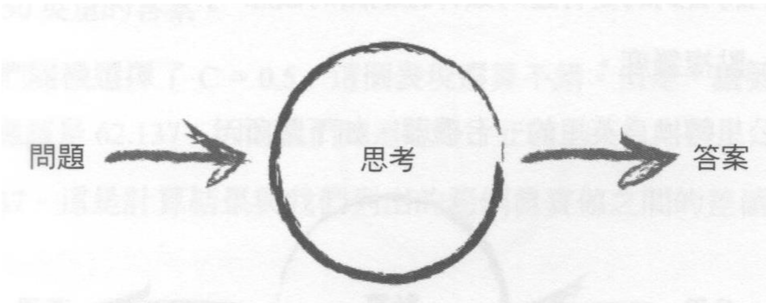
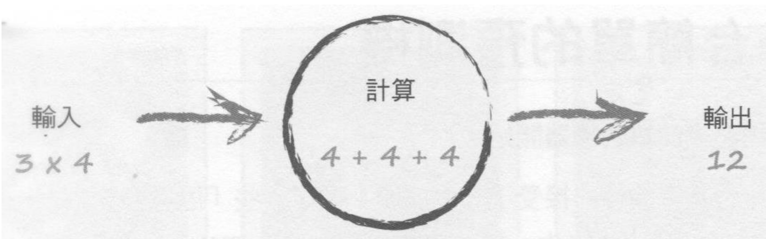
一台基本的機器接受了一個問題，做了一些「思考」，並輸出一個答案。



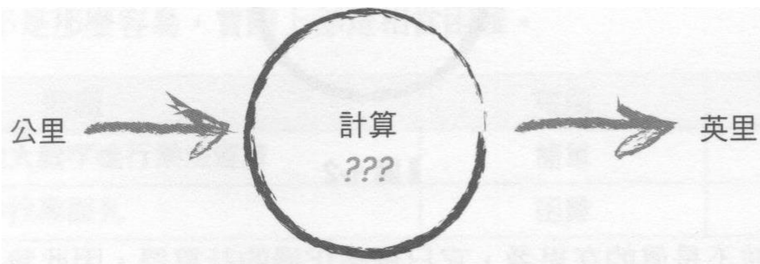
我們從眼睛輸入圖片，使用大腦分析場景，並得出場景中有那些物體。
一台電腦接受一些輸入，執行一些計算，然後輸出。

一台電腦對「3x4」的輸入進行處理，這個處理也許是將乘法變為簡單的一組加法，然後彈出答案「12」。



稍微增加一點複雜度：

將公里轉換為英里的一台機器，想像一下我們不知道公里和英里之間的轉換公式。我們所知道的是：兩者之間的關係是線性(Linear)的。

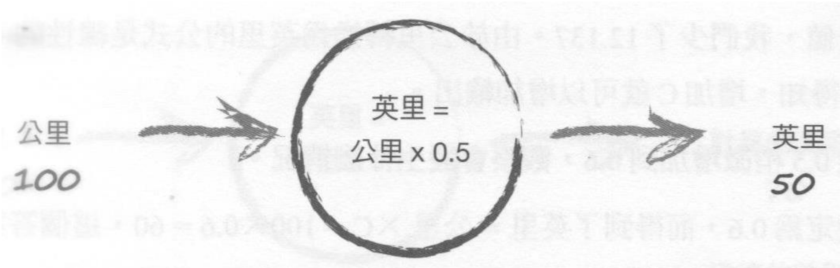


公里和英里之間的的線性關係，為我們提供了關於計算的線索，即它的形式應該是「英里=公里 x C」，其中 C 為常數。不過現在我們還不知道這個常數 C 為多少。

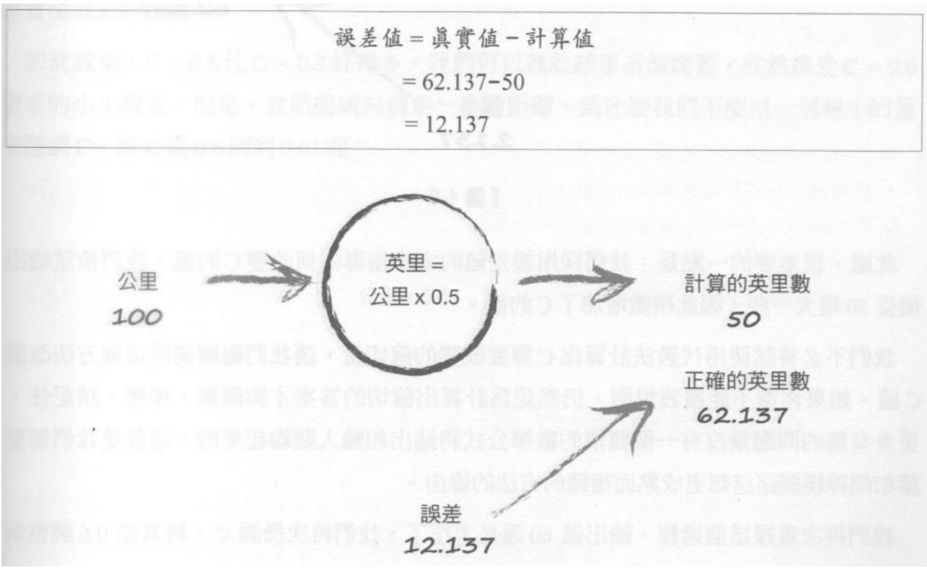
我們僅有的線索就是一些正確的公里/英里匹配數值對範例，如下表：

真實範例	公里	英里
1	0	0
2	100	62.137

我們應該如何做，才能計算出常數 C？
我們先選擇一個隨機的數值，讓機器試一試！這裡試著使用 C=0.5，看看會發生什麼情況？



我們令：英里=公里 xC，其中公里為 100，我們猜測 C 為 0.5。這台機器得到了 50 英里的答案。但是上表中，編號 2 的真實數據告訴我們，答案應該是 62.137，因此我們知道這是不正確的，答案少了 12.137。計算結果與我們的真實數據之間的差異，即誤差。如下所示：

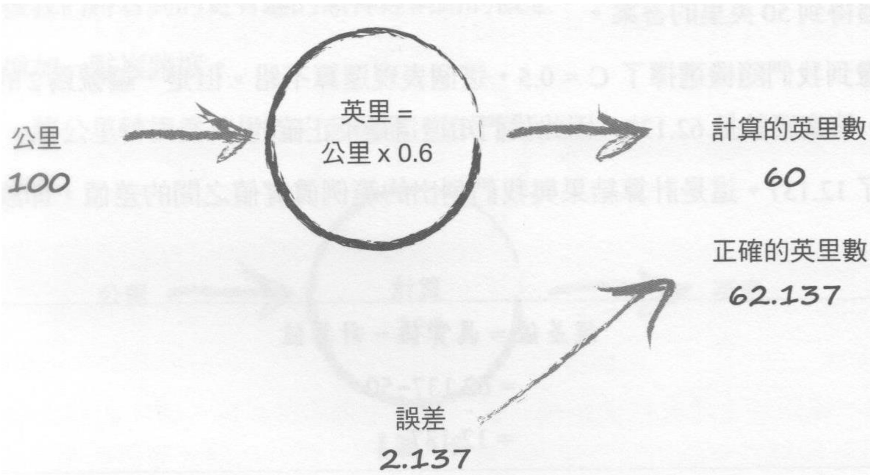


下一步，我們要做什麼呢?我們已經知道上一次的計算結果有錯，並且已經知道相差了多少。利用這個誤差值，可以指引我們得到第二個、更好的 C 猜測值。

我們回顧一下剛剛的誤差值，少了 12.137。由於公里轉換為英里的公式是線性的，即英里=公里 xC，由此得知，增加 C 就可以增加輸出。

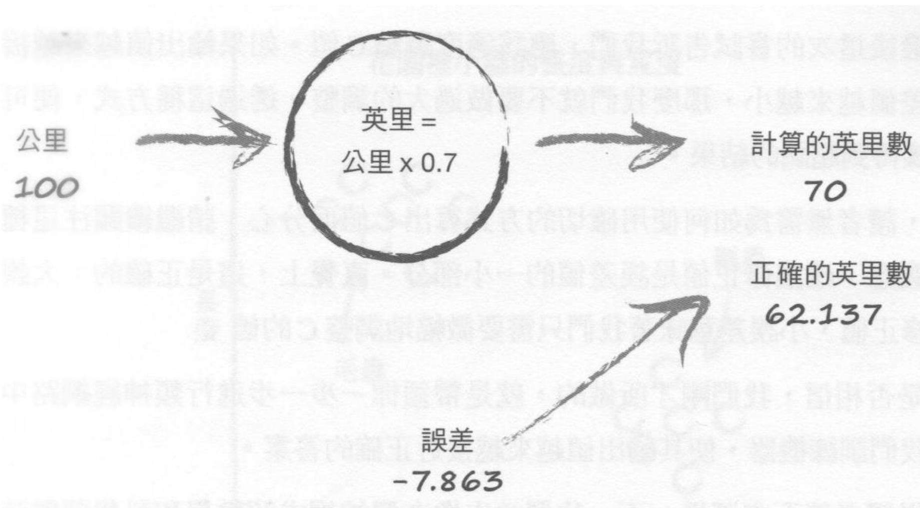
我們將 C 從 0.5 稍微增加到 0.6，觀察會發生甚麼變化。由於將 C 設定為 0.6，而得到英里=100 x 0.6 = 60，這個答案比先前的 50 更好，明顯有進步。

現在誤差值變得更小，為 2.137。這個數值甚至可能是我們可以接受的一個誤差範圍值。

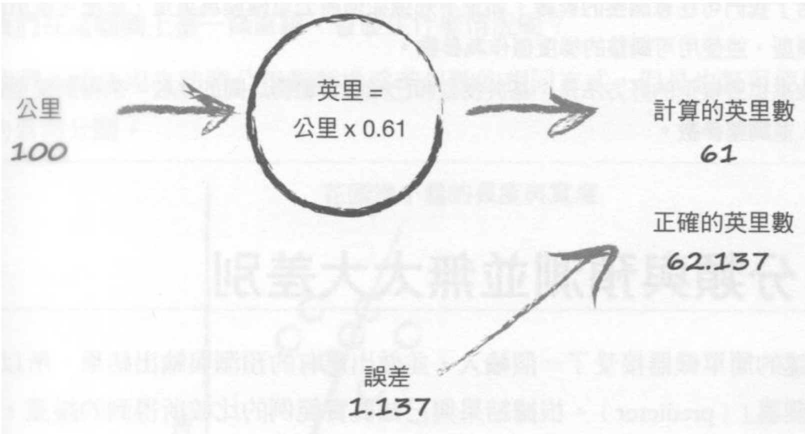


以上，很重要的一點是：我們利用了誤差值的大小指導如何改變 C 的值。我們希望輸出值從 50 增大一些，因此稍微增加了 C 的值。我們並不一定得使用代數法來計算出 C 所要改變的確切量，讓我們繼續使用這個方法改進 C 值。

我們繼續重複這個過程，輸出值 60 還是小了一點，我們再次微調 C，將 C 從 0.6 調整到 0.7，結果超過已知的正確答案。先前的誤差值為 2.137，現在的誤差值則為-7.863。這個數字告訴我們不是不足，而是已經超過了。如此說來，C=0.6 比 C=0.7 好得多，我們可以就此結束這個練習，欣然接受 C=0.6 所帶來的小小誤差。



回想一下我們剛剛繼續向前走的一小段距離，為甚麼我們不使用一個較小的量來微調 C，將 C 從 0.6 調到 0.61 呢？

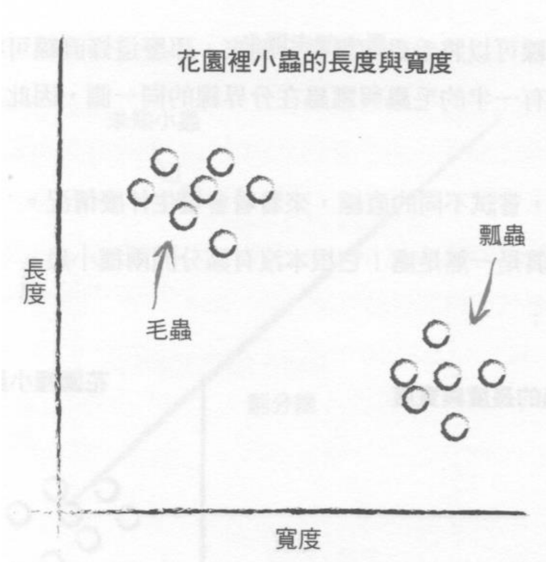


這比先前得到的答案要好得多。我們得到的輸出值是61，比起正確的答案62.137，只差了 1.137。因此最後的這次調整告訴我們，應該適度的調整 C 值。如果輸出值越來越接近正確答案時，即誤差越來越小，那麼我們不要做過大的調整。透過微調的方式，便可以避免出現剛剛得到的超過效果。

以上，我們剛剛所做的，就是帶領機器一步一步進行類神經網路中學習的核心過程。我們利用多筆真實數據來訓練機器，使其輸出值越來越接近正確的答案。一些人將這種方法稱為「迭代」(iteratve)，意思是持續、一點一點地接近答案，接近我們的目標值。

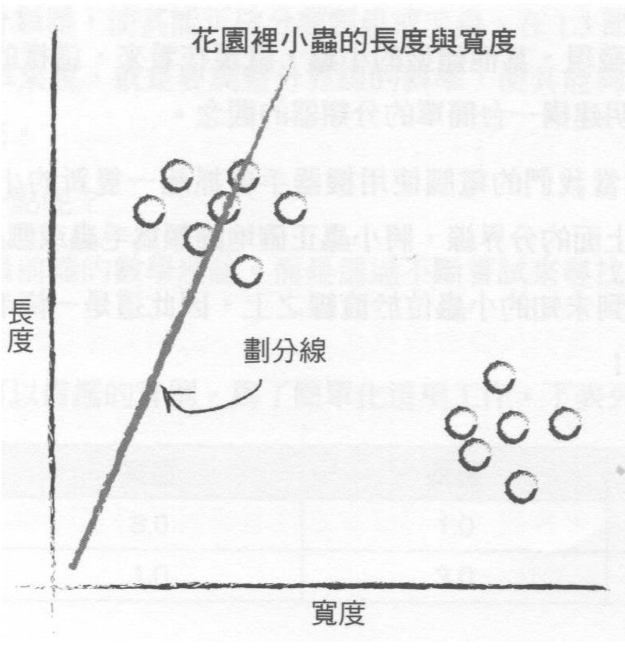
上述簡單的機器接受了一個輸入，並做出應有的預測與輸出結果，所以我們將其稱為「預測器」(predictor)。根據結果與已知真實數據的比較所得到的誤差，進而調整內部參數，使預測更加正確。

我們來看看下圖，其顯示了測量到的花園小蟲子的寬度與長度。



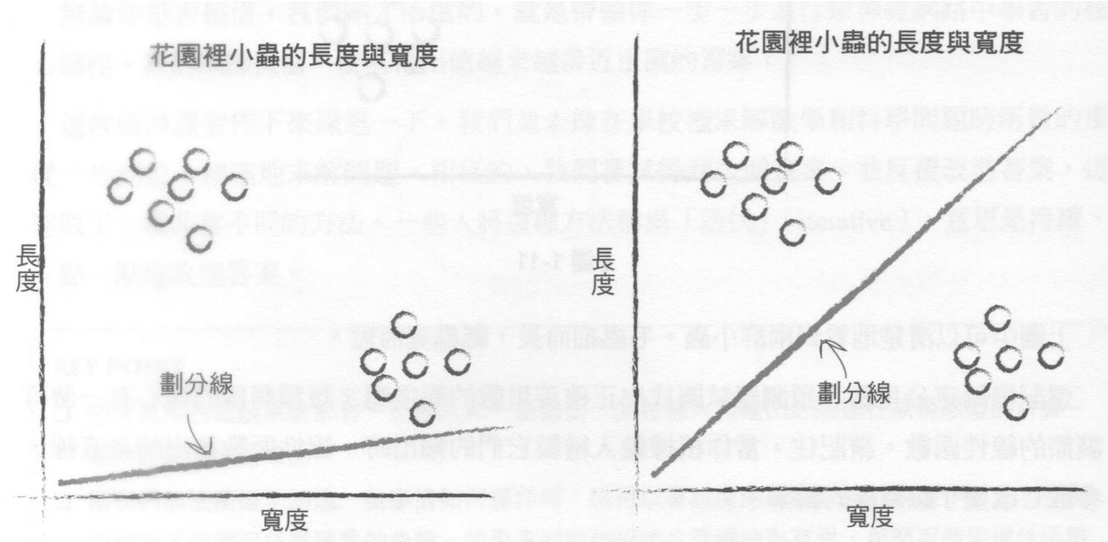
上圖中可以清楚的看到兩群小蟲。毛蟲細而長，瓢蟲寬而短。還記得前面我們所使用的公里-英里範例裡的可調節線性函數。當你根據輸入繪製他們的輸出時，線性函數是一直線，而參數 C 改變了該直線的斜率。

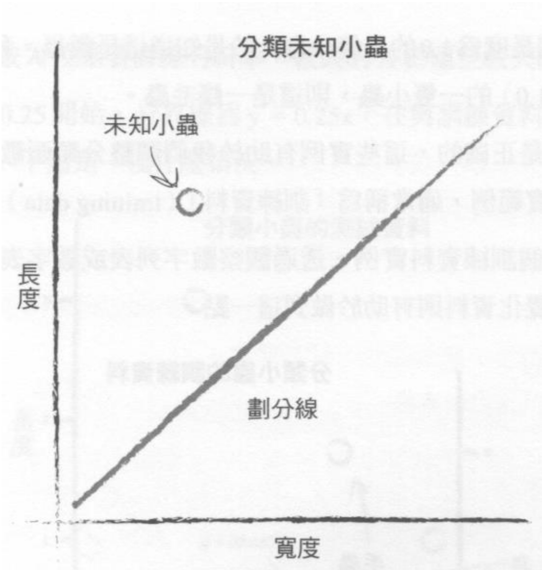
如果我們直接在這幅畫上畫一條直線，雖然我們不能使用先前的公里-英里的轉換方式，但是我們也許可以將不同性質的事物分開。



如果直線可以將毛蟲與瓢蟲分開，那麼這條直線可以根據測量值對未知的小蟲來進行分類。由於上圖中有一半的毛蟲與瓢蟲在分界線的同一側，因此上述直線並沒有做到這一點。

我們再次調整直線的斜率，來看看會發生甚麼情況。最終將可以讓直線整齊的將瓢蟲與毛蟲區分開來，我們就可以利用這條直線作為小蟲的分類器。



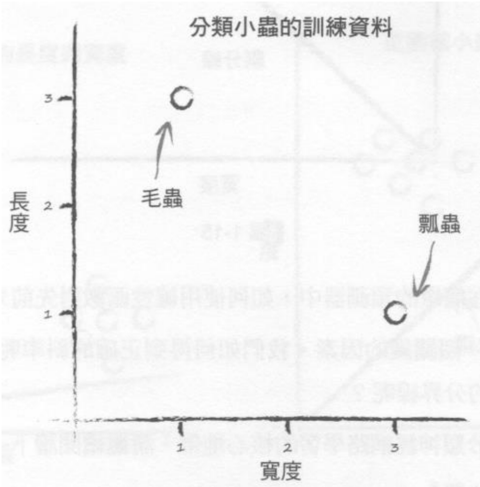


我們已經體驗到，在一個簡單的預測器中，如何使用線性函數對未知的資料進行分類。而我們同時應該要注意到一個關鍵的因素，我們如何得到正確的斜率呢？我們該如何改進(修正)，才能劃分好這兩種小蟲的分界線呢？

現在，我們想要訓練線性分類器，使其能正確的分類毛蟲與瓢蟲。根據之前的圖中觀察，若想做到這一點，就是要調整分界線的斜率。我們無須用到很高深的數學理論，而是利用不斷嘗試來尋找前進的方向。

我們有寬度為 3.0 和長度為 1.0 的一隻瓢蟲，我們還有長度較長(3.0)、寬度較小(1.0)的一隻毛蟲。這些實例數據有助於我們調整分類函數的斜率。而用於訓練預測器或分類器的真實範例，通常稱之為「訓練資料」(training data)。

實例	寬度	長度	小蟲
1	3.0	1.0	瓢蟲
2	1.0	3.0	毛蟲



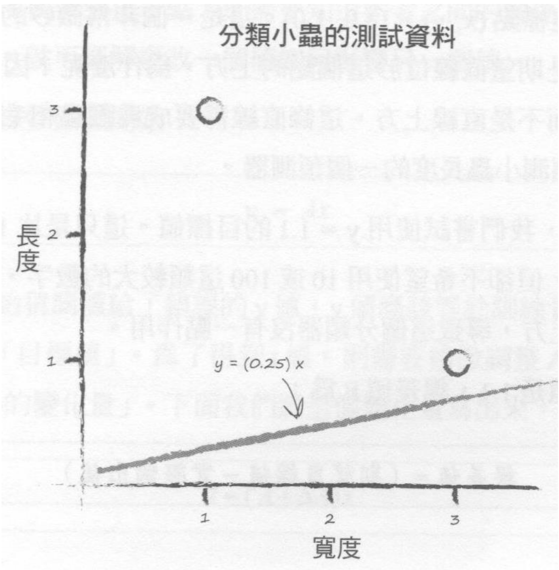
我們先使用一個隨機分界線來開始討論，回顧一下公里-英里轉換實例，我們也可以利用一條直線來進行相同的處理。

$$y = Ax$$

你也許會注意到，y = Ax 比完整的直線形式 y = Ax + B 更簡單。非零值 B 意味著直線不經過座標原點。但是目前看起來，B 不為零並沒有任何用途。

之前，我們看到參數 A 控制著直線的斜率，較大的 A 對應至較大的斜率。

我們試著從 A = 0.25 開始，分界線為 y = 0.25x。我們在圖上繪製出這條直線，觀察一下是甚麼情況。



無須任何計算式，我們可以觀察到 y = 0.25 x 並不是一台很好的分類器，這條直

線未將兩種類型的小蟲區分開來。

直觀上，我們觀察到需要將直線向上移動一點，我們不能透過觀察法來畫出一條合適的直線。我們希望找到一種可以重複的方法，也就是利用一系列的電腦指令來達到這個目標。這一系列的指令稱為「演算法」(algorithm)。

我們觀察第一個訓練樣本資料：寬度為 3.0 和長度為 1.0 的瓢蟲。如果我們使用這個數值代入測試函數 $y = Ax$，其中 x 為 3.0，就可以得到：
$y = 0.25 \times 3.0 = 0.75$

從訓練資料告訴我們，寬度為 3.0 的小蟲，長度必須為 1.0，因此我們知道這個數字太小了。現在我們有了誤差值，我們可以利用此誤差值來調整參數 A。

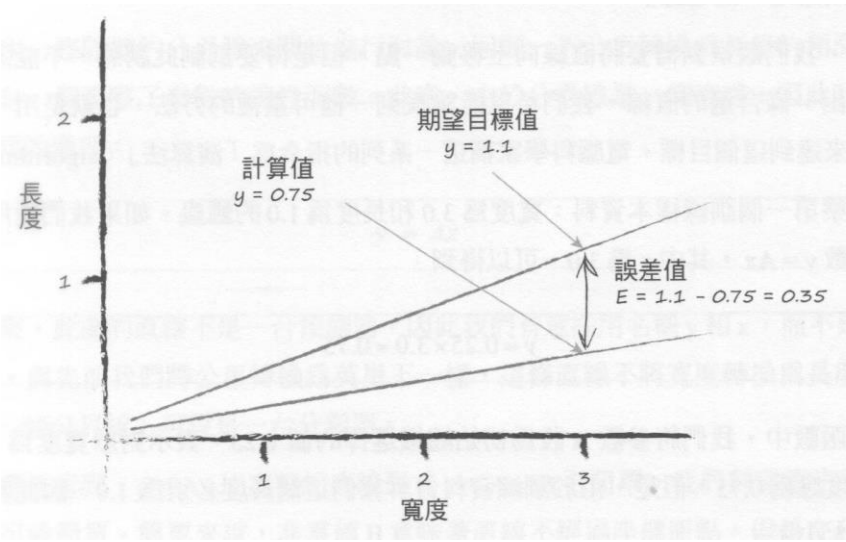
在調整參數 A 之前，我們要先考慮 y 應該是甚麼值。如果 y 為 1.0，那麼直線就會恰好經過瓢蟲所在的座標點 $(x, y) = (3.0, 1.0)$。我們其實不希望出現這樣的情況，而是期望直線位於這個點的上方。

因此，當 $x = 3.0$ 時，我們嘗試使用 $y = 1.1$ 為目標值。這只比 1.0 大一點的數，我們也可以選擇 1.2 或 1.3，但是我們不希望使用 10 或 100 這類較大的數字。

因此，期望的目標值是 1.1，誤差值 E 為：

誤差值 =（期望目標值 – 實際的輸出值）

$E = 1.1 - 0.75 = 0.35$



我們需要對 E 做甚麼，才能妥善的調整參數 A 呢?我們希望用 y 中稱為 E 的誤差值，來弄清楚參數 A 所需要改變的值。要做到這一點，則需要先知道兩者之間

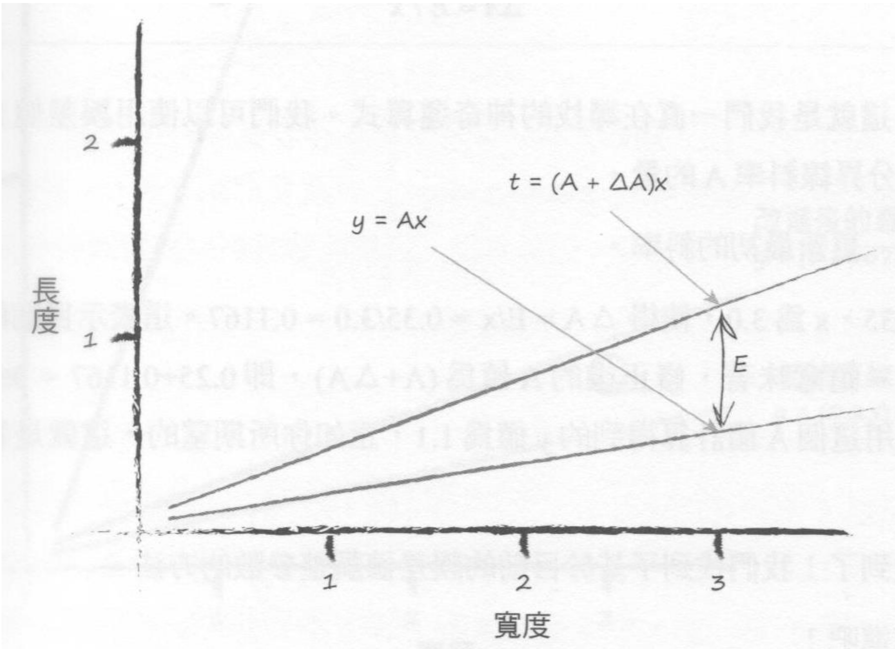
的關係。A 與 E 如何有關聯？

$$y = A x$$

A 的隨機產生的初始值，產生了錯誤的 y 值，y 值應該等於訓練資料給的定值。我們將正確的期望值 t 稱為「目標值」。為了得到這個 t 值，則需要微量調整 A 的值。我們使用 Δ 表示「微小的變化量」

$$t = (A + \Delta A)x$$

在圖形中表示為：



誤差值 E 是期望的正確值與基於 A 的隨機猜測值計算出來值之間的差異。也就是說 $E = t - y$ 。

$$t - y = (A + \Delta A)x - Ax$$

展開運算式並簡化：

$$E = t - y = Ax + (\Delta A)x - Ax$$

$$E = (\Delta A)x$$

得到了誤差值 E 與參數微調值 ΔA 存在一種簡單的關係。

現在我們想要知道需要將 A 調整多少，也就是 ΔA 要等於多少才能減少誤差值

若想要知道這點，我們只需要重新調整方程式的寫法，就可以將 ΔA 計算出來。

$$\Delta A = E/x$$

讓我們回到之前的誤差值 E = 0.35，x 為 3.0，使得

$$\Delta A = E/x = 0.35/3.0 = 0.1167$$

這表示目前的 A = 0.25 需要加上 0.1167。
修正後的 A 值為$(A + \Delta A)$ = 0.25 + 0.1167 = 0.3667，當 A = 0.3667 時，計算得到的 y = 1.1，就是我們所期望的目標值。

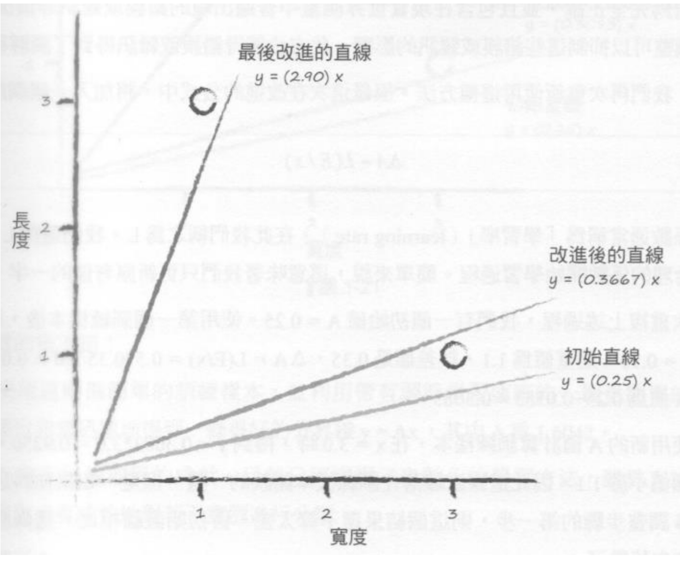
現在，我們已經完成一個實例訓練，接著從下一個實例中學習。

正確值為 x = 1.0 和 y = 3.0

線性函數使用更新後的 $A = 0.3667$，並把 x = 1.0 代入線性函數時，觀察會發生甚麼情況。我們得到 $y = 0.3667 * 1.0 = 0.3667$，這與訓練樣本中 $y = 3.0$相差甚遠。

基於先前相同的推理，我們希望直線不要經過訓練資料，而是稍微高於或低於訓練資料，所以我們將期望的目標值設定為 2.9。如此一來，毛蟲的訓練樣本就在直線的上方。

觀察向第一個訓練樣本學習後的改進直線，以及向第二個訓練樣本學習後的最終直線。
看這張圖，我們觀察出甚麼問題？



如果我們繼續這樣的操作，使用各個訓練樣本資料來進行改進，那麼我們得到的是最終改進的直線與最後一次訓練樣本非常匹配。實際上，最終樣本的直線並不會顧及所有先前的訓練樣本，而是拋棄所有訓練樣本的結果，只對最近的一個實際資料進行學習。

如何解決這個問題？

進行適度的改進與調整，也就是說我們不馬上很熱情的直接跳躍到新的 A 值，而是採用ΔA一小部分的變化值，而非整個ΔA。透過這個方法，小心謹慎的往訓練樣本指示的方向移動，保持先前訓練迭代週期中所得到值的一部分。

這種自我節制的調整，還帶來一種非常強大且意外的「副作用」，也就是當訓練資料本身不能確信為完全正確，並且包含現實生活中普遍出現的錯誤或雜訊等情況時，節制的調整可以抑制這些錯誤或雜訊的影響。

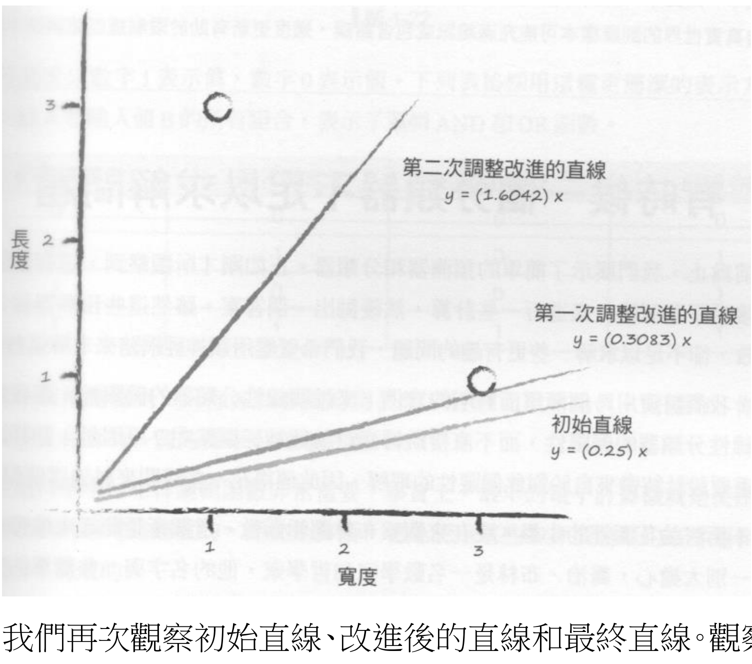
$$\Delta \mathrm{A} = \mathrm{L}(E/x)$$

我們再次使用這個方法，但是在這次的公式中將加入一個調節係數。調節係數通常稱為「學習率」(Learning rate)，在此我們稱之為 L。

我們選擇 L = 0.5 作為一個合理的係數開始學習過程。
再一次重複上述過程，引入第一筆資料：

$$\mathrm{A} = 0.25$$
$$\mathrm{y} = 0.25 * 3.0 = 0.75$$
$$\mathrm{E} = 1.1 - 0.75 = 0.35$$
$$\Delta \mathrm{A} = \mathrm{L}(E/x) = 0.5 * 0.35/3.0 = 0.0583$$
$$\mathrm{A} = \mathrm{A} + \Delta \mathrm{A} = 0.25 + 0.0583 = 0.3083$$

繼續使用第二筆資料來更新A：



我們再次觀察初始直線、改進後的直線和最終直線。觀察這種有節制的調節參數，

是否在瓢蟲和毛蟲之間得到了更好的分界線。