



# 类神经网络训练不起来怎么办

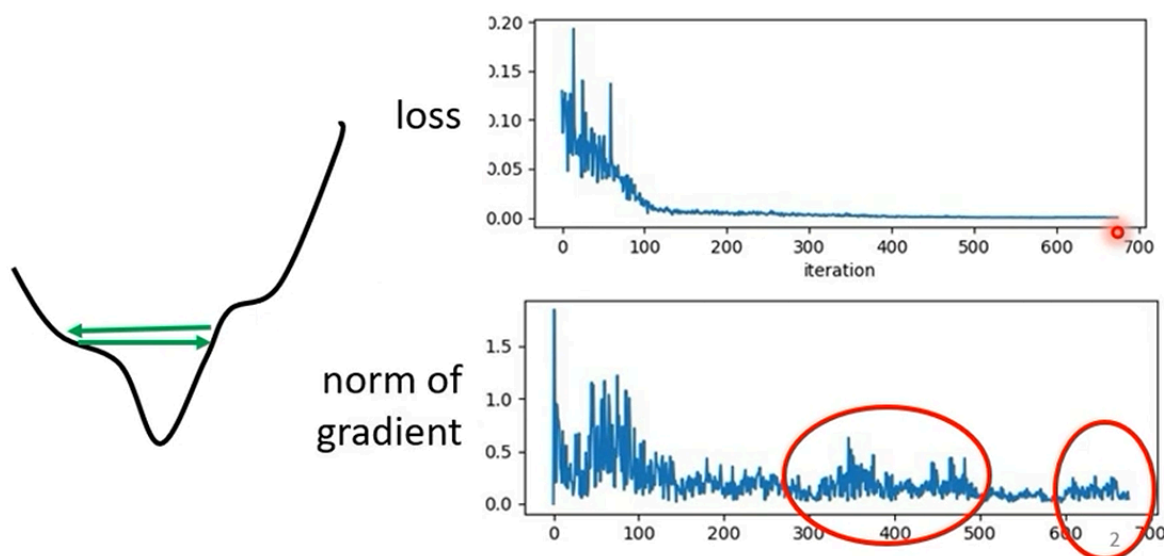
## (三) 自动调整学习率

### (Learning Rate)

loss不再下降不一定是卡在critical Point，有可能是单纯的loss不会下降。

Training stuck  $\neq$  Small Gradient

- People believe training stuck because the parameters are around a critical point ...



可以看到，其实还是有梯度的，只是步长太大，导致无法更新loss。

所以我们是需要在梯度较小时（平缓）学习率较大，而在梯度较大时（陡峭）学习率较小。

下面是Apagrad方法：

Root Mean Square  $\theta_i^{t+1} \leftarrow \theta_i^t - \boxed{\frac{\eta}{\sigma_i^t}} g_i^t$

$$\theta_i^1 \leftarrow \theta_i^0 - \frac{\eta}{\sigma_i^0} g_i^0 \quad \sigma_i^0 = \sqrt{(g_i^0)^2} = |g_i^0|$$

$$\theta_i^2 \leftarrow \theta_i^1 - \frac{\eta}{\sigma_i^1} g_i^1 \quad \sigma_i^1 = \sqrt{\frac{1}{2} [(g_i^0)^2 + (g_i^1)^2]}$$

$$\theta_i^3 \leftarrow \theta_i^2 - \frac{\eta}{\sigma_i^2} g_i^2 \quad \sigma_i^2 = \sqrt{\frac{1}{3} [(g_i^0)^2 + (g_i^1)^2 + (g_i^2)^2]}$$

$$\vdots$$

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t \quad \sigma_i^t = \sqrt{\frac{1}{t+1} \sum_{i=0}^t (g_i^t)^2}$$

6

这样就可以根据不同的梯度，自适应学习率的大小。

下面是RMSProp方法

## RMSProp

$$\theta_i^{t+1} \leftarrow \theta_i^t - \boxed{\frac{\eta}{\sigma_i^t}} g_i^t$$

$$\theta_i^1 \leftarrow \theta_i^0 - \frac{\eta}{\sigma_i^0} g_i^0 \quad \sigma_i^0 = \sqrt{(g_i^0)^2} \quad 0 < \alpha < 1$$

$$\theta_i^2 \leftarrow \theta_i^1 - \frac{\eta}{\sigma_i^1} g_i^1 \quad \sigma_i^1 = \sqrt{\alpha(\sigma_i^0)^2 + (1 - \alpha)(g_i^1)^2}$$

$$\theta_i^3 \leftarrow \theta_i^2 - \frac{\eta}{\sigma_i^2} g_i^2 \quad \sigma_i^2 = \sqrt{\alpha(\sigma_i^1)^2 + (1 - \alpha)(g_i^2)^2}$$

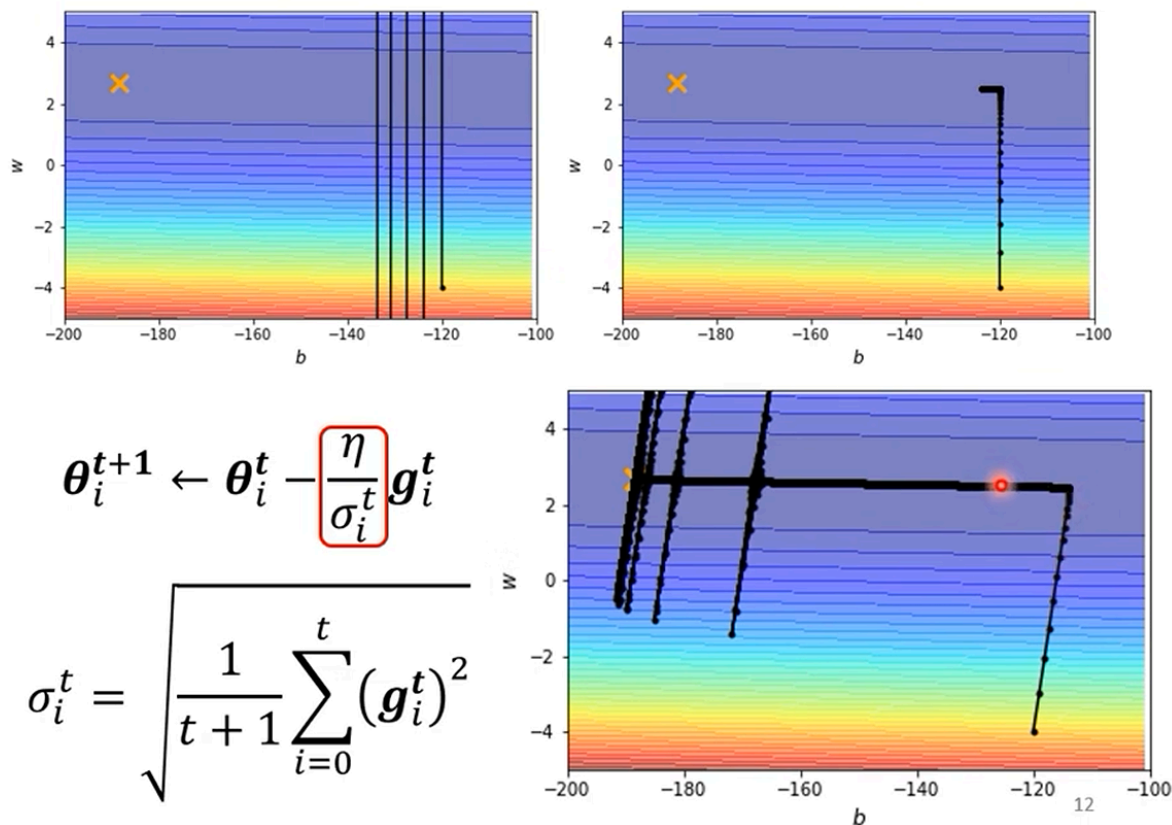
⋮

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t \quad \sigma_i^t = \sqrt{\alpha(\sigma_i^{t-1})^2 + (1 - \alpha)(g_i^t)^2}$$

9

现在常用的就是Adam，就是RMSProp+Momentum。

## Without Adaptive Learning Rate

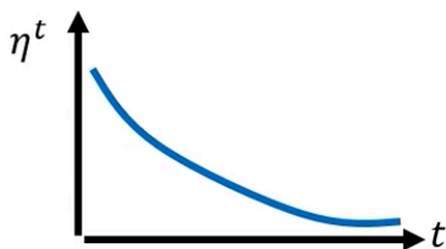


可以看到下面那个图就是加入了Adagrad的曲线，相对于没有自适应的Learning Rate，这个明显更靠近目标点，但是其中的爆炸，是因为在y方向先前积累了很多小的 $\sigma$ ，达到一定程度就爆炸，但是爆炸跑远之后又会使梯度变大，从而学习率下降，恢复到原来状态。

当然上面的情况也是可以解决的。可以用Learning Rate decay

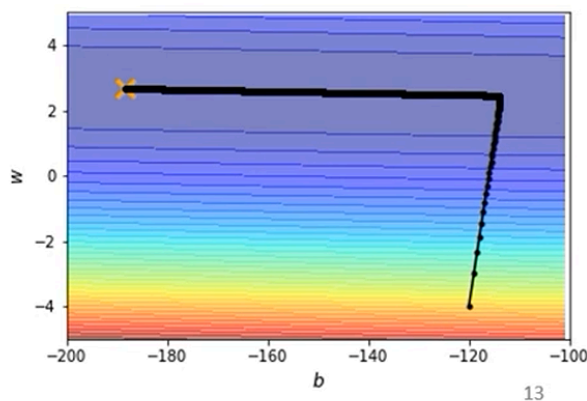
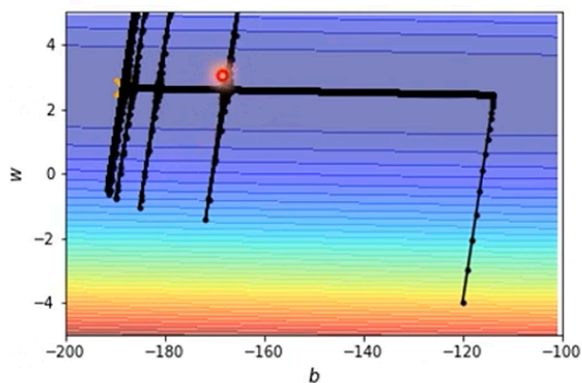
## Learning Rate Scheduling

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} g_i^t$$



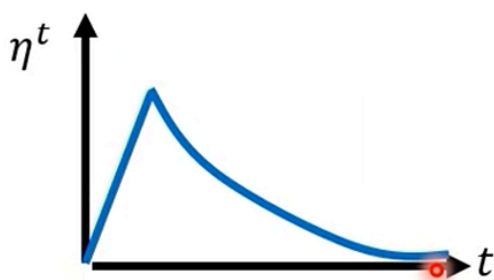
### Learning Rate Decay

As the training goes, we are closer to the destination, so we reduce the learning rate.



可以看到，我们随着训练次数的增加，不断地靠近目标点，所以我们不断地减少学习率。

除此以外我们也可以用warm up，先变大后变小。



### Warm Up

Increase and then decrease?

一般训练BERT时会用到warm up。

关于优化的改良版：

# Summary of Optimization

## (Vanilla) Gradient Descent

$$\theta_i^{t+1} \leftarrow \theta_i^t - \eta g_i^t$$

## Various Improvements

The diagram illustrates the improved gradient descent formula:  $\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} m_i^t$ . Annotations include: 

- $\eta^t$ : Learning rate scheduling
- $m_i^t$ : Momentum: weighted sum of the previous gradients
- $\sigma_i^t$ : root mean square of the gradients

 Two blue boxes provide additional context: "Consider direction" (pointing to  $m_i^t$ ) and "only magnitude" (pointing to  $\sigma_i^t$ ).

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} m_i^t$$

Learning rate scheduling

Momentum: weighted sum of the previous gradients

Consider direction

root mean square of the gradients

only magnitude

17