零基础自学神经网络 BP 算法

荣耀学院

2022 年 06 月

第一版序言

深度学习在围棋、图像识别、语音识别、游戏竞技、机器翻译、蛋白质结构预测、药物设计等领域取得令人瞩目的进展，在有些领域，能力已经超过了人类。

深度学习是生产力竞争的明星领域。理解深度学习是竞争的必需。

BP 算法是深度学习的核心。

BP 算法有一定难度，但并不是高不可攀。它的难，难在结构复杂，数学原理并不难，只需要懂一些微分求极值即可，这是微积分的入门内容。毫不夸张地说，找个小学高年级学生，教一点微积分，肯定可以推导出 BP 算法。正常的成年人不可能比小学生差，肯定也可以学会。

深刻理解 BP 算法，只需要记住三个关键词：2-2-1；微分；路径。这三个词，串起了全书的逻辑和细节："2-2-1" 是最小原型神经网络结构；"微分" 是神经网络参数优化方式；"路径" 是神经网络的参数在几何结构上对网络输出产生影响的方式，根据路径可以直接写出参数迭代公式。记住这三个词，对 BP 算法必然是 "莫失莫忘，仙寿恒昌"。

本书更注重于于问题的本质和思考的直觉，而不仅仅是学术式表达。

本书致力于让所有人都能完全掌握 BP 算法。

本书最佳使用方式：拿出纸和笔，亲自推导所有公式。为了便于理解和学习，本书所有的推导给出了所有步骤，不做任何省略。做完这一切，所有人都会觉得 BP 算法象 $1+1=2$ 一样简单、清晰、直接。

目 录

第一章　微分求极值

大多数机器学习问题都可以抽象为一个求极值问题[1]（不是所有问题，比如 K 近邻分类就不是）。求极值，就是求极大值或极小值。两者本质上是一样的，求极小值的目标函数取负就是求极大值。

如果一个可导连续函数有极值，不论极大值还是极小值，那么它在极值点的一阶导数必然是 0，叫零点。沿着函数自变量从小到大的方向，如果一阶导数由负值变为零，那么零点是函数的极小值点，如果一阶导数由正值变为零，那么零点是函数的极大值点。

以数值方式求解零点，任选一个点，这个点所在的一阶导数大概率不为零（其实为零也不要紧），然后用合适的步长移动，移动到一阶导数为 0 的地方。求函数极大值，自变量沿着一阶导数的方向移动，求函数极小值，沿着一阶导数的反方向移动。

以求 $y=(x-2)^2$ 的极小值为例，y 的一阶导数是 $y^{'}=2(x-2)$，迭代公式是

$$x = x - \eta * y^{'} = x - 2\eta(x-2)$$

其中，η 是学习速率，英语发音 |eta|，是一个 $(0,1)$ 区间的实数。学习速率不能太大，太大就是移动的步子大，容易跨过一阶导数零点，取 0.01、0.001 都是可以的，也可以在计算过程动态调整。

用 python 实现 $y=(x-2)^2$ 的极小值数值求解，初始值 $x=10$，$\eta=0.01$，迭代次数 1000 次，代码如下：

```
eta = 0.01
x = 10
iter_n = 1000

for i in range(iter_n):
    print('step', i, 'x=', x)
    x = x - eta * 2 * (x-2)
```

[1]极值不一定是最值，最值一定是极值，一个目标函数可能有多个极值，最大值或者最小值只有一个。

运行结果如下：

```
step 0 x= 10
step 1 x= 9.84
step 2 x= 9.6832
step 3 x= 9.529536
step 4 x= 9.37894528
step 5 x= 9.2313663744
...
step 331 x= 2.0099751868912272
...
step 999 x= 2.000000013738508
```

y 在 $x = 2$ 的时候有全局最小值，运行到第 331 步几乎非常接近了，也可以在 x 的变化很小的时候终止计算。

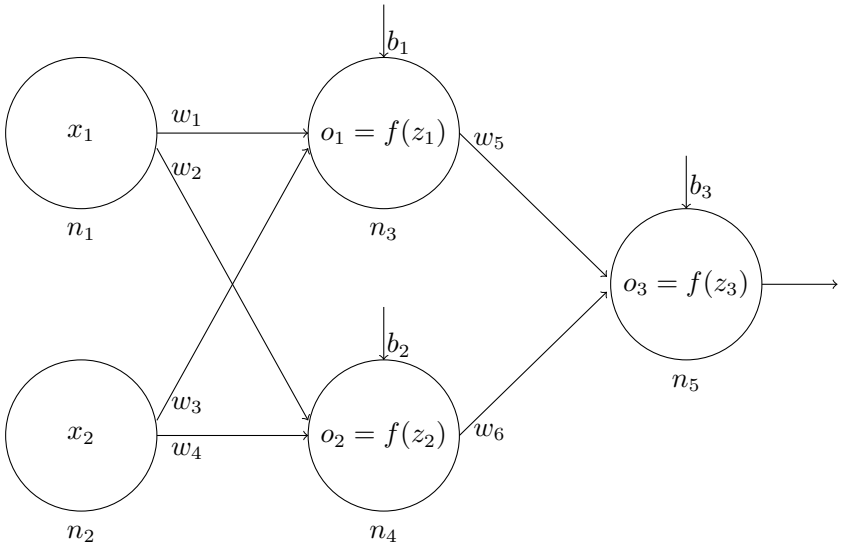
第二章　一个最简神经网络的 BP 算法推导

一个神经网络有多个神经元。每个神经元跟其他神经元的连接方式决定神经网络的类型。比如前馈神经网络，神经网络是多层的，每一层的神经元跟前后层的神经元都有连接关系。

推导 BP 算法，一定要从最简单的具体的神经网络开始，简单的好记，便于搞清楚所有细节，熟悉之后，再推导复杂、通用、抽象的神经网络，这叫最小原型原则。

用一个最简单的前馈神经网络演示 BP 算法推导过程。

这个神经网络只有三层：一个输入层，一个隐层，一个输出层。输入层有两个神经元，n_1 和 n_2，隐层有两个神经元，n_3 和 n_4，输出层有一个神经元，n_5，如下图所示。



神经元 n_1 的输出是 x_1，x_1 通过权重 w_1 和 w_2 分别连接到神经元 n_3 和

n_4 上。神经元 n_2 的输出是 x_2，x_2 通过权重 w_3 和 w_4 分别连接到神经元 n_3 和 n_4 上。

b_1 是神经元 n_3 的偏差，b_2 是神经元 n_4 的偏差。神经元 n_3 的输入是 z_1，z_1 是神经元 n_3 的所有输入 x_1、x_2、b_1 的线性组合，是一个中间变量。神经元 n_3 的输出是 o_1，o_1 通过权重 w_5 连接到神经元 n_5 上。神经元 n_4 的输出是 z_2，z_2 是神经元 n_4 所有输入 x_1、x_2、b_2 的线性组合。神经元 n_4 的输出是 o_2，o_2 通过权重 w_6 连接到神经元 n_5 上。

b_3 是神经元 n_5 的偏差。神经元 n_5 的输入是 z_3。神经元 n_5 的输出是 o_3，o_3 也是整个神经网络的输出。

神经网络的参数有两类：权重，包括 w_1、w_2、w_3、w_4、w_5、w_6；偏差，包括 b_1、b_2、b_3。

假设一个样本，它有两个特征 $[x_1, x_2]$，目标值是 y。神经网络的学习过程，就是多次调整权重和偏差，使得神经网络输入是 $[x_1, x_2]$ 的时候，输出值尽可能接近 y。

从输入层到输出层，做一次前向传播，有如下结果：

$$z_1 = x_1 w_1 + x_2 w_3 + b_1$$

$$o_1 = f(z_1)$$

$$z_2 = x_1 w_2 + x_2 w_4 + b_2$$

$$o_2 = f(z_2)$$

$$z_3 = o_1 w_5 + o_2 w_6 + b_3$$

$$o_3 = f(z_3)$$

其中，$f(x)$ 是神经元的激活函数。激活函数有多种，比如 sigmod 函数：

$$f(x) = \frac{e^x}{1 + e^x}$$

此时，sigmod 函数的一阶导数有一个特性，可以用 sigmod 函数本身表示：

$$f(x)' = f(x)(1 - f(x))$$

这里不做证明了，很简单。

o_3 是神经网络的输出值，其值跟样本目标值 y 是不一样的，训练神经网络即是让它们之间的差异越来越小。用 Err 衡量它们的差异[1]：

[1]Err 可以有多种形式，比如 $|o_3 - y|$，$(o_3 - y)^2$，视需求而定，这里设置为 $\frac{1}{2}(o_3 - y)^2$ 以便于求导求解。

$$Err = \frac{1}{2}(o_3 - y)^2$$

无论 o_3 比 y 大还是小，都可以用 Err 衡量它们之间的差异。

神经网络的训练，是一个求 Err 极小值的问题（对于该函数，求极小值等价于求最小值），也就是说，每一轮训练，都是计算 Err 对 w_1、w_2、w_3、w_4、w_5、b_1、b_2、b_3 的一阶偏导，然后迭代更新。

根据函数微分求极值的规则，权重 w_5、w_6 和偏差 b_3 的迭代公式如下：

$$w_5 = w_5 - \eta\frac{\partial Err}{\partial w_5}$$

$$w_6 = w_6 - \eta\frac{\partial Err}{\partial w_6}$$

$$b_3 = b_3 - \eta\frac{\partial Err}{\partial b_3}$$

其中：

$$\begin{aligned}
\frac{\partial Err}{\partial w_5} &= \frac{\partial(\frac{1}{2}(o_3 - y)^2)}{\partial w_5} \\
&= (o_3 - y)\frac{\partial(o_3 - y)}{\partial w_5} \\
&= (o_3 - y)\frac{\partial o_3}{\partial w_5} \\
&= (o_3 - y)\frac{\partial f(z_3)}{\partial w_5} \\
&= (o_3 - y)f^{'}(z_3)\frac{\partial(o_1 w_5 + o_2 w_6 + b_3)}{\partial w_5} \\
&= (o_3 - y)f^{'}(z_3)o_1
\end{aligned}$$

分析这个推导结果，可以推断一些有意思的“猜想”，先不做证明：

1) 如果 $f(x)$ 是 sigmod 函数，那么它的值域是 $(0,1)$，它的一阶导数 $f(x)^{'} = f(x)(1 - f(x))$ 的值域也是 $(0,1)$，所以 $f^{'}(z_3)$ 和 o_1 都是正数，而且取值范围在 $(0,1)$ 区域。

2) 如果 $o_3 > y$，也就是神经网络的输出值 o_3 比目标值 y 大，又因为 $f^{'}(z_3)$ 和 o_1 都是正数，那么必然有 $\frac{\partial Err}{\partial w_5} > 0$，根据迭代公式 $w_5 = w_5 - \eta\frac{\partial Err}{\partial w_5}$ 可知，迭代的结果让 w_5 变小了。也就是说，如果神经网络输出值比目标值大，迭代让权重 w_5 变小。

3) 同理，如果 $o_3 < y$，神经网络的输出值 o_3 比目标值 y 小，又因为 $f^{'}(z_3)$ 和 o_1 都是正数，那么必然有 $\frac{\partial Err}{\partial w_5} < 0$，根据迭代公式 $w_5 = w_5 - \eta\frac{\partial Err}{\partial w_5}$ 可知，迭代的结果让 w_5 变大。也就是说，如果神经网络输出值比目标值小，迭代让权重 w_5 变大。

4) 根据上两条可以合理地猜测一下，如果神经元的激活函数都是 sigmod 函数，神经网络的所有权重迭代行为可能都象 w_5 一样：如果神经网络输出值比目标值大，迭代让所有权重都变小一些，如果神经网络输出值比目标值小，迭代让所有权重都变大一些。

5) 根据上三条可以合理猜测一下，当迭代到一定次数后，权重迭代值会进入振荡，前一次迭代，导致神经网络输出值比目标值大，后一次迭代，导致神经网络输出值比目标值小。至于具体多少次，是学习速率 η 决定的，如果 η 比较大，小训练次数就会进入振荡，如果 η 比较小，大训练次数才会进入振荡。动态调整 eta 肯定是合理的，一开始取大一点的值，发现进入振荡，再把 η 调小，训练效果会更好。

6) 神经网络的输出值对迭代的影响，体现在推导结果的 $(o_3 - y)$ 上。从直觉上而言，如果神经网络的输出变了，那肯定跟所有的参数都有关，所以所有参数的推导结果都必然包含 $(o_3 - y)$。

7) 推导结果中的 o_1，是跟 w_5 相关的，w_5 连接 o_1 和 n_5。因此 w_5 只受 o_1 的影响，不受 o_2 的影响。

8) 可以合理推测，每个权重的迭代，受到三个影响：神经网络输出误差 $o_3 - y$；跟权重相连的前一个神经元的输出值；跟权重相连的后一个神经元的激活函数的一阶导数。

9) 根据上三条，可以合理猜测 $\frac{\partial Err}{\partial w_6} = (o_3 - y)f^{'}(z_3)o_2$，有待验证。

推导 Err 对 w_6 的一阶偏导：

$$\begin{aligned}
\frac{\partial Err}{\partial w_6} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial w_6} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial w_6} \\
&= (o_3-y)\frac{\partial o_3}{\partial w_6} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial w_6} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_3)}{\partial w_6} \\
&= (o_3-y)f^{'}(z_3)o_2
\end{aligned}$$

推导结果符合前面的猜想。
推导 Err 对 b_3 的一阶偏导:

$$\begin{aligned}
\frac{\partial Err}{\partial b_3} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial b_3} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial b_3} \\
&= (o_3-y)\frac{\partial o_3}{\partial b_3} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial b_3} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_3)}{\partial b_3} \\
&= (o_3-y)f^{'}(z_3)
\end{aligned}$$

隐层的参数 w_1、w_2、w_3、w_4、b_1、b_2 的迭代公式如下:

$$w_1 = w_1 - \eta\frac{\partial Err}{\partial w_1}$$

$$w_2 = w_2 - \eta\frac{\partial Err}{\partial w_2}$$

$$w_3 = w_3 - \eta\frac{\partial Err}{\partial w_3}$$

$$w_4 = w_4 - \eta \frac{\partial Err}{\partial w_4}$$

$$b_1 = b_1 - \eta \frac{\partial Err}{\partial b_1}$$

$$b_2 = b_2 - \eta \frac{\partial Err}{\partial b_2}$$

其中:

$$\begin{aligned}
\frac{\partial Err}{\partial w_1} &= \frac{\partial(\frac{1}{2}(o_3 - y)^2)}{\partial w_1} \\
&= (o_3 - y)\frac{\partial(o_3 - y)}{\partial w_1} \\
&= (o_3 - y)\frac{\partial o_3}{\partial w_1} \\
&= (o_3 - y)\frac{\partial f(z_3)}{\partial w_1} \\
&= (o_3 - y)f^{'}(z_3)\frac{\partial(z_3)}{\partial w_1} \\
&= (o_3 - y)f^{'}(z_3)\frac{\partial(o_1 w_5 + o_2 w_6 + b_3)}{\partial w_1} \\
&= (o_3 - y)f^{'}(z_3)(\frac{\partial(o_1 w_5)}{\partial w_1} + \frac{\partial(o_2 w_6)}{\partial w_1} + \frac{\partial b_3}{\partial w_1}) \\
&= (o_3 - y)f^{'}(z_3)w_5\frac{\partial o_1}{\partial w_1} \\
&= (o_3 - y)f^{'}(z_3)w_5\frac{\partial f(z_1)}{\partial w_1} \\
&= (o_3 - y)f^{'}(z_3)f^{'}(z_1)w_5\frac{\partial(x_1 w_1 + x_2 w_3 + b_1)}{\partial w_1} \\
&= (o_3 - y)f^{'}(z_3)f^{'}(z_1)w_5 x_1
\end{aligned}$$

分析这个推导结果，同样可以推断一些有意思的“猜想”:

1)w_1 受 $o_3 - y$ 影响，跟前面的分析类似。

2)w_1 受跟它相连的前一层神经元输出影响，跟前面的分析类似。在这里，前一层是输入层，跟 w_1 相连的是 x_1。

3)w_1 受跟它相连的后一层神经元的一阶导数的影响，跟前面的分析类似。在这里，是 $f^{'}(z_1)$。

4)w_1 受 $f^{'}(z_3)$ 影响和 w_5 的影响，从神经网络的结构上看，可以合理猜测，从 w_1 到神经网络输出 o_3 的整条路径：x_1，$f(z_1)$，w_5，o_3，都对 w_1 产生影响，根据这条“路径”规则，可以猜测其它权重的推导结果。

5) 根据上一条，可以合理猜测：

$$\frac{\partial Err}{\partial w_2} = (o_3 - y)f^{'}(z_3)f^{'}(z_2)w_6x_1$$

6)BP 算法的 Back Propagation，大概体现在求导结果的相似性：$\frac{\partial Err}{\partial w_1}$、$\frac{\partial Err}{\partial w_5}$ 和 $\frac{\partial Err}{\partial w_6}$，都有 $(o_3 - y)f^{'}(z_3)$。因此，在计算的时候，可以从神经网络的后端向前计算，使用已经算好的结果，节省计算量。

7) 可以观察到，$f^{'}(z_3)$ 和 $f^{'}(z_1)$ 的值域都在 $(0, 1)$ 上，因此它们相乘后，乘积比它们更小。可以合理猜测，如果神经网络的层数比较多，比如几十层或几百层，且激活函数都是 sigmod 函数，那么越往前计算，权重的更新量越小，因为几十或几百个 0 和 1 之间的数相乘，其结果必然趋近于零。如果对训练结果做预估，可以猜测，越靠近输出层，权重跟初始值相比变化越大，越靠近输入层，权重跟初始值相比变化越小。因此，设置太多的隐层是没有意义的，徒然增加了计算量，但并没有什么用处[2]。

推导 Err 对 w_2 的一阶偏导:

[2]学术领域提出了各种解决方式。如设计一个新的激活函数，它一阶导数永远不小于 1，这样就不会出现几百个一阶导数相乘结果趋近于零的情况-梯度消失，当然也不要大于 1，以免相乘结果太大失去意义-梯度爆炸，如 ReLU。BP 算法是从后往前计算权重和偏差，如果靠近输出层的某些权重过早接近零，可能会影响靠近输入层的权重，让它们也倾向于接近零，那可以让远离输出层的隐层，和靠近输出层的隐层建立直接连接，绕过可能病态的神经元，如 ResNet，比如随机删掉一些神经元，也许最好是随机删掉一些梯度接近零的神经元，如 Dropout。

$$
\begin{aligned}
\frac{\partial Err}{\partial w_2} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial w_2} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial w_2} \\
&= (o_3-y)\frac{\partial o_3}{\partial w_2} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(z_3)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_3)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)(w_5\frac{\partial o_1}{\partial w_2}+w_6\frac{\partial o_2}{\partial w_2}+\frac{\partial b_3}{\partial w_2}) \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial o_2}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial f(z_2)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial(z_2)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial(x_1w_2+x_2w_4+b_2)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6x_1
\end{aligned}
$$

推导结果符合前面的猜想。

$$
\begin{aligned}
\frac{\partial Err}{\partial w_3} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial w_3} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial w_3} \\
&= (o_3-y)\frac{\partial o_3}{\partial w_3} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial z_3}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_3)}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)(w_5\frac{\partial o_1}{\partial w_3}+w_6\frac{\partial o_2}{\partial w_3}+\frac{\partial b_3}{\partial w_3}) \\
&= (o_3-y)f^{'}(z_3)w_5\frac{\partial o_1}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)w_5\frac{\partial f(z_1)}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5\frac{\partial z_1}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5\frac{\partial(x_1w_1+x_2w_3+b_1)}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5x_2
\end{aligned}
$$

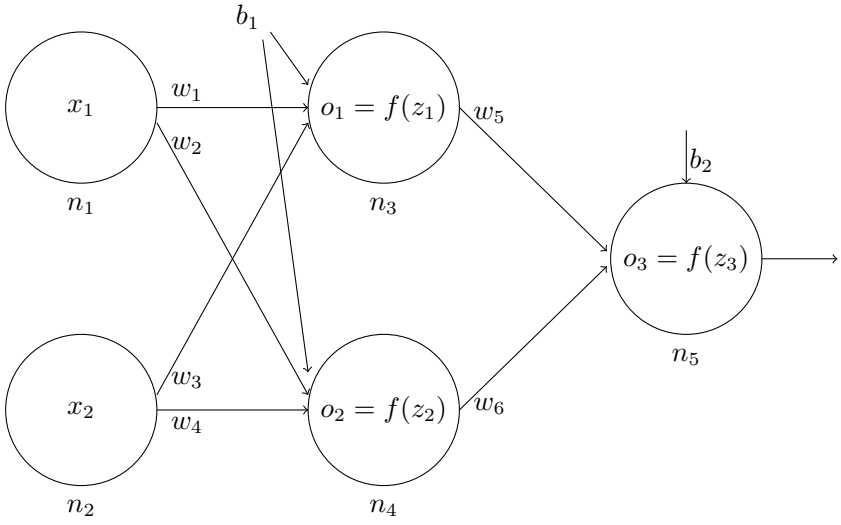
$$
\begin{aligned}
\frac{\partial Err}{\partial w_4} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial w_4} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial w_4} \\
&= (o_3-y)\frac{\partial o_3}{\partial w_4} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial z_3}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_3)}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)(w_5\frac{\partial o_1}{\partial w_4}+w_6\frac{\partial o_2}{\partial w_4}+\frac{\partial b_3}{\partial w_4}) \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial o_2}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial f(z_2)}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial z_2}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial(x_1w_2+x_2w_4+b)}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6x_2
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial Err}{\partial b_1} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial b_1} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial b_1} \\
&= (o_3-y)\frac{\partial o_3}{\partial b_1} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial z_3}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_3)}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)(\frac{\partial(o_1w_5)}{\partial b_1}+\frac{\partial(o_2w_6)}{\partial b_1}+\frac{\partial b_3}{\partial b_1}) \\
&= (o_3-y)f^{'}(z_3)w_5\frac{\partial o_1}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)w_5\frac{\partial f(z_1)}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5\frac{\partial z_1}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5\frac{\partial(x_1w_1+x_2w_3+b_1)}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial Err}{\partial b_2} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial b_2} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial b_2} \\
&= (o_3-y)\frac{\partial o_3}{\partial b_2} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial z_3}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_3)}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)(\frac{\partial(o_1w_5)}{\partial b_2}+\frac{\partial(o_2w_6)}{\partial b_2}+\frac{\partial b_3}{\partial b_2}) \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial o_2}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial f(z_2)}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial z_2}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial(x_1w_2+x_2w_4+b_2)}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6
\end{aligned}
$$

第三章　一个参数更少神经网络的 BP 算法推导

上一个神经网络，每个神经元都有一个 b 值。为减少参数，还可以设置成每层神经网络的神经元共享一个 b 值。如下：



假设一个样本有两个特征 $[x_1, x_2]$，样本标记是 y。那么，从输入层到输出层，做一次前向传播，有如下结果：

$$z_1 = x_1 w_1 + x_2 w_3 + b_1$$

$$o_1 = f(z_1)$$

$$z_2 = x_1 w_2 + x_2 w_4 + b_1$$

$$o_2 = f(z_2)$$

$$z_3 = o_1 w_5 + o_2 w_6 + b_2$$

$$o_3 = f(z_3)$$

其中，$f(x)$ 是神经元的激活函数。

输出层的参数 w_5、w_6、b_2 的迭代公式如下：

$$w_5 = w_5 - \eta \frac{\partial Err}{\partial w_5}$$

$$w_6 = w_6 - \eta \frac{\partial Err}{\partial w_6}$$

$$b_2 = b_2 - \eta \frac{\partial Err}{\partial b_2}$$

其中：

$$\begin{aligned}
\frac{\partial Err}{\partial w_5} &= \frac{\partial(\frac{1}{2}(o_3 - y)^2)}{\partial w_5} \\
&= (o_3 - y)\frac{\partial(o_3 - y)}{\partial w_5} \\
&= (o_3 - y)\frac{\partial o_3}{\partial w_5} \\
&= (o_3 - y)\frac{\partial f(z_3)}{\partial w_5} \\
&= (o_3 - y)f^{'}(z_3)\frac{\partial(o_1 w_5 + o_2 w_6 + b_2)}{\partial w_5} \\
&= (o_3 - y)f^{'}(z_3)o_1
\end{aligned}$$

$$\begin{aligned}
\frac{\partial Err}{\partial w_6} &= \frac{\partial(\frac{1}{2}(o_3 - y)^2)}{\partial w_6} \\
&= (o_3 - y)\frac{\partial(o_3 - y)}{\partial w_6} \\
&= (o_3 - y)\frac{\partial o_3}{\partial w_6} \\
&= (o_3 - y)\frac{\partial f(z_3)}{\partial w_6} \\
&= (o_3 - y)f^{'}(z_3)\frac{\partial(o_1 w_5 + o_2 w_6 + b_2)}{\partial w_6} \\
&= (o_3 - y)f^{'}(z_3)o_2
\end{aligned}$$

$$\begin{aligned}
\frac{\partial Err}{\partial b_2} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial b_2} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial b_2} \\
&= (o_3-y)\frac{\partial o_3}{\partial b_2} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_2)}{\partial b_2} \\
&= (o_3-y)f^{'}(z_3)
\end{aligned}$$

隐层的参数 w_1、w_2、w_3、w_4、b_1 的迭代公式如下:

$$w_1 = w_1 - \eta\frac{\partial Err}{\partial w_1}$$

$$w_2 = w_2 - \eta\frac{\partial Err}{\partial w_2}$$

$$w_3 = w_3 - \eta\frac{\partial Err}{\partial w_3}$$

$$w_4 = w_4 - \eta\frac{\partial Err}{\partial w_4}$$

$$b_1 = b_1 - \eta\frac{\partial Err}{\partial b_1}$$

其中:

$$
\begin{aligned}
\frac{\partial Err}{\partial w_1} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial w_1} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial w_1} \\
&= (o_3-y)\frac{\partial o_3}{\partial w_1} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial w_1} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(z_3)}{\partial w_1} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_2)}{\partial w_1} \\
&= (o_3-y)f^{'}(z_3)(\frac{\partial(o_1w_5)}{\partial w_1}+\frac{\partial(o_2w_6)}{\partial w_1}+\frac{\partial b_2}{\partial w_1}) \\
&= (o_3-y)f^{'}(z_3)w_5\frac{\partial o_1}{\partial w_1} \\
&= (o_3-y)f^{'}(z_3)w_5\frac{\partial f(z_1)}{\partial w_1} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5\frac{\partial(x_1w_1+x_2w_3+b_1)}{\partial w_1} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5x_1
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial Err}{\partial w_2} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial w_2} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial w_2} \\
&= (o_3-y)\frac{\partial o_3}{\partial w_2} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(z_3)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_2)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)(w_5\frac{\partial o_1}{\partial w_2}+w_6\frac{\partial o_2}{\partial w_2}+\frac{\partial b_2}{\partial w_2}) \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial o_2}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial f(z_2)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial(z_2)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial(x_1w_2+x_2w_4+b_1)}{\partial w_2} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6x_1
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial Err}{\partial w_3} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial w_3} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial w_3} \\
&= (o_3-y)\frac{\partial o_3}{\partial w_3} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial z_3}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_2)}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)(w_5\frac{\partial o_1}{\partial w_3}+w_6\frac{\partial o_2}{\partial w_3}+\frac{\partial b_2}{\partial w_3}) \\
&= (o_3-y)f^{'}(z_3)w_5\frac{\partial o_1}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)w_5\frac{\partial f(z_1)}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5\frac{\partial z_1}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5\frac{\partial(x_1w_1+x_2w_3+b_1)}{\partial w_3} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_1)w_5x_2
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial Err}{\partial w_4} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial w_4} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial w_4} \\
&= (o_3-y)\frac{\partial o_3}{\partial w_4} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial z_3}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_2)}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)(w_5\frac{\partial o_1}{\partial w_4}+w_6\frac{\partial o_2}{\partial w_4}+\frac{\partial b_2}{\partial w_4}) \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial o_2}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)w_6\frac{\partial f(z_2)}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial z_2}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6\frac{\partial(x_1w_2+x_2w_4+b_1)}{\partial w_4} \\
&= (o_3-y)f^{'}(z_3)f^{'}(z_2)w_6x_2
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial Err}{\partial b_1} &= \frac{\partial(\frac{1}{2}(o_3-y)^2)}{\partial b_1} \\
&= (o_3-y)\frac{\partial(o_3-y)}{\partial b_1} \\
&= (o_3-y)\frac{\partial o_3}{\partial b_1} \\
&= (o_3-y)\frac{\partial f(z_3)}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial z_3}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)\frac{\partial(o_1w_5+o_2w_6+b_2)}{\partial b_1} \\
&= (o_3-y)f^{'}(z_3)(\frac{\partial(o_1w_5)}{\partial b_1}+\frac{\partial(o_2w_6)}{\partial b_1}+\frac{\partial b_2}{\partial b_1}) \\
&= (o_3-y)f^{'}(z_3)(w_5\frac{\partial o_1}{\partial b_1}+w_6\frac{\partial o_2}{\partial b_1}) \\
&= (o_3-y)f^{'}(z_3)(w_5\frac{\partial f(z_1)}{\partial b_1}+w_6\frac{\partial f(z_2)}{\partial b_1}) \\
&= (o_3-y)f^{'}(z_3)(f^{'}(z_1)w_5\frac{\partial z_1}{\partial b_1}+f^{'}(z_2)w_6\frac{\partial z_2}{\partial b_1}) \\
&= (o_3-y)f^{'}(z_3)(f^{'}(z_1)w_5\frac{\partial(x_1w_1+x_2w_3+b_1)}{\partial b_1}+f^{'}(z_2)w_6\frac{\partial(x_1w_2+x_2w_4+b_1)}{\partial b_1}) \\
&= (o_3-y)f^{'}(z_3)(f^{'}(z_1)w_5+f^{'}(z_2)w_6)
\end{aligned}
$$

分析这个推导结果，发现一个跟前面不一样的东西：从神经网络的结构图上看，偏差 b_1 是经过两条“路径”传导到 o_3 的，因此，体现在推导结果，就是两条路径微分结果之和。

如何训练神经网络?

用一个样本训练神经网络，设定参数之后，多次训练，直到神经网络的输出跟样本标记的差别足够小为止。

用多个样本训练神经网络，本质上跟用一个样本训练神经网络是一样的：先用第一个样本训练，等训练到误差足够小，再用第二个样本训练... 直到用完所有样本。或者是，用所有样本训练一次，再用所有样本训练第二次，再用所有样本训练第三次.... 直到误差足够小...

如果样本特别多，可以把样本随机拆分成多份，一份份训练。每训练完一份，看看神经网络对其它份的预测误差是不是足够小，如果足够小，可以不用

继续训练了，节省时间。

这里给出一个用 Python 实现神经网络的一个样本训练过程，为便于理解，代码不做任何优化。参考这个例子，不难实现多样本训练，也不难实现更复杂神经网络的训练。

```
import random
import math

#训练样本[x1,x2],y = [3,2],0.75
x1 = 3
x2 = 2
y = (x1*x1+4*x2*x2+5)/40

def active(x):
    return math.exp(x)/(1+math.exp(x))

def dif_active(x):
    return active(x)*(1-active(x))

w1 =random.random()
w2 =random.random()
w3 =random.random()
w4 =random.random()
w5 =random.random()
w6 =random.random()
b1 =random.random()
b2 =random.random()
eta = 0.1

i = 0
while True:
    if i > 1000:
        break
    z1 = x1*w1 + x2*w3 + b1
    o1 = active(z1)
    z2 = x1*w2 + x2*w4 + b1
    o2 = active(z2)
    z3 = o1*w5 + o2*w6 + b2
    o3 = active(z3)
    Err = 0.5*pow((o3-y),2)
    print('i=', i,'o3=', o3, 'Err = ', Err, 'w5=', w5)
    w5 = w5 - eta * (o3 - y) * dif_active(z3) * o1
    w6 = w6 - eta * (o3 - y) * dif_active(z3) * o2
    b2 = b2 - eta * (o3 - y) * dif_active(z3)
    w4 = w4 - eta * (o3 - y) * dif_active(z3) * dif_active(z2) * w6 * x2
    w3 = w3 - eta * (o3 - y) * dif_active(z3) * dif_active(z1) * w5 * x2
    w2 = w2 - eta * (o3 - y) * dif_active(z3) * dif_active(z2) * w6 * x1
    w1 = w1 - eta * (o3 - y) * dif_active(z3) * dif_active(z1) * w5 * x1
    b1 = b1 - eta * (o3 - y) * dif_active(z3) * (dif_active(z1) * w5 +
                                            dif_active(z2) * w6)
    i+=1
```

代码运行，输出结果如下：

```
i= 0 o3= 0.8786813544239055 Err =  0.008279445488185391 w5= 0.7319211214325513
i= 1 o3= 0.8782648809204359 Err =  0.008225939838766795 w5= 0.7306353505605077
i= 2 o3= 0.877847334659751 Err =  0.008172470489801178 w5= 0.7293500220853857
i= 3 o3= 0.8774287244970262 Err =  0.008119039913469508 w5= 0.7280651657661371
i= 4 o3= 0.8770090594646875 Err =  0.00806565059305227 w5= 0.7267808115271741
...
i= 782 o3= 0.7501852654530606 Err =  1.7161644048881098e-08 w5= 0.
                                       440452665262092
i= 783 o3= 0.7501834540098297 Err =  1.6827686861291304e-08 w5= 0.
                                       4404494492897204
...
```

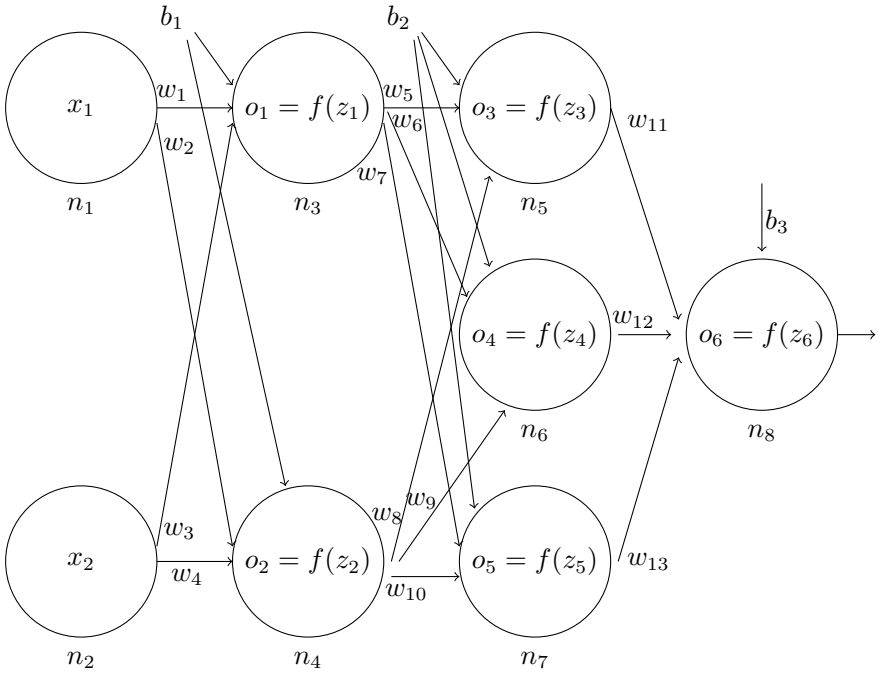
可以观察到当训练次数达到 782 次的时候，神经网络的输出非常接近样本目标值 0.75。

为了观察神经网络输出和权重迭代的变化关系，代码显示 w_5 的迭代值，可以发现，$o3$ 比 y 大的时候，w_5 是逐步变小的，跟前面的猜想是一致的。

第四章　两个隐层神经网络 BP 迭代公式的猜测与验证

前面分析的两个神经网络都是一个隐层，得到了一些有意思的猜想，也验证了猜想是合理的。

如果神经网络更复杂一点，这些猜想对不对？设计一个两个隐层的神经网络验证一下，结构如下图：



我们从前面得到的“路径”规则，继续用下去。

从 w_5 到神经网络的输出 o_6，只有一条路径：o_1、$f^{'}(z_3)$、w_{11}、$f^{'}(z_6)$、$o_6 - y$，所以 Err 对 w_5 的一阶偏导可以直接写出来：

$$\begin{aligned}\frac{\partial Err}{\partial w_5} &= (o_6-y)f^{'}(z_6)w_{11}f^{'}(z_3)o_1 \\ &= (o_6-y)f^{'}(z_6)f^{'}(z_3)w_{11}o_1\end{aligned}$$

w_6、w_7、w_8、w_9、w_{10}、w_{11}、w_{12}、w_{13}、b_3，跟 w_5 是一样的，也都是一条路径。b_2 到 o_6 是三条路径，跟第二个神经网络的 b_1 推导类似。这些推导结果，就不一一写出来了，体力活。

w_1 跟以前不一样了，从它到 o_6 有三条路径，分别是：

1)x_1，$f(z_1)$，w_5，$f(z_3)$，w_{11}，$f(z_6)$，o_6-y；

2)x_1，$f(z_1)$，w_6，$f(z_4)$，w_{12}，$f(z_6)$，o_6-y；

3)x_1，$f(z_1)$，w_7，$f(z_5)$，w_{13}，$f(z_6)$，o_6-y。

猜测 w_1 的推导结果是三条路径的累加，可以根据路径直接写下来：

$$\begin{aligned}\frac{\partial Err}{\partial w_1} &= x_1f^{'}(z_1)w_5f^{'}(z_3)w_{11}f^{'}(z_6)(o_6-y)+ \\ &\quad x_1f^{'}(z_1)w_6f^{'}(z_4)w_{12}f^{'}(z_6)(o_6-y)+ \\ &\quad x_1f^{'}(z_1)w_7f^{'}(z_5)w_{13}f^{'}(z_6)(o_6-y) \\ &= x_1f^{'}(z_1)(w_5f^{'}(z_3)w_{11}f^{'}(z_6)(o_6-y)+ \\ &\qquad w_6f^{'}(z_4)w_{12}f^{'}(z_6)(o_6-y)+ \\ &\qquad w_7f^{'}(z_5)w_{13}f^{'}(z_6)(o_6-y)) \\ &= x_1f^{'}(z_1)(w_5f^{'}(z_3)w_{11}+w_6f^{'}(z_4)w_{12}+w_7f^{'}(z_5)w_{13})f^{'}(z_6)(o_6-y)\end{aligned}$$

如果对 w_1 推导结果是这个结果，表明猜想正确。推导一下，看看是不是如此：

$$
\begin{aligned}
\frac{\partial Err}{\partial w_1} &= \frac{\partial(\frac{1}{2}(o_6-y)^2)}{\partial w_1} \\
&= (o_6-y)\frac{\partial(o_6-y)}{\partial w_1} \\
&= (o_6-y)\frac{\partial o_6}{\partial w_1} \\
&= (o_6-y)\frac{\partial f(z_6)}{\partial w_1} \\
&= (o_6-y)f^{'}(z_6)\frac{\partial z_6}{\partial w_1} \\
&= (o_6-y)f^{'}(z_6)\frac{\partial(o_3w_{11}+o_4w_{12}+o_5w_{13}+b_3)}{\partial w_1} \\
&= (o_6-y)f^{'}(z_6)(w_{11}\frac{\partial o_3}{\partial w_1}+w_{12}\frac{\partial o_4}{\partial w_1}+w_{13}\frac{\partial o_5}{\partial w_1}) \\
&= (o_6-y)f^{'}(z_6)(w_{11}f^{'}(z_3)\frac{\partial z_3}{\partial w_1}+w_{12}f^{'}(z_4)\frac{\partial z_4}{\partial w_1}+w_{13}f^{'}(z_5)\frac{\partial z_5}{\partial w_1}) \\
&= (o_6-y)f^{'}(z_6)(w_{11}f^{'}(z_3)\frac{\partial z_3}{\partial w_1}+w_{12}f^{'}(z_4)\frac{\partial z_4}{\partial w_1}+w_{13}f^{'}(z_5)\frac{\partial z_5}{\partial w_1}) \\
&= (o_6-y)f^{'}(z_6)(w_{11}f^{'}(z_3)\frac{\partial(w_5o_1+w_8o_2+b_2)}{\partial w_1}+ \\
&\qquad w_{12}f^{'}(z_4)\frac{\partial(w_6o_1+w_9o_2+b_2)}{\partial w_1}+ \\
&\qquad w_{13}f^{'}(z_5)\frac{\partial(w_7o_1+w_{10}o_2+b_2)}{\partial w_1}) \\
&= (o_6-y)f^{'}(z_6)(w_{11}f^{'}(z_3)\frac{\partial(w_5o_1)}{\partial w_1}+w_{12}f^{'}(z_4)\frac{\partial(w_6o_1)}{\partial w_1}+w_{13}f^{'}(z_5)\frac{\partial(w_7o_1)}{\partial w_1}) \\
&= (o_6-y)f^{'}(z_6)(w_{11}f^{'}(z_3)w_5\frac{\partial o_1}{\partial w_1}+w_{12}f^{'}(z_4)w_6\frac{\partial o_1}{\partial w_1}+w_{13}f^{'}(z_5)w_7\frac{\partial o_1}{\partial w_1}) \\
&= (o_6-y)f^{'}(z_6)(w_{11}f^{'}(z_3)w_5f^{'}(z_1)\frac{\partial z_1}{\partial w_1}+ \\
&\qquad w_{12}f^{'}(z_4)w_6f^{'}(z_1)\frac{\partial z_1}{\partial w_1}+ \\
&\qquad w_{13}f^{'}(z_5)w_7f^{'}(z_1)\frac{\partial z_1}{\partial w_1}) \\
&= (o_6-y)f^{'}(z_6)(w_{11}f^{'}(z_3)w_5f^{'}(z_1)+w_{12}f^{'}(z_4)w_6f^{'}(z_1)+w_{13}f^{'}(z_5)w_7f^{'}(z_1))\frac{\partial z_1}{\partial w_1} \\
&= (o_6-y)f^{'}(z_6)(w_{11}f^{'}(z_3)w_5+w_{12}f^{'}(z_4)w_6+w_{13}f^{'}(z_5)w_7)f^{'}(z_1)x_1 \\
&= x_1f^{'}(z_1)(w_5f^{'}(z_3)w_{11}+w_6f^{'}(z_4)w_{12}+w_7f^{'}(z_5)w_{13})f^{'}(z_6)(o_6-y)
\end{aligned}
$$

跟猜想一致，因此根据路径累加计算 w_1 的推导是合理的。

w_2、w_3、w_4、b_1 跟 w_1 的逻辑是一样的，结果不一一写了，体力活。

迄今为止，推导了三个神经网络，可以再分析点有意思的东西。

输出层的神经元是 n_8，它有四个输入，逐一写目标函数 Err 对它们一阶偏导：

$$\frac{\partial Err}{\partial w_{11}} = (o_6 - y)f^{'}(z_6)o_3$$

$$\frac{\partial Err}{\partial w_{12}} = (o_6 - y)f^{'}(z_6)o_4$$

$$\frac{\partial Err}{\partial w_{13}} = (o_6 - y)f^{'}(z_6)o_5$$

$$\frac{\partial Err}{\partial b_3} = (o_6 - y)f^{'}(z_6)$$

b_3 的公式，跟其它三个略有区别，如果改写成：

$$\frac{\partial Err}{\partial b_3} = (o_6 - y)f^{'}(z_6) \cdot 1$$

那么 b_3 也可以视为一种特殊的“权重”。

把 b_3 也视为权重，那么 n_8 的四个输入也就统一了：w_{11}、w_{12}、w_{13}、b_3 有两端，一端连接到 n_8 上，另一端是一个常数，w_{11}、w_{12}、w_{13} 的常数是神经网络运行时计算出来的，b_3 的常数是固定值 1。

Err 对它们的一阶偏导分为两部分，一部分是这些常数，另一部分是 $(o_6 - y)f^{'}(z_6)$。对神经元 n_8 而言，$(o_6 - y)$ 是输出，$f^{'}(z_6)$ 是神经元激活函数的一阶导数，以 n_8 中心，抽象而言又分别代表后续环节的“所有结果”和“自身”的关系，这个视角又可以泛化到所有神经元上。对于计算而言，为了方便起见，

可以给 n_8 增加一个“附加变量”，记录 $(o_6 - y)f^{'}(z_6)$ 的值，这样 w_{11}、w_{12}、w_{13}、b_3 一阶微分推导结果只需要关注 n_8 本身即可，而不再考虑 n_8 的后续环节。那么就得到一个重要结论：对任意一个神经元而言，推导权重的迭代公式，可以形式上只跟“权重”的两端有关。这就大大简化了操作。

再继续往下：

神经元 n_5 的“附加变量”是 $(o_6 - y)f^{'}(z_6)w_{11}f^{'}(z_3)$。

根据前面的规则，w_5 一端是 o_1，另一端是 n_5 的“附加变量”，根据两端可以直接写出 w_5 的一阶偏导：

$$\frac{\partial Err}{\partial w_5} = (o_6 - y)f^{'}(z_6)w_{11}f^{'}(z_3)o_1$$

神经元 n_6 的“附加变量”是 $(o_6 - y)f^{'}(z_6)w_{12}f^{'}(z_4)$。

同理，可以直接写出 w_6 的一阶偏导：

$$\frac{\partial Err}{\partial w_6} = (o_6 - y)f^{'}(z_6)w_{12}f^{'}(z_4)o_1$$

神经元 n_7 的“附加变量”是 $(o_6 - y)f^{'}(z_6)w_{13}f^{'}(z_5)$。

同理，可以直接写出 w_7 的一阶偏导：

$$\frac{\partial Err}{\partial w_7} = (o_6 - y)f^{'}(z_6)w_{13}f^{'}(z_5)o_1$$

加速一下：

$$\frac{\partial Err}{\partial w_8} = (o_6 - y)f^{'}(z_6)w_{11}f^{'}(z_3)o_2$$

$$\frac{\partial Err}{\partial w_9} = (o_6 - y)f^{'}(z_6)w_{12}f^{'}(z_4)o_2$$

$$\frac{\partial Err}{\partial w_{10}} = (o_6 - y)f^{'}(z_6)w_{13}f^{'}(z_5)o_2$$

b_2 有些特殊，是三条路径叠加：

$$\frac{\partial Err}{\partial b_2} = (o_6 - y)f^{'}(z_6)(w_{11}f^{'}(z_3) + w_{12}f^{'}(z_4) + w_{13}f^{'}(z_5))$$

再继续：

神经元 n_3 的 “附加变量” 是 $(o_6 - y)f^{'}(z_6)(w_{11}f^{'}(z_3)w_5 + w_{12}f^{'}(z_4)w_6 + w_{13}f^{'}(z_5)w_7)f^{'}(z_1)$

那么，直接写 w_1 的一阶偏导：

$$\frac{\partial Err}{\partial w_1} = (o_6 - y)f^{'}(z_6)(w_{11}f^{'}(z_3)w_5 + w_{12}f^{'}(z_4)w_6 + w_{13}f^{'}(z_5)w_7)f^{'}(z_1)x_1$$

跟本章开始部分的推导结果相同。

其它的 w_2、w_3、w_4、b_1，与此同理，不一一列出，体力活。

现在几乎得到 BP 算法的完整拼图了，只缺一个东西了–推导一个通用的抽象的神经网络的 BP 算法。鉴于我们从最小原型得到的经验和结论，在正式推导之前我们已经对推导结果了然于胸，毫无困难，只需要走个流程。这么看来，BP 算法不难吧？

第五章　通用神经网络的 BP 算法推导

这一章会让大家体会到一点痛苦。“通用”会让表达相当复杂，这也是大家记不住 BP 算法，面试的时候推不出的原因。

从前几章可知：

1）神经网络的参数是各层神经元之间的权重和偏差。

2) 偏差可以视为特殊的权重。

3) 目标函数 Err 对权重的一阶偏导，只跟权重连接的两个神经元相关：是前一个神经元的输出和后一个神经元“附加变量”的乘积。

4) 神经网络前向计算阶段，计算出所有神经元的输出，其中输入层神经元的输出是样本特征值，不需要计算，直接赋值。

5) 神经网络反向传播阶段，计算出所有隐层和输出层神经元的“附加变量”，其中输入层神经元的“附加变量”不用计算，因为输入层之前没有权重，不需要更新权重。

6) 根据神经元的输出和“附加变量”，计算所有权重和偏差的增量，更新权重和偏差。

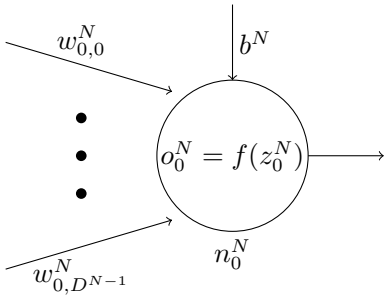
从 5) 可知，BP 算法的关键是计算“附加变量”，计算附加变量的关键有两点：计算输出层的附加变量；前一层神经元的附加变量，跟后一层神经元的附加变量是什么关系。有了这两点，就能从后向前，逐层计算所有神经元的附加变量了。

先计算输出层的附加变量。

假设神经网络有 $N+1$ 层（用 $N+1$ 而不是 N，是为了方便下文的符号标记，以下类同），其中，输入层是第 0 层，输出层是第 N 层。

先用一个最简单的情况示意一下推导过程：输出层只有一个神经元，只推导输出层的“附加变量”。

绘制输出层，结构是这样的：



其中，o_0^N 表示第 N 层的第 0 个神经元的输出，z_0^N 表示第 N 层的第 0 个神经元的输入，$o_0^N = f(z_0^N)$。

$D^N + 1$ 表示第 N 层神经元的数量。$D^{N-1} + 1$ 表示第 $N-1$ 层神经元的数量。$w_{0,0}^N$ 表示第 $N-1$ 层的第 0 个神经元和第 N 层的第 0 个神经元的权重，$w_{0,D^{N-1}}^N$ 第 $N-1$ 层的第 D^{N-1} 个神经元 (也就是最后一个神经元) 和第 N 层的第 0 个神经元的权重。o_0^{N-1} 表示表示第 $N-1$ 层的第 0 个神经元的输出，$o_{D^{N-1}}^{N-1}$ 表示第 $N-1$ 层的第 D^{N-1} 个神经元（也就是最后一个神经元）的输出。b^N 表示第 N 层的偏差。那么就有:

$$z_0^N = \sum_{i=0}^{D^{N-1}} w_{0,i}^N o_i^{N-1} + b^N$$

$$o_0^N = f(z_0^N)$$

$$Err = \frac{1}{2}(o_0^N - y)^2$$

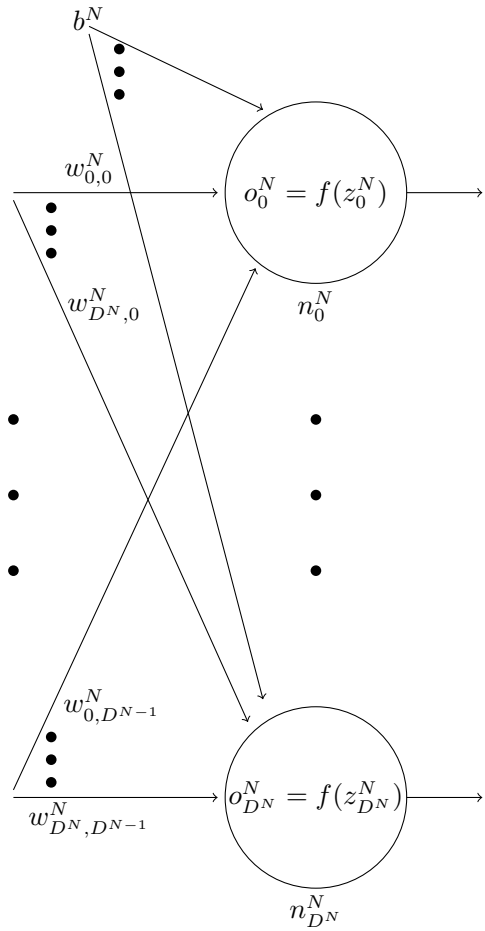
$w_{0,i}^N$ 是连接第 $N-1$ 层第 i 个神经元和第 N 层第 0 个神经元的权重，求 Err 对它的一阶偏导，如下:

$$
\begin{aligned}
\frac{\partial Err}{\partial w_{0,i}^N} &= \frac{\partial Err}{\partial o_0^N}\frac{\partial o_0^N}{\partial w_{0,i}^N} \\
&= \frac{\partial Err}{\partial o_0^N}\frac{\partial f(z_0^N)}{\partial w_{0,i}^N} \\
&= \frac{\partial Err}{\partial o_0^N} f^{'}(z_0^N)\frac{\partial z_0^N}{\partial w_{0,i}^N} \\
&= \frac{\partial Err}{\partial o_0^N} f^{'}(z_0^N)\frac{\partial(\sum_{n=0}^{D^{N-1}} w_{0,n}^N o_n^{N-1} + b^N)}{\partial w_{0,i}^N} \\
&= \frac{\partial Err}{\partial o_0^N} f^{'}(z_0^N) o_i^{N-1} \\
&= \frac{\partial(\frac{1}{2}(o_0^N - y)^2)}{\partial o_0^N} f^{'}(z_0^N) o_i^{N-1} \\
&= (o_0^N - y) f^{'}(z_0^N) o_i^{N-1}
\end{aligned}
$$

由推导可知，输出层神经元 n_0^N 的“附加变量”是 $(o_0^N - y)f^{'}(z_0^N)$。用 a 表示“附加变量”，则 a_0^N 表示第 N 层第 0 个神经元的附加变量：

$$a_0^N = (o_0^N - y)f^{'}(z_0^N)$$

现在推导最复杂的神经网络：输出层有多个神经元；每个隐层也有多个神经元；隐层数量至少一个。推导主要求解两个问题：推导输出层神经元的附加变量；推导前一层神经元和后一层神经元的附加变量之间的关系。



输出层有多个神经元，也就有多个输出，因此 Err 需要略做修改，如下：

$$Err = \frac{1}{2}\sum_{i=0}^{D^N}(o_i^N - y_i)^2$$

$w^N_{j,i}$ 是连接第 $N-1$ 层第 i 个神经元和第 N 层第 j 个神经元的权重。

$$
\begin{aligned}
\frac{\partial Err}{\partial w_{j,i}^N} &= \frac{\partial Err}{\partial o_j^N}\frac{\partial o_j^N}{\partial w_{j,i}^N} \\
&= \frac{\partial Err}{\partial o_j^N}\frac{\partial f(z_j^N)}{\partial w_{j,i}^N} \\
&= \frac{\partial Err}{\partial o_j^N} f^{'}(z_j^N)\frac{\partial z_j^N}{\partial w_{j,i}^N} \\
&= \frac{\partial Err}{\partial o_j^N} f^{'}(z_j^N)\frac{\partial(\sum_{l=0}^{D^{N-1}} w_{j,l}^N o_l^{N-1} + b^N)}{\partial w_{j,i}^N} \\
&= \frac{\partial Err}{\partial o_j^N} f^{'}(z_j^N)\frac{\partial(\sum_{l=0}^{D^{N-1}} w_{j,l}^N o_l^{N-1})}{\partial w_{j,i}^N} \\
&= \frac{\partial Err}{\partial o_j^N} f^{'}(z_j^N)\frac{\partial(w_{j,i}^N o_i^{N-1})}{\partial w_{j,i}^N} \\
&= \frac{\partial Err}{\partial o_j^N} f^{'}(z_j^N) o_i^{N-1} \\
&= \frac{\partial(\frac{1}{2}\sum_{l=0}^{D^N}(o_l^N - y_l)^2)}{\partial o_j^N} f^{'}(z_j^N) o_i^{N-1} \\
&= \frac{\partial(\frac{1}{2}\sum_{l=0}^{D^N}(o_l^N - y_l)^2)}{\partial o_j^N} f^{'}(z_j^N) o_i^{N-1} \\
&= (o_j^N - y_j) f^{'}(z_j^N) o_i^{N-1}
\end{aligned}
$$

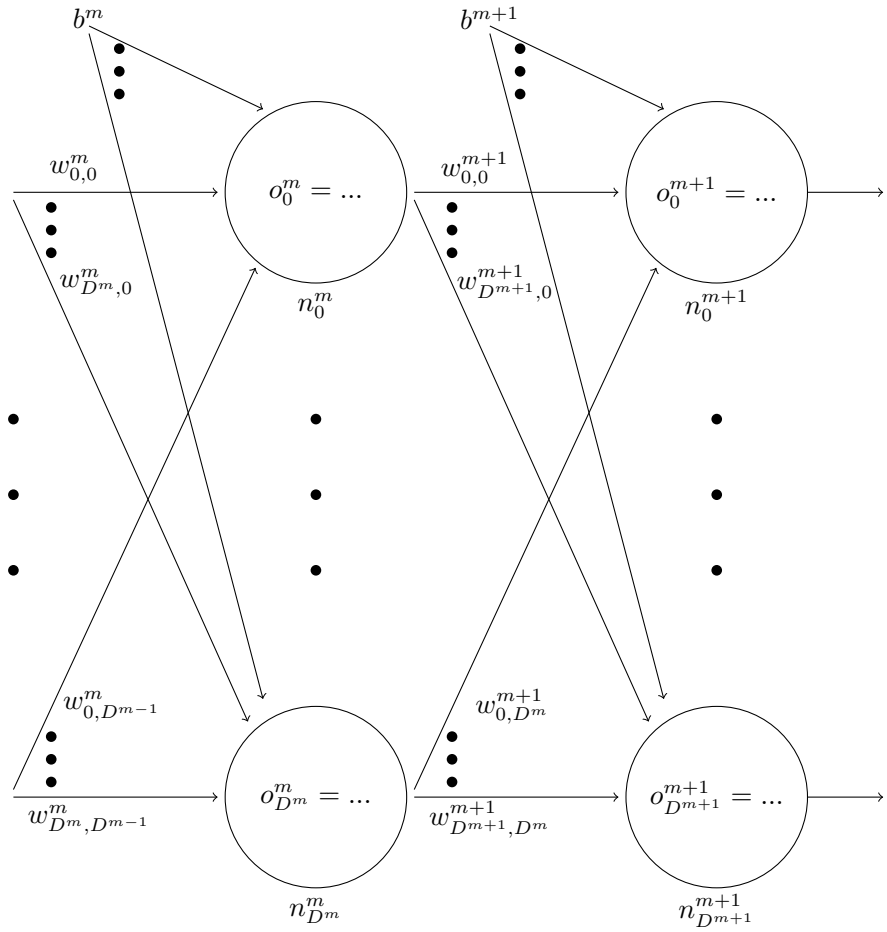
令 a_j^N 表示第 N 层第 j 个神经元的附加变量:

$$
a_j^N = (o_j^N - y_j) f^{'}(z_j^N)
$$

则:

$$
\frac{\partial Err}{\partial w_{j,i}^N} = a_j^N o_i^{N-1}
$$

对 $w_{j,i}^N$ 的推导结果，符合我们再前几章总结的路径规则。
b^N 的推导同理，不一列出来了，体力活。
再推导相邻两层的神经元的附加变量之间的关系。



设两个层的序号分别为 m 和 $m+1$。由于 $m+1$ 是后一层，因此在计算两层关系的时候，后一层所有神经元的附加变量 $a_j^{m+1}(j=0,...,D^{m+1}+1)$ 已经计算出来了，是已知的。

$w^m_{j,i}$ 是连接第 $m-1$ 层第 i 个神经元和第 m 层第 j 个神经元的权重。

$$
\begin{aligned}
\frac{\partial Err}{\partial w_{j,i}^m} &= \frac{\partial Err}{\partial o_j^m}\frac{\partial o_j^m}{\partial w_{j,i}^m} \\
&= \frac{\partial Err}{\partial o_j^m}\frac{\partial f(z_j^m)}{\partial w_{j,i}^m} \\
&= \frac{\partial Err}{\partial o_j^m} f^{'}(z_j^m)\frac{\partial z_j^m}{\partial w_{j,i}^m} \\
&= \frac{\partial Err}{\partial o_j^m} f^{'}(z_j^m)\frac{\partial(\sum_{l=0}^{D^{m-1}} w_{i,l}^m o_l^{m-1} + b^m)}{\partial w_{j,i}^m} \\
&= \frac{\partial Err}{\partial o_j^m} f^{'}(z_j^m)\frac{\partial(\sum_{l=0}^{D^{m-1}} w_{i,l}^m o_l^{m-1})}{\partial w_{j,i}^m} \\
&= \frac{\partial Err}{\partial o_j^m} f^{'}(z_j^m) o_i^{m-1} \\
&= (\frac{\partial Err}{\partial o_0^{m+1}}\frac{\partial o_0^{m+1}}{\partial o_j^m} + ... + \frac{\partial Err}{\partial o_{D^{m+1}}^{m+1}}\frac{\partial o_{D^{m+1}}^{m+1}}{\partial o_j^m}) f^{'}(z_j^m) o_i^{m-1} \\
&= (\sum_{l=0}^{D^{m+1}} (\frac{\partial Err}{\partial o_l^{m+1}}\frac{\partial o_l^{m+1}}{\partial o_j^m})) f^{'}(z_j^m) o_i^{m-1} \\
&= (\sum_{l=0}^{D^{m+1}} (\frac{\partial Err}{\partial o_l^{m+1}}\frac{\partial f(z_l^{m+1})}{\partial o_j^m})) f^{'}(z_j^m) o_i^{m-1} \\
&= (\sum_{l=0}^{D^{m+1}} (\frac{\partial Err}{\partial o_l^{m+1}} f'(z_l^{m+1})\frac{\partial z_l^{m+1}}{\partial o_j^m})) f^{'}(z_j^m) o_i^{m-1} \\
&= (\sum_{l=0}^{D^{m+1}} (\frac{\partial Err}{\partial o_l^{m+1}} f'(z_l^{m+1})\frac{\partial(\sum_{k=0}^{D^{m+1}} w_{l,k}^{m+1} o_k^m + b^{m+1})}{\partial o_j^m})) f^{'}(z_j^m) o_i^{m-1} \\
&= (\sum_{l=0}^{D^{m+1}} (\frac{\partial Err}{\partial o_l^{m+1}} f'(z_l^{m+1})\frac{\partial(\sum_{k=0}^{D^{m+1}} w_{l,k}^{m+1} o_k^m)}{\partial o_j^m})) f^{'}(z_j^m) o_i^{m-1} \\
&= (\sum_{l=0}^{D^{m+1}} (\frac{\partial Err}{\partial o_l^{m+1}} f'(z_l^{m+1}) w_{l,j}^{m+1})) f^{'}(z_j^m) o_i^{m-1}
\end{aligned}
$$

其中，

$$\frac{\partial Err}{\partial o_l^{m+1}} f'(z_l^{m+1}) w_{l,j}^{m+1} = a_l^{m+1}$$

a_l^{m+1} 是第 $m+1$ 层第 l 个神经元的附加变量，是已知量。

$$\begin{aligned}\frac{\partial Err}{\partial w_{j,i}^m} &= (\sum_{l=0}^{D^{m+1}}(a_l^{m+1}w_{l,j}^{m+1}))f^{'}(z_j^m)o_i^{m-1} \\ &= (f^{'}(z_j^m)\sum_{l=0}^{D^{m+1}}(a_l^{m+1}w_{l,j}^{m+1}))o_i^{m-1}\end{aligned}$$

令

$$a_j^m = (f^{'}(z_j^m)\sum_{l=0}^{D^{m+1}}(a_l^{m+1}w_{l,j}^{m+1}))$$

则 a_j^m 是第 m 层第 j 个神经元的附加变量。

因此：

$$\frac{\partial Err}{\partial w_{j,i}^m} = a_j^m o_i^{m-1}$$

b^m 的推导也是一样的，不列出来了，体力活。

至此，我们完成了通用 BP 算法的推导，验证了“路径”规则[1]的合理性，验证了“附加变量”给 BP 算法的具体实现带来的便捷。

[1]学术上有个类似的概念叫“计算图”

第六章　从 BP 算法到深度学习

1943 年，Warren S. McCulloch 和 Walter Pitts 在《The Bulletin of Mathematical Biophysics》发表论文《A Logical Calculus of the Ideas Immanent in Nervous Activity》，发明了第一种人工神经网络 MCP 模型。“MCP” 模型命名并没有在论文里明确给出，应当是两位作者名字 McCulloch 和 Pitts 的组合。

1957 年，Frank Rosenblatt 在《New York Times》上发表文章《Electronic Brain Teaches Itself》，发明感知机。

1969 年，Marvin Minsky 和 Seymour Papery 在《Perceptrons》一书分析了单层感知机在计算能力上的局限性，证明单层感知机不能解决简单的异或等线性不可分问题，让神经网络研究进入低潮。

1974 年，Paul J. Werbos 在博士论文《Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences》发明了 BP 算法。

1986 年，David E. Rumelhart、Geoffrey E. Hinton、Ronald J. Williams 在《Nature》发表论文《Learning Representations by Back-propagating Errors》，再次发明 BP 算法，优化多层感知机，解决了异或之类的非线性分类问题。

1989 年，Yann LeCun 等发表论文《Gradient-based Learning Applied to Document Recognition》，发明卷积神经网络 LeNet，用于数字识别，取得了较好的结果。

1991 年，Sepp Hochreiter 在毕业论文《Untersuchungen zu Dynamischen Neuronalen Netzen》（德语）指出 BP 算法的梯度消失问题。

2006 年，Geoffrey Hinton 和 Ruslan Salakhutdinov 在《Science》发表论文《Reducing the Dimensionality of Data with Neural Networks》，提出解决梯度消失的方法：无监督预训练对权值进行初始化（自动编码器），然后做有监督训练微调。在这篇论文首次提出“深度学习”概念。

2011 年，Xavier Glorot, Antoine Nordes, Yoshua Bengio 发表论文《Deep Sparse Recifier Neural Networks》提出 ReLU 激活函数，很好地抑制梯度消

失问题。

2012 年，深度学习神经网络 AlexNet 在 ImageNet 图像识别比赛获得冠军，效果显著领先第二名，论文 Alex Krizhevsky、Ilya Sutskever、Geoffrey E. Hinton《Imagenet Classification with Deep Convolutional Neural Networks》。取得优势的原因：首次采用 ReLU 激活函数，有监督学习，Dropout 减小过拟合，LRN 层增强泛化能力，GPU 加速。

2015 年，Kaiming He、Xiangyu Zhang、Shaoqing Ren、Jian Sun 发表论文《Deep Residual Learning for Image Recognition》，发明 ResNet，将神经网络训练层数提升到前所未有的高度。

2016 年，DeepMind 公司发明基于 ResNet 思想的 AlphaGo 与围棋世界冠军、职业九段棋手李世石进行围棋人机大战，以 4 比 1 的总比分获胜。

2017 年，AlphaGo 升级版 AlphaGo Zero 用“从零开始”、“左右互博”的学习模式，以 100:0 的比分打败 AlphaGo。