Machine Learning

题目: 中山大学

学习小组 05 机器学习作业

姓　　名　方桂安, 刘玥, 周敏

学　　号　20354027,20354229,20354187

院　　系　智能工程学院

专　　业　智能科学与技术

指导教师　彭卫文 (副教授)

2021 年 10 月 20 日

一、描述逻辑回归模型

1.1数据

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}, x_i \in \mathbb{R}^n, y_i \in \{0, 1\}$$

1.2模型

$$p(y=1|x) = \frac{1}{1+e^{-(w^T x+b)}}$$
$$p(y=0|x) = \frac{e^{-(w^T x+b)}}{1+e^{-(w^T x+b)}}$$

最初模型：

$$f(x_i) = \omega^T x_i + b, \text{使得} f(x_i) \simeq g(y_i)$$

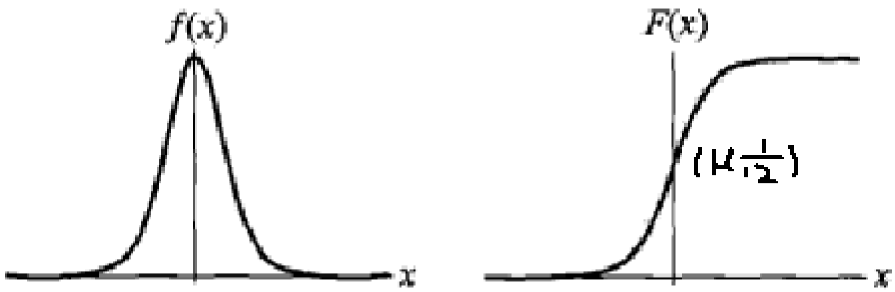
我们的标记变量y的范围是0或1，所以我们需要一个函数能够将上述x的线性组合转化为0或1，最理想的是阶跃函数。

$$\text{阶跃函数：} y = g^{-1}(\omega^T x + b) = \begin{cases} 0, & \omega^T x + b < 0 \\ 0.5, & \omega^T x + b = 0 \\ 1, & \omega^T x + b > 0 \end{cases}$$

但由于阶跃函数不连续，不满足单调可微的条件。所以我们希望通过一个一定程度上近似阶跃函数的“替代函数”，并且希望它单调可微。由此，我们想到了逻辑斯蒂函数。

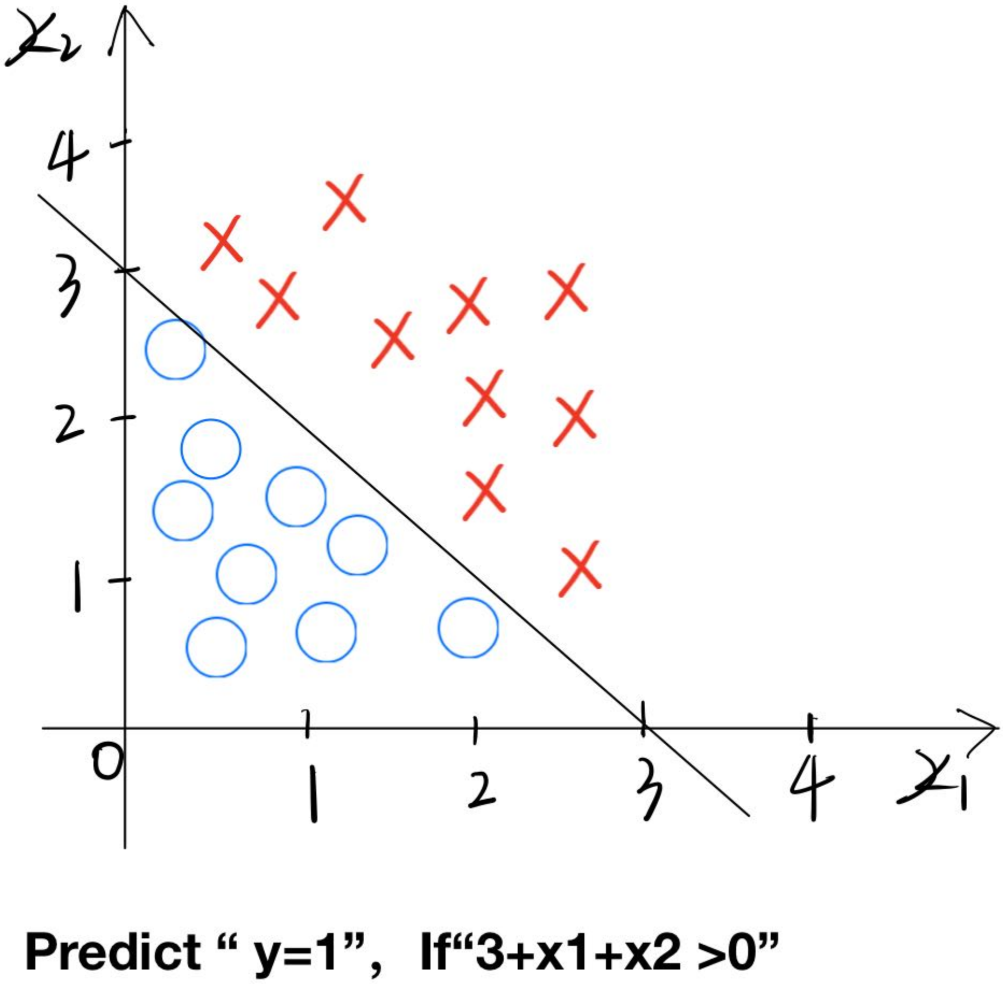
$$\begin{aligned} \text{逻辑斯蒂函数：} h_\omega(x) &= g^{-1}(\omega^T x_i + b) \\ &= \frac{1}{1+e^{-(w^T x_i + b)}} \end{aligned}$$

它的图像如下：



逻辑斯谛分布的密度函数与分布函数

因为Logistic 回归主要用于分类问题，以二分类为例，对于所给数据集假设存在这样的一条直线可以将数据完成线性可分。



当我们要找到分类概率 P(Y=1) 与输入向量 x 的直接关系时，我们引入Sigmoid函数，然后通过比较概率值来判断类别。

引入sigmoid函数具体实现如下：

但因为逻辑斯蒂函数的值域在[0,1]之间，无法直接输出0或1。在此基础上，考虑到$\omega^T x + b$取值是连续的，因此它不能拟合离散变量。可以考虑用它来拟合条件概率$p(y=1|x)$，因为概率的取值也是连续的,我们将逻辑斯蒂函数的输出作为输入x能预测到y为1的概率，并利用对数几率函数，得到下面三个式子。通过此方法，我们将线性模型转换为概率模型。

对数几率函数：
$$logit(p) = ln\frac{p}{1-p}$$

令
$$ln\frac{p(y=1|x)}{1-p(y=1|x)} = \omega^T x + b$$

推导：
$$\frac{p(y=1|x)}{1-p(y=1|x)} = e^{\omega^T x+b}$$
$$p(y=1|x)(1+e^{\omega^T x+b}) = e^{\omega^T x+b}$$
$$p(y=1|x) = \frac{1}{1+e^{-(\omega^T x+b)}}$$

所以我们的模型是：
$$p(y=1|x) = \frac{1}{1+e^{-(\omega^T x+b)}}$$
$$p(y=0|x) = \frac{e^{-(\omega^T x+b)}}{1+e^{-(\omega^T x+b)}}$$

1.3策略

在策略上，我们采用极大似然法。即选择最优的w，b使得我们输入x得到的正确的y的概率最大，即下式：

$$(w^*,b^*)=\underset{(w,b)}{argmax}\prod_{i=1}^{N}p(y_i|x_i;\omega,b)$$

我们这里做一点变换：

$$\begin{aligned}p(y|x,\omega)&=P(y=1|x,\omega)^y(1-P(y=0|x,\omega))^{(1-y)}\\&=(h_\omega(x))^y(1-h_\omega(x))^{(1-y)}\end{aligned}$$

因为上式是连乘的函数，我们通过对数似然函数将之转化为求和，即下式：

$$\begin{aligned}(w^*,b^*)&=\underset{(w,b)}{argmin}\sum_{i=1}^{N}-lnp(y_i|x_i;\omega,b)\\&=\underset{(w,b)}{argmax}\sum_{i=1}^{N}lnp(y_i|x_i;\omega,b)\\&=\underset{(w,b)}{argmax}\sum_{i=1}^{N}ln(P(y_i=1|x_i)_i^y(1-P(y_i=0|x_i))^{(1-y_i)})\\&=\underset{(w,b)}{argmax}\quad ln(\prod_{i=1}^{N}[P(y_i=1|x_i)_i^y(1-P(y_i=0|x_i))^{(1-y_i)}])\\&=\underset{(w,b)}{argmax}\sum_{i=1}^{N}[y_ilnP(y_i=1|x_i)+(1-y_i)ln(1-P(y_i=0|x_i))]\\&=\underset{(w,b)}{argmax}\sum_{i=1}^{N}[y_iln\frac{1}{1+e^{-(w^Tx+b)}}+(1-y_i)ln\frac{e^{-(w^Tx+b)}}{1+e^{-(w^Tx+b)}}]\\&=\underset{(w,b)}{argmax}\sum_{i=1}^{N}[y_i(w^Tx_i+b)-ln(1+e^{w^Tx+b}))+(1-y_i)(-ln(1+e^{w^Tx+b}))]\\&=\underset{(\omega,b)}{argmax}\sum_{i=1}^{N}[-ln(1+e^{\omega^T+b})+y_i(\omega^Tx_i+b)]\\&=\underset{(\omega,b)}{argmin}\sum_{i=1}^{N}[-y_i(\omega^Tx_i+b)+ln(1+e^{\omega^Tx_i+b})]\end{aligned}$$

为了方便计算，我们做以下处理

$$assume\ that\ \hat{\omega}=(\omega;b),\hat{x}=(x;1)$$

则上式可化为

$$\hat{\omega^*}=\underset{\hat{\omega}}{argmin}\sum_{i=1}^{N}(-y_i\hat{\omega}x_i+ln(1+e^{\hat{\omega}^T\hat{x}_i}))$$

这是一个凸函数，可用经典的数值优化算法，如梯度下降法、牛顿法求解。

最终，我们学得的逻辑斯蒂回归模型为

$$\hat{p}(y_{N+1}=1|x_{N+1})=\frac{1}{1+e^{-\hat{x}_{N+1}^T\hat{\omega}^*}}$$
$$\hat{p}(y_{N+1}=0|x_{N+1})=\frac{1}{1+e^{\hat{x}_{N+1}^T\hat{\omega}^*}}$$
$$\text{其中，}x_{\hat{N}+1}=(x_{N+1};1)\in\mathbb{R}^{n+1},\hat{\omega}^*=(\omega^*;b)\in\mathbb{R}^{n+1}$$

相关代码如下：

```
#logistic function
def sigmode(z):
    return 1 / (1+ np.exp(-z))

#logistic regression
def model(X, Omega):
    """
    X: 样本矩阵 （包括第一列的1）
    Omega : 行向量 [w0， w1, ....wn]
    返回预测的结果值
    :return:
    """
    return sigmode(np.dot(X, Omega.T))
#maximum likelihood
def cost(X, y, Omega):
    """
    获取样本的损失值，为所有样本的损失均值
    """
    # 注意，multipy和 np.dot的区别
    left = np.multiply(-y, np.log(model(X,Omega)))
    right = np.multiply(1-y, np.log(1 - model(X,Omega)))
    return np.sum(left-right) / len(X)

def gradient(X,y,Omega):
    """
    计算梯度，返回一个和Omega 一样的数组
    :return:
    """
    #二维数组，1*n
    grad = np.zeros(shape=Omega.shape)
    # error n*1
    error = y- model(X,Omega)
    for j in range(grad.shape[1]):
        # grad 1*n ,
        grad[0, j] = - np.dot(error.T, X[:,j]) / X.shape[0]
    return grad
```

二、描述训练模型所使用的算法

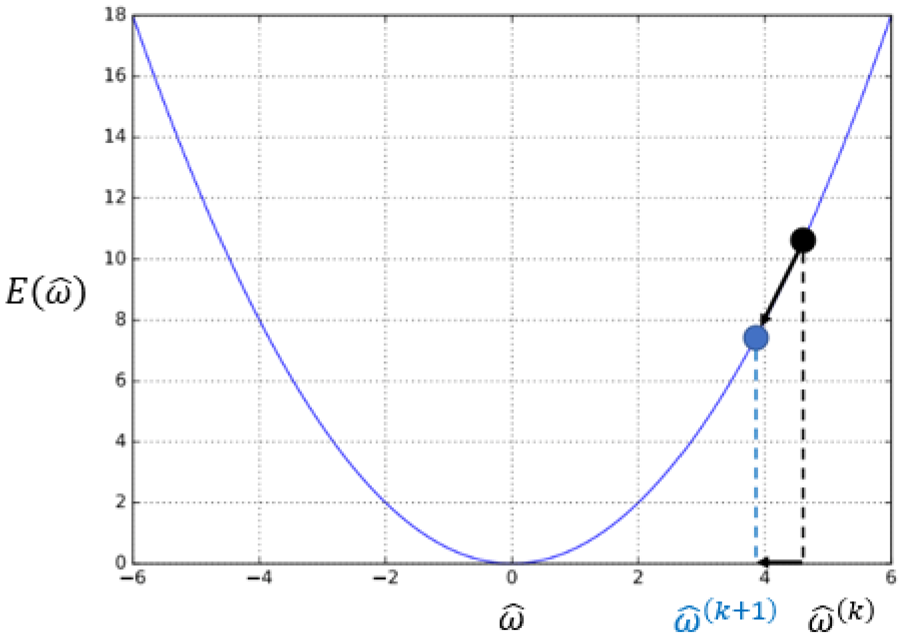
2.1梯度下降法

2.1.1问题分析

首先，我们的目标是下式

$$E(\hat{\omega})=\sum_{i=1}^{N}(-y_i\hat{\omega}^T\hat{x}_i+ln(1+e^{\hat{\omega}^T\hat{x}_i})),\hat{\omega}^*=\underset{\hat{\omega}}{argmin}E(\hat{\omega})$$

梯度下降法是一种迭代算法：我们选取适当的初始值$\hat{\omega}^{(0)}$，不断迭代，更新$\hat{\omega}$的值，进行目标函数的极小化，直到收敛。由于负梯度方向是使得函数值下降最快的方向，在迭代的每一步，以负梯度方向更新$\hat{\omega}$的值，从而达到减小函数值的目的。如下图形象化表示：



$$\widehat{\omega}^{(k+1)} \leftarrow \widehat{\omega}^{(k)} - \eta \times \frac{\partial E(\widehat{\omega})}{\partial \widehat{\omega}}\Bigg|_{\widehat{\omega}=\widehat{\omega}^{(k)}}$$

2.1.2核心思想：

1. $E(\hat{\omega})$是具有一阶连续偏导数的凸函数，其极值点在一阶导数为零的地方取得
2. 一阶泰勒展开：$E(\hat{\omega}) \approx E(\hat{\omega}^{(k)}) + \nabla E(\hat{\omega}^{(k)})(\hat{\omega} - \hat{\omega}^{(k)})$，其中，$\nabla E(\hat{\omega}^{(k)})$是$E(\hat{\omega})$在$\hat{\omega}^{(k)}$的梯度：

$$\nabla E(\hat{\omega}^{(k)}) = \frac{\partial E(\hat{\omega})}{\partial \hat{\omega}}|_{\hat{\omega}=\hat{\omega}^{(k)}}$$

3. 求取第k+1次迭代值：$\hat{\omega}^{k+1} = \hat{\omega}^{(k)} + \eta_k * (-\nabla E(\hat{\omega}^{(k)}))$，其中$\eta_k$是步长，由我们最初指定。梯度推导：

$$E(\hat{\omega}) = \sum_{i=1}^{N}(-y_i\hat{\omega}^T\hat{x}_i + ln(1 + e^{\hat{\omega}^T\hat{x}_i}))$$

$$\nabla E(\hat{\omega}^{(k)}) = \sum_{i=1}^{N} -y_i\hat{x}_i + \frac{1}{1 + e^{\hat{\omega}^T\hat{x}_i}} * e^{\hat{\omega}^T\hat{x}_i} * \hat{x}_i$$

$$= -\sum_{i=1}^{N} x_i(y_i - \frac{e^{\hat{\omega}^T\hat{x}_i}}{1 + e^{\hat{\omega}^T\hat{x}_i}})$$

2.1.3伪代码：

输入：目标函数$E(\hat{\omega})$，梯度函数$\nabla E(\hat{\omega})$，计算精度ε，步长η_k；

输出：$E(\hat{\omega})$的极小点$\hat{\omega}^*$。

（1）取初始值$\hat{\omega}^{(0)} \in \mathbb{R}^{d+1}$，置k=0；

（2）计算$E(\hat{\omega}^{(k)})$；

(3) 计算梯度$\nabla E(\hat{\omega}^{(k)})$，当$||\nabla E(\hat{\omega}^{(k)})|| < \varepsilon$时，令$\hat{\omega}^* = \hat{\omega}^{(k)}$，

停止迭代；

(4) 置$\hat{\omega}^{(k+1)} = \hat{\omega}^{(k)} + \eta_k(-\nabla E(\hat{\omega}^{(k)}))$，计算$E(\hat{\omega}^{(k+1)})$，

当$||E(\hat{\omega}^{(k+1)}) - E(\hat{\omega}^{(k)})|| < \varepsilon$或$||\hat{\omega}^{(k+1)} - \hat{\omega}^{(k)}|| < \varepsilon$时，

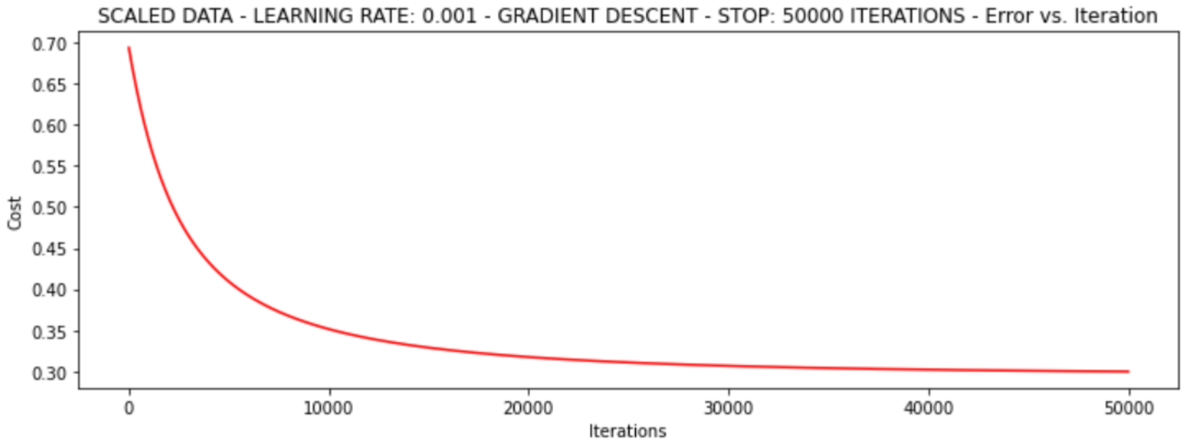
令$\hat{\omega}^* = \hat{\omega}^{(k)}$，停止迭代；

(5) 否则，置k=k+1，转步骤（3）。

2.1.4分析

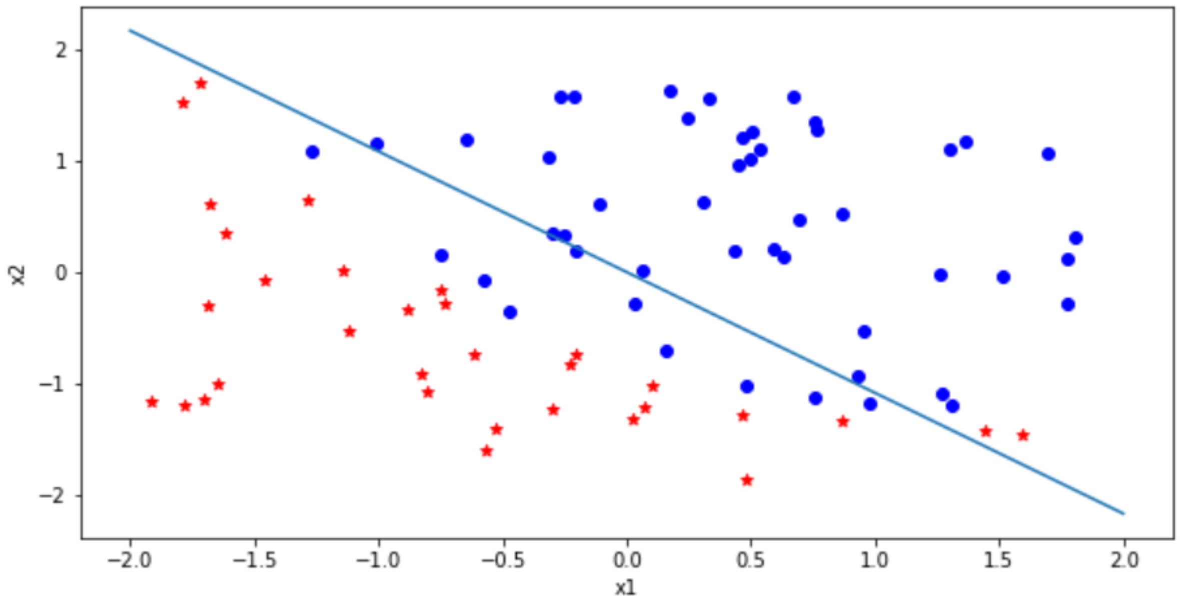
优点：方法简单，易理解

缺点：迭代次数多，下降速度慢，如下图，我们采用梯度下降法，迭代近50000次才收敛



且准确率如下，可以看出准确率不高。

```
准确率为:  0.85
```



2.2牛顿法

2.2.1核心思想：

$E(\hat{\omega})$是具有二阶连续偏导数的函数

二阶泰勒展开：$E(\hat{\omega}) \approx E(\hat{\omega}^{(k)}) + \nabla E(\hat{\omega}^{(k)})(\hat{\omega} - \hat{\omega}^{(k)}) + \frac{1}{2}(\hat{\omega} - \hat{\omega}^{(k)})^T H(\hat{\omega}^{(k)})(\hat{\omega} - \hat{\omega}^{(k)})$

$$\nabla E(\hat{\omega}) = \frac{\partial E(\hat{\omega})}{\partial \hat{\omega}}|_{(d+1)\times 1}, H(\hat{\omega}) = \frac{\partial^2 E(\hat{\omega})}{\partial \hat{\omega}_i \partial \hat{\omega}_j}|_{(d+1)\times 1}$$

利用二阶泰勒展开$E(\hat{\omega})$取极小点的必要条件$\nabla E(\hat{\omega}) = 0$，在第k次迭代$\hat{\omega}^{(k)}$，求$\nabla E(\hat{\omega}^{(k)}) + H(\hat{\omega}^{(k)})(\hat{\omega} - \hat{\omega}^{(k)}) = 0$的点，作为第k+1次迭代值$\hat{\omega}^{(k+1)}$

2.2.2伪代码

输入：目标函数$E(\hat{\omega})$，梯度函数$\nabla E(\hat{\omega})$，海森矩阵$H(\hat{\omega})$，精度ε；

输出：$E(\hat{\omega})$的极小点$\hat{\omega}^*$。

（1）取初始值$\hat{\omega}^{(0)} \in \mathbb{R}^{n+1}$，置k=0；

（2）计算梯度$\nabla E(\hat{\omega}^{(k)})$；

（3）当$||E(\hat{\omega}^{(k)})|| < \varepsilon$时，令$\hat{\omega}^* = \hat{\omega}^{(k)}$，停止迭代；

否则，计算海森矩阵$H(\hat{\omega}^{(k)})$；

（4）置$\hat{\omega}^{(k+1)} = \hat{\omega}^{(k)} - (H(\hat{\omega}))^{(-1)} \nabla E(\hat{\omega}^{(k)})$；

（5）置k=k+1，转步骤（2）。

2.2.3分析

牛顿法优点：下降速度快，属于二次收敛

缺点：海森矩阵计算复杂度高，且要求可逆才能计算，所以我们查阅资料，将采用拟牛顿法。

2.3 BFGS算法：

由于上述牛顿公式中可以看出，我们的海森矩阵不易得到，因此我们有以下迭代公式来逼近海森矩阵：

$$H_{k+1} = H_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{H_k s_k s_k^T H_k^T}{s_k^T H_k^T s_k}$$

但计算量还是很大，矩阵相乘太多。所以我们最终采取$Sherman - Morrison$公式进行变换可得：

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k} \quad (1)$$

公式推导如下：

Sherman Morrison 公式:

$$\left(\mathrm{A}+\frac{uu^T}{t}\right)^{-1}=A^{-1}-\frac{A^{-1}uu^TA^{-1}}{t+u^TA^{-1}u}$$

$$\left(\mathrm{H}+\frac{yy^T}{y^T\mathrm{~s}}-\frac{Hss^T\mathrm{H}}{s^THs}\right)^{-1}$$

$$=\left(\mathrm{H}+\frac{yy^T}{y^T\mathrm{~s}}\right)^{-1}+\left(\mathrm{H}+\frac{yy^T}{y^T\mathrm{~s}}\right)^{-1}\frac{Hss^TH}{s^TH^Ts-s^TH\left(\mathrm{H}+\frac{yy^T}{y^T\mathrm{~S}}\right)^{-1}\mathrm{Hs}}\left(\mathrm{H}+\frac{yy^T}{y^T\mathrm{~S}}\right)^{-1}$$

$$=\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^TS+y^TH^{-1}y}\right)+\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^TS+y^TH^{-1}y}\right)\frac{Hss^TH}{s^THS-s^TH\left(H^{-1}-\frac{H^{-1}yy^{T-1}}{y^Ts+y^TH^{-1}y}\right)Hs}\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^TS+y^TH^{-1}y}\right)$$

$$=\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^T\mathrm{~s}+y^TH^{-1}y}\right)+\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^T\mathrm{~s}+y^TH^{-1}y}\right)\frac{Hss^TH}{\frac{s^Tyy^Ts}{y^T\mathrm{~s}+y^TH^{-1}y}}\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^T\mathrm{~S}+y^TH^{-1}y}\right)$$

$$=\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^T\mathrm{~s}+y^TH^{-1}y}\right)+\frac{H^{-1}Hss^THH^{-1}}{\frac{s^Tyy^TS}{y^Ts+y^TH^{-1}y}}-\frac{H^{-1}Hss^TH}{\frac{s^Tyy^Ts}{}}y^T\mathrm{~s}+y^TH^{-1}yH^{-1}\frac{yy^T}{y^T\mathrm{~s}+y^TH^{-1}y}H^{-1}$$

$$-\frac{H^{-1}yy^TH^{-1}}{y^T\mathrm{~s}+y^TH^{-1}y}\frac{Hss^TH}{\frac{s^Tyy^TS}{y^T\mathrm{~s}+y^TH^{-1}y}}H^{-1}$$

$$+H^{-1}\frac{yy^T}{y^T\mathrm{~s}+y^TH^{-1}y}H^{-1}\frac{Hss^TH}{\frac{s^Tyy^TS}{y^T\mathrm{~s}+y^TH^{-1}y}}H^{-1}\frac{yy^T}{y^T\mathrm{~s}+y^TH^{-1}y}H^{-1}$$

$$=\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^T\mathrm{~s}+y^TH^{-1}y}\right)+\frac{ss^T\left(y^T\mathrm{~s}+y^TH^{-1}y\right)}{s^Tyy^Ts}-\frac{ss^Tyy^TH^{-1}}{s^Tyy^TS}-\frac{H^{-1}yy^TSS^T}{s^Tyy^TS}$$

$$+\frac{H^{-1}yy^TSs^Tyy^TH^{-1}}{(y^T\mathrm{~s}+y^TH^{-1}y)s^Tyy^TS}$$

$$=\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^T\mathrm{~s}+y^TH^{-1}y}\right)+\frac{ss^T\left(y^T\mathrm{~s}+y^TH^{-1}y\right)}{\left(s^Ty\right)^2}-\frac{s\left(s^Ty\right)y^TH^{-1}}{\left(s^Ty\right)^2}-\frac{H^{-1}y\left(y^Ts\right)s^T}{\left(s^Ty\right)^2}$$

$$+\frac{H^{-1}y\left(y^TSs^Ty\right)y^TH^{-1}}{(y^T\mathrm{~s}+y^TH^{-1}y)s^Tyy^TS}$$

$$=\left(H^{-1}-\frac{H^{-1}yy^TH^{-1}}{y^T\mathrm{~s}+y^TH^{-1}y}\right)+\frac{ss^T\left(y^T\mathrm{~s}+y^TH^{-1}y\right)}{\left(s^Ty\right)^2}-\frac{sy^TH^{-1}}{s^Ty}-\frac{H^{-1}ys^T}{s^Ty}+\frac{H^{-1}yy^TH^{-1}}{(y^T\mathrm{~s}+y^TH^{-1}y)}$$

$$=H^{-1}+\frac{ss^T\left(y^T\mathrm{~s}+y^TH^{-1}y\right)}{\left(s^Ty\right)^2}-\frac{sy^TH^{-1}}{s^Ty}-\frac{H^{-1}ys^T}{s^Ty}$$

$$=H^{-1}+\frac{ss^Ty^T\mathrm{~s}}{\left(s^Ty\right)^2}+\frac{ss^Ty^TH^{-1}y}{\left(s^Ty\right)^2}-\frac{sy^TH^{-1}}{s^Ty}-\frac{H^{-1}ys^T}{s^Ty}$$

$$=H^{-1}\left(I-\frac{ys^T}{s^Ty}\right)-\frac{sy^TH^{-1}}{s^Ty}\left(I-\frac{ys^T}{s^Ty}\right)+\frac{ss^T}{s^Ty}$$

$$=\left(I-\frac{sy^T}{s^Ty}\right)H^{-1}\left(I-\frac{ys^T}{s^Ty}\right)+\frac{ss^T}{s^Ty}$$

2.4 L-BFGS算法

在 L-BFGS 中, 不再保存完整的 H_k , 而是存储向量序列 $\{\mathrm{s}_k\},\{\mathrm{y}_k\}$,需要矩阵 H_k , 使用向量 序列 $\{\mathrm{s}_k\},\{\mathrm{y}_k\}$ 计算代替。而且向量序列也不是所有的都存, 可以只保存最新的m步向量即可。m由用户决定。每次计算 H_k 时, 只需要用最新 m 步 $\{\mathrm{s}_k\},\{\mathrm{y}_k\}$计算即可。$\mathrm{H}_k$的存储由原来的 $\mathrm{O}\left(\mathrm{N}^2\right)$ 降到 $\mathrm{O}(\mathrm{mN})$

记 $\rho_k=\frac{1}{y_k^Ts_k}, V_k=I-\rho_ky_ks_k^T$，则(1)可写成:

$$H_{k+1}=V_k^TH_kV_k+\rho_ks_ks_k^T$$

给定初始矩阵$H_0=I$，利用上式，可得:

$$
\begin{aligned}
H_1 &= V_0^T H_0 V_0 + \rho_0 s_0 s_0^T \\
H_2 &= V_1^T H_1 V_1 + \rho_1 s_1 s_1^T \\
&= V_1^T \left(V_0^T H_0 V_0 + \rho_0 s_0 s_0^T\right) V_1 + \rho_1 s_1 s_1^T \\
&= V_1^T V_0^T H_0 V_0 V_1 + V_1^T \rho_0 s_0 s_0^T\right) V_1 + \rho_1 s_1 s_1^T
\end{aligned}
$$

$$
\begin{aligned}
H_{k+1} = &\left(V_k^T V_{k-1}^T \ldots V_1^T V_0^T\right) H_0 \left(V_0 V_1 \ldots V_{k-1} V_k\right) \\
&+ \left(V_k^T V_{k-1}^T \ldots V_1^T\right) \rho_1 s_1 s_1^T \left(V_1 \ldots V_{k-1} V_k\right) \\
&+ \ldots \\
&+ \left(V_k^T\right) \rho_{k-1} s_{k-1} s_{k-1}^T \left(V_k\right) \\
&+ \rho_k s_k s_k^T
\end{aligned}
$$

只保留最近的m步后，上式的迭代公式变为：

$$
\begin{aligned}
H_{k+1} = &\left(V_k^T V_{k-1}^T \ldots V_{k-m}^T\right) H_0 \left(V_{k-m} \ldots V_{k-1} V_k\right) \\
&+ \left(V_k^T V_{k-1}^T \ldots V_{k-m+1}^T\right) \rho_{k-m} s_{k-m} s_{k-m}^T \left(V_{k-m+1} \ldots V_{k-1} V_k\right) \\
&+ \ldots \\
&+ \left(V_k^T\right) \rho_{k-1} s_{k-1} s_{k-1}^T \left(V_k\right) \\
&+ \rho_k s_k s_k^T
\end{aligned}
$$

所求方向为：

$$
\begin{aligned}
H_k \nabla f = &\left(V_{K-1}^T V_{K-2}^T \ldots V_{K-m}^T\right) H_0 \left(V_{K-m} V_{K-m+1} \ldots V_{K-1}\right) \nabla f \\
&+ \left(V_{K-1}^T \ldots V_{K-m+1}^T\right) \rho_{k-m} s_{k-m} s_{k-m}^T (V_{k-m+1} \ldots V_{k-1} V_k) \nabla f \\
&+ \ldots \\
&+ V_{k-1} \rho_{k-1} s_{k-1} s_{k-1}^T V_k \nabla f \\
&+ \rho_k s_k s_k^T \nabla f
\end{aligned}
$$

Two-Loop 算法：

$$
\begin{aligned}
&q_k \leftarrow \nabla f_k \\
&\text{ for } i = k-1 \text{ to } k-m \text{ do} \\
&\quad \alpha_i = \rho_i s_i^T q_{i+1} \\
&\quad q_i = q_{i+1} - \alpha_i y_i \\
&\text{ end for} \\
&r_{k-m-1} = H_0 q_{k-m} \\
&\text{ for } i = k-m, k-m+1 \text{ to } k-1 \text{ do} \\
&\quad \beta_i = \rho_i y_i^T r_{i-1} \\
&\quad r_i = r_{i-1} + s_i \alpha_i - \beta_i \\
&\text{ end for} \\
&\text{ End, The result is } H_{k+1} \nabla f = r
\end{aligned}
$$

Two-Loop算法解析---第一个循环：

初始条件：$q_k = \nabla f$

递推：$q_{k-i} = V_{k-i} q_{k-i+1}$

终止：$\qquad q_{k-m} = V_{k-m} V_{k-m+1} \ldots V_{k-1} \nabla f$

$$\alpha_{k-i} = \rho_{k-i} s_{k-i}^T V_{k-i+1} V_{k-i+2} \ldots V_{k-1} \nabla f$$

已知：$\alpha_{k-i}=\rho_{k-i}s_{k-i}^TV_{k-i+1}V_{k-i+2}\dots V_{k-1}\nabla f$

重写公式：

$$\begin{aligned}H_k\nabla f=&\left(V_{K-1}^TV_{K-2}^T\dots V_{K-m}^T\right)H_0\left(V_{K-m}V_{K-m+1}\dots V_{K-1}\right)\nabla f\\&+\left(V_{K-1}^T\dots V_{K-m+1}^T\right)s_{k-m}\alpha_{k-m}\\&+\dots\\&+V_{k-1}s_{k-1}\alpha_{k-2}\\&+s_{k-1}\alpha_{k-1}\end{aligned}$$

Two-Loop算法解析---第二个循环：

已知：$q_{k-m}=V_{k-m}V_{k-m+1}\dots V_{k-1}\nabla f,\ \ H_{k+1}=V_k^TH_kV_k+\rho_ks_ks_kT$

递推：

$$\begin{aligned}r_{k-\mathrm{m+i+1}}&=r_{k-\mathrm{m+i}}+\mathrm{s}_{k-\mathrm{m+i+1}}\left(\alpha_{k-m+i+1}-\beta_{k-m+i+1}\right)\\&=r_{k-\mathrm{m+i}}+\mathrm{s}_{k-\mathrm{m+i+1}}\left(\alpha_{k-m+i+1}-\rho_{k-m+i+1}y_{k-m+i+1}^Tr_{k-m+i}\right)\\&=\left(I-\mathrm{s}_{k-\mathrm{m+i+1}}\rho_{k-m+i+1}y_{k-m+i+1}^T\right)r_{k-\mathrm{m+i}}+\mathrm{s}_{k-\mathrm{m+i+1}}\alpha_{k-m+i+1}\\&=V_{k-\mathrm{m+i+1}}r_{k-\mathrm{m+i}}+\mathrm{s}_{k-\mathrm{m+i+1}}\alpha_{k-m+i+1}\end{aligned}$$

初始：

$$r_{k-\mathrm{m}}=V_{k-\mathrm{m}}H_0V_{k-\mathrm{m}}V_{k-\mathrm{m+1}}\dots V_{k-1}\nabla\mathrm{f}+\mathrm{s}_{k-\mathrm{m}}\alpha_{k-m}$$

得：

$$\begin{aligned}r_{k-\mathrm{m+i}}=&V_{k-\mathrm{m+i}}\dots V_{k-\mathrm{m}}H_0V_{k-\mathrm{m}}\dots V_{k-\mathrm{m+i}}\nabla\mathrm{f}\\&+(V_{k-\mathrm{m+i}}\dots V_{k-\mathrm{m+1}})\mathrm{s}_{k-\mathrm{m}}\alpha_{k-m}\\&+(V_{k-\mathrm{m+i}}\dots V_{k-\mathrm{m+2}})\mathrm{s}_{k-\mathrm{m+1}}\alpha_{k-m+1}\\&+\dots\\&s_{k-m+1}\alpha_{k-m+i}\end{aligned}$$

r_{k-1}即是所求的搜索方向d。

使用LBFGS求解逻辑回归模型代码如下：

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# 进一步处理数据集和测试集，将输入和输出分割
train.columns=list(['x1','x2','y'])
test.columns=list(['x1','x2','y'])
X_train = np.asarray(train.get(['x1', 'x2']))
y_train = np.asarray(train.get('y'))
X_test = np.asarray(test.get(['x1', 'x2']))
y_test = np.asarray(test.get('y'))
# 使用 sklearn 的 LogisticRegression 作为模型
# 其中有 penalty，solver，multi_class 几个比较重要的参数，不同的参数有不同的准确率
model = LogisticRegression(solver='newton-cg')
# newton-cg sag lbfgs liblinear


# 对数据进行标准化
ss = StandardScaler()
```

```
X_train = ss.fit_transform(X_train)
X_test = ss.fit_transform(X_test)
# 拟合
model.fit(X_train, y_train)

# 预测测试集
predictions = model.predict(X_test)

# 打印准确率
print('测试集准确率：', accuracy_score(y_test, predictions))

weights = np.column_stack((model.intercept_, model.coef_)).transpose()
#print(weights)
```

三、绘制ROC曲线和PR曲线

```
该部分出现的英语缩写：
TP: True Positive
FP: False Positive
FN: False Negative
TN: True Negative
P: Precision
R: Recall
TPR: True Positive Rate
FPR: False Positive Rate
```

3.1 ROC曲线

3.1.1介绍

ROC全称是“受试者工作特征”(Receiver Operating Characteristic)曲线，它源于“二战”中用于敌机检测的雷达信号分析技术，二十世纪六七十年代开始被用于一些心理学、医学检测应用中，此后被引入机器学习领域，用来评判分类、检测结果的好坏。因此，ROC曲线是非常重要和常见的统计分析方法。

为了绘制ROC曲线，我们需要计算出两个重要量的值（**TPR**、**FPR**），分别以它们为横、纵坐标作图。其中的TP、FP、TN、FN来自于**混淆矩阵**，且TP+FP+TN+FN=样本总数。

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

3.1.2画图流程

1. 给定m^+个正例和m^-个负例，根据学习器预测结果对样例进行排序
2. 然后把分类阈值设为最大，即把所有样例均预测为反例，此时真正例率和假正例率均为0，在坐标(0,0)处标记一个点
3. 将分类阈值依次设为每个样例的预测值，即依次将每个样例划分为正例，设前一个标记点坐标为(x,y)，当前若为真正例，则对应标记点的坐标为$\left(x, y+\frac{1}{m^+}\right)$；当前若为假正例，则对应标记点的坐标为$\left(x+\frac{1}{m^-}, y\right)$
4. 最后用线段连接相邻点

3.1.3 AUC分析

ROC曲线下方的面积也有着重要意义（英语：Area under the Curve of ROC (AUC ROC)），其意义是：

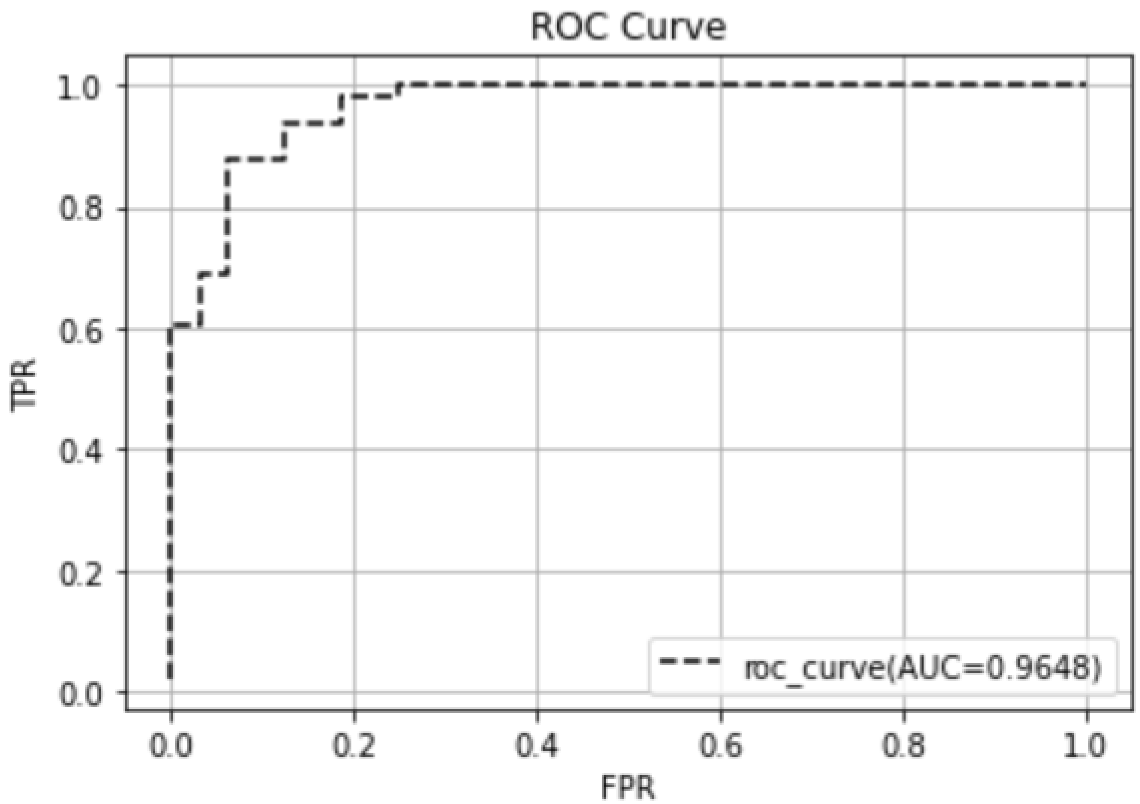
- 因为是在1x1的方格里求面积，AUC必在0~1之间。
- 假设阈值以上是正例，以下是反例；
- 简单说：**AUC值越大的分类器，正确率越高。**

从AUC判断分类器（预测模型）优劣的标准：

- AUC = 1，是完美分类器，采用这个预测模型时，存在至少一个阈值能得出完美预测。绝大多数预测的场合，不存在完美分类器。
- 0.5 < AUC < 1，优于随机猜测。这个分类器（模型）妥善设置阈值的话，能有预测价值。
- AUC = 0.5，跟随机猜测一样（例：丢铜板），模型没有预测价值。
- AUC < 0.5，比随机猜测还差；但只要总是反预测而行，就优于随机猜测。

假设ROC曲线由为$\{(x_1,y_1),\cdots,(x_{N'},y_{N'})\}$的点按需连接而成且有$x_1=0, x_{N'}=1$，则AUC可估算为：

$$AUC = \frac{1}{2}\sum_{j=1}^{N'-1}(x_{j+1}-x_j)(y_{j+1}+y_j)$$



如图即为使用本次作业所提供数据绘制的ROC曲线。由于测试样例有限，所以仅能获得有限个（真正例率，假正例率）坐标对，无法产生光滑的ROC曲线；由此计算得到的AUC的值为0.9648，可以得知该模型的性能较优。

完整代码如下：

```
def draw_roc(confidence_scores, data_labels):
    #真正率，假正率
    fpr, tpr, thresholds = roc_curve(data_labels, confidence_scores)
    plt.figure()
    plt.grid()
    plt.title('ROC Curve')
    plt.xlabel('FPR')
    plt.ylabel('TPR')

    from sklearn.metrics import auc
    auc=auc(fpr, tpr) #AUC计算
    plt.plot(fpr,tpr,'k--', label = 'roc_curve(AUC=%0.4f)' % auc)
    plt.legend()
    plt.show()
```

3.2 PR曲线

3.2.1介绍

PR曲线全称为查准率-查全率曲线，查准率P与查全率R分别定义为：

$$P=\frac{TP}{TP+FP}, \quad R=\frac{TP}{TP+FN}$$

查准率和查全率是一对矛盾的度量。一般来说，查准率高时，查全率往往偏低；而查全率高时，查准率往往偏低。

3.2.2画图流程

绘制PR曲线的流程与ROC曲线类似，我们需要根据学习器的预测结果按正例可能性大小对样例进行排序，再逐个样本的选择阈值，在该样本之前的都属于正例，该样本之后的都属于负例。每一个样本作为划分阈值时，都可以计算对应的precision和recall，那么就可以以此绘制曲线。

3.2.3 AP分析

其中平衡点是曲线上"查准率=查全率"时的取值，可用于度量PR曲线有交叉的分类器性能高低。与AUC类似，PR曲线下方面积也有重要意义。PR曲线下的面积称之为AP(Average Precision)，通常来说一个越好的分类器，AP值越高。

对于连续的PR曲线，有：

$$AP=\int_0^1 p(r)\mathrm{d}r$$

但由于曲线可能出现不可导的部分，故我们常常求其近似值：

$$p_{\text{interp}}(r)=\max_{\tilde{r}\geq r} p(\tilde{r})$$

对于离散的PR曲线，有：

$$\mathrm{AP}=\sum_{k=1}^{n}p(k)\Delta r(k)$$

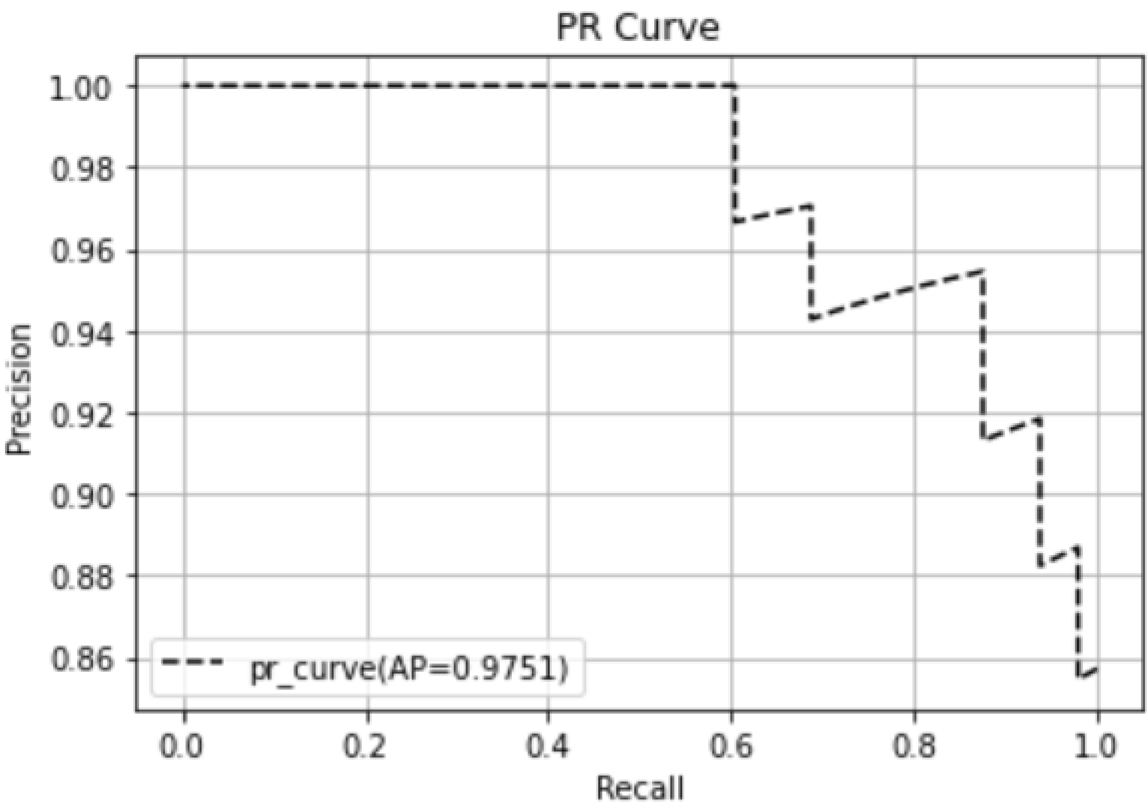
另外PR曲线平衡点更用常用的是F1度量：

$$F1=\frac{2\times P\times R}{P+R}=\frac{2\times TP}{\text{样例总数}\ +TP-TN}$$

比F1度量更一般的形式是F_β：

$$F_\beta=\frac{\left(1+\beta^2\right)\times P\times R}{\left(\beta^2\times P\right)+R}$$

- β=1：标准F1
- β>1：偏重查全率（逃犯信息检索）
- β<1：偏重查准率（商品推荐系统）



如图即为使用本次作业所提供数据绘制的PR曲线。在现实任务中，PR曲线是非单调、不平滑的，在很多局部有上下波动；由此计算得到的AP的值为0.9751，可以得知该模型的性能较优。

完整代码如下：

```
def draw_pr(confidence_scores, data_labels):
    plt.figure()
    plt.title('PR Curve')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.grid()

    #精确率，召回率，阈值
    precision,recall,thresholds =
precision_recall_curve(data_labels,confidence_scores)

    from sklearn.metrics import average_precision_score
```

```
    AP = average_precision_score(data_labels, confidence_scores) # 计算AP
    plt.plot(recall, precision,'k--', label = 'pr_curve(AP=%0.4f)' % AP)
    plt.legend()
    plt.show()
```

四、总结模型训练过程中的收获

4.1加深了对逻辑斯蒂回归的理解

4.1.1简述对模型的理解：

因为线性回归模型产生的预测值是一系列实值。为了使得输出的预测结果变成分类所需的0和1，我们需要在线性回归的基础式子外再套一个函数将其输出变成0和1，又要求该函数单调可微，所以我们引入logistic函数，将输出的预测结果成功转为概率值。这样，逻辑斯蒂回归模型被成功应用于解决分类模型。

4.1.2关于算法的择优：

在代码实现过程中，我们最开始使用的是梯度下降法，但是迭代速度较慢，拟合效果不是很好；之后我们选择了牛顿法，但是因为计算海森矩阵的复杂度太高，我们选择用一种拟牛顿法——‘L-BFGS’来逼近海森矩阵，最终达到了我们理想的效果。

梯度下降法和牛顿法/拟牛顿法相比，两者都是迭代求解，不过梯度下降法是梯度求解，而牛顿法/拟牛顿法是用二阶的海森矩阵的逆矩阵或伪逆矩阵求解。相对而言，使用牛顿法/拟牛顿法收敛更快。

4.2实现了代码技能的提升

在代码实现过程中，我们调用了机器学习工具包sklearn中的重要函数——LogisticRegression函数，熟悉了它的常用参数及意义，下面以表格形式列出我们在此次模型训练中使用到的参数。

参数	意义	备注
penalty	str类型，可选项有{‘L1’,‘L2’}，用来确定惩罚项的规范。‘newton-cg’，‘sag’和‘lbfgs’仅支持‘L2’惩罚项。	该参数是为了添加惩罚项避免过拟合，用以提高函数的泛化能力。我们在本次模型训练中使用的是‘L2’。
solver	可选的优化算法有{‘newton-cg’，‘lbfgs’,‘liblinear’,‘sag’}	小数据集中，liblinear是一个好选择，sag和saga对大数据更快； 多分类问题中，除了liblinear其它四种算法都可以使用；newton-cg，lbfgs和sag仅能使用L2惩罚项； 我们经过对比，选择的算法是lbfgs。
multi_class	str类型，可选参数有{‘ovr’，‘multinomial’} 如果是二元分类问题则两个选项一样，如果是多元分类则ovr将进行多次二分类，分别为一类别和剩余其它所有类别; multinomial则分别进行两两分类，需要T(T-1)/2次分类。	在多分类中，ovr快，精度低; multinomial慢，精度高。

4.3提高了公式推导和文章排版能力

报告中的所有公式，我们都脚踏实地，一步步手动推导，并学习使用latex将其手动输入并排版。在这个过程中，我们对算法中公式的来源更加清楚，对其原理理解更加深透。这提高了我们的公式推导能力和文章排版能力。

4.4锻炼了小组合作精神，提高了小组合作能力

在正式写报告之前，我们对本次作业任务以及对逻辑斯蒂回归模型的理解进行了讨论；然后为了加深彼此对知识的掌握程度，每个人都对代码进行了独立编写，在实现的过程中探讨互助；最后，我们根据彼此的优势项对任务进行了分工合作，齐心协力创作出了这份尽可能完善的报告。