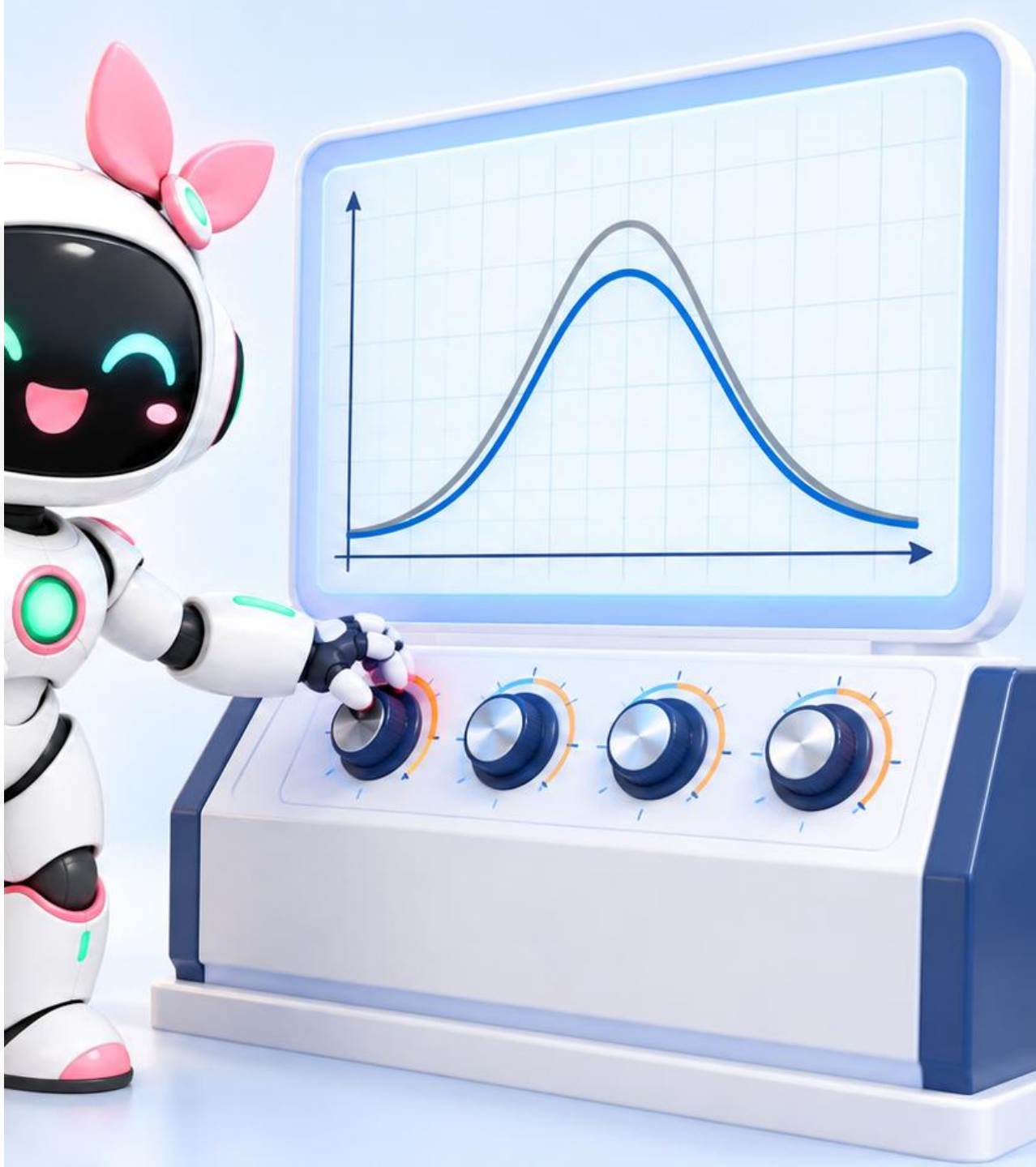
CLASS 1

AI、機器學習與神經網路入門

電腦如何從例子中學出一個函數？

函數、學習與第一次泛化檢查



三小時的學習路線

實作與六個理解檢查穿插進行。每題約 2 分鐘，進階推導可課後讀，保留最後的自動訓練時間。

0:00–1:05　函數與第一次手動操作

認識 Colab，完成 A 直線挑戰，再理解 Loss 與梯度下降。1:05–1:15 休息。

1:15–1:55　片段觀察與手動對齊

完成 B 的 w、b、c 實驗，再到 C 調整三片段模型，留下至少三次修改紀錄。

1:55–2:30　模型身分與評估

將剛才的片段對應到神經元與隱藏層，理解訓練、驗證與過擬合。

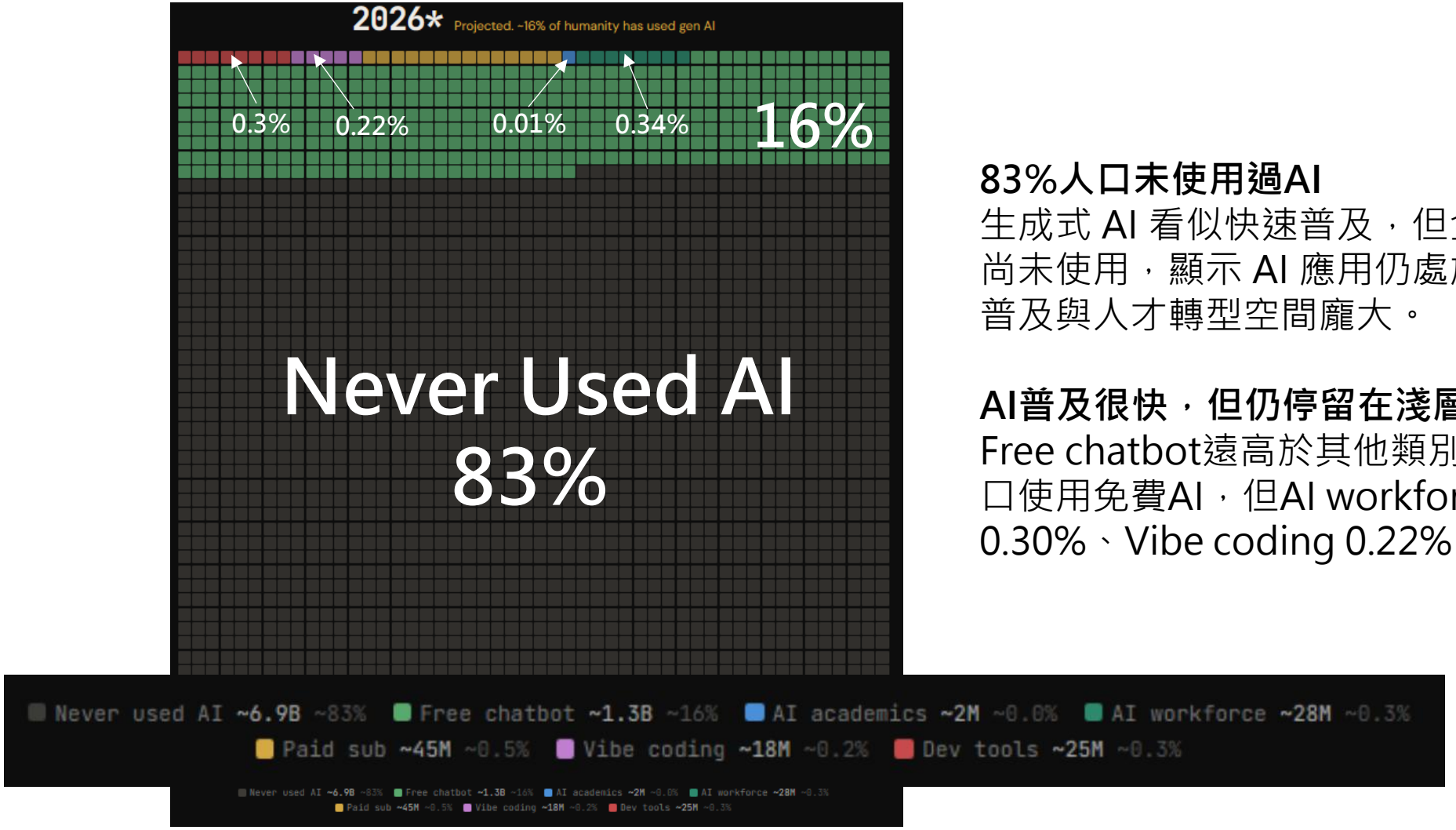
2:30–3:00　自動訓練與分享

E1 凍結手動起點，E2 執行訓練，觀察曲線如何接近目標。

最後保留結果與反思，繳交唯讀分享連結。

A、B、C 分段操作。E 區留到課程結尾，預設不會自動訓練。

課程淺談 – AI帶來的未來 (現況)

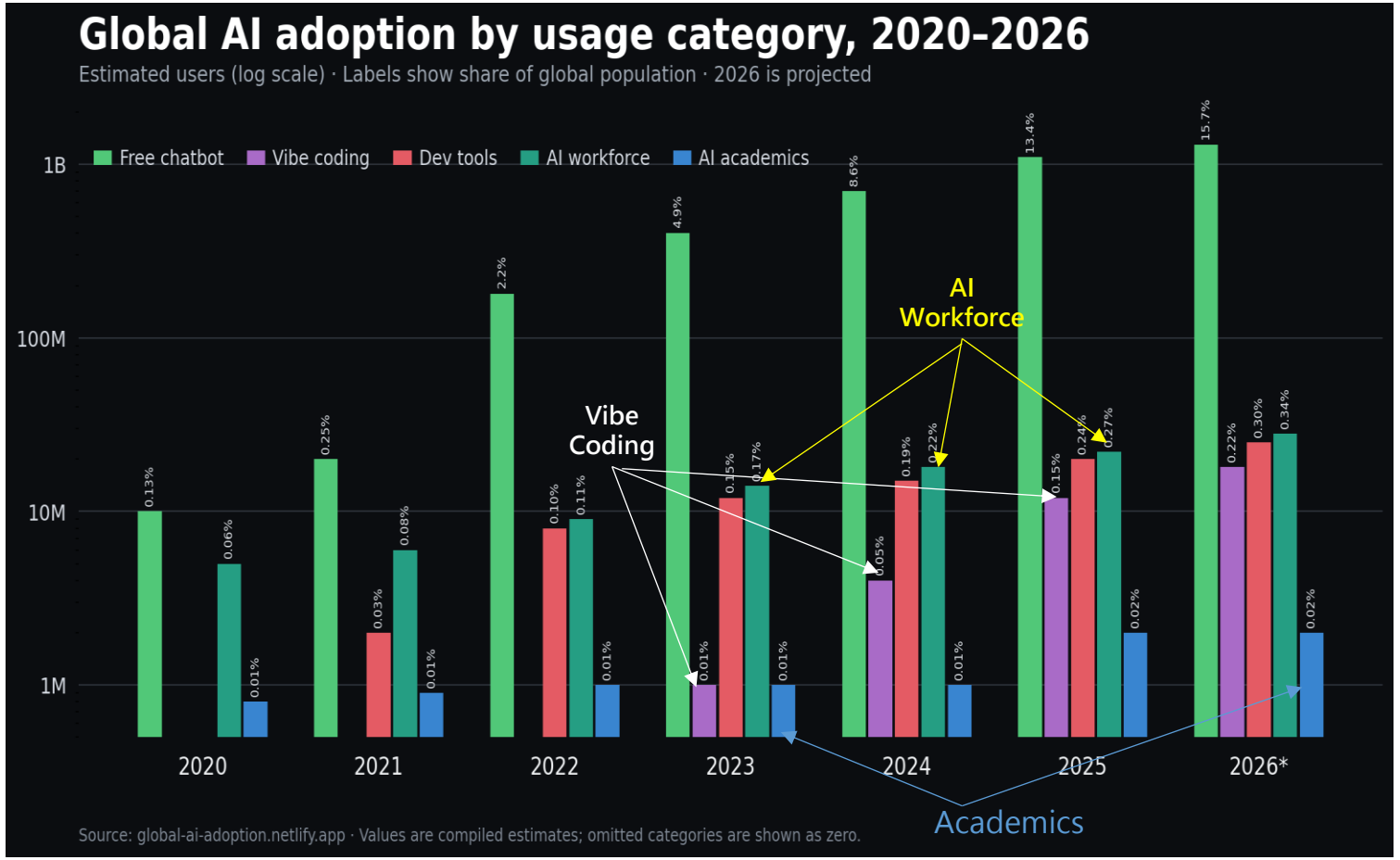


83%人口未使用過AI
生成式 AI 看似快速普及，但全球仍有約 83% 人口尚未使用，顯示 AI 應用仍處於早期成長階段，未來普及與人才轉型空間龐大。

AI普及很快，但仍停留在淺層使用
Free chatbot遠高於其他類別。2026年約15.7%人口使用免費AI，但AI workforce僅0.34%、開發工具0.30%、Vibe coding 0.22%。

網址：https://global-ai-adoption.netlify.app

課程淺談 – AI帶來的未來 (未來)



AI普及很快，但仍停留在淺層使用

Free chatbot遠高於其他類別。2026年約15.7%人口使用免費AI，但AI workforce僅0.34%、開發工具0.30%、Vibe coding 0.22%。**會跟AI聊天的人很多，真正把AI變成生產力系統的人仍很少。**

AI人才由學術界轉向產業擴張

AI academics約由80萬增加至200萬，成長相對緩慢；AI workforce則由500萬增至2,800萬。2026年產業AI人才約為學術研究人員的 **14倍**，**不用本科生，只要懂得如何正確使用AI就有源源不絕的工作機會**

等待適當的載具問世 (web2.0 vs. 手機)
迎來真正的大爆發時代

小艾的飲料：規則從哪裡來？

Class 0 的小艾會對配料打分。今天追問：這個打分函數可以怎麼得到？

人寫規則

設計者先決定珍珠加分、番薯扣分，再讓電腦依公式算分。這是人工指定偏好。

從例子學習

收集「配料份量、試喝分數」配對，讓模型調整參數，逼近已觀察到的分數。

新配方的用途

學完後輸入沒試過的配方，模型給出預測。要另做試喝，才知道預測是否可靠。

同樣是給飲料打分

人工版：珍珠每顆加幾分，由人設定。

學習版：每顆影響多少，由資料與訓練決定。

有輸出不代表曾經學習，要看參數如何取得。

AI、ML 與 DL 的包含關係

這三個名稱描述不同範圍。先看系統如何達成任務，再決定它屬於哪一類。

人工智慧 AI

讓系統執行辨識、推理、規劃等任務的領域，方法可以包含人寫的搜尋或推理規則。

機器學習 ML

讓系統利用資料改善任務表現。除了神經網路，也包含決策樹等方法。

深度學習 DL

使用多層神經網路學習表示。學到的中間表示能支援分類、回歸與生成。

判斷練習

人寫規則的棋局搜尋：可以屬於 AI。

用決策樹預測顧客是否回購：ML，通常不稱 DL。

用多層神經網路辨識影像：DL，也屬 ML 與 AI。

AI 包含 ML，ML 包含 DL；名稱範圍與模型效果是不同問題。

生成式 AI 描述的是輸出能力

「深度學習」主要描述方法，「生成式」描述產生內容的任務，兩者不是同一條分類軸。

判別 (回歸與分類)與生成

判別模型回答「這是什麼」或「數值多少」；生成模型產生符合條件的內容，例如文字、影像或聲音。

現代常見做法

大型語言模型與擴散模型多使用深度學習，但生成模型的概念並不限定於深度神經網路。

輸出像真不等於正確

模型學到資料中的規律，生成內容仍可能與事實或使用條件不符，需要另外查證與評估。

相同照片，不同任務

辨識照片中的飲料品項：分類。

估計杯中剩餘容量：回歸。

依照片寫一段商品介紹：生成。

先說清楚任務，再說明使用什麼模型。

回歸、分類與生成的差別

看輸出形式可以快速定位問題，但還要一起決定什麼樣的錯誤最重要。

任務	輸出與生活例子	如何評估
回歸	預測外送需要 32 分鐘	預測值與實際值差幾分鐘
分類	判斷簡訊為正常或垃圾	錯判、漏判各有多少
生成	產生客服回覆或菜單圖片	內容正確性、相關性與品質

分類結果也可用數字編碼；數字 0 / 1 不代表它就是回歸。

一筆資料裡有什麼？

監督式學習用已知答案的例子學習。輸入的欄位叫特徵，想預測的答案叫標籤。

特徵 x

模型在預測當下可以取得的資訊，例如距離、是否下雨、尖峰時段。

標籤 y

訓練時已記錄的結果，例如實際送達分鐘數。新訂單在預測時還沒有這個答案。

資料品質

欄位單位要一致，缺值與錯誤紀錄要處理。未來才知道的欄位不可拿來冒充現在的特徵。

一筆外送紀錄（教學資料）

距離 x = 5 公里，下雨 = 是。

實際送達 y = 28 分鐘。

「送達後顧客評分」在出發時拿不到，不應當成即時預測輸入。

資料欄位的可取得時間，和數值本身一樣重要。

機器學習的三個步驟

把「學習」變成可操作的流程，需要模型、衡量錯誤的方法，以及更新規則。

步驟	要決定什麼	外送例子
1 定義模型	哪些函數可以被選到？	$\hat{y} = b + wx$
2 計算損失	怎麼把多筆誤差合成分數？	平均平方誤差 MSE
3 最佳化參數	如何根據分數修改參數？	梯度下降調整 w、b

先有明確任務與資料，三步驟才有可比較的目標。

Colab 檔案與課程中的停靠點

本課使用 Class1_課中手動與結尾自動_v11.ipynb，整堂都在同一份副本操作。

0 與 A：環境、直線

先執行 0 環境準備，再到 A 改 linear_w、linear_b。完成直線任務後先停下。

B 與 C：折線片段、組合

B 觀察一個「有轉折的線段」。C 把三個片段相加，比較你的曲線與目標有多接近。

D 與 E：理解、訓練

D 說明這些片段組成什麼模型。課程結尾才開啟 E，讓電腦接手調整數字。

發生問題先定位

找不到變數：先檢查 0 環境準備是否執行。

圖沒有變：修改後要重新執行該格。

重新連線後沒有紀錄：執行狀態可能已清除，先看已保存的輸出。

Colab 依原順序操作；看圖問答與短實驗可另開 Class1_互動教室_v9.html。

操作 0：開啟 Notebook 與保存副本

將課程提供的 .ipynb 檔放進 Colab，先確認自己能編輯並保存。

1　開啟與上傳

進入 colab.research.google.com，登入 Google 帳號。開啟筆記本視窗選「上傳」，選擇 v11 的 .ipynb。

2　確認檔案位置

在檔案選單保存到 Google 雲端硬碟。若開啟的是教師共用版本，使用「在雲端硬碟中儲存副本」。

3　執行環境準備

將檔名改成含姓名或學號的版本。按 0 環境準備左側執行鈕，等待出現準備完成訊息。

本課所需環境

一般 Python CPU 即可。

不用申請 GPU，也不用掛載 Google Drive。

程式使用 NumPy 與 Matplotlib，工具定義已放在 0 區。

介面語言可能不同，請依 File、Upload、Save a copy 與 Share 等對應名稱辨識。

操作方法：改數字、執行、讀輸出

一個程式格會從上往下執行。修改畫面上的文字後，需要再次執行才會更新結果。

1　只改指定欄位

本課優先修改有註解的參數行。保留變數名稱、括號、逗號與英文符號，先不動下方繪圖程式。

2　執行目前這一格

點程式格左側執行鈕。等待執行結束，再讀這一格下方的新圖與數值。先不要使用「全部執行」。

3　回到同一格再試

一次只改一個數字。下方輸出更新後，比較曲線與前次結果，再決定下一次改動。

兩種常見錯誤

SyntaxError：檢查括號、逗號與全形符號。

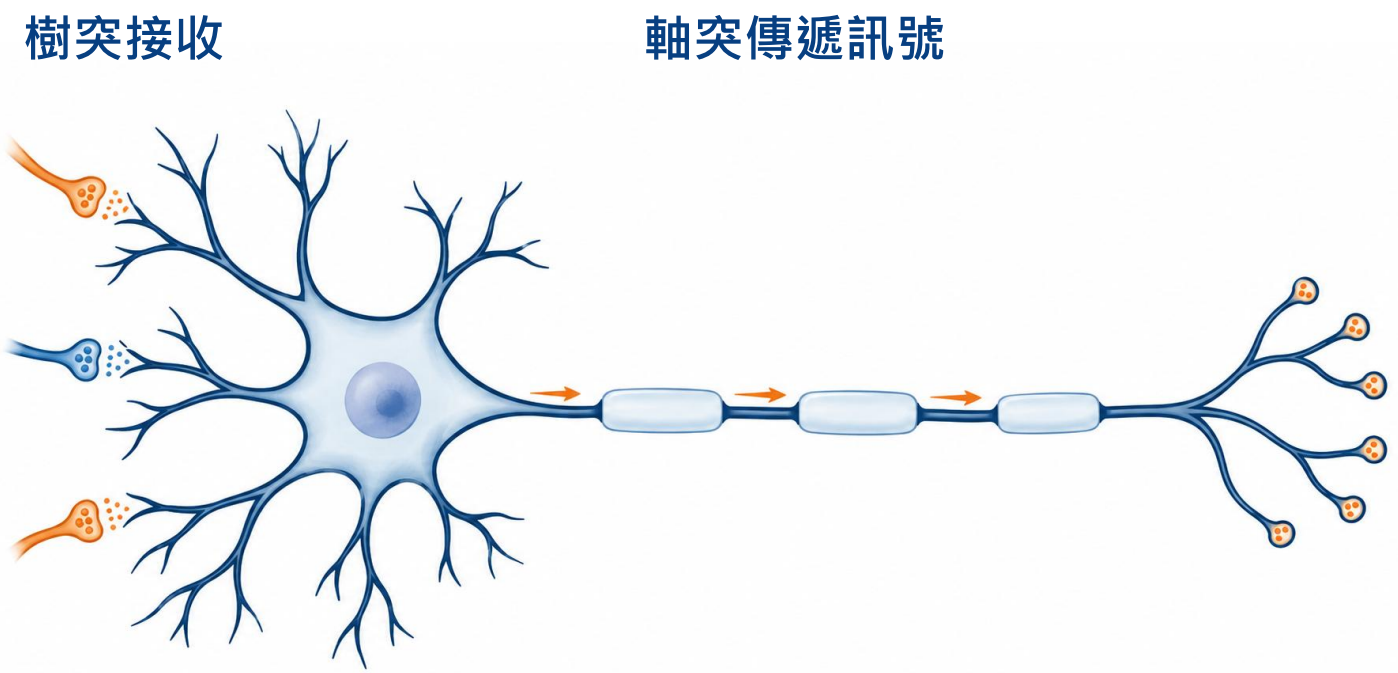
NameError：先執行 0 區，再依序執行目前關卡。

先停下讀錯誤最後一行，需要協助時指出區塊名稱。

同一格重跑會更新輸出。C 區另外累積實驗紀錄，請保留其表格。

神經元類比：訊號的影響有強有弱

先借用生物的「接收、整合、傳遞」理解權重，再回到可計算的數學模型。



接收訊號

樹突接收其他神經元傳來的訊號。

連結影響不同

突觸傳遞可能促進或抑制反應。

數學上的權重 w

用係數表示某個輸入影響多少。

單一輸入的數學版：影響量 = w × x

這是有限類比。人工模型用數字計算，並非完整模擬人腦的生理機制。

線性模型：ŷ = b + wx

把「輸入影響多少」寫成係數 w。外送例子先只用距離 x，預測時間記作 ŷ。

w 是權重 weight

在此單一特徵模型中，w 是斜率，表示距離增加一公里，預測增加多少分鐘。

b 是偏差 bias

b 是 x = 0 時的預測，可理解為出餐、裝袋等固定時間的簡化表示。

參數與資料分開

距離 x 來自訂單，w 和 b 是模型參數。不同參數會產生不同預測。

代入一次就能預測

假設 b = 10 分鐘，w = 3 分鐘 / 公里。

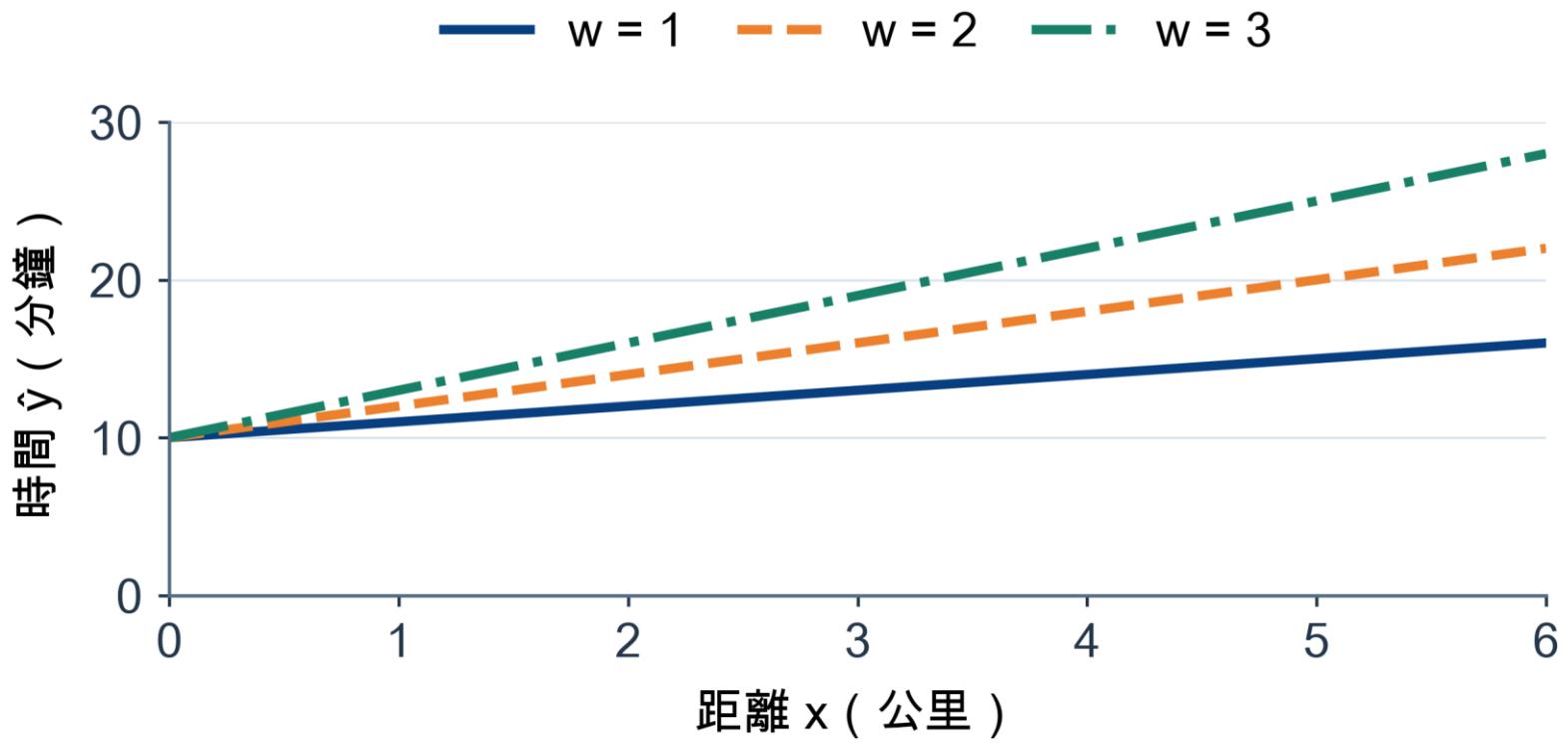
x = 5 公里時：ŷ = 10 + 3 × 5 = 25 分鐘。

這是教學用模型與數值，不是實際配送承諾。

w 的單位是「輸出單位 / 輸入單位」，b 與輸出同單位。

權重改變的是斜率

固定 b = 10，單獨改 w。距離越大，斜率不同造成的預測差異越明顯。



讀圖

三條線在 x = 0 都是 10。當 w 從 1 增加到 3，每多一公里的增量變大。

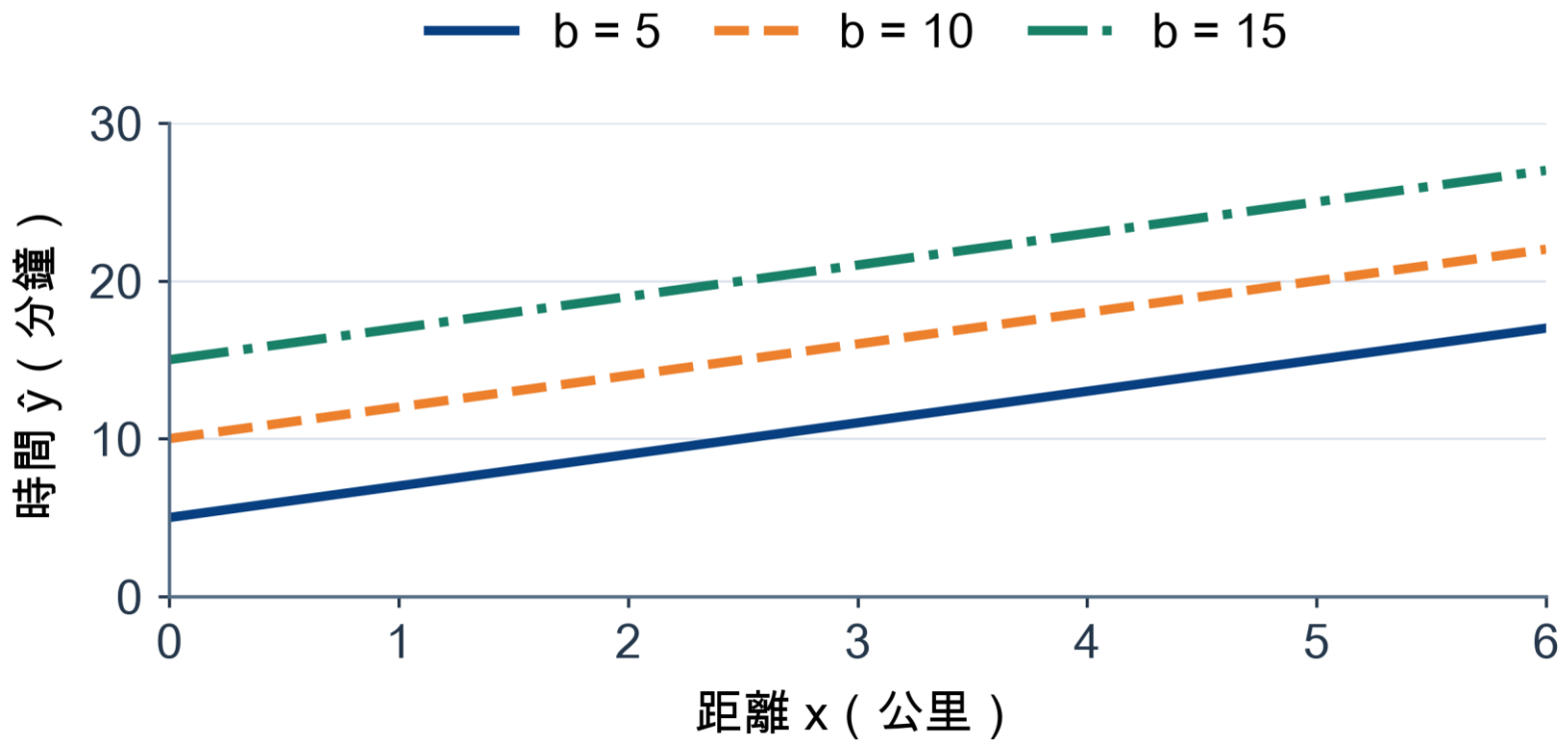
算差異

5 公里時，w = 1 預測 15 分鐘，w = 3 預測 25 分鐘，相差 10 分鐘。

固定其他參數後再比較，才能辨認單一參數的影響。

偏差改變的是整體高度

固定 w = 2，單獨改 b。每個 x 的預測都增加相同數量，因此三條線互相平行。



讀圖

b 從 5 改成 15，每個距離都多 10 分鐘，斜率沒有改變。

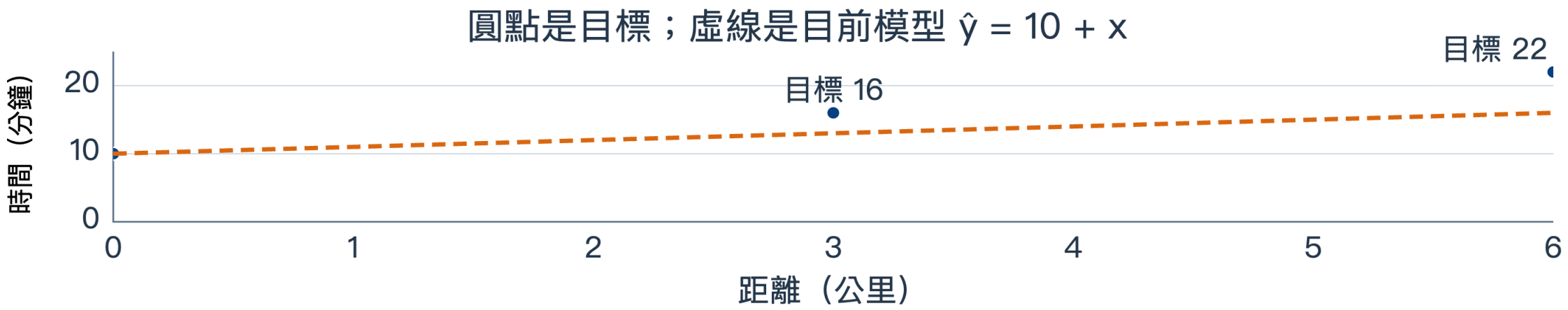
生活解釋

若出餐準備整體增加 10 分鐘，而每公里車程不變，可以先用 b 的變化表示。

先記住：直線中的 b 平移整條線。稍後換成有轉折的函數，再比較它的作用。

停下來想 ①：該改高度，還是斜率？

模型在 0 公里預測正確，距離越遠卻越低估。先改哪個參數最符合這個現象？



A 只把 b 增加，整條線往上移

B 把 w 增加，讓每公里增量變大

C 把 w 減少，讓線更平

先看圖選答案，再用圖中的證據說明。下一頁揭曉。

先獨立想 45 秒，再和同桌交換理由。選答案之外，請說出你用了哪個觀念。

A 關卡：手動讓直線接近目標

在 A 區先執行初始設定，觀察灰色目標與藍色預測。此關沒有自動訓練。

先調高度

固定 linear_w = 1，把 linear_b 從 8 改為 10，再執行。觀察 x = 0 的起點如何變動。

再調斜率

保留 linear_b = 10，把 linear_w 從 1 改為 2，再執行。比較近距離與遠距離的差異。

解釋自己的修改

說出哪個參數讓整條線平移、哪個改變每公里增量。數字對齊後，還要能解釋原因。

只修改這兩行

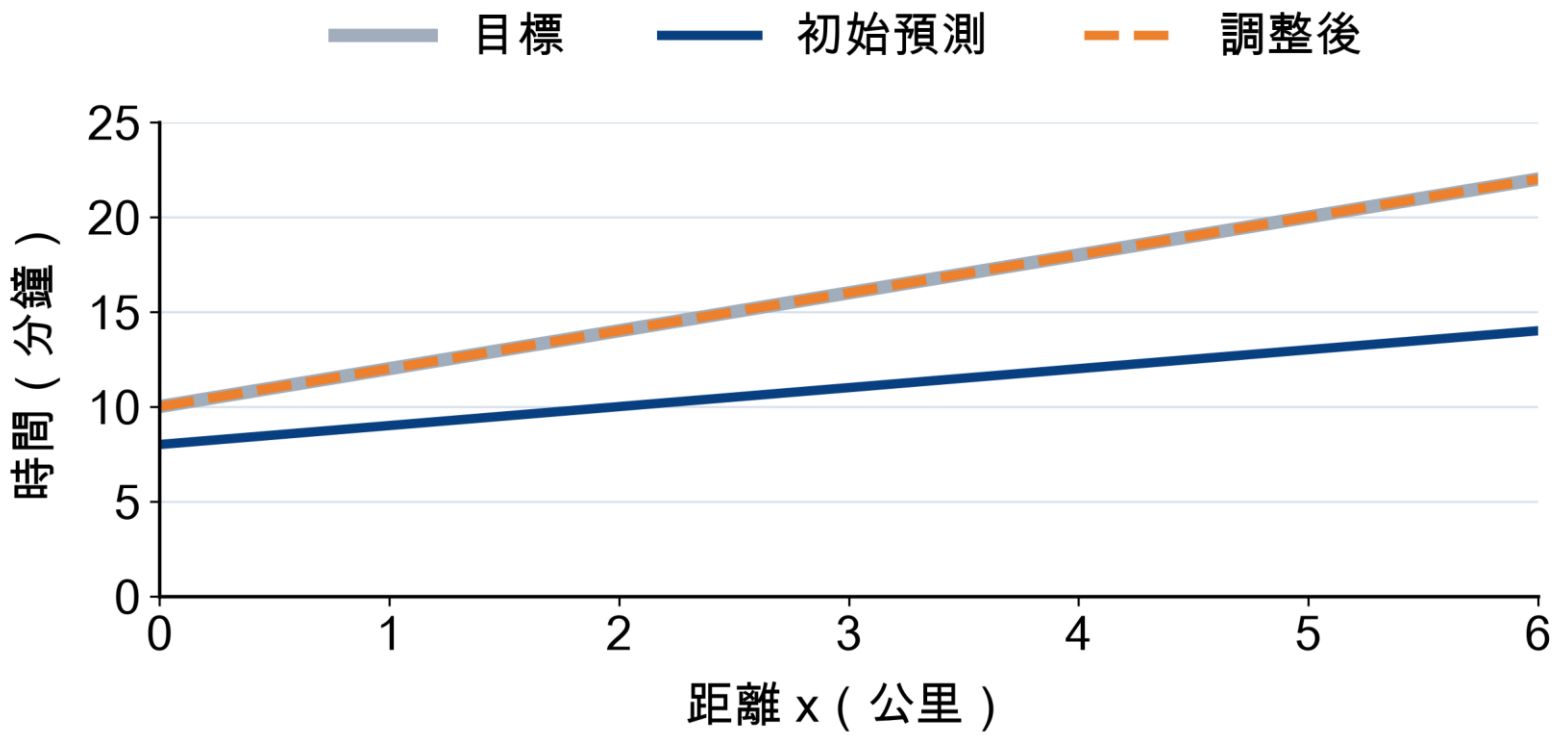
```
linear_w = 1.0

linear_b = 8.0
```

也可先開互動教室「手動降低 MSE」：調 w、b，保留三次嘗試，再說明理由。

A 結果判讀：誤差出現在哪裡？

A 區的 x 是距離，y 是教學設定的分鐘數。先看曲線是否接近，再讀下方誤差分數。



看起點與斜率

初始 w = 1、b = 8。先把 b 改成 10，再把 w 改成 2，預測就能接近目標。

誤差分數先看方向

x = 0 時是 10，x = 3 時是 16。本例完全重疊，分數為 0；MSE 的完整算法稍後說明。

A 使用無雜訊合成直線，適合學參數效果，不代表真實問題都能做到零誤差。

多個特徵可以有不同權重

一個輸入看懂後，也能把多個輸入的影響相加。用小艾配料比較正權重與負權重。

正權重與負權重

在其他特徵固定時，正權重讓預測隨特徵增加而提高，負權重讓它降低。

小艾的配料例子

珍珠數量可以正向影響分數，番薯份量可以負向影響分數。真實偏好可能還有飽和或交互作用。

解釋限制

權重大小受到單位與特徵相關性影響。大權重不一定代表重要，也不能直接推出因果關係。

先用線性近似理解加總

教學設定：分數 = 40 + 2 × 珍珠顆數 − 3 × 番薯份數。

珍珠 8 顆、番薯 2 份：40 + 16 − 6 = 50。

減少番薯增加分數，這是負向關係，不是數學上的反函數。

多輸入的加總先看到這裡。接著回到外送，練習替一組參數計算誤差。

這組外送參數，究竟預測得多準？

另取三筆教學訂單：距離與實際時間固定，先試 w = 4、b = 8，逐筆算預測與誤差。

模型設定：w = 4、b = 8　　預測：$\hat{y} = 8 + 4x$

距離 x（公里）	代入模型得到 $\hat{y}$（分鐘）	答案 y（分鐘）	誤差 $e = \hat{y} - y$
1	$8 + 4 \times 1 = 12$	14	$12 - 14 = -2$
2	$8 + 4 \times 2 = 16$	19	$16 - 19 = -3$
3	$8 + 4 \times 3 = 20$	22	$20 - 22 = -2$

固定不動：距離 x、答案 y　　可調整：w、b　　計算得到：預測 $\hat{y}$、誤差 e

三筆都低估，但差距不完全相同。下一頁把多筆誤差合成一個比較分數。

誤差與損失：從每一筆算到整體

先定義符號，再決定如何處理正負誤差。i 是第幾筆，N 是資料總筆數。

① 每筆預測與誤差

$$\hat{y}_i = b + wx_i \qquad e_i = \hat{y}_i - y_i$$

$\hat{y}_i$ 是第 i 筆預測，y_i 是答案。$e_i < 0$ 代表低估。

② 先把誤差平方

$$\ell_i = (\hat{y}_i - y_i)^2 = e_i^2$$

平方讓正、負誤差都產生非負的懲罰，避免互相抵銷。

③ N 筆再取平均

$$L(w,b) = (1/N)\,\Sigma_{i=1}^{N}\,[b + wx_i - y_i]^2$$

Σ 表示從 i = 1 加到 N，再除以 N。這就是本課的 MSE。

本課選平方誤差當損失。下一頁沿用剛才三筆訂單，完整算一次 MSE。

平均平方誤差 MSE 的計算

平均平方誤差 MSE（Mean Squared Error）：沿用誤差 −2、−3、−2，先平方再平均。

資料	誤差 e	平方 e^2	說明
第 1 筆	−2 分鐘	4 分鐘²	負誤差平方後仍為正
第 2 筆	−3 分鐘	9 分鐘²	大誤差受到更大懲罰
第 3 筆	−2 分鐘	4 分鐘²	每筆使用相同計算方法

MSE = [$(-2)^2 + (-3)^2 + (-2)^2$] / 3 = 17/3 ≈ 5.67 分鐘²。

MAE 與 MSE：公式如何改變懲罰？

兩種方法都先逐筆處理誤差，再對 N 筆取平均。差別就在絕對值與平方。

MAE | 平均絕對誤差

Mean Absolute Error

$$MAE = (1/N)\ \Sigma_{i=1}^{N} |\hat{y}_i - y_i|$$

絕對值保留「差多遠」。誤差放大 k 倍，該筆懲罰也放大 k 倍。

誤差 1、10　→　懲罰 1、10

單位與目標相同，例如分鐘。

MSE | 平均平方誤差

Mean Squared Error

$$MSE = (1/N)\ \Sigma_{i=1}^{N} (\hat{y}_i - y_i)^2$$

平方放大大誤差。誤差放大 k 倍，該筆懲罰變成 k^2 倍。

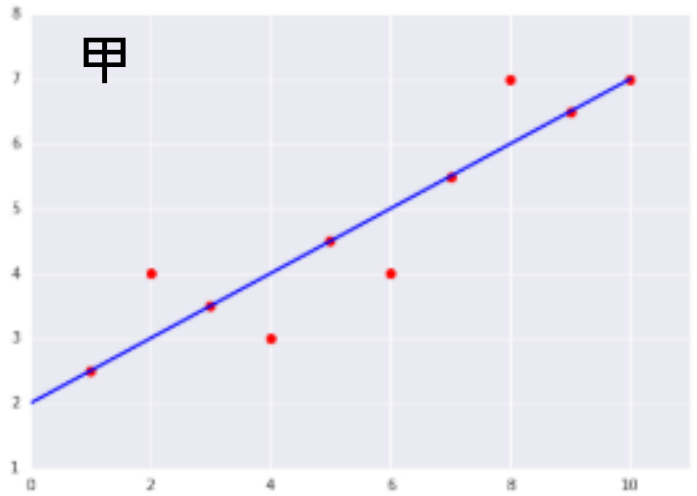
誤差 1、10　→　懲罰 1、100

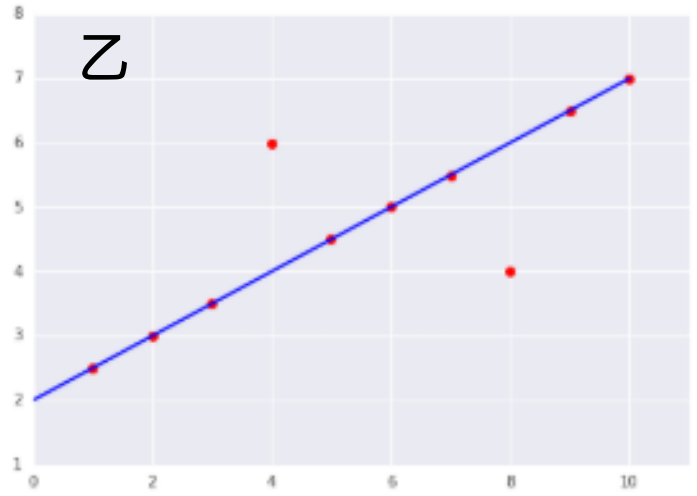
單位為目標的平方，例如分鐘^2。

較大的錯誤若代價更高，可考慮 MSE。仍要先確認極端資料是真實事件還是記錄錯誤。

停下來想 ②：錯得少筆，就一定較好？

同樣四筆資料，左圖甲、右圖乙顯示每筆預測誤差。哪個模型的 MSE 較高？先看圖，再選答案。





A 甲較高，因為錯了四筆

B 乙較高，因為平方放大大誤差

C 一樣高，正負誤差會抵銷

先看圖選答案，再用圖中的證據說明。下一頁揭曉。

先獨立想 45 秒，再和同桌交換理由。選答案之外，請說出你用了哪個觀念。

固定資料，替不同參數打分數

回到 A 的一筆資料：x = 1 公里、答案 y = 12 分鐘。固定 b = 10，只搜尋 w。

候選 w	$\hat{y} = 10 + w$	$e = \hat{y} - 12$	$Loss = e^2$
0	10	−2	4
1	11	−1	1
2	12	0	0
3	13	1	1
4	14	2	4
5	15	3	9

同一套計算，每組都重做

固定：x = 1、y = 12、b = 10

只改：w

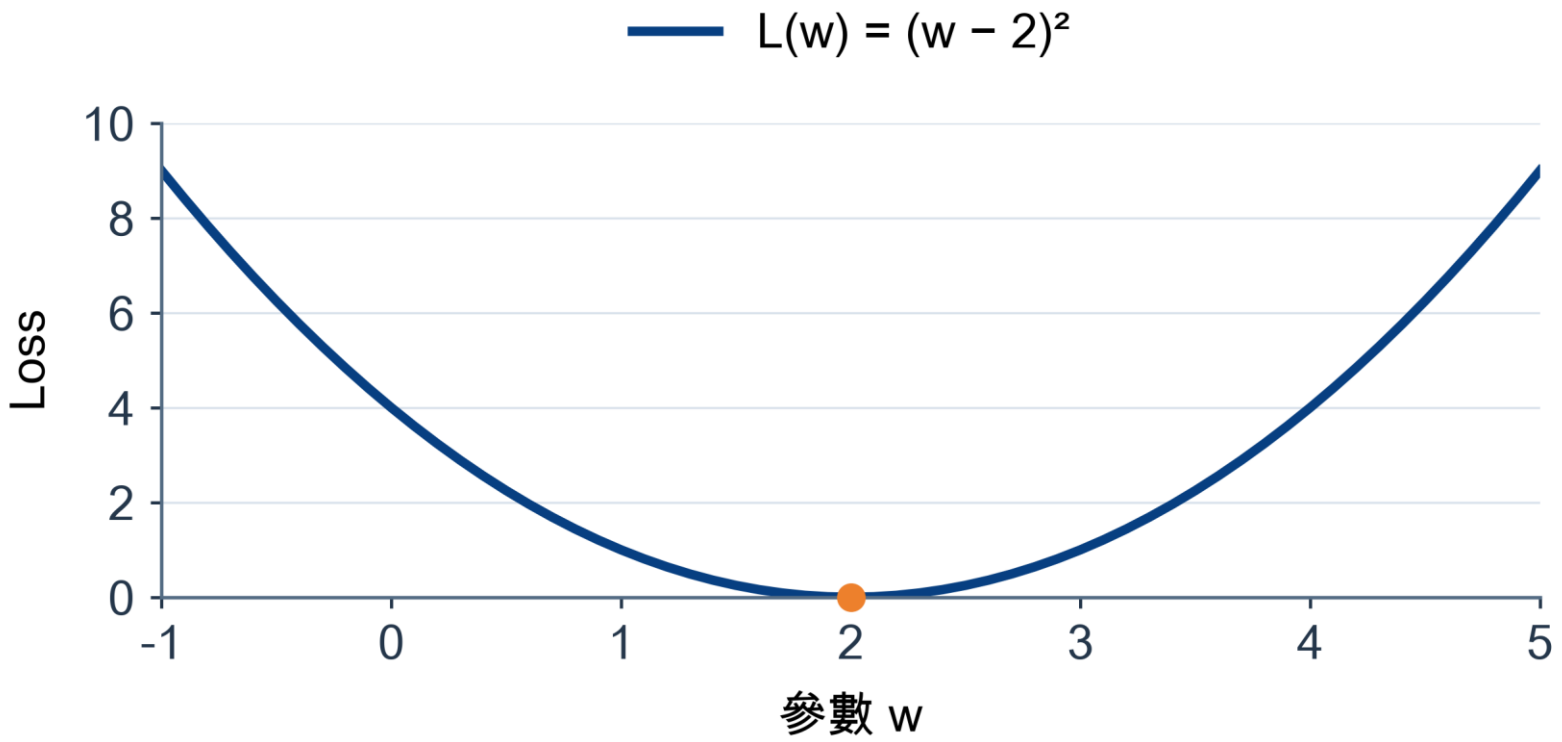
$$L(w) = (10 + w - 12)^2$$
$$= (w - 2)^2$$

此處只取一筆資料，目的是看懂搜尋過程。

每個候選 w 都得到一個 Loss。下一頁把「w 與它的分數」畫成曲線。

把候選分數畫成誤差曲線

沿用上一頁同一筆資料，$L(w) = (10 + w - 12)^2 = (w - 2)^2$。現在把表格畫成圖。



橫軸改成候選 w

w = 0、1、2、3、4、5，分數為 4、1、0、1、4、9。低點代表較好的設定。

「地形」只是比喻

曲線高度是 Loss。接下來的問題是：怎樣找低處，而不用先把所有候選都試完？

輸入 x 與答案 y 都固定。現在改的是參數 w，觀察的是這組參數的 Loss。

第一種找法：窮舉候選參數

窮舉法把指定候選值逐一代入、算 Loss，再選最低者。剛才的六列就能這樣比較。

只有一個參數時

試 w = 0、1、2、3、4、5 共 6 次，候選中 w = 2 的分數最低。

多個參數要試組合

若 w 有 21 個候選、b 也有 21 個，需要 21 × 21 = 441 組，而非只試 42 次。

參數越多，成本越高

10 個參數各選 21 個值，就是 21^{10}，約 16.7 兆組。細分候選還會增加成本。

下一個問題

小範圍窮舉可以很可靠，但只保證指定候選中的最好。

如果能知道「附近往哪邊分數較低」，可否少試很多組？

窮舉用候選分數逐一比較。接下來改用目前位置附近的變化，決定下一步。

附近的分數，能告訴我們往哪邊走

目前 w = 5、Loss = 9。先把 w 向左、向右各試 0.1，觀察分數怎麼變。

附近候選 w	Loss = $(w - 2)^2$	觀察
4.9	8.41	減少 w，分數下降
5.0	9.00	目前位置
5.1	9.61	增加 w，分數上升

附近的變化率 ≈ [L(5.1) – L(4.9)] / (5.1 – 4.9)

= (9.61 – 8.41) / 0.2 = 6

正值：w 增加時分數上升。
想降分，先往較小的 w 走。

附近的變化率提供方向線索。下一頁把這個局部變化率正式叫作導數與梯度。

梯度：目前位置的上升方向

剛才每增加 0.1，Loss 大約增加 0.6。導數把「附近改變多少」寫成精確的局部變化率。

只有 w 時，先看導數

$L(w) = (w - 2)^2$，導數 $dL/dw = 2(w - 2)$。$w = 5$ 時是 6，與剛才的試探相符。

方向由正負號判斷

導數 > 0，減少 w 可往低處走；導數 < 0，增加 w 可往低處走。只適用於附近的小步。

多個參數時，合成梯度

同時調 w、b，就分別算 $\partial L/\partial w$、$\partial L/\partial b$。這組局部變化率組成梯度 ∇L。

讀符號，不先背微分

dL/dw：w 稍微改變時，Loss 改變多快。

∂：其餘參數先固定，單看其中一個。

∇L：把各參數的偏導數排在一起。

梯度提供局部上升方向。為了降低 Loss，更新時往負梯度方向走。

梯度下降為何能減少搜尋次數？

梯度下降（Gradient Descent）反覆做三件事：算目前梯度、反向走小步、到新位置重算。

窮舉：先安排候選清單

在指定網格上逐一算 Loss。參數多時，候選組合以次方成長。

梯度下降：每次依方向更新

算一組梯度後，同時調整各參數。通常不需要先走過整個候選網格。

為何小步可能降低 Loss

局部可微且梯度非零時，負梯度是下降方向。步長足夠小才有下降依據。

公式的直覺

新參數 = 舊參數 − 學習率 × 梯度

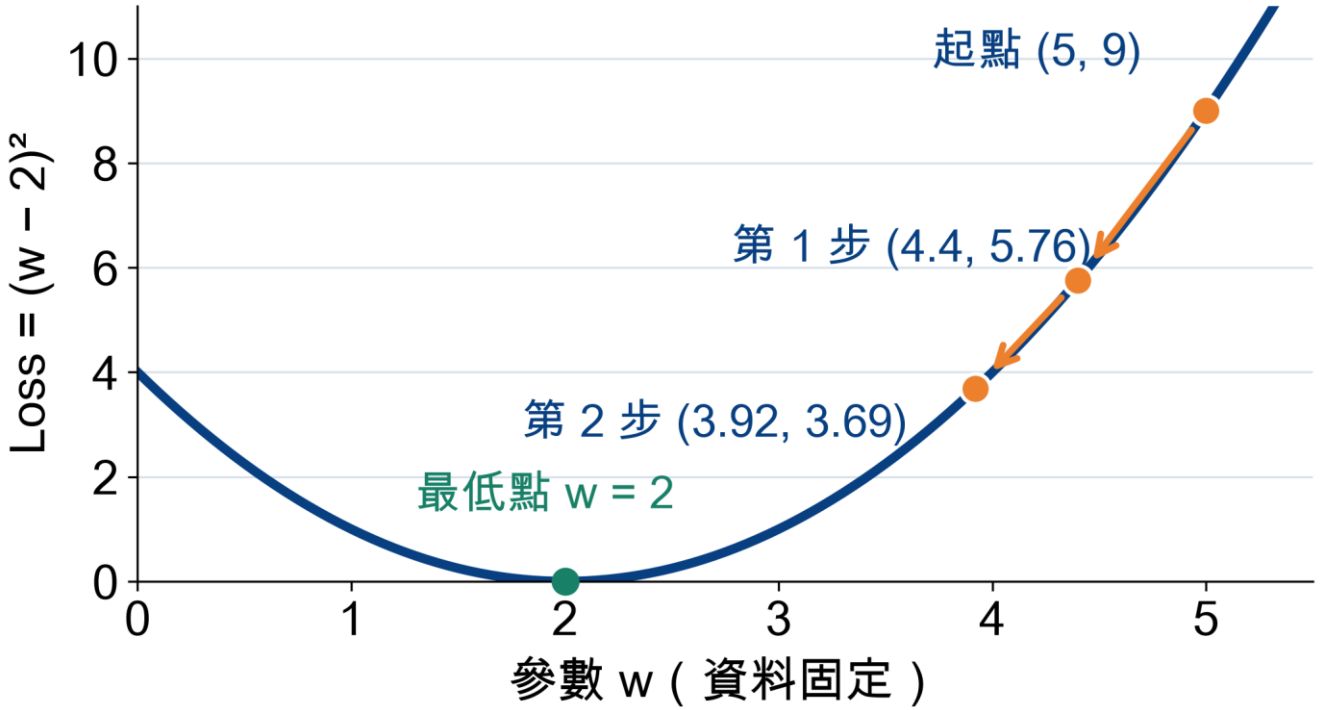
梯度只看目前附近，不能保證神經網路的全域最佳解。

不連續或離散選項，未必適合直接用梯度。

優點是利用方向資訊減少盲試。代價是局部方法：步長、起點與函數形狀都會影響結果。

梯度下降：位置、坡度與每一步

同一條 $L(w) = (w - 2)^2$，從 $w = 5$ 出發。學習率 $\eta = 0.1$，控制每一步走多遠。



① 起點：先算坡度

$w = 5$，$L = 9$
$g = 2(5 - 2) = 6$

② 第 1 步：反向移動

$w = 5 - 0.1 \times 6 = 4.4$
$L = 5.76$，新梯度 $g = 4.8$

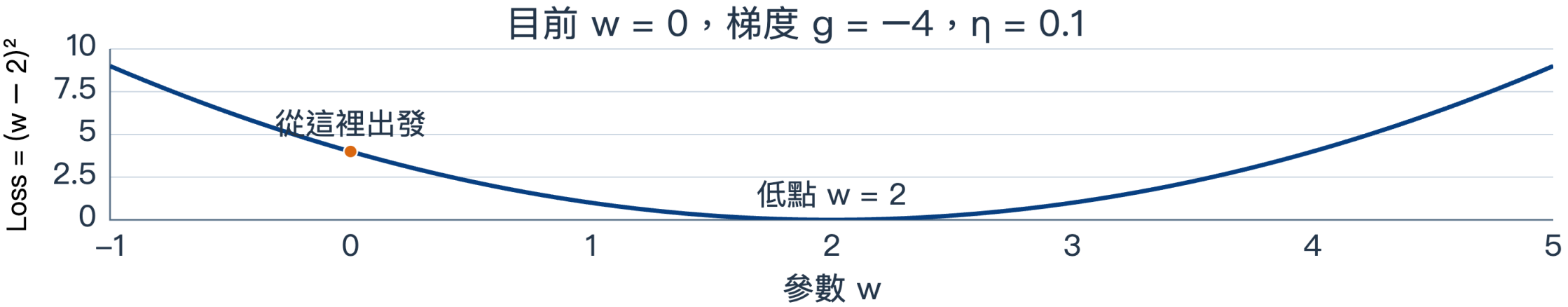
③ 第 2 步：用新坡度

$w = 4.4 - 0.1 \times 4.8 = 3.92$
$L \approx 3.69$，再算下一次梯度

每到新位置都要重算梯度。先練習判斷更新方向，再比較不同學習率的路徑。

停下來想 ③：負梯度時，w 往哪裡走？

$L(w) = (w - 2)^2$，目前 $w = 0$、梯度 $g = -4$，學習率 $\eta = 0.1$。下一步 w 是多少？



A −0.4

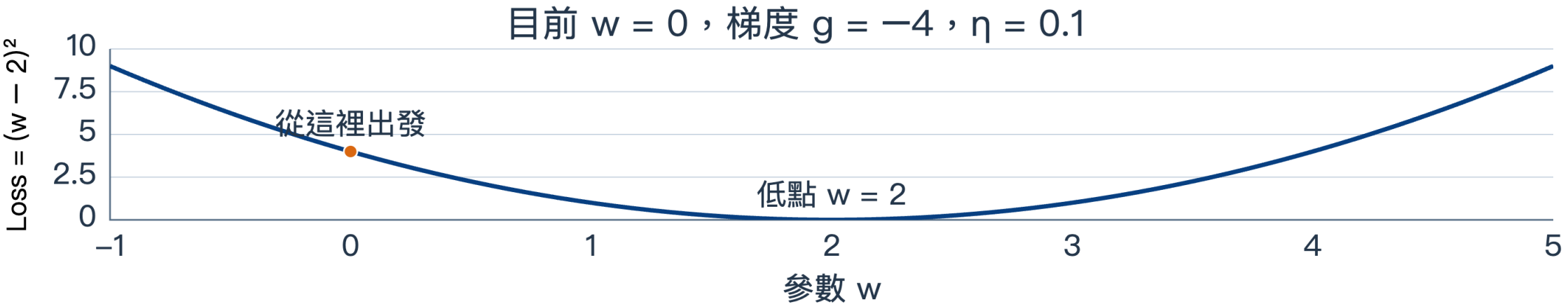
B 0.4

先看圖選答案，再用圖中的證據說明。下一頁揭曉。

先獨立想 45 秒，再和同桌交換理由。選答案之外，請說出你用了哪個觀念。

一起看答案 ③：負梯度時，w 往哪裡走？

$L(w) = (w - 2)^2$，目前 $w = 0$、梯度 $g = -4$，學習率 $\eta = 0.1$。下一步 w 是多少？



A −0.4

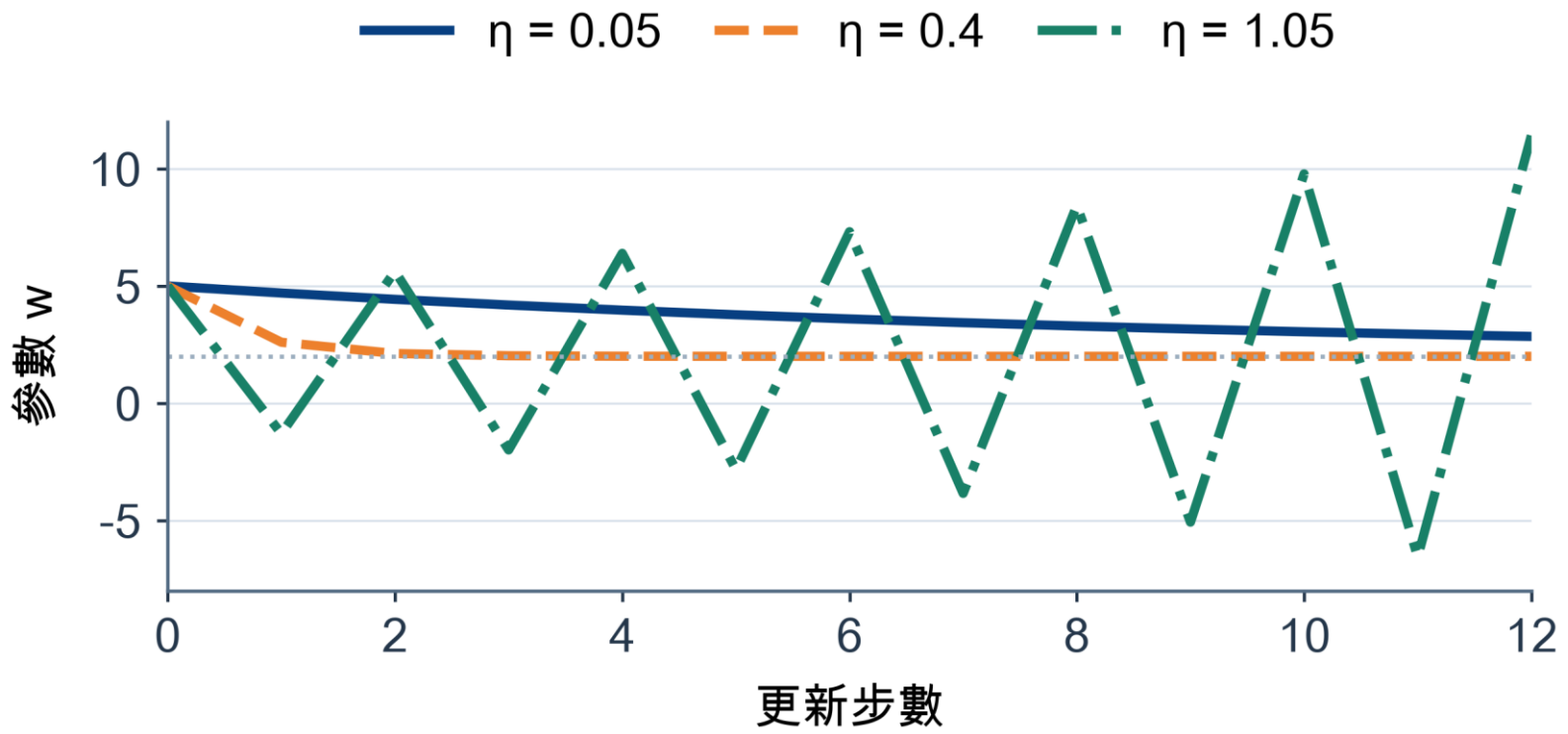
B 0.4

答案 B　w新 = 0 − 0.1 × (−4) = 0.4。往右靠近低點，Loss 從 4 降至 2.56。

延伸操作：互動教室「梯度下一步」，先猜方向，再按一次更新。

學習率太小、合適與太大

同一個 $L(w) = (w - 2)^2$，從 $w = 5$ 出發，只改學習率，就會得到不同的更新軌跡。



太小與合適

η = 0.05 靠近低點較慢；η = 0.4 在此例收斂較快。這些數字不是所有模型的通用設定。

太大

η = 1.05 會在低點兩側越跳越遠。實作若出現巨大 Loss 或非有限值，先檢查學習率與尺度。

水溫旋鈕的類比：轉太多容易過頭，下一次又過度修正。

剛才改了兩種數字：w 與學習率

w 決定預測內容，η 決定更新步伐。這就是「參數」與「超參數」的第一個具體區別。

角色	剛才的例子	在流程中做什麼
模型參數	w：5 → 4.4 → 3.92	帶入模型算預測，訓練會更新
超參數	η：0.1，或改試其他步長	控制怎麼更新，不直接當預測答案
資料	固定 x = 1、y = 12	提供輸入與比較答案，不隨分數改寫

先記住 w 與 η 的分工。稍後模型變大時，神經元數量也屬於需要設定的超參數。

休息前：把三步驟連起來

你已在 A 調整直線。再以早餐店等餐時間為例，把模型、Loss 與參數更新連起來。

模型選擇

x 是前面等待的訂單數，$\hat{y} = b + wx$ 預測分鐘數。w 表示每多一單增加的預測時間。

誤差檢查

用實際等餐時間計算 MSE。若所有預測都低估，可先檢查 b；若單數越多偏差越大，再檢查 w。

更新與限制

梯度下降可降低訓練 Loss。若大量訂單能平行製作，直線可能不足以描述真正關係。

討論 1 分鐘

A 直線中，你先調 w 還是 b？怎麼從圖上判斷？

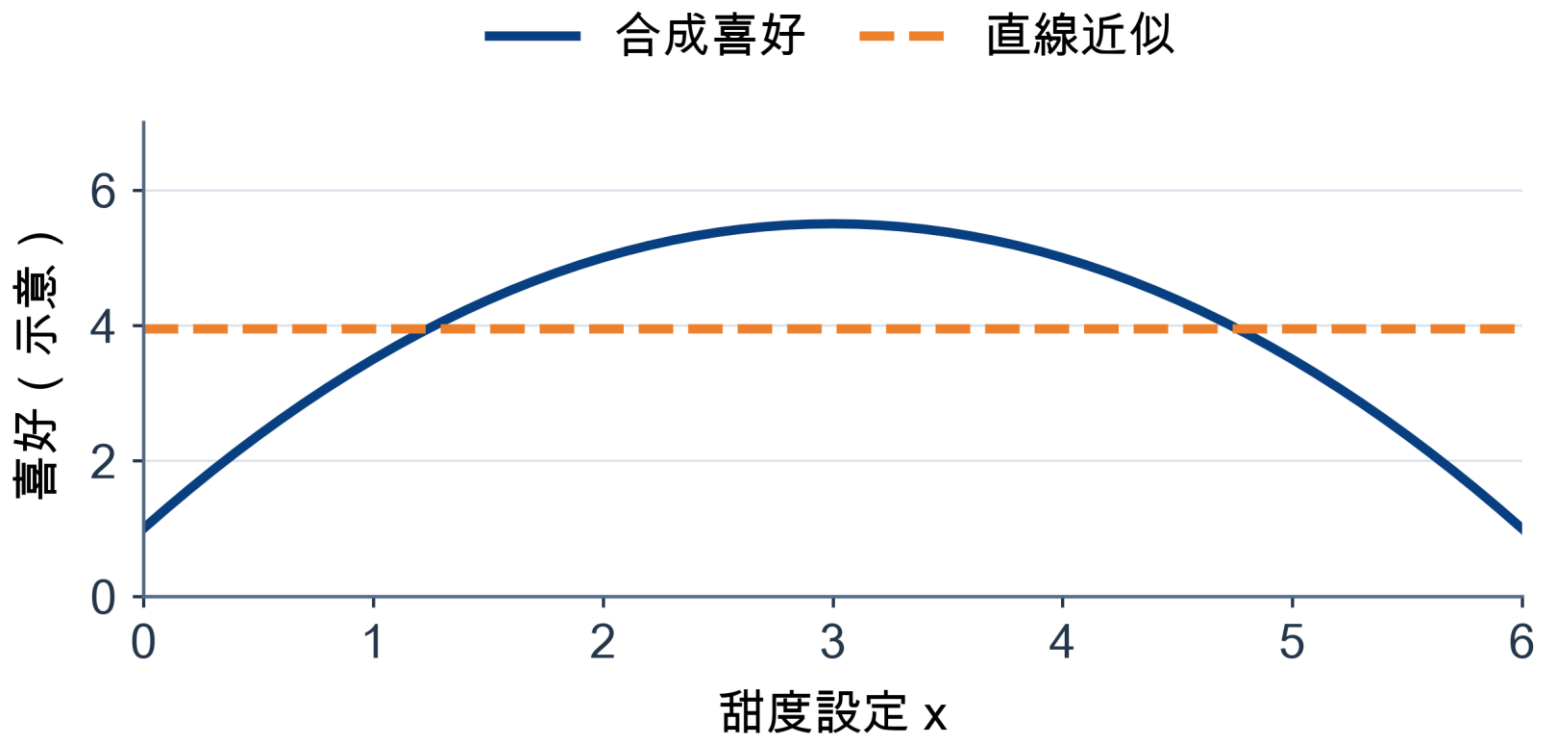
若目標有山峰，增加直線的訓練次數能解決嗎？

休息 10 分鐘，回來用片段拼出有轉折的曲線。

學習方法再好，也受到模型可表達形狀的限制。

生活中的關係常有轉折

剛才學會替一條直線找參數。若目標先上升再下降，找得再準的直線仍可能不夠。



固定斜率的限制

$\hat{y} = b + wx$ 每增加一單位 x，預測都增加固定的 w，無法同時表達上升和下降。

下一步要改模型形式

先試把多層計算串起來。若仍是直線，就需要加入會產生轉折的非線性運算。

本圖是合成喜好曲線。真實偏好要靠資料確認，不能直接套用。

只把直線運算疊起來，會產生轉折嗎？

接續上一頁的山形需求。先把兩次「乘權重、加偏差」串起來，看是否能畫出新形狀。

第一層

$h = 2x + 1$，把輸入放大兩倍，再加上 1。

第二層

$\hat{y} = 3h - 4$。把 h 代入後：$\hat{y} = 3(2x + 1) - 4 = 6x - 1$。

推到更多層

矩陣形式也一樣：線性權重與偏差可合併。中間需要非線性，才能突破單一仿射函數的限制。

一眼檢查

$x = 2$ 時，$h = 5$，$\hat{y} = 11$。

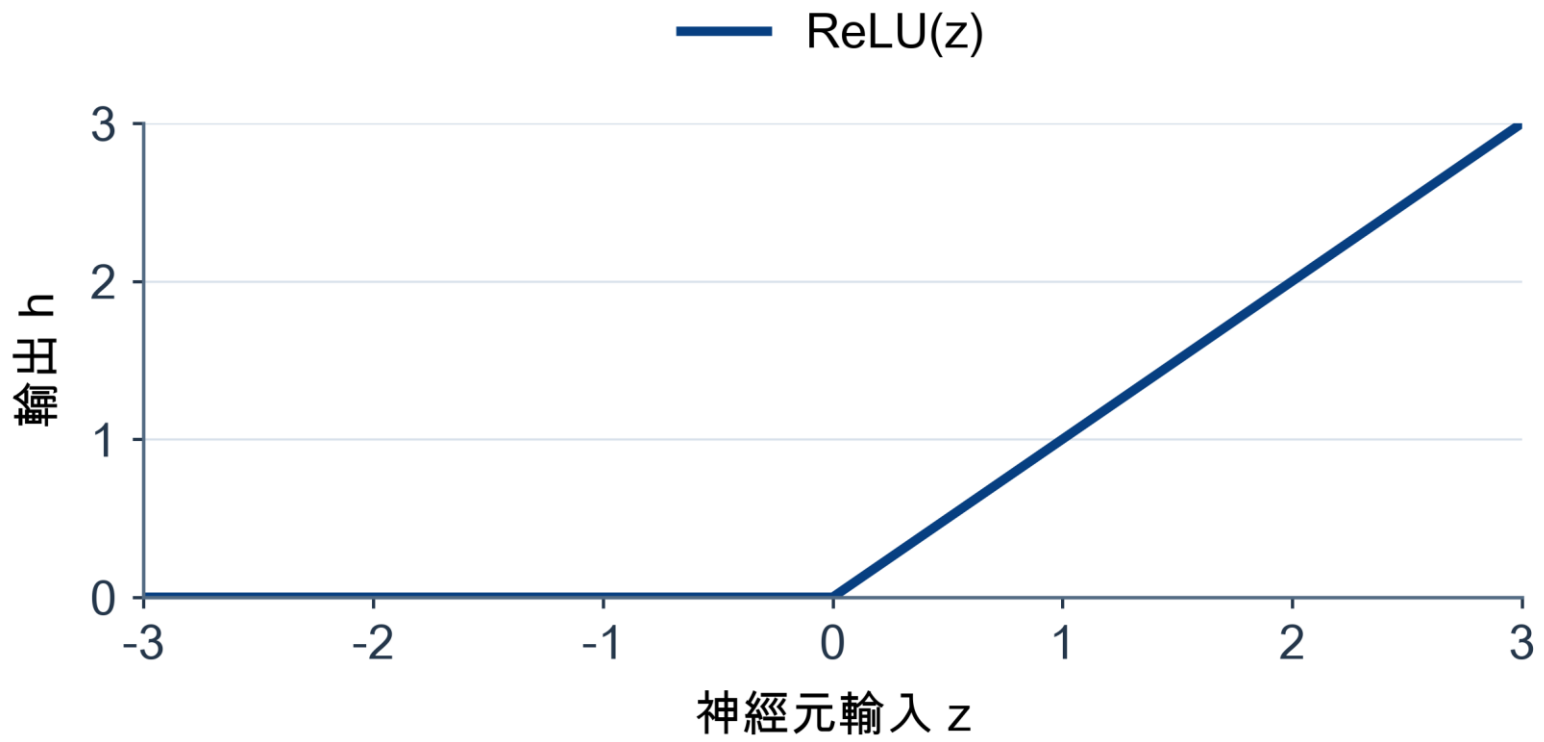
合併後 $6 \times 2 - 1 = 11$，結果相同。

兩層都能計算，卻沒有比一條直線更多的形狀能力。

需要在中間加入非線性轉換，稱為激活函數（activation function）。下一頁看 ReLU。

ReLU：把負值截成零

第一種激活函數：ReLU(z) = max(0, z)。z ≤ 0 輸出 0，z > 0 保留 z，於 0 產生轉折。



數字例子

輸入 −2、0、3，輸出依序是 0、0、3。正半邊是一條斜率為 1 的直線。

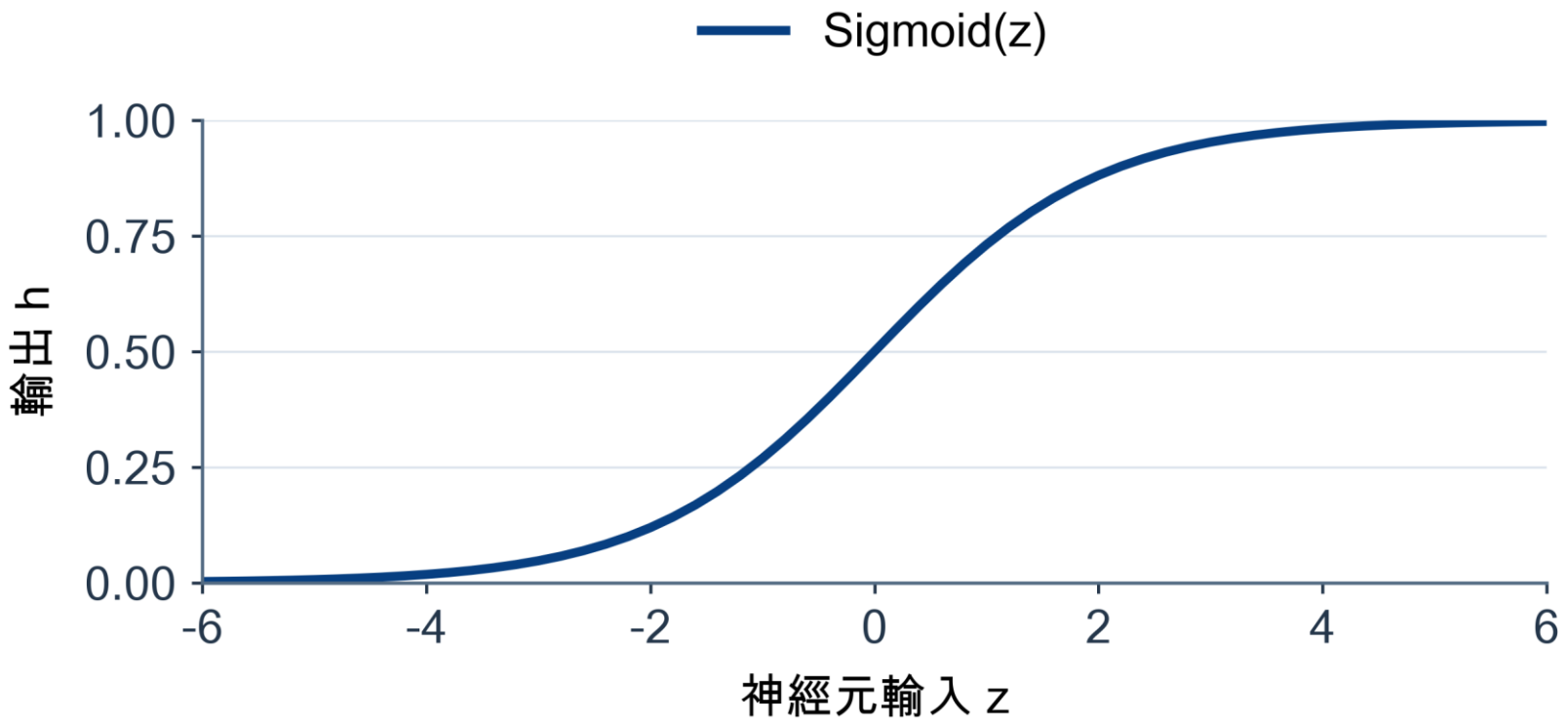
門檻的類比與限制

像超過門檻後才開始增加的計費。感應燈只有開關，而 ReLU 啟動後輸出還會持續增加。

ReLU 提供一個彎折，後面會用多個彎折拼出完整曲線。

Sigmoid：平滑地壓到 0 與 1 之間

另一種激活函數是平滑的 Sigmoid：σ(z) = 1 / (1 + exp(−z))。先比較形狀，再選本課用法。



輸出的用途

二元分類常用它把模型分數轉成 0 到 1 的數值；是否能解讀成可靠機率，還要看訓練與校準。

飽和區

曲線兩端變平，輸入變動對輸出的影響很小，梯度也小。很多層相乘時可能使學習變慢。

一個 Sigmoid 是平滑階梯，多個加權階梯也能組合出較複雜的形狀。

ReLU 與 Sigmoid 的選擇

激活函數會改變輸出範圍與梯度。這些差異會影響模型能表示的形狀，以及訓練的難易。

比較	ReLU	Sigmoid
輸出範圍	0 到正無窮	介於 0 與 1
形狀	一個明確彎折	平滑 S 形階梯
梯度	負半邊為 0，正半邊為 1	兩端接近 0
注意事項	整段輸入未啟動可能難以更新	深層相乘可能使梯度很小

今天用 ReLU 拼曲線，因為轉折與分段斜率容易直接看見。

一個片段的三個參數：w、b、c

片段 $g(x) = c \times \mathrm{ReLU}(wx + b)$。運算順序決定每個旋鈕的作用。

先算 $z = wx + b$

w 決定輸入變化的方向與尺度，b 改變 z 過零的位置。若 $w \neq 0$，轉折位置是 $x = -b/w$。

再算 $h = \mathrm{ReLU}(z)$

z 未超過零時，這個片段不貢獻輸出。啟動後才保留 z。

最後乘上 c

c 放大、縮小或翻轉片段貢獻。啟動區的斜率是 cw，而不是只看 w 或 c。

完整代入

令 $w = 2$，$b = -1$，$c = 3$。

$x = 1$：$z = 1$，$h = 1$，$g = 3$。

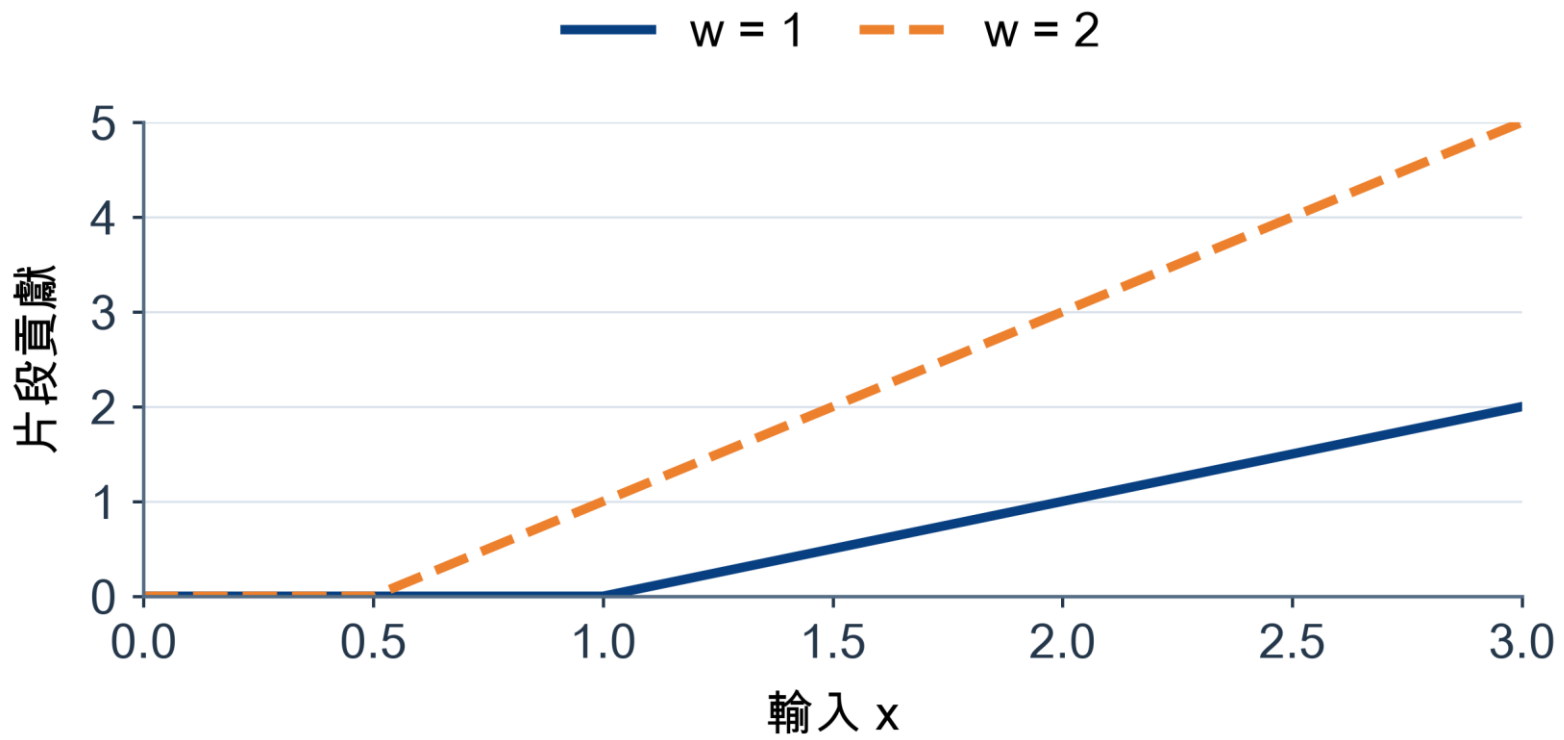
$x = 0$：$z = -1$，$h = 0$，$g = 0$。

轉折位置 $x = -(-1)/2 = 0.5$。

本頁只研究單一片段。稍後三個片段相加時，再加入調整整體高度的 d。

改 w：斜率與轉折都可能改變

固定 b = −1、c = 1，分別使用 w = 1 與 w = 2。



轉折位置

w = 1 時轉折在 x = 1；w = 2 時轉折在 x = 0.5。因為 x轉折 = −b/w。

啟動區斜率

斜率從 1 變成 2。若 w 改成負值，啟動側也可能換到左邊。不能只把 w 當作固定位置的轉角。

一次只改 w，觀察轉折位置與斜率兩件事。

B1 實驗：只修改 w

在 Notebook 的 B 區操作。先固定 b = −1、c = 1。執行兩次：w = 1 與 w = 2，每次都記錄轉折位置與啟動區斜率。

操作前預測

用 x轉折 = −b/w 計算，兩次轉折應分別在 1 與 0.5。

操作後解釋

線變陡的同時，轉折也往左移。這是同一個 w 同時影響位置與尺度的結果。

進階嘗試

把 w 改為 −1，觀察哪一側啟動。若 w = 0，z 就只剩 b，片段與 x 無關。

記錄格式

原本：w = 1，轉折 x = 1。

修改：w = 2，轉折 x = 0.5。

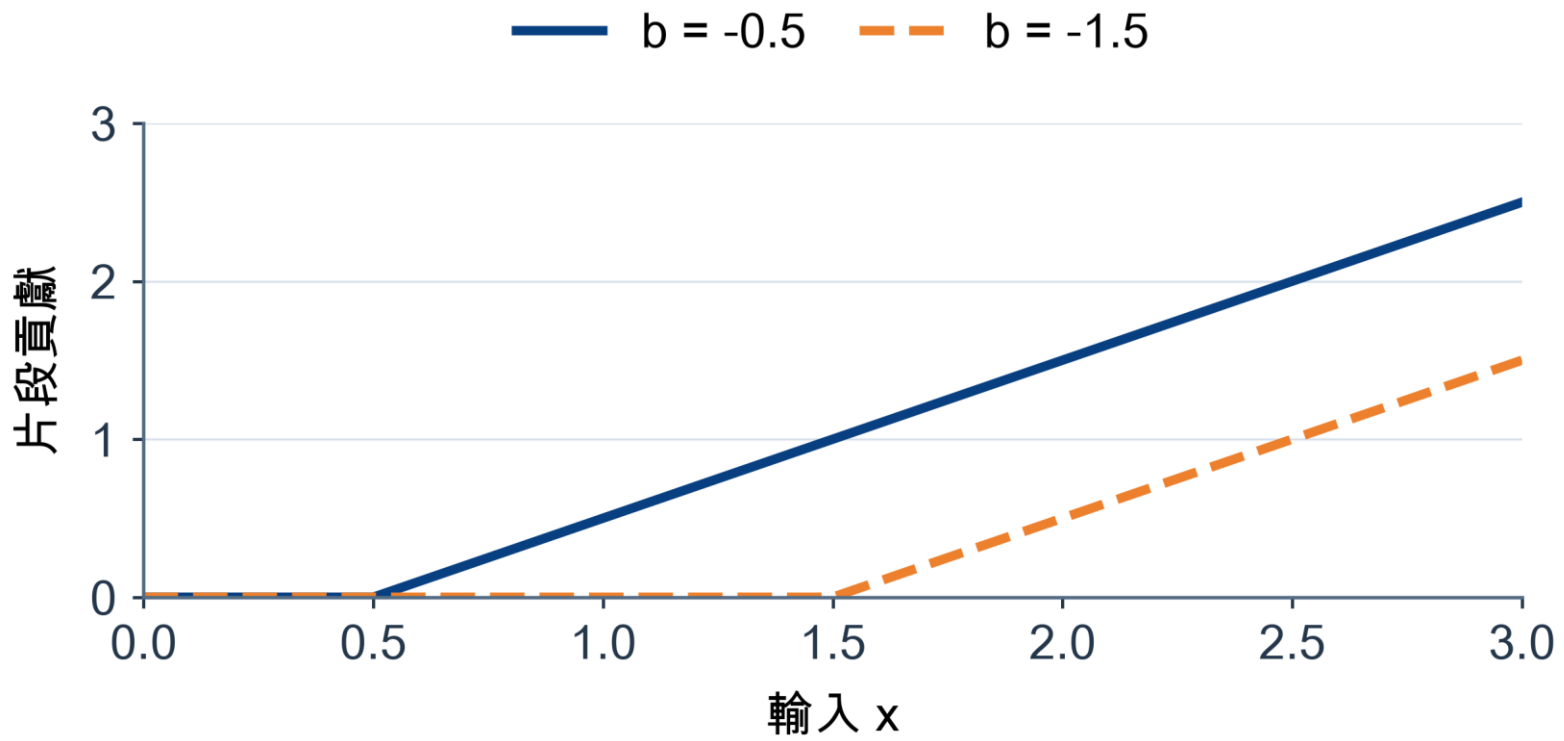
我的觀察：啟動區斜率從 1 增加到 2。

建議範圍：w = −5 到 5。

修改 B 格的 w 後，按左側執行鈕。此格不會改動 C 區三片段的參數。

改 b：轉折位置如何移動？

固定 w = 1、c = 1，將 b 從 −0.5 改成 −1.5，轉折由 x = 0.5 移到 1.5。



方向要依公式判斷

w > 0 時，b 變小會把轉折往右移。w < 0 時，方向相反。

不是整條往下移

未啟動區仍是 0。這與 y = b + wx 裡 b 讓整條直線平移的效果不同。

固定 w 時才能單獨解讀 b 的平移效果，w = 0 時公式 −b/w 不適用。

B2 實驗：只修改 b

在 Notebook 的 B 區操作。把 w 設回 1、c 設為 1，比較 b = −0.5 與 b = −1.5。

操作前預測

兩個轉折位置是 0.5 與 1.5，曲線啟動區的斜率都維持 1。

操作後解釋

b 控制神經元何時跨過 z = 0。在轉折左側，ReLU 仍為 0，不是整條曲線都下降。

先分清楚作用

改 b 會改變何時啟動。等到組合曲線時，會另外用 d 調整整體高度。

預測題

w = 2，想把轉折放在 x = 1。

由 b = −w × x轉折 得到 b = −2。

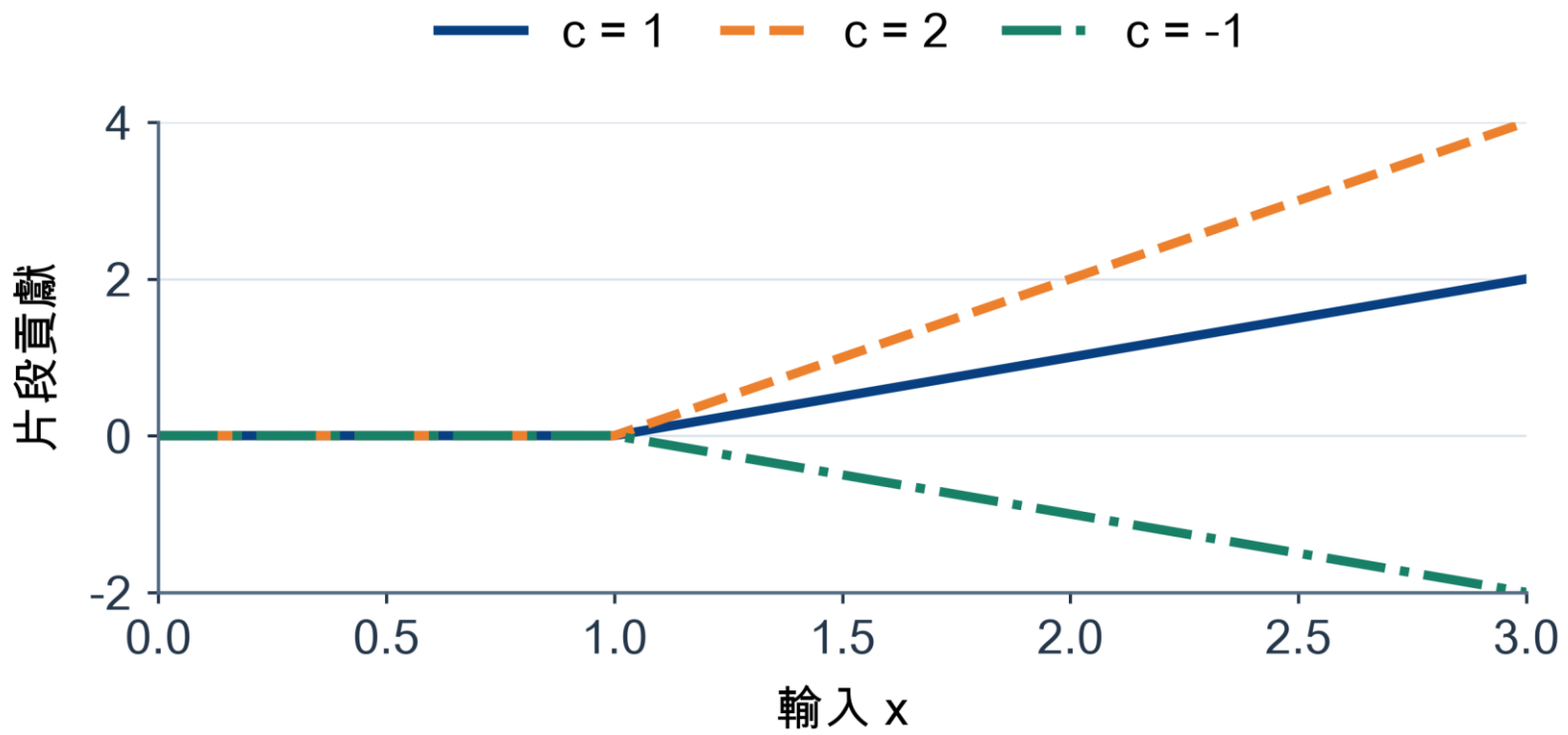
先算再跑，確認圖與預測相同。

建議範圍：b = −3 到 3。

先將 B 格 w、c 都設回 1，再改 b。每次修改後都重新執行這一格。

改 c：每個片段的貢獻大小

固定 w = 1、b = –1，比較 c = 1、2、–1。三個片段的轉折位置都在 x = 1。



正值放大

c = 2 把 ReLU 輸出放大兩倍，因此啟動區斜率也放大兩倍。

負值翻轉

ReLU 本身不輸出負值，但乘上負 c 後，片段的貢獻可以是負值，讓合成曲線往下轉。

先看 c 如何改變片段的方向與幅度。等到自動訓練，再討論它對梯度的影響。

B3 實驗：只修改 c

在 Notebook 的 B 區操作。固定 w = 1、b = −1，比較 c = 1、2、−1。請解釋曲線哪裡沒有變，哪裡改變。

不變的位置

轉折仍在 x = 1，因為判斷是否啟動只看 wx + b。

改變的部分

啟動區斜率 cw 分別為 1、2、−1。負 c 能讓片段往下，為山峰後的下降段提供貢獻。

連到組合曲線

若後半段太高，可嘗試讓對應片段 c 更負，但先確認這片段從正確位置開始啟動。

代入檢查

x = 2 時，ReLU(2 − 1) = 1。

c = 1、2、−1 時，貢獻依序是 1、2、−1。

建議範圍：c = −5 到 5。

B 格固定 w = 1、b = −1，再依序試 c = 1、2、−1，觀察方向與大小。

停下來想 ④：ReLU 不負，總貢獻也不負？

片段 g(x) = −2 × ReLU(x − 1)。它的轉折位置，以及 x = 2 時的片段貢獻，各是多少？



A 轉折在 1，g(2) = −2

B 轉折在 1，g(2) = +2

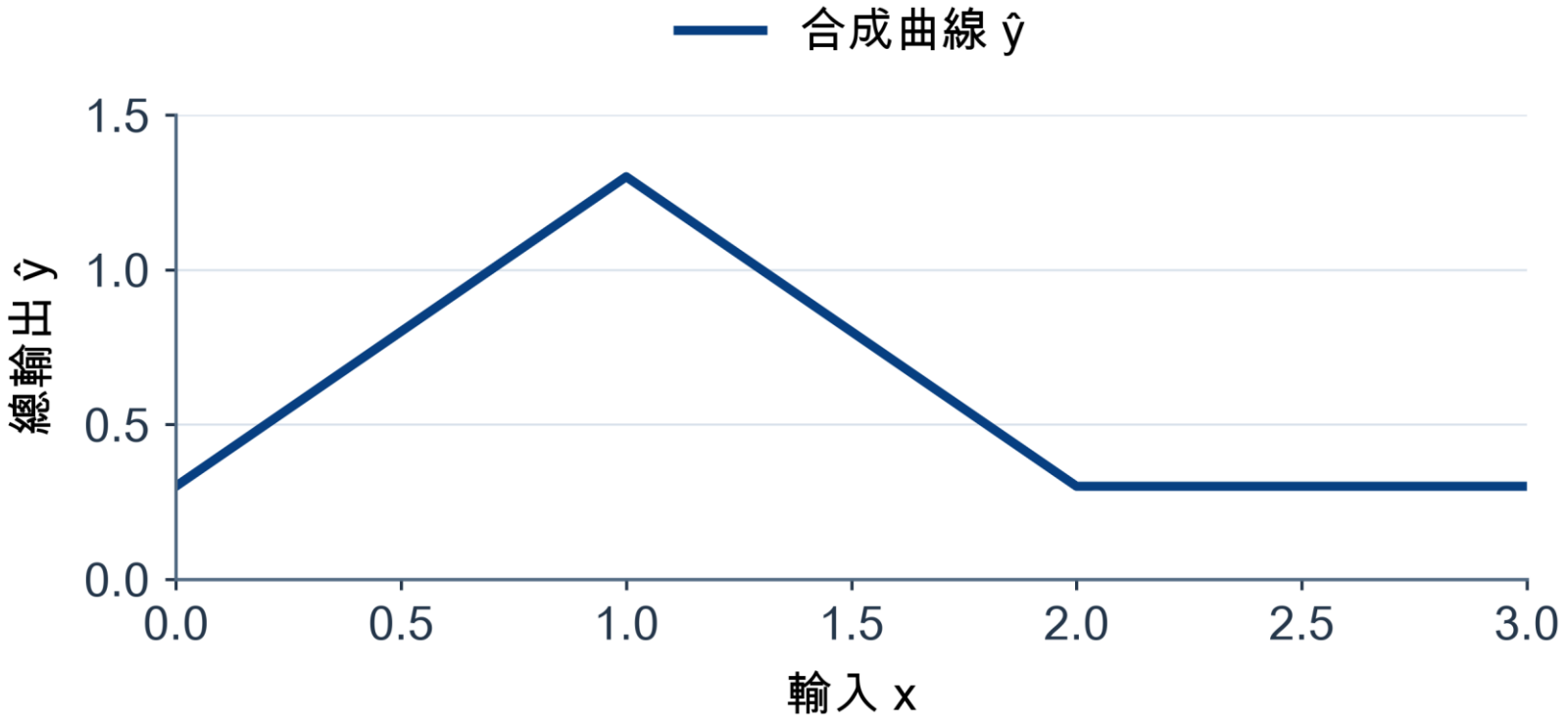
C 轉折在 −1，g(2) = −2

先看圖選答案，再用圖中的證據說明。下一頁揭曉。

先獨立想 45 秒，再和同桌交換理由。選答案之外，請說出你用了哪個觀念。

三個 ReLU 相加形成山形

B 已看懂單一片段。現在加總三個片段，再加整體高度 d：$\hat{y} = d + \Sigma\, c_i\, \mathrm{ReLU}(w_i x + b_i)$。



一組可理解的例子

d = 0.3；w = [1, 1, 1]；b = [0, −1, −2]；c = [1, −2, 1]。

圖中的三個區域

0 到 1 斜率為 1；1 到 2 斜率為 −1；2 之後斜率回到 0。負 c 讓曲線有機會轉向下降。

相加的是每個 x 的輸出值，不是把小圖首尾拼貼。

逐項加總：看得見的前向計算

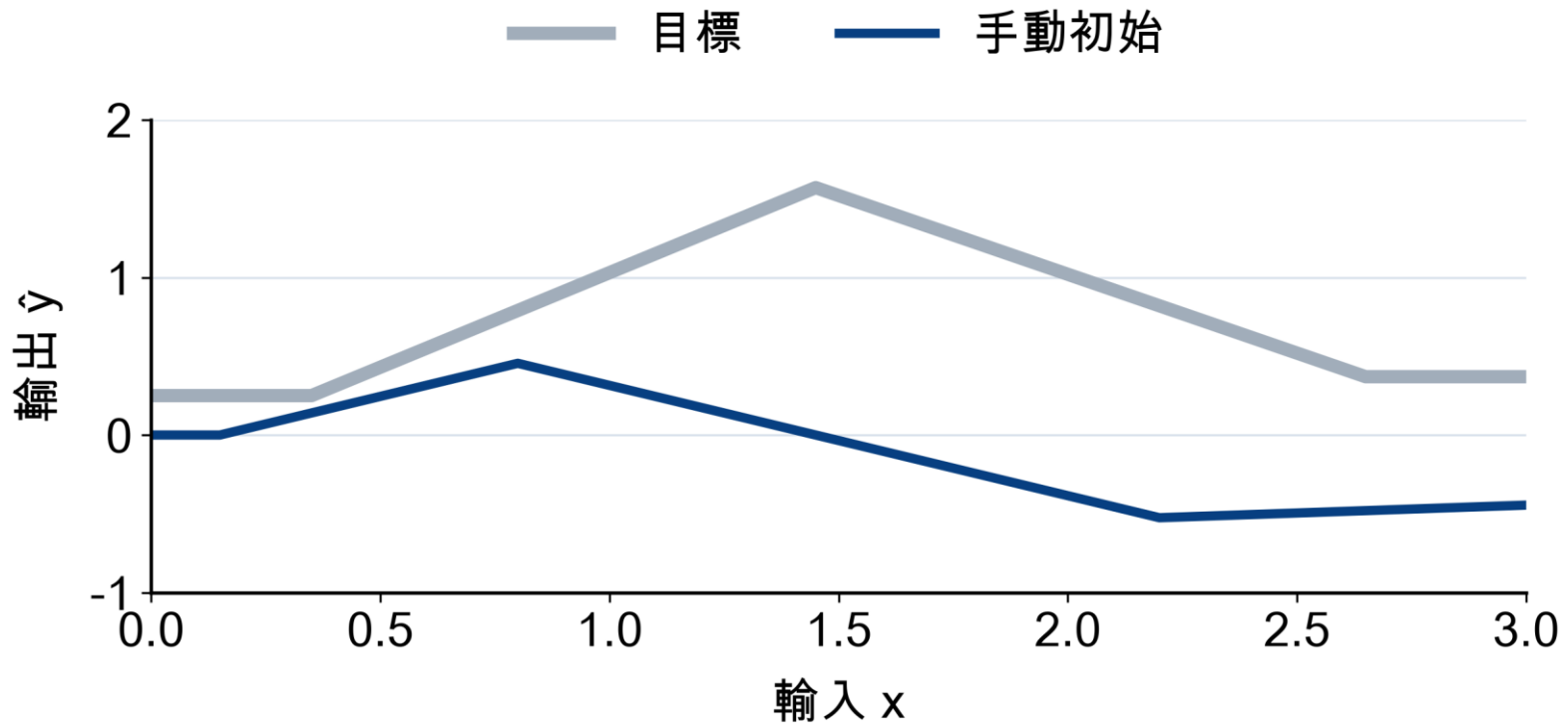
延續上一頁的山形模型。固定同一個 x，先計算每個片段，再加入 d = 0.3。

x	ReLU(x)	−2 ReLU(x−1)	ReLU(x−2)	總輸出 ŷ
0.5	0.5	0	0	0.8
1.0	1.0	0	0	1.3
1.5	1.5	−1.0	0	0.8
2.5	2.5	−3.0	0.5	0.3

x = 1.5：ŷ = 0.3 + 1.5 − 1.0 + 0 = 0.8。

C 關卡：三片段手動對齊

現在到 Notebook 的 C 區。這一關的目標與前面山形算例不同，請依圖形找出調整方向。



先看哪裡不同

灰線是本關目標，藍線是 C 的初始結果。先調 d 的高度，再觀察 b 的轉折與 c 的貢獻。

本關完成條件

保留基準與三次單項修改，至少一次讓 MSE 降低。每次執行 C 後都看紀錄，E 留到最後。

目標資料是教學合成曲線。對齊的是輸出形狀，參數可以有不同的解。

C 操作方法：找到要修改的那個數字

C 格的 w、b、c 各有三個數值，位置依序對應第一、第二、第三片段。

列表位置與索引

Python 從 0 開始編號。b[1] 指 b 列表中間的數字，也就是第二片段的偏差。

只替換一個數值

例如 b 原本是 [−0.15, −0.8, −2.2]。只把中間改成 −1.0，其餘數值保持上次結果。

填寫預期再執行

change_note 寫「第二片段 b 變小，預期轉折右移」。重跑 C 格，讀取本次 MSE 與實驗紀錄。

十個參數的配置

w：三個輸入權重。

b：三個片段偏差。

c：三個輸出係數。

d：一個整體高度。

B 區與 C 區各自使用自己的參數。改完 B 不會自動替你更新 C。

C 實驗紀錄：預期與實測分開

以下是記錄格式，修改方向先自己預測。實際 MSE 由 Notebook 計算。

回合	只改一項	預期圖形變化	修改前 / 後 MSE
基準	原始設定	先找出曲線差異	執行 C 格後讀值
1	d：0.0 改 0.25	整條往上 0.25	填入前次與本次 MSE
2	b[1]：−0.8 改 −1.0	w[1]=1 時轉折右移	依實際結果判斷
3	c[1]：−1.4 改 −1.6	第二片段向下更強	依實際結果判斷

格式範例中的數字為示意；請在 Notebook 表格填入自己的實際 MSE。

卡關時用圖形診斷參數

先確認錯誤發生的位置，再選擇會影響該位置的參數。修改無效也能提供線索。

看到的現象	先檢查	原因
整條都偏高或偏低	輸出偏差 d	它影響所有輸入
轉折位置不對	$-b_i / w_i$	這是片段啟動邊界
轉折對了，斜率不對	$c_i w_i$ 與其他片段	總斜率是貢獻的和
調 w 或 b 幾乎無效	ReLU 是否全關，c 是否為零	傳回的梯度可能為零

把 b 調到所有輸入都未啟動時，先恢復可啟動範圍，才有機會學到片段。

C 檢查點：保存手動實驗的證據

完成三次單項修改後，先執行 C2 查看紀錄，再進入模型解釋。

紀錄包含什麼

表格列出基準與每次修改的說明、MSE、參數快照。檢查最後一筆是否對應目前 C 格設定。

圖與數字一起解釋

選一次 MSE 降低的修改，指出曲線哪裡更接近。若某次變差，說明你的預期在哪裡落空。

保存可重讀的內容

先等待 Notebook 儲存輸出。若重新啟動執行環境，記憶體中的紀錄會清除，請保留已存的表格。

C2 反思欄位

最有效的修改是什麼？

為什麼它會改變這一段曲線？

下一步想調哪個參數？

先寫下答案，暫時不執行 E。

每次修改後都執行 C，才能留下當次紀錄。一次改多個數值較難判斷原因。

D 揭露：你剛才調整的就是神經網路

C 的三個 ReLU 片段可以寫成一個三神經元隱藏層。剛才你用手動方式選擇了它的參數。

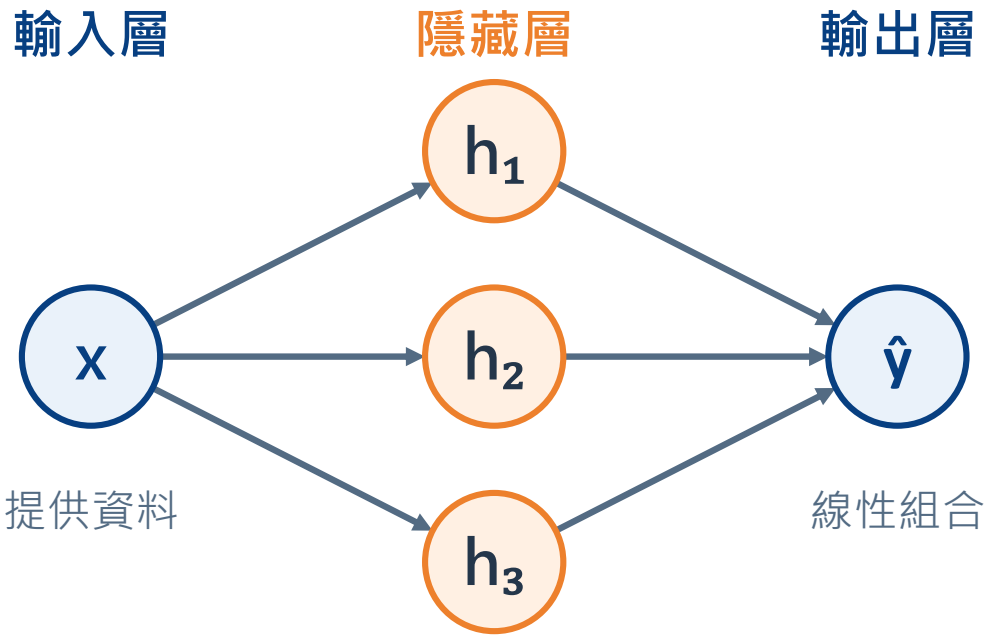
剛才看到的物件	神經網路中的名稱	計算方式
輸入座標 x	輸入層	將同一個 x 送到三個單元
每個轉折片段	隱藏神經元	$h_i = \mathrm{ReLU}(w_i x + b_i)$
三段加總與高度	線性輸出層	$\hat{y} = d + \Sigma\, c_i h_i$
你調整的十個數字	模型參數	3 個 w、3 個 b、3 個 c、1 個 d

接下來保留同一模型，由梯度下降代替手動調參數，觀察訓練帶來的變化。

隱藏層與輸出層的分工

前饋網路：訊息由輸入經隱藏層傳到輸出，連線不形成循環。

位置	運算與意義	參數數量
輸入 x	提供一個座標值	0
隱藏層 3 個神經元	$h_i = \mathrm{ReLU}(w_i x + b_i)$	3 個 w 3 個 b
輸出層 1 個神經元	$\hat{y} = d + c_1 h_1 + c_2 h_2 + c_3 h_3$	3 個 c 1 個 d
合計	用 10 個參數表示一條曲線	10



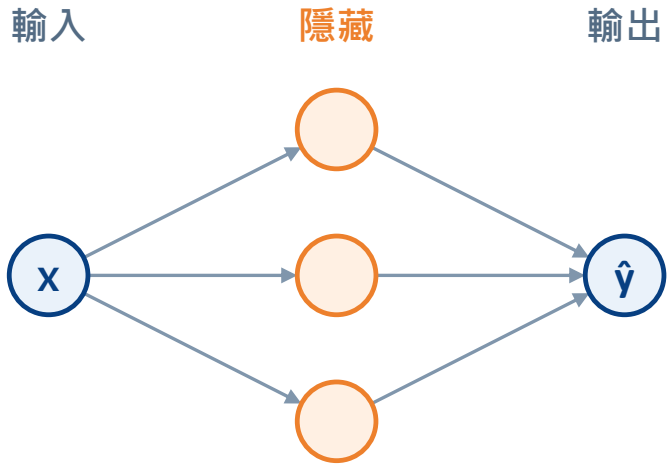
含輸入共 3 層；運算深度 2、隱藏層寬度 3

「隱藏」表示內部的中間結果；資料只給最終答案，未指定每個 h_i 應是多少。

深度、寬度與表示能力

寬度：同一隱藏層的神經元數。深度：串接的運算層數（本課不計輸入層）。

原本網路

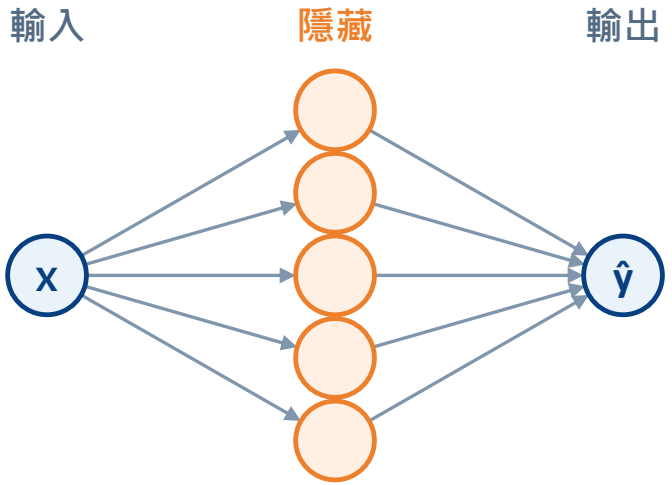


1–3–1

寬度 3　深度 2

3 個片段，合成一個輸出

加寬：同一層增加單元

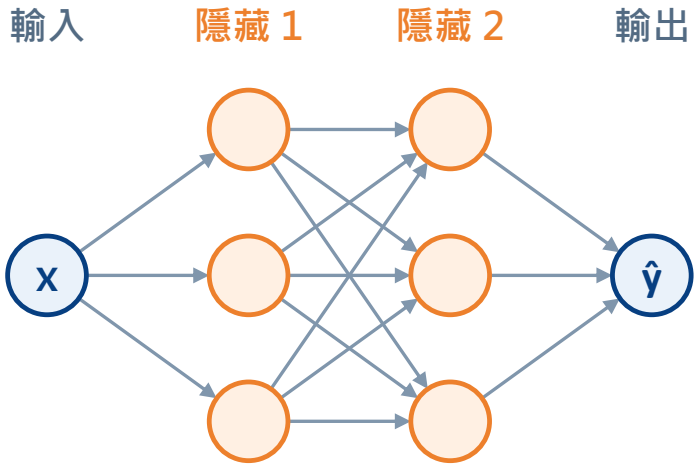


1–5–1

寬度 5　深度 2

更多片段，同一層一起組合

加深：多接一個隱藏層



1–3–3–1

各隱藏層寬度 3　深度 3

先組合一次，再交給下一層

更寬或更深會改變可表示的函數，但不保證訓練成功，也不保證新資料更準。

NumPy 如何一次計算整批資料

先把 P64 的一筆輸入，擴展成三筆資料。每一筆都使用同一組模型參數。

```
import numpy as np
x = np.array([0., 1., 2.])
w = np.array([1., -1., 2.])
b = np.array([0., 1., -1.])
c = np.array([2., -1., 0.5])
d = 0.5
```

資料與參數分開看

x 裡的 3 個值代表 3 筆資料。
w、b、c 各有 3 個值，對應 3 個隱藏神經元。

先理解每一格的意義

中間的 h 是 3 列 × 3 欄。
每列是一筆資料，每欄是一個神經元。

本段操作

開啟「NumPy 與反向傳播」補充 Colab，依 N1–N10 順序執行，先預測輸出再按執行。

3 筆資料與 3 個神經元剛好同數量，意義不同；後面會改用 4 筆來檢查。

陣列的 shape、ndim 與 dtype

shape 說明各軸有多少元素；ndim 說明有幾個軸。先確認資料長什麼樣。

建立方式	shape / ndim	意義
np.array(2.0)	() / 0	純量：一個數
np.array([0., 1., 2.])	(3,) / 1	一維陣列：3 個值，未指定列或欄
np.array([[0.], [1.], [2.]])	(3, 1) / 2	二維陣列：3 列、1 欄
np.array([[0., 1., 2.]])	(1, 3) / 2	二維陣列：1 列、3 欄

Python： [1, 2] * 2 得到 [1, 2, 1, 2]
NumPy： np.array([1, 2]) * 2 得到 [2, 4]

模型參數用 dtype=float，才能保留小數更新；(3,) 和 (3, 1) 不是同一種形狀。

索引與切片：選一格、一列或一欄

以下固定 H = [[0, 1, 0], [1, 0, 1], [2, 0, 3]]。索引從 0 開始。

語法	結果	為什麼
H[1, 2]	1	第 2 列、第 3 欄
H[0, :]	[0, 1, 0]，shape (3,)	選第 1 筆，保留全部神經元
H[:, 1]	[1, 0, 0]，shape (3,)	選所有資料的第 2 個神經元
H[:, 1:2]	[[1], [0], [0]]，shape (3,1)	切片保留欄軸；終點 2 不包含在內
H[:2, :]	前 2 列，shape (2,3)	冒號省略起點表示從 0 開始

基本切片通常共用原資料；要獨立修改，使用 H[:2, :].copy()。

x[:, None]：新增一個軸，不是新增資料

把 x 的每筆資料排成一列，才方便讓每筆資料分別送進三個神經元。

x，shape (3,)	x[:, None]，shape (3,1)	x[None, :]，shape (1,3)
0 1 2	0 1 2	0 1 2
只有一個軸 不是一列或一欄的二維表	每筆資料占一列 新增欄軸，長度為 1	所有值放在同一列 新增列軸，長度為 1

```
x.reshape(-1, 1)  # same as x[:, None]
```

一維 x 的 x.T 仍是 (3,)；要變成一欄，使用 x[:, None] 或 x.reshape(-1, 1)。

廣播：每筆 x 都套用同一組 w 與 b

z = x[:, None] * w + b。以下每列是一筆資料，每欄是一個神經元。

x[:, None]		w = [1, −1, 2]				b = [0, 1, −1]		
0								
1	×	1	-1	2	+	0	1	-1
2								
(3,1)								

每筆都套用同一組權重與偏差

以 x=2 為例：
[2×1+0, 2×(−1)+1, 2×2−1]
= [2, −1, 3]

x	z_1	z_2	z_3
0	0	1	−1
1	1	0	1
2	2	−1	3

廣播由最右軸比對：長度相同或其中一方為 1 才相容；缺少的軸視為 1。

np.maximum：讓每個神經元通過 ReLU

h = np.maximum(0, z) 會逐格比較 0 與 z，保留較大的值；形狀不變。

z：加權後的值

0	1	−1
1	0	1
2	−1	3



h：ReLU 後的值

0	1	0
1	0	1
2	0	3

```
h = np.maximum(0, z)
```

負值變 0，正值保留；兩邊都是 (3,3)。

np.maximum(0, z) 逐格比較；np.max(z) 是取整個陣列的最大值，兩者用途不同。

* 與 @：乘完後，要不要沿神經元加總？

固定 c = [2, −1, 0.5]。h * c 保留各片段貢獻，h @ c 得到每筆資料的加權總和。

h			h * c：各片段貢獻			h @ c：每列加總
0	1	0	0	−1	0	−1
1	0	1	2	0	0.5	2.5
2	0	3	4	0	1.5	5.5

第三筆：2×2 + 0×(−1) + 3×0.5 = 5.5

加上 d=0.5：y_hat = [−0.5, 3, 6]，shape (3,)

本例 h @ c 等於 np.sum(h * c, axis=1)；再加 d=0.5，才得到最後的預測。

axis、sum 與 mean：沿哪個方向彙整？

同一個 h，沿不同軸加總會回答不同問題。axis 指定被彙整掉的軸。

語法	結果 / shape	回答的問題
h.sum(axis=1)	[1, 2, 5] / (3,)	每筆資料的三個單元加總
h.sum(axis=0)	[3, 1, 4] / (3,)	每個單元跨三筆資料加總
h.sum(axis=1, keepdims=True)	[[1], [2], [5]] / (3,1)	每筆加總後，保留欄軸
np.mean((y_hat - y) ** 2)	0.75 / 純量	三筆平方誤差取平均

y_hat = [−0.5, 3, 6]，y = [0, 2, 5]
e = [−0.5, 1, 1]，MSE = (0.25 + 1 + 1) / 3 = 0.75

**** 2 才是平方；本課每筆一個輸出，mean 未指定 axis 時平均全部元素。**

換成多個特徵、多個輸出時怎麼寫？

N 是資料筆數，D 是每筆特徵數，K 是隱藏單元數，M 是輸出數。

情況	資料與參數形狀	運算與結果
一個特徵、整批資料	x:(N,)　w,b:(K,)	x[:,None]*w+b 得 (N,K)
D 個特徵、整批資料	X:(N,D)　W:(D,K)　b:(K,)	X @ W + b 得 (N,K)
只取一筆，保留批次軸	X[i:i+1,:] 是 (1,D)	X[i:i+1,:] @ W + b 得 (1,K)
每筆輸出一個值	H:(N,K)　c:(K,)	H @ c + d 得 (N,)
每筆輸出 M 個值	H:(N,K)　C:(K,M)　d:(M,)	H @ C + d 得 (N,M)

先決定每個軸的意義，再選運算；reshape 可以改形狀，不能修正錯誤的資料意義。

沒有報錯，也可能算錯 Loss

若預測是 (3,1)，答案卻是 (3,)，相減會廣播成 (3,3)，變成每筆互相比較。

y_hat，shape (3,1)
1
2
3

y，shape (3,)		
1	2	3

錯誤殘差，shape (3,3)		
0	−1	−2
1	0	−1
2	1	0

本來三筆都完全正確，Loss 應為 0。
錯誤廣播卻得到 (1+4+1+1+4+1)/9 = 4/3。

```
y_hat = y_hat.reshape(-1)  # one output per sample only
```

計算 Loss 前先檢查：assert y_hat.shape == y.shape；不要靠結果看起來像分數就相信它。

看圖問答：一次得到兩筆預測

H 有 2 列、3 欄，每列是一筆資料；c 有 3 個輸出權重。哪個選項直接得到兩筆加權總和？

H，shape (2,3)

1	0	2
0	3	1

c，shape (3,)

2	−1	0.5

A　H * c

B　H @ c

C　np.sum(H * c)

哪個結果保留「兩筆資料」，又把「三個神經元」加總？

先獨立選，再向同桌解釋：另外兩個選項各算出了什麼？

手動調參之後，電腦還缺什麼？

我們已經能算出 Loss。接下來要知道：每個參數稍微改動，Loss 會往哪裡變？

工作	得到什麼	還沒做什麼
前向計算	目前的預測與 Loss	還不知道各參數的變化率
反向傳播	每個參數對 Loss 的梯度	參數本身尚未更新
梯度下降更新	新參數 = 舊參數 − η × 梯度	還要重新前向，才能知道新 Loss

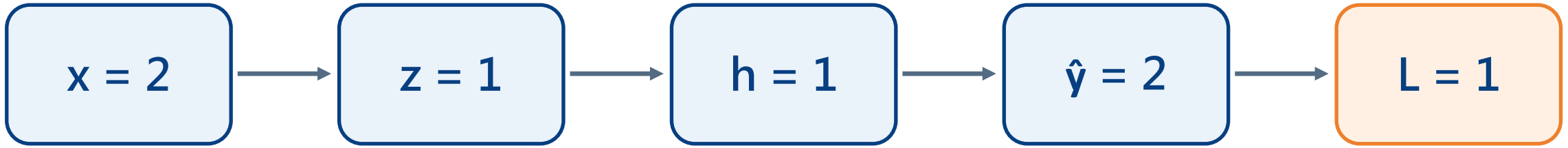
也可以逐一微調參數、重算 Loss 來估梯度。
反向傳播利用鏈鎖律，共用中間結果，避免每個參數都從頭試算。

先回答「梯度怎麼來」，再套用前面已學過的梯度下降；兩個步驟分開理解。

前向計算：先用一個神經元算到底

為了看清每一步，暫用 1 個隱藏神經元、1 筆資料：x=2、答案 y=1。

固定參數：w=1、b=−1、c=2、d=0



輸入資料　1×2−1　max(0,1)　2×1+0　(2−1)²

輸入與答案保持不變；接下來追蹤的是 w、b、c、d 如何影響 L。

先保存 z、h、ŷ 等中間結果；同一輪反向傳播會用到它們。

從 Loss 開始：預測多一點，誤差怎麼變？

目前 $\hat{y}=2$、$y=1$，$L=(\hat{y}-y)^2$。先固定其他量，只看 Loss 對預測值的敏感度。

附近的預測 $\hat{y}$	平方損失 L	觀察
1.99	0.9801	預測少一點，Loss 下降
2.00	1.0000	目前位置
2.01	1.0201	預測多一點，Loss 上升

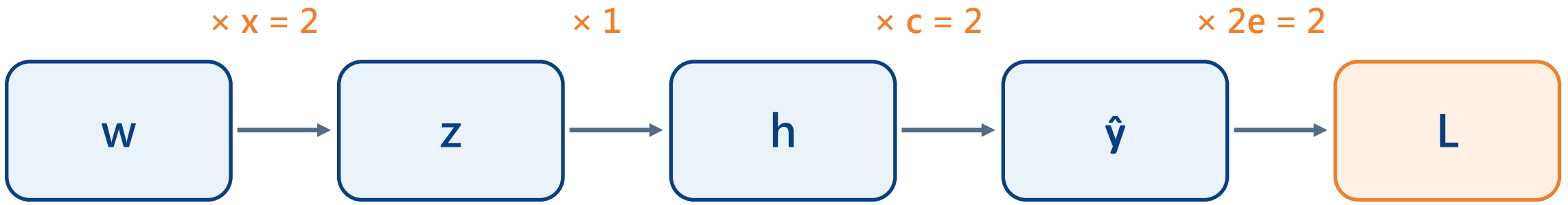
中央差分：(1.0201 – 0.9801) / 0.02 = 2

小幅改動時：$\Delta L \approx 2 \times \Delta\hat{y}$；這是變化率，不是新參數。

$\partial L/\partial\hat{y} = 2(\hat{y}-y) = 2$；正號表示附近增加預測會提高 Loss。

鏈鎖律：沿一條路徑，把局部影響相乘

w 並沒有直接出現在 Loss 的最外層；它先改變 z、h、ŷ，最後才影響 Loss。



待調參數　　$z = wx+b$　　$h = \mathrm{ReLU}(z)$　　$\hat{y} = ch+d$　　$L = (\hat{y}-y)^2$

$\partial L/\partial w = 8$　　$\partial L/\partial z = 4$　　$\partial L/\partial h = 4$　　$\partial \hat{y} / \partial h = c = 2$　　$\partial L/\partial \hat{y} = 2$

$(\partial L/\partial d = 2)$

$$\partial L/\partial w = (\partial L/\partial \hat{y}) \times (\partial \hat{y}/\partial h) \times (\partial h/\partial z) \times (\partial z/\partial w) = 2\times2\times1\times2 = 8$$

反向傳播：逐步算出四個參數的梯度

所有梯度都使用剛才那一組舊參數與中間值；先算完，再一起更新。

回傳到哪裡	計算	結果與解讀
輸出 $\hat{y}$	$g = \partial L/\partial \hat{y} = 2(2-1)$	$g = 2$
輸出權重 c、偏差 d	$\partial L/\partial c = g \times h$；$\partial L/\partial d = g$	c 梯度 2；d 梯度 2
隱藏輸出 h、輸入 z	$\partial L/\partial h = g \times c$；$\partial L/\partial z = g \times c \times 1$	h 梯度 4；z 梯度 4
輸入權重 w、偏差 b	$\partial L/\partial w = 4 \times x$；$\partial L/\partial b = 4 \times 1$	w 梯度 8；b 梯度 4

若 $z<0$，ReLU 的局部斜率為 0，此筆資料傳回 w、b 的梯度為 0。
同時 $h=0$，c 的梯度也為 0；輸出偏差 d 仍可能更新。

反向傳播回傳的是梯度，不是把正確答案 y 倒著塞回每個神經元。

梯度下降：真的改一次，再重算 Loss

學習率 η=0.01。這一步由最佳化器執行，反向傳播只負責提供剛才算出的梯度。

參數	舊值	梯度	新值
w	1	8	1−0.01×8 = 0.92
b	−1	4	−1−0.01×4 = −1.04
c	2	2	2−0.01×2 = 1.98
d	0	2	0−0.01×2 = −0.02

重新前向

$z = 0.92 \times 2 - 1.04 = 0.8$
$h = 0.8$
$\hat{y} = 1.98 \times 0.8 - 0.02$
$= 1.564$
$L = (1.564 - 1)^2$
$= 0.318096$

更新用同一組舊梯度；新 Loss 要用新參數重新算。

本例 Loss 從 1 降到 0.318096；每次更新後要重新前向，不能沿用舊的中間值。

回到三個隱藏神經元：同一訊號分到三條路

以下換回 1–3–1 模型，固定 x=2、y=0、d=0。各路都接收同一個輸出梯度 g。

單元	w_j	b_j	z_j	h_j	c_j	$\partial L/\partial w_j$	$\partial L/\partial b_j$	$\partial L/\partial c_j$
1	1	–1	1	1	2	8	4	2
2	0.5	0	1	1	–1	–4	–2	2
3	–1	1	–1	0	1	0	0	0

$\hat{y} = 2\times1 - 1\times1 + 1\times0 = 1$; $g = 2(1-0) = 2$

Hint: $\hat{y} = c_1h_1+c_2h_2+c_3h_3+d$, $g = \partial L/\partial\hat{y} = 2(\hat{y}-y)$

回到 z 的三個梯度：[4, –2, 0]；輸出偏差的梯度 $\partial L/\partial d = 2$。

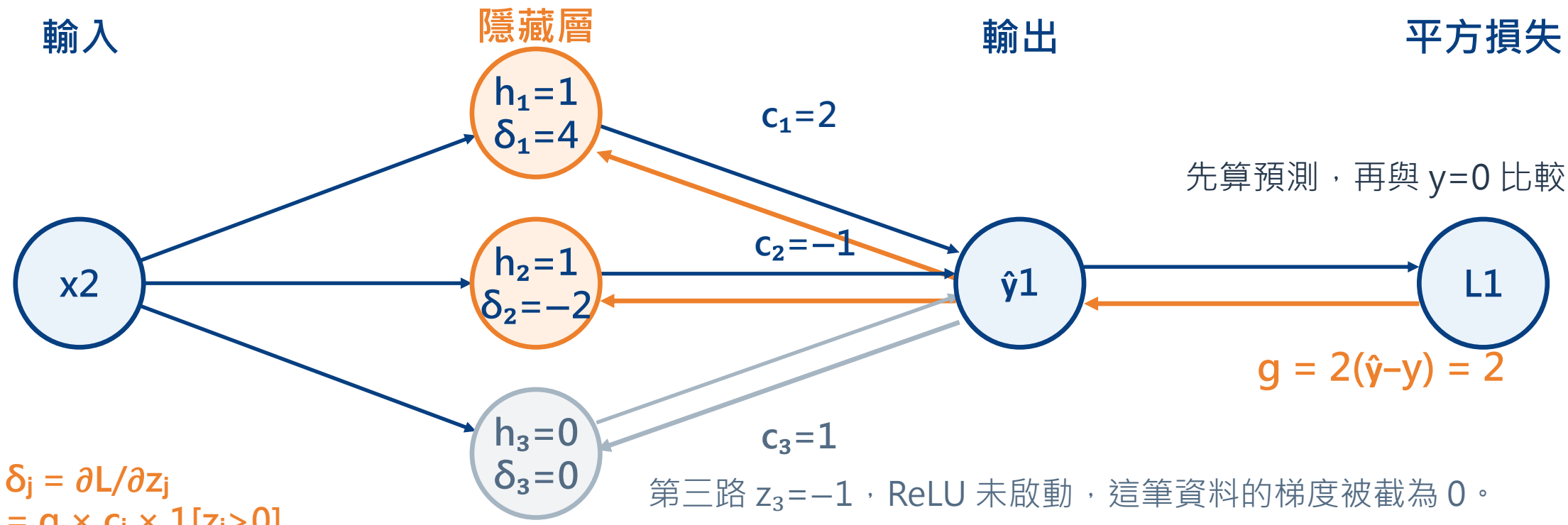
$$\frac{\partial L}{\partial z_j} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial h_j}\frac{\partial h_j}{\partial z_j} = g * c_j * 1$$

$$\frac{\partial L}{\partial d} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial d} = g * 1$$

每條路先乘自己的 c_j 與 ReLU 斜率；某一條路未啟動，不代表全部參數都停止。

把反向傳播畫回神經元網路

沿用上一頁：x=2、y=0、ŷ=1。藍色向右算預測，橘色向左傳回梯度。



回到參數：∂L/∂w = x*δ = [8, −4, 0]；∂L/∂b = δ = [4, −2, 0]

反向回傳的是 Loss 的敏感度；算完全部梯度後，才用梯度下降一起更新參數。

訓練迴圈：先算完梯度，再更新所有參數

同一輪的 z、h、y_hat 與梯度必須來自同一組參數。每次迭代都重新計算。

```
for step in range(steps):
    z = x[:, None] * w + b
    h = np.maximum(0, z)
    y_hat = h @ c + d
    # compute all gradients first
    dw, db, dc, dd = gradients(
        x, y, z, h, y_hat, c)
    w = w - lr * dw
    b = b - lr * db
    c = c - lr * dc
    d = d - lr * dd
```

前向

產生目前預測，計算並記錄 Loss。
初始化只做一次，不放在迴圈裡。

反向

先完成全部梯度，才開始改參數。
本課自己寫 gradients，NumPy 不會自動求導。

更新後檢查

必要時重新前向取得更新後 Loss。
比較曲線與分數，檢查是否有 NaN / Inf。

只算梯度不更新，模型不變；只更新卻不重新前向，看到的仍是舊預測。

實操：先驗證梯度，再讓模型接手

補充 Colab 的 B1–B3 與原課程 E 相互對照；保留「學生先手動、最後演算法接手」的順序。

操作	要觀察什麼	檢查點
B1：單神經元更新一步	先預測四個更新方向，再執行	梯度 8、4、2、2；新 Loss 0.318096
B2：三神經元梯度檢查	反向梯度與中央差分比較	避開 ReLU 折點；誤差應非常小
B3：訓練 200 步	比較起點與終點 Loss / 參數	從同一個起點，比較不同學習率
回到原 Colab E	演算法接手學生凍結的起點	不偷偷換資料、不重抽較容易的起點

中央差分：$g \approx [L(\theta+\varepsilon) - L(\theta-\varepsilon)] / (2\varepsilon)$
這是檢查自己有沒有寫錯梯度的方法，不是本課主要訓練方式。

先驗證「梯度算對了」，再討論「學習率是否合適」；Loss 下降仍不代表泛化成功。

看圖問答：算完梯度，模型改了嗎？

前向得到 ŷ=2、Loss=1；反向已算出四個梯度，但「更新參數」尚未執行。

前向計算	反向傳播	更新參數
ŷ=2，Loss=1 →	梯度：8、4、2、2 →	θ 新 = θ 舊 − ηg
已完成	已完成	尚未執行

A　預測仍是 2

B　預測變成 1.564

C　預測變成答案 1

先選 A、B 或 C，再說明圖中的哪個步驟真的會改變參數。

能力不足與最佳化不良

訓練 Loss 很高時，至少有兩種可能。模型畫不出目標，或模型畫得出但訓練還沒找到。

能力不足

拿一條直線去貼山形，增加訓練次數也不會讓直線出現轉折。可考慮非線性特徵或不同模型。

最佳化不良

三個 ReLU 可以形成山形，但學習率不合適、初始化全關閉或訓練步數不足，都可能卡住。

診斷方法

先用已知能表達目標的合成例子檢查程式，再看不同初始化與步長是否能改善。

不要只開一張處方

「Loss 太高就加神經元」不一定有效。

如果問題是所有 ReLU 都沒啟動，先調初始化更直接。

也要排除資料錯誤、標籤雜訊與前處理問題。

訓練資料上的問題先分清楚。即使訓練已經很準，下一步仍要檢查未見資料。

多筆資料，如何學到同一組參數？

前面固定一筆 (x,y)，是為了看清更新。真正訓練時，所有樣本共用同一組 θ=(w,b,c,d)。

簡化例：$\hat{y} = wx$，先固定 b=0

x	答案 y	w=1 的 $\hat{y}$	w=2 的 $\hat{y}$
1	2	1	2
2	5	2	4
3	5	3	6
MSE		14/3≈4.667	2/3≈0.667

本例的最小值：w*=27/14≈1.929
最小 MSE=9/14≈0.643，仍不會每筆都零誤差。

① 每筆各算預測與平方損失

$\hat{y}_i = f_\theta(x_i)$；$\ell_i = (\hat{y}_i - y_i)^2$

② 把所有樣本合成一個目標

$L(\theta) = (1/N) \sum_{i=1}^{N} \ell_i$
尋找讓這個平均 Loss 最小的同一組 θ。

③ 平均各筆梯度，再共同更新

$\nabla L = (1/N) \sum_{i=1}^{N} \nabla \ell_i$
θ 新 = θ 舊 − η∇L；重複前向、反向、更新。

神經網路可尋找較低 Loss，但不保證找到全域最小值。

訓練資料的平均 Loss 很低，遇到沒看過的新 x 還準嗎？下一頁用驗證資料來檢查。

訓練、驗證、測試資料的角色

訓練 Loss 低，只代表做過的題目更準。要檢查新資料，就要把資料分成不同用途。

資料	用途	生活比喻
訓練 train	用來算梯度、更新參數	平時練習題
驗證 validation	選超參數、模型與停止時機	模擬考，幫助選策略
測試 test	選定策略後做獨立最終評估	保留的正式驗收題

70 / 15 / 15 是示例比例；真正切法要配合資料量、時間與群組結構。

資料要怎麼切，取決於怎麼使用

隨機切分適合某些近似獨立同分布的樣本。時間資料或同一人的重複紀錄，常需要其他方式。

按時間切

用過去訓練、較晚資料驗證或測試，模擬未來預測。不能把未來資訊混入過去的決策。

按群組切

同一顧客、病人或設備的相近紀錄放在同一側，避免模型靠辨認相似個體拿到過度樂觀成績。

前處理也要切清楚

平均值、標準差與補值規則先在訓練集估計，再套用到驗證與測試。

外送情境

要預測下個月：以前幾個月訓練，後面的月份驗證。

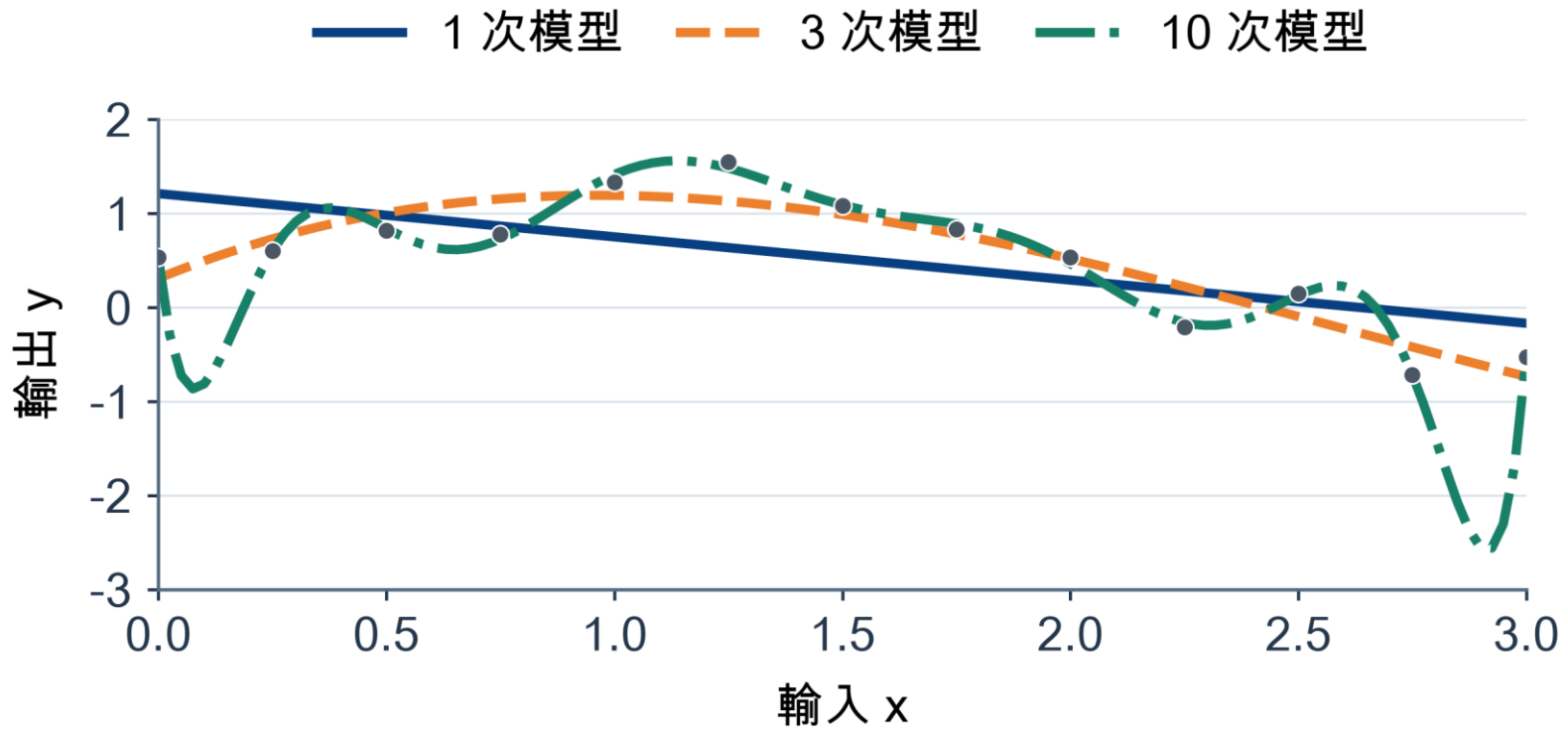
要服務新店家：評估時保留未見過的店家。

兩種切法回答的問題不同。

切分方式要模擬模型未來會遇到的新資料。

欠擬合、合適與過擬合

模型太僵硬可能抓不到趨勢，太靈活則可能把少量資料中的雜訊也跟著描出來。



欠擬合

在訓練資料上也留下明顯系統性偏差，例如直線無法跟上彎曲趨勢。

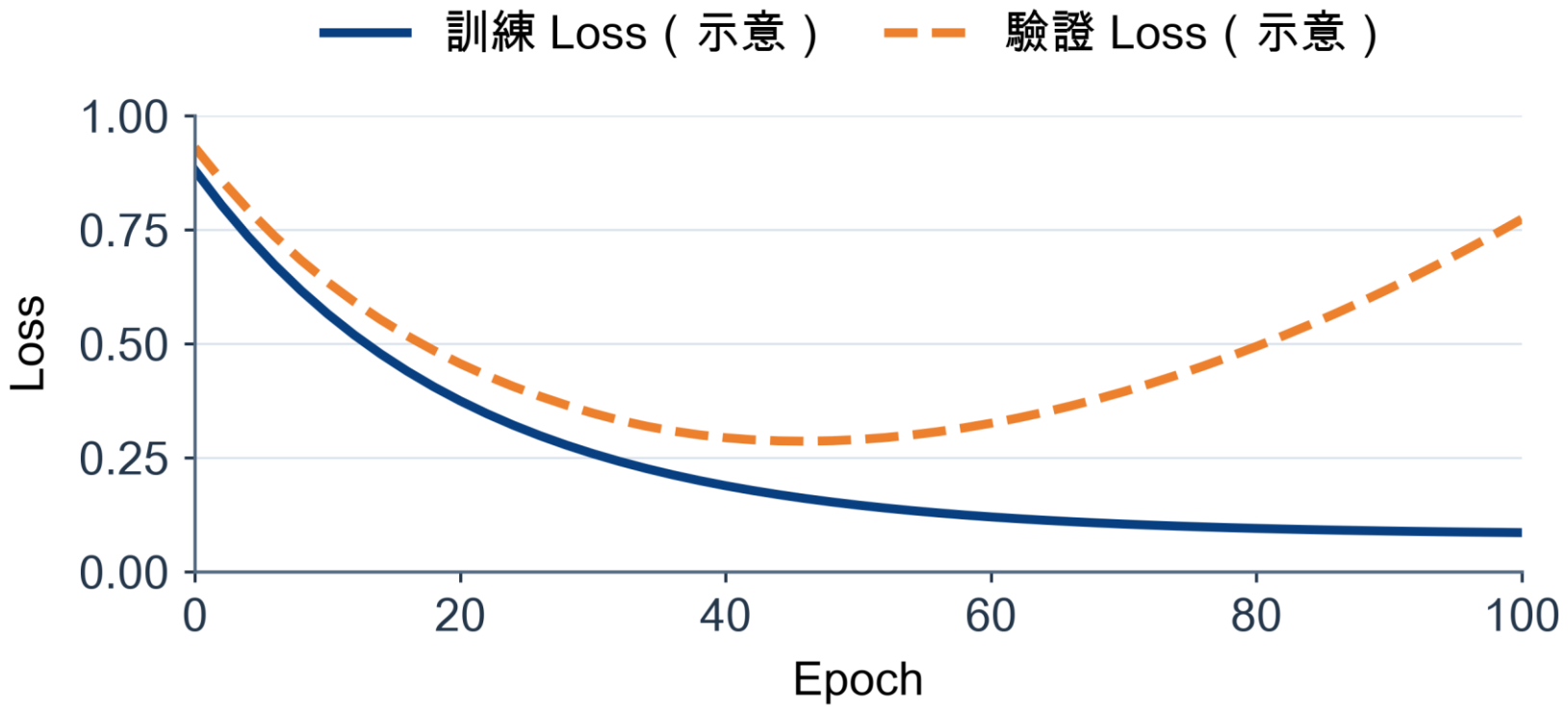
過擬合的數據證據

10 次模型訓練 MSE 約 0.002，驗證約 0.311；3 次模型驗證約 0.027。訓練更準，不代表新資料更準。

本圖為固定亂數種子的合成資料；彎曲本身不是過擬合的充分證據。

訓練與驗證曲線分開看

除了最後比分數，也要看訓練過程。橫軸 Epoch 是完整看過訓練資料的輪數，稍後再與 Step 比較。



Early stopping

依驗證表現決定何時停止，並保存驗證較佳的參數，而不是機械地選最後一步。

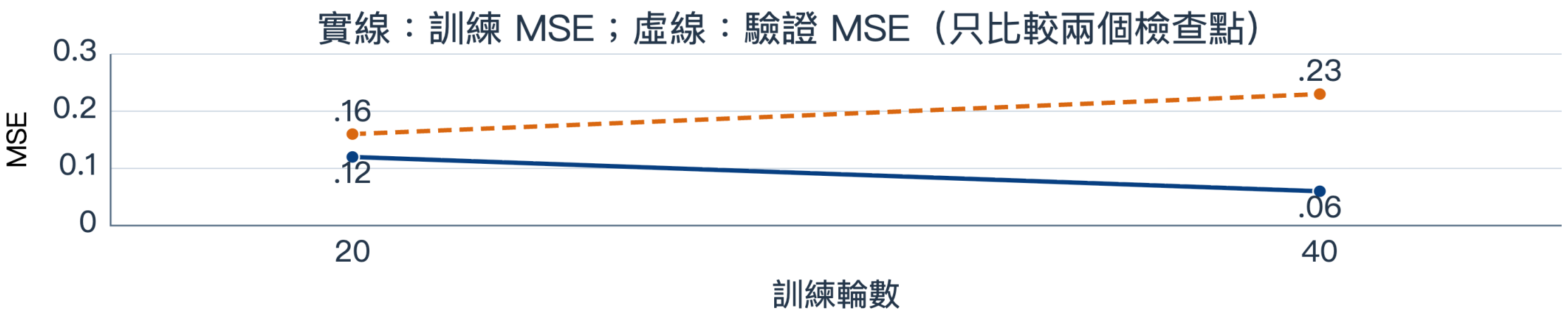
實務判讀

驗證值會波動，通常等一段耐心時間再決定停止。驗證資料太少或不匹配，也會影響判斷。

圖中曲線為機制示意；停止時機用驗證集選，測試集留給最後評估。

停下來想 ⑤：訓練最低，就選最後一步？

同一模型、相同資料切分，只比較下列兩個檢查點。哪個結論最合理？



A 選第 40 輪，訓練 MSE 最低

B 用測試集反覆挑到最好的輪數

C 先保留第 20 輪，並檢查驗證惡化原因

先看圖選答案，再用圖中的證據說明。下一頁揭曉。

先獨立想 45 秒，再和同桌交換理由。選答案之外，請說出你用了哪個觀念。

過擬合的改善方法與代價

訓練越準、驗證卻變差時，可以限制模型對細節的追逐。下列方法先認識作用，下堂再實作。

方法	為何可能有效	代價或限制
增加代表性資料	減少只靠少數樣本學到的偶然規律	資料與標註有成本
簡化模型	減少可擬合的複雜細節	太簡單會欠擬合
正規化 / Dropout	限制權重，或訓練時隨機遮蔽部分單元	限制太強可能欠擬合
Early stopping	選驗證較佳的訓練時刻	依賴可靠的驗證集

效果要由驗證資料證明，沒有單一方法適用所有模型與資料。

資料洩漏：模型偷看了什麼？

資料洩漏是訓練或選模過程使用了原本不該取得的資訊，讓離線成績過度樂觀。

把未來當特徵

出發前預測送達時間，卻放入「實際送達時刻」。這個欄位已直接透露答案。

相似樣本跨資料集

同一照片稍微裁切後，一張在訓練、一張在測試，可能測到記憶而非新情境能力。

全資料先做統計

切分前用所有資料估計補值或標準化參數，使測試資料的分布資訊進入訓練流程。

檢查問題

這個欄位在預測當下真的拿得到嗎？

同一事件或個體是否跨了資料集？

測試成績是否曾被用來反覆改模型？

高分值得高興，也值得確認測法。

防止洩漏要檢查整個資料流程，不能只檢查模型公式。

分布不匹配：熟悉的世界改變了

即使切分沒有洩漏，未來資料也可能與訓練資料不同。這會使過去的關係失去適用性。

生活例子

平日外送資料訓練出的模型，遇到連假、豪雨或新店家，誤差可能明顯增加。

與過擬合不同

過擬合關心對訓練細節過度貼合；分布變化關心使用情境不同。兩者可能同時發生。

如何處理

分群監控誤差，確認新情境是否有代表性資料，再決定補資料、改特徵或重訓。

回到 Class 0 的開車上班

熟悉路線練得很好，不代表封路後仍能準時。

Class 1 學會如何改參數。

Class 2 要驗證新路況下，這套經驗是否仍然適用。

新情境要另外評估。現在回到本課已知資料，準備讓演算法接手手動模型。

Step、Batch、Epoch 的差別

開始 E 自動訓練前，先看程式中的 steps 到底計算什麼。一次更新與看完一輪資料可能不同。

名稱	定義	100 筆資料、batch = 20
Batch	一次用來計算梯度的資料批次	每批 20 筆
Step	執行一次參數更新	每個 batch 更新一次
Epoch	把整份訓練資料走過一輪	一輪 5 steps
10 epochs	重複走過資料 10 輪	共 50 steps

本課每步使用全部樣本，是 full batch；因此此實作一個 step 對應一個 epoch。

最佳化器的入門地圖

剛才釐清資料批次。再確認 E 要用哪種更新方法：本課採全部樣本的梯度下降。

方法	核心想法	需要注意
Full-batch GD	全部訓練資料算一次梯度	每步成本較高
Mini-batch SGD	抽一小批估計梯度	方向有雜訊，批次大小影響過程
Momentum	累積歷史更新方向	可能減少來回震盪，也可能過頭
Adam	結合歷史方向與梯度平方尺度	仍需選學習率，並非一定最好

下一堂再實作 batch 與 Adam，今天先能說出每種方法改變哪一部分。

E1 操作：凍結手動起點

課程進入最後階段。先決定保留哪一組手動結果，再開啟 E1 的交接開關。

1　確認 C 的目前結果

若要採用先前某組參數，先把紀錄中的值填回 C 並執行。E1 使用目前設定，不會自動挑最低分。

2　確認並保存起點

在 E1 將 CONFIRM_HANDOFF 改為 True，再執行。程式複製 manual_params 成為 start_params，並印出起點 MSE。

3　比較期間保持固定

開始比較學習率後，只重跑 E2。不要重跑 E1，否則會重新建立交接起點。

E1 的開關

```
CONFIRM_HANDOFF = True
```

E1 只保存起點，尚未訓練。E2 另有 RUN_AUTOMATIC 開關。

E2 結尾挑戰：同一模型自動學習

E1 已保存 start_params。E2 從這份固定起點訓練，手動曲線保留作為比較。

模型與資料保持相同

仍用 C 的三個 ReLU、同一份 x、y 與 MSE。只有參數更新的方式改成梯度下降。

一次更新做什麼

先計算預測，再算 MSE 與梯度，同時更新 w、b、c、d。下一步用新的參數重算。

比較條件保持相同

第一次用 0.01 與 2,000 步。之後只更換學習率，E2 每次都從 start_params 重新開始。

更新主線（精簡版）

```
p = copy_params(start_params)

for step in range(2000):

    g = gradients(p, x, y)

    p = update(p, g, 0.01)
```

演算法結果不保證每次勝過手動；遇到失敗，要用 Loss 與梯度找原因。

E2 操作方法：開啟訓練與比較學習率

確認 E1 已完成，再操作 E2。每次試驗都從凍結的同一組參數出發。

1　第一次自動訓練

RUN_AUTOMATIC 設 True，learning_rate 設 0.01，steps 設 2000。執行後先讀完成步數與 MSE。

2　觀察中間過程

E2 顯示初始、10、100、500、最後一步的曲線快照，再顯示目標、手動與自動的比較圖。

3　只更換學習率

保留起點與步數，再試 0.001、0.03。0.3 可作延伸觀察，若失穩就讀停止訊息並縮小步伐。

E2 的三個設定

```
RUN_AUTOMATIC = True

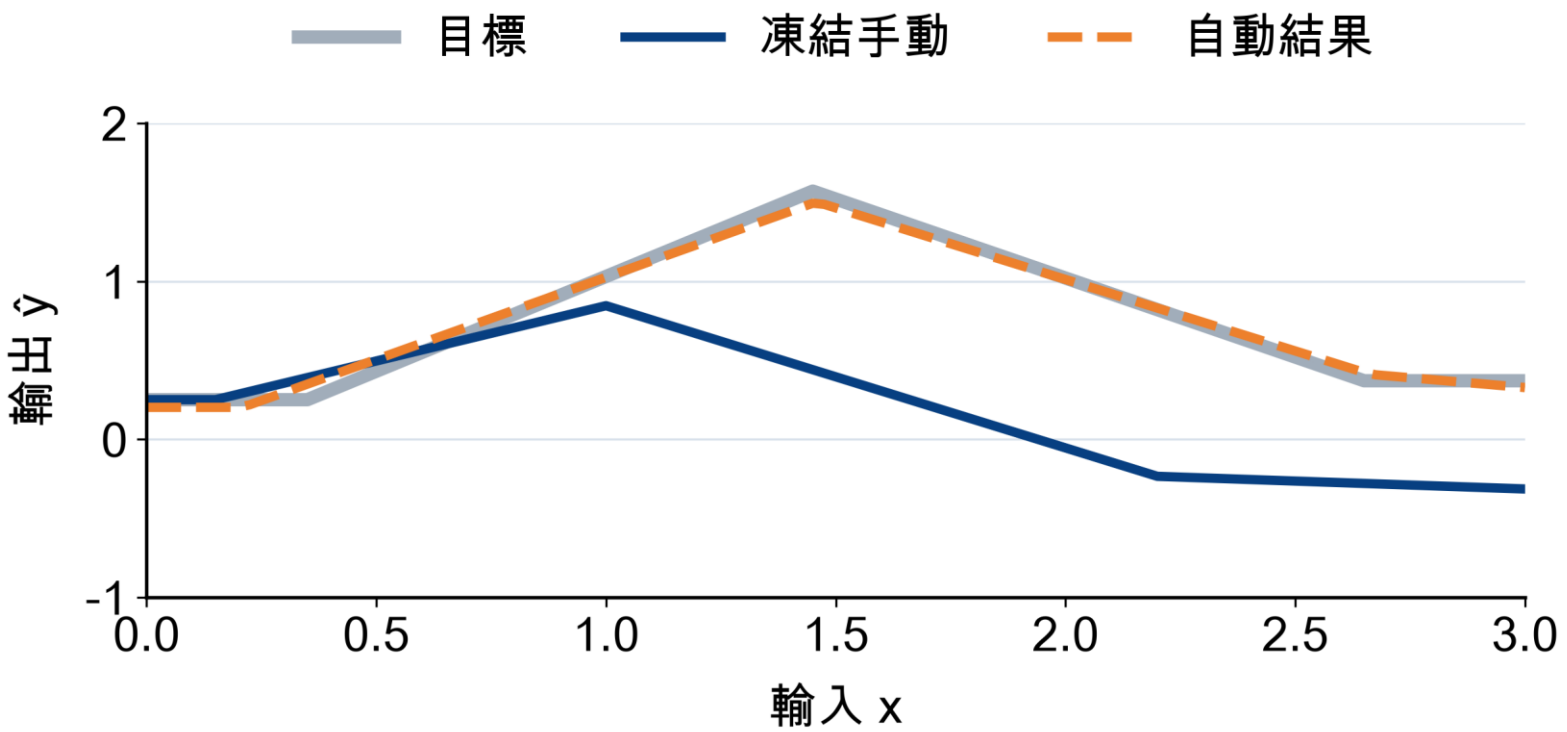
learning_rate = 0.01

steps = 2000
```

若只有「尚未啟動」訊息，檢查開關是否為 True。若沒有起點，先完成 E1。

E2 訓練結果怎麼讀？

此圖接續 C 的三次範例修改，以學習率 0.01 更新 2,000 步。你的結果會依手動起點而異。



先看曲線與分數

橘線比藍線更接近灰色目標。本例 MSE 從約 0.538 降至 0.00179，曲線仍有局部差異。

再看訓練歷程

E2 另有每步 MSE 圖。下降表示這份資料的預測改善，震盪或停滯時再檢查步長與起點。

比較圖與訓練紀錄一起讀。低訓練誤差仍需要新資料評估，才能談泛化。

停下來想 ⑥：怎樣比較學習率才有依據？

你已用 η = 0.01 跑完 E2，現在想比較 η = 0.001。哪個做法最能只比較學習率的影響？

E1 已保存手動起點 start_params。每次 E2 都複製這份起點，再開始訓練。

A

用上一輪自動結果，接著再訓練

B

保留凍結起點、資料與步數，只改 η

你選哪一邊？請用一句話說明理由，下一頁再核對答案。

先獨立想 45 秒，再和同桌交換理由。選答案之外，請說出你用了哪個觀念。

一起看答案 ⑥：怎樣比較學習率才有依據？

你已用 η = 0.01 跑完 E2，現在想比較 η = 0.001。哪個做法最能只比較學習率的影響？

E1 已保存手動起點 start_params。每次 E2 都複製這份起點，再開始訓練。

A

用上一輪自動結果，接著再訓練

B

保留凍結起點、資料與步數，只改 η

答案 B

固定起點、資料、模型與步數，只改學習率。重跑 E2 即可，不要改用上一輪訓練完的參數。

選對後，再說明其他選項為什麼不合適。用理由確認理解，不只記住選項字母。

人類與演算法怎麼公平比較？

兩者都在搜尋參數，但使用的資訊與更新方式不同。比較過程時，先固定評分條件。

人類的觀察

用整體形狀、轉折位置與先前的數學理解提出調整方向。記錄推測才能回頭檢查。

演算法的計算

依梯度反覆做小幅更新。E2 從你的起點繼續，讓你看見自動更新的貢獻。

比較的限制

固定起點、模型與資料，比較學習率和步數。本活動並未控制人與電腦的總耗時，不能當作公平競賽。

同樣的曲線，參數可以不同

兩組參數可能畫出相同曲線，沒有唯一的秘密答案。

先比較輸出與 MSE，進階縮放公式留在備忘稿。

這次是人機接力示範，並未控制雙方相同運算時間。

自動接手展示的是更新機制。結果變好仍需要檢查曲線與新資料表現。

離堂檢查：你能解釋這四件事嗎？

用自己的話回答，比背名詞更能確認理解。先兩人討論，再對照答案。

問題	作答時應包含的觀念
w=2、b=−1，轉折在哪？	x = −b/w = 0.5
沒有激活函數，多層會如何？	可合併為一個仿射變換
訓練 MSE 降低證明了什麼？	只證明這份資料、這個 Loss 上改善
驗證變差，下一步看什麼？	檢查切分、過擬合與資料分布

先口頭回答這四題，再到 F 反思區寫下本次操作心得，執行後保留輸出。

F 完成分享：保存結果與檢查權限

繳交自己的 Colab 唯讀分享連結，不另上傳分數、密碼或截圖。

1　確認成果有保存

保留 C 的紀錄表、E2 的比較圖與 F 的反思。確認畫面顯示儲存完成，且沒有設定省略程式輸出。

2　設定可檢視的分享權限

右上「共用 / Share」，一般存取權設為知道連結的任何人，角色選檢視者，再複製連結。

3　用未登入視窗驗證

開無痕視窗貼上連結，確認可讀到成果。若校方禁止公開分享，改與教師帳號共用並告知限制。

交件前最後檢查

有基準與三次單項修改嗎？

自動更新的起點、學習率與實際步數寫清楚了嗎？

反思有解釋 MSE 改善能證明什麼嗎？

共用前移除不需公開的個資或私密內容。

分享的是文字、程式與已保存輸出。執行中的記憶體不會隨連結分享。

延伸閱讀與教材來源

下列材料用來核對概念與安排學習順序。生活情境、算例與本課曲線資料為原創教學設計。

Google Machine Learning Crash Course

Linear regression、Loss、Gradient descent、Datasets 與 Overfitting。適合用短單元複習定義與例題。

Dive into Deep Learning

Multilayer Perceptrons：激活函數、非線性與多層模型。想從概念接到程式，可讀此章。

大學課程與操作文件

Toronto CSC311：MLP 與反向傳播。MIT 6.S191：神經網路入門。Colab 官方 FAQ：儲存、分享與執行環境。

閱讀建議

先重算本課 MSE 與一次梯度更新。

再看 ReLU 的 w、b、c 實驗。

最後閱讀訓練與驗證的區別。

來源連結可在講者備註與配套講義直接開啟。

學公式時持續問：這個數字來自哪裡、代表什麼、改了會影響哪裡？

附錄｜三神經元模型的完整梯度

i 是樣本編號，j 是隱藏神經元編號。N 筆樣本、K 個單元；本課 K=3。

$$z_{ij} = w_j x_i + b_j \;;\; h_{ij} = \mathrm{ReLU}(z_{ij}) \;;\; e_i = \hat{y}_i - y_i$$

參數	MSE 的梯度（Σ 對 i=1,...,N 加總）
c_j	$\partial L/\partial c_j = (1/N)\,\Sigma\, 2 e_i h_{ij}$
d	$\partial L/\partial d = (1/N)\,\Sigma\, 2 e_i$
b_j	$\partial L/\partial b_j = (1/N)\,\Sigma\, 2 e_i c_j\, 1[z_{ij}>0]$
w_j	$\partial L/\partial w_j = (1/N)\,\Sigma\, 2 e_i c_j\, 1[z_{ij}>0]\, x_i$

梯度要在同一組參數上計算；ReLU 零點本課採 0，完整公式供課後對照程式。

附錄｜NumPy 形狀與常見錯誤速查

先印 shape，再逐步執行中間運算。每次只修正一個問題，避免用 reshape 掩蓋資料意義。

症狀	常見原因	檢查或修正
broadcast 錯誤	最右軸不相等，且都不為 1	印出左右 shape；依意義新增軸
matmul 維度錯誤	內側 K 不一致	確認 (N,K) @ (K,M)
Loss 非預期但沒報錯	(N,1) 與 (N,) 相減	先 assert 形狀一致
少一筆就壞掉	squeeze 拿掉批次軸	指定 squeeze(axis=1) 或保留切片
起點紀錄也跟著改	陣列共用資料	保存參數用 .copy()
不能做小數更新	參數陣列 dtype 為整數	初始化用 dtype=float

@ 用於矩陣乘法、* 用於逐元素乘法、 用於乘方；三者不可混用。**