深度学习系统第一次作业

黄海浪 ZY2206117*

北京航空航天大学计算机学院

1 人工神经网络原理说明

1.1 神经元

神经元 (neuron) 是神经网络的基本计算单元，也被称作节点 (node) 或者单元 (unit)。它可以接受来自其他神经元的输入或者是外部的数据，然后计算一个输出。每个输入值都有一个权重 (weight)，权重的大小取决于这个输入相比于其他输入值的重要性。然后在神经元上执行一个特定的函数 f，定义如下图所示，这个函数会该神经元的所有输入值以及其权重进行一个操作。

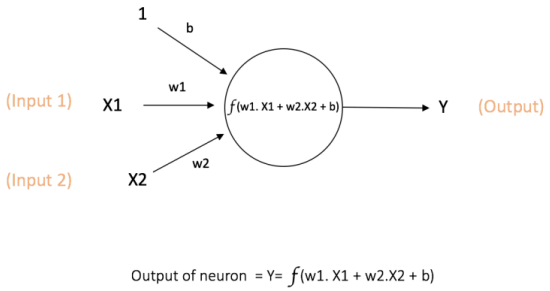

图 1: 神经元

除了权重外，还有一个输入值是 1 的偏置值 bias。这里的函数 f 就是一个被称为激活函数的非线性函数。它的目的是给神经元的输出引入非线性。因为在现实世界中的数据都是非线性的，因此需要神经元可以学习到这些非线性的表示。

1.2 激活函数

1. Sigmoid

   Sigmoid 激活函数的输出范围是 [0,1]，其公式如下：

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

   记 Sigmoid 函数为 S，其导数为 $S*(1-S)$，故选取它为本实验的激活函数。

2. tanh

   tanh 激活函数的输出范围是 [-1,1]，其公式如下：

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = 2\sigma(2x) - 1 \tag{2}$$

3. ReLU

   ReLU 激活函数公式如下：

$$f(x) = \max(0, x) \tag{3}$$

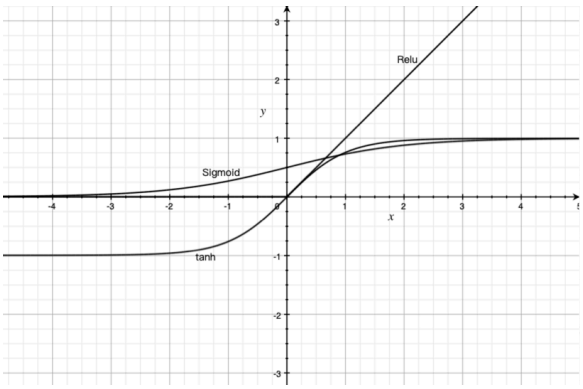

图 2: 激活函数

1.3 前向神经网络

前向神经网络是最简单的一种人工神经网络，图3给出了一个前向神经网络的简单例子示意图。如图3所示，这个神经网络分为 3 个网络层，分别是输入层，隐藏层和输出层，每个网络层都包含有多个神经元，每个神经元都会跟相邻的前一个层的神经元有

*e-mail:lerogo@buaa.edu.cn

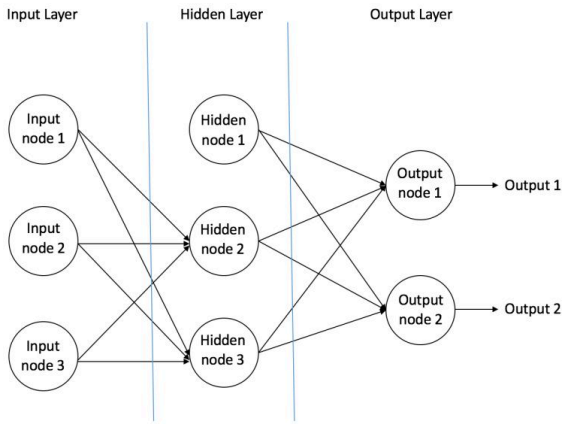

图 3: 前向神经网络

连接，这些连接是该神经元的输入。根据神经元所在层的不同，前向神经网络的神经元分为三种，分别为：

1. 输入神经元：位于输入层，主要是传递来自外界的信息进入神经网络中。
2. 隐藏神经元：位于隐藏层，隐藏层的神经元不与外界有直接的连接，它都是通过前面的输入层和后面的输出层与外界有间接的联系。
3. 输出神经元：位于输出层，输出神经元就是将来自隐藏层的信息输出到外界中，也就是输出最终的结果。

前向网络中，信息是从输入层传递到输出层，只有前向这一个方向。

1.4 反向传播算法

神经网络除了前向传播计算，还有反向传播算法，通过反向传播来更新权值，从而获得更好的训练结果。

反向传播误差，也被简称为反向传播，是用于训练神经网络的方法之一，它是一个有监督学习方法，也就是说它是从训练数据的标签来进行学习，即相当于有一个监督者负责指导它的学习。反向传播算法，在初始阶段，所有权重都是随机分配的。对于训练集的每个输入值，经过神经网络的前向计算后，得到的输出值将会与期望的输出进行比较，然后得到的误差会传回给前面的网络层。这个误差会被记下，然后权重会进行相应的调整。如图4 这个过程会不断重复，直到输出的误差低于一个设定好的阈值。

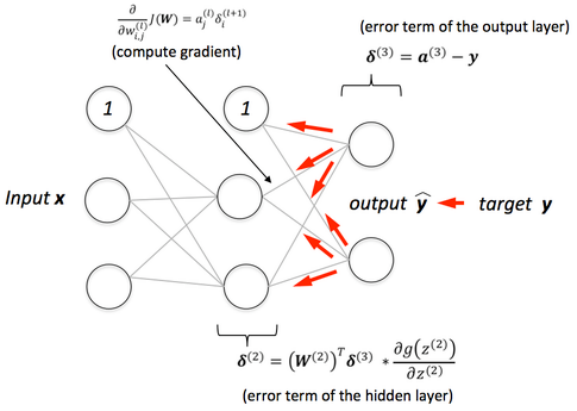

图 4: 反向传播

2 本次实验介绍

2.1 目标任务

手动搭建基本的多层神经网络，识别目标图片中的数字，并在给定数据集上训练测试。

2.2 数据集

实验采用 Mnist 官方数据集，数据文件分为训练集和测试集，每个数据集分别使用两个文件存储图片样本和标签，数据集样本具体如表1所示。

表 1: *Mnist* 数据集

文件名	样本数	文件说明
t10k-images.idx3-ubyte	10000	测试集数据文件
t10k-labels.idx1-ubyt	10000	测试集标签文件
train-images.idx3-ubyte	60000	训练集数据文件
train-labels.idx1-ubyt	60000	训练集标签文件

```
# 初始化纬度和学习率
self.input_nodes = input_nodes
self.hn1 = hidden_nodes_1
self.hn2 = hidden_nodes_2
self.hn3 = hidden_nodes_3
self.output_nodes = output_nodes
self.lr = learning_rate
# 输入
self.inputs_hn1 = None
# 输出
self.hn1_hn2 = None
self.hn2_hn3 = None
self.hn3_outputs = None
self.outputs = None
# 初始化权重 (0.0分布均值, 分布的标准偏差, shape)
self.w_i_h1 = np.random.normal(0.0, pow(self.input_nodes, -0.5), (self.hn1, self.input_nodes))
self.w_h1_h2 = np.random.normal(0.0, pow(self.hn1, -0.5), (self.hn2, self.hn1))
self.w_h2_h3 = np.random.normal(0.0, pow(self.hn2, -0.5), (self.hn3, self.hn2))
self.w_h3_o = np.random.normal(0.0, pow(self.hn3, -0.5), (self.output_nodes, self.hn3))
# 初始化激活函数
self.activation_function = lambda x: 1. / (1 + np.exp(-x))
```

图 5: 模型初始化

```
# 前向传播计算输出
self.inputs_hn1 = np.array(input_feature, ndmin=2).T
self.hn1_hn2 = self.activation_function(np.dot(self.w_i_h1, self.inputs_hn1))
self.hn2_hn3 = self.activation_function(np.dot(self.w_h1_h2, self.hn1_hn2))
self.hn3_outputs = self.activation_function(np.dot(self.w_h2_h3, self.hn2_hn3))
self.outputs = self.activation_function(np.dot(self.w_h3_o, self.hn3_outputs))
```

图 6: 模型前向传播

```
# 反向传播误差 更新权重
# 记sigmoid激活函数为S, S的导数为S*(1-S), 根据链式法则, 负梯度方向更新
targets = np.array(targets_list, ndmin=2).T
output_loss = targets - self.outputs
hidden3_loss = np.dot(self.w_h3_o.T, output_loss)
hidden2_loss = np.dot(self.w_h2_h3.T, hidden3_loss)
hidden1_loss = np.dot(self.w_h1_h2.T, hidden2_loss)

self.w_h3_o += self.lr * np.dot((output_loss * self.outputs * (1.0 - self.outputs)),
                                np.transpose(self.hn3_outputs))
self.w_h2_h3 += self.lr * np.dot((hidden3_loss * self.hn3_outputs * (1.0 - self.hn3_outputs)),
                                 np.transpose(self.hn2_hn3))
self.w_h1_h2 += self.lr * np.dot((hidden2_loss * self.hn2_hn3 * (1.0 - self.hn2_hn3)),
                                 np.transpose(self.hn1_hn2))
self.w_i_h1 += self.lr * np.dot((hidden1_loss * self.hn1_hn2 * (1.0 - self.hn1_hn2)),
                                np.transpose(self.inputs_hn1))

return np.sum(output_loss ** 2)
```

图 7: 模型反向传播

2.3 实验设置

本次实验采用 1 层输入层，1 层输出层，3 层隐藏层，其中隐藏层和输出层均为全连接层。其中输入层有 784 个节点，第一层隐藏层有 512 个节点，第二层隐藏层有 256 个节点，第三层隐藏层有 128 个节点，输出层即图片数字分类，为 0～9，共 10 个节点。

其中激活函数使用 Sigmoid（公式1）激活函数，学习率设置为 0.01，训练轮数为 2。模型初始化、前向传播、反向传播代码分别如图5、图6、图7所示。

3 测试结果

测试结果截图如图8所示，一层输入层，三层隐藏层和一层输出层，训练 160 秒左右，在训练集和测试集上的精度均为 96%。

其中，训练过程保存在附件 zip 的 out.txt 中，训练的参数保存在了 weights 文件夹里面，本文采用 latex 编写，源文件保存在了 latex 文件夹。

```
Epoch 00000 | Loss 3.9763
Epoch 00000 | Loss 2.4226
Epoch 00000 | Loss 1.6688
Epoch 00000 | Loss 1.0910
Epoch 00000 | Loss 0.8224
Epoch 00000 | Loss 0.7059
Epoch 00000 | Loss 0.6537
Epoch 00000 | Loss 0.5523
Epoch 00000 | Loss 0.4965
Epoch 00000 | Loss 0.5271
Epoch 00000 | Loss 0.4331
Epoch 00000 | Loss 0.2959
Training Accuracy: 0.94
Epoch 00001 | Loss 0.3450
Epoch 00001 | Loss 0.3630
Epoch 00001 | Loss 0.3492
Epoch 00001 | Loss 0.3222
Epoch 00001 | Loss 0.3257
Epoch 00001 | Loss 0.3114
Epoch 00001 | Loss 0.3000
Epoch 00001 | Loss 0.2841
Epoch 00001 | Loss 0.2668
Epoch 00001 | Loss 0.3326
Epoch 00001 | Loss 0.2620
Epoch 00001 | Loss 0.1849
Training Accuracy: 0.96
Training time: 165.544s
Testing Accuracy: 0.96
Done.
```

图 8: 测试结果