

# 天氣現象的神秘函數

## 人工智慧作業一：實作類神經網路計算神秘函數

60708009E 資教碩三 李盈潔

### 一、簡介

#### 1. 資料選擇原因與介紹

第一次知道大數據的我的母親得知我這學期修習人工智慧後，讓我幫她製作一個可以預測降雨機率的程式，這樣她就可以知道哪天可以把衣服拿去頂樓曬。一開始我使用的dataset是「氣象資料開放平台 30天觀測資料-局屬地面測站觀測資料」，從中取「氣壓(帕)、溫度(C)、相對濕度(%)、風速(m/s)」作為輸入，以觀測到的真實降雨與否(1降雨、0無降雨)作為輸出，但卻發現不論怎麼訓練誤差都十分巨大。查詢了相關網站後，發現降雨與否的原因有太多是目前未知的因素在。(詳細請見六.困難點的「2. 無論訓練多少次，誤差仍然很大(解決辦法:更換dataset)」)

後來決定更換dataset為「鄉鎮天氣預報-臺北市未來1週天氣預報」，以「大安區」為主要訓練資料，然而「降雨機率」資料不完整，因此以「平均溫度、平均露點溫度、平均相對濕度、最大風速」為輸入，以「天氣現象」作為輸出。

#### 2. 資料來源

氣象資料開放平台—鄉鎮天氣預報-臺北市未來1週天氣預報  
<https://opendata.cwb.gov.tw/dataset/statisticAll/F-D0047-063>

#### 3. 資料詳情

輸入: 平均溫度(攝氏, 除以100, 以小數做為表示)  
平均露點溫度(攝氏, 除以100, 以小數做為表示)  
平均相對濕度(% , 以小數做為表示)  
最大風速(m/s, 除以10, 以小數做為表示)

輸出: 天氣現象(除以10, 以小數做為表示)

- 02: 晴時多雲
- 04: 多雲
- 08: 多雲短暫雨
- 10: 陰時多雲短暫雨
- 11: 陰短暫雨
- 其他代號詳見附錄一、天氣預報欄位說明

訓練資料: 從2021-03-16 18:00 至 2021-03-17 06:00

到2021-03-22 06:00 至 2021-03-22 18:00, 共12筆資料

測試資料: 2021-03-22 18:00 至 2021-03-23 06:00, 共1筆資料

#### 4. 原始資料

時間	平均溫度	平均露點溫度	平均相對濕度	最大風速	天氣現象
2021-03-16 18:00 至 2021-03-17 06:00	0.22	0.19	0.85	0.3	0.2
2021-03-17 06:00 至 2021-03-17 18:00	0.25	0.2	0.72	0.3	0.2
2021-03-17 18:00 至 2021-03-18 06:00	0.23	0.21	0.9	0.4	0.2
2021-03-18 06:00 至 2021-03-18 18:00	0.25	0.21	0.8	0.3	0.2
2021-03-18 18:00 至 2021-03-19 06:00	0.22	0.21	0.93	0.3	0.4
2021-03-19 06:00 至 2021-03-19 18:00	0.24	0.21	0.81	0.4	0.2
2021-03-19 18:00 至 2021-03-20 06:00	0.22	0.2	0.92	0.3	0.2
2021-03-20 06:00 至 2021-03-20 18:00	0.24	0.2	0.79	0.3	0.2
2021-03-20 18:00 至 2021-03-21 06:00	0.21	0.2	0.93	0.3	0.8
2021-03-21 06:00 至 2021-03-21	0.2	0.17	0.86	0.3	1.1

18:00					
2021-03-21 18:00 至 2021-03-22 06:00	0.17	0.15	0.9	0.3	1.1
2021-03-22 06:00 至 2021-03-22 18:00	0.18	0.13	0.73	0.4	1
2021-03-22 18:00 至 2021-03-23 06:00	0.16	0.11	0.73	0.4	0.2

## 5. 硬體規格與作業系統

### Aspire E5-573G

裝置名稱	LAPTOP-C1D9T0IB
處理器	Intel(R) Pentium(R) CPU 3825U @ 1.90GHz 1.90 GHz
已安裝記憶體(RAM)	4.00 GB
裝置識別碼	49C9108A-1D6D-4AEC-A312-133F5349D205
產品識別碼	00325-80000-00000-AAOEM
系統類型	64 位元作業系統，x64 型處理器
手寫筆與觸控	此顯示器不提供手寫筆或觸控式輸入功能

### Windows 規格

版本	Windows 10 家用版
版本	1909
安裝於	2019/10/12
OS 組建	18363.1379

## 二、一個神經元

### 1. 實作

程式碼: HW1\_1.py

主要是參考了網路上bias的作法並修正老師給的程式碼，將bias當作另一個輸入值一起訓練，最初將bias設為1一起進行訓練。上網查詢的過程中認識了learning rate (學習率，控制基於損失梯度調整神經網絡權值的速度，learning rate越小，損失梯度下降的速度越慢。) 因此也試著加入learning rate進程式碼中。

執行結果以不同Learning rate(以下簡稱L\_rate)及不同迭代次數(Generation, 以下簡稱G)比較訓練的Mean squared error(以下簡稱Train\_Mse)和測試的Mean squared error(以下簡稱Test\_Mse)、以及測試的輸出結果(Result, 正確答案為0.2)呈現。

執行結果：

	L_rate = 0.1			L_rate = 0.5			L_rate = 1		
G	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result
2	0.1655	0.0225	0.3501	0.1462	0.0681	0.4610	0.1406	0.0751	0.4742
20	0.1397	0.0762	0.4762	0.1381	0.0839	0.4897	0.1362	0.0869	0.4948
200	0.1360	0.0869	0.4948	0.1219	0.1125	0.5354	0.1077	0.1481	0.5849
2000	0.1076	0.1482	0.5849	0.0575	0.3951	0.8285	0.0414	0.5147	0.9174
20000	0.0414	0.5147	0.9174	0.0273	0.6247	0.9904	0.0257	0.6344	0.9964

討論：

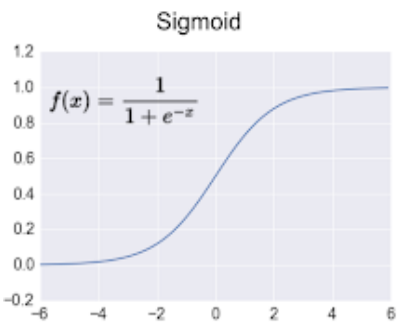
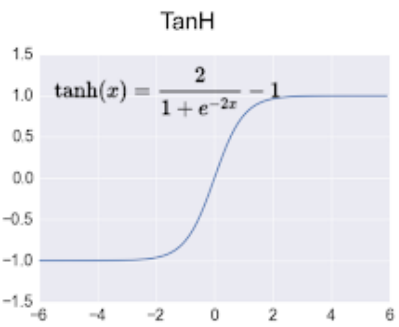
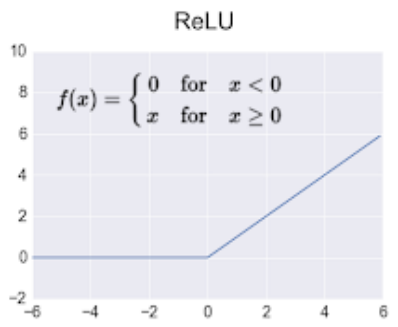
迭代次數增加，Train\_Mse都越小，Test\_Mse卻越大。可以發現，迭代次數越多，訓練結果就越接近訓練資料的答案，但卻離測試資料越來越遠。且L\_rate越大，Train\_Mse越小，Test\_Mse越大，和迭代次數相同。

當L\_rate = 1，且迭代次數為20000時，Train\_Mse有測試資料中的最小值0.0257；當L\_rate = 0.1，且迭代次數為2時，Test\_Mse有測試資料中的最小值0.0225，且Result為0.3501。

## 2. 不同的Activation functions間的比較

常見的激勵函數選擇有 sigmoid, tanh (雙曲正切函數), Relu (線性修正單元)：

	Sigmoid	Tanh	Relu
公式	$f(x) = 1 / (1 + \exp(-x))$	$f(x) = (1 - \exp(-2x)) / (1 + \exp(-2x))$	$f(x) = 0$ for $x < 0$ $f(x) = x$ for $x \geq 0$
函數	<pre>def sigmoid(x):     r = 1 / (1 + exp(-(x)))     return r</pre>	<pre>def tanh(x):     r = (exp(x) - exp(-x)) / (exp(x) + exp(-x))     return r</pre>	<pre>def relu(x):     r = maximum(0, x)     return r</pre>
導數	<pre>def dsigmoid(y):     return y*(1-y)</pre>	<pre>def dtanh(y):     return 1.0 - y**2</pre>	<pre>def drelu(y):     y[y &lt;= 0] = 0     y[y &gt; 0] = 1     return y</pre>

圖			
值	$0 < \text{output} < 1$	$-1 < \text{output} < 1$	0以上
優點	容易理解和應用	輸出以0中心	避免和糾正梯度消失問題
缺點	1. 梯度消失問題 2. 輸出不是以0為中心 3. 收斂緩慢	梯度消失問題	1. 只能在神經網絡模型的隱藏層中使用 2. 可能會導致死亡神經元

使用Tanh作為Activation function:

程式碼: HW1\_2.py

將Sigmoid的公式及導數進行修正, 並以函式的方式進程式碼改寫, 方便進行Sigmoid和Tanh的比較。

執行結果:

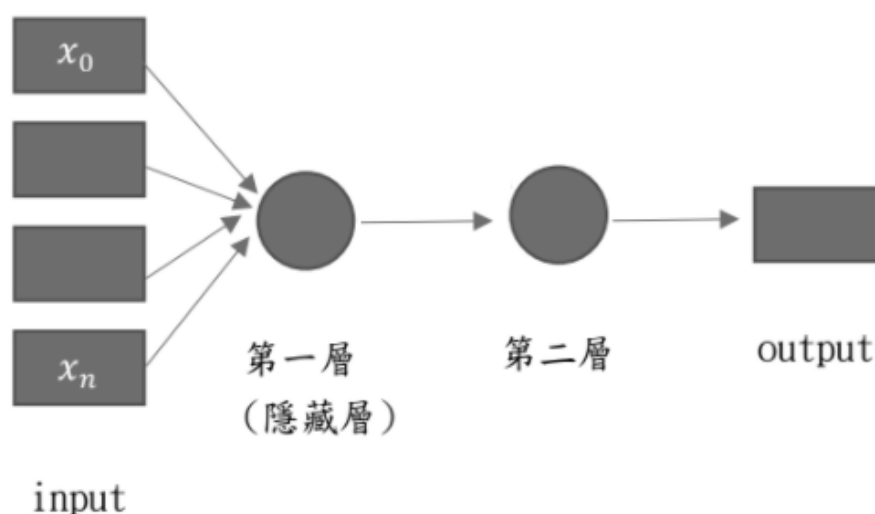
	L_rate = 0.1			L_rate = 0.5			L_rate = 1		
G	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result
2	0.1421	0.0624	0.4499	0.4083	0.6399	0.9999	0.4083	0.6399	1.
20	0.1343	0.0731	0.4704	0.4083	0.6399	0.9999	0.4083	0.6399	1.
200	0.1083	0.1222	0.5496	0.4083	0.6399	0.9999	0.4083	0.6399	1.
2000	0.0478	0.4452	0.8672	0.4083	0.6399	0.9999	0.4083	0.6399	1.
20000	0.0343	0.6127	0.9828	0.4083	0.6399	0.9999	0.4083	0.6399	1.

討論:

當L\_rate = 0.1時比Sigmoid的L\_rate = 0.1的Train\_Mse更小(Tanh 2000代 = 0.0343; Sigmoid 2000代 = 0.0414); 然而當L\_rate = 0.5以及以上時, 整個Activation function好像停擺了, 甚至當L\_rate = 1時, 輸出都成為1.。

### 三、兩個神經元

#### 1. 推導公式



將兩個神經元並排再一起，前面的神經元為隱藏層，後一個神經元為輸出層。在計算誤差時，要注意輸出層的誤差會傳遞到隱藏層。

#### 2. 實作

程式碼: HW1\_3.py

要注意將權重分為輸出層的權重和隱藏層的權重，輸入經過隱藏層神經元的Activation function激活輸出後，再當做輸出層神經元的輸入再激活一次。

執行結果：

	L_rate = 0.1			L_rate = 0.5			L_rate = 1		
G	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result
2	0.1436	0.0562	0.4371	0.1434	0.0580	0.4410	0.1432	0.0601	0.4453
20	0.1429	0.0599	0.4448	0.1419	0.0692	0.4631	0.1415	0.0743	0.4727
200	0.1415	0.0741	0.4722	0.1413	0.0800	0.4829	0.1413	0.0793	0.4817
2000	0.1413	0.0793	0.4817	0.1400	0.0703	0.4653	0.1326	0.0609	0.4469
20000	0.1326	0.0609	0.4469	0.1063	0.0860	0.4940	0.1037	0.0890	0.4983

討論：

兩個神經元明顯比一個神經元更好，雖然最終結果仍然不是0.2，但已經從0.9964下降至0.4983 (L\_rate = 1, 迭代20000次)；但是最佳值 (L\_rate = 0.1, 迭代2次) 卻從0.3511升至0.4371。且當L\_rate = 0.5時，迭代2000次的測試成果 (Test\_Mse = 0.0703, Result = 0.4653) 比迭代20000次的測試成果 (Test\_Mse = 0.0860, Result = 0.4940) 更好；同樣的，當L\_rate = 1時，迭代2000次的測試成果 (

Test\_Mse = 0.0609, Result = 0.4469)比迭代20000次的測試成果 (Test\_Mse = 0.0890, Result = 0.4983)更好。

### 3. 使用Tanh為Activation functions與Sigmoid的比較

程式碼: HW1\_4.py

執行結果:

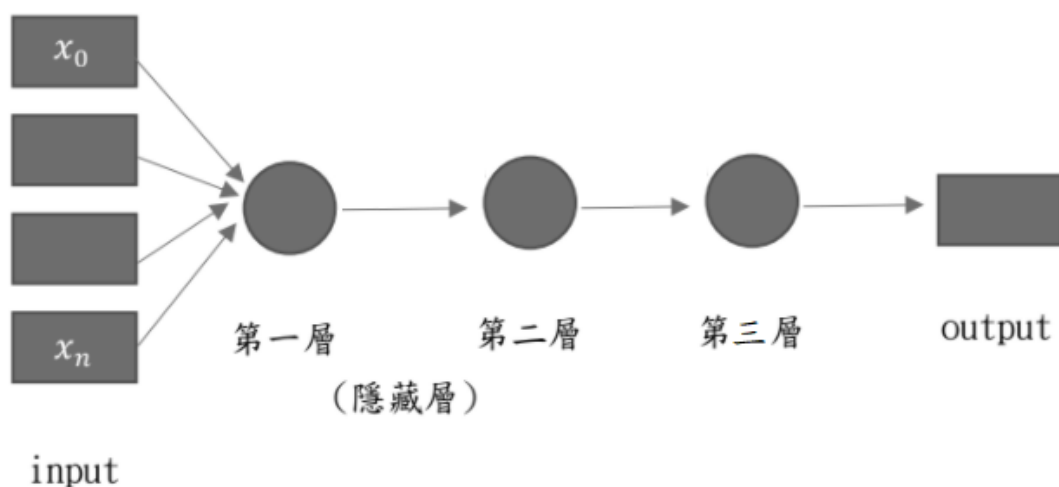
	L_rate = 0.1			L_rate = 0.5			L_rate = 1		
G	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result
2	0.1434	0.0746	0.4731	0.1440	0.1046	0.5234	0.1685	0.3070	0.7541
20	0.1426	0.0737	0.4716	0.1604	0.2425	0.6925	0.4027	0.6301	0.9937
200	0.1314	0.0873	0.4956	0.1835	0.0112	0.3060	0.4083	0.6399	0.9999
2000	0.0665	0.4109	0.8410	0.1674	4.5822e-06	0.1978	0.4083	0.6399	0.9999
20000	0.0522	0.4513	0.8718	0.1692	0.1911	0.6372	0.4083	0.6399	0.9999

討論:

在L\_rate = 0.5, 迭代2000次時有到目前為止(兩個神經元)最佳解Test\_Mse = 4.5822e-06, Result = 0.1978。Tanh作為Activation functions可以很明顯看出迭代次數並非越多越好, L\_rate也是並非越高越好, L\_rate = 1時梯度崩壞了。

## 四、三個神經元

### 1. 推導公式



和兩層的做法類似, 只是多增加了一個神經元在中間。

## 2. 實作

程式碼: HW1\_5.py

執行結果:

	L_rate = 0.1			L_rate = 0.5			L_rate = 1		
G	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result
2	0.1439	0.0544	0.4334	0.1436	0.0571	0.4389	0.1433	0.0602	0.4454
20	0.1429	0.0598	0.4446	0.1415	0.0736	0.4713	0.1413	0.0787	0.4806
200	0.1413	0.0784	0.4801	0.1413	0.0801	0.4830	0.1413	0.0801	0.4830
2000	0.1413	0.0801	0.4830	0.1413	0.0799	0.4826	0.1413	0.0793	0.4816
20000	0.1413	0.0793	0.4816	0.1010	0.0899	0.4999	0.0994	0.0899	0.4999

討論:

Sigmoid為Activation functions的算法, 看起來增加了第三層神經元也沒有和兩個神經元差太多, 且不同的L\_rate看起來也沒有差太多, 迭代次數也看起來沒有影響太多, 改變的幅度趨向於平緩。

## 3. 使用Tanh為Activation functions與Sigmoid的比較

程式碼: HW1\_6.py

執行結果:

	L_rate = 0.1			L_rate = 0.5			L_rate = 1		
G	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result	Train_Mse	Test_Mse	Result
2	0.4224	0.0433	-0.0080	0.1406	0.0378	0.3945	0.4064	0.6369	0.9980
20	0.1421	0.0756	0.4750	0.1560	0.1821	0.6268	0.3993	0.6226	0.9890
200	0.1409	0.0775	0.4785	0.1557	0.2549	0.7049	0.4083	0.6399	0.9999
2000	0.0961	0.1762	0.6198	0.1674	0.0233	0.3527	0.4083	0.6399	0.9999
20000	0.0627	0.4281	0.8543	0.1674	0.0233	0.3527	0.4083	0.6399	0.9999

討論:

看起來Tanh的梯度真的很容易壞, 但它比Sigmoid更能貼近測試結果。可以看得出來在L\_rate = 0.5的時候, 迭代200次和2000次有明顯的差別。在使用兩個神經元時, 也是L\_rate = 0.5, 迭代2000次時得到了最佳解, 看起來, 我所測試的資料在使用Tanh為Activation functions, L\_rate = 0.5, 迭代2000次可以得到最佳解。(不過只有一個神經元的時候梯度崩壞了。)



## 五、自由申論

1. 原以為迭代次數越多越好，但原來有「局部最佳解」這件事，所以迭代次數不一定是越多越好。而 Learning rate 也是，原以為是越高越好，但從 Tanh 的結果可以發現，如果權重一次跑太多，也會錯過「局部最佳解」，而且還有可能會導致梯度崩壞。
2. 天氣狀態除了這些因素（平均溫度、平均露點溫度、平均相對濕度、最大風速）之外，其實還有許多不為人知的因素在裡面，當然也可能是我的資料量太少。如果真的還要繼續訓練天氣狀態預報的神經元的話，我想我會從今天開始收集「今天的天氣預報」，收集各地的降雨機率、天氣狀態、濕度、風向等等資料，然後到隔天收集前一天的天氣資料進行比對與訓練。如果可以的話還是希望能夠找到過往的天氣預報，直接拿過去的資料來做就好，只是真的找不到。
3. 總覺得訓練一個神經元的時候很像統計課裡提到的迴歸線。我們利用已知資料畫出資料的回歸線後來預測下一筆資料，但有時候會預估錯誤，然後再進行回歸線的修正。
4. 以 Sigmoid 作為 Activation functions 看起來比以 Tanh 為 Activation functions 整體來說更有包容力（總覺得 Tanh 的梯度很容易壞？）；以 Tanh 為 Activation functions 訓練後的成果更好（以我的資料而言，尤其是  $L\_rate = 0.5$ ，迭代 2000 次時得到了最小誤差）。還是應該說以 Sigmoid 為 Activation functions 得到的改變比較平緩，而以 Tanh 為 Activation functions 得到的改變比較巨大，因此有機會得到最小誤差，但也有機會讓梯度崩壞？

## 六、困難點

### 1. 加入 bias

一開始覺得 bias 應該是最後再加入，然後思考很久是不是要將 bias 也用 random 的方式做，後來經過修過的學姊提示，再去查詢網頁，發現原來 bias 再時做的時候也是可以放入權重一起做改變的！假設輸入為 1，就可以跟著權重改變時一起訓練改變！

### 2. 無論訓練多少次，誤差仍然很大（解決辦法：更換 dataset）

原本使用的 dataset 是「氣象資料開放平台 30 天觀測資料-局屬地面測站觀測資料」，但發現不論將訓練次數增加到多少，誤差仍然很大，因此判斷降雨與否可能不僅只與氣壓、溫度、相對濕度、風速有關。查詢了「財團法人氣象推廣應用基金會-下雨定義與降雨機率」資料後發現，原來降雨與否的原因有太多是目前未知的因素在。因此天氣預報也只能指出「降雨機率」而不能肯定該時段是否會降雨。

真正的气象預報員會根據雲量、濕度、受天氣系統影響是否有降雨現象、天氣系統的移動速度、地形及過去的降雨量、降雨時數、範圍等，經過整理、分析、研判、討論之後，對某一個地區及某個時段內會不會降雨的把握是多少，這個「把握多少」換成百分比數，就是降雨機率。

曾想找過去的天氣預報觀看「降雨機率」，但發現查不到過去的降雨「機率」，因此將 dataset 更換成「鄉鎮天氣預報-臺北市未來 1 週天氣預報」。

### 3. 使用不同的 Activation functions 要更改導數

一開始改寫 Tanh 的程式時沒有改寫導數，以至於無論怎麼跑最後的結果都是 1，抱著這個疑問去詢問老師，老師告訴我我的輸入可能太大，導致梯度不見了。後來查資料才發現原來是導數的關係，導數改了以後就正常了一些。

#### 4. 兩個以上的神經元

兩個以上的神經元應該長什麼樣子呢？兩個還可以想像成一前一後，但三個要怎麼擺比較好？老師在課堂上有提示，其實神經元沒有一個固定的擺法，可以兩個先擺一起，再放一個擺在他們後面，也可以讓三個並排。

### 七、參考文獻

#### 1. 人工神經網路-維基百科

<https://zh.wikipedia.org/wiki/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C>

#### 2. 深度學習使用激勵函數的目的、如何選擇激勵函數 Deep Learning : the role of the activation function

<https://mropengate.blogspot.com/2017/02/deep-learning-role-of-activation.html>

#### 3. 什麼是激活函數？有哪些類型？哪個好用？別急，你要的這裡都有！-每日頭條

<https://kknews.cc/zh-tw/news/4mlnr82.html>

#### 4. 人工神經網路(2)--使用Python實作後向傳遞神經網路演算法(Backproagation artificial neature network)

<https://arbu00.blogspot.com/2016/11/2-pythonbackproagation-artificial.html>

#### 5. 多層感知器與反傳遞演算法實作 - 使用 JavaScript+Node.js

<http://programmermagazine.github.io/201404/htm/focus3.html>

#### 6. Backpropagation(BP) 倒傳遞法 #3 貓貓分類器-N層類神經網路

<https://www.brilliantcode.net/1527/backpropagation-3-n-layer-neural-networks/>

#### 7. 下雨定義與降雨機率

<http://www.metapp.org.tw/index.php/weatherknowledge/39-rainfall/56-2008-12-25-08-45-32>