1.class LogisticsRegression(nn.Module)

1.1 def \_\_init\_\_(self, in\_dim)

定义模型参数w：

self.w=nn.Parameter(torch.randn(in\_dim+1,1))

定义非线性sigmoid函数：

self.sigmoid=torch.nn.Sigmoid()

1.2 def forward(self, x):

定义模型函数：f = sigmoid(<w, x>)

x=torch.cat([x,torch.ones((x.shape[0],1))],dim=1)

x=torch.matmul(x,self.w)

output=self.sigmoid(x)

输出（返回值）：每一个数据样本的输出（预测）

return output

2.class Logistics\_Model():

2.1 def \_\_init\_\_(self, in\_dim):

2.1.1输入：

输入数据的维度 in\_dim

2.1.2定义类的属性：

训练步长：

self.learning\_rate=0.01

轮数：

self.epoches=3000

逻辑斯蒂回归模型：

self.model=LogisticsRegression(in\_dim)

优化器：使用Adam算法

self.optimizer=torch.optim.Adam(self.model.parameters(), lr=self.learning\_rate)

损失函数：二分类交叉熵损失函数BCE

self.loss\_function=nn.BCELoss()

2.2def train(self, x, y):

2.2.1 训练模型

self.model.train()

2.2.2 迭代优化过程：

按照指定的轮数self.epoches进行迭代优化

for epoch in range(self.epoches):

prediction = self.model(x)

loss = self.loss\_function(prediction, y)

self.optimizer.zero\_grad()#所有参数的梯度设置为0

loss.backward()#计算梯度

self.optimizer.step()#所有参数更新一次

losses.append(loss.item())#tensor.item():把tensor变为float

if epoch % 500 == 0:

print("epoch: {}, loss is: {}".format(epoch, loss.item()))

2.2.3返回:

losses: 训练中每一轮的损失值（画图用）

2.3 def test(self, x, y):

用训练好的模型做测试

2.3.1预测x的可能标签的概率prediction

num\_correct=0

self.model.eval()

pred = self.model(x)

2.3.2根据模型结果，输出对应x的预测标签pred\_label

y\_shape=(pred.shape[0],pred.shape[1])

y1=torch.ones(y\_shape)

y0=torch.zeros(y\_shape)

pred\_label=torch.where(pred>0.5,y1,y0).view(y\_shape)

2.3.3根据预测标签与真实标签，输出准确率accuracy

num\_correct += torch.eq(pred\_label, y).sum().float().item()

accuracy=num\_correct/len(y)

print('num\_correct',num\_correct)

print('len(y)',len(y))

2.3.3返回:return pred, accuracy

prediction: 每一个测试数据的预测值

accuracy: 数值，模型的精度