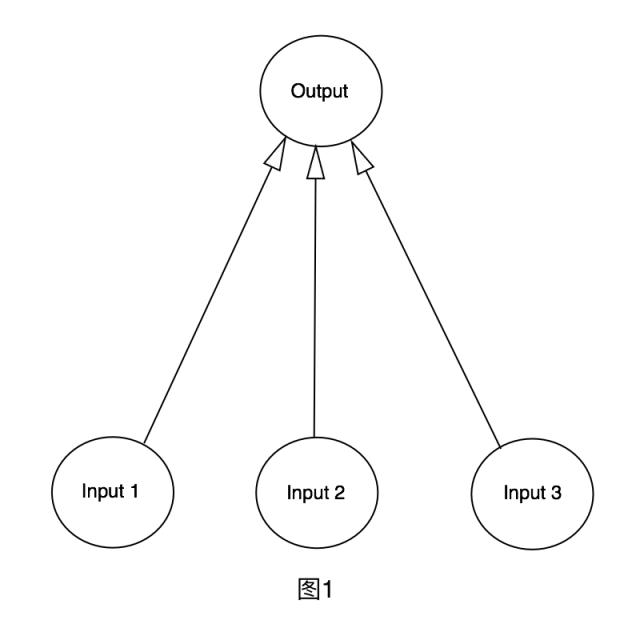
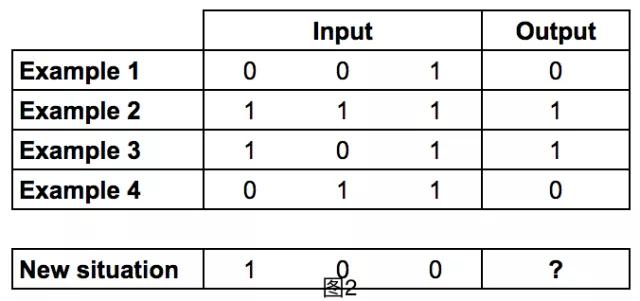
神经网络：人脑由几千亿由突触相互连接的细胞（神经元）组成。突触传入足够的兴奋就会引起神经元的兴奋。这个过程被称为“思考”



我们可以在计算机上写一个神经网络来模拟这个过程。不需要在生物分子水平模拟人脑，只需模拟更高层级的规则。我们使用矩阵（二维数据表格）这一数学工具，并且为了简单明了，只模拟一个有3个输入和一个输出的神经元。

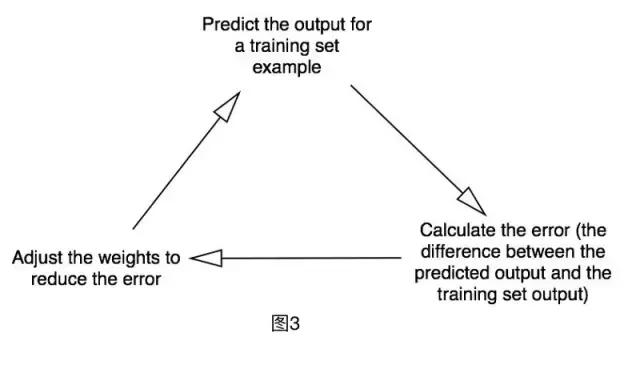
我们将训练神经元解决下面的问题。前四个例子被称作训练集。你发现规律了吗？‘？’是0还是1？



**训练过程**

但是如何使我们的神经元回答正确呢？赋予每个输入一个权重，可以是一个正的或负的数字。拥有较大正（或负）权重的输入将决定神经元的输出。首先设置每个权重的初始值为一个随机数字，然后开始训练过程：

1. 取一个训练样本的输入，使用权重调整它们，通过一个特殊的公式计算神经元的输出。
2. 计算误差，即神经元的输出与训练样本中的期待输出之间的差值。
3. 根据误差略微地调整权重。
4. 重复这个过程1万次。



最终权重将会变为符合训练集的一个最优解。如果使用神经元考虑这种规律的一个新情形，它将会给出一个很棒的预测。

这个过程就是back propagation。

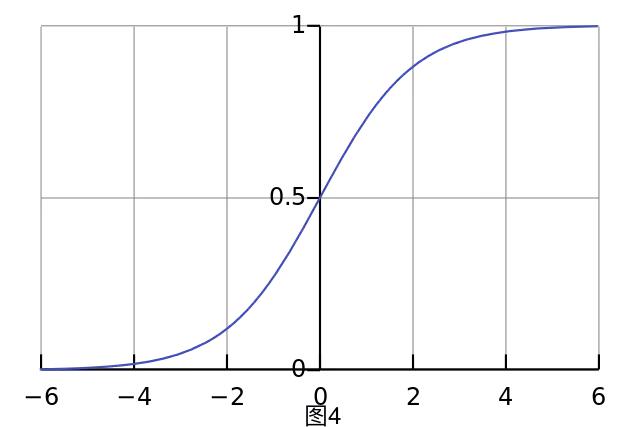
# ****计算神经元输出的公式****

你可能会想，计算神经元输出的公式是什么？首先，计算神经元输入的加权和，即

用9行Python代码编写一个简易神经网络？你敢信？看起来超简单

接着使之规范化，结果在0，1之间。为此使用一个数学函数－－Sigmoid函数：用9行Python代码编写一个简易神经网络？你敢信？看起来超简单

Sigmoid函数的图形是一条“S”状的曲线。



把第一个方程代入第二个，计算神经元输出的最终公式为：

用9行Python代码编写一个简易神经网络？你敢信？看起来超简单

你可能注意到了，为了简单，我们没有引入最低兴奋阈值。

# ****调整权重的公式****

我们在训练时不断调整权重。但是怎么调整呢？可以使用“Error Weighted Derivative”公式：

用9行Python代码编写一个简易神经网络？你敢信？看起来超简单

为什么使用这个公式？首先，我们想使调整和误差的大小成比例。其次，乘以输入（0或1），如果输入是0，权重就不会调整。最后，乘以Sigmoid曲线的斜率（图4）。为了理解最后一条，考虑这些：

1. 我们使用Sigmoid曲线计算神经元的输出
2. 如果输出是一个大的正（或负）数，这意味着神经元采用这种（或另一种）方式
3. 从图四可以看出，在较大数值处，Sigmoid曲线斜率小
4. 如果神经元认为当前权重是正确的，就不会对它进行很大调整。乘以Sigmoid曲线斜率便可以实现这一点

Sigmoid曲线的斜率可以通过求导得到：

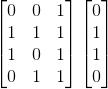
用9行Python代码编写一个简易神经网络？你敢信？看起来超简单

把第二个等式代入第一个等式里，得到调整权重的最终公式：

用9行Python代码编写一个简易神经网络？你敢信？看起来超简单

当然有其他公式，它们可以使神经元学习得更快，但是这个公式的优点是非常简单。

“.T”方法用于矩阵转置（行变列）。所以，计算机这样存储数字：



我觉得我们可以开始构建更优美的源代码了。给出这个源代码后，我会做一个总结。

我对每一行源代码都添加了注释来解释所有内容。注意在每次迭代时，我们同时处理所有训练集数据。所以变量都是矩阵（二维数据表格）。下面是一个用Python写地完整的示例代码。

**构造Python代码**

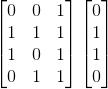
虽然我们没有使用神经网络库，但是将导入Python数学库numpy里的4个方法。分别是：

* exp－－自然指数
* array－－创建矩阵
* dot－－进行矩阵乘法
* random－－产生随机数

比如， 我们可以使用array()方法表示前面展示的训练集：

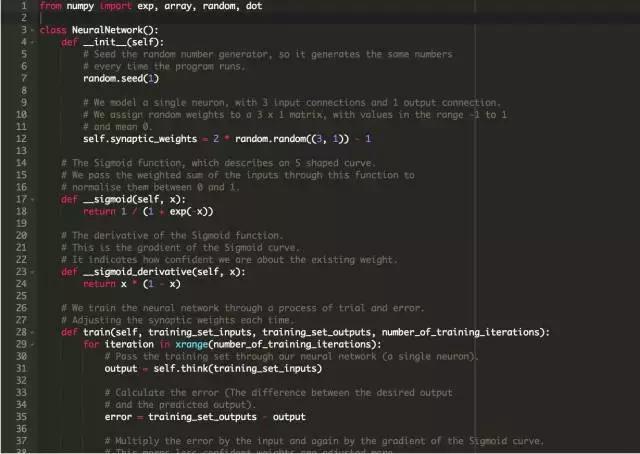
用9行Python代码编写一个简易神经网络？你敢信？看起来超简单

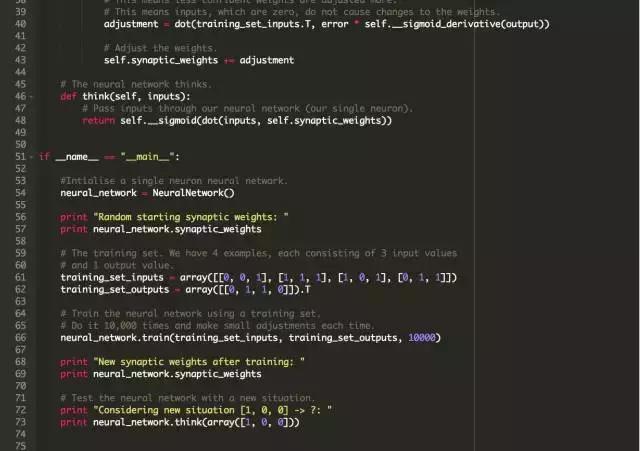
“.T”方法用于矩阵转置（行变列）。所以，计算机这样存储数字：



我觉得我们可以开始构建更优美的源代码了。给出这个源代码后，我会做一个总结。

我对每一行源代码都添加了注释来解释所有内容。注意在每次迭代时，我们同时处理所有训练集数据。所以变量都是矩阵（二维数据表格）。下面是一个用Python写地完整的示例代码。





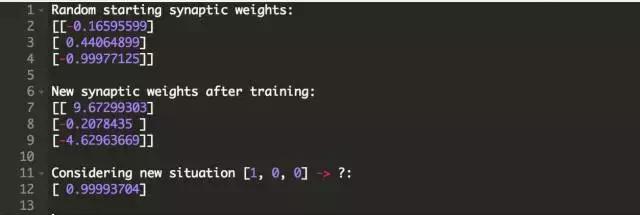
github地址为：https://github.com/miloharper/simple-neural-network

# ****结语****

试着在命令行运行神经网络：

用9行Python代码编写一个简易神经网络？你敢信？看起来超简单

你应该看到这样的结果：



我们做到了！我们用Python构建了一个简单的神经网络！

首先神经网络对自己赋予随机权重，然后使用训练集训练自己。接着，它考虑一种新的情形[1, 0, 0]并且预测了0.99993704。正确答案是1。非常接近！

传统计算机程序通常不会学习。而神经网络却能自己学习，适应并对新情形做出反应，这是多么神奇，就像人类一样。

当然，这只是一个神经元进行了一个非常简单的任务，但是如果我们把数百万的神经元连接起来呢？我们会在某一天创造出有意识的东西吗？