Artificial Neural Networks (人工神經網路)

> Reference：
>
> 1. Artificial Neural Networks - Dr. Tun-Wen Pai
> 2. Neural Networks Pt. 1: Inside the Black Box
> 3. Neural Networks Pt. 2: Backpropagation Main Ideas
> 4. Backpropagation Details Pt. 1: Optimizing 3 parameters simultaneously.
> 5. Backpropagation Details Pt. 2: Going bonkers with The Chain Rule

概述

人工神經網路（ANN）使用了**一種曲線**，能夠近乎完美的符合資料集。

使用的曲線為激勵函數，利用參數、權重等等來製作，藉由神經元來構造曲線，進而符合資料集。

本篇所講述的人工神經網路均屬於前饋神經網路（前饋神經網路）。

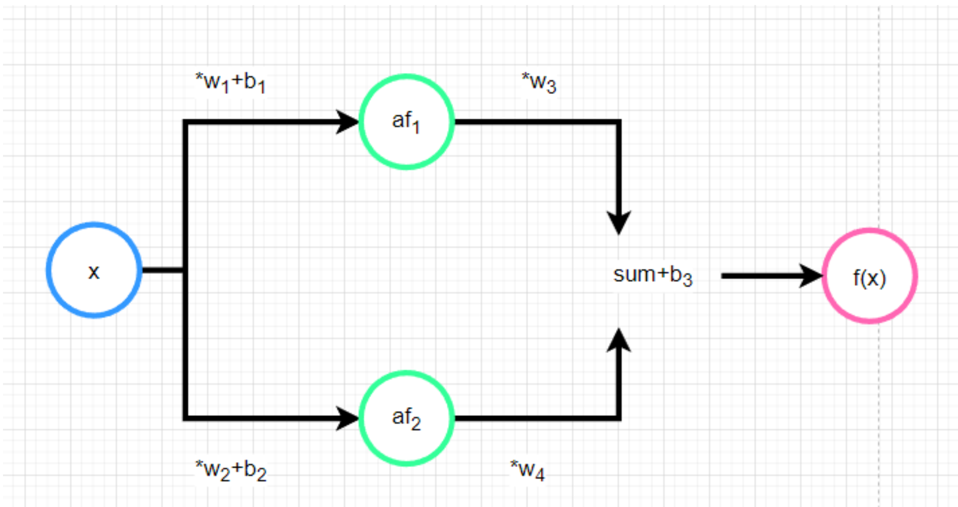
我們可以把他想成將一個參數放入 input 神經元後。

這個 input 神經元會隨著箭頭進入到 Hidden layer 的神經元，通常是一個激勵函數。

箭頭會逐漸塑造激勵函數，直到 output 神經元將曲線輸出。

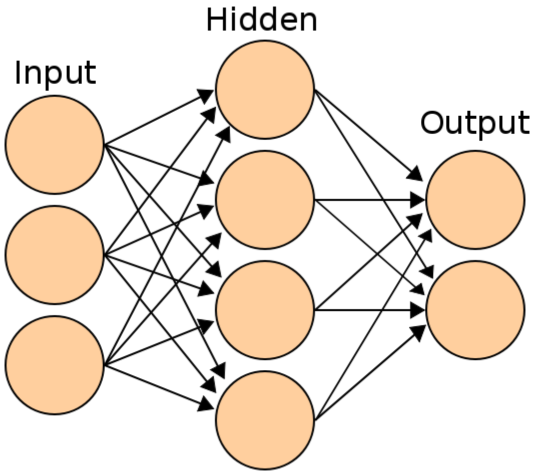
下圖是一個簡單的 ANN，我們將藍色圈圈稱為 input，綠色圈圈稱為 hidden，粉紅色圈圈稱為 output

w_1, w_2, w_3, w_4 為 weight，b_1, b_2, b_3 為 bias，而 af_1, af_2 為激勵函數。



而實際上可能會這麼複雜：

> Artificial neural network.svg - 維基百科，自由嘅百科全書



激勵函數

激勵函數在塑造曲線的時候扮演了重要的角色，主要分成四種：

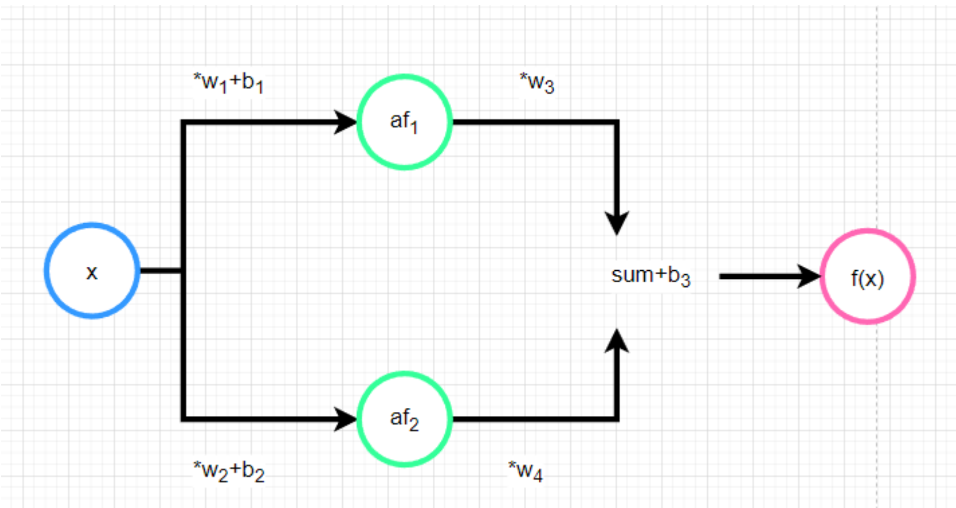
1. Tanh：$f(x)=\tan x=\frac{e^x-e^{-x}}{e^x+e^{-x}}$
2. Sigmoid / Logistic：$f(x)=\frac{1}{1+e^{-x}}$
3. ReLu：$f(x)=x^+=\max(0,x)$
4. Softplus：$f(x)=\ln(1+e^x)$

所謂的激勵函數本質上就是函數，可以想像成把參數放入激勵函數後，可以使激勵函數最後塑造出我們想要的曲線。

建構簡單 ANN

建立概述

我們以簡單的例子來說明，以下圖為例。



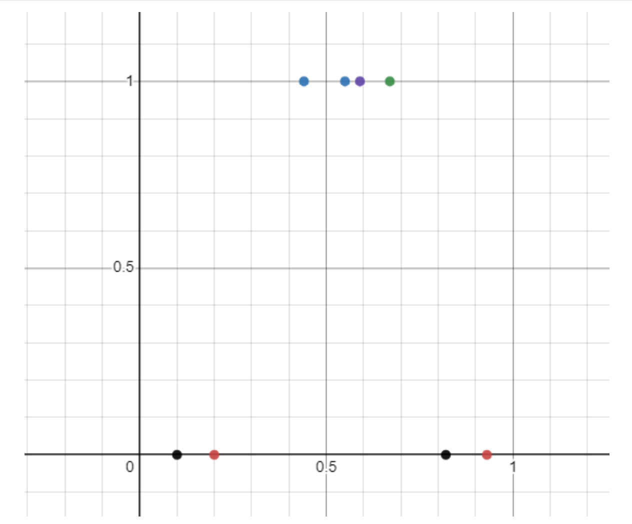
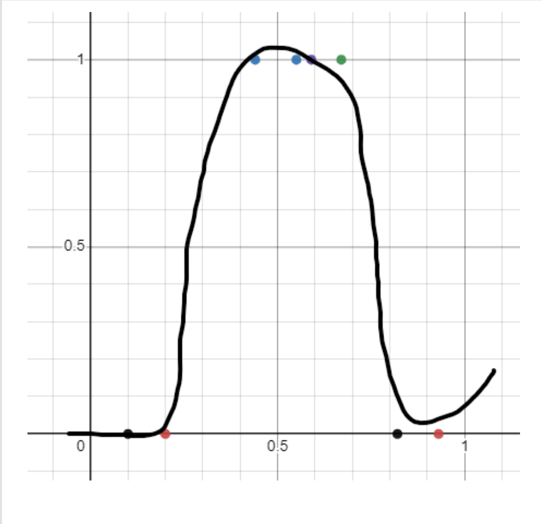
我們有一個*簡單的資料集*，分成三類，值域界於 0 到 1：

1. 服用少數量 ntut-xuan 筆記的人 → 考不好 (0)
2. 服用中等數量 ntut-xuan 筆記的人 → 考得好 (1)
3. 服用多數量 ntut-xuan 筆記的人 → 考不好 (0)

可以得到左圖。

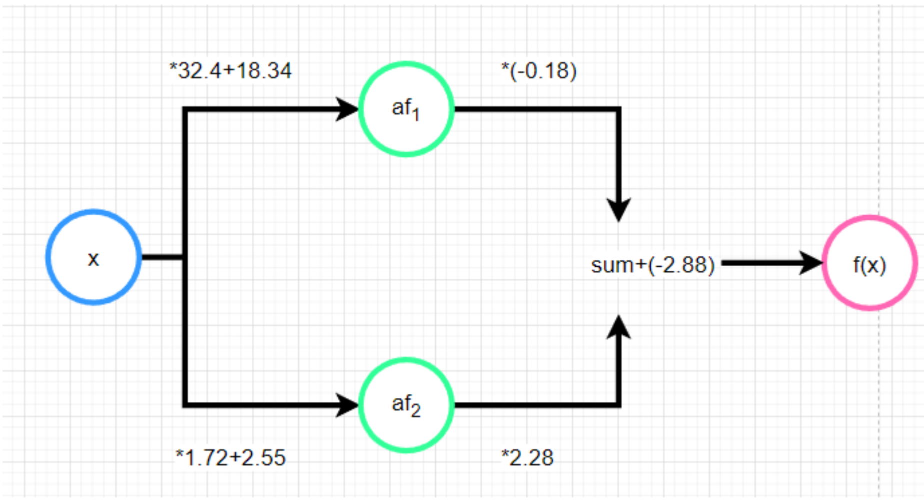
此時我們可能會想要用一條直線來分割這些資料，但這條直線可能不存在，因為不管怎麼畫都沒有辦法概括完全的資料。

如果這時候有一條*神奇的函數*來讓這些資料 match 就好，就像右圖。

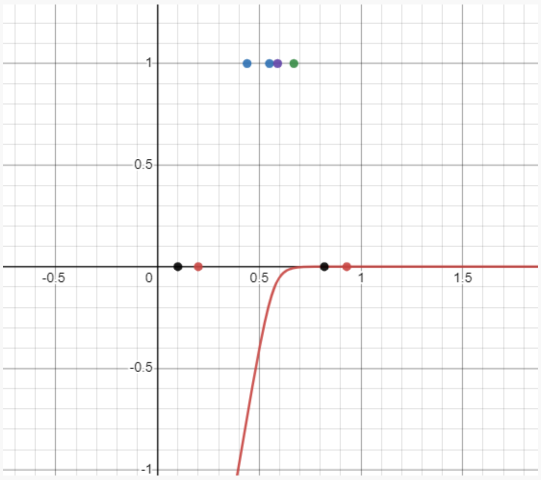
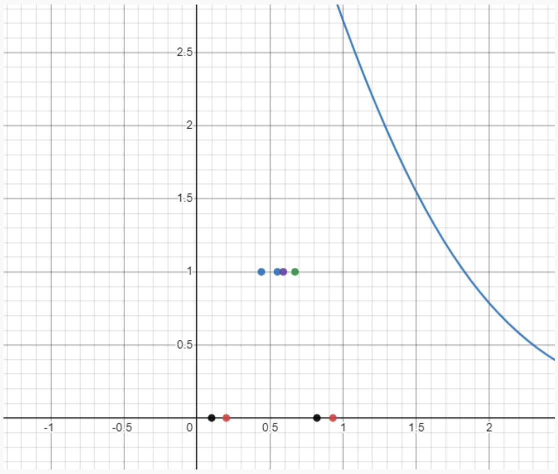
	

我們假設已經優化了類神經網路的 $w_1, w_2, b_1, b_2, w_3, w_4, b_3$ 參數，我們可以這樣建構我們的類神經網路。

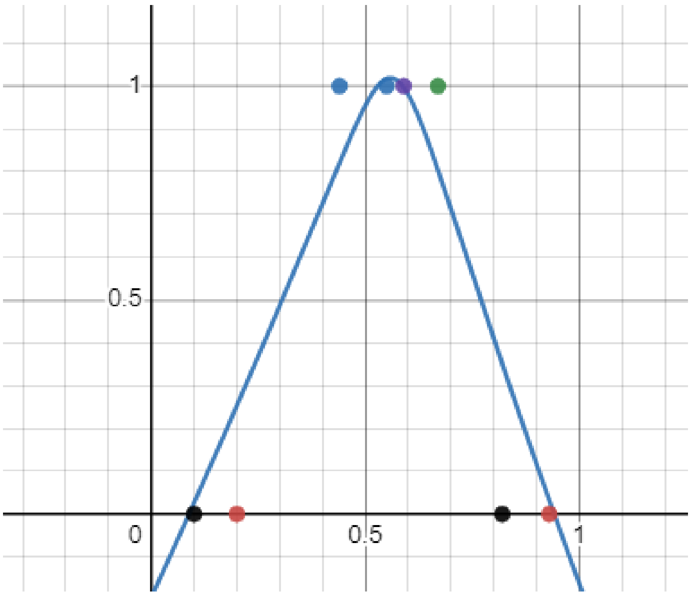
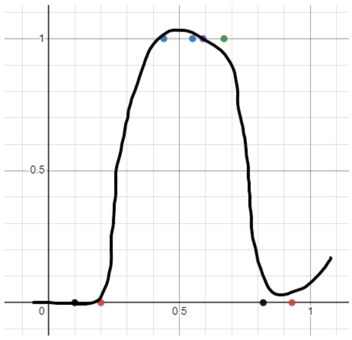
假設我們使用的激勵函數 af_1, af_2 為 Softplus：$f(x) = \ln(1 + e^x)$



我們可以由我們建構的類神經網路，往上走建構出一條曲線，往下走建構出另一條曲線，如下圖：

往上走	往下走
$f\left(x\right)=-0.18\ln\left(1+e^{\left(-32.4x+18.34\right)}\right)$	$g\left(x\right)=2.28\ln\left(1+e^{\left(-1.72x+2.55\right)}\right)$
	

最後將兩個曲線加起來，並減去 2.88，得到以下的曲線，就能夠得到我們幾乎亂畫出來的曲線了！

$h(x)=-0.18\ln\left(1+e^{\left(-32.4x+18.34\right)}\right)+2.28\ln\left(1+e^{\left(-1.72x+2.55\right)}\right)-2.88$	我亂畫的
	

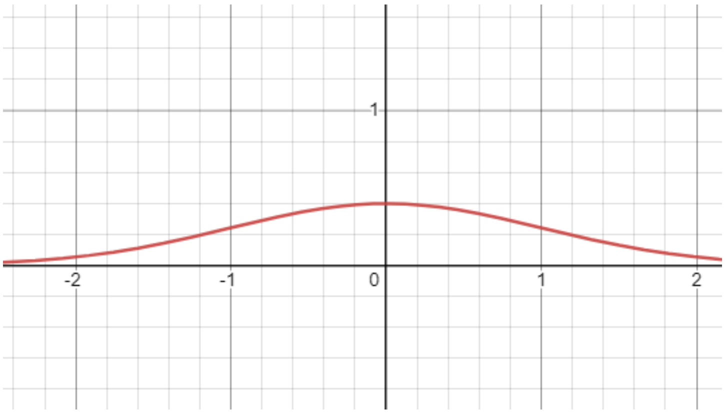
這時候我們就可以用這條曲線來判別我們特定資料集所出現的結果，所以人工神經網路*理論上*能夠成功分類所有的資料。

問題在於如何找出參數，來建構我們想要的曲線。

簡單 ANN 的參數優化 (Backward Propagation)

對於找出參數，我們可以先給定一個初始值，然後進行參數優化。

對於 weight 的部分，我們的初始值可以先給定為**標準常態分布**的隨機一個值，而 bias 可以先預設為 0

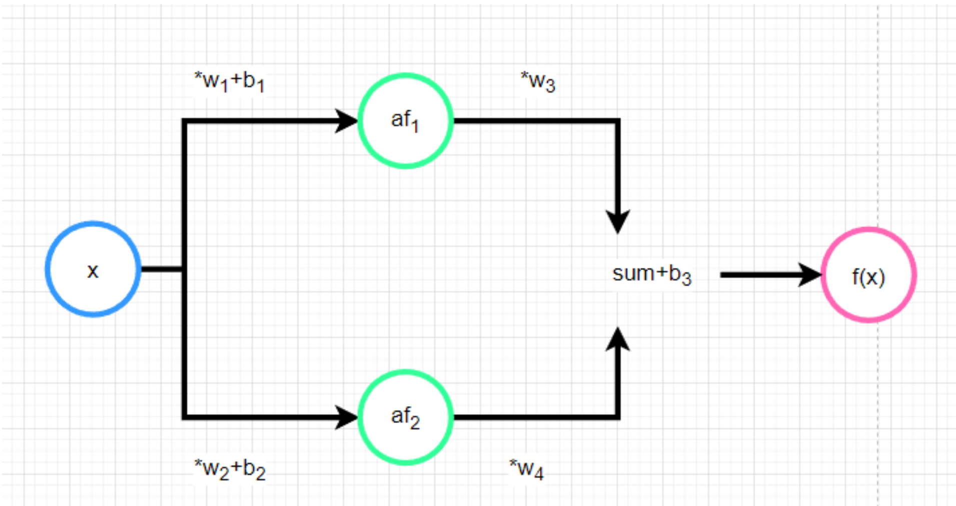


若要使曲線越來越擬合我們的資料集，我們得要先考慮是否能夠限縮 SSR，使他越小，讓曲線 $f(x_i)$ 越能擬合資料集。

SSR 即為殘差平方和，即為對於在 $x=i$ 上的資料集，其所有資料 $(y_i - f(x_i))^2$ 的和，可以定義為 $\mathrm{SSR} = \sum_{i=1}^{n}(y_i - f(x_i))^2$

若我們想要優化參數，可以找出參數對 SSR 的導函數，接著使用梯度下降法來同步優化所有參數

同樣以下圖為例：



我們令 $af_1(x_i)$ 運算出的結果叫做 $y_{1,i}$，$af_2(x_i)$ 運算出的結果叫做 $y_{2,i}$

可以得到 $f(x_i) = w_3 \times y_{1,i} + w_4 \times y_{2,i} + b_3$

又 $y_{1,i} = \ln(1+e^{x_{1,i}}), y_{2,i} = \ln(1+e^{x_{2,i}}), x_{1,i} = x_i \times w_1 + b_1, x_{2,i} = x_i \times w_2 + b_2$，以 Softplus 為例。

找出導函數，可以使用鎖鏈法則來尋找。

$$\frac{d\mathrm{SSR}}{db_3} = \frac{dSSR}{df(x_i)} \times \frac{df(x_i)}{db_3} = \sum_{i=1}^{n} -2(y_i - f(x_i)) \times 1$$

$$\frac{d\mathrm{SSR}}{dw_3} = \frac{dSSR}{df(x_i)} \times \frac{df(x_i)}{dw_3} = \sum_{i=1}^{n} -2(y_i - f(x_i)) \times y_{1,i}$$

$$\frac{d\mathrm{SSR}}{dw_4} = \frac{dSSR}{df(x_i)} \times \frac{df(x_i)}{dw_4} = \sum_{i=1}^{n} -2(y_i - f(x_i)) \times y_{2,i}$$

$$\frac{d\text{SSR}}{db_1} = \frac{dSSR}{df(x_i)} \times \frac{df(x_i)}{dy_1} \times \frac{dy_1}{dx_1} \times \frac{dx_1}{db_1} = \sum_{i=1}^{n} -2(y_i - f(x_i)) \times w_3 \times \frac{e^{x_{1,i}}}{1+e^{x_{1,i}}} \times 1$$

$$\frac{d\text{SSR}}{dw_1} = \frac{dSSR}{df(x_i)} \times \frac{df(x_i)}{dy_1} \times \frac{dy_1}{dx_1} \times \frac{dx_1}{dw_1} = \sum_{i=1}^{n} -2(y_i - f(x_i)) \times w_3 \times \frac{e^{x_{1,i}}}{1+e^{x_{1,i}}} \times x_{1i}$$

$$\frac{d\text{SSR}}{db_2} = \frac{dSSR}{df(x_i)} \times \frac{df(x_i)}{dy_2} \times \frac{dy_2}{dx_2} \times \frac{dx_2}{db_2} = \sum_{i=1}^{n} -2(y_i - f(x_i)) \times w_4 \times \frac{e^{x_{2,i}}}{1+e^{x_{2,i}}} \times 1$$

$$\frac{d\text{SSR}}{dw_2} = \frac{dSSR}{df(x_i)} \times \frac{df(x_i)}{dy_2} \times \frac{dy_2}{dx_2} \times \frac{dx_2}{dw_2} = \sum_{i=1}^{n} -2(y_i - f(x_i)) \times w_4 \times \frac{e^{x_{2,i}}}{1+e^{x_{2,i}}} \times x_{2i}$$

接著使用梯度下降

$\text{Step size} = \text{Derivative} \times \text{Learning Rate}$

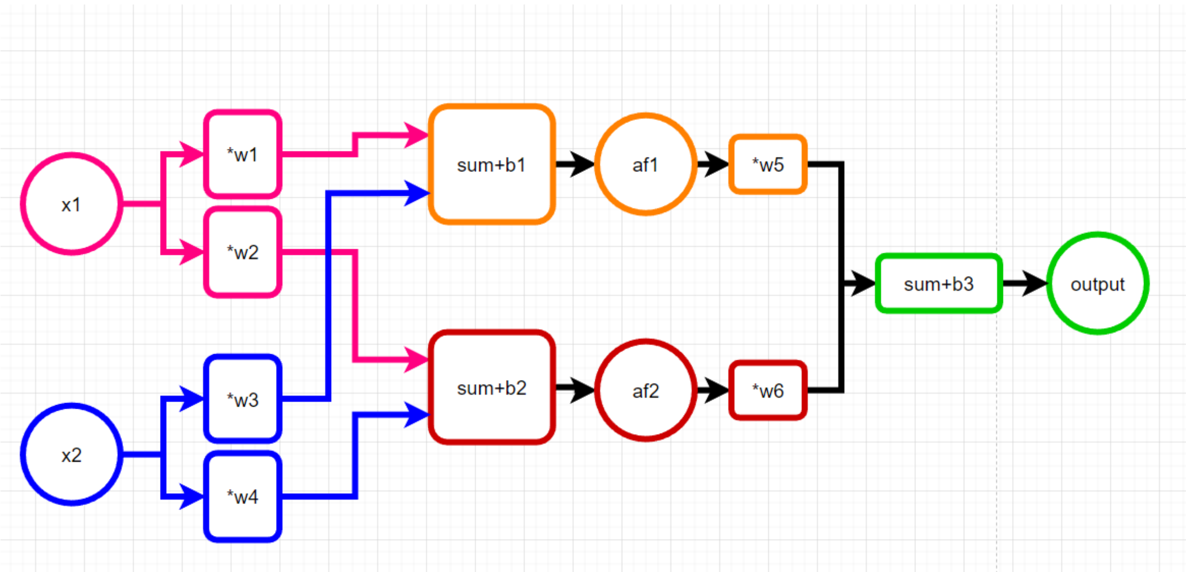
$\text{New value} = \text{Old value} - \text{Step size}$

不斷更新直到值變小到無法再小，或者步驟完成。

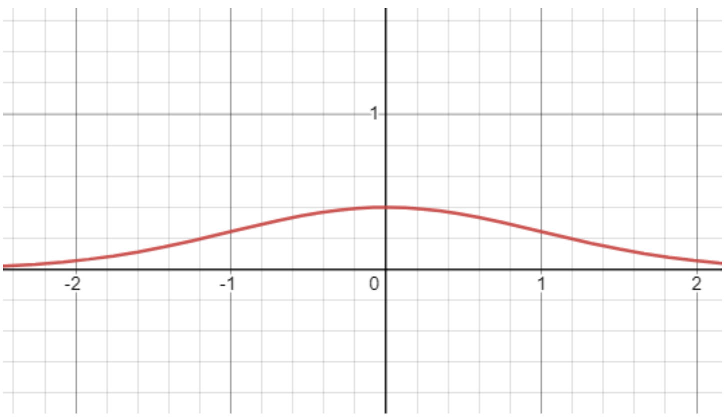
建構 2-input ANN

建立概述

我們以複雜的例子來說明，以下圖為例。



對於 weight 的部分，我們的初始值可以先給定為**標準常態分布**的隨機一個值，而 bias 可以先預設為 0



2-input ANN 的參數優化 (Backward Propagation)

若要使曲線越來越擬合我們的資料集，我們得要先考慮是否能夠限縮 SSR，使他越小，讓曲線 $f(x_i)$ 越能擬合資料集。

SSR 即為殘差平方和，即為對於在 $(x_1, x_2) = (i, j)$ 上的資料集，其所有資料 $(y(x_1, x_2) - f(x_1, x_2))^2$ 的和

可以定義為 $\text{SSR} = \sum\limits_{(x_1,x_2)\in S} (y(x_1, x_2) - f(x_1, x_2))^2$

若我們想要優化參數，可以找出參數對 SSR 的導函數，接著使用梯度下降法來同步優化所有參數。

我們令 $af_1(x_{1i}, x_{2i})$ 運算出的結果叫做 $y_1(x_{1i}, x_{2i})$，$af_2(x_{1i}, x_{2i})$ 運算出的結果叫做 $y_2(x_{1i}, x_{2i})$

可以得到 $f(x_{1i}, x_{2i}) = w_5 \times y_1 + w_6 \times y_2 + b_3$

又 $y_1 = \ln(1 + e^{x_1}), y_2 = \ln(1 + e^{x_2})$，以 Softplus 為例。

$x_1(x_{1i}, x_{2i}) = x_{1i} \times w_1 + x_{2i} \times w_3$，$x_2(x_{1i}, x_{2i}) = x_{1i} \times w_2 + x_{2i} \times w_4$

找出導函數，可以使用鎖鏈法則來尋找。

$$\frac{d\text{SSR}}{db_3} = \frac{d\text{SSR}}{df(x_1, x_2)} \frac{df(x_1, x_2)}{db_3} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times 1$$

$$\frac{d\text{SSR}}{dw_5} = \frac{d\text{SSR}}{df(x_1, x_2)} \times \frac{df(x_1, x_2)}{dw_5} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times y_1$$

$$\frac{d\text{SSR}}{dw_6} = \frac{d\text{SSR}}{df(x_1, x_2)} \times \frac{df(x_1, x_2)}{dw_6} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times y_2$$

$$\frac{d\text{SSR}}{db_1} = \frac{d\text{SSR}}{df(x_1, x_2)} \times \frac{df(x_1, x_2)}{dy_1} \times \frac{dy_1}{dx_1} \times \frac{dx_1}{db_1} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times w_5 \times \frac{e^{x_1}}{1 + e^{x_1}} \times 1$$

$$\frac{d\text{SSR}}{db_2} = \frac{d\text{SSR}}{df(x_1, x_2)} \times \frac{df(x_1, x_2)}{dy_2} \times \frac{dy_2}{dx_2} \times \frac{dx_2}{db_2} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times w_6 \times \frac{e^{x_2}}{1 + e^{x_2}} \times 1$$

$$\frac{d\text{SSR}}{dw_1} = \frac{d\text{SSR}}{df(x_1, x_2)} \times \frac{df(x_1, x_2)}{dy_1} \times \frac{dy_1}{dx_1} \times \frac{dx_1}{dw_1} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times w_5 \times \frac{e^{x_1}}{1 + e^{x_1}} \times x_{1i}$$

$$\frac{d\text{SSR}}{dw_2} = \frac{d\text{SSR}}{df(x_1, x_2)} \times \frac{df(x_1, x_2)}{dy_2} \times \frac{dy_2}{dx_2} \times \frac{dx_2}{dw_2} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times w_6 \times \frac{e^{x_2}}{1 + e^{x_2}} \times x_{1i}$$

$$\frac{d\text{SSR}}{dw_3} = \frac{d\text{SSR}}{df(x_1, x_2)} \times \frac{df(x_1, x_2)}{dy_1} \times \frac{dy_1}{dx_1} \times \frac{dx_1}{dw_3} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times w_5 \times \frac{e^{x_1}}{1 + e^{x_1}} \times x_{2i}$$

$$\frac{d\text{SSR}}{dw_4} = \frac{d\text{SSR}}{df(x_1, x_2)} \times \frac{df(x_1, x_2)}{dy_2} \times \frac{dy_2}{dx_2} \times \frac{dx_2}{dw_4} = \sum_{(x_1,x_2)\in S} -2(y(x_1, x_2) - f(x_1, x_2)) \times w_6 \times \frac{e^{x_2}}{1 + e^{x_2}} \times x_{2i}$$

接著使用梯度下降

$\text{Step size} = \text{Derivative} \times \text{Learning Rate}$

$\text{New value} = \text{Old value} - \text{Step size}$

不斷更新直到值變小到無法再小，或者步驟完成。

ANN 的優缺點

- 優點：
    1. 準確度高
    2. 可以處理很多種類的問題
    3. 可以包含很多種類的資料（數值、名目...）
    4. 可以得到非常好的 R-Score，只要訓練充足就可以
- 缺點
    1. 可能永遠訓練不完
    2. 黑箱，所以很難向別人描述這個原理
    3. 需要很大量的資料來訓練，才有準確度
    4. 對於多變量來說可能會讓訓練過程變得非常長