Lecture Notes of NOAI Training

Day 2, June 8, 2024

1. MNIST

```
# 定义数据集
test_dataset=FashionDataset(
    datadir='FashionMNIST/raw',
    transform=T.Compose([
        T.ToTensor(),
        T.Normalize((0.5,),(0.5,))
    ]),
    is_train=True
)

# 定义测试集
test_dataset=FashionDataset(
    datadir='FashionMNIST/raw',
    transform=T.Compose([
        T.ToTensor(),
        T.Normalize((0.5,),(0.5,))
    ]),
    is_train=False
)

# 网络结构
def __init__(self):
    super(LeNet,self).__init__()
    self.conv1=nn.Conv2d(1,6,5,1,2)
    self.pool1=nn.MaxPool2d(2,2)
    self.conv2=nn.Conv2d(6,16,5)
    self.pool2=nn.MaxPool2d(2,2)
    self.fc1=nn.Linear(16*5*5,120)
    self.fc2=nn.Linear(120,84)
    self.fc3=nn.Linear(84,10)

# 前向传播
def forward(self,x):
    x=F.relu(self.conv1(x))
    x=self.pool1(x)
    x=F.relu(self.conv2(x))
    x=self.pool2(x)
    x=x.view(x.size()[0],-1)
    x=F.relu(self.fc1(x))
```

```
    x=F.relu(self.fc2(x))
    x=self.fc3(x)
    return x

# 训练
def train(epoch):
    model.train()
    total_loss=0
    for iter,(data,target) in enumerate(train_loader):
        opt.zero_grad()
        output=model(data)
        loss=loss_fn(output,target)
        loss.backward()
        opt.step()
        total_loss+=loss.item()
    total_loss/=len(train_loader)
    print(f'Epoch: {epoch} Loss: {total_loss:.3f}')

# 测试
def test(epoch):
    model.eval()
    correct=0
    tot=0
    for data,target in test_loader:
        output=model(data)
        pred=[]
        for i in range(output.size(0)):
            max_index=0
            max_value=output[i][0]
            for j in range(1,output.size(1)):
                if output[i][j]>max_value:
                    max_value=output[i][j]
                    max_index=j
            pred.append(max_index)
    for i in range(len(pred)):
        if pred[i]==target[i]:
            correct+=1
    tot+=data.shape[0]
print(f'Test Epoch: {epoch} Accuracy: {correct/tot*100:.2f}%')

# 训练模型
for epoch in range(10):
    train(epoch)
    test(epoch)
```

2. 多变量

```
1
2  # 训练
3  def train(epoch):
4      #...
5      #for...
6          output=model(data)
7          output=torch.sigmoid(output)
8
9  # 测试
10 def test(epoch):
11     #...
12     #for...
13         output=model(data)
14         output=torch.sigmoid(output)
15         _,output_indices=torch.topk(output,k=2,dim=-1,largest=True,sorted=True)
16         _,target_indices=torch.topk(target,k=2,dim=-1,largest=True,sorted=True)
17         output_indices=output_indices.numpy()
18         output_indices=target_indices.numpy()
19     #for...
20         if np.array_equal(output_indices[i],target_indices[i]):
21             correct+=1
```

3. GAN 生成对抗网络

流程：

1. 设置随机种子
2. dataloader
3. 判别器网络结构
4. 生成器网络结构
5. 实例化判别器和生成器
6. 使用未训练的生成器生成数据并可视化
7. 超参数
8. 损失函数
9. 优化器设置
10. 训练判别器和生成器
11. 模型测评

$$\begin{gathered}\text{nn.Linear} \leftarrow \text{2D} \\ \text{nn.Linear} \rightarrow \text{nn.ReLU} \rightarrow \text{nn.Dropout} \\ \text{nn.Linear 2D} \rightarrow \text{1D}\end{gathered} \tag{1}$$