逻辑回归

笔记本:	pytorch		
创建时间:	2023/3/26 21:06	**更新时间:**	2023/3/26 21:20
作者:	22qdnlbn		
URL:	about:blank		

1、逻辑回归用来解决2分类问题

2、一般而言的回归问题，是定量的即我们是根据x来输出其y值
但是分类问题，是离散的，定性的即我们不是为了输出其具体的y值，而是根据其概率来输出其是否属于这一类别。

比如2分类问题，有0和1两类，在逻辑回归中 概率都是指类别为1的概率，当该概率大于0.5时，我们就确定其属于1类别。

因此我们要将其输出确定在0-1内，我们使用逻辑函数一般都称为sigmoid函数

3、
回归问题的损失函数与分类问题不同。
回归问题中我们只用比较实际y值与计算y值之间的差值即可。但是在分类问题中，我们并不是看其差值，而是看其在该类上的概率是否接近。

该概率论中我们一般使用kn散度和交叉熵 cross-entropy

4、
交叉熵越小即我们所得到的模型越好。

在pytorch中使用torch.BCELoss即可

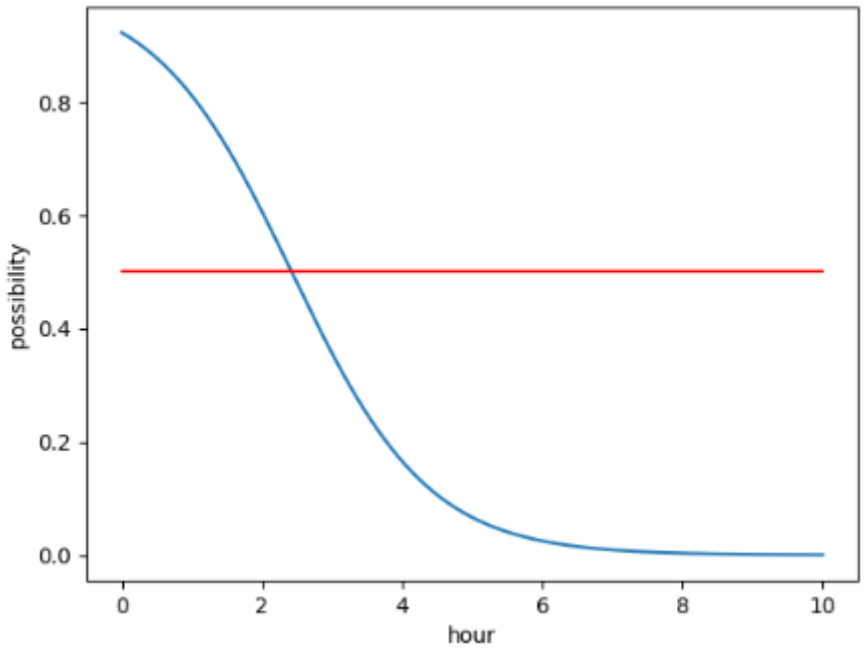
Loss Function for Binary Classification

$$loss = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

我们可以画出log图像 就知道为什么要加负号了，因为loss是越小越好。
5.代码：

```
# 时间：2023/3/25 21:52
# cky
import torch
import numpy as np
import matplotlib.pyplot as plt
from torch import nn as nn
#prepare dataset
x_data=torch.Tensor([[1.0],[2.0],[3.0]])
y_data=torch.Tensor([[1],[1],[0]])


#design modeul using class
class LinearModel(nn.Module):
    def __init__(self):
        super(LinearModel,self).__init__() #just do it
        self.linear=nn.Linear(1,1)  #in_put_size(int),out_put_size(int),bais(bool)
        #y^=wx+b
```

```
        #nn.Linear 也是继承自modeul，其对象被调用时，如forward中的第一句，就会执行
nn.linear的forward，
        #一个计算图将生成，
    def forward(self,x):
        y_hat=torch.sigmoid(self.linear(x))
        return y_hat


#继承自modeul 都有__call__ 即当继承自nn.Modeul的对象被调用时会自动执行forward函数
model=LinearModel() #实例化一个对象


#construct loss and optimizer
l=nn.BCELoss(reduction='mean')  #size_average=None, reduce=None
#size_average 即我们求得的损失是否需要除以样本个数 是否求平均损失
#reduce 即我们求得的loss 其实是一个张量向量，如果我们需要将其转为一个值，即求sum，就需要
为true
#也是继承自nn.modeul
optim=torch.optim.SGD(model.parameters(),lr=0.01)#params: _params_t, lr: float,
momentum: float=..., dampening: float=..., weight_decay:float=...,
nesterov:bool=...) -> None


#traing cycle
epoch_list=[]
loss_list=[]
for epoch in range(3001):
    ys_hat=model(x_data) #对象被调用，会自动执行forward
    loss=l(ys_hat,y_data)
    epoch_list.append(epoch)
    loss_list.append(loss.item())
    print(epoch,loss.item())
    optim.zero_grad()  #如之前讲的，要将梯度清零
    loss.backward() #反向传播，计算梯度
    optim.step() #梯度更新
print('w:',model.linear.weight.item())
print('b:',model.linear.bias.item())
x=np.linspace(0,10,200)  #0-10之间取200个点
x_t=torch.Tensor(x).view(200,1) #转换为200行1列
y_t=model(x_t)
y=y_t.data.numpy() #将y_t转换为numpy数组
plt.plot(x,y)
plt.ylabel('possibility')
plt.xlabel('hour')
plt.plot([0,10],[0.5,0.5],c='r')
plt.show()
```



6、至于图为什么是这样子：
我的理解：
逻辑回归用来解决线性模型，我们首先可以根据准备的数据来绘制出一条直线，
之后经过激活函数，得到模型。

我们的数据是0-10之间的连续线性值，便可以根据我们的y=wx+b得到y值，再经过激活函数，便可以得到如图所示的图像。