代码如下：

1. import torch
2. import torch.nn as nn
3. import torchvision
4. import PIL.Image as Image
5. import math  *# 主要做最后输出楼层向上取整*
6. import random
7. data = []
8. flag = []  *# 期望值标签列表组成的矩阵*
9. with open("d:/dataset/deeplearningclass/class.txt") as f:
10. for line in f:  *# 循环line次*
11. words = line.split(",")  *# words为列表*
12. img = Image.open("d:/dataset/deeplearningclass/" + words[0])  *# 打开图片装载进内存，这里第0项是文件名*
13. img = img.convert("L")  *# convert 把原来的图像做转换，L为二值图像只有黑色白色 tips：原图片像素为三原色*
14. imdl = list(img.getdata())  *# 转换成列表  getdata从图里面把每一个像素的值取出*
15. data.append([imdl])
16. i = int(words[1])  *# 取标签*
17. d = [0.0] \* 10  *# 初始化标签列表*
18. d[i - 1] = 1.0
19. flag.append([d])
20. *# print(data)*
21. dt = torch.tensor(data)  *# 列表转换为张量*
22. print(dt.size())
23. x = dt.view((30, 1, 20, 20))  *# view改变维度*
24. X = x / 255.0  *# 转换为0-1之间浮点数*
25. S = torch.tensor(flag)  *# 标签列表转换成tensor*
26. *# print(X)*
27. *# print(len(X))*
28. class Myconv(nn.Module):
29. def \_\_init\_\_(self):
30. super().\_\_init\_\_()  *# 初始化*
31. self.conv1 = nn.Conv2d(1, 16, kernel\_size=3, stride=1, padding=1)
32. self.conv2 = nn.Conv2d(16, 16, kernel\_size=3, stride=1, padding=1)
33. self.pool = nn.MaxPool2d(kernel\_size=2, stride=2)
34. self.pool2 = nn.AvgPool2d(kernel\_size=2, stride=2)
35. self.out = nn.Linear(16 \* 5 \* 5, 10)  *# 输入，输出*
36. self.loss = nn.MSELoss()  *# MSEloss就是前面的误差(误差平方），self.loss现在是函数形式*
37. self.optmiser = torch.optim.SGD(self.parameters(), lr=0.05)  *# SGD优化器，学习率选择为0.05*
38. self.count = 0  *# 训练计数*
39. self.progress = []  *# 进度表示需要存储的内容，用列表来表示*
40. def forward(self, x):  *# 将输入矩阵转换为输出*
41. x = self.conv1(x)
42. x = self.pool(x)
43. x = self.conv2(x)
44. x = self.pool2(x)
45. x = x.view((16 \* 5 \* 5))  *# 变成一纬然后送到fc层*
46. x = torch.sigmoid(self.out(x))
47. return x
48. def train(self, input, target):  *# 定义训练方法*
49. output = self.forward(input)
50. myloss = self.loss(output, target)  *# output,target为矩阵可能包含很多数据*
51. self.optmiser.zero\_grad()  *# 梯度清零*
52. myloss.backward()  *# 用误差做反向传播计算*
53. self.optmiser.step()  *# 调用优化器step方法进行梯度更新*
54. if self.count % 1000 == 0:
55. self.progress.append(myloss.item())  *# 每隔1000次记录一次误差，并添加到progress列表里*
56. self.count = self.count + 1
57. image = Image.open("d:/dataset/deeplearningclass/0-1.png")  *# 20\*20*
58. toPilImage = torchvision.transforms.ToPILImage()  *# topilimage处理图片*
59. image = image.convert("L")     *# l 转换为灰度图像*
60. data = list(image.getdata())     *# getdata获取图像数据，按行读取，像素*
61. print(len(data))
62. d = torch.tensor(data)    *# 列表转换为张量*
63. x = d.view((1, 1, 20, 20))  *# view改变维度*
64. x = x / 255.0   *# 转换为0-1之间浮点数*
65. net = Myconv()  *# 定义对象net*
66. epoch = 100000
67. for j in range(epoch):  *# 循环epoch次，这里为100000次*
68. i = random.randint(0, 29)
69. net.train(X[i], S[i])
70. ot = net.forward(X[i])
71. loss = net.progress[-1]
72. tag = math.ceil((i + 1) / 3)  *# 输出的时候对应的标签值，为了方便观察输出结果*
73. if j % 1000 == 0:
74. print('i=', i, 'tag：', tag, 'loss=', loss, 'final output is:', ot)

一个警告（关于损失函数的输入数据大小的），目前暂时没做处理：

UserWarning: Using a target size (torch.Size([1, 10])) that is different to the input size (torch.Size([10])). This will likely lead to incorrect results due to broadcasting. Please ensure they have the same size.return F.mse\_loss(input, target, reduction=self.reduction)

运行结果：

文本

描述已自动生成