

中山大学数据科学与计算机学院

移动信息工程专业-人工智能

本科生实验报告

(2017-2018学年秋季学期)

课程名称: Artificial Intelligence

一、实验题目

- 实现Back Propagation Neural Network

二、实验内容

1. 算法原理

- 神经网络:

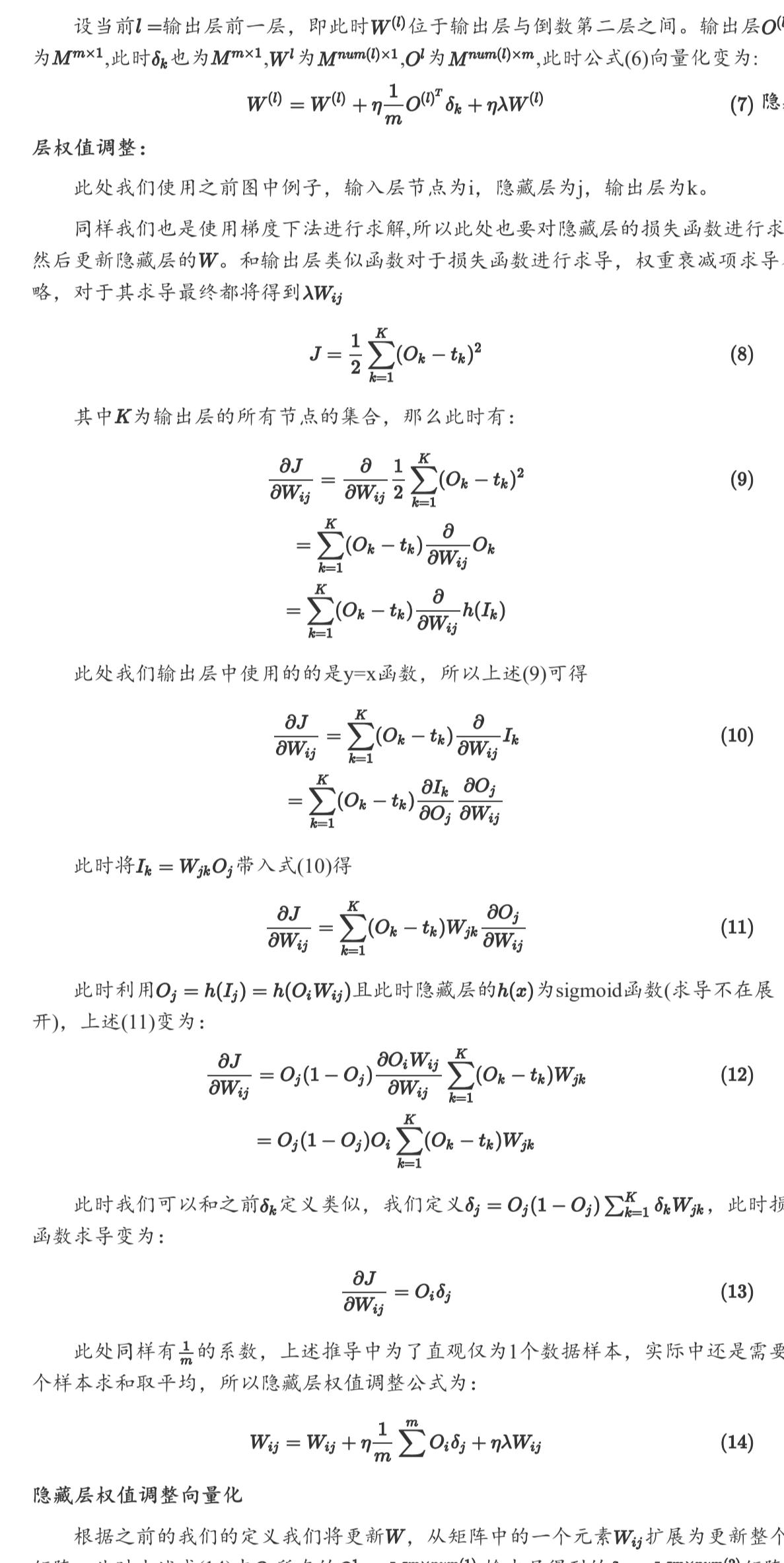
神经网络模型主要是在模拟神经系统的工作方式，由大量节点也就是人工神经元构成。人工神经元由：输入信号 x_i ，代表数据的输入，可以是原始数据的输入也可以是来自其他神经元的数据；权值 W_{ij} 代表神经元和其他神经元之间的连接；一个激活层 $\sum_i w_i x_i$ 对所有的输入信号进行加权求和确定最终输入值，代表最终传入的刺激的大小；一个激活函数 $h(x)$ 根据输入的大小决定是否输出，代表接收刺激之后是否产生兴奋。

神经网络就是多个神经元的级联，上一级神经元的输出是下一级神经元的输入，而且信号在两级的两个神经元之间传播的时候需要乘上这两个神经元对应的权值。

- BP神经网络模型：

BP是后向传播的英文缩写，PB神经网络主要是依靠后向传播误差进行学习。传播的目的是得到所有层的估计误差，后向是说由后层误差推导前层误差。它的学习规则是使用梯度下降法，通过后向传播来不断调整网络的权值和阈值，使网络的误差平方和最小。

BP的思想可以总结为：利用输出后的误差来估计输出层的前一层的误差，再用这个误差估计更前一层的误差，如此一层一层的反传下去，就获得了所有其他各层的误差估计，之后就可以用这些误差更新所有的权值 W 。



使用圆圈来表示神经网络的神经元，+1为偏置节点，也就是截距项。神经网络最左边的一层为输入层，最右的一层为输出层。中间所有节点组成的一层为隐藏层，因为我们不能在训练样本集中观测到它们的值。

我们使用 $O_i^{(l)}$ 表示第 l 层第 i 个节点的输出值，用 $I_i^{(l)}$ 表示第 l 层第 i 个节点的输入值，用 $W_{ij}^{(l)}$ 表示第 l 层第 j 个节点与第 $l+1$ 层的第 i 个节点之间的连接参数也就是权重，用 $b_i^{(l)}$ 表示第 $l+1$ 层的第 i 个节点的偏置，用 $h_w,b(x)$ 表示输出结果。

在此先把这几个参数使用向量化表示 $W^{(l)} = M^{num(l) \times num(l+1)}$, $b^{(l)} = M^{1 \times num(l+1)}$, $O^{(l)} = I^{(l)} = M^{m \times num(l)}$, m 表示数据集大小, $num(l)$ 表示 l 层的节点数量。此处 W 的矩阵设置方式是为之后直接计算时不需要加一步转置，直接又乘即可。

- BP神经网络流程：

- 前向传播：

BP神经网络的思想虽然是后向传播更新误差，再而更新 W 实现学习，但是其并不意味着可以跳过前向传播的学习。因为如果后向传播对应训练的话，那么前向传播就对应预测，并且训练的时候计算误差也要用到预测的输出值来计算误差。

- 对于输入层有 $O_i^{(1)} = x_i I_i^{(1)} = x_i$
- 对于隐藏层和输出层有 $O_j^{(l)} = h(I_j^{(l)})$, $I_j^{(l)} = \sum_i W_{ij}^{(l-1)} O_i^{(l-1)} + b_j^{(l-1)}$, 其中 $h(x)$ 为激活函数

在数据经理了输入层，隐藏层，输出层，之后的输出值就是神经网络模型的预测结果，此时这个结果就可以用于误差计算。

- 前向传播过程的向量化：

假设输入时一个 $n \times m$ 列的矩阵(M)，也就是一个 n 个数据，每个数据有 m 维度的训练集。

- 对于输入层 $O^1 = M$

- 对于隐藏层和输出层 $O^{(l)} = O^{(l-1)}W^{(l-1)} + b^{(l-1)}$, 此处的 $+b^{(l-1)}$ 为在每一行都加上 $b^{(l-1)}$ ，之后 $O^{(l)} = h(I^{(l)})$ 的到改层的输出

- 后向传播：

后向传播指的是在训练的时候，根据最终输出的误差来调整倒数第二层、倒数第三层……第一层的参数的过程。此处我们使用批梯度下降进行求解。

对于单个样例 (x, y) , 其代价函数为：

$$J(W, b; x, y) = \frac{1}{2} \|h_w, b(x) - y\|^2 \quad (1)$$

对于一个 m 个样例的数据集，代价函数为：

$$\begin{aligned} J(W, b) &= \frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)}) + \frac{\lambda}{2} \sum_{i=1}^{n-1} \sum_{j=1}^{s_i} \sum_{j=1}^{s_{i+1}} (W_{ij}^{(i)})^2 \\ &= \frac{1}{m} \sum_{i=1}^m \left(\frac{1}{2} \|h_w, b(x^{(i)}) - y^{(i)}\|^2 \right) + \frac{\lambda}{2} \sum_{i=1}^{n-1} \sum_{j=1}^{s_i} \sum_{j=1}^{s_{i+1}} (W_{ij}^{(i)})^2 \end{aligned} \quad (2)$$

上式中第一项是一个均方差项。第二项是权重衰减项，其目的是减小权重的幅度，防止过度拟合。

- 输出层权值调整：

我们要使(2)达到最小值，按照梯度下降法的理念此时就是对损失函数进行求导。

对于输出层其损失函数可以改为

$$J = MSE = \frac{1}{2m} \sum_{i=1}^m (O_k - t_k)^2 + \frac{\lambda}{2} \sum_{i=1}^{n-1} \sum_{j=1}^{s_i} \sum_{j=1}^{s_{i+1}} (W_{jk}^{(i)})^2 \quad (3) \quad \text{其中 } O_k \text{ 为输出层输出 } t_k \text{ 为训练集的答案。此时求导如下(对于单个样本):}$$

$$\frac{\partial J}{\partial W_{jk}} = \frac{\partial}{\partial W_{jk}} \frac{1}{2} (O_k - t_k)^2 + \lambda W_{jk} \quad (4)$$

$= (O_k - t_k) \frac{\partial}{\partial W_{jk}} O_k + \lambda W_{jk}$ 此

$$= (O_k - t_k) \frac{\partial}{\partial W_{jk}} h(I_k) + \lambda W_{jk}$$

时我们把上式中 $(O_k - t_k)$ 记为 $\delta_k = (O_k - t_k)$ ，此时梯度公式更新公式变成：

$$W_{jk} = W_{jk} + \eta \frac{1}{m} \sum_{i=1}^m \delta_k O_j + \eta \lambda W_{jk} \quad (6) \quad \text{这里额外的 } \frac{1}{m} \text{ 主要是因为有 } m \text{ 个样本每个样本的求导出来的梯度需要进行求和平均然后用于梯度更新。}$$

下表为使用sigmoid、relu、tanh作为激活函数时的输出层求导的公式，主要是根据(4)然后对激活函数求导，具体求导过程省略。

sigmoid	$\frac{\partial J}{\partial W_{jk}} = O_k(1 - O_k)(O_k - t_k)O_j + \lambda W_{jk}$	(1)
relu	$\frac{\partial J}{\partial W_{jk}} = \begin{cases} (O_k - t_k)O_j + \lambda W_{jk}, & I_k \geq 0 \\ \alpha(O_k - t_k)O_j + \lambda W_{jk}, & I_k < 0 \end{cases}$	(2)
tanh	$\frac{\partial J}{\partial W_{jk}} = (1 - O_k^2)(O_k - t_k)O_j + \lambda W_{jk}$	(3)

输出层权值调整公式向量化：

设当前 $l=$ 输出层前一层，即此时 $W^{(l)}$ 位于输出层与倒数第二层之间。输出层 $O^{(l+1)}$ 为 $M^{m \times 1}$, 此时 δ_l 也为 $M^{m \times 1}$, $W^{(l)}$ 为 $M^{num(l) \times 1}$, $O^{(l)}$ 为 $M^{num(l) \times m}$, 此时公式(6)向量化变为：

$$W^{(l)} = W^{(l)} + \eta \frac{1}{m} O^{(l)^T} \delta_l + \eta \lambda W^{(l)} \quad (7) \quad \text{隐藏}$$

层权值调整：

此处我们使用之前图中例子，输入层节点为 i ，隐藏层为 j ，输出层为 k 。

同样我们也是使用梯度下降法进行求解, 所以此处也要对隐藏层的损失函数进行求导，然后更新隐藏层的 W 。和输出层类似函数对于损失函数进行求导，权重衰减项求导省略，对于其求导最终都将得到 λW_{jk}

$$J = \frac{1}{2} \sum_{k=1}^K (O_k - t_k)^2 + \frac{\lambda}{2} \sum_{i=1}^{n-1} \sum_{j=1}^{s_i} \sum_{k=1}^{s_{i+1}} (W_{jk}^{(i)})^2 \quad (8)$$

其中 K 为输出层的所有节点的集合，那么此时有：

$$\begin{aligned} \frac{\partial J}{\partial W_{ij}} &= \frac{\partial}{\partial W_{ij}} \frac{1}{2} \sum_{k=1}^K (O_k - t_k)^2 + \lambda W_{jk} \\ &= \sum_{k=1}^K (O_k - t_k) \frac{\partial}{\partial W_{ij}} O_k \end{aligned} \quad (9)$$

$$= \sum_{k=1}^K (O_k - t_k) \frac{\partial}{\partial O_j} \frac{\partial O_j}{\partial W_{ij}}$$

此时将 $I_k = W_{jk} O_j$ 带入式(10)得

$$\frac{\partial J}{\partial W_{ij}} = \sum_{k=1}^K (O_k - t_k) W_{jk} \frac{\partial O_j}{\partial W_{ij}} \quad (10)$$

此时利用 $O_j = h(I_j) = h(O_i W_{ij})$ 且此时隐藏层的 $h(x)$ 为sigmoid函数(求导不在展开)，上述(11)变为：

$$\frac{\partial J}{\partial W_{ij}} = O_j(1 - O_j) \frac{\partial}{\partial W_{ij}} h(O_i W_{ij}) + \lambda W_{jk} \quad (5) \quad \text{此}$$

$$= (O_k - t_k) O_j + \lambda W_{jk}$$

时我们把上式中 $(O_k - t_k)$ 记为 $\delta_k = (O_k - t_k)$ ，此时梯度公式更新公式变成：

$$W_{jk} = W_{jk} + \eta \frac{1}{m} \sum_{i=1}^m \delta_k O_j + \eta \lambda W_{jk} \quad (6) \quad \text{这}$$

里额外的 $\frac{1}{m}$ 主要是因为有 m 个样本每个样本的求导出来的梯度需要进行求和平均然后用于梯度更新。

下表为使用sigmoid、relu、tanh作为激活函数时的输出层求导的公式，主要是根据(4)然后对激活函数求导，具体求导过程省略。

sigmoid	$\frac{\partial J}{\partial W_{jk}} = O_k(1 - O_k)(O_k - t_k)O_j + \lambda W_{jk}$	(1)
relu	$\frac{\partial J}{\partial W_{jk}} = \begin{cases} (O_k - t_k)O_j + \lambda W_{jk}, & I_k \geq 0 \\ \alpha(O_k - t_k)O_j + \lambda W_{jk}, & I_k < 0 \end{cases}$	(2)
tanh	$\frac{\partial J}{\partial W_{jk}} = (1 - O_k^2)(O_k - t_k)O_j + \lambda W_{jk}$	(3)

输出层权值调整公式向量化：

设当前 $l=$ 输出层前一层，即此时 $W^{(l)}$ 位于输出层与倒数第二层之间。输出层 $O^{(l+1)}$ 为 $M^{m \times 1}$, 此时 δ_l 也为 $M^{m \times 1}$, $W^{(l)}$ 为 $M^{num(l) \times 1}$, $O^{(l)}$ 为 $M^{num(l) \times m}$, 此时公式(6)向量化变为：

$$W^{(l)} = W^{(l)} + \eta \frac{1}{m} O^{(l)^T} \delta_l + \eta \lambda W^{(l)} \quad (7) \quad \text{隐藏}$$

层权值调整：

此处我们使用之前图中例子，输入层节点为 i ，隐藏层为 j ，输出层为 k 。

同样我们也是使用梯度下降法进行求解, 所以此处也要对隐藏层的损失函数进行求导，然后更新隐藏层的 W 。和输出层类似函数对于损失函数进行求导，权重衰减项求导省略，对于其求导最终都将得到 λW_{jk}

$$J = \frac{1}{2} \sum_{k=1}^K (O_k - t_k)^2 + \frac{\lambda}{2} \sum_{i=1}^{n-1} \sum_{j=1}^{s_i} \sum_{k=1}^{s_{i+1}} (W_{jk}^{(i)})^2 \quad (8)$$

其中 K 为输出层的所有节点的集合，那么此时有：

$$\begin{aligned} \frac{\partial J}{\partial W_{ij}} &= \frac{\partial}{\partial W_{ij}} \frac{1}{2} \sum_{k=1}^K (O_k - t_k)^2 + \lambda W_{jk} \\ &= \sum_{k=1}^K (O_k - t_k) \frac{\partial}{\partial W_{ij}} O_k \end{aligned} \quad (9)$$

$$= \sum_{k=1}^K (O_k - t_k) \frac{\partial}{\partial O_j} \frac{\partial O_j}{\partial W_{ij}}$$

此时将 $I_k = W_{jk} O_j$ 带入式(10)得

$$\frac{\partial J}{\partial W_{ij}} = \sum_{k=1}^K (O_k - t_k) W_{jk} \frac{\partial O_j}{\partial W_{ij}} \quad (11)$$

此时利用 $O_j = h(I_j) = h(O_i W_{ij})$ 且此时隐藏层的 $h(x)$ 为sigmoid函数(求导不在展开)，上述(11)变为：

$$\frac{\partial J}{\partial W_{ij}} = O_j(1 - O_j) \frac{\partial}{\partial W_{ij}} h(O_i W_{ij}) + \lambda W_{jk} \quad (5) \quad \text{此}$$

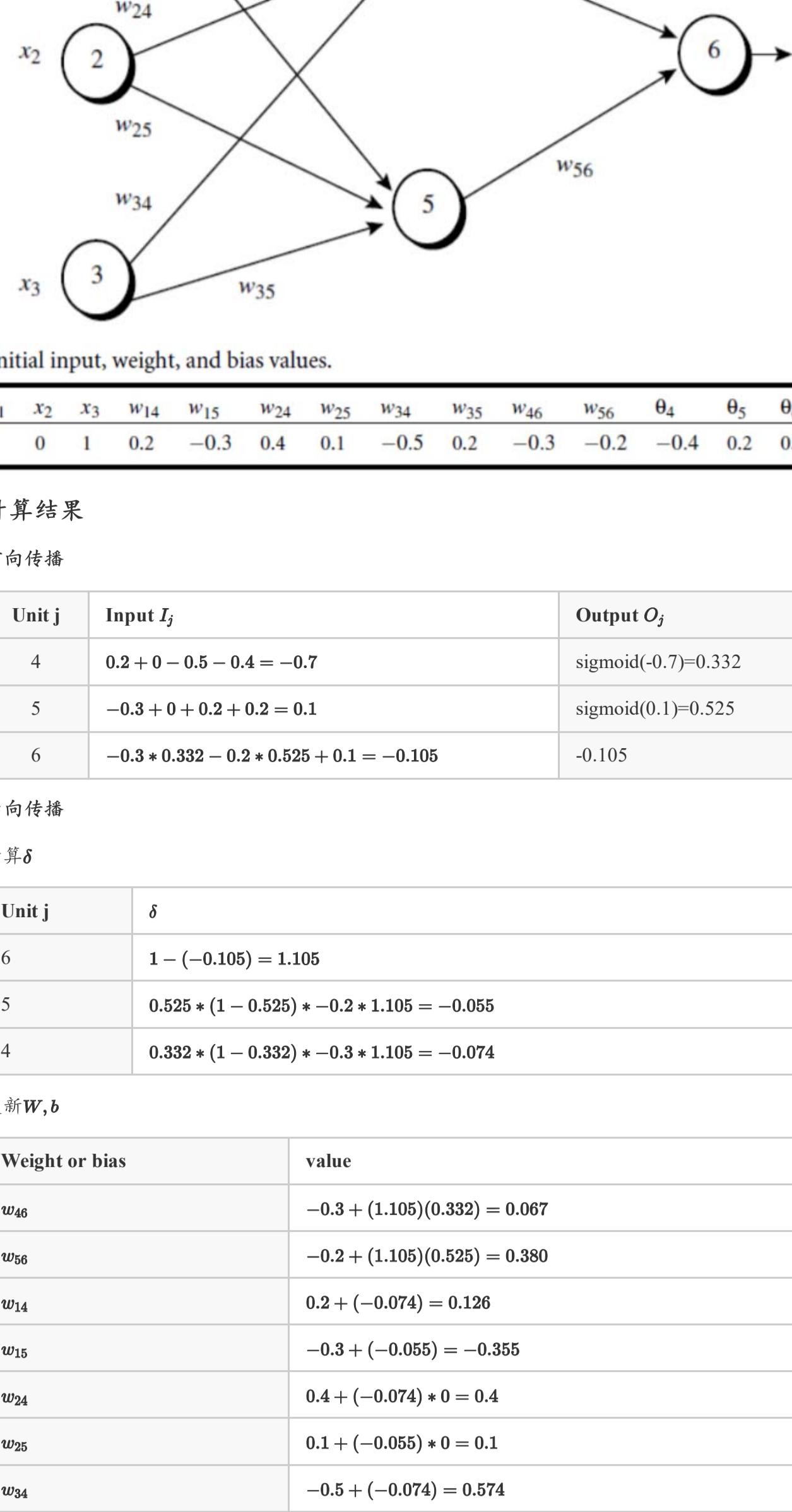
$$= (O_k - t_k) O_j + \lambda W_{jk}$$

时我们把上式中 $(O_k - t_k)$ 记为 $\delta_k = (O_k - t_k)$ ，此时梯度公式更新公式变成：

$$W_{jk} = W_{jk} + \eta \frac{1}{m} \sum_{i=1}^m \delta_k O_j + \eta \lambda W_{jk} \quad (6) \quad \text{这}$$

层权值调整：

此处我们使用之前图中例子，输入层节点为 i ，隐藏层为 j



Initial input, weight, and bias values.

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	0.2	-0.3	-0.2	-0.4	0.2

计算结果

前向传播

Unit j	Input I_j	Output O_j
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$\text{sigmoid}(-0.7) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.1 = 0.1$	$\text{sigmoid}(0.1) = 0.525$
6	$-0.3 * 0.332 - 0.2 * 0.525 + 0.1 = -0.105$	-0.105

后向传播

计算 δ

Unit j	δ
6	$1 - (-0.105) = 1.105$
5	$0.525 * (1 - 0.525) * -0.2 * 1.105 = -0.055$
4	$0.332 * (1 - 0.332) * -0.3 * 1.105 = -0.074$

更新 W, b

Weight or bias	value
w_{46}	$-0.3 + (1.105)(0.332) = 0.067$
w_{56}	$-0.2 + (1.105)(0.525) = 0.380$
w_{14}	$0.2 + (-0.074) = 0.126$
w_{15}	$-0.3 + (-0.055) = -0.355$
w_{24}	$0.4 + (-0.074) * 0 = 0.4$
w_{25}	$0.1 + (-0.055) * 0 = 0.1$
w_{34}	$-0.5 + (-0.074) = 0.574$
w_{35}	$0.2 + (-0.055) = 0.145$
θ_6	$0.1 + (1.105) = 1.205$
θ_5	$0.2 + (-0.055) = 0.145$
θ_4	$-0.4 + (-0.074) = -0.474$

程序运行结果验证

此处结果为矩阵化结果，定义参照实验原理

前向传播

out{1} 第一项为 Unit4 的输出，第二项为 Unit5 的输出，out{2} 为 Unit6 的输出

```
>> out[1]
ans =
0.331812227831834  0.524979187478940
>> out[2]
ans =
-0.104539505845338
```

后向传播

计算结果，其中 δ {1} 第一项为 Unit4 的 δ ，第二项为 Unit5 的 δ ， δ {2} 为 Unit6 的 δ

```
>> delta[1]
ans =
-0.473467188252016 -0.055089137640865
>> delta[2]
ans =
1.104539505845338
```

更新后的 W，其中 w{1} 第一行第一项为 w_{14} 第二项为 w_{15} 。以此类推。

```
>> w[1]
ans =
0.126532811747984 -0.355089137640865
0.4000000000000000 0.1000000000000000
-0.573467188252016 0.144910862359135
>> w[2]
ans =
0.066499714162815
0.379860252317076
>> |
```

更新后的 b，其中 theta{1} 第一项为 θ_4 ，第二项为 θ_5 ， δ {2} 为 θ_6

```
>> theta[1]
ans =
-0.473467188252016 0.144910862359135
>> theta[2]
ans =
1.204539505845338
>> |
```

2. 评测指标展示即分析

1. 原始神经网络

使用 1-11 月作为训练集，12 月作为验证集，无任何优化，输出层激活函数为 $y=x$

Loss function



此处可见在 10 折交叉验证的情况下在一定迭代次数内模型不会出现过拟合且 loss function 最终都降至 2000 以下。

3. 数据集处理

此处使用处理过后的数据集加之多层次神经网络，进行最终的预测，由于最终预测需要用到 12 月的最后 20 天，就直接使用 12 月作为验证集进行调参，至于 10 折交叉验证结果就不放了（CUDA 跑一折都需要 2 分钟）。

按照要求使用 1-11 月作为训练集，12 月作为验证集进行测试。

Loss function

此处我们可以看到此时的 Loss function 在没有经过调参和任何优化时结果全部都在 10000 以上，此时预测结果和实际结果也有很大的偏差。

2. 有什么方法可以实现传递过程中不激活所有节点？

将与其相关的 W 加上一个权值，在前向传播的计算过程和在后向传播更新 W 的时需要乘上这个权值，如果我们不想更新这个节点我们将其置 0，或者进行随机置 0，随机关掉这个节点。

同时在训练的时候发现如果我们依然使用 $y=x$ 作为输出层的激活函数此时因为负数部分 δ 依然很大，会把结果带入往下拉导致预测出很多负值，所以此处该用 ReLU 作为输出层激活函数，此时会控制负数部分的 δ 值进而控制此部分对梯度的影响从而实现负值的优化。

只要我们不把 W 初始设得很大一般不会出现这种问题。

此处我们可以看到此时的 Loss function 降至 1000 以下，经过测试在迭代次数继续上升之后会出现过拟合的问题，当迭代次数控制在 5W 次左右的时候可以达到一个较优解，此时的预测结果基本上实际值和预测值相差不大。说明使用层数上升和增加节点之后对于神经网络的训练效果有很大的提升。

同时在训练的时候发现如果我们依然使用 $y=x$ 作为输出层的激活函数此时因为负数部分 δ 依然很大，会把结果带入往下拉导致预测出很多负值，所以此处该用 ReLU 作为输出层激活函数，此时会控制负数部分的 δ 值进而控制此部分对梯度的影响从而实现负值的优化。

四、思考题

1. 尝试说明下其他激活函数的优缺点。

激活函数	优点	缺点
sigmoid	能把数据压缩到 0,1 区间内，在使用中相对于其他线性的激活函数比较不容易出现过了一个较大的梯度之后，直接使得 W 变成 NAN 也就是无穷，变成 NAN 这个节点基本报废，会导致之后一系列节点出现 NAN。	当输入为极大或者极小时其逼近激活函数得出的梯度接近 0，使得出现学习困难，再加之我们很小的学习率，很容易出现传播几层之后出现梯度消失。
tanh	这个和 sigmoid 差不多，不过效果比 sigmoid 好一点，大概也就 loss function 降低一点，但是不是很明显。	和 sigmoid 类似
ReLU	收敛速度快，输出层能将结果映射回我们需要的范围更适合描述实数。减少梯度消失的出现情况。	当出现一个大梯度时，其更新得到的 W 会比前两种激活函数大很多，此时其对之后的数据基本不会有激活现象。

2. 有什么方法可以实现传递过程中不激活所有节点？

将与其相关的 W 加上一个权值，在前向传播的计算过程和在后向传播更新 W 的时需要乘上这个权值，如果我们不想更新这个节点我们将其置 0，或者进行随机置 0，随机关掉这个节点。

同时在训练的时候发现如果我们依然使用 $y=x$ 作为输出层的激活函数此时因为负数部分 δ 依然很大，会把结果带入往下拉导致预测出很多负值，所以此处该用 ReLU 作为输出层激活函数，此时会控制负数部分的 δ 值进而控制此部分对梯度的影响从而实现负值的优化。

只要我们不把 W 初始设得很大一般不会出现这种问题。

此处我们可以看到此时的 Loss function 降至 1000 以下，经过测试在迭代次数继续上升之后会出现过拟合的问题，当迭代次数控制在 5W 次左右的时候可以达到一个较优解，此时的预测结果基本上实际值和预测值相差不大。说明使用层数上升和增加节点之后对于神经网络的训练效果有很大的提升。

同时在训练的时候发现如果我们依然使用 $y=x$ 作为输出层的激活函数此时因为负数部分 δ 依然很大，会把结果带入往下拉导致预测出很多负值，所以此处该用 ReLU 作为输出层激活函数，此时会控制负数部分的 δ 值进而控制此部分对梯度的影响从而实现负值的优化。

只要我们不把 W 初始设得很大一般不会出现这种问题。

此处我们可以看到此时的 Loss function 降至 1000 以下，经过测试在迭代次数继续上升之后会出现过拟合的问题，当迭代次数控制在 5W 次左右的时候可以达到一个较优解，此时的预测结果基本上实际值和预测值相差不大。说明使用层数上升和增加节点之后对于神经网络的训练效果有很大的提升。

同时在训练的时候发现如果我们依然使用 $y=x$ 作为输出层的激活函数此时因为负数部分 δ 依然很大，会把结果带入往下拉导致预测出很多负值，所以此处该用 ReLU 作为输出层激活函数，此时会控制负数部分的 δ 值进而控制此部分对梯度的影响从而实现负值的优化。

只要我们不把 W 初始设得很大一般不会出现这种问题。

此处我们可以看到此时的 Loss function 降至 1000 以下，经过测试在迭代次数继续上升之后会出现过拟合的问题，当迭代次数控制在 5W 次左右的时候可以达到一个较优解，此时的预测结果基本上实际值和预测值相差不大。说明使用层数上升和增加节点之后对于神经网络的训练效果有很大的提升。

同时在训练的时候发现如果我们依然使用 $y=x$ 作为输出层的激活函数此时因为负数部分 δ 依然很大，会把结果带入往下拉导致预测出很多负值，所以此处该用 ReLU 作为输出层激活函数，此时会控制负数部分的 δ 值进而控制此部分对梯度的影响从而实现负值的优化。

只要我们不把 W 初始设得很大一般不会出现这种问题。

此处我们可以看到此时的 Loss function 降至 1000 以下，经过测试在迭代次数继续上升之后会出现过拟合的问题，当迭代次数控制在 5W 次左右的时候可以达到一个较优解，此时的预测结果基本上实际值和预测值相差不大。说明使用层数上升和增加节点之后对于神经网络的训练效果有很大的提升。

同时在训练的时候发现如果我们依然使用 $y=x$ 作为输出层的激活函数此时因为负数部分 δ 依然很大，会把结果带入往下拉导致预测出很多负值，所以此处该用 ReLU 作为输出层激活函数，此时会控制负数部分的 δ 值进而控制此部分对梯度的影响从而实现负值的优化。

只要我们不把 W 初始设得很大一般不会出现这种问题。

此处我们可以看到此时的 Loss function 降至 1000 以下，经过测试在迭代次数继续上升之后会出现过拟合的问题，当迭代次数控制在 5W 次左右的时候可以达到一个较优解，此时的预测结果基本上实际值和预测值相差不大。说明使用层数上升和增加节点之后对于神经网络的训练效果有很大的提升。

同时在训练的时候发现如果我们依然使用 $y=x$ 作为输出层的激活函数此时因为负数部分 δ 依然很大，会把结果带入往下拉导致预测出很多负值，所以此处该用 ReLU 作为输出层激活函数，此时会控制负