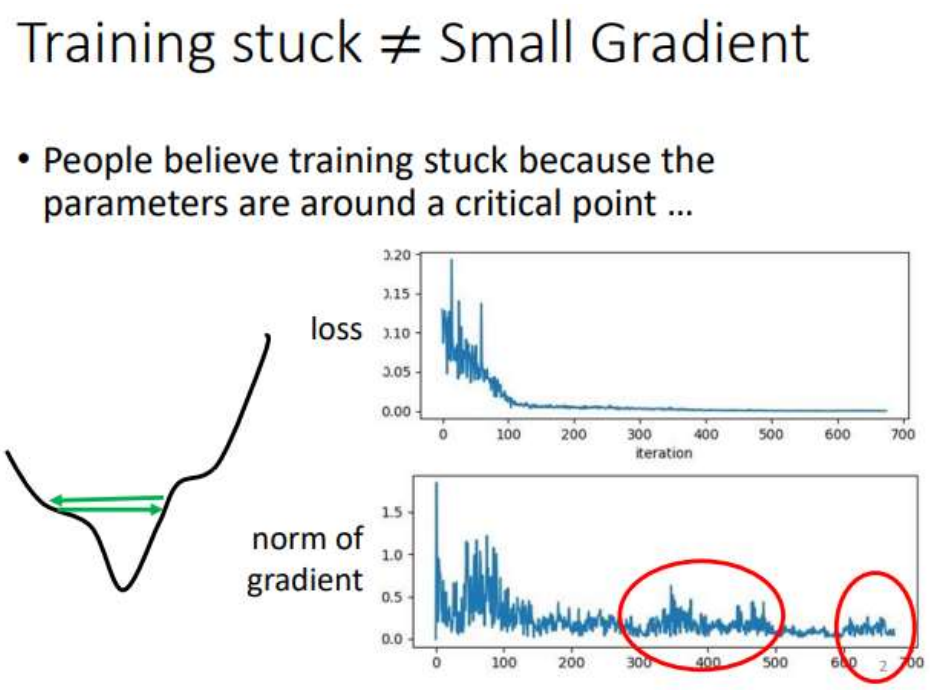
類神經網路訓練不起來怎麼辦 (三)：自動調整學習速率 (Learning Rate)

Create at 2022/06/08

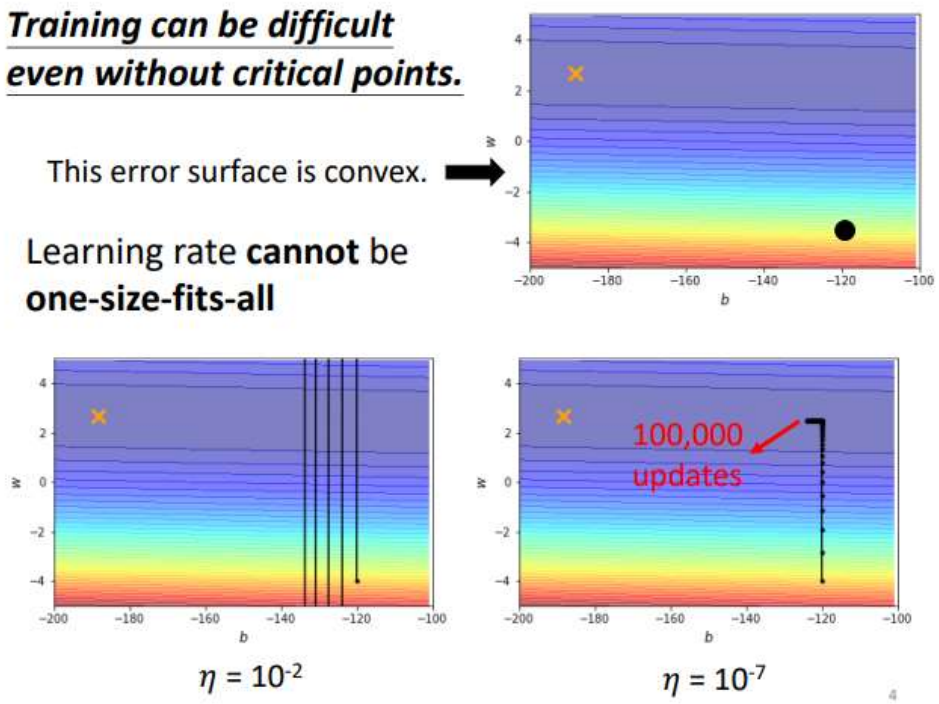
- 上課資源：
  1. 類神經網路訓練不起來怎麼辦 (三)：自動調整學習速率 (Learning Rate) (https://www.youtube.com/watch?v=HYUXEeh3kwY)

Adaptive Learning Rate 技術

要給每一個參數不同的 learning rate



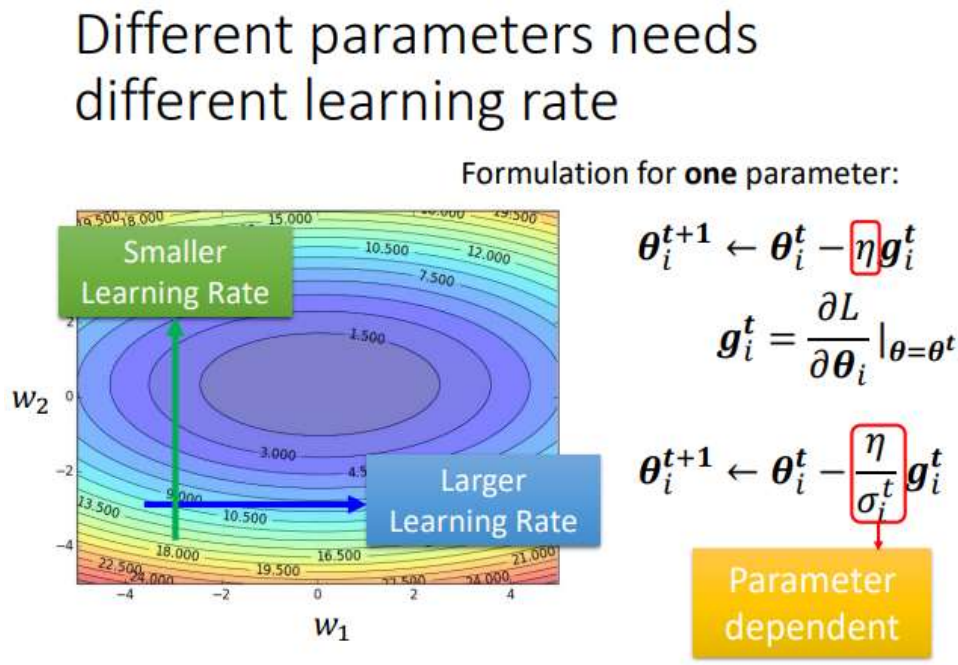
- critical point 不一定是在訓練的時候會遇到的最大障礙
- 在訓練 network 時往往會把 loss 記錄下來
- 隨著參數不斷地 update ，loss 會越來越小，最後 loss 不再下降
- 當走到 critical point 時，代表 gradient 非常小，但有確認過當 loss 不再下降時，gradient 真的很小嗎？
- 下圖是 gradient 向量的長度，隨著參數更新的時候的變化
- 在這個例子裡面，當 loss 不再下降時，gradient 沒有真的變很小
- 左圖是 error surface，現在的 gradient 在 error surface 山谷的兩個谷壁之間不斷的來回震盪，此時 loss 不再下降，但並不是真的卡到 critical point、saddle point、local minima，它的 gradient 仍然很大，只是 loss 不見得再減小



- learning rate 決定了 update 參數時步伐有多大，設太大會造成沒辦法慢慢地滑到山谷
- 就算是一個 convex 的 error surface，用 gradient descend 也很難 train 好
- 之前我們的 gradient descend 裡面，所有的參數都是設同樣的 learning rate，但 learning rate 應該要為每一個參數客製化

客製化的 learning rate

不同的參數需要什麼樣的 learning rate



- 如果在某一個方向上 gradient 值很小 (很平坦)，那會希望 learning rate 調大一點
- 如果在某一個方向上非常陡峭，會希望 learning rate 調小一點

learning rate 如何自動的根據 gradient 的大小做調整？

- 要改 gradient descent 原來的式子
- η : learning rate
- θ_i^t : θ_i 在第 t 個 iteration 的值
- g_i^t : 在第 t 個 iteration，參數 i 算出來的 gradient，(在 $\theta = \theta^t$ 時，θ_i 對 $Loss$ 的微分)
- $\theta_i^t - \eta g_i^t$ 會更新 learning rate 到 θ_i^{t+1}

需要會客製化的 learning rate

- 不同的參數 i、不同的 t、給不同的 σ
- 把 η 改寫成 η/σ_i^t，就有 parameter dependent 的 learning rate

Parameter dependent 的 learning rate 有甚麼常見的計算方式？

Root Mean Square

Root Mean Square

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

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

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

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

$$\vdots$$

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

6

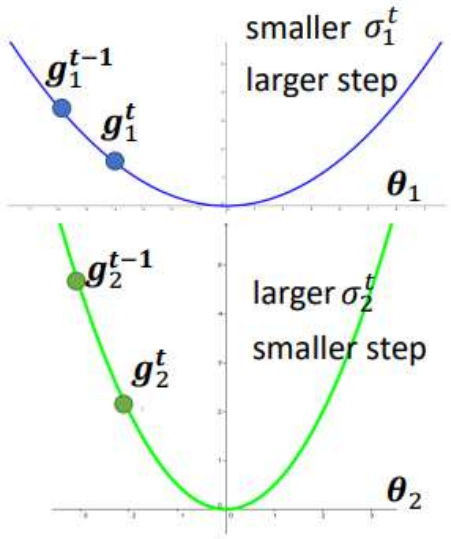
σ 是算出來的 g 平方和平均再開根號

Root Mean Square

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

$$\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}\left(g_i^t\right)^2}$$

Used in **Adagrad**

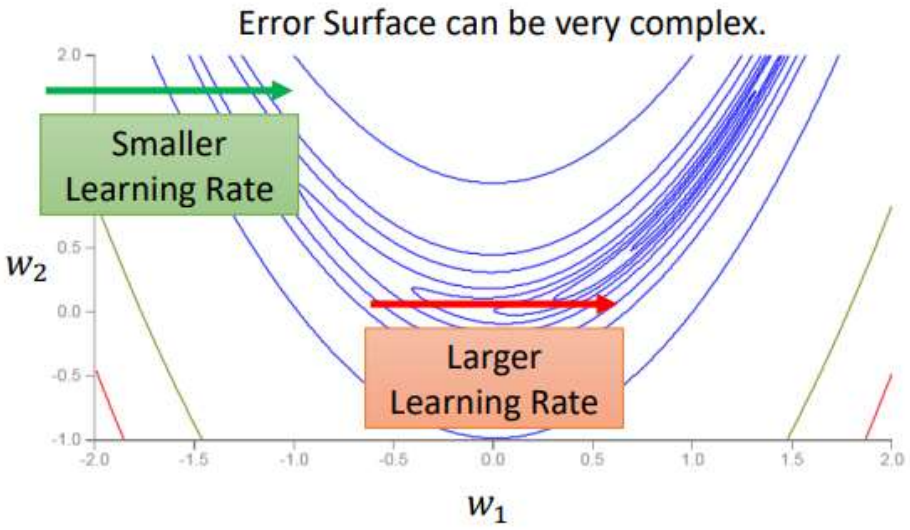


上面的方式被用在 Adagrade 的方法裡面

為甚麼這個方法可以做到 gradient 大時，learning rate 減小，gradient 小時，learning rate 放大呢？

- 上圖 θ_i^1 因為坡度小的關係，所以算出來的 gradient 值都比較小，所以算出來的 σ 就比較小，所以 η 就比較大
- 下圖 θ_i^2 因為坡度大的關係，$Loss$ 變化比較大，所以算出來的 gradient 值都比較大，所以算出來的 σ 就比較大，所以 η 就比較小

Learning rate adapts dynamically



- 就算是同一個參數，需要的 learning rate 也會隨時間改變

RMSProp

RMSProp

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t - \frac{\eta}{\sigma_i^t}\boldsymbol{g}_i^t$$

$$\boldsymbol{\theta}_i^1 \leftarrow \boldsymbol{\theta}_i^0 - \frac{\eta}{\sigma_i^0}\boldsymbol{g}_i^0 \qquad \sigma_i^0 = \sqrt{\left(\boldsymbol{g}_i^0\right)^2}$$

$$0 < \alpha < 1$$

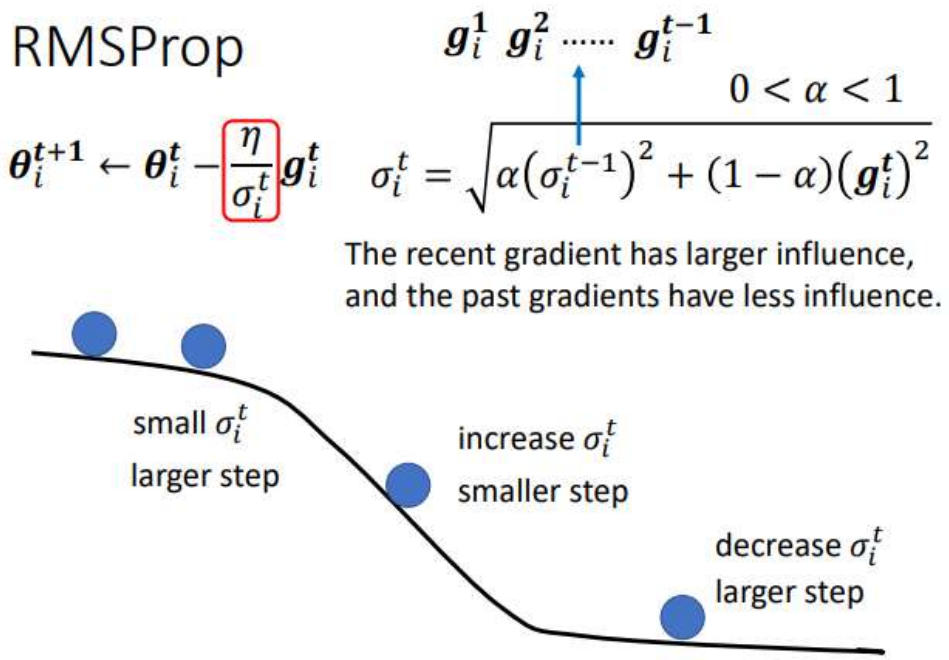
$$\boldsymbol{\theta}_i^2 \leftarrow \boldsymbol{\theta}_i^1 - \frac{\eta}{\sigma_i^1}\boldsymbol{g}_i^1 \qquad \sigma_i^1 = \sqrt{\alpha\left(\sigma_i^0\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^1\right)^2}$$

$$\boldsymbol{\theta}_i^3 \leftarrow \boldsymbol{\theta}_i^2 - \frac{\eta}{\sigma_i^2}\boldsymbol{g}_i^2 \qquad \sigma_i^2 = \sqrt{\alpha\left(\sigma_i^1\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^2\right)^2}$$

$$\vdots$$

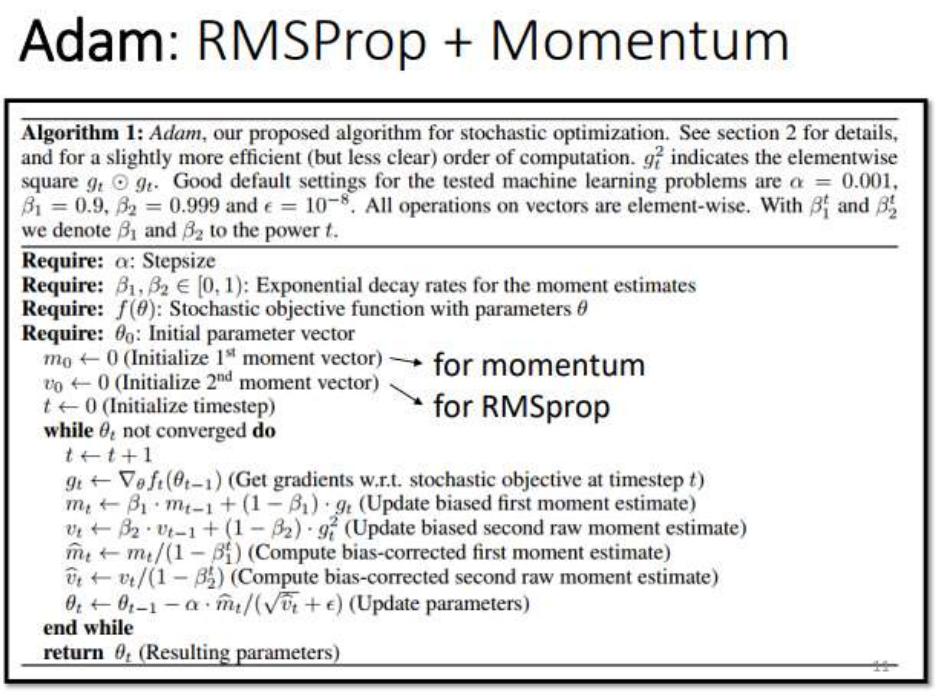
$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t - \frac{\eta}{\sigma_i^t}\boldsymbol{g}_i^t \qquad \sigma_i^t = \sqrt{\alpha\left(\sigma_i^{t-1}\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^t\right)^2}$$

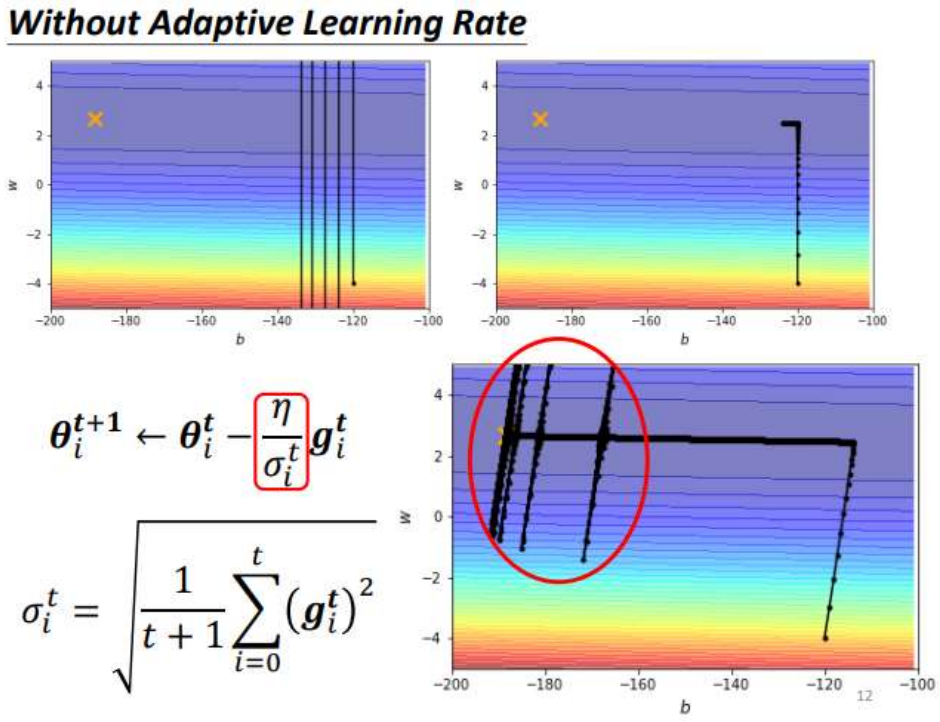
可以自己調整現在的 gradient 認為它有多重要



透過 α，可以動態決定 g_i^t 相較於之前存在 σ_i^{t-1} 裡面的 g_i^1 到 g_i^{t-1} 而言，重要性有多大

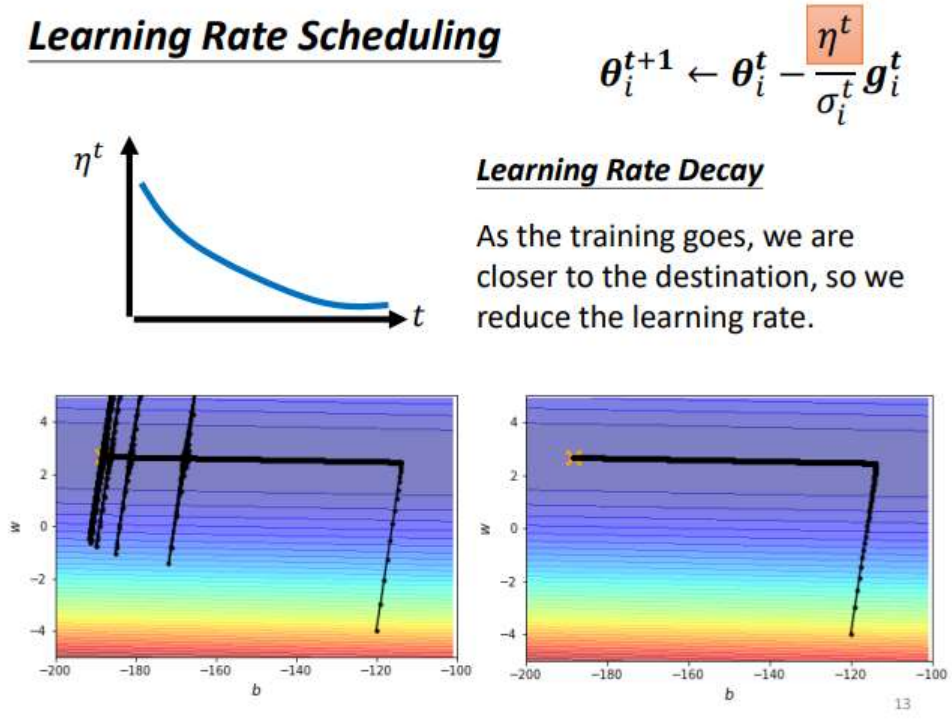
最常用的 optimization 策略：RMS Prop + Momentum



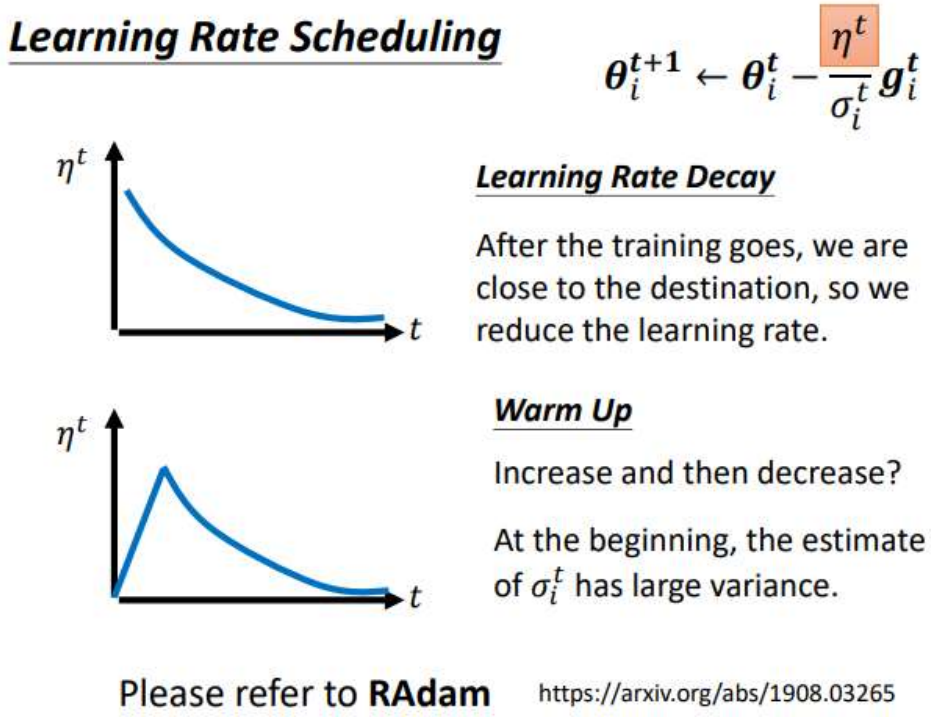


- 現在有 Adagrade 之後可以繼續走下去，走道非常接近終點的位置
- 左右方向的 gradient 很小，所以 learning rate 會自動調整變大，為甚麼後來會突然爆炸呢？
  - 累積了很多很小的 σ，累積到一個地步之後 step 就變很大，然後就爆走了，爆走之後走到 gradient 大的地方 σ 又慢慢變大，參數 update 的步伐又慢慢變小

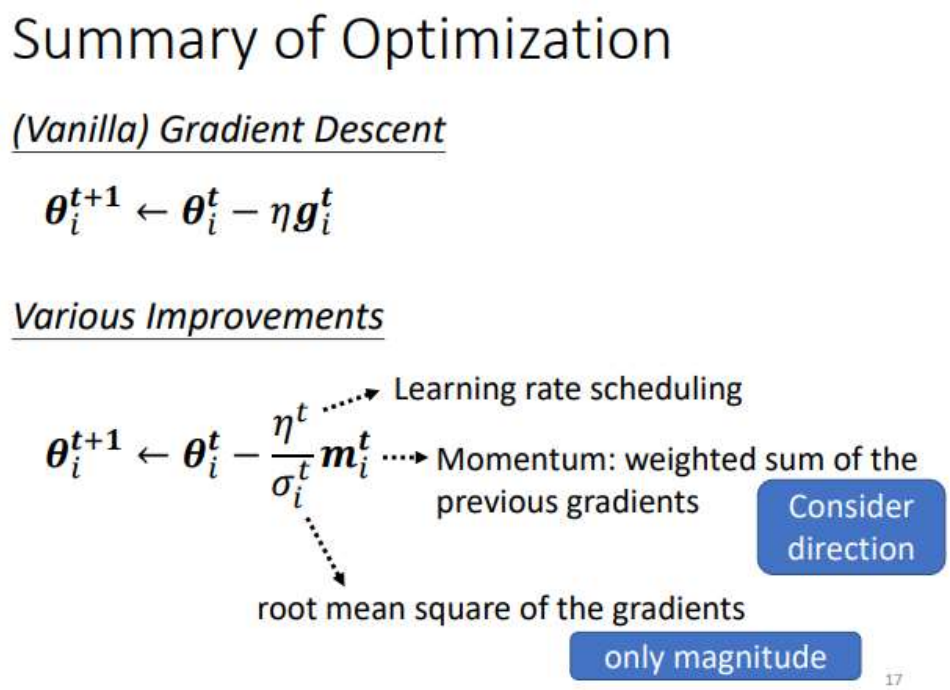
Learning Rate Scheduling 可以解決這個問題



- η 要是跟時間有關的，不要把它當一個常數
- 常見的策略稱為 Learning Rate Decay
  - 隨著時間不斷地前進、隨著參數不斷的 update
  - 讓 η 越來越小
  - 因為隨著參數不斷的 update，距離終點越來越近，所以把 learning rate 減小
  - 如果加上 learning rate decay，就可以順利地走到終點



- 另外一個常用的方式是 Warm Up
  - learning rate 要先變大再變小
  - 在訓練 BERT 的時候，會需要用到 Warm Up
- 為甚麼需要 Warm Up
  - 在用 Adam RMS Prop 或 Adagrad 的時候，會需要計算 σ
  - 先收集有關 σ 的統計數據，等統計比較精準之後，再讓 learning rate 慢慢提升



總結

- 補充教材：
  1. Optimization for Deep Learning (1/2) (https://www.youtube.com/watch?v=4pUmZ8hXlHM)
  2. Optimization for Deep Learning (2/2) (https://www.youtube.com/watch?v=e03YKGHXnL8)

課程網頁 (https://speech.ee.ntu.edu.tw/~hylee/ml/2022-spring.php)

tags: 2022 李宏毅_機器學習