一年多前，吴军博士写了一本畅销书《智能时代》[1]。书里提到，在人工智能领域，有一个流派叫“鸟飞派”，亦称之为“模仿派”。说的是，当人们要学习飞翔的时候，最先想到的是模仿鸟一样去飞翔。

很多年前，印度诗人泰戈尔出了本《飞鸟集》，里面有个名句：“天空没有留下翅膀的痕迹，但我已经飞过”。有人对此解读为，“人世间，很多事情虽然做过了，却不为人所知，但那又如何？重要的是，我已做过，并从中获得了许多。”

两千多年前，司马迁在《史记•滑稽列传》写到：“此鸟不飞则已，一飞冲天；不鸣则已，一鸣惊人。”说的是当年楚庄王在“势不眷我”时，选择了“蛰伏”。蛰伏，只是一个储势过程，迟早有一天，蓄势待发，“发”则达天。

这三者的情感交集，让我联想到出了本章的主人公杰弗里•辛顿（Geoffrey Hinton）教授，在学术界里，他就是这样的一个“励志”人物！

1986年，辛顿教授和他的小伙伴们重新设计了BP算法，以“人工神经网络”模仿大脑工作机理，“吻”醒了沉睡多年的“人工智能”公主，一时风光无限。

但“好花不常开，好景不常在”。当风光不再时，辛顿和他的研究方向，逐渐被世人所淡忘。

这被“淡忘”的冷板凳一坐，就是30年。

但在这30年里，辛顿又如“飞鸟”一般，即使“飞过无痕”，也从不放弃。从哪里跌倒，就从哪里爬起。实在不行，即使换个马甲，也要重过一生。

玉汝于成，功不唐捐。

终于，在2006年，辛顿等人提出了“深度信念网（Deep Belief Nets，DBN）”（这实际上就是多层神经网络的马甲）[2]。这个“深度信念网”后期被称为“深度学习”。终于，辛顿再次闪耀于人工智能世界，随后被封为“深度学习教父”。

但细心的读者可发现，即使辛顿等人提出了“深度信念网”，在随后的小10年里，这个概念亦是不温不火地发展着（如图1所示）。直到后期（2012年以后），随着大数据和大计算（GPU、云计算等）的兴起，深度学习才开始大行其道，一时间甚嚣尘上。

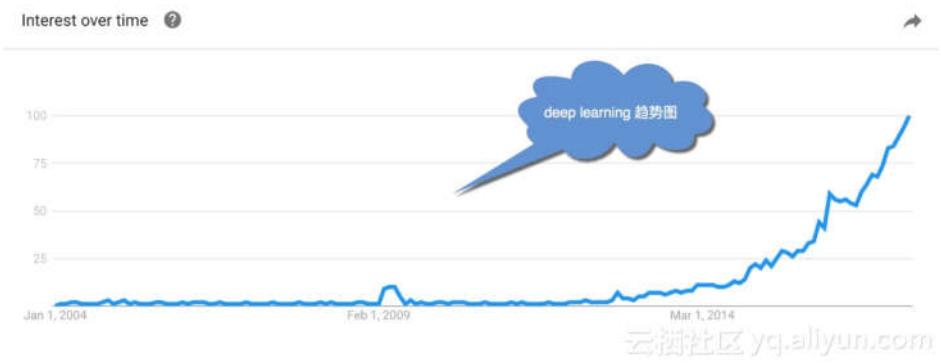


图7-1 深度学习的谷歌趋势图

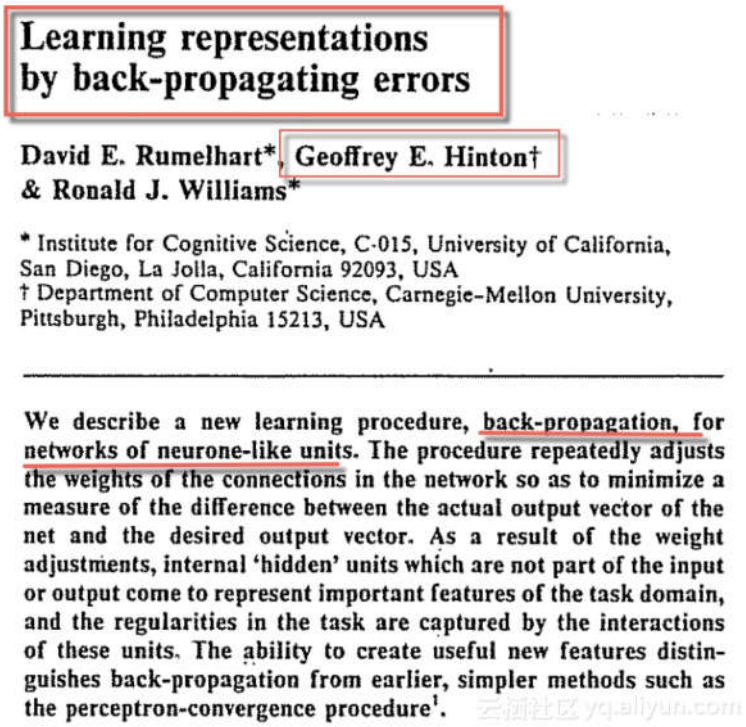
回顾起杰弗里•辛顿过往40多年的学术生涯，可谓是顾跌宕起伏，但最终修得正果。倘若细细说起，这“牛逼”，还得从1986年吹起。

7.1 1986年的那篇神作

1986年10月，杰弗里•辛顿还在卡内基梅隆大学任职。他和在加州大学圣迭戈分校的认知心理学家大卫·鲁梅尔哈特（David Rumelhart）等人，在著名学术期刊《自然》上联合发表题为：“通过反向传播算法的学习表征（Learning Representations by Back-propagating errors）”的论文[3]。该文首次系统简洁地阐述反向传播算法（BP）在神经网络模型上的应用，该算法把网络权值纠错的运算量从原来的与神经元数目的平方成正比，下降到只和神经元数目本身成正比。

与此同时，当时的大背景是，在八十年代末，Intel x86系列的微处理器和内存技术的发展，让计算机的运行速度和数据访存速度也比二十年前高了几个数量级。这一下（运算量下降）一上（计算速度上升），加之多层神经网络可通过设置隐含层 (hidden layer)，极大增强了数据特征的表征能力，从而轻易解决感知机无法实现的异或门 (XOR gate)难题，这些"天时地利人和"的大好环境，极大缓解了当年明斯基对神经网络的责难。

于是，人工神经网络的研究，渐渐得以复苏。



Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

图7-2 1986年杰弗里•辛顿的那篇神作

值得一提的是，在文献[3]中，杰弗里•辛顿并不是第一作者，鲁梅尔哈特才是，而辛顿仅仅"屈居"第二（如图7-2所示）。但为什么我们提起BP算法时，总是说起辛顿呢？其实原因也很简单，主要有二：第一、鲁梅尔哈特毕竟并非计算机科学领域之内的人士，我们计算机科学家，总不能找一个脑科学家去"拜码头"吧；第二、辛顿是这篇论文的通信作者，通常而言，通信作者才是论文思路的核心提供者，这样一来，即使作者排名第二，也没有埋没掉辛顿教授的贡献。

同在1986年，鲁梅尔哈特也和自己的小伙伴们合作发表了一篇题为"并行分布式处理：来自认知微结构的探索"的论文[4]。仅仅从论文题目的前半部分来看，我们很可能误解这是一个有关"高性能计算"的文章，但从标题的后半部分可以得知，这是鲁梅尔哈特等人对人类大脑研究的最新认知。鲁梅尔哈特对大脑工作机理的深入观察，极大地启发了辛顿。辛顿灵光一现，觉得可以把这个想法迁移到"人工神经网络"当中。于是，就有了他们神来一笔的合作。

我们知道，1986年，辛顿和鲁梅尔哈特能在大名鼎鼎的《自然》期刊上发表论文，自然不是泛泛而谈，它一定是解决了什么大问题。下面我们就聊聊这个话题。

7.2 多层感知机网络遇到的大问题

由于历史的惯性，在第六讲中提到的多层前馈网络，有时也被称为多层感知机（Multilayer Perceptron，MLP）。但这个提法导致概念多少有些混淆。这是因为，在多层前馈网络中，神经元的内部构造已悄然发生变化，即激活函数从简单粗暴的"阶跃函数"变成了比较平滑的挤压函数Sigmoid（如图7-3所示）。

激活函数为什么要换成Sigmoid呢？其实原因并不复杂，这是因为感知机的激活函数是阶跃函数，不

利于函数求导，进而求损失函数的极小值。我们知道，当分类对象是线性可分，且学习率（learning rate）η足够小时，由感知机还堪胜任，由其构建的网络，还可以训练达到收敛。但分类对象不是线性可分时，感知机就有点"黔驴技穷"了。因此，通常感知机并不能推广到一般前馈网络中。

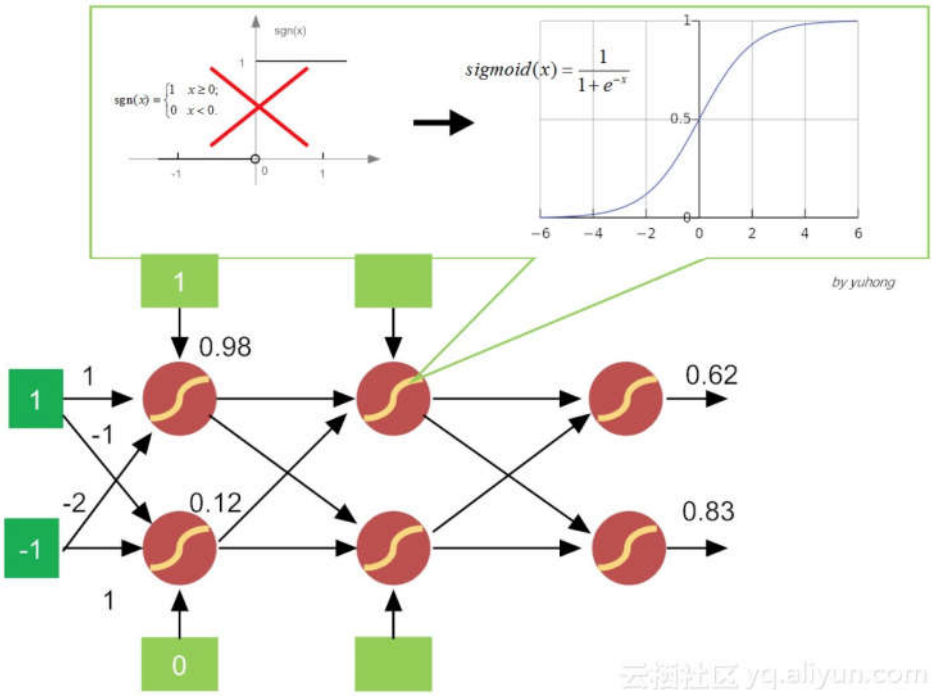


图 7-3 变更激活函数的前馈多层神经网络

按照我们前面章节的说法，所谓的机器学习，简单来说，就是找到一个好用的函数（function），从而较好地实现某个特定的功能（function）。一言蔽之，函数就是功能。而对于某个特定的前馈神经网络，给定网络参数（连接权值与阈值），其实就是定义了一个具备数据采集（输入层）、加工处理（隐含层），然后输出结果（输出层）的函数。

如果仅仅给定一个网络结构，其实它定义的是一个函数集合。因为不同的网络参数（连接权值与阈值），实现的功能"大相径庭"。功能不同，自然函数也是不同的！

针对前馈神经网络，我们需要实现的目的很简单，就是想让损失函数达到最小值，因为只有这样，实际输出和预期输出的差值才最小。那么，如何从众多网络参数（神经元之间的链接权值和阈值）中选择最佳的参数呢？

最简单粗暴的方法，当然就是枚举所有可能值了！

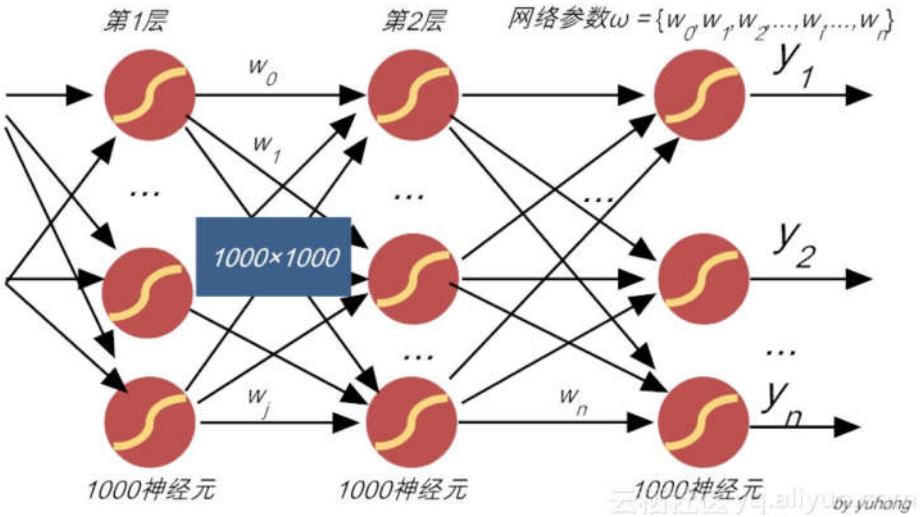


图7-4 暴力调参不可取

但这中暴力策略，对稍微复杂一点的网络就不可取了！例如，用于语音识别的神经网络，假设网络结构有7层，每一层有1000个神经元，那么仅一层之间的全连接权值，就达到$1000 \times 1000 = 10^6$个，一旦层次多了，那权值数量就海了去了！（如图7-4所示）。故此，这种暴力调参找最优参数，既不优雅，也不高效，故实不可取！

7.3到底什么是梯度？

为了克服多层感知机存在的问题，人们设计了一种名为delta（$Delta$）法则（delta rule）的启发式方法，该方法可以让目标收敛到最佳解的近似值[5]。
delta法则的核心思想在于，使用梯度下降（gradient descent）的方法找极值。具体说来，就是在假设空间中搜索可能的权值向量，并以"最佳"的姿态，来拟合训练集合中的样本。那么，何谓最佳拟合呢？当然就是让前文提到的损失函数达到最小值！

我们知道，求某个函数的极值，难免就要用到"导数"等概念。既然我们把这个系列文章定位为入门层次，那不妨就再讲细致一点。什么是导数呢？所谓导数，就是用来分析函数"变化率"的一种度量。针对函数中的某个特定点x0，该点的导数就是x0点的"瞬间斜率"，也即切线斜率，见公式（7.1）。

$$x_0) = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} \qquad (7.1)$$
$$\frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

如果这个斜率越大，就表明其上升趋势越强劲。当这个斜率为0时，就达到了这个函数的"强弩之末"，即达到了极值点。而前文提到的损失函数，如果要达到损失最小，就难免用到导数"反向指导"如何快速抵达极小值。
在单变量的实值函数中，梯度就可以简单地理解为只是导数，或者说对于一个线性函数而言，梯度就是线的斜率。但对于多维变量的函数，它的梯度概念就不那么容易理解了。下面我们来谈谈这个概念。
在向量微积分中，标量场的梯度其实是一个向量场（vector field）。对于特定函数的某个特定点，它的梯度就表示从该点出发，该函数值增长最为迅猛的方向（direction of greatest increase of a function）[6]。假设一个标量函数f的梯度记为：f或$grad f$，这里的表示向量微分算子。那么，在一个三维直角坐标系，该函数的梯度就可以表示为公式（7.2）：

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) \quad (7.2)$$

求这个梯度值，难免要用到"偏导"的概念。说到"偏导"，这里顺便"轻拍"一下国内的翻译。"偏导"的英文本意是"partial derivatives（局部导数）"，书本上常翻译为"偏导"，可能会把读者的思路引导"偏"了。
那什么是"局部导数"呢？对于多维变量函数而言，当求某个变量的导数（相比于全部变量，这里只求一个变量，即为"局部"），就是把其它变量视为常量，然后整个函数求其导数。之后，这个过程对每个变量都"临幸"一遍，放在向量场中，就得到了这个函数的梯度了。举例来说，对于3变量函数$f = x^2 + 3xy + y^2 + z^3$，它的梯度可以这样求得：
（1） 把y，z视为常量，得x的"局部导数"：

$$\frac{\partial f}{\partial x} = 2x + 3y$$

（2） 然后把x，z视为常量，得y的"局部导数"：

$$\frac{\partial f}{\partial y} = 3x + 2y$$

（3） 最后把x，y视为常量，得z的“局部导数”：

$$\frac{\partial f}{\partial y} = 3z^2$$

于是，函数f的梯度可表示为：

$$rad(f)$$
$$3x + 2y, 3z^2$$

针对某个特定点，如点A（1, 2, 3），带入对应的值即可得到该点的梯度：

$$\begin{aligned}\nabla f &= grad(f) \\ &= (2x + 3y, 3x + 2y, 3z^2)\big|_{\substack{x=1 \\ y=2 \\ z=3}} \\ &= (8, 7, 27)\end{aligned}$$

这时，梯度可理解为，站在向量点A（1, 2, 3），如果想让函数f的值增长得最快，那么它的下一个前进的方向，就是朝着向量点B（8,7,27）方向进发（如图7-5所示）。很显然，梯度最明显的应用，就是快速找到多维变量函数的极(大/小)值。

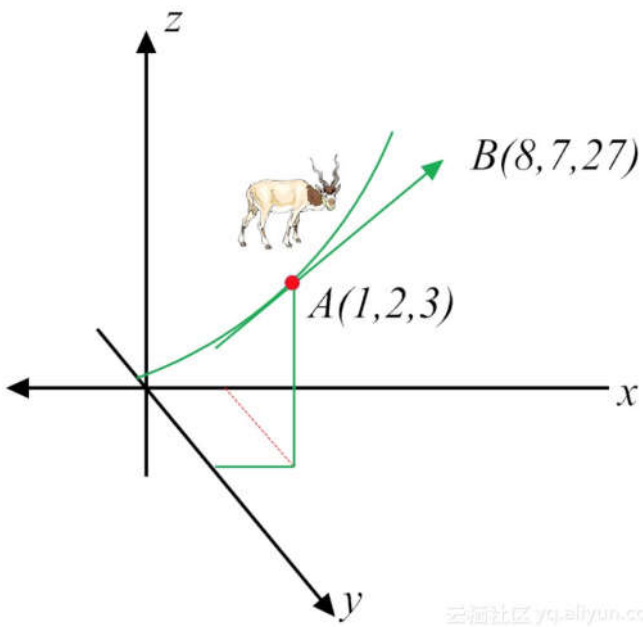


图7-5 梯度概念的示意图

在这里需要说明的是，我们用“局部导数”的翻译，仅仅是用来加深大家对“偏导”的理解，并不是想纠正大家已经约定俗成的叫法。所以为了简单起见，在后文我们还是将“局部导数”称呼为“偏导”。

7.4 到底什么是梯度下降？

上面我们提到了梯度的概念，下面我们说说在求损失函数极小值过程中，常常提到的“梯度递减”的概念。

我们先给出一个形象的案例。爬过山的人，可能会有这样的体会，爬坡愈平缓（相当于斜率较小），抵达山峰（函数峰值）的过程就越缓慢，而如果不考虑爬山的重力阻力（对于计算机而言不存在这样的阻力），山坡越陡峭（相当于斜率越大），顺着这样的山坡爬山，就越能快速抵达山峰（对于函数而言，就是愈加快速收敛到极值点）。

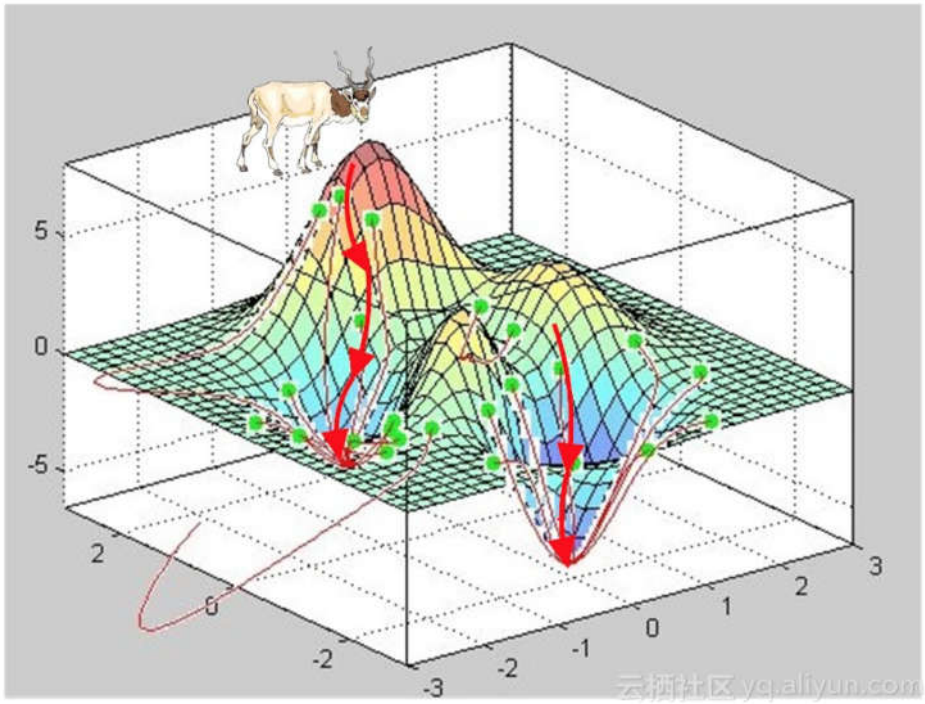


图7-6 梯度递减求极小值

如果我们把山峰"乾坤大挪移"，把爬山峰变成找谷底（即求极小值），这时找斜率最陡峭的坡而攀爬山峰的方法，并没有本质变化，不过是方向相反而已。如果把登山过程中求某点的斜率称为"梯度（gradient）"，而找谷底的方法，就可以把它称之为"梯度递减（gradient descent）"，示意图如图7-6所示。

依据"梯度递减"作为指导，走一步，算一步，一直遵循"最陡峭"的方向，探索着前进，这个过程，是不是有点像邓公的名句"摸着石头过河"？

这个"梯度递减"体现出来的指导意义，就是"机器学习"中的"学习"内涵，即使在大名鼎鼎的"AlphaGo"中，"学习"这是这么玩的！你是不是有点失望？这机器学习也太不高大上了！

但别忘了，在第一讲中，我们就已经讲到"学习"的本质，在于性能的提升。利用"梯度递减"的方法，的确在很大程度上，提升了机器的性能，所以，它就是"学习"！

当然，从图7-6中，我们也很容易看到"梯度递减"的问题所在，那就是它很容易收敛到局部最小值。正如攀登高峰，我们会感叹"一山还比一山高"，探寻谷底时，我们也可能发现，"一谷还比一谷低"。但是"只缘身在此山中"，当前的眼界让我们像"蚂蚁寻路"一样，很难让我们有全局观，因为我们都没有"上帝视角"。

7.5 重温神经网络的损失函数

针对前馈神经网络的设计，输入和输出层设计比较直观。比如说，假如我们尝试判断一张手写数字图片上面是否写着数字"2"。很自然地，我们可以把图片像素的灰度值作为网络的输入。如果图片的维度是16×16，那么我们输入层神经元就可以设计为256个（也就是说，输入层是一个包括256个灰度值向量），每个神经元接受的输入值，就是规格化的灰度值。

而输出层的设计也很简单，就是需要包含10神经元，输出是数字"0~9"的分类概率（也就是说，输出层是一个包括10个概率值的向量）。择其大者而判之，如图7-7所示，如果判定为"2"的概率（比如说80%）远远大于其他数字，那么整个神经网络的最终判定，就是手写图片中的数字是"2"，而非其它数字。

相比于神经网络输入、输出层设计的简单直观，它的隐含层设计，可就没有那么简单了。说好听点，它是一门艺术，依赖于"工匠"的打磨。说不好听点，它就是一个体力活，需要不断地"试错"。

但通过不断地"折腾"，研究人员还真是掌握了一些针对隐层的启发式设计规则（如下文即将提到的BP算法），以此降低训练网络所花的开销，并尽量提升网络的性能。

那么，怎样才算是提升神经网络性能呢？这就需要用到我们前面提到的损失函数了。在第六章我们提到，所谓"损失函数"，就是一个刻画实际输出值和期望输出值之间"落差"的函数。

为了达到理想状态，我们当然希望这种"落差"最小，也就是说，我们希望快速配置好网络参数，从而让这个损失函数达到极小值。这时，神经网络的性能也就接近最优！

关于求损失函数极小值，台湾大学李宏毅博士给出了一个通俗易懂的例子，下面我们来说说。对于识别手写数字的神经网络，训练数据都是一些"0，1 2， ...，9"等数字图像，如图7-8所示。

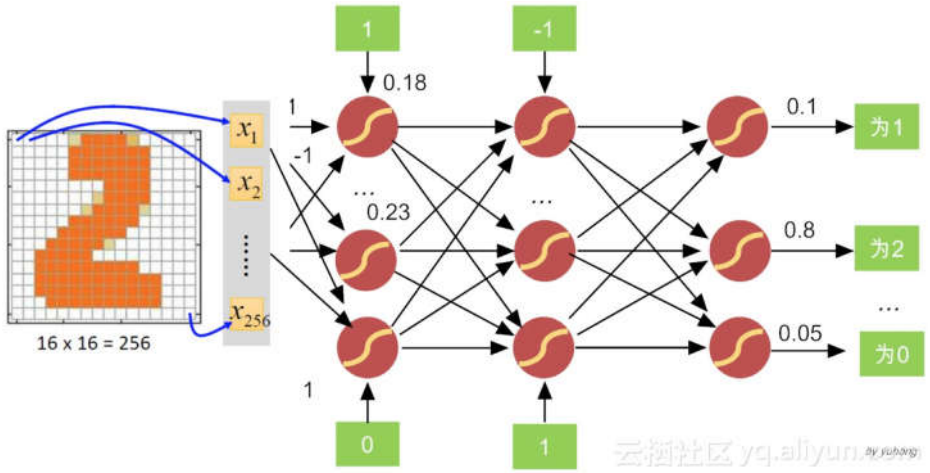


图7-8 识别手写数字的神经网络

由于人们手写数字的风格不同，图像的残缺程度不同，输出的结果有时并不能"十全十美"，于是我们就用损失函数来衡量二者的误差。前面我们提到，常用的损失函数是：

$$(Y,f(X)) = \frac{1}{2}(Y - f(X))^2 \quad (7.3)$$

机器学习的任务，在很大程度上，找一个模型，拟合（fitting）或者说"适配"给定的训练数据，然后再用这个模型预测新数据。这个模型的表现形式，具体说来，就是设计一个好用的函数，用以揭示这些训练样本随自变量的变化关系。揭示拟合好坏的程度，就要用到损失函数。"损失"越小，说明拟合的效果就越好。

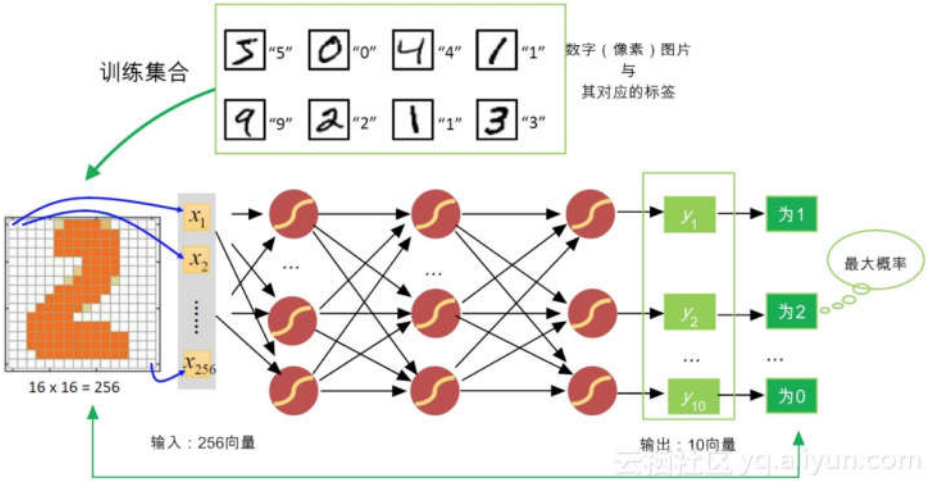


图7-9 用梯度递减，更新网络权值

在我们训练神经网络时，损失函数说白了，就是有关"权值参数"的函数。为了求损失函数的极小值，就不可避免地需要计算损失函数中每一个权值参数的偏导数，这时前文中提到的"梯度递减"方法就派上用场了。训练线性单元的梯度递减算法示意图如图7-9所示，图中的参数η就是"学习率"，它决定了梯度递减搜索的步长，这个步长"过犹不及"。如果值太小，则收敛慢，如果值太大，则容易越过极值，导致网络震荡，难以收敛。

图7-9仅仅给出一个权值变量w_i的梯度示意图，而实际上，神经网络中的参数是非常多的，因此针对损失函数L的权值向量的梯度$mathopwlimits^to$可以记作：

$$\nabla L(w) \equiv \left[\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_n}\right] \quad (7.4)$$

在这里，$nablaL(mathopwlimits^to)$本身就是一个向量，它的多个维度分别由损失函数L对多个权值参数w_i求偏导所得。当梯度被解释为权值空间的一个向量时，它就确定了L对陡峭上升的方向。

如果需要根据图7-9所示的公式来更新权值，我们需要一个更加实用的办法，在每一步重复计算。幸运的是，这个过程并不复杂，通过简易的数学推导，我们可以得到每个权值分量w_i更加简明的计算公式：

$$\begin{aligned}\frac{\partial L}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}(Y - f(X))^2 = \frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(y_d - y'_d)^2 \\ &= \frac{1}{2}\sum_{d\in D}2(y_d - y'_d)\frac{\partial}{\partial w_i}(y_d - y'_d) \\ &= \sum_{d\in D}(y_d - y'_d)\frac{\partial}{\partial w_i}(y_d - w\cdot x_d)\end{aligned} \quad (7.5)$$

其中，x_{id}表示训练集合第d个样例的输入分量x_i，y_d表示第d样例的期望输出值，y'_d表示第d样例的实际输出值，这二者的差值就是"损失（loss）"，也称之为误差（error）。有了公式（7.5）做支撑，图7-9所示的算法就可行之有"章法"了。

如前文所言，对于特定训练集合，第d个样本的预期输出 y_d 和实际输出y'_d，都是"尘埃落定"的常数，对于求权值分量w_i 的偏导（部分导数）来说，除了作为变量 的系数可以保留之外，其他统统都可以"看做浮云化作零"。此外，注意到：

$$w^T\cdot\mathbf{x}_d = w_0x_{d0} + w_1x_{d1} + \ldots w_ix_{di} + \ldots w_nx_{dn} \quad (7.6)$$

因此，公式（7.5）可进一步化简为更加简洁的公式（7.7）：

$$\frac{\partial L}{\partial w_i} = \sum_{d\in D}(y_d - y'_d)(-x_{id}) = -\sum_{d\in D}(y_d - y'_d)x_{id} \quad (7.7)$$

有了公式（7.7）做支撑，梯度下降的权值更新法则可用如公式（7.8）所示:

$$w_i \leftarrow w_i - \eta\frac{\partial L}{\partial w_i} = w_i + \eta\sum_{d\in D}(y_d - y'_d)x_{id} \quad (7.8)$$

有了前面的知识铺垫，我们终于可以在下一章谈谈误差反向传播(Back Propagation，BP)算法了。

7.6 小结

在本章中，我们主要讲解了梯度的概念。所谓梯度，就是该函数值增长最为迅猛的方向，然后我们介绍了梯度下降法则。

在下一章中，我们将用最为通俗易懂的图文并茂的方式，给你详细解释误差反向传播（BP）算法。BP算法不仅仅是作为经典留在我们的记忆里，而且，它还"历久弥新"活在当下。要知道，深度信念网（也就是深度学习）之所以性能奇佳，不仅仅是因为它有一个"无监督"的逐层预训练（unsupervised layer-wise training），除此之外，预训练之后的"微调（fine-tuning）"，还是需要"有监督"的BP算法

作为支撑。

由此可见，BP算法影响之深，以至于"深度学习"都离不开它！

"世上没有白走的路，每一步都算数"。希望你能持续关注。

7.7 请你思考

通过本章的学习，请你思考如下问题：

（1）在前馈神经网络中，隐含层设计多少层、每一层有多少神经元比较合适呢？我们可以设定一种自动确定网络结构的方法吗？

（2）神经网络具有强大的特征表征能力，但"成也萧何，败也萧何"，BP算法常常遭遇"过拟合（over fitting）"，它可能会把噪音当做有效信号，你知道有什么策略来避免过拟合吗？

【参考文献】

[1] 吴军. 智能时代. 中信出版集团. 2016.8

[2] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.

[3] Williams D, Hinton G. Learning representations by back-propagating errors[J]. Nature, 1986, 323(6088): 533-538.

[4] Rumelhart D E, McClelland J L, PDP Research Group. Parallel Distributed Processing, Volume 1 Explorations in the Microstructure of Cognition: Foundations[J]. 1986.

[5] Tom Mitchell著.曾华军等译. 机器学习. 机器工业出版社. 2007.4

[6] Better Explained. Vector Calculus: Understanding the Gradient (https://betterexplained.com/articles/vector-calculus-understanding-the-gradient/?spm=a2c4e.11153940.blogcont105339.17.70863bc0orYPaD)

文章作者：张玉宏，著有《品味大数据》(http://product.dangdang.com/24048575.html?spm=a2c4e.11153940.blogcont105339.18.70863bc0orYPaD)、本文节选自《深度学习之美》(https://item.jd.com/12382640.html?spm=a2c4e.11153940.blogcont105339.19.70863bc0orYPaD)2018年6月电子工业出版社出版。

审校：我是主题曲哥哥。

相关文章

一入侯门"深"似海，深度学习深几许（深度学习入门系列之一）(https://yq.aliyun.com/articles/86580?spm=a2c4e.11153940.blogcont105339.20.70863bc0orYPaD)

人工"碳"索意犹尽，智能"硅"来未可知（深度学习入门系列之二）(https://yq.aliyun.com/articles/88300?spm=a2c4e.11153940.blogcont105339.21.70863bc0orYPaD)

神经网络不胜语， M-P模型似可寻（深度学习入门系列之三）(https://yq.aliyun.com/articles/90565?spm=a2c4e.11153940.blogcont105339.22.70863bc0orYPaD)

"机器学习"三重门，"中庸之道"趋若人（深度学习入门系列之四）(https://yq.aliyun.com/articles/91436?spm=a2c4e.11153940.blogcont105339.23.70863bc0orYPaD)

Hello World感知机，懂你我心才安息(深度学习入门系列之五) (https://yq.aliyun.com/articles/93540?spm=a2c4e.11153940.blogcont105339.24.70863bc0orYPaD)

损失函数减肥用，神经网络调权重（深度学习入门系列之六）(https://yq.aliyun.com/articles/96427?spm=a2c4e.11153940.blogcont105339.25.70863bc0orYPaD)

山重水复疑无路，最快下降问梯度（深度学习入门系列之七）(https://yq.aliyun.com/articles/105339?spm=a2c4e.11153940.blogcont105339.26.70863bc0orYPaD)

BP算法双向传，链式求导最缠绵（深度学习入门系列之八）(https://yq.aliyun.com/articles/110025?spm=a2c4e.11153940.blogcont105339.27.70863bc0orYPaD)