PyTorch 深度学习实践 第9讲

第9讲  多分类问题 源代码

B站 刘二大人 ，传送门PyTorch深度学习实践——多分类问题

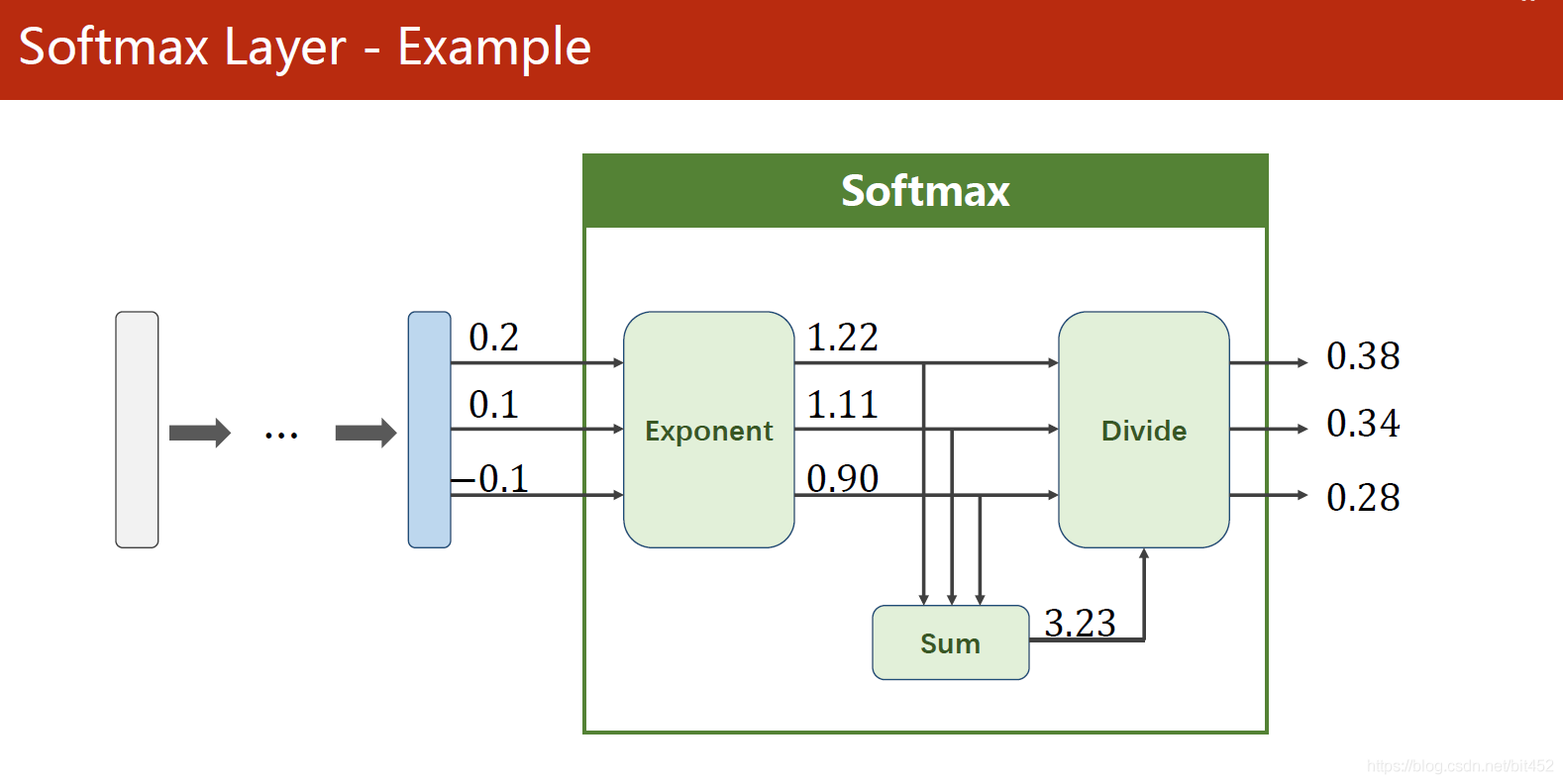
视频中截图

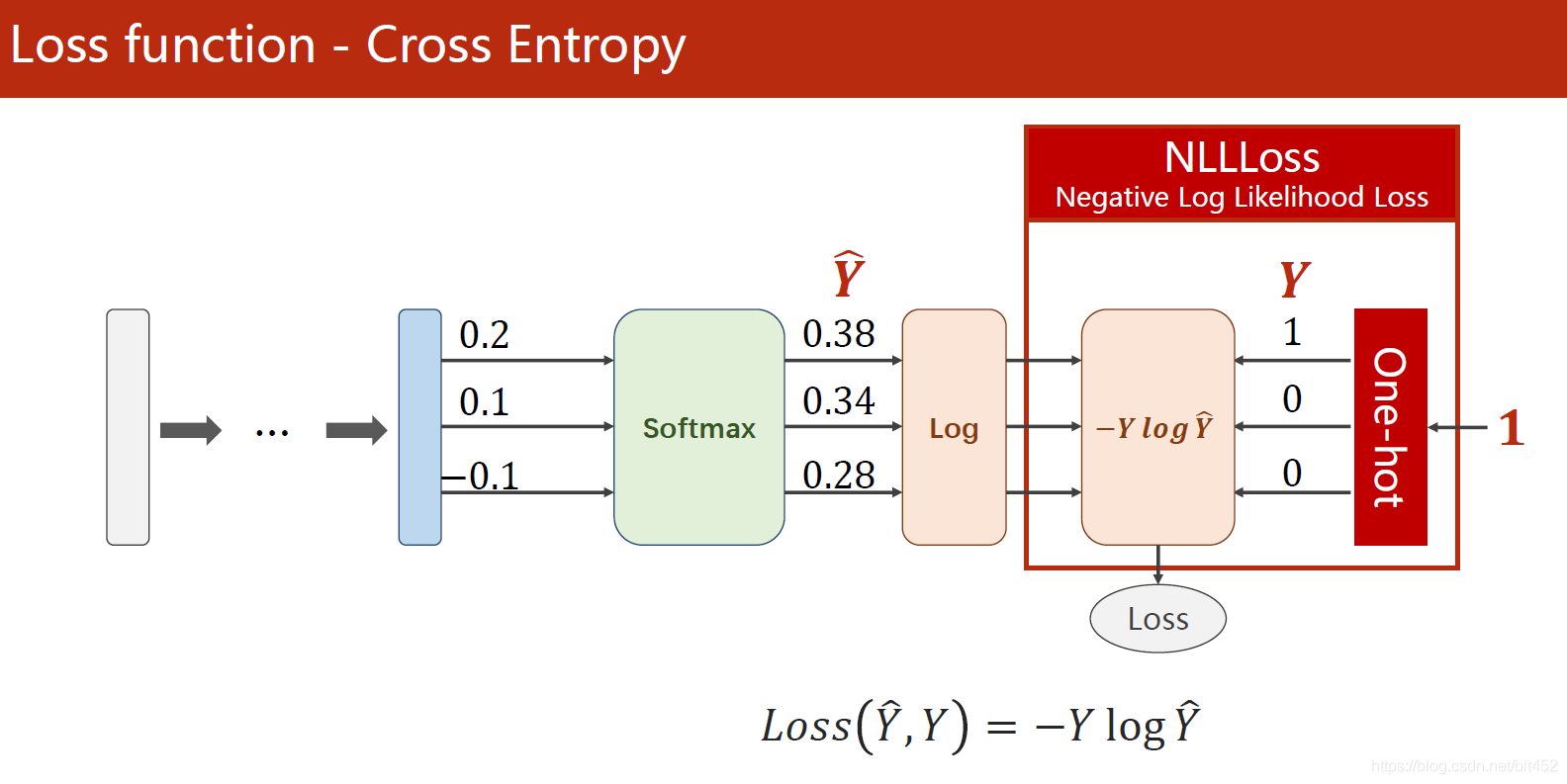
说明： 1、softmax的输入不需要再做非线性变换，也就是说softmax之前不再需要激活函数(relu)。softmax两个作用，如果在进行softmax前的input有负数，通过指数变换，得到正数。所有类的概率求和为1。

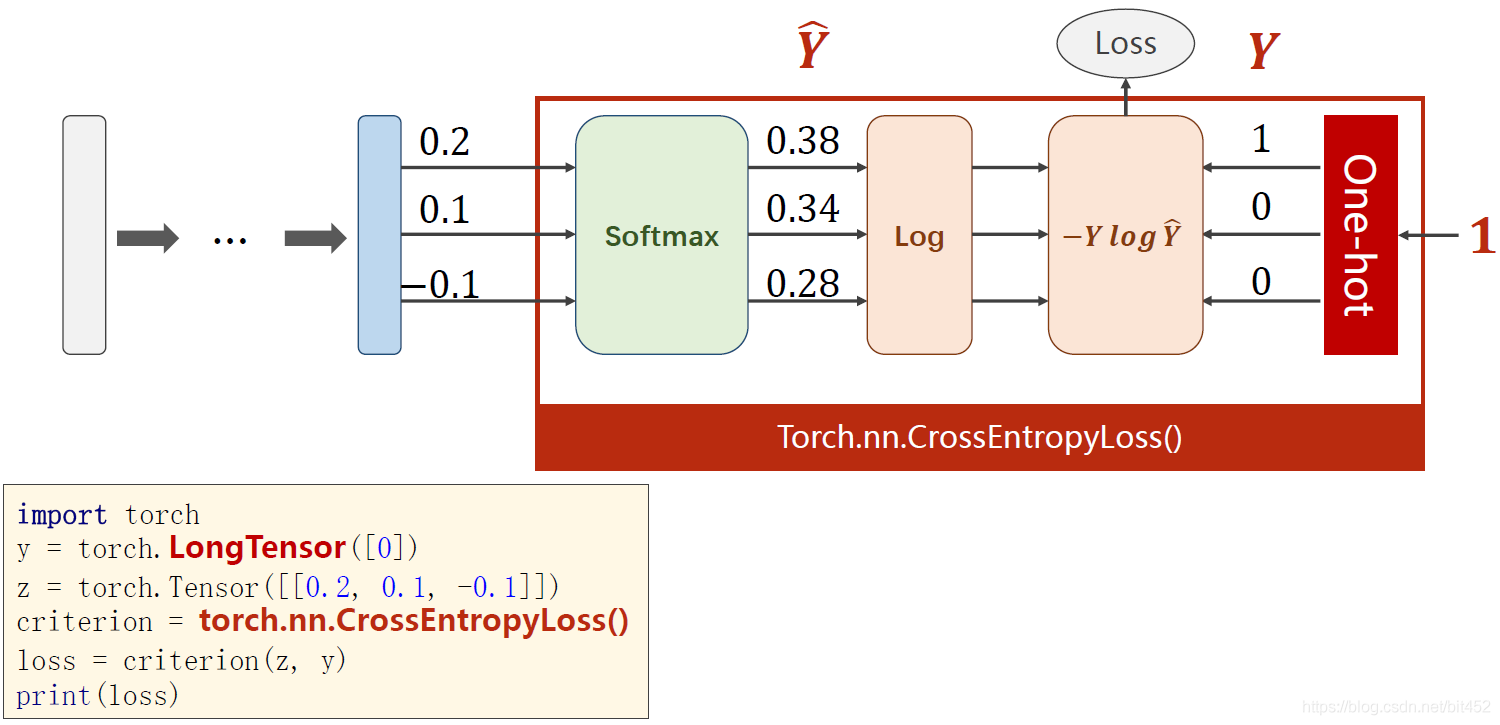
2、y的标签编码方式是one-hot。我对one-hot的理解是只有一位是1，其他位为0

3、多分类问题，标签y的类型是LongTensor。比如说0-9分类问题，如果y = torch.LongTensor([3])，对应的one-hot是[0,0,0,1,0,0,0,0,0,0].

4、CrossEntropyLoss <==> LogSoftmax + NLLLoss







代码说明：1、第8讲 from torch.utils.data import Dataset，第9讲 from torchvision import datasets。该datasets里面init，getitem,len魔法函数已实现。

2、torch.max的返回值有两个，第一个是每一行的最大值是多少，第二个是每一行最大值的下标(索引)是多少。

3、全连接神经网络

4、torch.no\_grad()   Python中with的用法

5、代码中"\_"的说明  Python中各种下划线的操作

6、torch.max( )的用法  torch.max( )使用讲解

|  |
| --- |
| import torch  from torchvision import transforms  from torchvision import datasets  from torch.utils.data import DataLoader  import torch.nn.functional as F  import torch.optim as optim    # prepare dataset    batch\_size = 64  transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))]) # 归一化,均值和方差    train\_dataset = datasets.MNIST(root='../dataset/mnist/', train=True, download=True, transform=transform)  train\_loader = DataLoader(train\_dataset, shuffle=True, batch\_size=batch\_size)  test\_dataset = datasets.MNIST(root='../dataset/mnist/', train=False, download=True, transform=transform)  test\_loader = DataLoader(test\_dataset, shuffle=False, batch\_size=batch\_size)    # design model using class      class Net(torch.nn.Module):  def \_\_init\_\_(self):  super(Net, self).\_\_init\_\_()  self.l1 = torch.nn.Linear(784, 512)  self.l2 = torch.nn.Linear(512, 256)  self.l3 = torch.nn.Linear(256, 128)  self.l4 = torch.nn.Linear(128, 64)  self.l5 = torch.nn.Linear(64, 10)    def forward(self, x):  x = x.view(-1, 784) # -1其实就是自动获取mini\_batch  x = F.relu(self.l1(x))  x = F.relu(self.l2(x))  x = F.relu(self.l3(x))  x = F.relu(self.l4(x))  return self.l5(x) # 最后一层不做激活，不进行非线性变换      model = Net()    # construct loss and optimizer  criterion = torch.nn.CrossEntropyLoss()  optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.5)    # training cycle forward, backward, update      def train(epoch):  running\_loss = 0.0  for batch\_idx, data in enumerate(train\_loader, 0):  # 获得一个批次的数据和标签  inputs, target = data  optimizer.zero\_grad()  # 获得模型预测结果(64, 10)  outputs = model(inputs)  # 交叉熵代价函数outputs(64,10),target（64）  loss = criterion(outputs, target)  loss.backward()  optimizer.step()    running\_loss += loss.item()  if batch\_idx % 300 == 299:  print('[%d, %5d] loss: %.3f' % (epoch+1, batch\_idx+1, running\_loss/300))  running\_loss = 0.0      def test():  correct = 0  total = 0  with torch.no\_grad():  for data in test\_loader:  images, labels = data  outputs = model(images)  \_, predicted = torch.max(outputs.data, dim=1) # dim = 1 列是第0个维度，行是第1个维度  total += labels.size(0)  correct += (predicted == labels).sum().item() # 张量之间的比较运算  print('accuracy on test set: %d %% ' % (100\*correct/total))      if \_\_name\_\_ == '\_\_main\_\_':  for epoch in range(10):  train(epoch)  test() |

————————————————

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/109686936