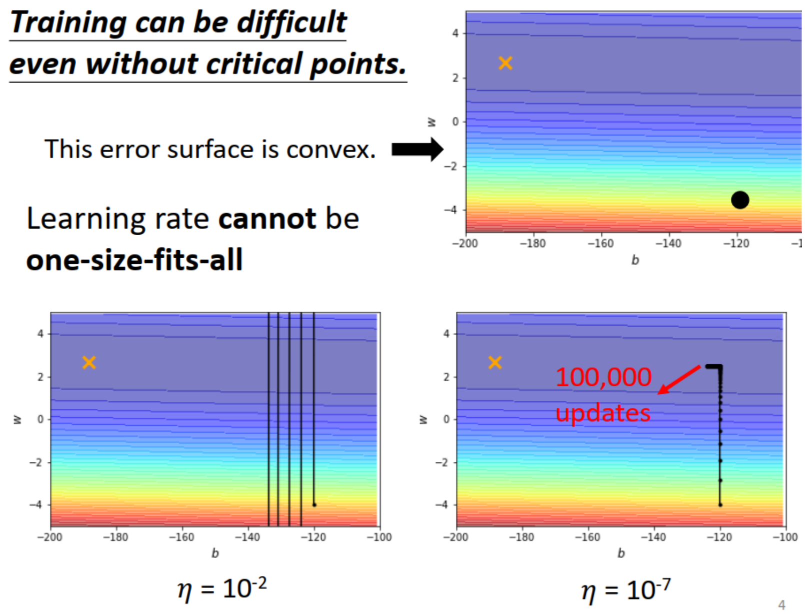
4.Adaptive Leaning Rate

這份筆記整理了關於類神經網路訓練中的最佳化方法，特別是適應性學習率（Adaptive Learning Rate）的概念與幾種常見的演算法。

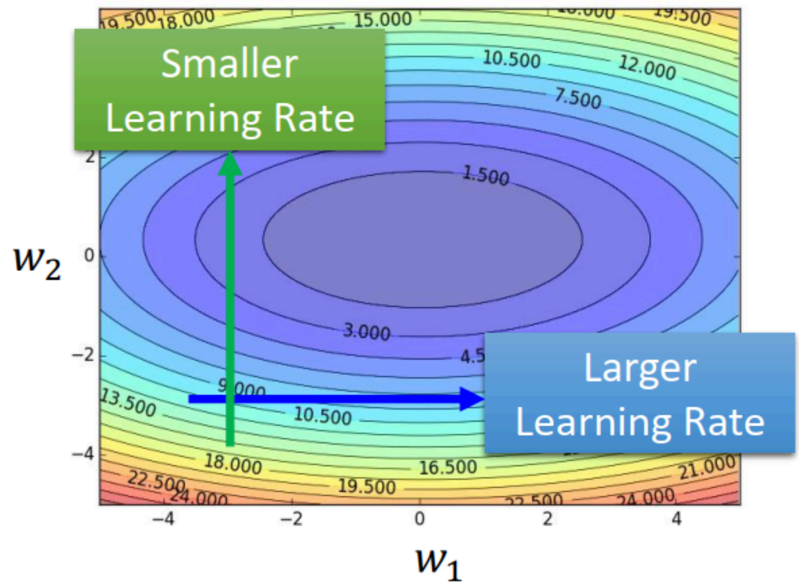
訓練卡住 ≠ 梯度過小



在訓練過程中，模型可能會陷入一個看起來訓練停滯的狀態（Training stuck）。人們常常誤以為這是因為參數處於臨界點（critical point），導致梯度（gradient）非常小。然而，PDF 中指出，這不一定是事實。

- 有時即使梯度範數（norm of gradient）不是特別小，訓練也可能卡住。
- 訓練困難可能發生在一個凸面（convex）的誤差曲面（error surface）上，即使沒有臨界點。

為什麼需要適應性學習率？



標準的梯度下降法（Vanilla Gradient Descent）使用一個固定的學習率 η。

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

其中：

- θ_i^t 是第 i個參數在第 t 次迭代時的值。
- g_i^t是第i個參數在第 t 次迭代時的梯度，即 $g_i^t = \frac{\partial L}{\partial \theta_i}\Big|_{\theta=\theta^t}$。

然而，對於不同的參數，可能需要不同的學習率。對於那些梯度變化劇烈的方向，需要較小的學習率；而對於梯度變化平緩的方向，則需要較大的學習率。

Adagrad

Adagrad（Adaptive Gradient Algorithm）是一種適應性學習率的方法，它為每個參數獨立調整學習率。

公式如下：

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

其中，σ_i^t 是過去所有梯度平方的根均方（Root Mean Square）：

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

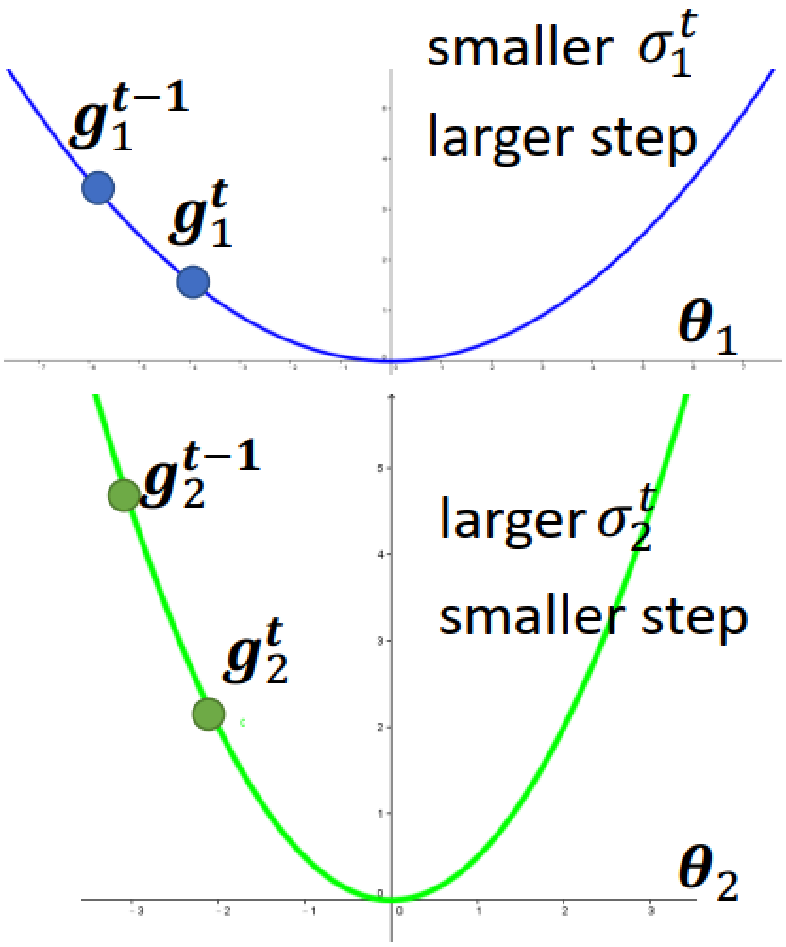
$$\theta_i^1 \leftarrow \theta_i^0 - \frac{\eta}{\sigma_i^0} g_i^0 \qquad \sigma_i^0 = \sqrt{\left(g_i^0\right)^2} = \left|g_i^0\right|$$

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

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

$$\vdots$$

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



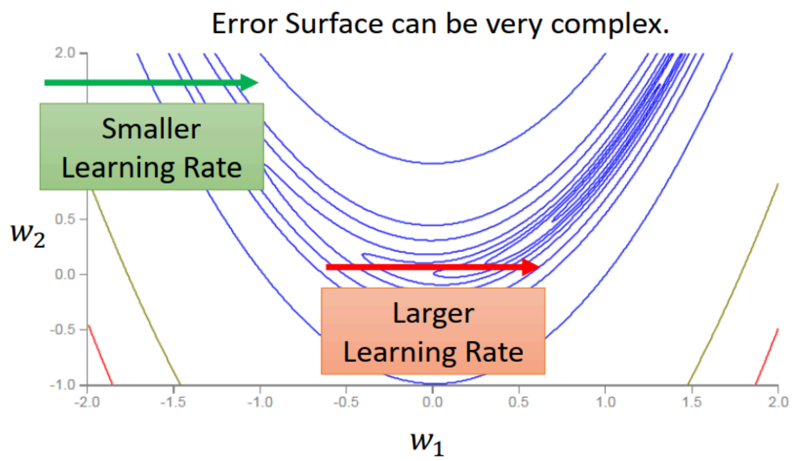
$$\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g_i^t)^2}$$

解釋：

- 如果某個參數的梯度在過去的迭代中一直很大，那麼 σ_i^t 就會變大，導致學習率 $\frac{\eta}{\sigma_i^t}$ 變小，步長（step）也變小。
- 反之，如果梯度一直很小，那麼 σ_i^t 就會變小，學習率變大，步長也變大。

- Adagrad 的缺點是，隨著訓練的進行，σ_i^t 會不斷累積並單調遞增，導致學習率變得極小，使得訓練提前停止。

Learning Rates Adaptive Dynamically (學習率的動態適應)



在複雜的誤差曲面（error surface）中，不同方向的曲率（curvature）可能差異很大。例如，在一個狹長的谷地（ravine）中，沿著長軸方向的梯度很小，而沿著短軸方向的梯度則很大。

如果使用固定的學習率，會面臨兩難：

- **學習率太小**：在平緩方向上進展太慢。
- **學習率太大**：在陡峭方向上會因為震盪（oscillation）而無法收斂。

動態適應性學習率方法（如 Adagrad, RMSProp, Adam）的優勢在於，它們會根據每個參數的歷史梯度來調整學習率。這使得演算法在長軸方向能採取較大的步長，快速前進；同時在短軸方向採取較小的步長，有效避免震盪，最終實現更快的收斂。

RMSProp

為了改進 Adagrad 的缺點，RMSProp（Root Mean Square Propagation）被提出。它使用指數加權移動平均（exponentially weighted moving average）來計算梯度的平方，使得近期梯度對學習率的影響更大。

公式如下：

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

RMSProp

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

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

$$0 < \alpha < 1$$

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

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

$$\vdots$$

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

其中，σ_i^t的計算方式為：

$$\sigma_i^t = \alpha(\sigma_i^{t-1})^2 + (1-\alpha)(g_i^t)^2$$

其中$0 < \alpha < 1$。

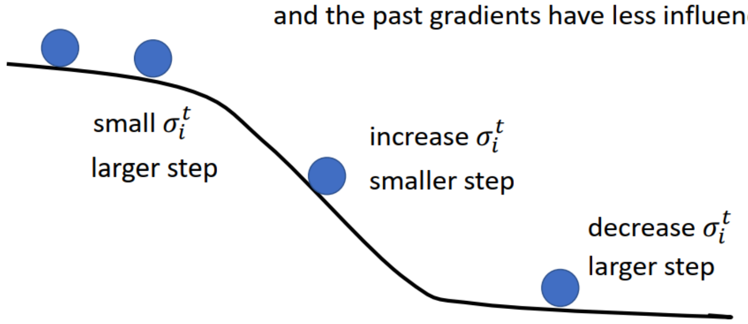
RMSProp

$$\boldsymbol{g}_i^1 \ \boldsymbol{g}_i^2 \ \cdots\cdots \ \boldsymbol{g}_i^{t-1}$$

$$0 < \alpha < 1$$

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

The recent gradient has larger influence, and the past gradients have less influence.



解釋：

- 這裡的 σ_i^t 不再是所有歷史梯度的簡單平均，而是根據衰減率 α 進行加權。
- 這使得梯度累積的影響被限制，學習率可以保持在一個更合理的範圍內，避免過早衰減至零。

Adam (RMSProp + Momentum)

Adam: RMSProp + Momentum

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
$v_0 \leftarrow 0$ (Initialize 2nd moment vector) → for RMSprop
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Adam（Adaptive Moment Estimation）是目前最受歡迎的最佳化演算法之一。它結合了 RMSProp 的適應性學習率和 Momentum（動量）的概念。

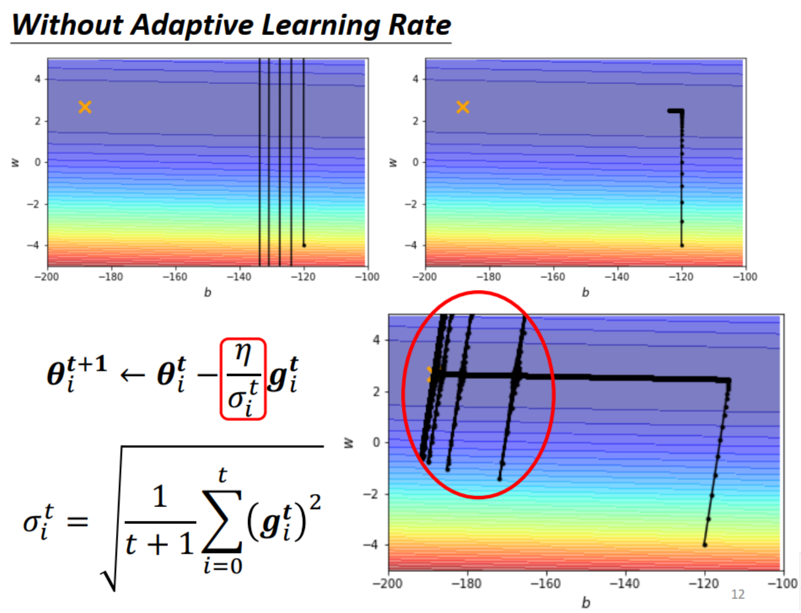
Adam 的核心思想是同時追蹤兩個指數加權移動平均：

1. **一階動量（First moment）：** 梯度的指數加權移動平均。
2. **二階動量（Second moment）：** 梯度平方的指數加權移動平均。

Adam 演算法步驟（PDF 提供的偽代碼簡化）：

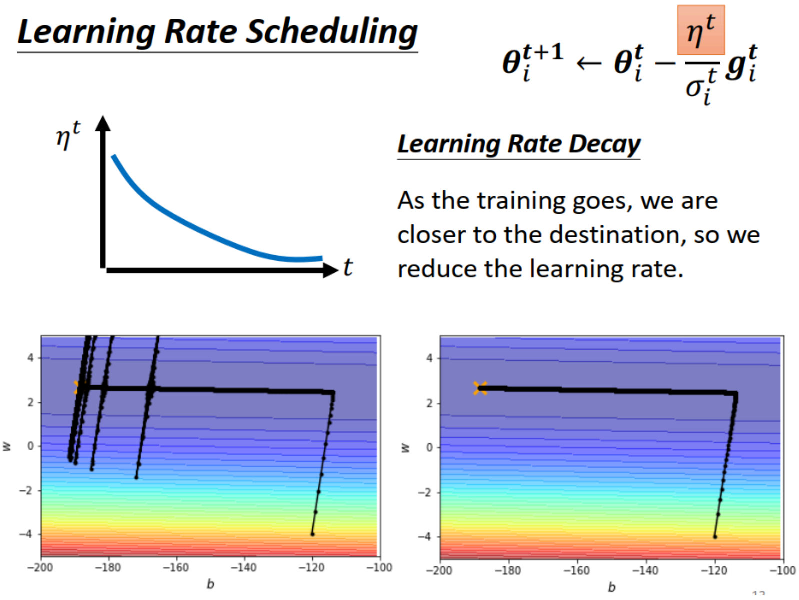
1. 初始化一階動量 mt 和二階動量 vt 為零。
2. 在每個訓練步驟 t：
   - 計算梯度 g_t。
   - 更新一階動量估計：$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t$
   - 更新二階動量估計：$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2$
   - 進行偏差校正（Bias Correction）：
     - $\hat{m}_t = m_t / (1 - \beta_1^t)$
     - $\hat{v}_t = v_t / (1 - \beta_2^t)$
   - 更新參數：$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$

學習率排程（Learning Rate Scheduling）



除了適應性學習率之外，還可以透過學習率排程來調整學習率。

學習率衰減（Learning Rate Decay）：



- 隨著訓練的進行，模型越來越接近目標，此時可以降低學習率。
- 這有助於模型在訓練後期更精確地收斂到最佳點，避免在最小值附近震盪。

學習率暖啟（Warm Up）：

> Pasted image 20250822183701.png

- 在訓練開始時，先緩慢增加學習率，然後再進行衰減。
- PDF 中提到，在訓練初期，σit 的估計可能會有較大的變異性，因此暖啟有助於穩定訓練過程。

總結

方法	更新公式	學習率調整方式
梯度下降	$\theta_i^{t+1} \leftarrow \theta_i^t - \eta g_i^t$	固定的學習率 η
Adagrad	$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$	根據歷史所有梯度的根均方來適應性調整
RMSProp	$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$	根據歷史近期的梯度來適應性調整
Adam	$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$	結合了動量和適應性學習率，是最全面的方法