

# 深度学习入门笔记

## 反向传播算法

梯度向量的含义：

$-\nabla C(\dots) =$   
All weights and biases

Direction in 13,002

梯度向量每一项的大小是在告诉大家  
The magnitude of each component here is telling you

$C(w_0, w_1, \dots, w_{13,001}) = 2.85$

代价函数对于每个参数有多敏感  
how sensitive the cost function is to each weight and bias.

对于单个训练数据：

**2** You can only adjust weights and biases

784

0.5 0 ○ 0  
 0.8 1 ○ 1  
 0.2 2 ● 2  
 1.0 3 ○ 3  
 0.4 4 ○ 4  
 ...  
 0.6 5 ○ 5  
 1.0 6 ○ 6  
 0.0 7 ○ 7  
 0.2 8 ○ 8

**我们并不能直接改动这些激活值 只能改变权重和偏置值**

Now we can't directly change those activations, we only have influence on the weights and biases.

下一层的activation受到三个方面的值的影响：

- 与上一层连线的权重 weight
- 上一层的激活值 activation
- 当前节点的偏置值 bias

**2**

$$a_2 = \sigma(w_0 a_0 + w_1 a_1 + \dots + w_{n-1} a_{n-1} + b)$$

0.2 ↑ 2

**所以要增加这个激活值 我们有三条大路可走**

So there are three different avenues that can team up together to help increase that activation.

当调整权重 weight 的时候：

+关注

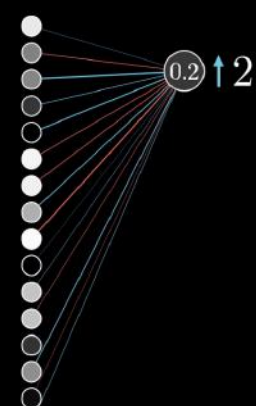
2

$$\textcircled{0.2} = \sigma(w_0 a_0 + w_1 a_1 + \dots + w_{n-1} a_{n-1} + b)$$

Increase  $b$

Increase  $w_i$

Change  $a_i$



**各个权重它们的影响力各不相同**  
notice how the weights actually have differing levels of influence:

▶

**连接前一层最亮的神经元的权重 影响力也最大**  
the connections with the brightest neurons from the preceding layer have the biggest effect,

▶

**因为这些权重会与大的激活值相乘**  
since those weights are multiplied by larger activation values.

▶

当调整上一层的激活值 activation 的时候：

**就是改变前一年的激活值**  
is by changing all the activations in the previous layer,

▶

**更具体地说 如果所有正权重连接的神经元更亮**  
namely, if everything connected to that digit 2 neuron with a positive weight got brighter,

▶

**所有负权重连接的神经元更暗的话**  
and if everything connected with a negative weight got dimmer,

▶

**那么数字2的神经元就会更强烈地激发**  
then that digit 2 neuron would become more active.

▶

(ps: 但是记住, 我们只能改变 weight 和 bias)

将最后一层的所有的神经元各自提出的变化量相加：

**2** Propagate backwards

Increase  $b$

Increase  $w_i$   
in proportion to  $a_i$

Change  $a_i$   
in proportion to  $w_i$

我们把所有期待的变化加起来  
By adding together all these desired effects,

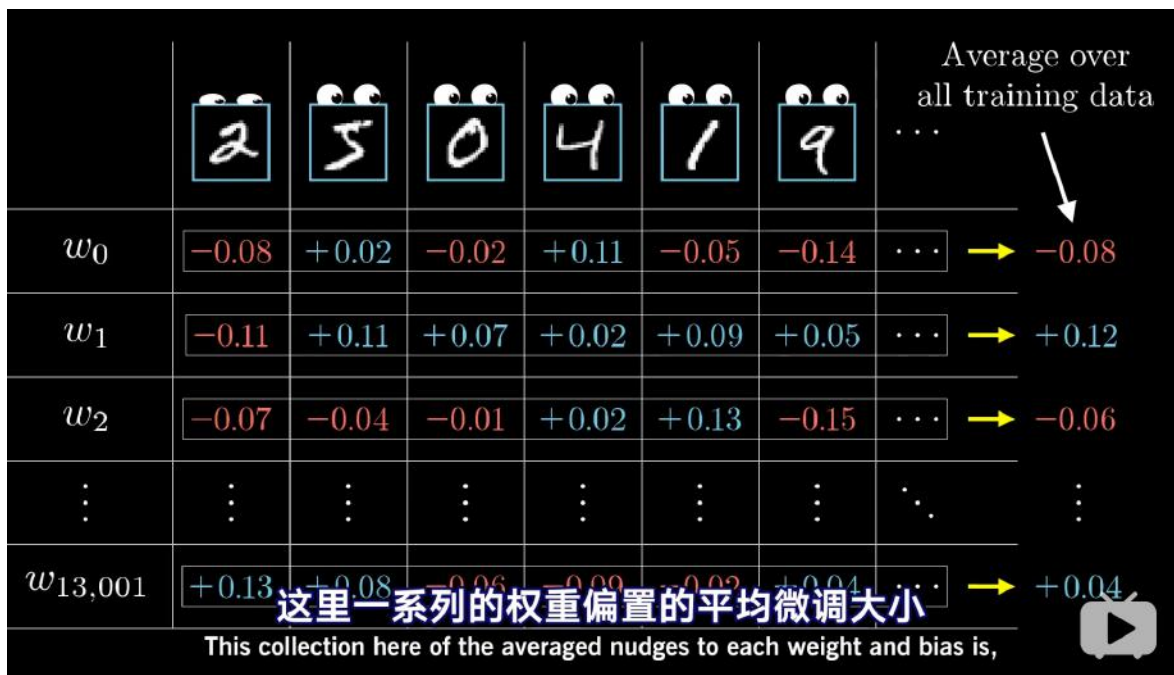
就得到了一串对倒数第二层改动的变化量  
you basically get a list of nudges that you want to happen to the second-to-last layer.

**2** Propagate backwards

我们就可以重复这个过程  
you can recursively apply the same process

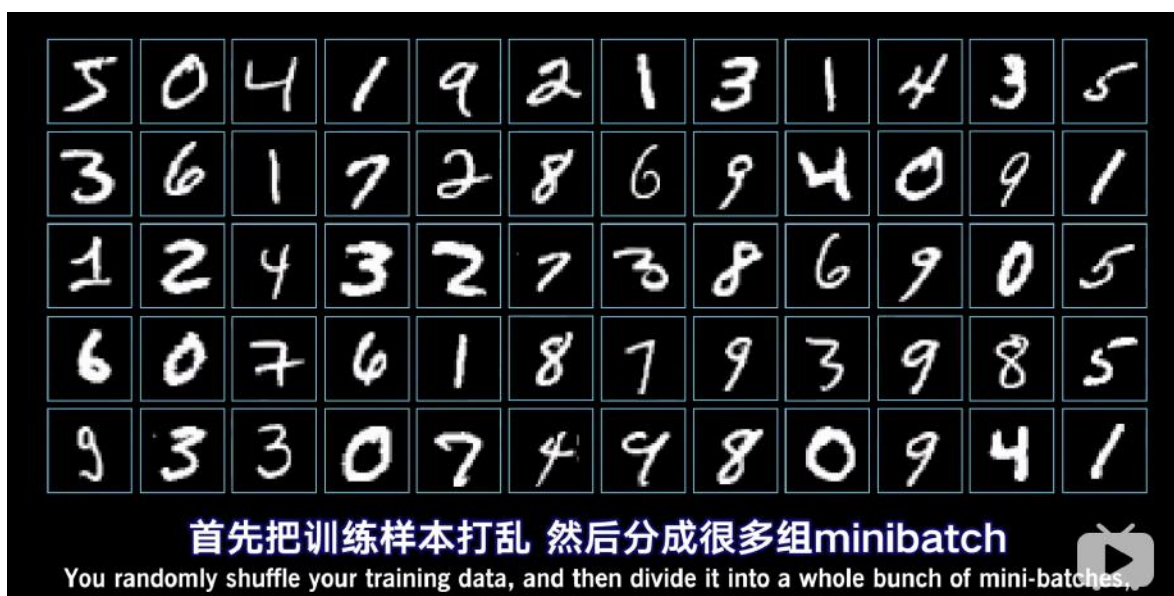
这其实就是反向传播算法(back propagation-BP)的主要思想，到此只考虑了一个训练样本“2”的反向传播对权重和偏置的影响

当我们考虑多个样本时：



$w_{13,001}$  不严格地说 就是上期视频提到的代价函数的负梯度<sup>+0.04</sup>  
loosely speaking, the negative gradient of the cost function referenced in the last video,

实际的训练过程中，为了减少计算量。通常不会在每一次的梯度下降之前将所有的样本都计算一遍，而是将样本集划分成一个个字集 minibatch



每个minibatch就当包含100个训练样本好了  
let's say, each one having 100 training examples.

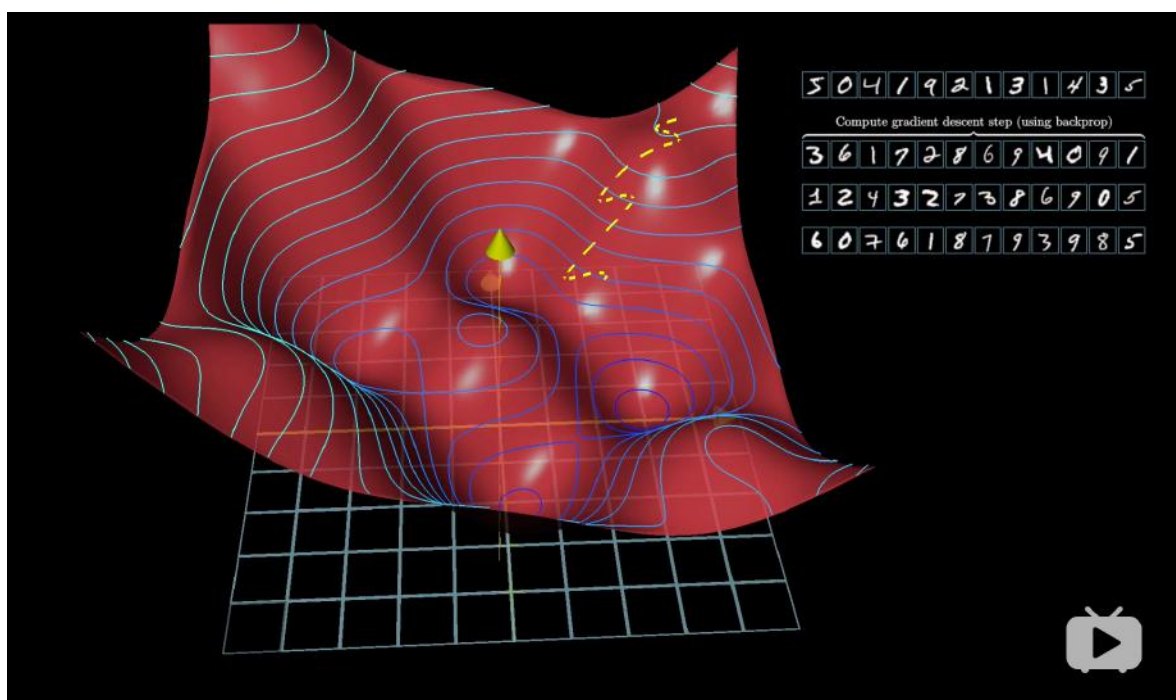
每次梯度下降都计算一个 mini-batch 得到梯度下降向量，这样计算量会下降不少

然而 每个minibatch都会给你一个不错的近似  
But each mini batch does give you a pretty good approximation,

计算一个minibatch来作为梯度下降的一步  
and compute each step with respect to a mini-batch.

计算每个minibatch的梯度 调整参数 不断循环  
Repeatedly going through all of the mini batches and making these adjustments,

最终你就会收敛到代价函数的一个局部最小值上  
you will converge towards a local minimum of the cost function,



这样梯度下降的过程就是曲折的，而非最快的。但最终还是能到达目标函数的局部最优值。这叫做“随机梯度下降(stochastic gradient descent)”

总结By RandomYane <https://weibo.com/3229623314>