MNIST分类任务实验报告

王颖

2020 年 4 月 10 日

1 任务描述

本次实验要求实现人工神经网络，求解MNIST任务，数据集和测试集来自THE MNIST DATABASE

2 实验分析

该实验具体包含数据预处理、实现数据加载器、构造神经网络模型、训练网络并储存模型、测试模型几个部分，下面分别介绍这些内容。

2.1 数据预处理

基于标签集与样本集的特征，这里实现了一个预处理数据的函数，将以字节形式存储的标签和图片分别读取到两个numpy array类型中，其中标签集的前8个字节为文件协议描述以及标签集大小（图片个数），样本集的前16个字节表示文件协议、样本集大小（图片个数）、图片的规格（行数和列数）。同时，这里将值域在$[0, 255]$的灰度值转成$[0, 1]$的float32类型，将单通道图转换为三通道图。预处理后的数据保存到$train_data/label.npy$、$test_data/label.npy$中，代码如下。

```
def Load(kind):
    labelsSrc = './' + kind + '-labels.idx1-ubyte'
    imagesSrc = './' + kind + '-images.idx3-ubyte'
    with open(labelsSrc, 'rb') as lrc:
        magic, n = struct.unpack('>II', lrc.read(8))
        labels = np.fromfile(lrc, dtype=np.uint8)

    with open(imagesSrc, 'rb') as irc:
        magic, n, r, c = struct.unpack('>IIII', irc.read(16))
        images = np.fromfile(irc, dtype=np.uint8).reshape(len(labels), 28 * 28)


    images = np.array(images)
    images = images.astype(np.float32) / 255
    images = np.repeat(images, 3)
```

```
    images = np.reshape(images, (-1, 28, 28, 3))

    labels = np.reshape(np.array(labels), (-1, )).astype(np.int64)

    return images, labels

trainX, trainY = Load('train')
np.save("train_data.npy", trainX)
np.save("train_label.npy", trainY)

testX, testY = Load('t10k')
np.save("test_data.npy", testX)
np.save("test_label.npy", testY)
```

2.2 实现数据加载器

实现数据加载器

```
class MNISTDataset(torch.utils.data.Dataset):
    def __init__(self, transform, data, label):
        super(MNISTDataset, self).__init__()
        self.transform = transform
        self.images = data
        self.labels = label

    def __getitem__(self, idx):
        img = self.images[idx]
        img = self.transform(img)
        label = self.labels[idx]
        return img, label

    def __len__(self):
        return len(self.images)
```

从$train_data/label.npy$、$test_data/label.npy$中载入数据，$transform$中对输入的数据进一步变换，使其分布在$[-0.5, 0.5]$。

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

train_data = np.load('train_data.npy')
train_label = np.load('train_label.npy')
test_data = np.load('test_data.npy')
test_label = np.load('test_label.npy')

trainset = MNISTDataset(transform=transform, data=train_data, label=train_label)
```

```
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4, shuffle=True,
                                          num_workers=0)

testset = MNISTDataset(transform=transform, data=test_data, label=test_label)
testloader = torch.utils.data.DataLoader(testset, batch_size=4, shuffle=False,
                                          num_workers=0)
```

2.3 构造神经网络模型

利用*torch.nn*中的*Module*构造神经网络模型，下面为该模型的具体内容，包含两个卷积层和三个全连接层。

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(3, 4, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(4, 10, 5)

        self.fc1 = nn.Linear(8 * 8 * 10, 150)
        self.fc2 = nn.Linear(150, 80)
        self.fc3 = nn.Linear(80, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))

        x = x.view(-1, 8 * 8 * 10)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

2.4 训练网络并储存模型

利用上述神经网络模型，实例化一个网络，并对它进行训练。将训练好的网络保存到*net.pth*中。

```
net = Net()

import torch.optim as optim
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
running_loss = 0
```

```
for epoch in range(300):
    for i, data in enumerate(trainloader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = net(inputs)
        labels = torch.LongTensor(labels)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        if i % 2000 == 1999:
            print('[%d, %5d] loss: %.3f' %
            (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0
            PATH = './net.pth'
            torch.save(net.state_dict(), PATH)
```

2.5 测试模型

从*net.pth*中加载出训练好的模型参数，并进行测试。

```
net = Net()
net.load_state_dict(torch.load('./net.pth'))
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))

with torch.no_grad():
    for data in testloader:
        images, labels = data
        outputs = net(images)
        _, predicted = torch.max(outputs, 1)
        c = (predicted == labels).squeeze()
        for i in range(4):
            label = labels[i]
            class_correct[label] += c[i].item()
            class_total[label] += 1

total, correct = 0, 0
for i in range(10):
    correct += class_correct[i]
    total += class_total[i]

print(correct / total)

for i in range(10):
```

```
    print('Accuracy of %5s : %2d %%' % (classes[i], 100 * class_correct[i] /
                                                 class_total[i]))
```

2.6 实验结果

实验结果显示，该模型在测试集中的正确率达到99.02%。

3 总结

本次实验中，我利用上一次实验的数据输入实现预处理，进一步实现数据加载器。之后利用$torch.nn$中的$Module$构造神经网络模型，实例化网络后对它进行训练、测试。通过这个实验，我对人工神经网络的认识有了加深。