

人工智能基础实验报告

刘砺志

(2014 级计算机 1 班 22920142203873)

本文是人工智能基础——神经网络实验的实验报告。

1 实验概述

本次实验要求实现人工神经网络的反向传播算法，设置合适的参数，对样本进行训练，给出训练结果。

2 实验原理

2.1 描述

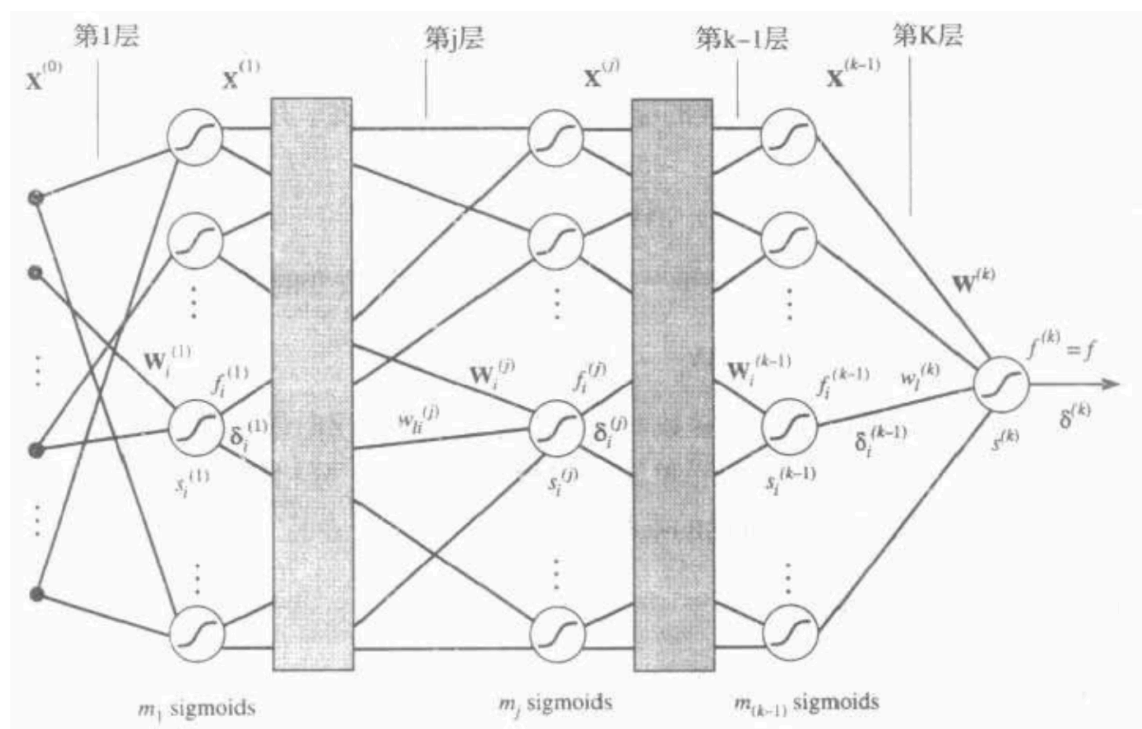


图 1 k 层 sigmoid 神经网络

现有 k 层 sigmoid 神经网络，第 0 层为输入层，第 1 到第 $(k-1)$ 层为隐含层，第 k 层为输出层。符号描述如下：

- k : 网络层数
- m_j : 第 j 层中 sigmoid 神经元的个数
- $\mathbf{X}^{(0)}$: 输入向量（是增广向量!!! 最后一个元素为 1）
- $\mathbf{X}^{(j)} (1 \leq j \leq k)$: 第 j 层所有 sigmoid 神经元的输出（是增广向量!!! 最后一个元素为 1）
- $\mathbf{W}_i^{(j)} (1 \leq j \leq k)$: 输入进第 j 层第 i 个 sigmoid 神经元的权向量（是增广向量!!! 包括了阈值）， $\mathbf{W}_i^{(j)} = (w_{1i}^{(j)}, w_{2i}^{(j)}, \dots, w_{m_{j-1},i}^{(j)}, w_{m_{j-1}+1,i}^{(j)})$ ，即向量维数为前一层神经元的个数加一
- $s_i^{(j)}$: 第 j 层第 i 个 sigmoid 神经元的加权总和输入， $s_i^{(j)} = \mathbf{X}^{(j-1)} \cdot \mathbf{W}_i^{(j)}$
- $f_i^{(j)}$: 第 j 层第 i 个 sigmoid 神经元的实际输出， $f_i^{(j)} = f(s_i^{(j)})$ ， f 为 sigmoid 函数。且 $\mathbf{X}^{(j)} = (f_1^{(j)}, f_2^{(j)}, \dots, f_{m_j}^{(j)}, 1)$

用形式化的语言描述，sigmoid 神经网络可以表示为三元组

$$(k, (m_j), (\mathbf{W}_i^{(j)}))$$

即神经网络由层数、每一层神经元的个数以及每个神经元的权值向量所确定。

2.2 反向传播

若 f 为神经网络的实际输出， d 为期望输出，则误差函数依然可以表示为

$$\epsilon = (d - f)^2$$

因为 $s_i^{(j)} = \mathbf{X}^{(j-1)} \cdot \mathbf{W}_i^{(j)}$ ，类似的，有

$$\begin{aligned} \frac{\partial \epsilon}{\partial \mathbf{W}_i^{(j)}} &= \frac{\partial \epsilon}{\partial f} \cdot \frac{\partial f}{\partial s_i^{(j)}} \cdot \frac{\partial s_i^{(j)}}{\partial \mathbf{W}_i^{(j)}} \\ &= -2(d - f) \frac{\partial f}{\partial s_i^{(j)}} \mathbf{X}^{(j-1)} \end{aligned}$$

令

$$\delta_i^{(j)} = (d - f) \frac{\partial f}{\partial s_i^{(j)}}$$

于是

$$\frac{\partial \epsilon}{\partial \mathbf{W}_i^{(j)}} = -2\delta_i^{(j)} \mathbf{X}^{(j-1)}$$

所以，权值更新公式为

$$\mathbf{W}^{(j)} \leftarrow \mathbf{W}^{(j)} + c_i^{(j)} \delta_i^{(j)} \mathbf{X}^{(j-1)}$$

2.2.1 最后一层

由于最后一层（即第 k 层，输出层）仅有一个神经元，则 δ 公式可以简化为

$$\delta^{(k)} = (d - f) \frac{\partial f}{\partial s^{(k)}}$$

根据 sigmoid 函数的性质， $\frac{\partial f}{\partial s^{(k)}} = f(1 - f)$ ，所以

$$\delta^{(k)} = (d - f)f(1 - f)$$

所以最后一层的权值更新公式为

$$\mathbf{W}^{(k)} \leftarrow \mathbf{W}^{(k)} + c^{(k)}(d - f)f(1 - f)\mathbf{X}^{(k-1)}$$

2.2.2 中间层

首先

$$\begin{aligned} \delta_i^{(j)} &= (d - f) \frac{\partial f}{\partial s_i^{(j)}} \\ &= (d - f) \left[\frac{\partial f}{\partial s_1^{(j+1)}} \frac{\partial s_1^{(j+1)}}{\partial s_i^{(j)}} + \frac{\partial f}{\partial s_2^{(j+1)}} \frac{\partial s_2^{(j+1)}}{\partial s_i^{(j)}} + \cdots + \frac{\partial f}{\partial s_{m_{j+1}}^{(j+1)}} \frac{\partial s_{m_{j+1}}^{(j+1)}}{\partial s_i^{(j)}} \right] \\ &= \sum_{l=1}^{m_{j+1}} (d - f) \frac{\partial f}{\partial s_l^{(j+1)}} \frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} \\ &= \sum_{l=1}^{m_{j+1}} \delta_l^{(j+1)} \frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} \end{aligned}$$

而

$$\begin{aligned} s_l^{(j+1)} &= \mathbf{X}^{(j)} \cdot \mathbf{W}_l^{(j+1)} \\ &= \sum_{v=1}^{m_j+1} f_v^{(j)} w_{vl}^{(j+1)} \end{aligned}$$

所以

$$\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} = \sum_{v=1}^{m_j+1} w_{vl}^{(j+1)} \frac{\partial f_v^{(j)}}{\partial s_i^{(j)}}$$

而

$$\frac{\partial f_v^{(j)}}{\partial s_i^{(j)}} = \begin{cases} f_v^{(j)}(1 - f_v^{(j)}), v = i \\ 0, v \neq i \end{cases}$$

所以

$$\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} = w_{il}^{(j+1)} f_i^{(j)}(1 - f_i^{(j)})$$

因而

$$\begin{aligned}\delta_i^{(j)} &= \sum_{l=1}^{m_{j+1}} \delta_l^{(j+1)} w_{il}^{(j+1)} f_i^{(j)} (1 - f_i^{(j)}) \\ &= f_i^{(j)} (1 - f_i^{(j)}) \sum_{l=1}^{m_{j+1}} \delta_l^{(j+1)} w_{il}^{(j+1)}\end{aligned}$$

所以权值更新公式为

$$\mathbf{W}_i^{(j)} \leftarrow \mathbf{W}_i^{(j)} + c_i^{(j)} \delta_i^{(j)} \mathbf{X}^{(j-1)}$$

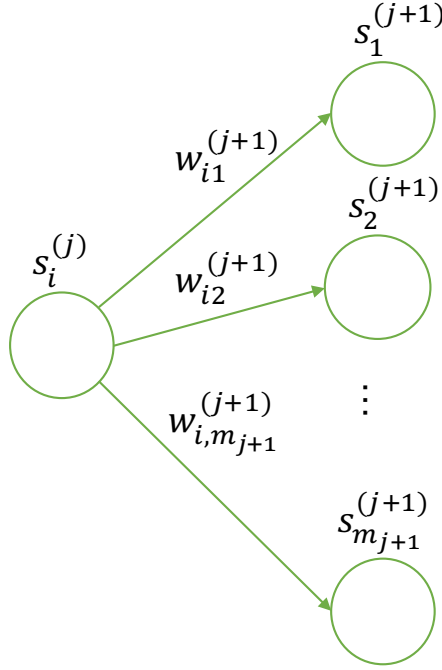


图 2 反向传播过程

3 实验结果

3.1 二分类问题

首先以一个经典的异或（XOR）问题作为例子进行说明。数据集为 4×2 的矩阵，即包含 4 个 2 维样本，它们分别是 (1,0), (0,0), (0,1), (1,1)。期望的输出结果为 0,1,0,1。设定神经网络的层数为 2（不含输入层），中间层神经元个数为 2，输出层神经元个数为 1，学习率为 0.9，收敛阈值为 0.05，最大迭代次数为 5000。我使用 Matlab R2016b 编写了程序，训练部分程序见neural_network_train.m，测试程序见neural_network_test.m，主程序见neural_network_main_1.m。从neural_network_main_1.m开始运行程序，训练过程如图3所示。可以看到，训练误差不断减小，最后到达收敛阈值 0.05，训练结束。用训练数据进行测试，测试的准确率为 100%。

```
Command Window
error rate: 0.059705
error rate: 0.058950
error rate: 0.058188
error rate: 0.057420
error rate: 0.056646
error rate: 0.055867
error rate: 0.055084
error rate: 0.054298
error rate: 0.053509
error rate: 0.052717
error rate: 0.051925
error rate: 0.051132
error rate: 0.050340
error rate: 0.049549
accuracy: 1.000000
fx >>
```

图 3 异或问题程序运行过程

下面测试一个复杂的例子。我选取 UCI 数据集中由台湾新竹成功大学贡献的 Blood Transfusion Service Center 数据集作为测试样例 [2]。设定神经网络的层数为 2（不含输入层），中间层神经元个数为 5，输出层神经元个数为 1，学习率为 1.1，收敛阈值为 0.1，最大迭代次数为 100。主程序见neural_network_main_2.m，我设计了十折交叉检验进行测试，训练过程如图4所示。可见，该网络的平均准确度为 76.20%。

```
Command Window
error rate: 0.101251
error rate: 0.101251
error rate: 0.104582
error rate: 0.099523
accuracy: 0.773333
accuracy: 0.826667
accuracy: 0.826667
accuracy: 0.666667
accuracy: 0.773333
accuracy: 0.720000
accuracy: 0.706667
accuracy: 0.773333
accuracy: 0.800000
accuracy: 0.753425
average accuracy: 0.762009
fx >>
```

图 4 UCI 数据集测试程序运行过程

最后，再来测试两个规模更大的例子。第一个测试数据集选自 UCI 数据集的 Diabetic Retinopathy Debrecen Data Set[3]。首先我对数据进行了归一化。设定神经网络的层数为 2（不含输入层），中间层神经元个数为 6，输出层神经元个数为 1，学习率为 1.2，收敛阈值为 0.08，最大迭代次数为 2000。主程序见neural_network_main_3.m，我设计了十折交叉检验进行测试，训练过程如图5所示。可见，该网络的平均准确度为 68.82%。显然，神经网络对于这个问题的解决效果一般，当然这和参数的选取、训练的次数不大有关。

第二个测试数据集选自 UCI 数据集中的 Breast Cancer Wisconsin (Original) Data Set[4]。首先我对数据进行了归一化。设定神经网络的层数为 2（不含输入层），中间层神经元个数为 10，输出层神经元个数为 1，学习率为 1.0，收敛阈值为 0.05，最大迭代次数为 10000。主程序见neural_network_main_4.m。训练过程如图6所示。可见，该网络的平均准确度为 77.10%，同样效果一般。

```
Command Window
error rate: 0.080194
error rate: 0.080082
error rate: 0.080097
error rate: 0.079937
accuracy: 0.773913
accuracy: 0.652174
accuracy: 0.626087
accuracy: 0.669565
accuracy: 0.695652
accuracy: 0.652174
accuracy: 0.704348
accuracy: 0.704348
accuracy: 0.782609
accuracy: 0.620690
average accuracy: 0.688156
fx >>
```

图 5 较大规模 UCI 数据集一测试程序运行过程

```
Command Window
error rate: 0.051029
error rate: 0.050537
error rate: 0.050311
error rate: 0.049691
accuracy: 0.766234
accuracy: 0.792208
accuracy: 0.714286
accuracy: 0.831169
accuracy: 0.740260
accuracy: 0.675325
accuracy: 0.818182
accuracy: 0.805195
accuracy: 0.740260
accuracy: 0.826667
average accuracy: 0.770978
fx >>
```

图 6 较大规模 UCI 数据集二测试程序运行过程

3.2 多分类问题

对于多分类问题，我采用 one-vs-all 策略，即在训练时，对于某个类，令其为 1，其他类统统认为 0；在测试时，认为逻辑斯蒂函数值最高的那个网络所对应的类为应划分入的类。

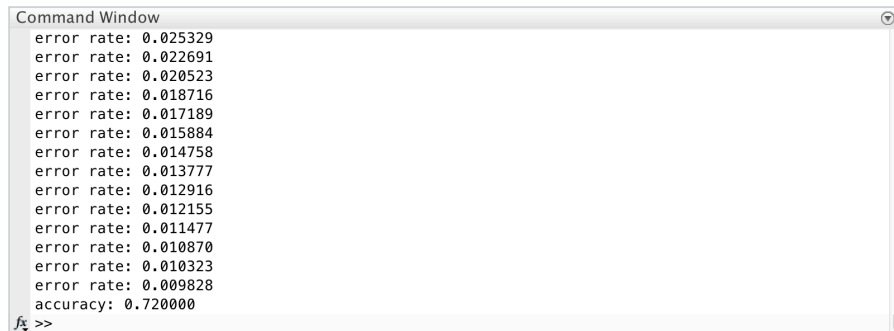
首先来测试提供的 ORL 测试集。首先我对数据进行了归一化。设定神经网络的层数为 3（不含输入层），中间层神经元个数为 15 和 5，输出层神经元个数为 1，学习率为 0.1，收敛阈值为 0.01，最大迭代次数为 100。主程序见neural_network_main_5.m。训练过程如图7所示。可见，该网络的准确度却只有 3.5%。

```
Command Window
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
error rate: 0.012192
accuracy: 0.035000
fx >>
```

图 7 ORL 数据集测试结果

下面测试著名的 Iris 数据集 [5]。首先我对数据进行了归一化。设定神经网络的层数为 2（不含输入层），中间层神经元个数为 5，输出层神经元个数为 1，学习率为 0.01，收敛阈值为

0.01, 最大迭代次数为 1000。主程序见neural_network_main_6.m。训练过程如图8所示。可见, 该网络的准确度为 72.00%。



```
Command Window
error rate: 0.025329
error rate: 0.022691
error rate: 0.020523
error rate: 0.018716
error rate: 0.017189
error rate: 0.015884
error rate: 0.014758
error rate: 0.013777
error rate: 0.012916
error rate: 0.012155
error rate: 0.011477
error rate: 0.010870
error rate: 0.010323
error rate: 0.009828
accuracy: 0.720000
fx >>
```

图 8 Iris 数据集测试结果

参考文献

- [1] 人工智能, Nils J. Nilsson 著, 郑扣根, 庄越挺译, 潘云鹤校, 北京: 机械工业出版社, 2000 年 9 月
- [2] <http://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center>
- [3] <http://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set>
- [4] <http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29>
- [5] <http://archive.ics.uci.edu/ml/datasets/Iris>