱。即使要保证一个小舱内有一个样本,也需要 1.6×10^{60} 个样本,这是很难达到的,所以,估计时很多小舱内将会没有样本,方法失效。但在低维问题中,直方图方法还是一个用来进行建模和可视化的有效工具。

3.3.2　Parzen 窗法

Parzen 窗法又称核密度估计(Kernel Density Estimation,KDE)法,采用滑动小舱估计每一点的概率密度。

1. 基本原理

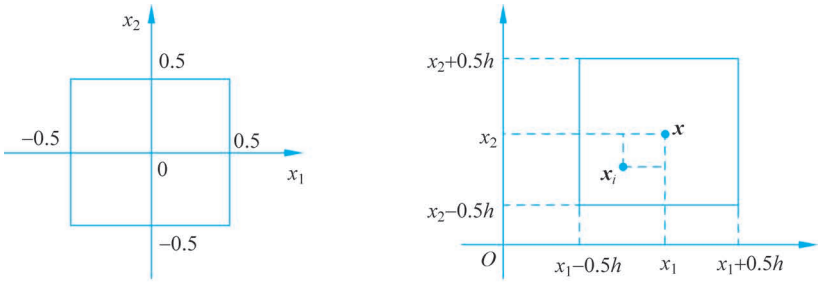
样本 $\boldsymbol{x}\in\mathbf{R}^n$,假设每个小舱是一个超立方体,在每一维的棱长都为 h,则小舱的体积是

$$V=h^n \tag{3-25}$$

定义 n 维单位方窗函数为

$$\varphi([u_1 \quad u_2 \quad \cdots \quad u_n]^{\mathrm{T}})=\begin{cases}1, & |u_j|\leqslant\dfrac{1}{2},j=1,2,\cdots,n\\ 0, & \text{其他}\end{cases} \tag{3-26}$$

该式的含义是:在以原点为中心的 n 维单位超立方体内的所有点的函数值为 1,其他点的函数值为 0,如图 3-5(a)所示。



(a) 原点为中心棱长为1的正方形　　(b) x点为中心棱长为h的正方形

图 3-5　二维的方窗函数示意图

如果一个样本 $\boldsymbol{x}_i$ 在以 $\boldsymbol{x}$ 为中心的棱长为 h 的超立方体内,$\boldsymbol{x}_i$ 到 $\boldsymbol{x}$ 的每一维距离都小于或等于 $h/2$,即向量 $|\boldsymbol{x}-\boldsymbol{x}_i|/h$ 的每一维都小于或等于 1/2,则 $\varphi[(\boldsymbol{x}-\boldsymbol{x}_i)/h]=1$;否则 $\varphi[(\boldsymbol{x}-\boldsymbol{x}_i)/h]=0$,如图 3-5(b)所示。将统计落入以 $\boldsymbol{x}$ 为中心的超立方体内的样本数,转换为统计 $\varphi[(\boldsymbol{x}-\boldsymbol{x}_i)/h]=1$ 的次数,即

$$k=\sum_{i=1}^{N}\varphi\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right) \tag{3-27}$$

将式(3-27)代入式(3-23),得任意一点 $\boldsymbol{x}$ 的概率密度估计表达式

$$\hat{p}(\boldsymbol{x})=\frac{1}{NV}\sum_{i=1}^{N}\varphi\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right) \tag{3-28}$$

或

$$\hat{p}(\boldsymbol{x})=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{V}\varphi\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right) \tag{3-29}$$

定义窗函数(核函数)为

$$K(\boldsymbol{x},\boldsymbol{x}_i)=\frac{1}{V}\varphi\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right) \tag{3-30}$$

反映了一个观察样本 $\boldsymbol{x}_i$ 对在 $\boldsymbol{x}$ 处的概率密度估计的贡献与样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}$ 的距离有关。概率密度估计即在每一点上把所有观测样本的贡献进行平均,为

$$\hat{p}(\boldsymbol{x})=\frac{1}{N}\sum_{i=1}^{N}K(\boldsymbol{x},\boldsymbol{x}_i) \tag{3-31}$$

所以，这种用窗函数估计概率密度的方法被称作 Parzen 窗法或核密度估计法，参数 h 也常被称为带宽(Bandwidth)、窗宽或平滑参数。

2. 窗函数选择

由式(3-31)可知，如果通过定义窗函数直接概率密度估计，要求估计出的概率密度函数满足非负且积分为 1，也就是要求窗函数满足

$$K(\boldsymbol{x},\boldsymbol{x}_i)\geqslant 0 \quad 且 \quad \int K(\boldsymbol{x},\boldsymbol{x}_i)\mathrm{d}\boldsymbol{x}=1 \tag{3-32}$$

常用的窗函数如下。

1) 方窗函数

综合式(3-26)和式(3-30)，得方窗函数

$$K(\boldsymbol{x},\boldsymbol{x}_i)=\begin{cases}\dfrac{1}{h^n}, & |\boldsymbol{x}^j-\boldsymbol{x}_i^j|\leqslant\dfrac{h}{2}, \quad j=1,2,\cdots,n\\ 0, & 其他\end{cases} \tag{3-33}$$

2) 高斯窗函数

一维的单位高斯窗函数为

$$\varphi(u)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}u^2\right) \tag{3-34}$$

得一维情况下的高斯窗函数为

$$K(x,x_i)=\frac{1}{\sqrt{2\pi}h}\exp\left[-\frac{(x-x_i)^2}{2h^2}\right] \tag{3-35}$$

假设样本 $\boldsymbol{x}$ 属性条件独立，n 维情况下的高斯窗函数表示为

$$K(\boldsymbol{x},\boldsymbol{x}_i)=\frac{1}{(2\pi)^{n/2}h^n}\exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}_i)}{2h^2}\right] \tag{3-36}$$

3) 三角窗函数

一维的单位三角窗函数为

$$\varphi(u)=\begin{cases}1-|u|, & |u|\leqslant 1\\ 0, & 其他\end{cases} \tag{3-37}$$

得一维情况下的三角窗函数

$$K(x,x_i)=\begin{cases}\dfrac{1}{h}\left(1-\dfrac{|x-x_i|}{h}\right), & |x-x_i|\leqslant h\\ 0, & 其他\end{cases} \tag{3-38}$$

n 维情况下的三角窗函数为

$$K(\boldsymbol{x},\boldsymbol{x}_i)=\begin{cases}\dfrac{1}{h^n}\left(1-\dfrac{|\boldsymbol{x}-\boldsymbol{x}_i|}{h}\right), & |\boldsymbol{x}^j-\boldsymbol{x}_i^j|\leqslant h, \quad j=1,2,\cdots,n\\ 0, & 其他\end{cases} \tag{3-39}$$

4) Epanechnikov 窗函数

一维的单位 Epanechnikov 窗函数为

$$\varphi(u)=\begin{cases}\dfrac{3}{4\sqrt{5}}\left(1-\dfrac{u^2}{5}\right), & |u|\leqslant\sqrt{5}\\ 0, & \text{其他}\end{cases}\tag{3-40}$$

得一维情况下的 Epanechnikov 窗函数为

$$K(x,x_i)=\begin{cases}\dfrac{3}{4\sqrt{5}h}\left[1-\dfrac{(x-x_i)^2}{5h^2}\right], & |x-x_i|\leqslant\sqrt{5}h\\ 0, & \text{其他}\end{cases}\tag{3-41}$$

设样本 $\boldsymbol{x}$ 属性条件独立，n 维情况下的 Epanechnikov 窗函数为

$$K(\boldsymbol{x},\boldsymbol{x}_i)=\begin{cases}\dfrac{3}{4\sqrt{5}h^n}\left[1-\dfrac{(\boldsymbol{x}-\boldsymbol{x}_i)^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}_i)}{5h^2}\right], & |\boldsymbol{x}^j-\boldsymbol{x}_i^j|\leqslant\sqrt{5}h,j=1,2,\cdots,n\\ 0, & \text{其他}\end{cases}\tag{3-42}$$

Epanechnikov 窗函数被证明是最小平方误差意义上的最优窗函数，证明可参看非参数统计类书籍；实践中高斯窗应用更广泛。

3. 带宽的选择

带宽 h 对估计有很大的影响，带宽的选择是核密度估计中一个很重要的问题。

估计的密度函数 $\hat{p}(\boldsymbol{x})=\dfrac{1}{VN}\sum_{i=1}^{N}\varphi\left(\dfrac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right)$，由 $V=h^n$ 可知：h 较大时，$\dfrac{1}{V}\varphi\left(\dfrac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right)$ 的峰值较小，宽度较大，$\hat{p}(\boldsymbol{x})$ 是 N 个宽度较大的缓慢变化函数的迭加，估计值 $\hat{p}(\boldsymbol{x})$ 受到过度的平均作用，$p(\boldsymbol{x})$ 比较细致的性质不能显露，估计分辨率较低。

如果 h 较小，$\dfrac{1}{V}\varphi\left(\dfrac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right)$ 的峰值较大，宽度较小，$\hat{p}(\boldsymbol{x})$ 就变成 N 个以 $\boldsymbol{x}_i$ 为中心的尖脉冲的迭加，呈现不规则的形状，从而使估计不稳定。

h 应随着样本数 N 增大而缓慢下降，从理论上来说，会随着 $N\to\infty$ 而趋于零。实际问题中样本数有限，只能进行折中选择。

【例 3-11】 设待估计的 $p(x)$ 是均值为零、方差为 1 的正态密度函数，用 Parzen 窗法估计 $p(x)$。

设计思路如下。

(1) 确定窗函数：选择正态窗函数并确定窗宽，窗函数为

$$K(x,x_i)=\frac{1}{\sqrt{2\pi}h}\exp\left[-\frac{(x-x_i)^2}{2h^2}\right]$$

(2) 计算估计值

$$\hat{p}(x)=\frac{1}{N}\sum_{i=1}^{N}K(x,x_i)=\frac{1}{hN\sqrt{2\pi}}\sum_{i=1}^{N}\exp\left[-\frac{(x-x_i)^2}{2h^2}\right]$$

当采集到样本后，可以根据上式计算估计式 $\hat{p}(x)$。

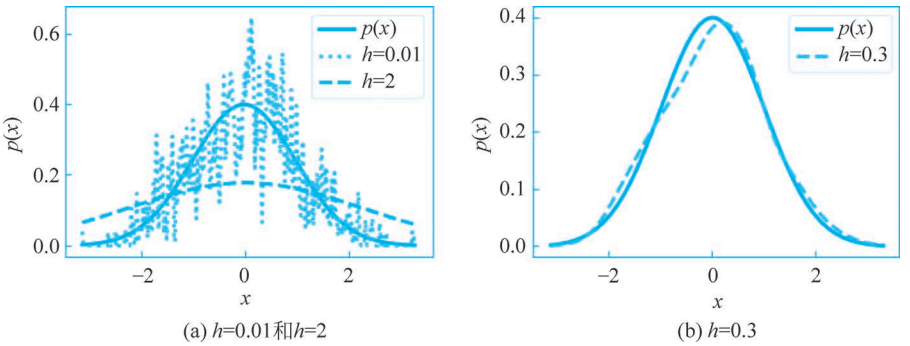
程序如下：

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm
```

```
def kde(x, sample, h, len_x):                                    #利用高斯窗进行估计的函数
    pxe = np.zeros([len_x, 1])
    for j in range(len_x):
        for i in range(len_x):
            pxe[j] += np.exp(-((sample[j] - x[i]) / h) ** 2 / 2) / np.sqrt(2 * np.pi)
        pxe[j] /= (len_x * h)
    return pxe

N, mu, sd = 1000, 0, 1
training = norm.rvs(mu, sd, N)                                   #生成1000个样本
min_x, max_x = training.min(), training.max()
x_value = np.linspace(min_x, max_x, N)                           #确定采样点x
px = norm.pdf(x_value, mu, sd)                                   #确定概率密度函数
h1 = 0.01                                                        #设置较小的带宽
pxe1 = kde(training, x_value, h1, N)                             #估计概率密度函数
h2 = 2                                                           #设置较大的带宽
pxe2 = kde(training, x_value, h2, N)
h3 = 0.3                                                         #设置较适中的带宽
pxe3 = kde(training, x_value, h3, N)
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.subplot(121)
plt.plot(x_value, px, 'k-', lw=2, label='p(x)')                  #绘制原始的概率密度函数曲线
plt.plot(x_value, pxe1, 'r:', lw=2, label='h=0.01')              #绘制估计的概率密度函数曲线
plt.plot(x_value, pxe2, 'b--', lw=2, label='h=2')                #绘制估计的概率密度函数曲线
plt.xlabel('x')
plt.ylabel('p(x)')
plt.legend()
plt.subplot(122)
plt.plot(x_value, px, 'k-', lw=2, label='p(x)')
plt.plot(x_value, pxe3, 'g--', lw=2, label='h=0.3')
plt.xlabel('x')
plt.ylabel('p(x)')
plt.legend()
plt.show()
```

程序运行结果如图 3-6 所示。图 3-6(a)中实线为原始的概率密度函数曲线；点线为带宽 $h=0.01$ 时估计的概率密度函数曲线，由于 h 较小，因此估计的曲线相对于原始曲线起伏较大，估计不稳定；虚线是带宽 $h=2$ 时估计的概率密度函数曲线，由于 h 较大，因此估计的曲线相对于原始曲线较平滑，但跟不上函数 $p(x)$ 的变化。图 3-6(b)中虚线为带宽 $h=0.3$ 时估计的概率密度函数曲线，估计的较准确。正如前面分析时所述，带宽 h 对估计结果影响较大。



(a) h=0.01和h=2　　(b) h=0.3

图 3-6　利用高斯窗函数估计概率密度函数

【例 3-12】　产生 1/64/256/10000 个服从一维标准正态分布的样本，用带宽为 $h=1/\sqrt{N}$ 的 Epanechnikov 窗函数估计样本的概率密度函数。

程序如下：

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm
N = np.array([1, 64, 256, 10000])
mu, sd, x_num = 0, 1, 800
x_value = np.linspace(-4, 4, x_num)
h = 1/np.sqrt(N)                    #设置带宽随样本数增多逐渐下降
R = np.zeros([N.size, N[3]])
for m in range(N.size):
    R[m, 0:N[m]] = norm.rvs(mu, sd, N[m])    #生成样本矩阵,每一行对应一种样本数
px = norm.pdf(x_value, mu, sd)       #真实的概率密度函数
for m in range(N.size):              #针对不同样本数分别估计概率密度函数
    pxe = np.zeros([x_num, 1])
    for i in range(x_num):
        for j in range(N[m]):
            if np.abs(x_value[i] - R[m, j]) <= np.sqrt(5) * h[m]:
                pxe[i] += (1-((x_value[i]-R[m, j])/h[m])**2/5)*3/4/np.sqrt(5)/h[m]
        pxe[i] /= N[m]                          #采用 Epanechnikov 窗函数进行估计
    plt.subplot(1, 4, m+1)
    plt.plot(x_value, px, 'k', label='p(x)')    #绘制真实的概率密度函数
    strr = 'N=' + str(N[m])
    plt.plot(x_value, pxe, 'r--', lw=1, label=strr)
    plt.xlim(-3, 3)
    plt.ylim(0, 1)
    plt.xlabel('x')
    plt.legend(loc='upper center')
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.show()
```

程序运行结果如图 3-7 所示，图中实线为原始的概率密度函数曲线；虚线为估计的概率密度函数曲线，可以看出，样本数越多估计效果越好。

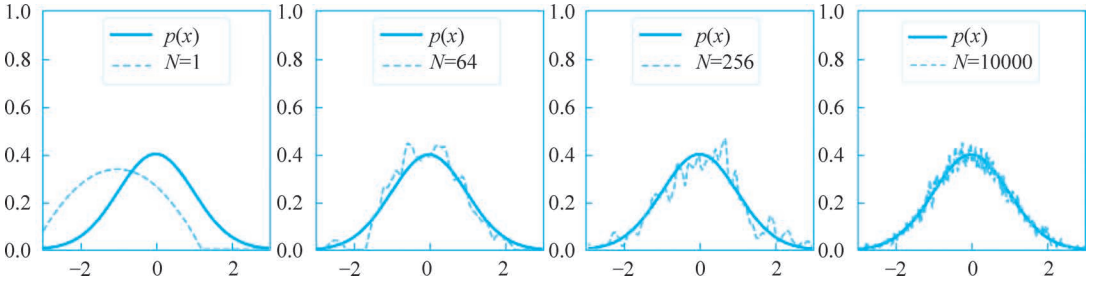


图 3-7　利用 Epanechnikov 窗函数估计概率密度函数

4. 仿真实现

scikit-learn 库的 neighbors 模块提供了进行核密度估计的 KernelDensity 类，其调用格式为

```
KernelDensity(*, bandwidth=1.0, algorithm='auto', kernel='gaussian', metric='euclidean',
atol=0, rtol=0, breadth_first=True, leaf_size=40, metric_params=None)
```

其参数如表 3-2 所示。

表 3-2 KernelDensity 类参数

名　称	含　义
bandwidth	设置为浮点数时指窗函数窗宽，设置为字符串“scott”或“silverman”时采用对应方法估计窗宽
algorithm	指定搜索算法以提高运算速度，可取'kd_tree'、'ball_tree'或'auto'
kernel	指定窗函数，可取'gaussian'、'tophat'、'epanechnikov'、'exponential'、'linear'和'cosine'，对应高斯窗函数、方窗函数、Epanechnikov 窗函数、指数窗函数、三角窗函数和余弦窗函数
metric	指定距离度量方法，可取'cityblock'、'cosine'、'euclidean'、'haversine'、'l1'、'l2'、'manhattan'、'nan_euclidean'等值，采用不同搜索方法时并不是所有距离度量都有效
atol 和 rtol	绝对容差和相对容差，给出终止搜索算法的条件
breadth_first	取 true，采用广度优先搜索；取 false，采用深度优先搜索
leaf_size	树的叶节点数目
metric_params	采用的距离度量的相关参数

主要的方法如下。

(1) fit(X[, y, sample_weight])：根据训练样本 X 及其权值 sample_weight 拟合模型。

(2) sample([n_samples, random_state])：根据模型生成随机样本，目前仅适用于'gaussian'、'tophat'窗函数。

(3) score(X[, y])：计算总的对数似然函数。

(4) score_samples(X)：计算每个样本的对数似然函数。

【例 3-13】 生成双峰分布的数据集，采用 KernelDensity 类估计概率密度函数，并绘制估计的概率密度函数曲线。

程序如下：

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.neighbors import KernelDensity
from scipy.stats import norm
N, mu1, mu2, sd1, sd2 = 200, 0, 5, 1, 1      #设置样本数、均值和标准差
X = np.concatenate((norm.rvs(mu1, sd1, int(0.3 * N), random_state=1),
                    norm.rvs(mu2, sd2, int(0.7 * N), random_state=1)))[:, np.newaxis]
                                             #生成随机数,用列向量表示
#以下生成真实的概率密度函数并用填充的方式绘制图形
x_value = np.linspace(-5, 10, 1000)[:, np.newaxis]
px = 0.3 * norm(mu1, sd1).pdf(x_value) + 0.7 * norm(mu2, sd2).pdf(x_value)
plt.rcParams['font.sans-serif'] = ['SimSun']
plt.fill(x_value, px, fc="black", alpha=0.2)
#分别采用高斯窗函数和 Epanechnikov 窗函数估计概率密度并绘制图形
fmt = ['r:', 'b--']
kernels = ['gaussian', 'epanechnikov']
for id_fmt, id_k in zip(fmt, kernels):
    kde = KernelDensity(kernel=id_k, bandwidth=0.5).fit(X)
    pxe = np.exp(kde.score_samples(x_value))
    plt.plot(x_value, pxe, id_fmt, lw=2)
plt.legend(['双峰分布', '高斯窗估计', 'Epanechnikov 窗估计'], loc="upper left")
plt.yticks(fontproperties='Times New Roman')
```

```
plt.xticks(fontproperties = 'Times New Roman')
plt.show()
```

程序运行结果如图 3-8 所示。

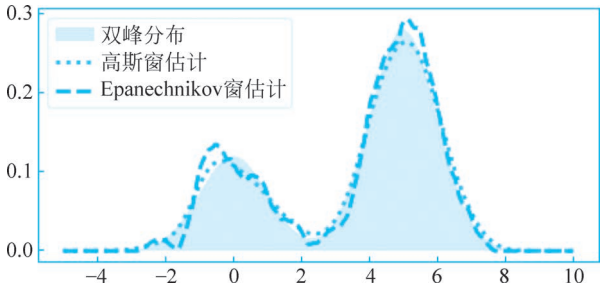


图 3-8　采用 KernelDensity 类估计双峰分布的概率密度函数

3.3.3　k_N 近邻密度估计法

用直方图方法和 Parzen 窗法估计概率密度函数时，点 $\boldsymbol{x}$ 周围的体积固定为 $V=h^n$，其中的样本数 k_N 随机变化。k_N 近邻密度估计法采用可变大小的小舱，固定每个小舱内拥有的样本数 k_N，调整包含 $\boldsymbol{x}$ 的小舱的体积，直到小舱内恰好落入 k_N 个样本，再估计概率密度函数，表示为

$$\hat{p}(\boldsymbol{x})=\frac{k_N}{NV(\boldsymbol{x})} \tag{3-43}$$

$V(\boldsymbol{x})$ 依赖于 $\boldsymbol{x}$，在低密度区，体积大；在高密度区，体积小。

实际中，计算所有样本点之间的距离，对于每一个样本点，确定其周围 k_N 个近邻，以到最远近邻点的距离为半径 r，得到超球体并计算体积。如果采用 Mahalanobis 距离，将会得到超椭球体，体积定义为

$$V=V_0\,|\boldsymbol{\Sigma}|^{\frac{1}{2}}r^n \tag{3-44}$$

其中，V_0 是半径为 1 的超球体的体积，定义为

$$V_0=\begin{cases}\pi^{\frac{n}{2}}/\left(\dfrac{n}{2}\right)!, & n\text{ 为偶数}\\[2ex] 2^n\pi^{\frac{n-1}{2}}\left(\dfrac{n-1}{2}\right)!/n!, & n\text{ 为奇数}\end{cases} \tag{3-45}$$

可以证明，对于二维空间的圆，面积为 πr^2；三维空间的球，体积为 $4\pi r^3/3$。

【例 3-14】　设定参数生成正态分布数据集，利用 k_N 近邻密度估计法估计概率密度函数并绘制函数曲线。

程序如下：

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm
N, mu, sd, x_num = 1000, 0, 0.8, 600
training = norm.rvs(mu, sd, N)
kn1, kn2, kn3 = 10, 50, 100                    # 设定三个不同的 k_N
min_x, max_x = training.min(), training.max()
```

```
x_value = np.linspace(min_x, max_x, x_num)
px = norm.pdf(x_value, mu, sd)
index = 0
pxe1, pxe2, pxe3 = np.zeros([x_num, 1]), np.zeros([x_num, 1]), np.zeros([x_num, 1])
for j in range(x_num):
    distance = np.abs(x_value[j] - training)    #计算当前点和样本点之间的距离
    D = sorted(distance)                        #距离升序排列
    V1, V2, V3 = 2 * D[kn1 - 1], 2 * D[kn2 - 1], 2 * D[kn3 - 1]
    pxe1[index], pxe2[index], pxe3[index] = kn1/N/V1, kn2/N/V2, kn3/N/V3
    index += 1
        #第 kN 个距离作为半径,一维空间,确定长度 2r,并按式(3-43)估计密度值
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.subplot(131)                                #绘制第一种 kN 情况下的对比图
plt.plot(x_value, px, 'k-')
plt.plot(x_value, pxe1, 'r--', lw=1)
plt.subplot(132)                                #绘制第二种 kN 情况下的对比图
plt.plot(x_value, px, 'k-')
plt.plot(x_value, pxe2, 'r--', lw=1)
plt.subplot(133)                                #绘制第三种 kN 情况下的对比图
plt.plot(x_value, px, 'k-')
plt.plot(x_value, pxe3, 'r--', lw=1)
plt.show()
```

程序运行结果如图 3-9 所示。可以看出，k_N 的取值影响到估计的结果，取值大估计的相对准确。可以自行更改样本数目或 k_N 值，观察估计结果。

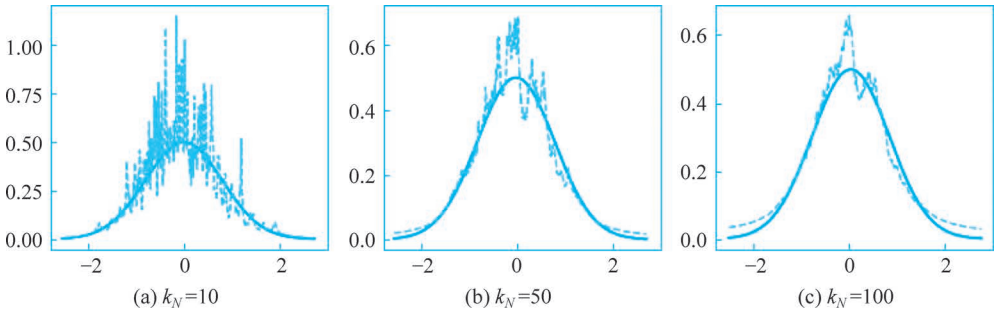


图 3-9　采用 k_N 近邻密度估计法进行概率密度函数估计

非参数估计的共同问题是对样本数目需求较大，只要样本数目足够大，总可以保证收敛于任何复杂的位置密度，但计算量和存储量都比较大。当样本数很少时，若对密度函数有先验认识，参数估计方法能取得更好的估计效果。

3.4　最小错误率贝叶斯决策的实例

【例 3-15】　scikit-learn 库中的 iris 数据集，对 3 种鸢尾花(setosa、versicolor 和 virginica)各抽取了 50 个样本，每个样本含 4 个特征，分别为花萼的长和宽及花瓣的长和宽，单位为厘米。对样本 $\boldsymbol{x}=[4.8 \quad 3.5 \quad 1.5 \quad 0.2]^{\mathrm{T}}$ 进行最小错误率贝叶斯决策。

设计思路如下。

贝叶斯决策需要先了解先验概率和类条件概率密度。题目中仅给出了样本集，样本

集中有3类，样本数相同，可以假设各类先验概率相等，但需要估计类条件概率密度函数。对样本进行最小错误率贝叶斯决策，只需要比较样本对应的类条件概率密度的大小即可归类。

由于并不清楚样本服从什么分布，需要对类条件概率密度进行非参数估计。样本为四维，若每维取10个等距点，则四维空间有10^4个等距点，但每类样本仅50个，样本数量太少，难以支撑四维空间的概率密度函数估计，因此，考虑将样本降维以保证估计的效果。本例中简单取原始样本的第三维和第四维构成新的样本。

综上所述，得设计流程：首先对于输入的样本降维；再进行非参数概率密度函数估计；最后比较样本对应的各类的概率密度函数值的大小，将样本归入取值最大的类。

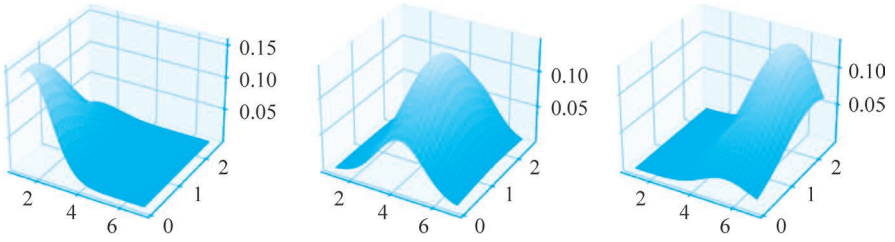
程序如下：

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.neighbors import KernelDensity
from sklearn.datasets import load_iris
iris = load_iris()                                          #导入样本集
X_train = iris.data[:, 2:4]                                 #样本降维
X_se = X_train[iris.target == 0, :]
X_ve = X_train[iris.target == 1, :]
X_vi = X_train[iris.target == 2, :]                         #分别获取3类样本
min_x, max_x = np.min(X_train, 0), np.max(X_train, 0)       #确定估计点取值范围
dx = 0.1
x1, x2 = np.mgrid[min_x[0]:max_x[0] + dx:dx, min_x[1]:max_x[1] + dx:dx]   #构建二维网格
x_test = np.concatenate((x1.reshape(-1, 1), x2.reshape(-1, 1)), 1)    #确定采样点
kde_se = KernelDensity().fit(X_se)
kde_ve = KernelDensity().fit(X_ve)
kde_vi = KernelDensity().fit(X_vi)              #用核密度估计法对各类概率密度进行估计
pxe_se = np.exp(kde_se.score_samples(x_test)).reshape(x1.shape)
pxe_ve = np.exp(kde_ve.score_samples(x_test)).reshape(x1.shape)
pxe_vi = np.exp(kde_vi.score_samples(x_test)).reshape(x1.shape)
                                                #将一维概率密度向量表达为二维形式
X_test = np.array([4.8, 3.5, 1.5, 0.2])
X = X_test[2:4]
pos = ((X - min_x) / dx).astype('int')
P_se = pxe_se[pos[0], pos[1]]
P_ve = pxe_ve[pos[0], pos[1]]
P_vi = pxe_vi[pos[0], pos[1]]                   #获取待决策样本在3类概率密度函数中的取值
maxP = np.max([P_se, P_ve, P_vi])
if P_se == maxP:                                #比较大小并归类
    print('样本[4.8, 3.5, 1.5, 0.2]为setosa类')
elif P_ve == maxP:
    print('样本[4.8, 3.5, 1.5, 0.2]为versicolor类')
else:
    print('样本[4.8, 3.5, 1.5, 0.2]为virginica类')
count_se, count_ve, count_vi = 0, 0, 0
for i in range(len(X_train)):                   #对所有样本进行测试，计算总体识别率
    X = X_train[i, :]
    pos = ((X - min_x) / dx).astype('int')
```

```
        P_se = pxe_se[pos[0], pos[1]]
        P_ve = pxe_ve[pos[0], pos[1]]
        P_vi = pxe_vi[pos[0], pos[1]]
        maxP = np.max([P_se, P_ve, P_vi])
        if P_se == maxP and i < 50:
            count_se += 1
        elif P_ve == maxP and 50 <= i < 100:
            count_ve += 1
        elif P_vi == maxP and i >= 100:
            count_vi += 1
ratio = (count_se + count_ve + count_vi) / 150
print("训练样本识别正确率为:{:.2%}".format(ratio))
plt.rcParams['font.sans-serif'] = ['Times New Roman']
fig = plt.figure()
ax1 = fig.add_subplot(131, projection='3d')
ax1.plot_surface(x1, x2, pxe_se, cmap='viridis')
ax2 = fig.add_subplot(132, projection='3d')
ax2.plot_surface(x1, x2, pxe_ve, cmap='viridis')
ax3 = fig.add_subplot(133, projection='3d')
ax3.plot_surface(x1, x2, pxe_vi, cmap='viridis')
plt.show()
```

运行程序,绘制的概率密度函数图形如图 3-10 所示,且输出:

```
样本[4.8, 3.5, 1.5, 0.2]为 setosa 类
训练样本识别正确率为: 95.33%
```



(a) setosa类　(b) versicolor类　(c) virginica类

图 3-10　估计的概率密度函数

习题

1. 简述参数估计和非参数估计的含义。
2. 列举参数估计的方法并简述各方法的含义。
3. 列举非参数估计的方法并简述各方法的含义。
4. 设样本集呈现对数正态分布 $p(x)=\dfrac{1}{\sigma x\sqrt{2\pi}}\exp\left[-\dfrac{(\ln x-\theta)^2}{2\sigma^2}\right]$,$(x>0)$,求 θ 的最大似然估计。
5. 设正态分布的均值分别为$\boldsymbol{\mu}_1=[1\quad 1]^{\mathrm{T}}$ 和$\boldsymbol{\mu}_2=[1.5\quad 1.5]^{\mathrm{T}}$,协方差矩阵均为 $0.2\boldsymbol{I}$,

先验概率相等，决策表 $\boldsymbol{\lambda}=\begin{bmatrix}0 & 1\\ 0.5 & 0\end{bmatrix}$。编写程序，由正态分布生成各 1000 个二维向量的数据集，利用其中的 800 个样本，采用最大似然估计方法估计样本分布的参数，利用最小风险贝叶斯决策方法对其余 200 个样本进行决策，并计算识别率。

6. 打开 diabetes 数据集，分别采用直方图方法、Parzen 窗法和 k_N 近邻密度估计法对其中的血压数据（二维）进行概率密度函数估计。