**反向传播算法实例**

在这里，我们用一个简单的3层神经网络来说明反向传播算法[[1]](#footnote-1)。为了简单起见，这里的传导函数不做任何非线性变换，每个神经元把输入直接作为输出。

该网络的输入层有2个神经元，隐藏层有2个神经元，输出层有1个神经元，如图1所示。图1同时标注了各个神经元之间的连接权重，比如i1和h1的连接权重是w1。

Input Layer Hidden Layer Output Layer

W1

W5

W2

W6

W3

W4

图1. 一个简单的神经网络

反向传播算法的目的，是为神经网络找到一组权重。利用该组权重，神经网络通过前向传导做出的预测，预测误差(prediction error)最小。刚开始，我们给神经网络一组随机的权重组合，比如w1=0.11，w2=0.21，w3=0.12，w4=0.08，w5=0.14，w6=0.15。

训练集只有一个样本(有多个样本的训练方法是类似的），输入为向量<2, 3>，输出为向量<1>。前向传导过程描述如下：

; ; 。

使用初始化的网络权重，经过前向传导过程，计算输出，获得神经网络的预测值。具体可以用矩阵形式来表达如下：

=

神经网络的输出是0.191，而预期的输出是1，两者的误差计算如下(此处使用均方误差)：

=0.327

反向传播算法的基础是梯度下降算法。为了对网络连接的权重进行调整，需要计算误差相对于每个权重的梯度。首先有

Prediction = out = =

梯度下降的基本公式为：W = w -。比如，为了调整w6，我们需要计算误差针对w6的梯度，然后代入。

=

备注：为。

同样地，w5的更新公式为，而=。

接下来，我们需要继续计算，以便对w1进行更新。

= =

=。

同样道理，=, =, =。

即有

可以写成矩阵的形式，具体如下：

我们尝试进行一趟反向传播，看看参数调整的效果。

==-0.809

=0.05

=

利用调整过的权重，重新进行预测。

=

新的预测值为0.2571，比起上一个预测值0.191，更加接近目标值1。利用训练样本，不断迭代前向传导和反向传播过程，直到误差接近0或者到达某个阈值之下[[2]](#footnote-2)。

读者可以打开网址[2]进行试验，了解前向传导和反向传播的效果。

1. https://hmkcode.com/ai/backpropagation-step-by-step/. [↑](#footnote-ref-1)
2. https://hmkcode.com/netflow/. [↑](#footnote-ref-2)