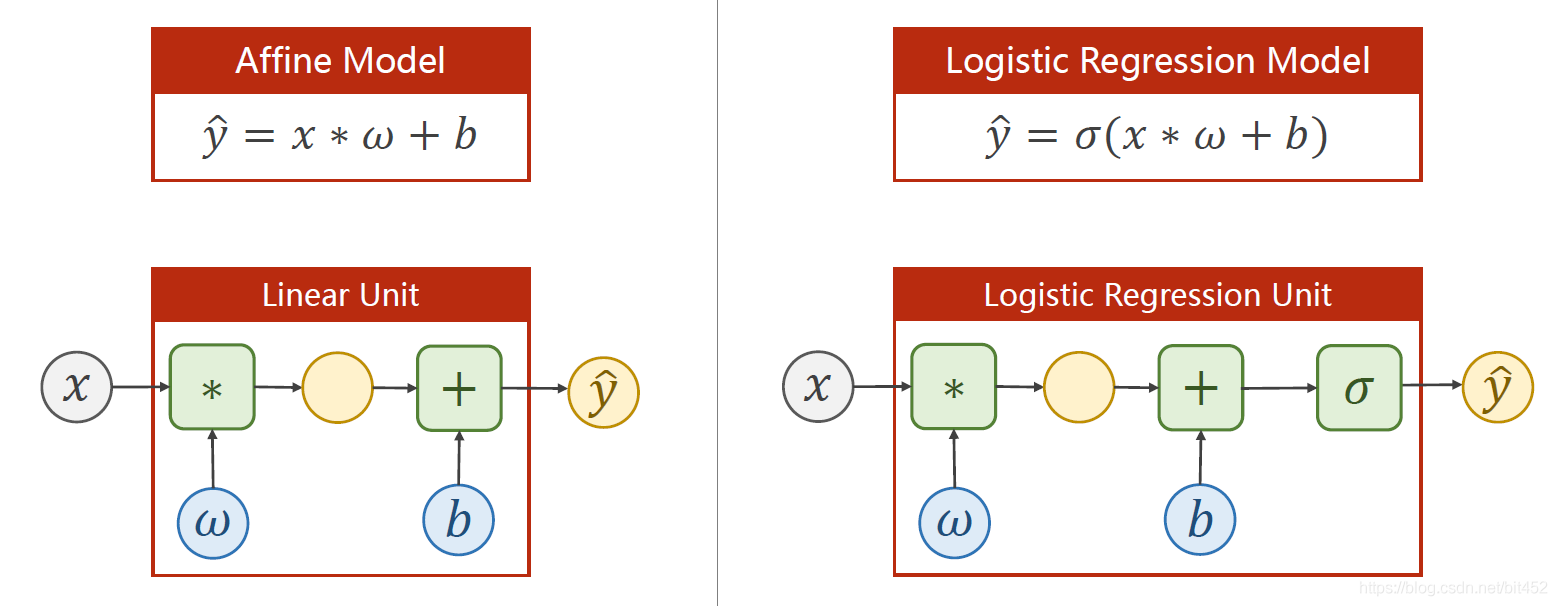
PyTorch 深度学习实践 第6讲

第6讲 逻辑斯蒂回归 源代码

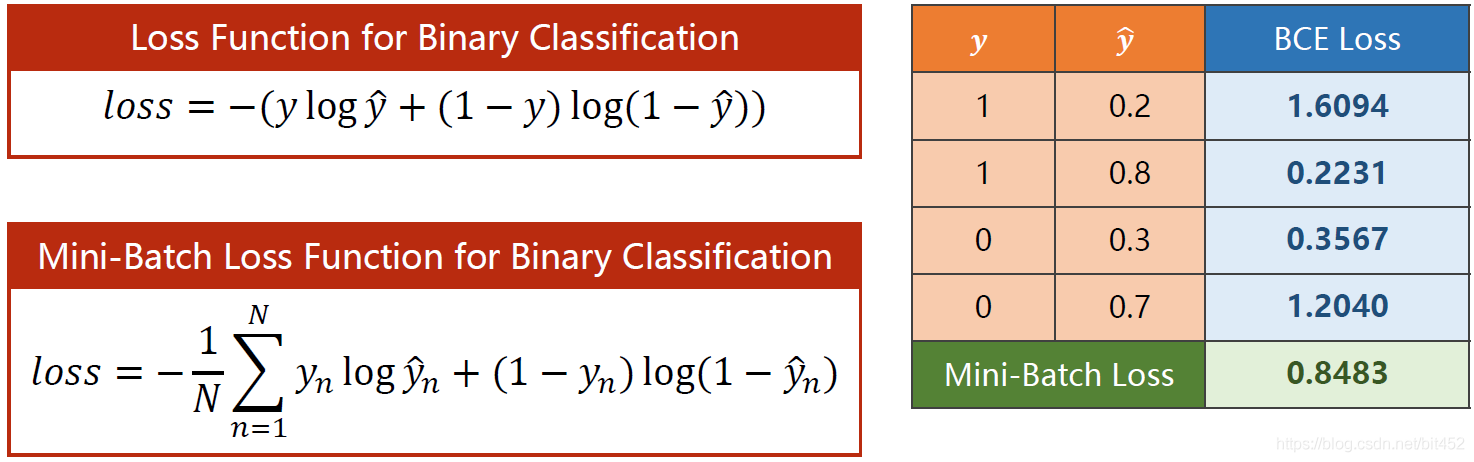
B站 刘二大人 ，传送门PyTorch深度学习实践——逻辑斯蒂回归

视频中截图



说明：1、 逻辑斯蒂回归和线性模型的明显区别是在线性模型的后面，添加了激活函数(非线性变换)

2、分布的差异：KL散度，cross-entropy交叉熵



说明：预测与标签越接近，BCE损失越小。

代码说明：

1、视频中代码F.sigmoid(self.linear(x))会引发warning，此处更改为torch.sigmoid(self.linear(x))

     torch.sigmoid() 与 torch.nn.Sigmoid() 对比

     torch.sigmoid()、torch.nn.Sigmoid()和torch.nn.functional.sigmoid()三者之间的区别

2、BCELoss - Binary CrossEntropyLoss

     BCELoss 是CrossEntropyLoss的一个特例，只用于二分类问题，而CrossEntropyLoss可以用于二分类，也可以用于多分类。

     如果是二分类问题，建议BCELoss

    损失函数（BCELoss）

   BCE和CE交叉熵损失函数的区别

|  |
| --- |
| import torch  # import torch.nn.functional as F    # prepare dataset  x\_data = torch.Tensor([[1.0], [2.0], [3.0]])  y\_data = torch.Tensor([[0], [0], [1]])    #design model using class  class LogisticRegressionModel(torch.nn.Module):  def \_\_init\_\_(self):  super(LogisticRegressionModel, self).\_\_init\_\_()  self.linear = torch.nn.Linear(1,1)    def forward(self, x):  # y\_pred = F.sigmoid(self.linear(x))  y\_pred = torch.sigmoid(self.linear(x))  return y\_pred  model = LogisticRegressionModel()    # construct loss and optimizer  # 默认情况下，loss会基于element平均，如果size\_average=False的话，loss会被累加。  criterion = torch.nn.BCELoss(size\_average = False)  optimizer = torch.optim.SGD(model.parameters(), lr = 0.01)    # training cycle forward, backward, update  for epoch in range(1000):  y\_pred = model(x\_data)  loss = criterion(y\_pred, y\_data)  print(epoch, loss.item())    optimizer.zero\_grad()  loss.backward()  optimizer.step()    print('w = ', model.linear.weight.item())  print('b = ', model.linear.bias.item())    x\_test = torch.Tensor([[4.0]])  y\_test = model(x\_test)  print('y\_pred = ', y\_test.data) |

————————————————

版权声明：本文为CSDN博主「错错莫」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/bit452/article/details/109680909