2151131-朱沙桐-课后作业3

2151131 朱沙桐

PyTorch

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(
                # patch 7 * 7 ; 1  in channels ; 32 out channels ; ; stride is 1
                # padding style is same(that means the convolution opration's input and output have the same size)
                in_channels = 1,
                out_channels= 32,
                kernel_size= 5,
                stride= 1,
                padding= 2,
            ),
            nn.ReLU(),        # activation function
            nn.MaxPool2d(2),  # pooling operation
        )
        self.conv2 = nn.Sequential(
            # line 1 : convolution function, patch 5*5 , 32 in channels ;64 out channels; padding style is same; stride is 1
            # line 2 : choosing your activation funciont
            # line 3 : pooling operation function.
            nn.Conv2d(
                in_channels=32,
                out_channels=64,
                kernel_size=5,
                stride=1,
                padding=2,
            ),
            nn.ReLU(),
            nn.MaxPool2d(2),
        )
        self.out1 = nn.Linear( 7*7*64 , 1024 , bias= True)   # full connection layer one

        self.dropout = nn.Dropout(keep_prob_rate)
        self.out2 = nn.Linear(1024,10,bias=True)



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

        # flatten the output of coonv2 to (batch_size ,32 * 7 * 7)
        x = x.view(x.size(0), -1)
```

```
        out1 = self.out1(x)
        out1 = F.relu(out1)
        out1 = self.dropout(out1)
        out2 = self.out2(out1)
        output = F.softmax(out2)
        return output


def test(cnn):
    global prediction
    y_pre = cnn(test_x)
    _,pre_index= torch.max(y_pre,1)
    pre_index= pre_index.view(-1)
    prediction = pre_index.data.numpy()
    correct  = np.sum(prediction == test_y)
    return correct / 500.0


def train(cnn):
    optimizer = torch.optim.Adam(cnn.parameters(), lr=learning_rate )
    loss_func = nn.CrossEntropyLoss()
    for epoch in range(max_epoch):
        for step, (x_, y_) in enumerate(train_loader):
            x ,y= Variable(x_),Variable(y_)
            output = cnn(x)
            loss = loss_func(output,y)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            if step != 0 and step % 20 ==0:
                print("=" * 10,step,"="*5,"="*5, "test accuracy is
",test(cnn) ,"=" * 10 )
```

TensorFlow

```
def conv2d(x, W):
    # 每一维度  滑动步长全部是 1,  padding 方式 选择 same
    # 提示 使用函数  tf.nn.conv2d
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
    # 滑动步长 是 2步; 池化窗口的尺度 高和宽度都是2; padding 方式 请选择 same
    # 提示 使用函数  tf.nn.max_pool
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
padding='SAME')


#  卷积层 1
## conv1 layer ##
W_conv1 = weight_variable([7, 7, 1, 32])                    # patch 7x7, in
size 1, out size 32
b_conv1 = bias_variable([32])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)    # 卷积 选择激活函数
```

```
h_pool1 = max_pool_2x2(h_conv1)                                    # 池化

# 卷积层 2
W_conv2 = weight_variable([5, 5, 32, 64])                          # patch 5x5, in
size 32, out size 64
b_conv2 = bias_variable([64])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)           # 卷积 选择激活函数
h_pool2 = max_pool_2x2(h_conv2)                                    # 池化
```