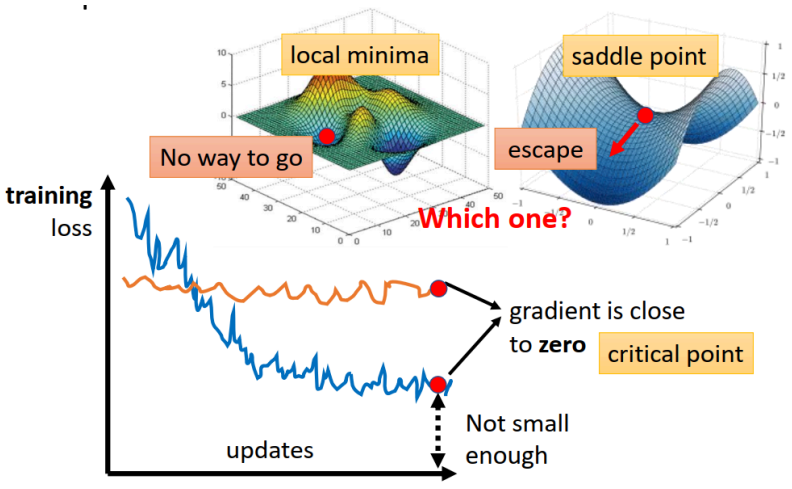
3.Optimization

這份筆記整理了關於機器學習最佳化過程中可能遇到的問題，特別是當梯度很小時的情況，並探討了局部最小值（local minima）、鞍點（saddle point）以及如何運用動量（Momentum）和批次大小（Batch size）來改善訓練過程。

梯度消失與最佳化失敗

在訓練神經網路時，我們希望透過梯度下降法（Gradient Descent）來尋找損失函數（Loss）的最小值。然而，當梯度值很小時，參數的更新會變得非常緩慢，導致訓練停滯。這種停滯可能發生在兩種不同的「臨界點」（critical point）：

- **局部最小值 (Local Minima)**：在這個點，損失函數的值比周圍所有點都低，梯度為零，且無法再透過梯度下降法降低損失。這就像困在一個盆地的底部，無法往外移動。
- **鞍點 (Saddle Point)**：這是一個點，在某些方向上損失函數是最小值，但在其他方向上是最大值。梯度在這個點也是零，但與局部最小值不同，我們可以沿著某些方向移動來逃離這個點。



泰勒級數與臨界點判斷

我們可以利用泰勒級數（Taylor Series）來近似損失函數 $L(\theta)$ 在 $\theta = \theta'$ 附近的行為。

泰勒級數近似式：

$L(\boldsymbol{\theta})$ around $\boldsymbol{\theta} = \boldsymbol{\theta}'$ can be approximated below

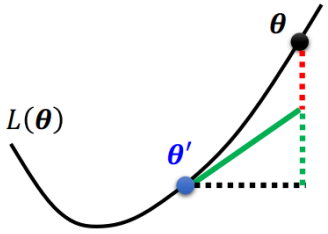
$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + (\boldsymbol{\theta} - \boldsymbol{\theta}')^T \boldsymbol{g} + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')$$

Gradient $\boldsymbol{g}$ is a ***vector***

$$\boldsymbol{g} = \nabla L(\boldsymbol{\theta}') \quad \boldsymbol{g}_i = \frac{\partial L(\boldsymbol{\theta}')}{\partial \boldsymbol{\theta}_i}$$

Hessian H is a ***matrix***

$$H_{ij} = \frac{\partial^2}{\partial \boldsymbol{\theta}_i \partial \boldsymbol{\theta}_j} L(\boldsymbol{\theta}')$$



$$L(\theta) \approx L(\theta\prime) + (\theta - \theta\prime)Tg + 21(\theta - \theta\prime)TH(\theta - \theta$$

其中：

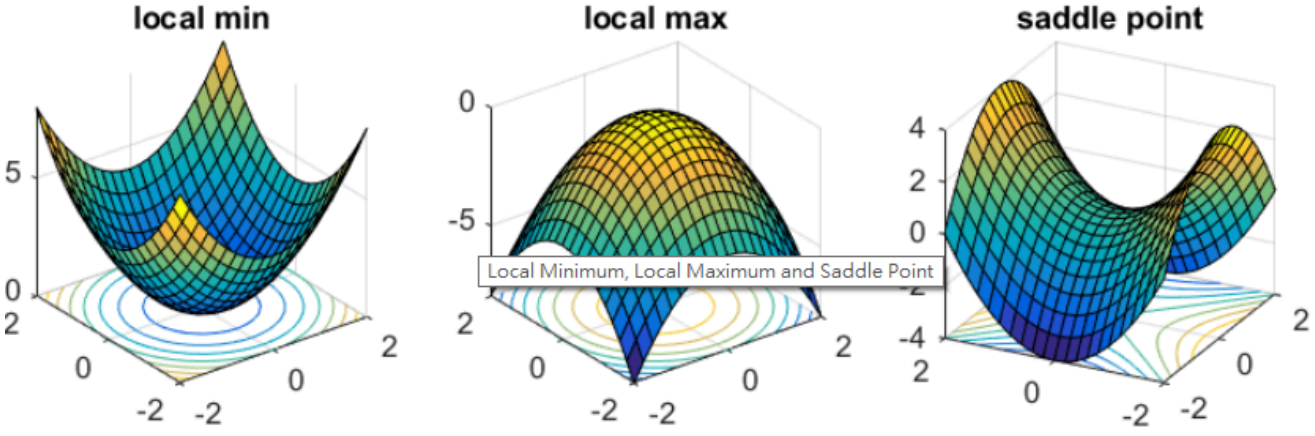
- g 是梯度向量（Gradient），定義為 $g = \nabla L(\theta')$，其分量為：$g_i = \frac{\partial L(\theta')}{\theta_i}$
- H 是海森矩陣（Hessian Matrix），其分量為：$Hij = \frac{\partial^2 i}{\partial \theta_i \partial_j} L(\theta')$

臨界點的性質：

在臨界點上，梯度 g=0，因此泰勒級數近似式簡化為：

$L(\theta) \approx L(\theta') + \frac{1}{2}(\theta - \theta')^T H\,(\theta - \theta')$
令 $\theta - \theta'$ 為v，則
$L(\theta) \approx L(\theta') + \frac{1}{2} v^T H v$

根據海森矩陣的性質，我們可以判斷臨界點的類型：



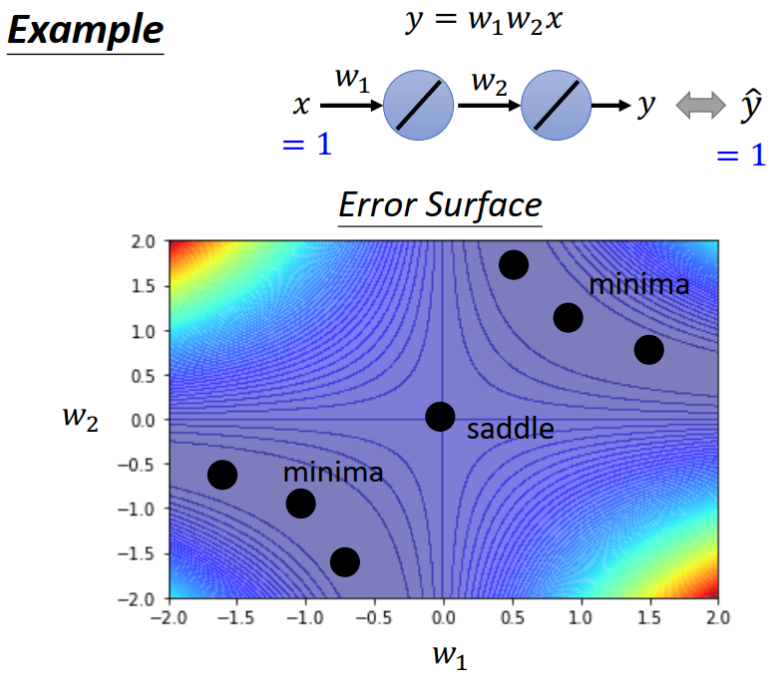
- **局部最小值 (Local Minima)**：對於所有向量 v，$v^T H v > 0$。此時海森矩陣是正定（positive definite），所有特徵值（eigenvalues）都為正。這表示在 θ′ 附近，損失函數值都大於 L(θ′)。

- **局部最大值 (Local Maxima)**：對於所有向量 v，$v^T Hv < 0$。此時海森矩陣是負定（negative definite），所有特徵值都為負。這表示在 θ′ 附近，損失函數值都小於 L(θ′)。
- **鞍點 (Saddle Point)**：$v^T Hv$有時大於零，有時小於零。這表示海森矩陣有正特徵值也有負特徵值。我們可以沿著負特徵值對應的特徵向量方向移動來降低損失。

逃離鞍點的方法

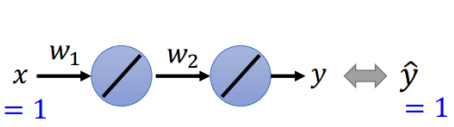
對於鞍點，我們可以沿著海森矩陣的負特徵值所對應的特徵向量方向來更新參數，以逃離鞍點並降低損失。

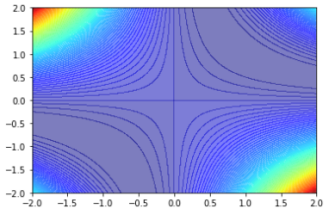
一個簡單的神經網路例子：



考慮一個簡單的線性神經網路，其輸出為 $y = w_1 w2 x$。假設我們希望輸出 y 逼近目標值 $\hat{y}$。
如果我們設定 x=1 且 $\hat{y}$，損失函數（Loss Function）可以定義為：

$$L = (\hat{y} - y)^2 = (1 - w1w2x)^2 = (1 - w_1 w_2)^2$$





$L = (\hat{y} - w_1 w_2 x)^2 = (1 - w_1 w_2)^2$

$$g = \begin{bmatrix} \frac{\partial L}{\partial w_1} = 2(1 - w_1 w_2)(-w_2) = 0 \\ \frac{\partial L}{\partial w_2} = 2(1 - w_1 w_2)(-w_1) = 0 \end{bmatrix}$$

Critical point: $w_1 = 0, w_2 = 0$

$H = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix}$ $\lambda_1 = 2, \lambda_2 = -2$

Saddle point

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial w_1^2} = 2(-w_2)(-w_2) = 0 & \frac{\partial^2 L}{\partial w_1 \partial w_2} = -2 + 4w_1 w_2 = -2 \\ \frac{\partial^2 L}{\partial w_2 \partial w_1} = -2 + 4w_1 w_2 = -2 & \frac{\partial^2 L}{\partial w_2^2} = 2(-w_1)(-w_1) = 0 \end{bmatrix}$$

當我們計算這個損失函數的梯度，並將其設為零時，可以找到臨界點：

$$\frac{\partial L}{\partial w_1} = 2(1 - w_1 w_2)(-w_2) = 0$$
$$\frac{\partial L}{\partial w2} = 2(1 - w_1 w_2)(-w_1) = 0$$

解這個方程組，我們發現一個臨界點在 w1=0,w2=0。

Don't afraid of saddle point?

At critical point: $L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')$ ($\boldsymbol{v}^T H \boldsymbol{v}$)

Sometimes $\boldsymbol{v}^T H \boldsymbol{v} > 0$, sometimes $\boldsymbol{v}^T H \boldsymbol{v} < 0$ ➡ Saddle point

H may tell us parameter update direction!

$\boldsymbol{u}$ is an eigen vector of H
λ is the eigen value of $\boldsymbol{u}$
$\lambda < 0$

➡ $\boldsymbol{u}^T H \boldsymbol{u} = \boldsymbol{u}^T (\lambda \boldsymbol{u}) = \lambda \|\boldsymbol{u}\|^2$
< 0 < 0

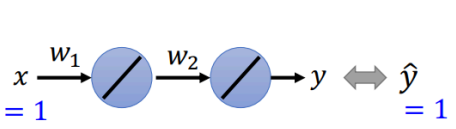
$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')$ ($\boldsymbol{u}$) ➡ $L(\boldsymbol{\theta}) < L(\boldsymbol{\theta}')$

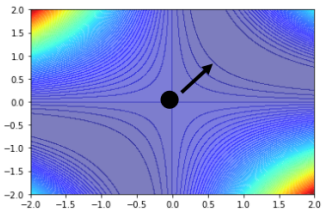
$\boldsymbol{\theta} - \boldsymbol{\theta}' = \boldsymbol{u}$ $\quad \boldsymbol{\theta} = \boldsymbol{\theta}' + \boldsymbol{u}$ $\quad$ Decrease L

在上述例子中，臨界點在 w1=0,w2=0。這個點的海森矩陣為：

$$H = \begin{bmatrix} 0, & -2 \\ -2, & 0 \end{bmatrix}$$

其特徵值為 λ1=2,λ2=−2。由於存在負特徵值，這個點是**鞍點**。





$$L = (\hat{y} - w_1 w_2 x)^2 = (1 - w_1 w_2)^2$$

$$\frac{\partial L}{\partial w_1} = 2(1 - w_1 w_2)(-w_2)$$

$$\frac{\partial L}{\partial w_2} = 2(1 - w_1 w_2)(-w_1)$$

Critical point: $w_1 = 0, w_2 = 0$

$$H = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix} \quad \lambda_1 = 2, \lambda_2 = -2$$

Saddle point

$\lambda_2 = -2$ Has eigenvector $\boldsymbol{u} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Update the parameter along the direction of $\boldsymbol{u}$

You can escape the saddle point and decrease the loss.

(this method is seldom used in practice)

我們可以沿著負特徵值 λ2=−2 所對應的特徵向量 u 方向來更新參數，以降低損失。對應的特徵向量為：

$$u = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

沿著這個方向更新參數，可以成功逃離鞍點並繼續降低損失。

步驟：

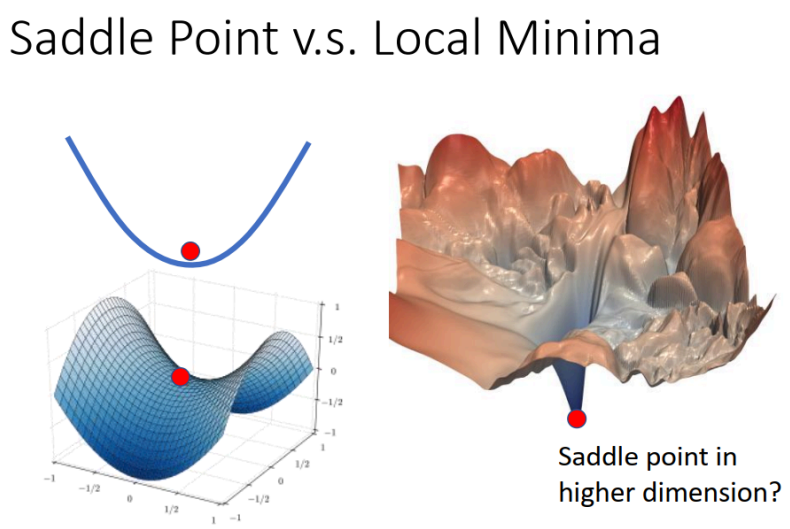
1. 計算海森矩陣 H。
2. 找到 H 的負特徵值 λ 和對應的特徵向量 u。
3. 沿著 u 的方向更新參數：

   θ=θ′+u

   這將導致損失函數值$L(\theta)$ 小於 $L(\theta')$。

雖然這個方法在理論上可行，但在實際應用中很少使用，因為計算海森矩陣的成本非常高。

局部最小值 vs. 鞍點：比較與區別



特性	局部最小值（Local Minimu…	鞍點（Saddle Point）
梯度 (Gradient)	梯度為零	梯度為零
海森矩陣 (Hessian)	正定 (Positive Definite)	非定 (Indefinite)
特徵值 (Eigenvalues)	所有特徵值皆為正數	同時有正數和負數的特徵值
損失函數行為	在所有方向上，損失函數值都比該點高	在某些方向上損失函數值更高，在另一些方向上更低
逃離難度	難以逃離，因為梯度為零，且所有方向都導致損失增加	可以沿著負特徵值對應的方向逃離，以降低損失
實務上的重要性	傳統上認為是最佳化中的主要障礙，但研究發現對於大型網路可能不常見	對於大型、高維度的神經網路來說，是更常見的訓練停滯原因

訓練技巧：批次與動量

除了上述方法，還有兩種常用的技巧可以幫助最佳化過程：批次訓練（Batch Training）和動量（Momentum）。

批次訓練 (Batch Training)

- **全批次 (Full Batch)**：每次更新參數時，使用所有訓練資料來計算梯度。
- **小批次 (Small Batch)**：將訓練資料分成多個小批次（batches），每次更新只使用其中一個小批次。

小批次訓練的優點：

- **速度更快**：雖然單次更新的計算時間可能更長，但一個 Epoch（看完所有資料）所花的時間更短。
- **更佳的最佳化**：由於每個小批次的梯度都帶有雜訊（noisy），這種隨機性有助於跳脫局部最小值或鞍點，找到更好的解。
- **更好的泛化能力**：研究顯示，小批次訓練傾向於找到「平坦」的最小值（Flat Minima），這代表模型對參數的微小變化不敏感，從而提供更好的泛化能力。

動量 (Momentum)

動量的概念類似於物理世界中的慣性。它不僅考慮當前的梯度，還考慮前一步的移動方向，從而加速訓練過程。

動量更新公式：

$$m_t\theta_t = \lambda m_{t-1} - \eta g_t = \theta_{t-1} + m_t$$

其中：

- m_t 是當前的「動量」。
- m_{t-1} 是前一步的動量。
- λ 是動量參數，控制前一步動量對當前步的影響。
- η 是學習率（learning rate）。
- g_t 是當前的梯度。

動量的優點：

- **加速收斂**：在平坦區域（梯度小）或梯度方向不一致時，動量可以幫助快速穿越，避免停滯。
- **逃離鞍點**：即使在鞍點梯度為零，如果之前的動量非零，參數依然會繼續移動，從而逃離鞍點。

最佳化算法

這兩項技術可以結合使用，發揮互補作用。

- **動量**可以使訓練過程像一個球在山谷中滾動，即使遇到小坡或平坦區域，也會因為慣性而繼續前進。這有助於穿越**鞍點**。
- **小批次訓練**則讓這個球的行進路線帶有一些隨機性，使其有機會跳出**局部最小值**的陷阱。

簡單來說，動量透過累積先前的移動方向來加速收斂，而小批次訓練則透過隨機性來探索更廣闊的參數空間。

這些只是其中一部分最佳化技術，其他常見的算法如 RMSprop 和 Adam 則是對動量的進一步改進，它們在訓練過程中能動態調整每個參數的學習率，達到更好的效果。

總結

- **臨界點的特性**：臨界點的梯度為零。這些點可能是鞍點（Saddle Points）或局部最小值（Local Minima）。
- **如何判斷臨界點**：可以使用海森矩陣來判斷。
- **逃離鞍點**：可以沿著海森矩陣特徵向量的方向移動來逃離鞍點。
- **局部最小值**：在實務上，局部最小值可能並不常見。
- **解決方案**：使用較小的批次大小和動量（Momentum）有助於逃離臨界點。