**深度学习第二次作业报告**

胡雅娴-2020311448

3.3小节

（1）对lr除以batch\_size，即减小lr。

（3）使用grad()函数，例如：要访问net[0].weight的梯度，使用net[0].weight.grad。

4.1小节

（1）prelu的导数为：y=1(x>0),y=a,(x<0)

（2）每层ReLU产生一个分段，且在每个分段中使用ReLU激活函数时进行线性变换。Relu(x)是连续分段函数，所以Relu(Relu(x)W+b)也是连续的分段线性函数

（3）

4.用多层感知机再次实现第一次作业中的第8题，即用多层感知机对MNIST数据集进

行训练，要求多层感知机至少有两个隐藏层，需要用到非线性激活函数。

模型代码如下：

|  |
| --- |
| # 导入相应的工具包  import numpy as np  from matplotlib import pyplot as plt  import tensorflow as tf  from tensorflow.keras.datasets import mnist  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense, Dropout, Activation, BatchNormalization  from tensorflow.keras import utils  from tensorflow.keras import regularizers  from torch.utils import data  from torchvision import transforms  import torch  from torch import nn  from d2l import torch as d2l  import torch  import torchvision  from IPython import display  d2l.use\_svg\_display() |
| #导入数据  trans = transforms.ToTensor()  batch\_size = 256  mnist\_train = torchvision.datasets.MNIST(  root="./data/",  train=True,  transform=transforms.ToTensor(),  download=True)  mnist\_test= torchvision.datasets.MNIST(  root="./data/",  train=False,  transform=transforms.ToTensor(),  download=True) |
| def get\_dataloader\_workers():  """调用进程来读取数据"""  return 0  train\_iter = data.DataLoader(mnist\_train, batch\_size, shuffle=True,  num\_workers=get\_dataloader\_workers())  test\_iter = data.DataLoader(mnist\_test,batch\_size,shuffle=True,  num\_workers=get\_dataloader\_workers()) |
| num\_inputs, num\_outputs, num\_hiddens1, num\_hiddens2 = 784, 10, 256, 64  W1 = nn.Parameter(torch.randn(  num\_inputs, num\_hiddens1, requires\_grad=True) \* 0.01)  b1 = nn.Parameter(torch.zeros(num\_hiddens1, requires\_grad=True))  W2 = nn.Parameter(torch.randn(  num\_hiddens1, num\_hiddens2, requires\_grad=True) \* 0.01)  b2 = nn.Parameter(torch.zeros(num\_hiddens2, requires\_grad=True))  W3 = nn.Parameter(torch.randn(  num\_hiddens2, num\_outputs, requires\_grad=True) \* 0.01)  b3 = nn.Parameter(torch.zeros(num\_outputs, requires\_grad=True))  params = [W1, b1, W2, b2, W3, b3] |
| def relu(X):  a = torch.zeros\_like(X)  return torch.max(X, a) |
| def net(X):  X = X.reshape((-1, num\_inputs))  H = relu(X@W1 + b1) # 这里“@”代表矩阵乘法  return (H@W2 + b2) |
| loss = nn.CrossEntropyLoss(reduction='none') |
| # 训练  num\_epochs, lr = 10, 0.1  updater = torch.optim.SGD(params, lr=lr)  d2l.train\_ch3(net, train\_iter, test\_iter, loss, num\_epochs, updater) |

调试过程中，先固定其他变量，调整num\_hiddens，找到最佳num\_hiddens后，固定其及其他参数，增加隐藏层，隐藏层数变多后效果更好，但也运行更慢，因此只增加到3层。其他参数不变调整各隐藏层单元数，找到最佳的隐藏层单元数。接下来只改变学习率，找到最佳学习率，最后只改变轮数，找到最佳轮数。

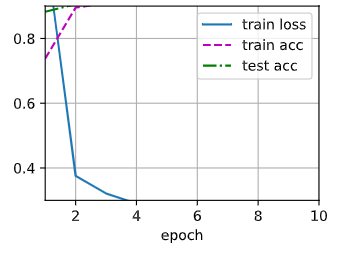
最终参数见下一题。

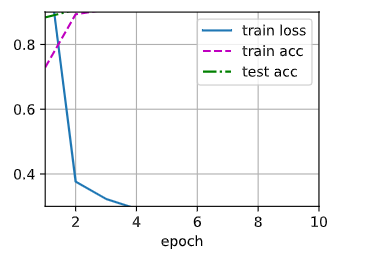
5. 基于第4题的模型，调试参数/超参数

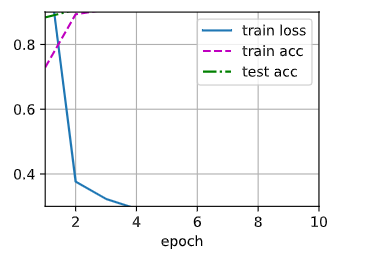
4.2小节

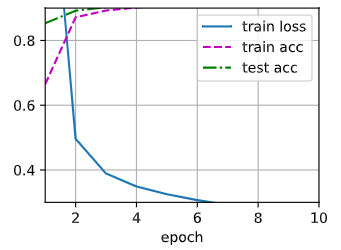
（1）超参数变化会改变预测准确率，确定num\_hiddens1为512，num\_hiddens2为64。

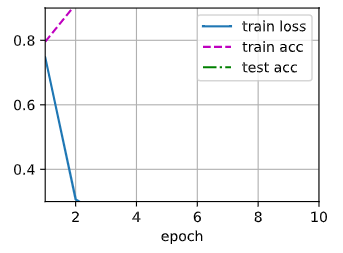
（2）增加隐藏层，效果变好了。

2隐藏层：

3隐藏层：

（3）0.1学习率：

0.05学习率：

0.2学习率：

最后选用0.1学习率.

（4）最终参数：

隐藏层2

num\_hiddens1为512

num\_hiddens2为64

学习率0.1

学习轮数20。