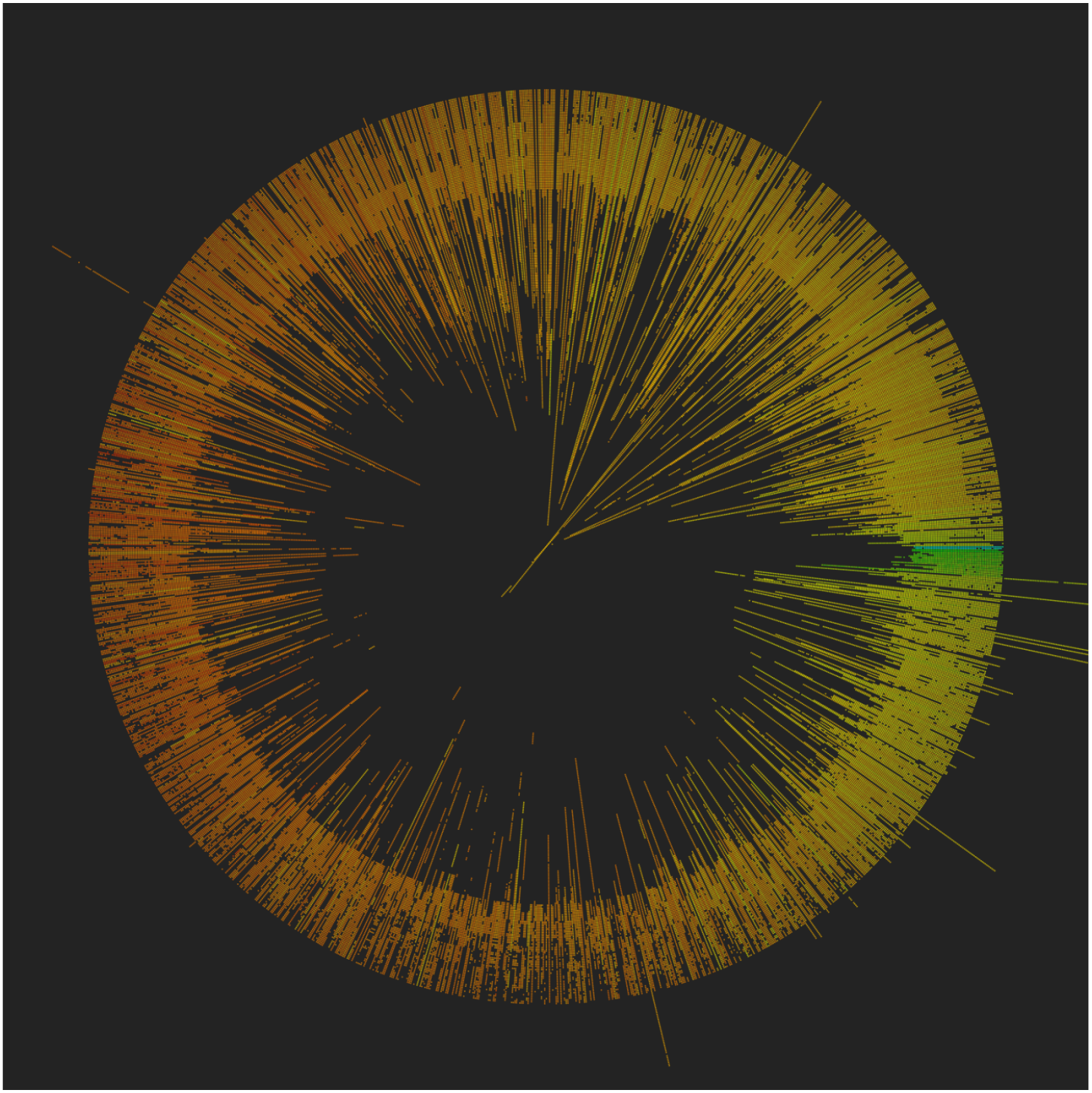
Epic

類神經網路-感知機的原理以及實作



寫在前面

這篇文主要是希望更多人能夠跨越機器學習的高門檻所寫作的文章，初衷是希望更多人可以了解目前人工智慧最具有希望的一條分支-機器學習-類神經網路的原理，如果文章內容有幫助到你(妳)順手幫我分享一下，可以幫助更多人和這個神祕的領域相遇歐!

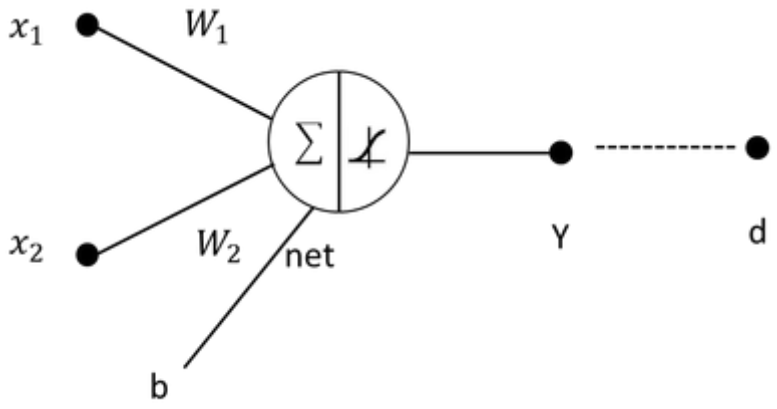
另外，如果有特別疑問的可以直接留言在底下或者直接FB找我!

什麼是「人工智慧」

人工智慧簡單來說就是模擬人類的智慧的一種技術，就方法而言，是橫跨資訊、數學、心理、生物、等多個領域的一門有趣的科學。人類智慧的現象裡有一種最常見的能力「經驗學習」也就是透過不斷的Try-Error來學會規則的方法，機器學習也就是透過數學的方法、並運用電腦程式來模擬「經驗學習」。

感知機Perceptron

感知機的概念與數學模型最早於1943被心裡學家與數學家合作的一篇論文(A logical calculus of the ideas immanent in nervous activity)所提出，目的是希望能夠模擬類似神經元的傳遞資訊的機制，感知機就是一顆神經元的運作的模擬。一個感知機原則上可以表示成以下架構:



感知機的基本架構

其中 x 是輸入信號，w 是感知機的權重，b是偏權值(也稱閥值)，net 是感知機的狀態，Y是感知機的輸出，d是期望的輸出值。

感知機的狀態數學的關係式可以表示成:

$$net=w_1*x_1+w_2*x_2+b$$

神經元狀態的計算

也就是:

感知機的狀態=加總所有的(輸入信號*對應權重)，直觀來說，一顆神經元的狀態取決於輸入信號的強度多寡與信號的重要性(權重)，不同種的輸入訊號有不同的重要性，而我們要決定的就是重要性的大小。

感知機的輸出的關係式可以表示成:

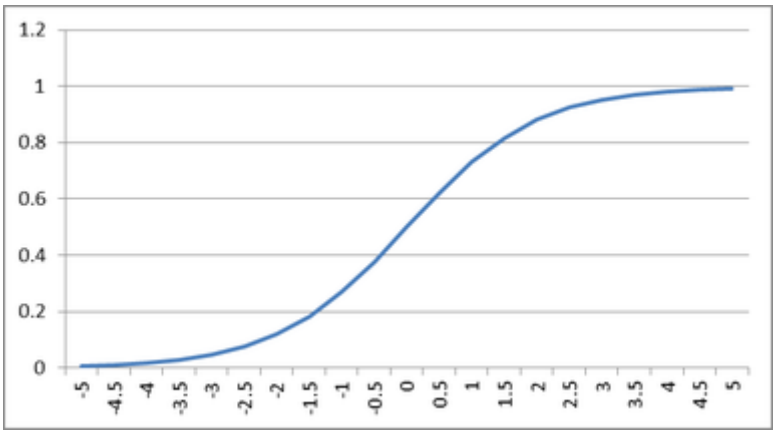
Y= Activation function(net)

Activation function 激活函數是一種非線性函數，用來將感知機的狀態值重新映射出去，在不同的類神經網路的架構裡有不同的激活函數，常見有sigmoid、ReLU等等

以Sigmoid函數為例子:

$Y= \frac{1}{1+e^{-net}}$ (e 是指自然數,值=2.718…)

Sigmoid函數



Sigmoid函數圖形

可以看得出來Sigmoid函數輸出介於0~1之間，輸入越大輸出越接近1，這種非線性的現象是在模擬神經元的傳導原則，當給予的刺激低於一定程度時，輸出電流趨近於0反之趨近於1。

用感知機來分類

感知機的行為大致上可以總結成:

1.輸入信號

2.信號加工(計算感知機狀態)

3.輸出信號

如果我們希望感知機能幫我們對一群數據做分類，我們可以利用輸出信號強弱來判斷，強(輸出值趨近於1)可以代表某樣本是某類別，弱(輸出值趨近於0)可以代表某樣本不是某類別。

感知機的失誤度量

為了衡量感知機輸出與期望輸出的失誤(loss)，我們用損失函數(loss function)來量化輸出預測與實際值的差異，最常見且本文使用的函數為: Square loss function 平方損失函數:

(目標輸出-實際輸出)^2

* loss function常見翻譯做「損失函數」，但我認為翻譯有歧義，應做「失誤函數」，loss function 的loss是指「失誤」，而非「損失」。

*Square loss function 通常會乘1/2方便取微分

感知機的學習

感知機的輸出信號取決於權重和閥值，因此我們希望透過經驗法則不斷的根據輸入以及期望輸出來修正權重和閥值，透過不斷重複迭代，讓輸出值逼近我們想要的輸出值。修正的方法通常利用梯度下降法來修正權重。

修正權重的量取決於損失函數對權重的偏微分，具體的細節可以參考我之間寫的另一篇文章: <Back Propagation with a Perceptron>。

具體而言的幾個實作感知機的重要公式:

公式編號	函數表示符號	實際計算方式	值表示符號	意義
1	Net(x)	$w_1*x_1+w_2*x_2+b$	net	神經元狀態值
2	Y(net)	$\frac{1}{1+e^{-net}}$	Y	神經元輸出值
3	D(Y)	d-Y	D	目標與實際數出差
4	E(D)	$\frac{1}{2}(D^2)$	E	誤差平方
$w_{1\,new}=w_{1\,old}-\eta\Delta w_1$ ， Δw_1 =(D)*-1* Y*(1- Y)* x_1 ，η = 學習速率 $w_{2\,new}=w_{2\,old}-\eta\Delta w_2$ ， Δw_2 =(D)*-1* Y*(1- Y)* x_2 ，η = 學習速率 $b_{new}=b_{old}-\eta\Delta b$ ， Δb =(D)*-1* Y*(1- Y)，η = 學習速率				

感知機的實作程式碼

```
1  public class Neural//神經元
2  {
3      private double [] weight;//神經元權重組態
4      private double bias;//神經元偏權組態
5      public Neural(int dimension)//建構子 ,感知機的維度
6      {
7          weight=new double [dimension];
8          bias=0;
9      }
10     public double []getWeight()
11     {
12         return weight;
13     }
14     public double getBias()
```

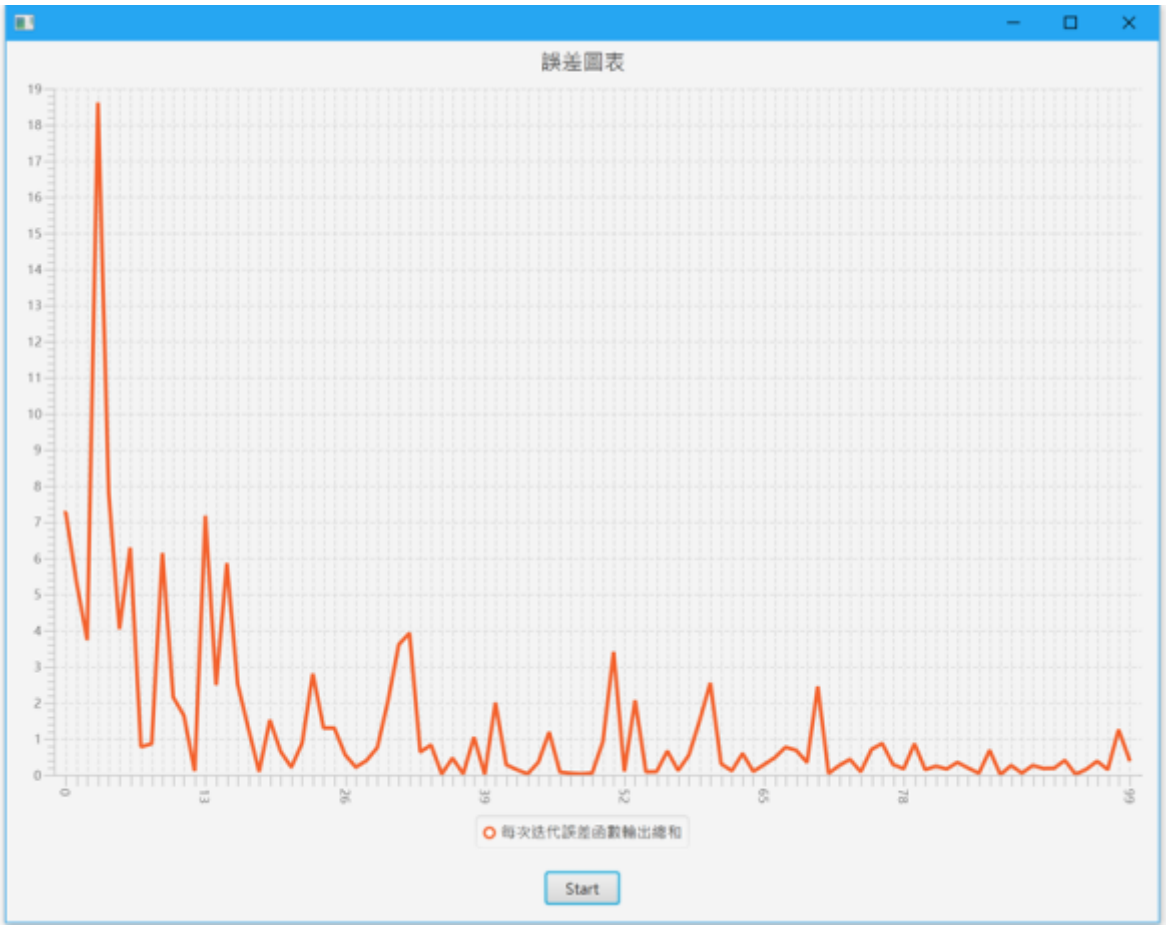
```
    {
        return bias;
    }
    private  double Net(double [] weight,double [] xi,double bias)//神
    {
         double net=0;
         for(int i=0;i&lt;xi.length;i++)
         {
           net+= xi[i]* weight[i];//加總(每個神經元權重*輸入)
         }
         net+=bias;//最後加篇權值
         return net;
    }
    private  double Y (double net)//神經元輸出
    {
        return Sigmoid(net) ;
    }
    private   double Sigmoid(double x) //Sigmoid函數
    {
         return (1/( 1 + Math.pow(Math.E,(-1*x))));
    }
    private double D(double d,double Y)//誤差
    {
        return d-Y;
    }
    private  double E(double D)//平方誤差
    {
        double E=0.5*D*D;
        return E;
    }

    public  double Iterate(double [] xi,double d,double learningRate),
    {
        //---前向傳遞
        double net=Net( weight,xi,bias);
        //System.out.println("net"+net);
        double Y=Y(net);
        //---

        //---誤差計算
        double D=D(d,Y);
        //---

        //---平方誤差(誤差函數)
        double E=E(D);
        //System.out.println(E);
        //---

        ////誤差倒傳遞

        //---對每個weight做更新
        for(int i=0;i&lt;weight.length;i++)
        {
             double deltaWeight=D*-1*Y*(1-Y)*xi[i];//weight 修正量
             weight[i]= weight[i]-learningRate*deltaWeight;//更新
        }
        //---

        //---對bias做更新
        double deltaBias=D*Y*(1-Y)*-1;//bias 修正量
        bias=bias-learningRate*deltaBias;//更新
```

```
76          //---
77
78          ////---
79          return E;
80      }
81
82  }
```

感知機訓練數據

感知機訓練可以使用Anderson’s Iris data set鳶尾花數據集，這個數據集是統計學、機器學習常使用的一數據集，我這裡有一份把setosa種類標記為1，versicolor標記為0的數據，總共一百筆資料。利用這個數據來進行感知機的分類訓練。可以在這邊下載!

感知機誤差降低的過程



以上的圖表是經過100次迭代產生的誤差圖表，橫軸是迭代的第x次，縱軸則是該次的誤差函數輸出總和，可以發現梯度下降法的誤差逐漸降低，也就是輸出逐漸逼近我們想要的值，當低於一定程度時，我們成為收斂，也就是感知機已經透過經驗學習到什麼樣的輸入模式該給出什麼樣的輸出。

結語

類神經網路是非常靈活自由的一種技術，從激活函數、誤差函數以及網路的連結架構(例如 多層感知機)、層數、神經元單元的變化(例如LSTM)，基本上可以隨意搭配，只要能有效解決問題即可。而梯度下降法的更新法則會因為不同的網路結構而改變，因此如果想要實做自己的類神經網路，是有必要熟悉偏微分以及鏈鎖律，才能推算出權重的更新規則。

後記

難得寫一篇稍微完整一點的機器學習教學文，今天剛好是情人節，因此這篇文章就獻給我的女友囉哈哈XD也祝福各位情人節快樂~

whuang022

 2017 年 **02** 月 **14** 日
類神經網路**,** 資料探勘**,** 人工智慧

類神經網路**,** 感知機**,** 梯度下降法

15 thoughts on “類神經網路-感知機的原理以及實作”

1. 王懿博

   2017 年 02 月 14 日 at 13:19:15

   謝謝你的分享，寫的很詳細與深入淺出的。

   - **whuang022**

     2017 年 02 月 14 日 at 14:54:45

     謝謝你!

2. **schub**

   2017 年 02 月 16 日 at 12:01:57

   您好，感謝分享，有個問題想請教您。

   原始的鳶尾花數據，
   可以表示為 [100][4] 的 double[][]，(一百筆，每筆四個數據)

而鳶尾花的輸出數據，
可以表示為 [100] 的 double[]，
亦可表示為 [100][1] 的 double[][]，(一百筆，每筆一個數據)

但程式碼的 Iterate 迭代部分，
輸入的 xi 資料為 double[]，d 為 double，緯度好像都少了一緯？

```
// 一次迭代 ,輸入,期望輸出 回傳本次迭代誤差函數輸出
public double Iterate(double[] xi, double d, double learningRate)
```

若是在每一個迭代才放入一筆鳶尾花數據，感覺好像也不太對，
請問在這樣的架構底下，要如何把輸入資料放入 Net 中才對呢？

```
m_iIteration = 100;
for (int i = 0; i < m_iIteration; i++)
{
error = NN.Iterate(Input[i], Output, LearningRate);
}
```

- **whuang022**

  2017 年 02 月 16 日 at 17:43:43

  您好歐,這個感知機由於是Sigmoid通常我們都只會拿來做二分法的分類,我示範的是把鳶尾花的前兩類拿出來做分類,所以是"是某類 d=1″,"不是某類 d=0″,我跑的數據有附在原文裡面,xi[]則是輸入的特徵,以這個例子而言是xi[4],
  實際上我來有個DTO物件把訓練資料封裝起來,類似這樣:perception.Iterate(trainDatas.get(i).feature, trainDatas.get(i).d, 0.7);

  - **schub**

    2017 年 02 月 17 日 at 14:19:33

    您好，鳶尾花資料的確應有三個分類，資料總數是150筆，
    我查看csv檔時有發現您只取了原始數據中的前兩個分類來做範例，
    先前不明所以，原來是因為Sigmoid的原因，感謝說明。

    由底下這行程式來看，

    perception.Iterate(trainDatas.get(i).feature, trainDatas.get(i).d, 0.7);

    您似乎是在神經元每個迭代的過程，動態輸入一筆鳶尾花資料至神經元中，
    也就是神經元迭代的總圈數，會等於 trainDatas.Length，我的理解正確嗎？

    感謝！

    PS：
    先前想請教的問題其實是，您是先把資料全部放入網路中再開始迭代，
    還是在神經元迭代的過程一邊迭代一邊餵資料？

  - **whuang022**

    2017 年 02 月 17 日 at 17:19:04

    沒有錯歐,Iterate是一邊迭代一邊餵資料,有個訓練器會跑trainDatas.Length的迴圈,每輪有trainDatas.Length次的權重修正,跑幾輪視收斂程度而定,細講下去有批次梯度下降與隨機梯度下降,這個範例是後者 🙂

3. 陳民濤

   2017 年 02 月 16 日 at 13:46:28

   請問Xi的變數是在哪邊宣告的 它的意義是訓練範例嗎

   - 陳民濤

     2017 年 02 月 16 日 at 13:47:39

     痾抱歉我搞懂了

   - **whuang022**

     2017 年 02 月 16 日 at 17:44:40

     Xi是迭代函數傳入的變數 🙂

4. **Frixion**

   2017 年 02 月 20 日 at 10:09:11

   您好,請問Neural建構子中,感知機的維度要如何決定?

   Neural(int dimension)

   dimension是否即為資料維度,也就是此例的dimension固定為4? thanks.

   - **whuang022**

     2017 年 02 月 20 日 at 10:53:59

     是的沒錯歐 🙂

5. **Sion**

   2017 年 06 月 23 日 at 01:15:40

   您好，感謝您的分享，有個問題想要請教一下~
   關於權重跟閥值的初始值請問您是如何定義呢?
   是隨機產生嗎?如果是那有限定什麼範圍嗎?
   感謝。

   - **whuang022**

     2017 年 06 月 24 日 at 01:47:23

     初始權重的方法有隨機 或給0的 或是先用其他方式估計的都有這方面建議你可以參考 :http://www.jianshu.com/p/03009cfdf733

6. **sion3280**

   2017 年 06 月 23 日 at 09:39:37

   您好，感謝您的分享，有個問題想要請教一下~
   關於權重跟閥值的初始值請問您是如何定義呢?
   是隨機產生嗎?如果是那有限定什麼範圍嗎?
   感謝。

   - **whuang022**

2017 年 06 月 24 日 at 01:47:54

如同我在你另外一個帳號回應的 🙂

向上 ↑