作业3：贝叶斯决策与（非）参数估计

1. 贝叶斯估计

当损失函数为平方误差函数时，证明在给定样本集X下，θ的贝叶斯估计如下：

$$\theta^* = E[\theta|X] = \int_\theta \theta p(\theta|X)d\theta$$

证明：

$$\theta^* = E[\theta|X] = arg\ min_{\hat{\theta}} R(\hat{\theta}|X)$$
$$R(\hat{\theta}|X) = \int_\theta \lambda(\hat{\theta},\theta)p(\theta|X)d\theta = E(\lambda(\hat{\theta},\theta)|X)$$

由于$\lambda(\hat{\theta},\theta) = (\hat{\theta}-\theta)^2$

$$\begin{aligned} R(\hat{\theta}|X) &= \int_\theta (\hat{\theta}-\theta)^2 p(\theta|X)d\theta \\ &= E((\hat{\theta}-\theta)^2|X) \\ &= E(\hat{\theta}^2 - 2\theta\cdot\hat{\theta} + \theta^2|X) \\ &= \hat{\theta}^2 - 2\hat{\theta}E[\theta|X] + E[\theta^2|X] \end{aligned}$$

给定θ关于X的条件分布后，后两项均为常数，故对$\hat{\theta}$求导有：

$$\frac{\partial R}{\partial \hat{\theta}} = 2\hat{\theta} - 2E[\theta|X] = 0$$
$$\Rightarrow \theta^* = E[\theta|X] = \int_\theta \theta p(\theta|X)d\theta$$

进一步求二阶导有：

$$\frac{\partial^2 R}{\partial \hat{\theta}^2} = 2 > 0$$

故$\theta^* = E[\theta|X]$确实为极小值点，即最小值点，θ的贝叶斯估计是该式。

2. Parzen窗

（1）证明对P（x）估计的期望服从下式

$$\overline{P}(x)=E[\hat{P}(x)]=\begin{cases}0 & x<0\\ \frac{1}{a}(1-e^{-\frac{x}{h}}) & 0\le x\le a\\ \frac{1}{a}(e^{\frac{a}{h}}-1)e^{-\frac{x}{h}} & a<x\end{cases}$$

证明：

$$\varphi(x)\begin{cases}e^{-x} & x>0\\ 0 & x\le 0\end{cases}$$

设小舱的体积为V=h

故核函数为：

$$K(x,x_i)=\frac{1}{V}\varphi(\frac{x-x_i}{h})=\begin{cases}\frac{1}{h}e^{-\frac{x-x_i}{h}} & 0\le x_i<x\\ 0 & x_i\ge x\end{cases}$$

经检验满足$K(x,x_i)\ge 0$且$\int K(x,x_i)dx=1$

则$\hat{P}(x)$的表达式为

$$\hat{P}(x)=\frac{1}{N}\sum_{i=1}^{N}K(x,x_i)=\frac{1}{Nh}\sum_{i=1}^{N}\varphi(\frac{x-x_i}{h})$$

当$x<0$时，对$\forall x_i$，$p(x_i)\sim U[0,a]$且$x>x_i$，故有

$$\hat{P}(x)=0,\quad \overline{P}(x)=E[\hat{P}(x)]=0$$

当$0\le x\le a$时，

$$\begin{aligned}\overline{P}(x)=E[\hat{P}(x)]&=\frac{1}{Nh}E(\sum_{i=1}^{N}\varphi(\frac{x-x_i}{h}))\\ &=\frac{1}{Nh}N\int_0^x e^{-\frac{x-x_i}{h}}p(x_i)dx_i\\ &=\frac{1}{ah}\int_0^x e^{-\frac{x-x_i}{h}}dx_i\\ &=\frac{1}{a}(1-e^{-\frac{x}{h}})\end{aligned}$$

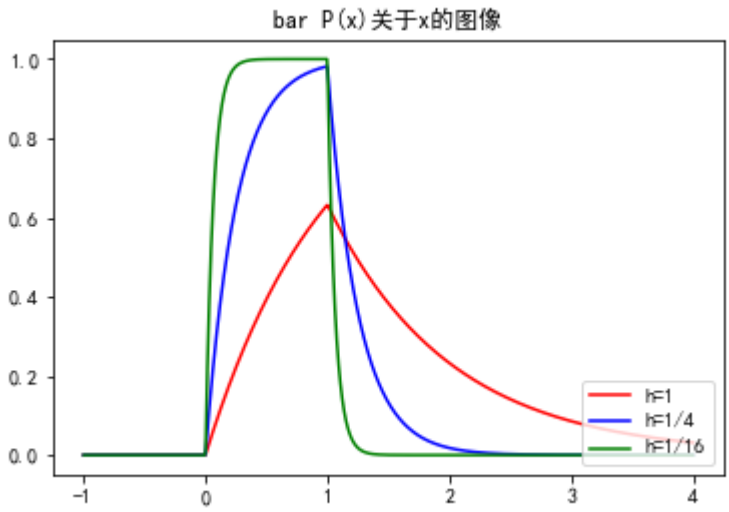
当$a<x$时，同理可得

$$
\begin{aligned}
\overline{P}(x) = E[\hat{P}(x)] &= \frac{1}{Nh} E(\sum_{i=1}^{N} \varphi(\frac{x - x_i}{h})) \\
&= \frac{1}{Nh} N \int_0^a e^{-\frac{x-x_i}{h}} p(x_i) dx_i \\
&= \frac{1}{ah} \int_0^a e^{-\frac{x-x_i}{h}} dx_i \\
&= \frac{1}{a}(e^{\frac{a}{h}} - 1)e^{-\frac{x}{h}}
\end{aligned}
$$

故原结论得证

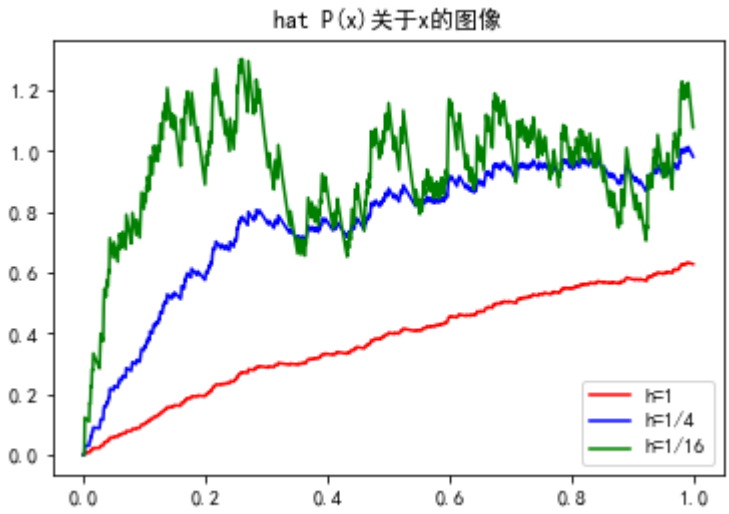
（2）a=1, h=1,1/4, 1/16时，绘制$\bar{P}(x)$关于x的图像

```
#HW2_2_1.py
x = np.linspace(-1, 4,num=1000)
def parzen(x,h):
  n = len(x)#所有的位置
  y=np.zeros((n,1))
  for i in np.arange(0,n):
        if x[i] < 0:
            y[i] = 0
        elif x[i] >= 0 and x[i] <= 1:
            y[i] = 1-pow(np.e,-x[i]/h)
        else:
            y[i] = pow(np.e,-x[i]/h)*(pow(np.e,1/h)-1)
  return y

p1=parzen(x,1) # h=1
p2=parzen(x,0.25) # h=1/4
p3=parzen(x,1/16) # h=1/16

#plt.plot(x1,y1,'y.'); #画图
plt.plot(x,p1,'r',label='h=1')
plt.plot(x,p2,'b',label='h=1/4')
plt.plot(x,p3,'g',label='h=1/16')
plt.title("bar P(x)关于x的图像")
plt.legend(loc='lower right')
plt.savefig('./2_2_1.png')
```



可进一步绘制$\hat{P}(x)$关于x的图像，比较不同h取值时的区别

```
#HW2_2_2.py
xi = np.random.rand(256)#样本点
x = np.linspace(0, 1,num=1000)

def phi(x, xi, h):
# 判断x, xi之间的距离
    if xi < x:
        phi = pow(np.e,-(x-xi)/h);
    else:
        phi = 0;
    return phi

def parzen(xi,x,h):
  n = len(x)#所有的位置
  y=np.zeros((n,1))
  for i in np.arange(0,n):
    m = 0
    for j in np.arange(0,len(xi)):
        m = m + phi(x[i], xi[j], h)
    y[i] = m / (len(xi) * h)
  return y

p1=parzen(xi,x,1) # h=1
p2=parzen(xi,x,0.25) # h=1/4
p3=parzen(xi,x,1/16) # h=1/16
#画图
plt.plot(x,p1,'r',label='h=1')
plt.plot(x,p2,'b',label='h=1/4')
plt.plot(x,p3,'g',label='h=1/16')
plt.title("hat P(x)关于x的图像")
plt.legend(loc='lower right')
plt.savefig('./2_2.png')
```



取样本点N=256时，得到的$\hat{P}(x)$关于x的图像如上图

（3）当0<x<a时，h需要满足什么条件时，保证99%的x对应的$\bar{P}(x)$的偏差才能保持在1%以内?

当0<x<a时

$$\varepsilon = \frac{p(x) - \bar{p}(x)}{p(x)} = e^{-\frac{x}{h}}$$

$$\frac{\partial \varepsilon}{\partial x} = -\frac{1}{h} e^{-\frac{x}{h}} < 0$$

因此x接近于0时偏差最大。若要满足保证99%的x对应的$\bar{P}(x)$的偏差均在1%以内，由于$P(x) \sim U[0, a]$，需要满足$x \in [0.01a, a]$时，$\varepsilon < 0.01$，即只需保证$x = \frac{a}{100}$时偏差在1%以内即可保证条件满足。

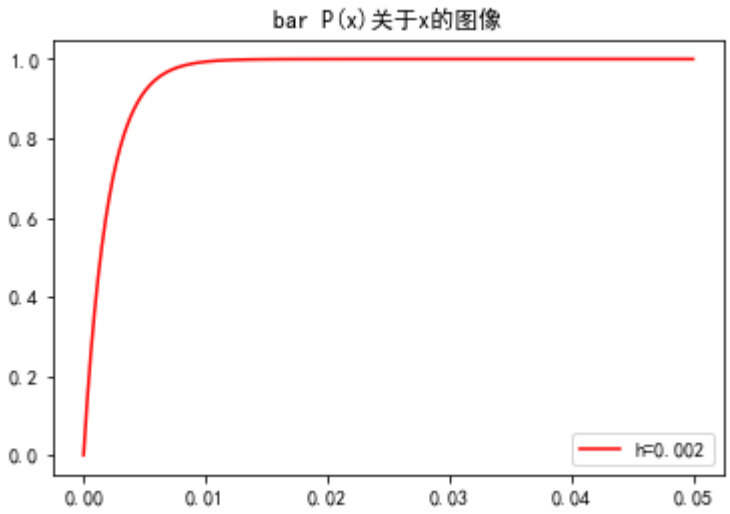
$$e^{-\frac{a}{100h}} < 1\%$$

解得

$$h < 2.171 * 10^{-3} a$$

（4）a=1, 请给出h并绘制在$0 \leq x \leq 0.05$的$\bar{P}(x)$的图像

根据第（3）问，不妨取h=0.002



由图像可以观察到$0.01 \leq x \leq 0.05$时对应的$\bar{P}(x)$的偏差可以保持在1%以内

3. 最大似然估计

（1）求μ的最大似然估计

记已知样本$D = x_1, x_2, \ldots, x_n$

$$L(\mu) = p(D|\mu) = \prod_{i=1}^{n} p(x_i|\mu)$$

$$ln\ p(x^{(i)}|\mu) = -\frac{1}{2}ln(2\pi\sigma^2) - \frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}$$

$$ln\ L(\mu) = \sum_{i=1}^{n} ln\ p(x^{(i)}|\mu) = -\frac{n}{2}ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x^{(i)} - \mu)^2$$

$$\frac{\partial}{\partial\mu} ln\ L(\mu) = \frac{1}{\sigma^2}\sum_{i=1}^{n}(x^{(i)} - \mu)$$

令$\frac{\partial}{\partial\mu} ln\ L(\mu) = 0$, 解得

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

可验证$\frac{\partial^2}{\partial\mu} ln\ L(\mu) < 0$, 故$\mu = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$即为最大似然估计

当添加条件要求$\mu > \mu_0$时

若$\frac{1}{n}\sum_{i=1}^{n} x^{(i)} > \mu_0$, μ的最大似然估计为$\frac{1}{n}\sum_{i=1}^{n} x^{(i)}$

若$\frac{1}{n}\sum_{i=1}^{n} x^{(i)} \leq \mu_0$, μ的最大似然估计为μ_0

（2）利用最大似然估计求正态分布假设下的模型参数

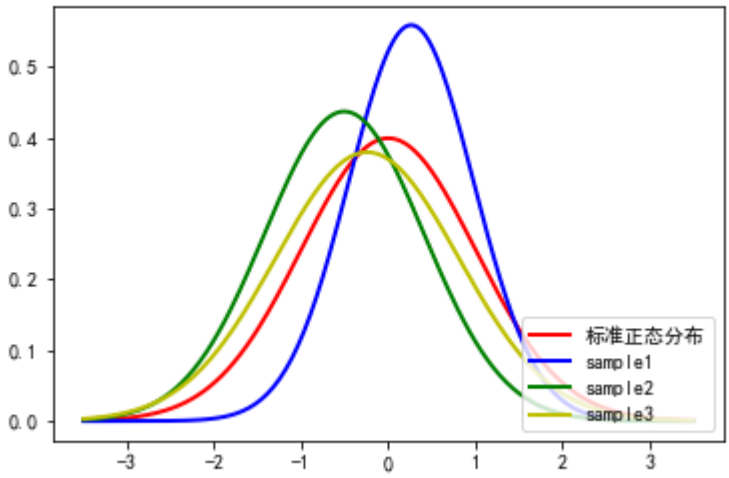
若μ，σ均未知，设模型参数为$\theta = [\mu, \sigma^2]^T$,则

$$\frac{\partial}{\partial \sigma^2} ln\ L(\theta) = -\frac{1}{2\sigma^2} + \frac{(x-\mu)^2}{2\sigma^4}$$

$$\Rightarrow \quad \mu = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

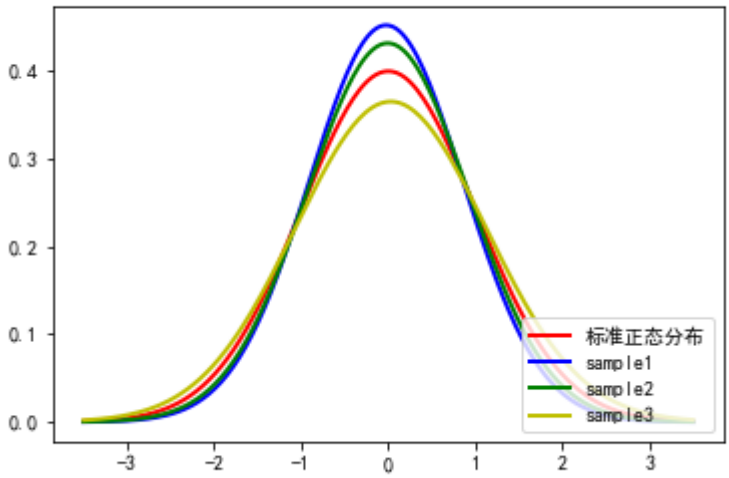
$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \mu)^2$$

```
#以样本数为10为例
num1 = 10
mu = 0
sigma = 1

m=[]
std=[]
s=[]
for i in range(3):
    s1 = np.random.normal(mu, sigma, num1)
    m1= np.mean(s1)
    std1= np.std(s1,ddof=0)
    m.append(m1)
    std.append(std1)
    s.append(s1)

for i in range(3):
    print("mean:%f" %m[i])
    print("std:%f" %std[i])

s_fit = np.linspace(-3.5, 3.5,num=200)
plt.plot(s_fit, st.norm(mu, sigma).pdf(s_fit), lw=2, c='r',label=u'标准正态分布')
plt.plot(s_fit, st.norm(m[0], std[0]).pdf(s_fit), lw=2, c='b',label='sample1')
plt.plot(s_fit, st.norm(m[1], std[1]).pdf(s_fit), lw=2, c='g',label='sample2')
plt.plot(s_fit, st.norm(m[2], std[2]).pdf(s_fit), lw=2, c='y',label='sample3')
plt.legend(loc='lower right')
plt.rcParams['axes.unicode_minus'] = False
plt.savefig('./normal_10.png')
```



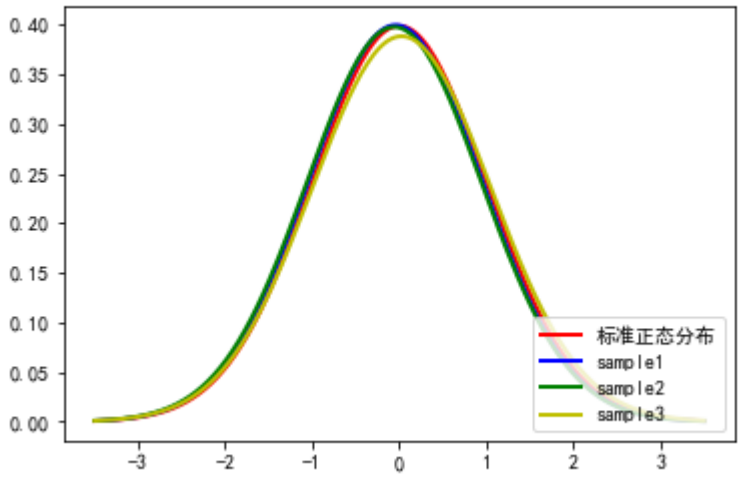

```
#此时模型参数如下:
#sample1
mu:0.262633     sigma:0.714270
#sample2
mu:-0.506162    sigma:0.913685
#sample3
mu:-0.242101    sigma:1.052114
```

样本数为100 时:




```
#此时模型参数如下:
#sample1
mu:-0.025780     sigma:0.884000
#sample2
mu:-0.007205     sigma:0.926024
#sample3
mu:0.027590      sigma:1.094777
```

样本数为1000 时:



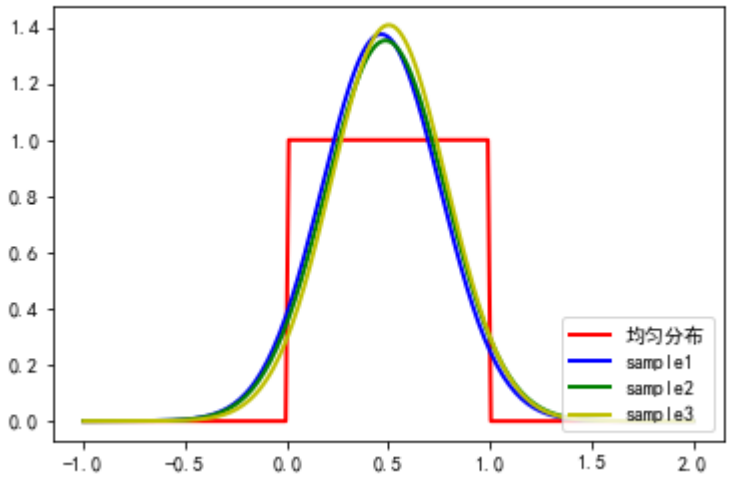

```
#此时模型参数如下:
#sample1
mu:-0.042186    sigma:0.997770
#sample2
mu:-0.060235    sigma:1.002662
#sample3
mu:0.026117    sigma:1.026782
```

具体比较分析见第（4）问

（3）利用最大似然估计求正态分布假设下的均匀分布模型参数

```
num1 = 100
mu = 0
sigma = 1

m=[]
std=[]
s=[]
for i in range(3):
    s1 = np.random.rand(num1)
    m1= np.mean(s1)
    std1= np.std(s1,ddof=0)#分母为n
    m.append(m1)
    std.append(std1)
    s.append(s1)

for i in range(3):
    print("mean:%f" %m[i])
    print("std:%f" %std[i])

s_fit = np.linspace(-1, 2,num=200)
plt.plot(s_fit, st.uniform.pdf(s_fit), lw=2, c='r',label=u'均匀分布')
plt.plot(s_fit, st.norm(m[0], std[0]).pdf(s_fit), lw=2, c='b',label='sample1')
plt.plot(s_fit, st.norm(m[1], std[1]).pdf(s_fit), lw=2, c='g',label='sample2')
```

```
plt.plot(s_fit, st.norm(m[2], std[2]).pdf(s_fit), lw=2, c='y',label='sample3')
plt.legend(loc='lower right')
plt.rcParams['axes.unicode_minus'] = False
plt.savefig('./uniform_100.png')
```




```
#此时模型参数如下:
#sample1
mu:0.464270    sigma:0.289834
#sample2
mu:0.486074    sigma:0.294606
#sample3
mu:0.502207    sigma:0.283086
```

(4)分析讨论模型选择与样本量对参数估计的影响

根据以上实验，分析讨论模型选择和样本量对模型参数估计的影响如下:

进行模型参数估计前首先要进行正确的模型假设，确定概率密度函数的形式。模型假设不同，对应的参数估计问题也不同。模型不合适时，模型参数的估计是没有意义的。此时学习得到的概率密度分布曲线不会趋近于原来的曲线，也无法用于之后的决策。因此所选用的模型假设要能够尽可能好地对样本数据进行拟合。

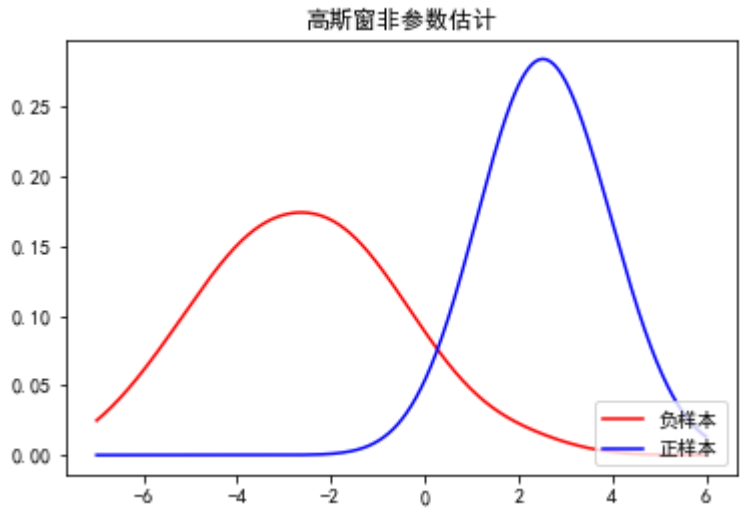
其次，在相同的模型下，样本量越大，模型的学习参数越稳定。一般而言，样本量越大，参数估计越接近真值。样本量较小时，模型参数估计可能有较大的误差， 绘制出的概率密度分布曲线和真实曲线区别较大；样本量较大时，随机误差的影响逐渐减小，模型参数估计逐渐趋于真值。 (需满足估计是渐进无偏的前提)

因此模型选择决定学习结果的正确与否，样本量决定学习结果的精度。

4. 非参数估计与贝叶斯决策

4.1利用高斯核Parzen窗进行非参数估计

采用高斯窗对训练集中的正样本和负样本分别进行非参数估计，即$K(x,x_i)=\frac{1}{\sqrt{2\pi}\sigma}exp\{-\frac{(x-x_i)^2}{2\sigma^2}\}$, 选用$\sigma=1$，得到的正负样本的概率密度函数为图所示




```
#HW4_1.py
#非参数估计函数
def phi(x, xi):
    phi = 1/(np.sqrt(2*np.pi))*np.e**((-1/2)*pow((x-xi),2))
    return phi

def parzen(xi,x):
  n = len(x)
  y=np.zeros((n,1))
  for i in np.arange(0,n):
    m = 0
    for j in np.arange(0,len(xi)):
        m = m + phi(x[i], xi[j])
    y[i] = m / (len(xi))
  return y
#数据生成与处理
np.random.seed(12)#设置种子,保证每次数据相同
x1=np.random.normal(-2.5,2,350);#负样本
x2=np.random.normal(2.5,1,250);#正样本
label1 = [0 for i in np.arange(0, 350, 1)];
label2 = [1 for i in np.arange(0, 250, 1)];
test_data1 = np.vstack((x1, label1));
test_data2 = np.vstack((x2, label2));
data = np.transpose(np.hstack((test_data1, test_data2)));
x_train, x_test, y_train, y_test =
train_test_split(data[:,0],data[:,1],test_size=0.3,random_state=1)#划分训练集与测试集

#以训练集进行非参数估计
x_train_df=pd.DataFrame(x_train)
y_train_df=pd.DataFrame(y_train)
df_train=pd.concat([x_train_df,y_train_df],axis=1,ignore_index=False)
df_train.columns=["Data","Class"]
group_by_class=df_train.groupby('Class')
train0_df=group_by_class.get_group(0)
train1_df=group_by_class.get_group(1)
x_0=np.array(train0_df)[:,0]
```

```
x_1=np.array(train1_df)[:,0]
#绘图
x = np.linspace(-7, 6,num=1000)
p0=parzen(x_0,x)
p1=parzen(x_1,x)
plt.plot(x,p0,'r',label='负样本')
plt.plot(x,p1,'b',label='正样本')
plt.title(u"高斯窗非参数估计")
```

4.2 最小错误率贝叶斯决策

```
p_0=parzen(x_0,x_test)
p_1=parzen(x_1,x_test)
#最小错误率贝叶斯决策
def bayesian_classifier(data,rate1,rate2):
    res=np.zeros((len(rate1)))
    right=0.0
    for i in range(0,len(rate1)):
        if rate1[i]>rate2[i]:
            res[i]=0
        else:
            res[i]=1
        if data[i]==res[i]:
            right+=1
    return right/len(rate1)

accuracy1=bayesian_classifier(y_test,p_0,p_1)
print("Accuracy of Bayes Decision for Minimum Errors:",accuracy1)
```

利用(1)非参数估计的概率密度条件下，采用最小错误率对测试样本进行预测的正确率为96.7%

```
Accuracy of Bayes Decision for Minimum Errors: 0.9666666666666667
```

4.3 最小风险贝叶斯决策

```
#最小风险贝叶斯决策
def bayesian_classifier_minimum_risk(data,rate1,rate2,x_test_0,x_test_1):
    res=np.zeros((len(rate1)))
    risk1=np.zeros((len(rate1)))
    risk2=np.zeros((len(rate1)))
    right=0.0
    negative_right=0.0
    positive_right=0.0
    for i in range(0,len(rate1)):
        risk1[i]=10*rate2[i]
        risk2[i]=rate1[i]
        if risk1[i]<risk2[i]:
            res[i]=0
        else:
            res[i]=1
        if data[i]==res[i] and data[i]==0:
```

```
            negative_right+=1
            right+=1
        elif data[i]==res[i] and data[i]==1:
            positive_right+=1
            right+=1
    print("Accuracy of Negative Sample:",negative_right/len(x_test_0))
    print("Accuracy of Positive Sample:",positive_right/len(x_test_1))
    return right/len(rate1)

accuracy2=bayesian_classifier_minimum_risk(y_test,p_0,p_1,x_test_0,x_test_1)
print("Average Accuracy of Bayes Decision for Minimum Risk:",accuracy2)
```

由于将正样本预测为负样本的代价较大，采用最小风险的贝叶斯决策时可得到正样本的预测正确率为100%，而负样本预测的正确率为78.2%，平均的预测正确率为87.8%

```
Accuracy of Negative Sample: 0.7821782178217822
Accuracy of Positive Sample: 1.0
Average Accuracy of Bayes Decision for Minimum Risk: 0.8777777777777778
```

4.4预测结果比较分析

改变随机划分数据所使用的种子参数`random_state`，在数据集不变的前提下随机划分训练集和测试集样本三次，重复上述步骤，得到预测结果如下

```
#第一次结果
Accuracy of Bayes Decision for Minimum Errors: 0.9722222222222222
Average Accuracy of Bayes Decision for Minimum Risk: 0.9111111111111111
Accuracy of Negative Sample: 0.8490566037735849
Accuracy of Positive Sample: 1.0
#第二次结果
Accuracy of Bayes Decision for Minimum Errors: 0.95
Average Accuracy of Bayes Decision for Minimum Risk: 0.8944444444444445
Accuracy of Negative Sample: 0.8240740740740741
Accuracy of Positive Sample: 1.0
#第三次结果
Accuracy of Bayes Decision for Minimum Errors: 0.9444444444444444
Average Accuracy of Bayes Decision for Minimum Risk: 0.8722222222222222
Accuracy of Negative Sample: 0.8103448275862069
Accuracy of Positive Sample: 0.984375
```

多次测试可以得到，采用最小错误率的贝叶斯决策正确率普遍大于采用最小风险贝叶斯决策的平均正确率。最小风险贝叶斯决策通过牺牲风险较小的决策的正确性，提高了风险较大的决策的正确性。