序

- 神经网络，是一种编程范式，让计算机能够从指定样本中学习
- 深度学习，是一套在让神经网络能自我学习的技术集合
- 大多用来解决图片语音识别，自然语言处理这一类问题

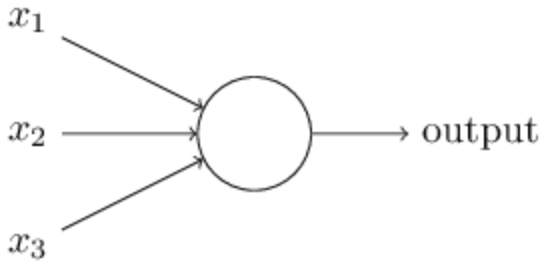
第一章

神经元

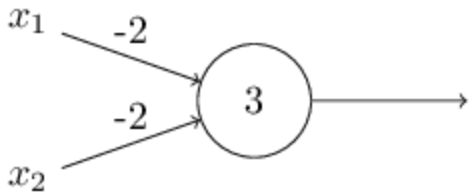
- 神经网络的基础单元是一个个「神经元」，常见的人工神经元有两种：Perceptron 和 Sigmoid 神经元。
- 人工神经元简单理解就是处理多个输入，根据一些规则形成一个输出。

Perceptron

- 先来看 Perceptron，它可以用来进行简单的逻辑运算□。其特点是输入和输出端的值都只能取 0 或者 1。



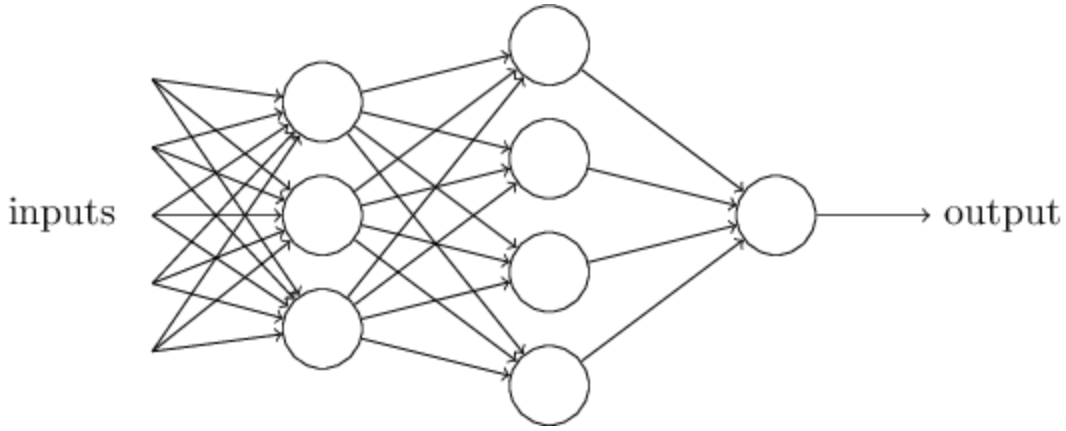
- 如果给每个输入都加上一个权重，再给 Perceptron 本身加上一个激活阈值，那么就可以显式的「计算」出该 Perceptron 的输出是什么：



在这个例子中，如果 x_1 和 x_2 的输入都是 1，那么 $x_1 * -2 + x_2 * -2 = -4$，加上激活阈值 3

后，等于 -1，那我们又规定当最终输出小于等于 0 时，整个神经元的输出为 0，而大于 0 时，神经元输出为 1。因此在 x_1 和 x_2 都输入 1 的情况下，整个神经元输出为 0 。

- 单个神经元当然不是人类大脑思考决策的全过程，让我们把多个神经元组装在一起看看：



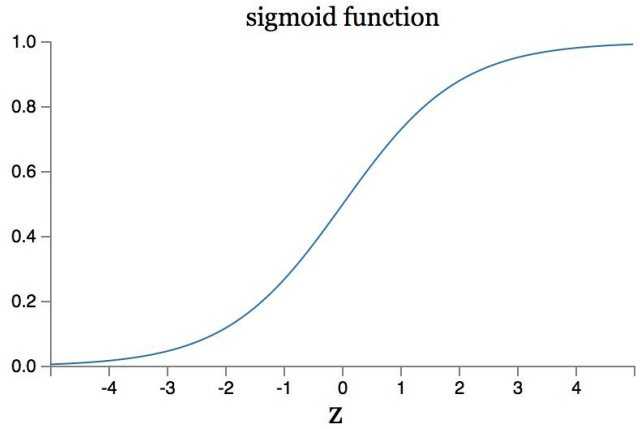
- 上图结构中，左侧的叫做输入层，右侧的叫输出层，中间的两层叫隐藏层。不要觉得隐藏层听起来很神秘，叫这个名字仅仅是因为它既不直接接收外部输入，也不对外部直接输出而已。
- 在这样的多个神经元组成的网络结构中，最终的输出取决于前面每一层的输入值、输入权重、单个神经元的激活阈值。
- 再回头看看 Perceptron 组成的神经网络，它是没法做到输入只有一点点变化时，输出也只变化一点点，因为都是 0 到 1 这样的两级变化，每一个输入的改变，对整个系统输出改变太大，不稳定，没法做「学习」这件事。
- 这里的「学习」可以理解为「调教」，就像你开车偏了一点点，就轻转方向盘 5 度，车子也同时转了 5 度。但如果你的方向盘是左转 5 度，车子立马向右转 90 度，或者左转 180 度，你就没法控制这辆车。因此我们需要一种新的神经元，由其构成的神经网络满足前面提到的「输入只有一点点变化时，输出也只变化一点点」，从而实现慢慢调教。

Sigmoid 神经元

- Sigmoid 神经元，就是用来解决上述问题的。它输入值不再仅仅是 0 或 1，而是 0 到 1 当中的任何值，比如 0.619，0.987 这些。但其输出值不再只是 0 和 1，而是通过一个叫 Sigmoid 函数 (σ) 计算出的值，函数计算公式如下：

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}.$$

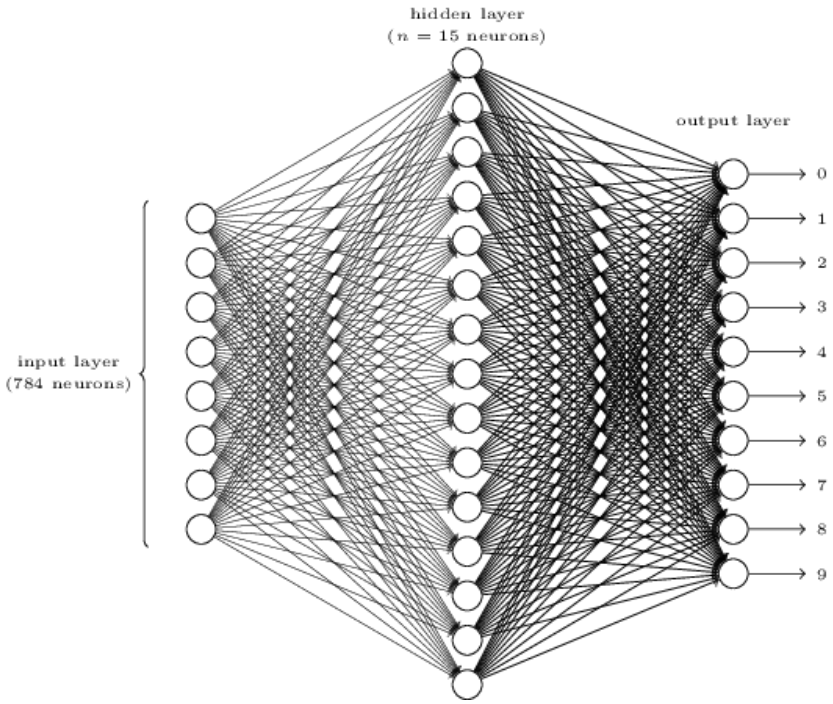
- 让我们来尝试理解下上面这个函数的意义，$\sum_j w_j x_j - b$ 你可以将之理解为神经元所有的输入值分别乘上该输入的权重相加，最后减去激活阈值（聪明的你应该想起来了，前面的 Perceptron 就是这么计算输出值的）。明白这个后，再放回 Sigmoid 函数中，你会发现 $\sum_j w_j x_j - b$ 这一块越大，那么 Sigmoid 函数输出值越接近于 1；而 $\sum_j w_j x_j - b$ 越小，比如是很小的一个负数，那么 Sigmoid 函数输出值接近于 0（这里需要一点点的高中数学知识，回头去看看上面那个公式，分母分子，能想明白的）。
- 因此 Sigmoid 函数的图像是下面这样：



- 在这样一个表现的神经元帮助下，我们就可以实现「输入只有一点点变化时，输出也只变化一点点」了。

机器学习

- 回到处理数字识别上来，如果要处理一个 28*28 分辨率的手写数字灰度图像。那么作为输入层的神经元就是 784个，将每一个像素的灰度值缩放到 0 到 1 以内即可。生成一个神经网络如下图：

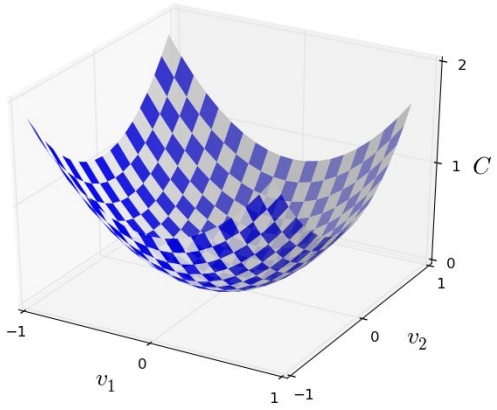


怎么理解这个神经网络？其实就是有一个 28 * 28 分辨率总共 784 个像素点的图片，将每一个像素作为一个输入，然后通过隐藏层（这一层不一定就是图上的 15 个神经元，可以多，可以少，取决于后期参数调优以及对运算效率的控制）的神经元计算，最后在输出层给出当前图片到底是哪个数字的输出。但请记住（画重点了！），这只是一个神经网络的架子而已，每一个神经元输入的权重值、激活阈值都没有设定好，这个时候它还不能去识别手写数字。

- 理论上，你要是时间够多，精力充沛，可以人肉去修改上图中网络里的每一根输入权重以及每一个神经元的激活阈值，然后一边改，一边看输出结果是不是对了。估计调节一个这么简单的神经网络，就够花费正常人一辈子的时间了。
- 那么机器学习是什么？其实就是通过不断的自动化尝试，找出神经网络中各个神经元能够准确识别手写数字时的「输入权重」和「激活阈值」。
- 理解了机器学习就是在一大堆百废待兴的神经元中，找到适合任务目标的最优参数配置后，理解下面两个概念就能轻松很多。
- 调整参数，说起来容易，但输入权重值和激活阈值可以为任何值，如果就让计算随机一遍遍的瞎逼猜，再强大的计算机也没法遍历完世界上所有的数字组合可能。因此我们需要两个工具：
    a. 一种算法，可以评估每一次参数调整后，整个系统对于解决最终任务目标而言，是优化了还是退化了。如果是优化，那么机器就把这个参数往高了再调调；如果退化

了，机器就把这个参数往低了调调。这个算法工具叫做 cost 函数，其值越高说明效果越差；值越低说明效果约好。

b. 另一种算法，可以不断的改变输入权重和激活阈值，使得整个系统的 cost 函数值越来越低，也就说效果越来越好，最终达到 cost 函数的最低值，也就是效果最好的时候。这个算法叫做 gradient descend，文科生视角简单理解下这个算法，其实就是扔一个球到函数图像中，让球自由下落，最终总会滚到函数图像的最低处。



类似于在这样的曲面中，运用 gradient descend 算法，可以得到该曲面最低位置的值。

- 理解了以上概念后，一个很简单的程序，78 行 Python 代码，即可把神经网络跑起来。不断的运用 gradient descend 算法去一遍遍修改各个神经元的参数，从而逼近 cost 函数的最低值，获得这套能实现手写识别功能的神经网络中每一个神经元的输入权重和激活阈值参数。机器学习，就是学到了这套参数。
- 理解了这一点后，你可能也明白了为什么机器学习解决了的问题，人类还是看不懂。因为那套神经网络的配置参数本身没有任何「意义」，它们只是机器为了实现目标而随机改来改去，所谓「学习」出来的一堆参数而已。