**卷积神经网络**

**介绍**

在本实验中，你将实现一个简单卷积神经网络，并将其应用于手写数字识别的任务。

**本实验中包含的文件**

ex3.py-帮助你完成实验的Python脚本

ex3data1.mat – 手写数字数据集

displayData.py -可视化数据集函数

[\*]maxpool.py –最大池化函数

[\*]softmax.py - softmax函数

[\*]conv.py – 卷积函数

\* 代表你需要完成的文件

**1神经网络**

在这个实验中，你将徒手实现一个简单的卷积神经网络。

**1.1数据可视化**

在ex3.py的第一部分代码将加载数据并通过调用函数displayData将其显示在一个二维图形上(图1)。

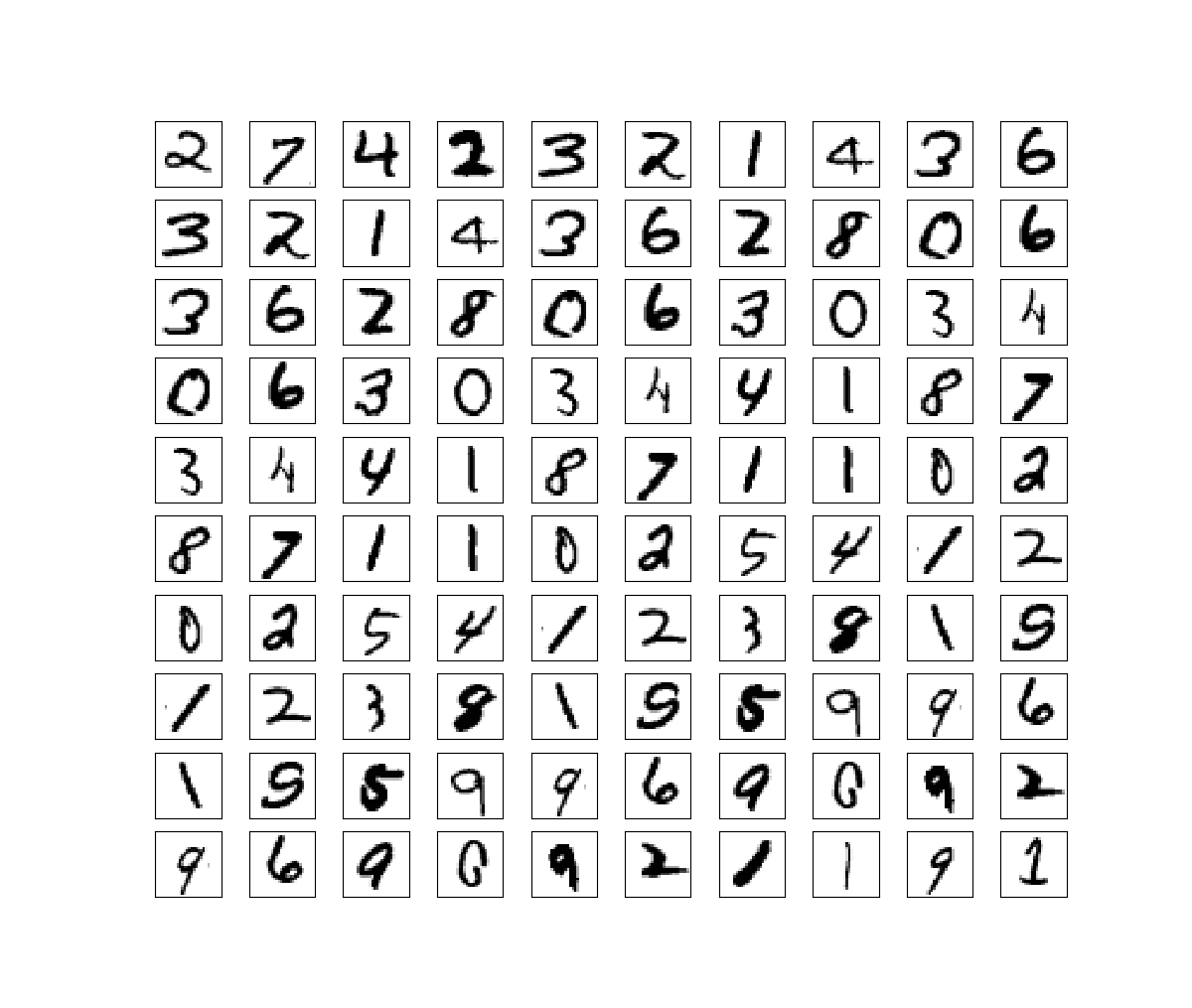


图1:部分数据集

在ex3data1.mat中有60000个训练数据和10000个测试数据，其中每个训练数据和测试数据都是28像素× 28像素灰度图像的数字。每个像素由一个浮点数表示，表示该位置的灰度强度。28 × 28像素矩阵被“展开”成一个784维的矢量。

训练集和测试集的第二部分是一个向量y，它包含训练集和测试集的标签。

标签为数字“0”到“9”。在本实验中选取了2000个数据作为训练集，500个数据作为测试集。

**1.2模型表示**

在本实验中我们将实现一个简单的卷积神经网络如图2所示。由一个 Conv 层、一个 Max Pooling 层和一个 Softmax 层组成。

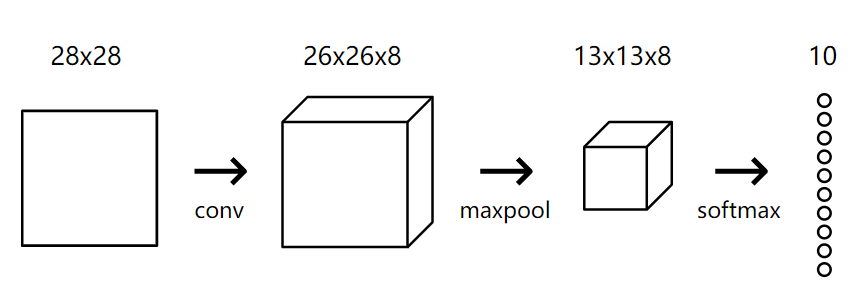


图2：卷积神经网络

代码中编写了三个类，每层包含一个类：Conv3x3、MaxPool和Softmax。每个类都实现了一个forward()方法（代码中有详细解释），用来构建CNN的向前传播。

**1.3模型的训练**

训练一个神经网络主要包括两部分：第一部分是前向传播（代码已给出），第二部分是反向传播。

本实验将按照这个模式来训练CNN。本实验还将使用两种主要的具体实现思想：

1.在 forward phase 中，每一层都需要存储一些数据（例如输入数据，中间值等）。这些数据将会在 backward phase 中得到使用。因此每一个backward phase 都需要在相应的forward phase之后运行。

2.在backward phase中，每一层都要获取 gradient 并且也返回 gradient。获取的是 loss 对于该层输出的 gradient，返回的是 loss 对于该层输入的 gradient。

**1.4反向传播：Softmax（Backprop: Softmax）**

我们需要从最后开始朝着最前面计算，这就是 backprop 的工作原理。首先回想下交叉熵损失 函数（cross-entropy loss）：



其中：为真实概率,为预测概率为预测结果与真实结果的差距

在我们的具体问题中，对于真实概率，只有分类正确数字对应的概率为 1，其他均为 0，因此 交叉熵损失函数 可以写成如下形式：



其中，是正确类（也就是图片中的数字）的预测概率,L的值越小越好。首先需要计算softmax 层的 backward phase 的输入数据，其中（下标s表示softmax层）是指 softmax 层的输出值：一个含有 10 个概率值的向量。由于只出现在了 loss 方程中，因此很容易计算：  

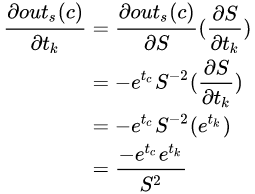

在ex3.py中对gradient进行了初始化。

|  |
| --- |
| # Calculate initial gradient gradient = np.zeros(10) gradient[label] = -1 / out[label] |

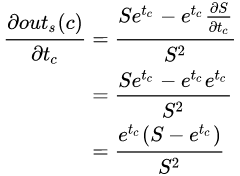
在开始实现第一个 backward phase之前，需要首先在 forward phase 中存储前向数据，softmax.py中已给出forward的代码。接下来我们可以获取 backprop phase 的 gradient。 我们已经获取 softmax backward phase 的输入 gradient：。由于只有一个是有值的，其他都是 0，因此我们可以忽略除了之外的其他值。

首先，计算对于totals（softmax.py中forwawrd函数中的totals，softmax 转换前的值）的gradient。让来表示 total的类。然后可以把写作：

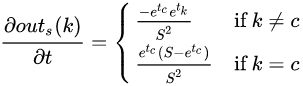
假如有类k，若，则=则由链式法则：



若则：



合并如下：



**你应该根据此公式完成softmax.py中Backprop的第一部分。**

最终是想要计算 weights，biases 和 input 的 gradient对应的loss：

1.我们要使用 weights gradient，，来更新层的 weights。

2.我们要使用 biases gradient，，来更新层的 biases。

3.我们要返回input（每一层的向前传播的输入） 的 gradient即。

为了计算上面 3 个 loss gradient，我们首先需要获取另外 3 个结果：totals（做 softmax 之前的向量，10 个元素）对应 weights，biases 和 input 的 gradient。相关公式如下：



这些 gradient 很容易计算：







根据链式法则把它们放在一起：







其中，

1. **L**：loss 函数
2. **out**：softmax 的输出结果，与 loss 公式直接相关的 概率
3. **t**：softmax 的输入参数，通过 weights，bias 以及 softmax 层的输入来获取。

**你应该根据以上公式完成softmax.py中Backprop的第二部分**

计算出gradient之后，剩下的就是训练 softmax 层。通过 [SGD](https://zhuanlan.zhihu.com/p/357963858)（Stochastic Gradient Decent）来更新weights和bias，并返回d\_L\_d\_inputs。

**1.5反向传播：池化层（Backprop: Max Pooling）**

在forward pass的过程中，Max Pooling 层选取 2x2 块的最大值进行输入，如下图所示：

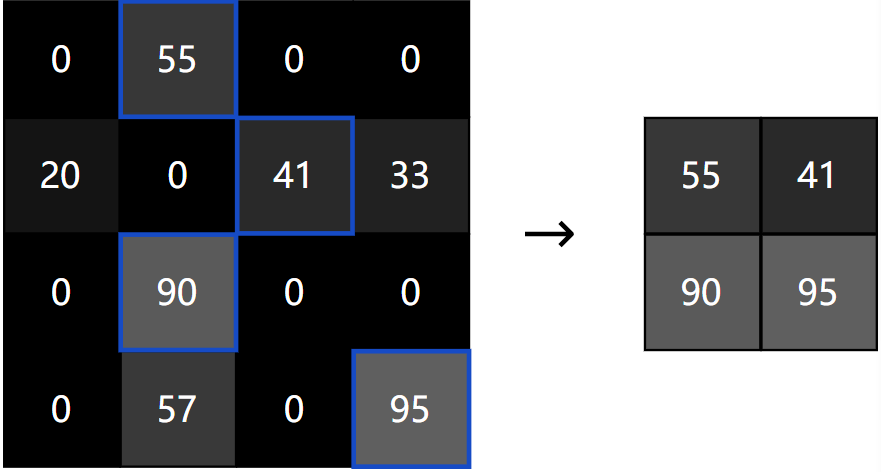
****

图3：将 4x4 输入转换为 2x2 输出的前向传播示例

backward phase 中的相同层如下图所示：

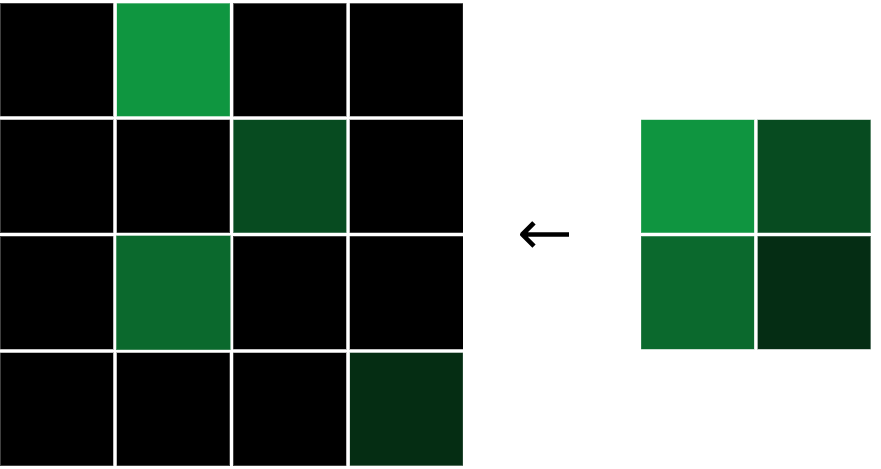


图3：将 2x2 熟人转换为 4x4输出 的反向传播示例

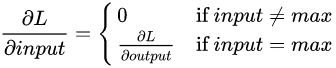
每一个 gradient 的值都被赋值到原始的最大值的位置，其他的值都是 0。在最大池化的backprop中对于非最大值的像素点：



最大值的像素点会将值传递给输出，所以：



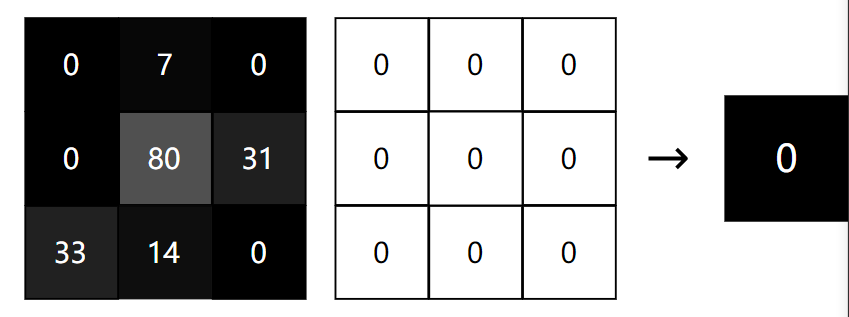
将两个合并即：



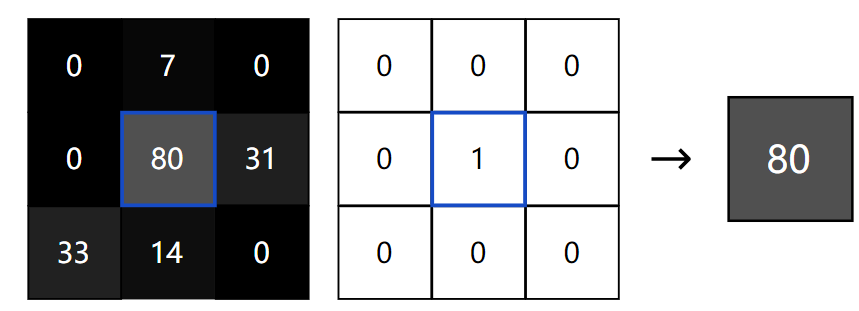
**在这部分你应该完成maxpool.py中的代码。**

**1.7** **反向传播：卷积层（Backprop: Conv）**

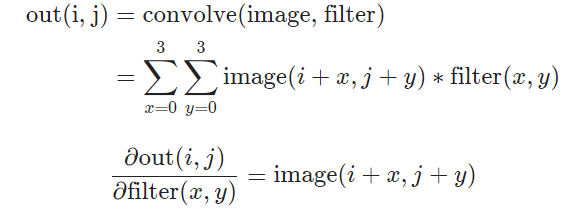
卷积层的反向传播是 CNN 模型训练的核心。在maxpool.py中的到了，在卷积层的**Backprop中**我们需要获取。实际上，改变任何 filter 的 weight 都会影响到整个输出图片的信息，因为在卷积过程中，每一个输出的像素都会使用每一个 filter 的 weight。为了简单起见，我们试想下一次只有一个输出：如何修改 filter 来改变那个具体输出像素的值？下面这个例子有助于我们思考这个问题：



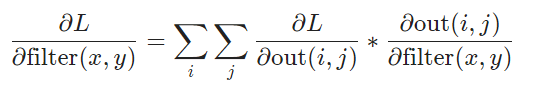
我们有一个 3x3 的图片与一个都是 0 的 3x3 的 filter 进行卷积运算，结果只有一个 1x1 的输出。如果我们把 filter 中间的 weight 增加到 1 呢？输出将会随着中心值来增加到 80：



增加任何 filter 的其他权重到 1，都会最终增加相应的输出图片像素值，这说明一个具体的输出像素对于具体的 filter 的 weight 的 gradient 就是对应的像素值。推导如下：



我们可以把和放在一起，找到特定filter的wight的损失gradient：



你应该根据以上公式完成conv.py中的代码。最后和之前一样通过 [SGD](https://zhuanlan.zhihu.com/p/357963858)（Stochastic Gradient Decent）来更新filters。

**2训练 CNN（Training a CNN）**

在ex3.py的第四部分将对CNN 模型训练3个 epoch，并且在ex3.py的第五部分对测试集进行测试。