

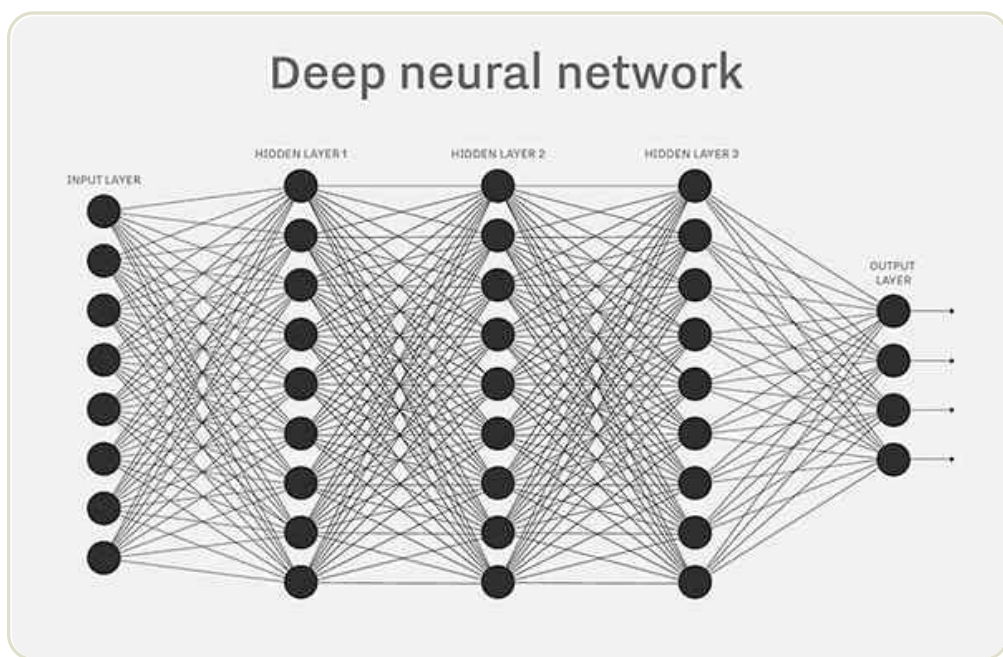
神经网络入门

作者： 阮一峰

日期： 2017年7月13日

眼下最热门的技术，绝对是人工智能。

人工智能的底层模型是“[神经网络](#)”（neural network）。许多复杂的应用（比如模式识别、自动控制）和高级模型（比如深度学习）都基于它。学习人工智能，一定是从它开始。



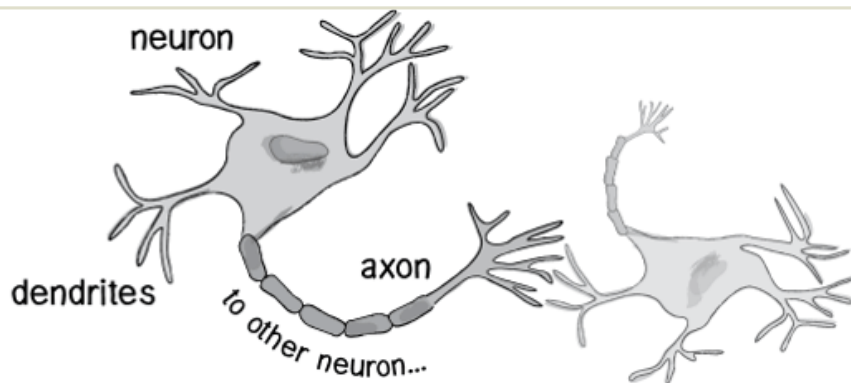
什么是神经网络呢？网上似乎[缺乏](#)通俗的解释。

前两天，我读到 Michael Nielsen 的开源教材[《神经网络与深度学习》](#)（Neural Networks and Deep Learning），意外发现里面的解释非常好懂。下面，我就按照这本书，介绍什么是神经网络。

这里我要感谢[优达学城](#)的赞助，本文[结尾](#)有他们的[《前端开发（进阶）》](#)课程的消息，欢迎关注。

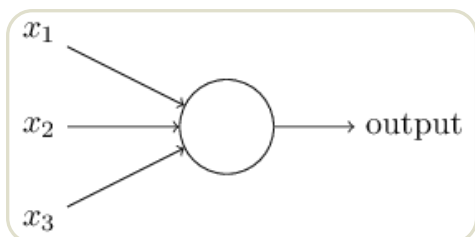
一、感知器

历史上，科学家一直希望模拟人的大脑，造出可以思考的机器。人为什么能够思考？科学家发现，原因在于人体的神经网络。



1. 外部刺激通过神经末梢，转化为电信号，转导到神经细胞（又叫神经元）。
2. 无数神经元构成神经中枢。
3. 神经中枢综合各种信号，做出判断。
4. 人体根据神经中枢的指令，对外部刺激做出反应。

既然思考的基础是神经元，如果能够"人造神经元"（**artificial neuron**），就能组成人工神经网络，模拟思考。上个世纪六十年代，提出了最早的"人造神经元"模型，叫做"感知器"（**perceptron**），直到今天还在用。



上图的圆圈就代表一个感知器。它接受多个输入（ x_1 , x_2 , $x_3...$ ），产生一个输出（output），好比神经末梢感受各种外部环境的变化，最后产生电信号。

为了简化模型，我们约定每种输入只有两种可能：**1** 或 **0**。如果所有输入都是**1**，表示各种条件都成立，输出就是**1**；如果所有输入都是**0**，表示条件都不成立，输出就是**0**。

二、感知器的例子

下面来看一个例子。城里正在举办一年一度的游戏动漫展览，小明拿不定主意，周末要不要去参观。



他决定考虑三个因素。

1. 天气：周末是否晴天？
2. 同伴：能否找到人一起去？
3. 价格：门票是否可承受？

这就构成一个感知器。上面三个因素就是外部输入，最后的决定就是感知器的输出。如果三个因素都是 Yes（使用 1 表示），输出就是1（去参观）；如果都是 No（使用 0 表示），输出就是0（不去参观）。

三、权重和阈值

看到这里，你肯定会问：如果某些因素成立，另一些因素不成立，输出是什么？比如，周末是好天气，门票也不贵，但是小明找不到同伴，他还要不要去参观呢？

现实中，各种因素很少具有同等重要性：某些因素是决定性因素，另一些因素是次要因素。因此，可以给这些因素指定权重（weight），代表它们不同的重要性。

- 天气：权重为8
- 同伴：权重为4
- 价格：权重为4

上面的权重表示，天气是决定性因素，同伴和价格都是次要因素。

如果三个因素都为1，它们乘以权重的总和就是 $8 + 4 + 4 = 16$ 。如果天气和价格因素为1，同伴因素为0，总和就变为 $8 + 0 + 4 = 12$ 。

这时，还需要指定一个阈值（threshold）。如果总和大于阈值，感知器输出1，否则输出0。假定阈值为8，那么 $12 > 8$ ，小明决定去参观。阈值的高低代表了意愿的强烈，阈值越低就表示越想去，越高就越不想去。

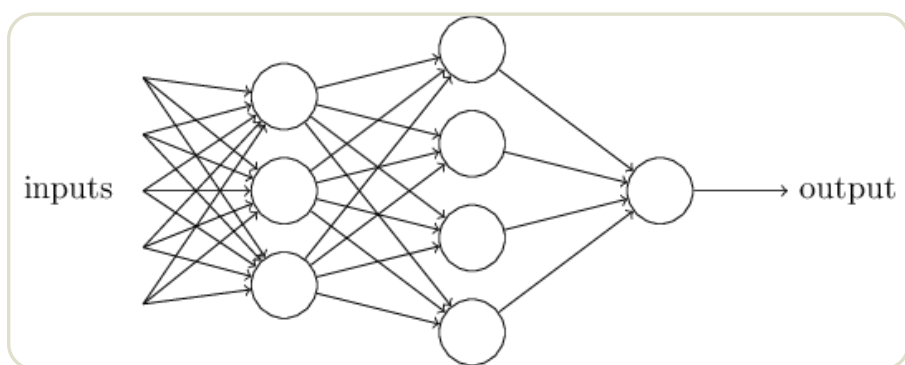
上面的决策过程，使用数学表达如下。

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

上面公式中， x 表示各种外部因素， w 表示对应的权重。

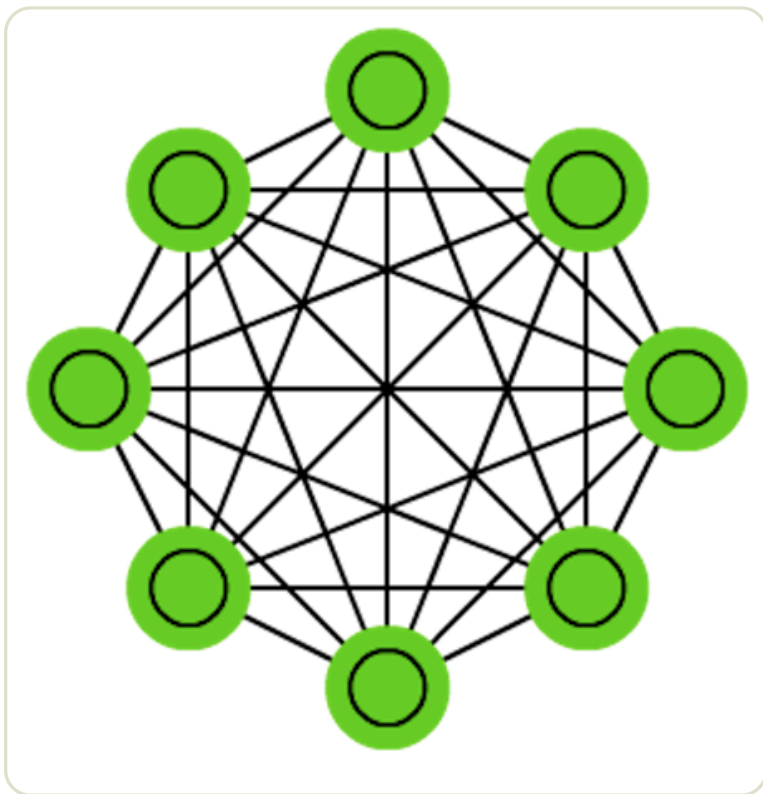
四、决策模型

单个的感知器构成了一个简单的决策模型，已经可以拿来用了。真实世界中，实际的决策模型则要复杂得多，是由多个感知器组成的多层网络。



上图中，底层感知器接收外部输入，做出判断以后，再发出信号，作为上层感知器的输入，直至得到最后的结果。（注意：感知器的输出依然只有一个，但是可以发送给多个目标。）

这张图里，信号都是单向的，即下层感知器的输出总是上层感知器的输入。现实中，有可能发生循环传递，即 A 传给 B，B 传给 C，C 又传给 A，这称为"递归神经网络"（recurrent neural network），本文不涉及。



五、矢量化

为了方便后面的讨论，需要对上面的模型进行一些数学处理。

- 外部因素 x_1 、 x_2 、 x_3 写成矢量 $\langle x_1, x_2, x_3 \rangle$ ，简写为 x
- 权重 w_1 、 w_2 、 w_3 也写成矢量 (w_1, w_2, w_3) ，简写为 w
- 定义运算 $w \cdot x = \sum w_i x_i$ ，即 w 和 x 的点运算，等于因素与权重的乘积之和
- 定义 b 等于负的阈值 $b = -\text{threshold}$

感知器模型就变成了下面这样。

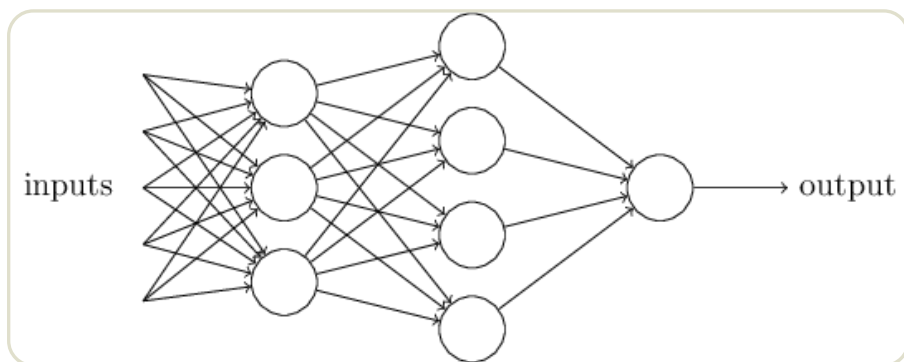
$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

六、神经网络的运作过程

一个神经网络的搭建，需要满足三个条件。

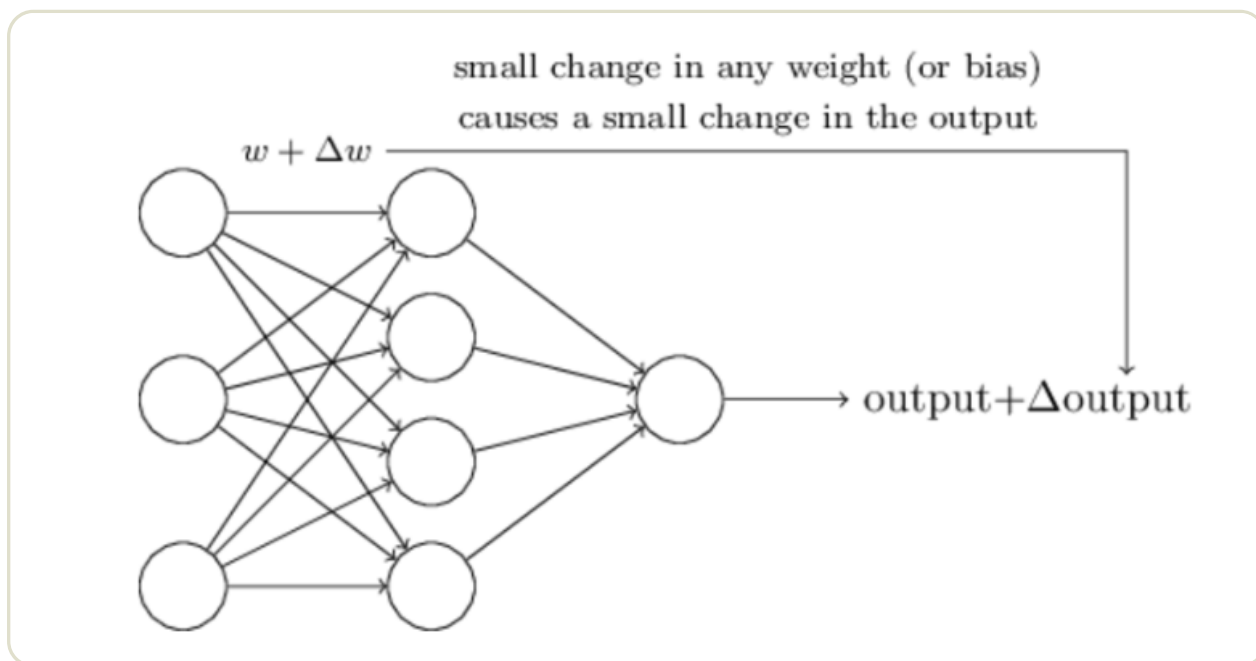
- 输入和输出
- 权重 (w) 和阈值 (b)
- 多层感知器的结构

也就是说，需要事先画出上面出现的那张图。



其中，最困难的部分就是确定权重（ w ）和阈值（ b ）。目前为止，这两个值都是主观给出的，但现实中很难估计它们的值，必需有一种方法，可以找出答案。

这种方法就是试错法。其他参数都不变， w （或 b ）的微小变动，记作 Δw （或 Δb ），然后观察输出有什么变化。不断重复这个过程，直至得到对应最精确输出的那组 w 和 b ，就是我们要的值。这个过程称为模型的训练。



因此，神经网络的运作过程如下。

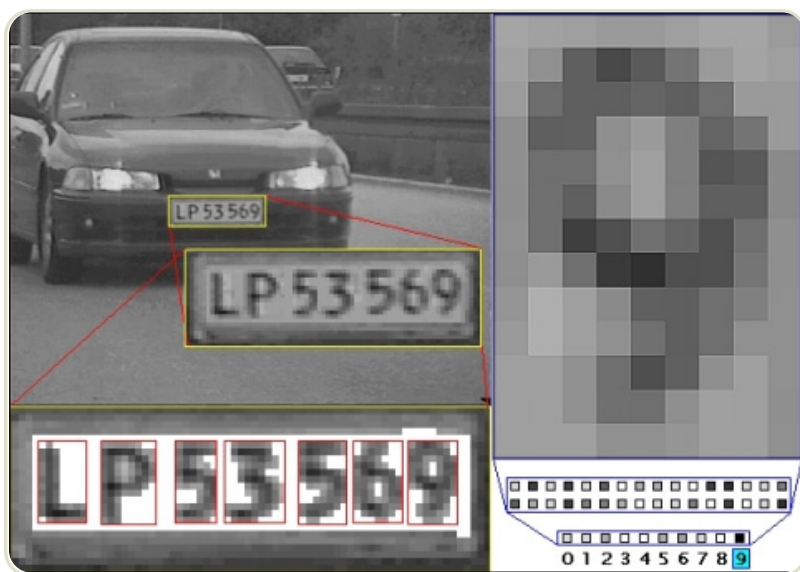
1. 确定输入和输出
2. 找到一种或多种算法，可以从输入得到输出
3. 找到一组已知答案的数据集，用来训练模型，估算 w 和 b
4. 一旦新的数据产生，输入模型，就可以得到结果，同时对 w 和 b 进行校正

可以看到，整个过程需要海量计算。所以，神经网络直到最近这几年才有实用价值，而且一般的 CPU 还不行，要使用专门为机器学习定制的 GPU 来计算。

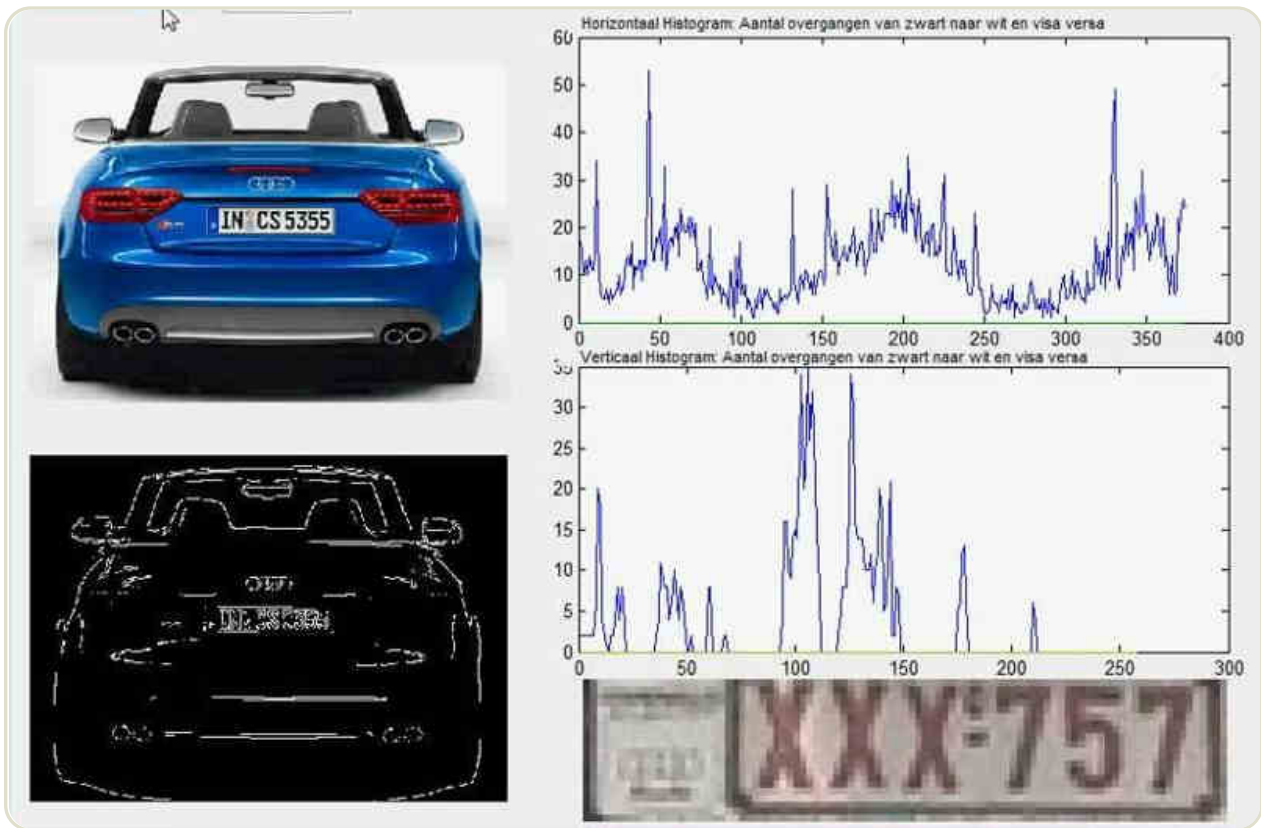


七、神经网络的例子

下面通过车牌自动识别的例子，来解释神经网络。



所谓"车牌自动识别"，就是高速公路的探头拍下车牌照照片，计算机识别出照片里的数字。



这个例子里面，车牌照片就是输入，车牌号码就是输出，照片的清晰度可以设置权重（ w ）。然后，找到一种或多种图像比对算法，作为感知器。算法的得到结果是一个概率，比如75%的概率可以确定是数字 1。这就需要设置一个阈值（ b ）（比如85%的可信度），低于这个门槛结果就无效。

一组已经识别好的车牌照片，作为训练集数据，输入模型。不断调整各种参数，直至找到正确率最高的参数组合。以后拿到新照片，就可以直接给出结果了。



八、输出的连续性

上面的模型有一个问题没有解决，按照假设，输出只有两种结果：0和1。但是，模型要求 w 或 b 的微小变化，会引发输出的变化。如果只输出 0 和 1，未免也太不敏感了，无法保证训练的正确性，因此必须将"输出"改造成一个连续性函数。

这就需要进行一点简单的数学改造。

首先，将感知器的计算结果 $w x + b$ 记为 z 。

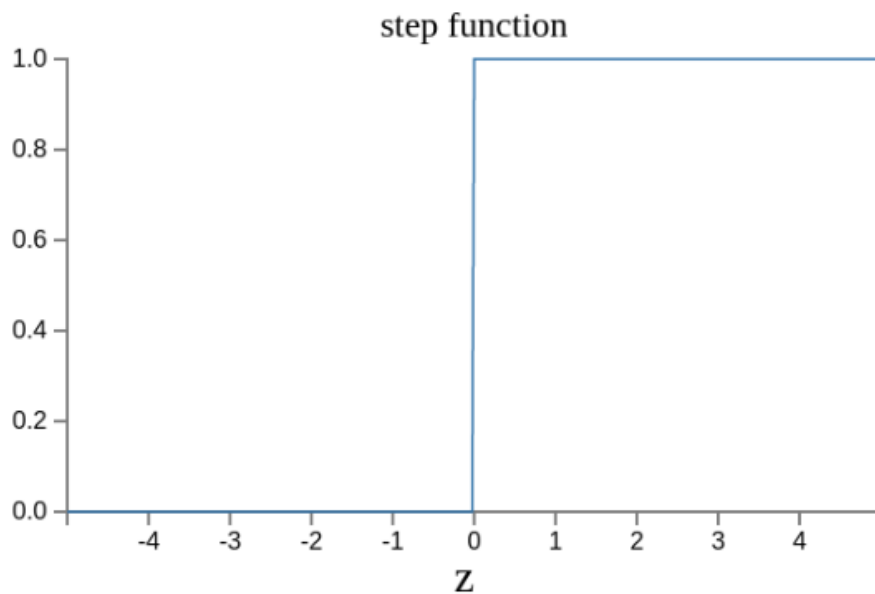
$$z = w x + b$$

然后，计算下面的式子，将结果记为 $\sigma(z)$ 。

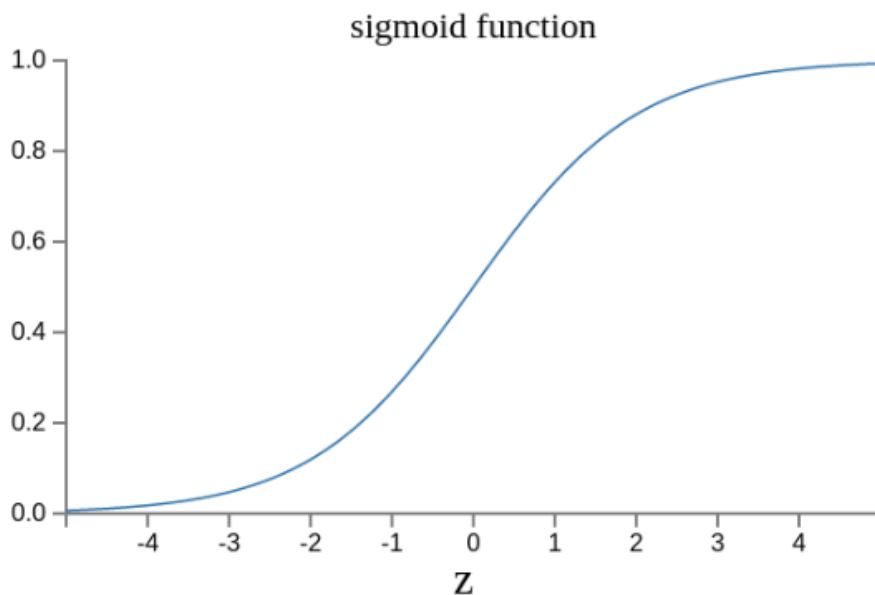
$$\sigma(z) = 1 / (1 + e^{(-z)})$$

这是因为如果 z 趋向正无穷 $z \rightarrow +\infty$ （表示感知器强烈匹配），那么 $\sigma(z) \rightarrow 1$ ；如果 z 趋向负无穷 $z \rightarrow -\infty$ （表示感知器强烈不匹配），那么 $\sigma(z) \rightarrow 0$ 。也就是说，只要使用 $\sigma(z)$ 当作输出结果，那么输出就会变成一个连续性函数。

原来的输出曲线是下面这样。



现在变成了这样。



实际上，还可以证明 $\Delta \sigma$ 满足下面的公式。

$$\Delta \text{output} \approx \sum_j \frac{\partial \text{output}}{\partial w_j} \Delta w_j + \frac{\partial \text{output}}{\partial b} \Delta b,$$

即 $\Delta \sigma$ 和 Δw 和 Δb 之间是线性关系，变化率是偏导数。这就有利于精确推算出 w 和 b 的值了。

（正文完）

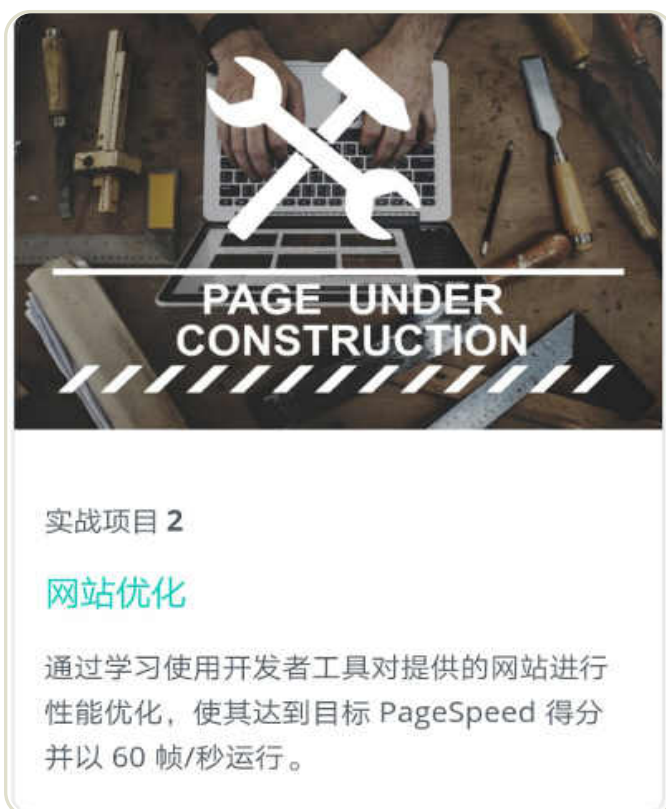
=====

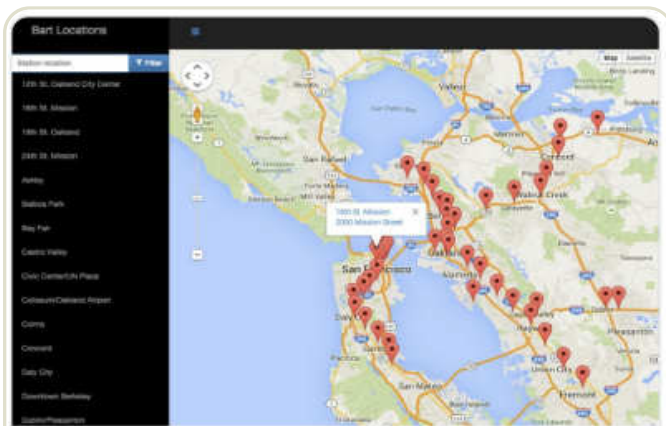
下面是推广时间。

前端开发是[优达学城](#) Udacity 的纳米学位课程，分成两个级别：入门班和进阶班。今年下半年的[进阶班](#)，今天（7月13日）开始报名了，感兴趣的朋友不要错过！



本课程的重点是讲解如何应用 CSS 框架和 JavaScript 框架，做出高性能、高可用性的产品，并且使用测试工具保证代码质量。学习过程中，学员必须完成以下课堂练习。





实战项目 3

街区地图

学习框架与 API 的使用，开发一个单页应用，展示你所在街区或你想要参观的街区的地图，并向此应用添加更多功能。



实战项目 4

订阅阅读器测试




通过 Jasmine 测试框架，完成一个读取 RSS Feeds 的网络应用。

该课程是纳米学位课程，学员提交的每一行代码都有导师 code review，并且每周可以预约导师一对一辅导，对于提高个人能力极有帮助。

由于有真人 code review 环节，所以招生人数有限制，本期只有200个名额，目前已经预定了67个。[点击这里](#)了解详情，报名从速哦。

（完）

文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（创意共享3.0许可证）
- 发表日期：2017年7月13日
- 更多内容：档案 » 算法与数学
- 博客文集：《前方的路》，《未来世界的幸存者》
- 社交媒体： twitter,  weibo
- Feed订阅：

打造中国最权威的《前端-全栈-工程化课程》

八年专注前端， 珠峰培训让你高薪就业

快戳我！了解详情 

年薪50万不是梦
从前端小工到BAT中高级工程师的必备技能



13大模块 / 52 个课时 / 3个月强化学习

相关文章

- 2016.07.22: [如何识别图像边缘？](#)

图像识别（image recognition）是现在的热门技术。

- 2015.09.01: [理解矩阵乘法](#)

大多数人在高中，或者大学低年级，都上过一门课《线性代数》。这门课其实是教矩阵。

- 2015.07.27: [蒙特卡罗方法入门](#)

本文通过五个例子，介绍蒙特卡罗方法（Monte Carlo Method）。

- 2015.06.10: [泊松分布和指数分布：10分钟教程](#)

大学时，我一直觉得统计学很难，还差点挂科。

广告（购买广告位）

如何在遍布谎言和套路的世界里学到点真东西?

妙味课堂：新班招生进行中 ...

