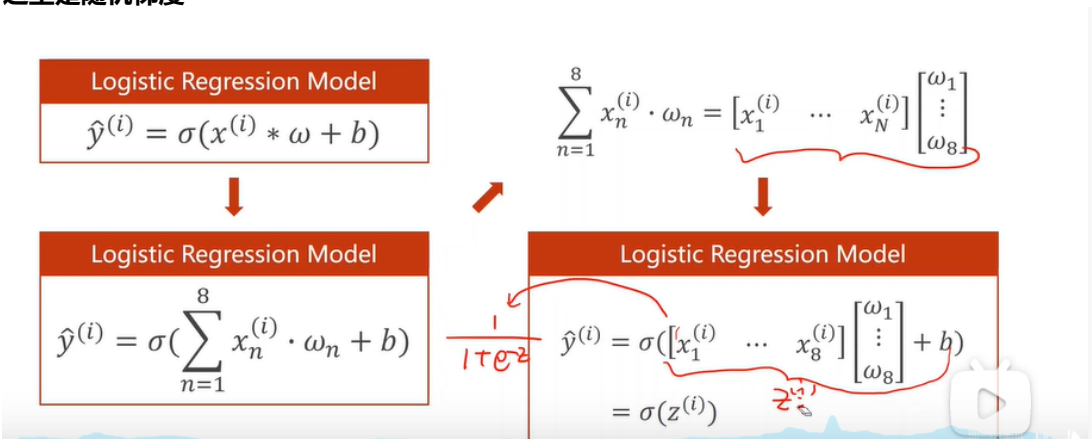
处理多维数据的输入

笔记本:	pytorch		
创建时间:	2023/3/27 20:22	**更新时间:**	2023/3/27 21:27
作者:	22qdnlbn		
URL:	about:blank		

1、在深度学习中，我们的样本特征往往是多维的。
这时候我们的输入就是一个多维数据

如果我们输入的是一个八维数据的话，那么我们的逻辑回归模型就将变成：
这里是随机梯度

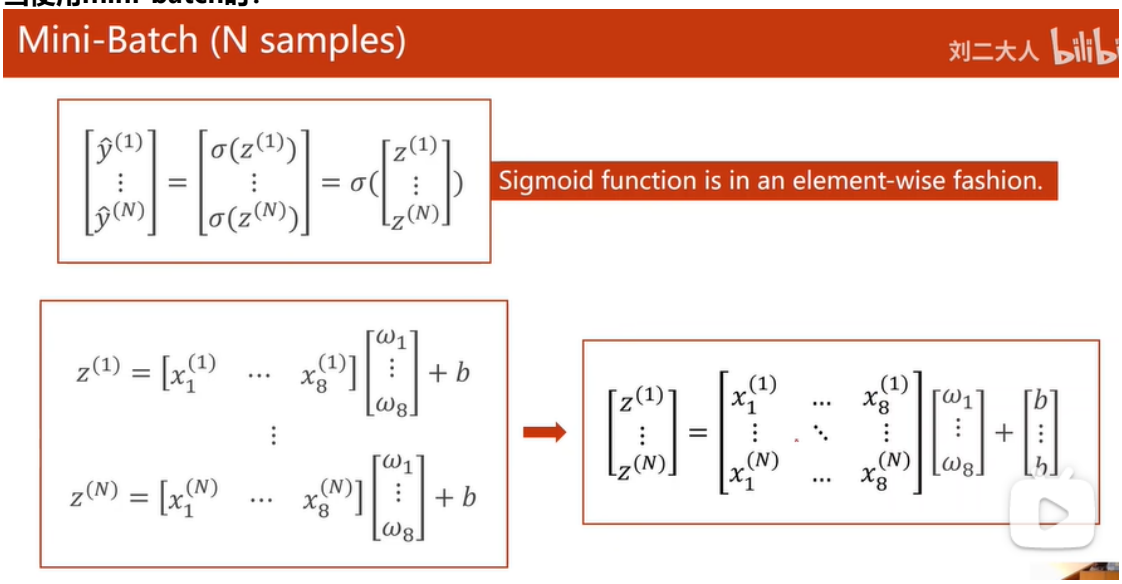


当然除了x*w+b 我们也可以写成w的转置*x

x*w+b 中x是1行8列的，8是每个样本的特征数。
w是8行1列，

w的转置 即w是一行8列，x是8列n行，此时行代表特征数，列代表样本数。

当使用mini-batch时：



x就是n行8列，w仍然是8行1列。因为是mini-batch
所以n是样本个数。

b是通过广播机制，变成n行1列。

sigmoid函数是一个逐个元素计算的函数

2、**深度学习即是一个非线性的空间变换。**

即上述图，我们把一个8维空间变成了1维空间，同时，我们使用了sigmoid函数，变成了非线性。

我们也可以把一个n维空间先升维再降维，反之亦然，同时我们也可以依次降维：如上图我们可以先变成6维，再变2维，再变1维。这样有助于我们更好的保留特征，因为在维度变化中是会损失特征的。

超参数的多少是深度学习一个很重要的东西，如果我们参数过多，可能会导致其过拟合，泛化能力不好。

3、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
xy=np.loadtxt('diabetes.csv.gz',delimiter=',',dtype=np.float32)
#delimiter 表示我们数据的分隔符
x=torch.from_numpy(xy[:,:-1])
#取所有行，前8列 即所有特征
y=torch.from_numpy(xy[:,[-1]])
#取所有行，最后一列 记得要加【】,因为我们的x，y应该是一个矩阵，而不是向量
#torch.from_numpy 即将我们的数据把numpy 变为了Tensor类型
#design modeul using class
class Model(nn.Module):
    def __init__(self):
        super(Model,self).__init__() #just do it
        #实例化 但并非使用，只有在我们对其实例化的对象调用时才会有计算图的生成
        self.linear1=nn.Linear(8,6)  #in_put_size(int),out_put_size(int),bais(bool)
        self.linear2=nn.Linear(6,2)
        self.linear3=nn.Linear(2,1)
        #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，
        #我们的激活函数
        self.relu=torch.nn.ReLU6()
        self.sigmoid=torch.nn.Sigmoid()
    def forward(self,x):
        x=self.relu(self.linear1(x))
        x=self.relu(self.linear2(x))
        x=self.sigmoid(self.linear3(x))
        # x = self.sigmoid(self.linear1(x))
        # x = self.sigmoid(self.linear2(x))
        # x = self.sigmoid(self.linear3(x))
        return x


#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=Model() #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=0.1)#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(301):
    ys_hat=model(x) #对象被调用，会自动执行forward
    loss=l(ys_hat,y)
    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
plt.plot(epoch_list,loss_list)
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```

我们可以使用不同的激活函数 来训练模型，看那种激活函数更好些。

4、如果我们使用了relu激活函数，最后一层应该使用sigmoid，因为relu激活函数，如果输入小于0，就全变为0，但是我们在计算损失的过程中，会用到ln，如果等于0的话，就会造成计算错误。

sigmoid是将其映射在（0，1）之间