人工智能实验：Topic 7——卷积神经网络

3190102060　黄嘉欣

一、卷积层

CNN 一般由三种不同的层组成，即卷积层、池化层、连接层。其中，卷积层可以实现图像特征的提取。卷积神经网络中每层卷积层由若干卷积单元组成，每个卷积单元的参数通过反向传播算法最佳化得到。卷积运算的目的是提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网路能从低级特征中迭代提取更复杂的特征。若记卷积层为一个$F \times F \times D$的 filter，则其深度必须和输入图像的深度相同。当选择不同的 filter 时，得到的特征输出也将会出现不同，如图 1.1 所示。

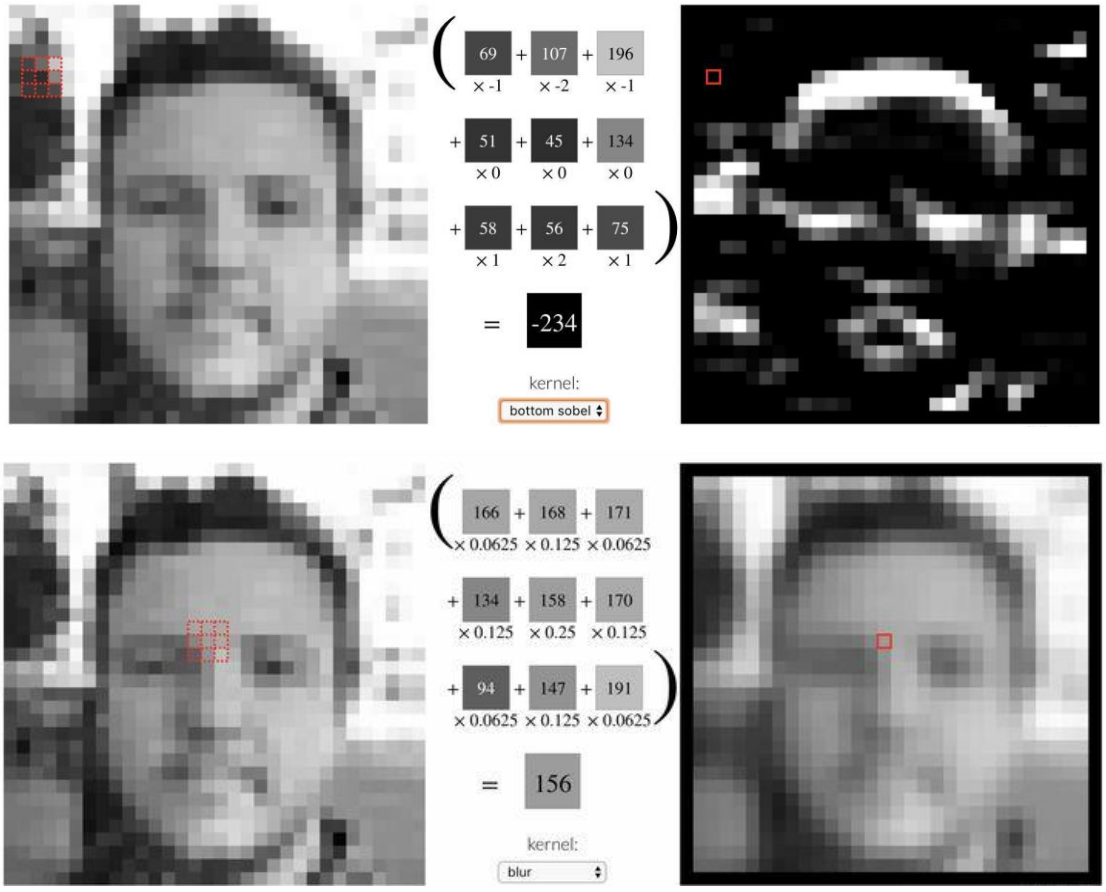

图 1.1 卷积层输出

二、池化层（Pooling layer）

池化层是当前卷积神经网络中常用组件之一，其模仿人的视觉系统对数据进行降维，用更高层次的特征表示图像。一般而言，实施池化的目的包括：(1) 降低信息冗余；(2)

提升模型的尺度不变性、旋转不变性；(3) 防止过拟合。在进行池化时，常见的操作包含以下几种：最大值池化，均值池化，随机池化，中值池化，组合池化等。其中，最大值池化在前向过程中选择图像区域中的最大值作为该区域池化后的值；在后向过程中，梯度通过前向过程时的最大值反向传播，其他位置的梯度为 0，如图 2.1 所示。而均值池化则是在前向传播时，计算图像区域中的均值作为该区域池化后的值；在反向传播时，梯度特征分配到各个位置。

总的来说，Pooling 层是一个特征选择、信息过滤的过程。虽然我们在采样时损失了一部分信息，但却在一定程度上提高了计算性能，不失为一种折中与妥协。

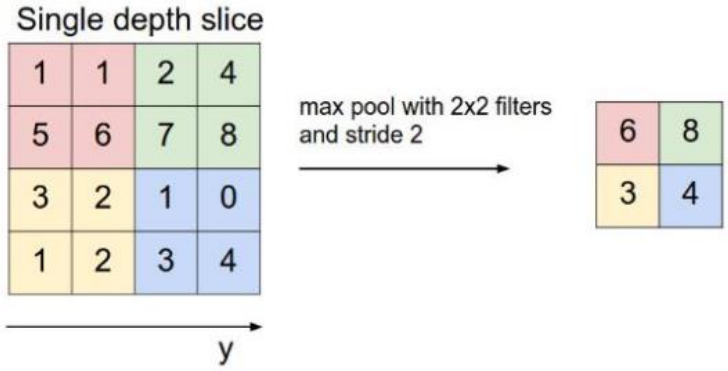

图 2.1 最大值池化

三、实验 7-1

① 实验题目

通过给定的 Pool、Conv 和 LoadMnistData 函数，使用 momentum 规则，对如图 3.1 所示结构的卷积神经网络进行训练，每层规模如图 3.2 所示；采用训练数据：测试数据=8：2 的比例（训练数据 2000），进行训练和测试，并输出训练后预测结果的准确度。

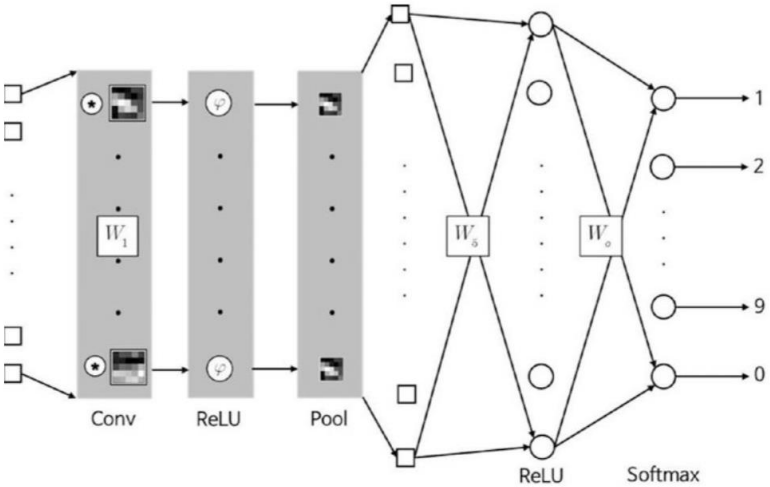

图 3.1 网络结构

Layer	Remark	Activation Function
Input	28×28 nodes	-
Convolution	20 convolution filters (9×9)	ReLU
Pooling	1 mean pooling (2×2)	-
Hidden	100 nodes	ReLU
Output	10 nodes	Softmax

图 3.2 网络规模

② 实验结果与分析

设置学习率$\alpha = 0.01$，momentum 参数$\beta = 0.9$，采用 mini-batch 策略，batch 大小为 100，当训练 5 轮时，得到的准确率为 0.84，如下所示：

```
The accuracy of the CNN net is 0.84
```

将训练集大小增加到 8000，此时测试结果的准确率为：0.971。可见，准确率有了一定程度的提升，网络设计正确。

```
The accuracy of the CNN net is 0.971
```

四、附录：实验 Python 代码——*CNN.py*

```
from Conv import *
from LoadMnistData import *
from Pool import *
import numpy as np
from scipy import signal


def sigmoid(x):
    """
    sigmoid 函数
    Param:
        x: 输入
    Return:
        y: 对应的 sigmoid 函数值
    """
    y = 1./(1+np.exp(-x))
    return y


def softmax(x):
    """
```

```
    softmax 函数
    Param:
        x: 输入
    Return:
        y: 对应的 softmax 函数值
    """
    x = x - np.max(x)
    exp_x = np.exp(x)
    y = exp_x / np.sum(exp_x)
    return y


def ReLU(x):
    """
    ReLU 函数
    Param:
        x: 输入
    Return:
        y: 对应的 ReLU 函数值
    """
    y = np.maximum(0, x)
    return y


def trainCNN(alpha, beta, batchSize, W1, W5, Wo, X, D, epochs):
    """
    训练 CNN 网络
    Param:
        alpha: 学习率
        beta: momentum 参数
        batchSize: 每个 batch 的大小
        W1, W5, Wo: 网络的初始权重
        X: 图像
        D: 标签
        epochs: 训练轮数
    Return:
        W1, W5, Wo: 网络训练好的权重
    """
    m1 = np.zeros_like(W1)   # momentum
    m5 = np.zeros_like(W5)
    mo = np.zeros_like(Wo)

    batchList = np.arange(0, X.shape[0], batchSize)

    for i in range(epochs):
```

```
        print('epoch: {}'.format(i))
        for index in range(len(batchList)):    # batch
            dW1 = np.zeros_like(W1)
            dW5 = np.zeros_like(W5)
            dWo = np.zeros_like(Wo)

            start = batchList[index]

            for j in range(start, start+batchSize):
                # 前向传输
                x = X[j, :, :]             # input, 28*28
                y1 = Conv(x, W1)           # 卷积层输出, 20*20*20
                y2 = ReLU(y1)              # 第二层输出, 20*20*20
                y3 = Pool(y2)              # 池化层输出, 10*10*20
                y4 = y3.reshape(2000, 1)   # 调整维度, 2000*1
                v5 = np.dot(W5, y4)
                y5 = ReLU(v5)              # 第五层输出, 100*1
                v = np.dot(Wo, y5)
                y = softmax(v)             # 输出, 10*1

                # one-hot 编码
                gt = np.zeros((1, 10))     # 行向量, 1*10
                gt[0, D[j][0]] = 1

                # 后向传输
                e = gt.T - y               # 输出误差, 10*1
                delta = e                  # softmax 导数为 1, 10*1
                dWo += alpha*np.dot(delta, y5.T)  # 输出更新值, 10*100

                e5 = np.dot(Wo.T, delta)   # 第五层误差, 100*1
                delta5 = (v5>0)*e5         # 第五层 delta, 100*1
                dW5 += alpha*np.dot(delta5, y4.T) # 第五层更新, 100*2000

                e4 = np.dot(W5.T, delta5)  # 第四层误差, 2000*1

                e3 = e4.reshape(y3.shape)  # 池化层误差, 10*10*20

                e2 = np.zeros_like(y2)     # 初始化第二层误差, 20*20*20
                W3 = np.ones_like(y2) / (2*2)
                for c in range(20):
                    e2[:, :, c] = np.kron(e3[:, :, c], np.ones((2,
                                          2)))*W3[:, :, c]
                delta2 = (y2>0)*e2         # 第二层 delta, 20*20*20
```

```
                delta1_x = np.zeros_like(W1) # 初始化卷积层 delta
                for c in range(20):
                    delta1_x[:, :, c] = signal.convolve2d(x[:, :],
                                    np.rot90(delta2[:, :, c], 2), 'valid')
                dW1 += delta1_x            # 卷积层更新值，9*9*20

            dWo = dWo / batchSize      # 取平均
            dW5 = dW5 / batchSize
            dW1 = dW1 / batchSize

            mo = dWo + beta*mo         # momentum
            m5 = dW5 + beta*m5
            m1 = dW1 + beta*m1

            Wo = Wo + mo               # 更新权重
            W5 = W5 + m5
            W1 = W1 + m1
    return W1, W5, Wo


def testCNN(batchSize, X_test, D_test, W1, W5, Wo):
    """
    测试 CNN 网络
    Param:
        batchSize: 每个 batch 的大小
        X_test: 图像
        D_test: 标签
        W1, W5, Wo: 网络的权重
    Return:
        accuracy: 准确度
    """
    N = len(D_test)    # 测试集长度
    count = 0

    for i in range(X_test.shape[0]):
        # 前向传输
        x = X_test[i, :, :]        # input, 28*28
        y1 = Conv(x, W1)           # 卷积层输出，20*20*20
        y2 = ReLU(y1)              # 第二层输出，20*20*20
        y3 = Pool(y2)              # 池化层输出，10*10*20
        y4 = y3.reshape(2000, 1)   # 调整维度，2000*1
        v5 = np.dot(W5, y4)
        y5 = ReLU(v5)              # 第五层输出，100*1
        v = np.dot(Wo, y5)
        y = softmax(v)             # 输出，10*1
```

```
        num = np.argmax(y)        # 判定的数字

        if D_test[i][0] == num:    # 识别正确
            count += 1

    accuracy = count / N
    return accuracy


def main():
    # 加载数据
    data, label = LoadMnistData('MNIST\\t10k-images-idx3-ubyte.gz',
                                'MNIST\\t10k-labels-idx1-ubyte.gz')
    data = np.divide(data, 255)
    X = data[0:2000, :, :]
    D = label[0:2000]

    # 初始化权重
    W1 = 0.01*np.random.randn(9, 9, 20)       # W1 的维度为 9*9*20
    W5 = 0.01*(2*np.random.rand(100, 2000)-1) # W5 的维度为 100*2000
    Wo = 0.01*(2*np.random.rand(10, 100)-1)   # Wo 的维度为 10*100

    # 初始化参数
    alpha = 0.01      # 学习率
    beta = 0.9        # momentum 参数
    batchSize = 100   # batch 的大小
    epochs = 5        # 训练轮数

    # 训练
    W1,W5,Wo = trainCNN(alpha, beta, batchSize, W1, W5, Wo, X, D,epochs)

    # 测试
    X_test = data[2000:2500, :, :]
    D_test = label[2000:2500]
    accuracy = testCNN(batchSize, X_test, D_test, W1, W5, Wo)
    print('The accuracy of the CNN net is {}'.format(accuracy))


if __name__ == '__main__':
    main()
```