

我的小小AI 天地

歡迎光臨Darwin在痞客邦的小天地 在這裡我將分享各種AI的工具與資訊 希望把AI的有趣透過部落格分享給大家 第一次來我部落格的可以到首頁我的網站學習地圖 裡面統整我部落格網站的資料 喜歡的話~幫小編多多衝人氣喔~ 這樣小編就更有毅力教大家 AI 囉~~

部落格全站分類：數位生活

Jan 09 Mon 2017 22:27

倒傳遞類神經網路(neural network backpropagation) 筆記

分享:    讚 103

類神經網路 筆記

小編看了很多的書籍介紹類神經的，發現這本講的最淺顯易懂而且還附有程式碼，我認為電腦科學與其他物理學、電磁學...等最不同的點在於他是可以看出效果的，是感覺的到的，就算不太懂他的數學原理，反覆地去看程式碼再回來對照公式很快就可以更深入的了解，但是類似電磁學那種理論，即使你經由公式算出了電場是多少，電荷是多少，我依然沒有什麼感覺因為他看不到也摸不到，但電腦科學就不同了，你可以經由寫程式透過軟體去驗證你的理論正不正確，下面就帶各位一步一步地進入類神經的科學理論以及用個實際的例子說明這理論要如何使用。

PS: 小編找了很久的word 轉成部落格資料，一直遇到公式無法直接轉換問題，終於有天讓我找到了pandoc無痛轉法，但是圖片卻無法轉，至使我還要一個一個貼，如果有網友知道word轉部落格的方法的話，歡迎通知小編我，小編會感激不盡的，



類神經網路與模糊控制理論入門與應用(附範例程式光碟片)

類神經生物模型

當今最神秘的器官不外乎就是我們人類的大腦，就算在21世紀的今日我們還是對大腦還是有許多不解的地方，因此許多科學家就想解開大腦的奧秘，當中一些電腦科學家就想把腦神經的運作原理透過數學的方式推導出一套可行的演算法出來，因此第一代類神經就這樣產生了。

各位可以先看這部影片，看看我們腦中的類神經是如何運作的，有助於理解模型。

活動快報

嬌蘭臨櫃試香體驗



全新推出的 Mon Guerlain
《我的印记》牡丹淡香

個人資料



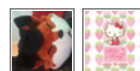
暱稱：Darwin的AI天地

分類：數位生活

好友：共2位
(看全部)

地區：台南市

我的好友



顯示共 2 名好友

熱門文章

(120126)python 新手的救星--Anaconda介紹與安裝

(47966)tensorflow教學(1) -- 建置一個基礎神經網路

(19385)Windows 安裝 Tensorflow

(18444)倒傳遞類神經網路(neural network backpropagation) 筆記

(17918)Deep learning model--caffe 使用教學



下圖為一個典型的類神經圖解，人的腦神經系統分為下面幾個構造：

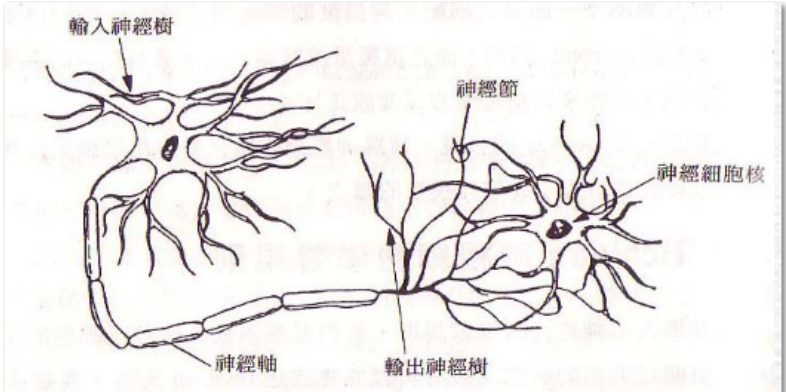


圖 1-1

圖1中繪有兩個神經細胞，每個神經細胞主要由：（1）神經細胞核（Soma）、（2）神經軸（Axon）、（3）神經樹（Dendrites）、（4）神經節（Synapses）等四部分構成。底下，我們依序介紹此四部份。

神經細胞核（Soma）

它是神經細胞的中心體，它的作用，目前並沒有完全徹底的了解，大概是將神經樹收集到的信號，在此作加總後再作一次非線性轉換，再由神經軸將信號傳送到其它的神經細胞中。

神經軸（Axon）

連接在神經細胞核上，用來傳送由神經細胞核產生的信號至其它的神經細胞中。

神經樹（Dendrites）

神經樹分為兩種：輸入神經樹及輸出神經樹。在圖1中左邊接到神經核的神經樹是用來接收其他神經細胞傳來的信號，稱為輸入神經樹。而在圖1右側接到神經軸的神經樹是用來傳送信號至其它神經細胞，稱為輸出神經樹。所以我們可以說：神經樹是神經細胞呈樹枝狀的輸出入機構。

神經節（Synapse）

輸入神經樹和輸出神經樹相連接的點稱為神經節，如圖1中以小圓圈框起來的接點即是。每個神經細胞大約有1000個神經節。神經節是神經網路上的記憶體，它表示兩個神經細胞間的聯結強度，我們將此聯結強度以一個數值來表示，並稱之為加權值（weight）。

一般言之，當神經網路在進行學習時，外界刺激神經細胞所產生的電流會去改變神經節上的加權值，在學習過程中，外界刺激所產生的電流反覆在神經網路上流動，神經節上的加權值也反覆地改變，最後會慢慢的趨向穩定，此時即表示學習已告完成。若神經網路是處於認知或辨識的過程中，由外界刺激所產生的電流，在進入神經網路後，會與貯存在神經節上的加權值作簡單的運算處理，若處理後的信號為可辨識的信號，則外界事物便是神經網路可認知或辨識的事物了。


類神經人工神經元模型

了解了生物神經細胞模型後，我們將介紹如何以人工神經元來模仿生物神經細胞。

讓我們再把相關的知識作個簡單的陳述，請參考圖1，當神經細胞透過輸入神經樹由其它神經細胞輸入脈波訊號後，經過神經細胞核的處理，其處理大約是將收集到的訊號作加總，再作一次非線性轉換後，產生一個新的脈波信號，如果這個訊號夠強，則新的脈波信號會由神經軸傳送到輸出神經樹，再透過神經節將此訊號傳給其它神經細胞。值得注意的是：當訊號經過神經節後，由於神經節加權值的影響，其訊號大小值會改變。經由上述的說明，我們提出人工神經元的模型如圖2所示。

- 文獻閱讀 (2)
- opencv (1)
- 生活紀事、新聞 (3)
- 我的作品 (5)
- AI數學 (6)
- python (9)
- Linux (3)
- caffe (7)
- tensorflow (10)
- 未分類文章 (11)

最新文章

- Alexnet 讀後筆記
- Anaconda安裝package方法 (以opencv示範)
- Windows 安裝 Tensorflow
- Excel VBA 貪婪演算法實作
-  我的網站學習地圖

最新留言

- 03/01 版主回覆：
不太懂你的意思 可以打開應...
- 03/01 版主回覆：
navigator 是指？
- 02/28 ryan：
不好意思我下載完加入環境變...
- 02/28 bogi：
不好意思我裝完以後只有anaco...
- 02/26 版主回覆：
應該是沒差 我記得我安裝Linu...

動態訂閱

-  痞客邦站方公告
文章更新
6天前
-  痞客邦站方公告
文章更新
25天前
-  痞客邦站方公告
文章更新
1個月前
-  痞客邦站方公告
文章更新
1個月前
-  痞客邦站方公告
文章更新
2個月前
-  痞客邦站方公告
文章更新
超過3個月以上
-  痞客邦站方公告
文章更新
超過3個月以上
-  痞客邦站方公告
文章更新
超過3個月以上
-  痞客邦站方公告
文章更新
超過3個月以上

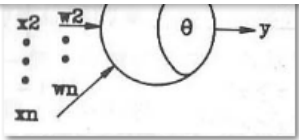


圖 2-1

在圖2中，每一個人工神經元皆有多個輸入 x_1, x_2, \dots, x_n 及一個輸出 y ，輸入值與輸出值的關係式，一般可用輸入值的加權乘積和的函數來表示，即

$$y(t) = f[\sum_{i=1}^n W_i * X_i(t) - \theta]$$

其中

- w_i = 模仿生物神經細胞的神經節加權值。
- θ = 模仿生物神經細胞的細胞核偏權值（bias），即輸入訊號的加權乘積和必須要大於偏權值後，才能被傳輸至其它人工神經元中。
- $f(\theta)$ = 模仿生物神經細胞的細胞核非線性轉換函數。
- t = 時間。
- n = 人工神經元輸入數目。

常用的非線性轉換函數

Sigmoid

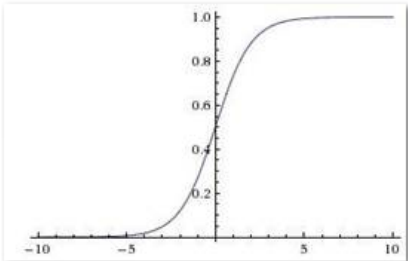


圖 3-1

Tanh

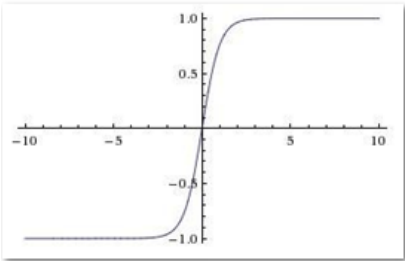


圖 3-2

ReLU

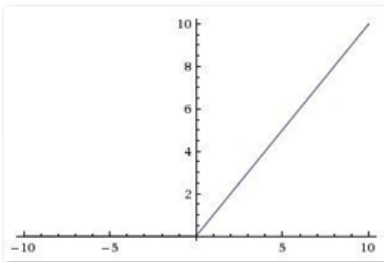
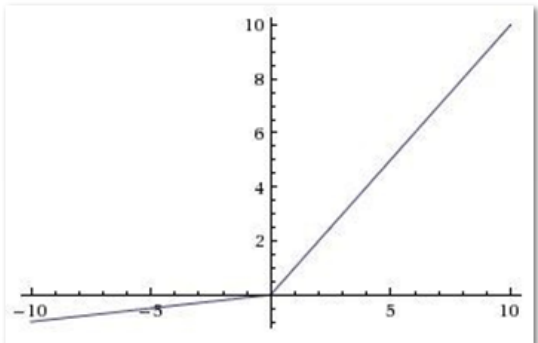


圖 3-3

Leaky ReLU



超過 4 個月未上

所有訂閱

文章精選

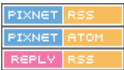
文章精選

所有文章列表

文章搜尋

搜尋

新聞交換(RSS)



誰來我家



參觀人氣

本日人氣：496
累積人氣：383251

QR Code



POWERED BY



(登入)

公式

站方公告

- [公告] 痞客邦『新發文工具』上線囉！快來分享你的第一篇『新型態文章』吧！
- [公告] 2018年度農曆春節期間服務公告
- [公告] PIXNET「客服系統」1月24日進行改版升級，暫停服務通知

單層感知機(perceptron)

此種類神經網路是經由F.Rosenblatt在1957年所提出，為最簡單形式的前饋神經網路，是一種二元線性分類器，當初他之所以會提出此種網路，是希望以此網路來模仿動物的大腦及視覺系統。雖然最初被認為有著良好的發展潛能，但感知機最終被證明不能處理諸多的模式識別問題。1969年，人工智慧之父Marvin Minsky和Seymour Papert在《Perceptrons》書中，仔細分析了以感知機為代表的單層神經網路系統的功能及局限，證明感知機不能解決簡單的XOR等線性不可分問題。

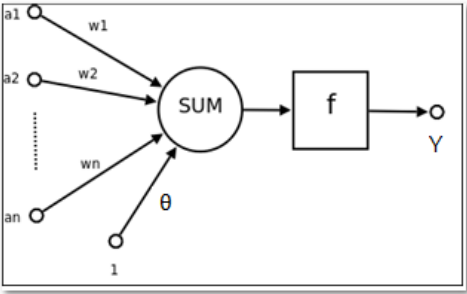


圖 4-1
單層感知機的數學模型如圖4-1所示，其中ai表示網路的輸入信號，Wi表示網路的加權值，θ為網路的偏權值，而其輸出方程式為：

$$Y = \begin{cases} 1, & \text{if } \sum_i a_i W_i \geq \theta \\ 0, & \text{if } \sum_i a_i W_i < \theta \end{cases}$$

因此轉換函數為Step function

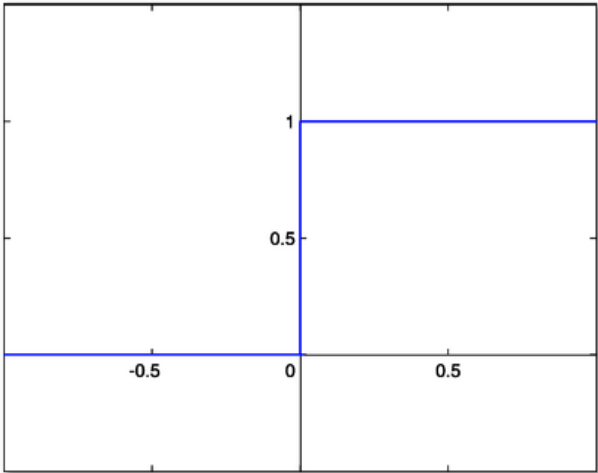


圖 4-2
單層感知機早期應用在印刷字體的識別，如果輸入樣本是線性可分的，那麼單層感知機便可以成功地做分類，如以下例子。底下我們實現一個AND邏輯閘的單層感知機，我們將網路加權值設為 W1=1，W2=1，θ=1.5。

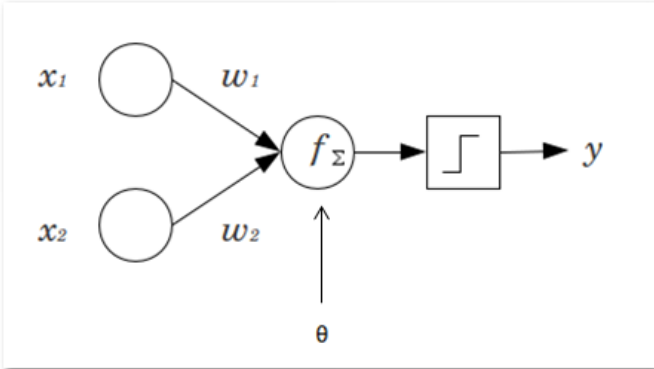


圖 4-3
我們簡單的套用真實表來檢視這組參數，首先測試第一組X1=0,X2=0，代入公式後呈現

$X_1W_1 + X_2W_2 - \theta = -1.5 < 0$
 $F_{step}(-1.5)=0$
繼續測試第二組 X1=1,X2=1，代入公式後呈現

$X_1W_1 + X_2W_2 - \theta = 0.5 > 0$
 $F_{step}(0.5)=1$
各位可以繼續的測試其他兩樣，做出來也都是同樣符合真實表的描述，如果以X軸當資料X1，Y軸當作資料X2畫圖出來的話，可以發現我們所設的參數所描述的就是一條直線方程式，所以只要是畫在平面圖上且資料呈現線性可分



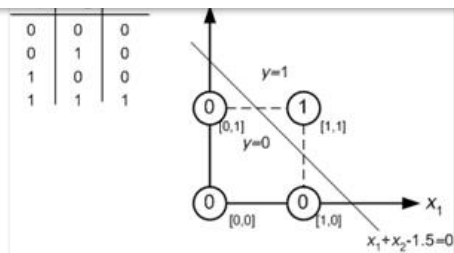


圖 4-4

讓我們再來看一個例子，我們來找可以使得感知機可以解決Xor邏輯閘的參數。

根據第一組資料，我們可以推導出式子如下，因此 $\theta > 0$

$$X_1 = 0, X_2 = 0 \rightarrow X_1 W_1 + X_2 W_2 = 0 < \theta \quad (3-4-1)$$

再根據第二組資料，推出 $W_2 > \theta$

$$X_1 = 0, X_2 = 1 \rightarrow X_1 W_1 + X_2 W_2 = W_2 > \theta \quad (4-2)$$

繼續第三組資料，推出 $W_1 > \theta$

$$X_1 = 1, X_2 = 0 \rightarrow X_1 W_1 + X_2 W_2 = W_1 > \theta \quad (4-3)$$

最後一組推出 $W_1 + W_2 < \theta$

$$X_1 = 1, X_2 = 1 \rightarrow X_1 W_1 + X_2 W_2 = W_1 + W_2 < \theta \quad (4-4)$$

由 (3-4-2) (4-2) (4-3) 我們得知 $W_1 + W_2 > \theta$ ，但是 (4-4) 卻推出 $W_1 + W_2 < \theta$ ，因此我們無法找到一組解使得單層感知機解決 Xor 問題，所以從此第一代類神經走入第一個寒冬中，因為它連Xor這麼簡單的問題都解決不了。

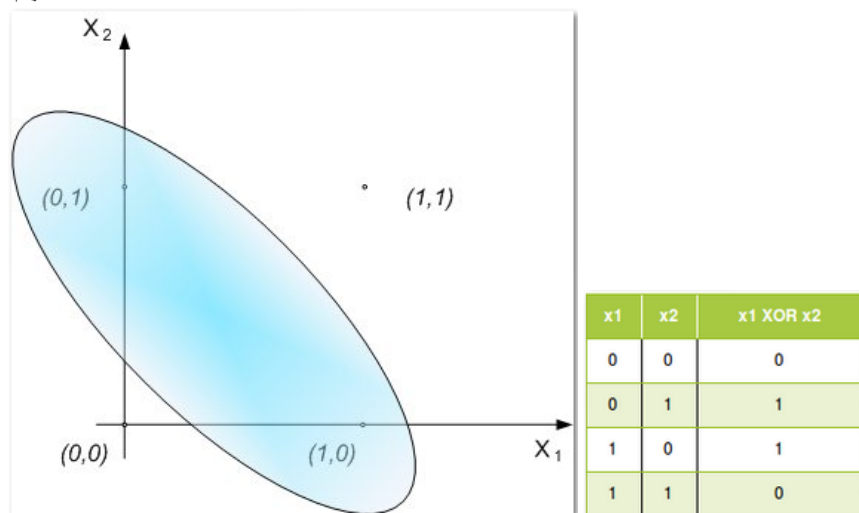


圖 4-5 圖 4-6

加入隱藏層的感知機

雖然單層感知機無法解決Xor問題，但是有科學家後續發現加入一個隱藏單元就可以順利排除這困難，原因是加入一個隱藏單元及等於是增加了一維變數，它會形成類似如圖4-5的橢圓曲線來做分類。一個可行的網路架構如圖5-1所示，它的輸出方程式為

$$y' = \begin{cases} 1, & \text{if } X_1 + X_2 > 1.5 \\ 0, & \text{if } X_1 + X_2 < 1.5 \end{cases}$$

$$y = \begin{cases} 1, & \text{if } X_1 + X_2 - 2y' > 0.5 \\ 0, & \text{if } X_1 + X_2 - 2y' \leq 0.5 \end{cases}$$

我們仿造剛剛的運算，繼續將第一組數據帶入驗證測試，第一科神經元為

$$X_1 = 0, X_2 = 0 \rightarrow X_1 W_{11} + X_2 W_{21} - \theta_1 = -1.5 < 0 \rightarrow y' = 0$$

$$X_1 W_{12} + X_2 W_{22} - 2y' - \theta_2 = -0.5 < 0 \rightarrow y = 0$$

第二組數據為 $X_1 = 0, X_2 = 1$ ，第一二科神經元為

$$X_1 = 0, X_2 = 1 \rightarrow X_1 W_{11} + X_2 W_{21} - \theta_1 = -0.5 < 0 \rightarrow y' = 0$$

$$X_1 W_{12} + X_2 W_{22} - 2y' - \theta_2 = 0.5 > 0 \rightarrow y = 1$$

第三組

$$X_1 W_{12} + X_2 W_{22} - 2y' - \theta_2 = 0.5 > 0 \rightarrow y = 1$$

第四組

$$X_1 = 1, X_2 = 1 \rightarrow X_1 W_{11} + X_2 W_{21} - \theta_1 = 0.5 > 0 \rightarrow y' = 1$$

$$X_1 W_{11} + X_2 W_{22} - 2y' - \theta_2 = -0.5 < 0 \rightarrow y = 0$$

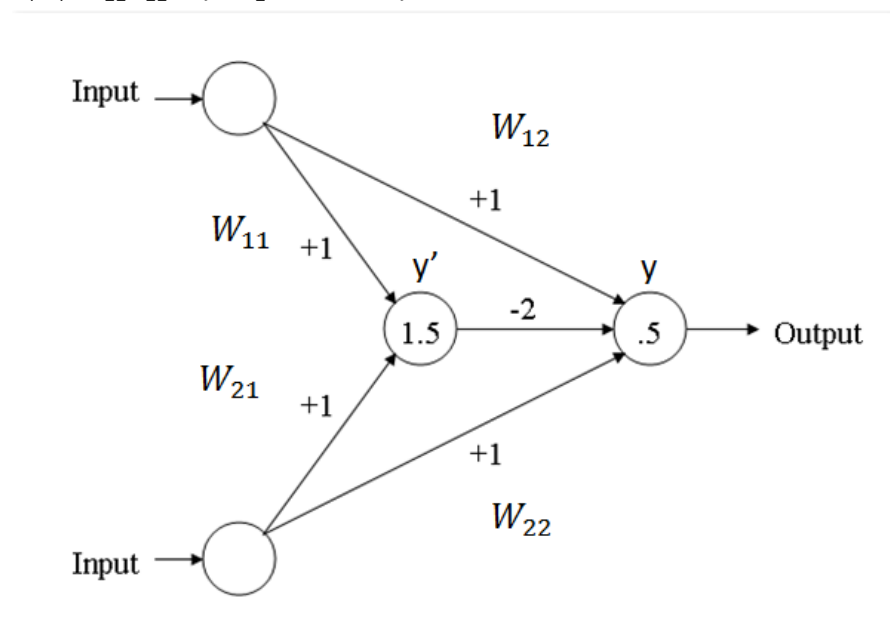


圖 5-1

我們可以將上列方程式簡單的用 **Matlab** 跑圖驗證一下，可以發現產生如圖5-2 的效果，紅色代表輸出為1，藍色代表輸出為0，多加了一顆隱藏神經元就會產生另一條直線來畫分。

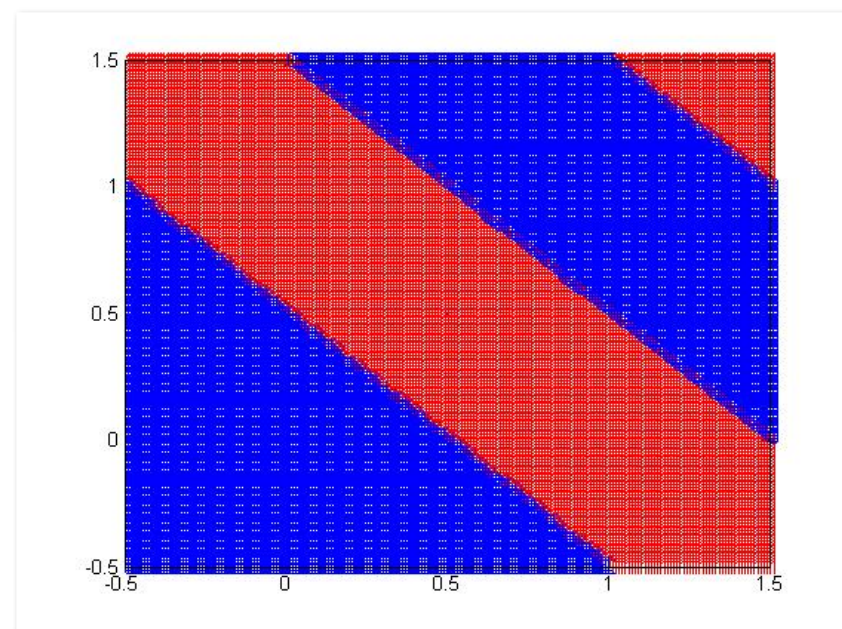


圖 5-2

後續的研究人員重複著類似的實驗，並把神經元做多項的變化，比較加一層以及加兩層神經元的效果圖，可以統計出一張圖表如下(摘自書中)，他們發現神經元加越多層或是神經元加越多顆，所能區分的圖形就更複雜、圖形就更多樣化。


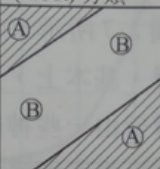
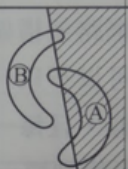
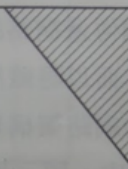

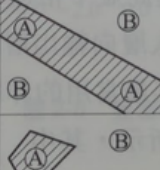
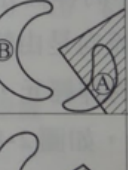
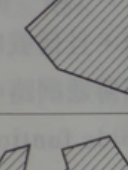

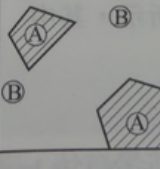

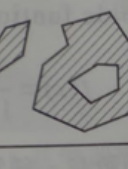
網路結構	分類區域類型	對互斥或(XOR)分類	網狀分類區域	分類區域一般形狀
無隱層 	由超平面分成二個區域			
單隱層 	開凸區域或閉凸區域			
2 個隱層 	任意形狀 (但與節點數之間的關係複雜)			

圖 5 -3

倒傳遞神經網路

倒傳遞神經網路(Back-propagation Neural Network) 是一種具有學習能力的多層前投型網路。此網路是由Rumelhart、McClelland在1985年所提出的，當初他們之所以會提出此種網路，是希望提出一種平行分布訊息的處理方法來探索人類認知的微結構。

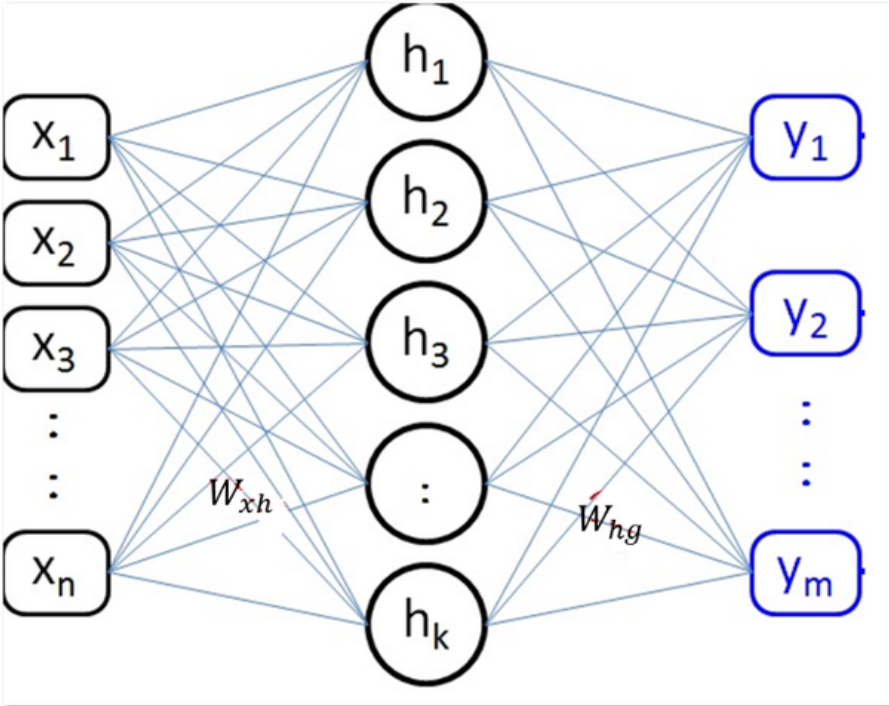


圖 6 -1

簡單手算版

我找尋許多資料發現大多數講解倒傳遞類神經模型時都是用一堆數學公式，的確，類神經是經由精巧的微積分並運用微積分的連鎖律，一層一層地從後面傳遞誤差回去，因此我們就可以求出神經元權重的改變量，但直接看公式的話會很容易陷入迷思，因此我在網路上找到一份手算倒傳遞網路的資料 [6]，我把它翻譯成中文，相信經過這個例子的說明後再去看公式會容易許多。

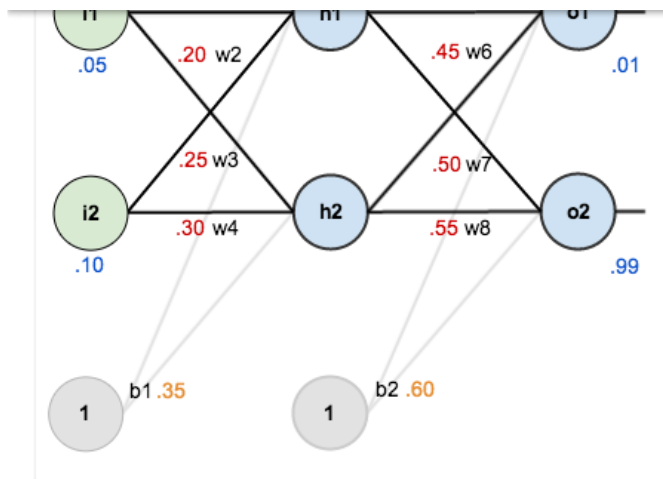


圖 6-2

我們先從簡單的模型開始講起，倒傳遞網路的连接方式與perceptron不同，它是採取全部神經元互相連接的方式，但是只有層與層的神經元互相連結，層之間的神經元不做溝通，旁邊的數字代表參數的初始值為了等一下方便計算，這裡的**b**就是相當於剛剛所講的 θ ，事實上，大部分類神經書籍文獻都用**b**表示，只是書中為了與人腦神經元模型相呼應因此改成 θ ，在這裡我們將**b**命名為 **bias**，並將以前的公式改為如下

$$y(t) = f\left[\sum_{i=1}^n W_i * X_i(t) + b\right]$$

假如我們這裡選的函數**f**是sigmoid函數，因此，這裡h1的輸出值就為

$$\begin{aligned} y_{h1} &= f(w1 * i1 + w2 * i2 + b1) = f(0.15 * 0.05 + 0.2 * 0.1 + 0.35) = f(0.3825) \\ &= \frac{1}{1 + e^{-0.3775}} = 0.6 \end{aligned}$$

用同樣的算法我們可以得出 $y_{h2} = 0.596884378$

接下來以同樣的方法計算出o1, o2

$$\text{out}_{o1} = f(h1 * w5 + h2 * w6 + b2) = 0.75$$

$$\text{out}_{o2} = f(h1 * w7 + h2 * w8 + b2) = 0.773$$

這裡要注意的是作者似乎把**bias**項目簡化了，事實上每科神經元都各自有一個**bias**，但這裡作者為了計算方便將兩顆神經元的**bias**都設為一樣。

接下來我們要定義總體誤差為

$$E_{\text{total}} = \sum \frac{1}{2} (\text{target} - \text{output})^2$$

其中**target**就是我們所想得到的值，整個演算法運作的目的就是想讓我的輸出越接近**target**越好，平方是為了消除正負號所帶來的影響，0.5是為了後面微分方便，加總是指，假如輸出神經元有**N**個，那就是代表要將**N**科神經元的誤差都加總起來，所以類神經訓練到最後**E_{total}**會越來越小，代表神經元的輸出已經和我的樣本資料幾乎一模一樣了。

這裡輸出神經元有兩個，因此

$$\begin{aligned} E_{\text{total}} &= E_{o1} + E_{o2} = \frac{1}{2} (\text{target}_{o1} - \text{out}_{o1})^2 + \frac{1}{2} (\text{target}_{o2} - \text{out}_{o2})^2 \\ &= \frac{1}{2} (0.01 - 0.75)^2 + \frac{1}{2} (0.99 - 0.773)^2 = 0.3 \end{aligned}$$

再來我們要用微積分的連鎖律求出**E_{total}**和**w5**的關係式，我們想要知道**w5**該如何調整能使我的**E_{total}**變得更小，因此我們可以求出 $\frac{\partial E_{\text{total}}}{\partial w5}$ ，找出微分關係式後就可以套用底下公式找出調整**w5**的方向，其中 α 稱作學習率。

$$w5_{\text{new}} = w5_{\text{old}} - \alpha \frac{\partial E_{\text{total}}}{\partial w5}$$

根據微積分連鎖率我們可以得知

$$\frac{\partial E_{\text{total}}}{\partial w5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} * \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} * \frac{\partial \text{net}_{o1}}{\partial w5}$$

其中**net_{o1}**指的是**f**函式中裡面的加總式子，也就是說

接下來就是分別算出各項偏微分，然後再將它們組合起來就可以得到 $\frac{\partial E_{total}}{\partial w_5}$

首先算出第一項 $\frac{\partial E_{total}}{\partial out_{o1}}$ ，回想起剛剛定義的 E_{total} 為

$$E_{total} = \frac{1}{2} (target_{o1} - out_{o1})^2 + \frac{1}{2} (target_{o2} - out_{o2})^2$$

因此

$$\frac{\partial E_{total}}{\partial out_{o1}} = \frac{1}{2} * 2 * (target_{o1} - out_{o1}) * -1 + 0 = -(0.01 - 0.75) = 0.74$$

再來計算第二項 $\frac{\partial out_{o1}}{\partial net_{o1}}$ ，回想起剛剛解釋的 net_{o1} ，因此

$$out_{o1} = f(net_{o1}) = \frac{1}{1 + e^{-net_{o1}}}$$

所以

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1} * (1 - out_{o1}) = 0.75 * (1 - 0.75) = 0.186$$

最後一項 $\frac{\partial net_{o1}}{\partial w_5}$ ，因為 $net_{o1} = h_1 * w_5 + h_2 * w_6 + b_2$ ，所以

$$\frac{\partial net_{o1}}{\partial w_5} = h_1 = 0.6$$

全部組裝再一起可以得到

$$\frac{\partial E_{total}}{\partial w_5} = 0.74 * 0.186 * 0.6 = 0.082$$

假設學習率為0.5，因此我們就可以得知

$$w_{5_{new}} = w_{5_{old}} - \alpha * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082 = 0.359$$

圖6-3為作者所畫的流程圖，這就是為什麼網路稱為倒傳遞神經網路的原因，從最後一層的Error項傳遞誤差到 out_{o1} ，然後再到 net_{o1} ，一直層層傳遞到 w_5 。

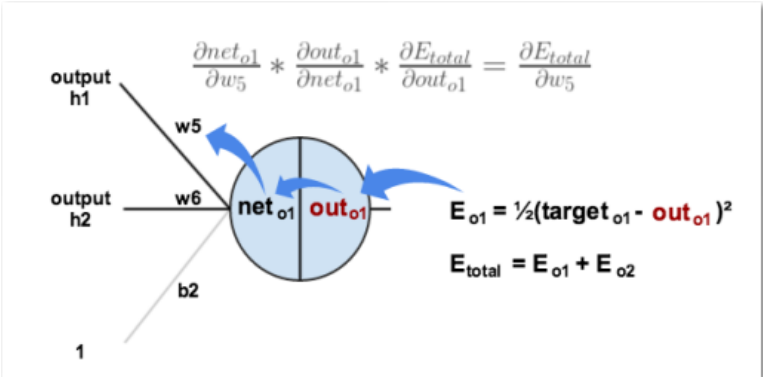


圖 6 -3
我們可以用同樣的方法算出其他神經元

$w_{6_{new}} = 0.4$
 $w_{7_{new}} = 0.51$
 $w_{8_{new}} = 0.56$

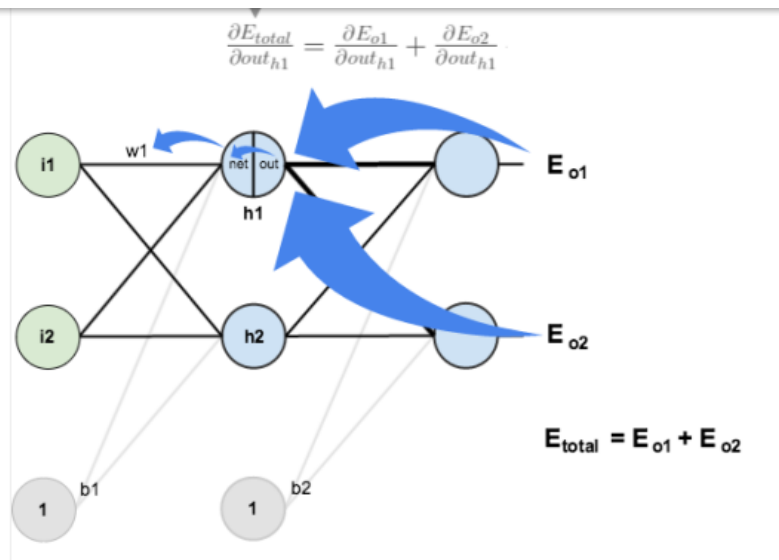


圖 6-4

再來我們要將Error繼續傳遞到 w_1 ，因此我們要算出 $\frac{\partial E_{total}}{\partial w_1}$
 我們一樣套用連鎖率概念可以得知

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

其中 $\frac{\partial E_{total}}{\partial out_{h1}}$ 為如下，因為改變 out_{h1} 對後面每一項的誤差皆有影響，因此必須把它們加總起來

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

我們開始像剛剛一樣逐個計算偏微分項目，從 $\frac{\partial E_{o1}}{\partial out_{h1}}$ 開始，我們可以套用剛剛已經求得的項目

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.74 * 0.18 * \frac{\partial net_{o1}}{\partial out_{h1}}$$

因為 $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$ ，所以

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.4$$

組合起來就得到

$$\frac{\partial E_{o1}}{\partial out_{h1}} = 0.74 * 0.18 * 0.4 = 0.0554$$

同樣的程序我們可以得到

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.02$$

因此

$$\frac{\partial E_{total}}{\partial out_{h1}} = 0.0554 - 0.02 = 0.0354$$

再來繼續算第二項 $\frac{\partial out_{h1}}{\partial net_{h1}}$

$$\because out_{h1} = \frac{1}{1+e^{-net_{h1}}} \quad \therefore \frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1} * (1 - out_{h1}) = 0.6 * (1 - 0.6) = 0.24$$

最後一項 $\frac{\partial net_{h1}}{\partial w_1}$

$$\because net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1 \quad \therefore \frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05 \text{ (也就是前一層的輸出)}$$

全部組裝再一起可以得到

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1} = 0.0354 * 0.24 * 0.05 = 0.0004248$$

因此

$$w_{2_{\text{new}}} = 0.19956143$$

$$w_{3_{\text{new}}} = 0.24975114$$

$$w_{4_{\text{new}}} = 0.29950229$$

有了前面的例子作簡介後，接下來我們要代入公式版的**backpropagation**推導，如果在公式版的有哪個地方看不懂，請反覆去回顧剛剛講的簡介，如此來回推敲多次相信很快就能理解的。

複雜公式版

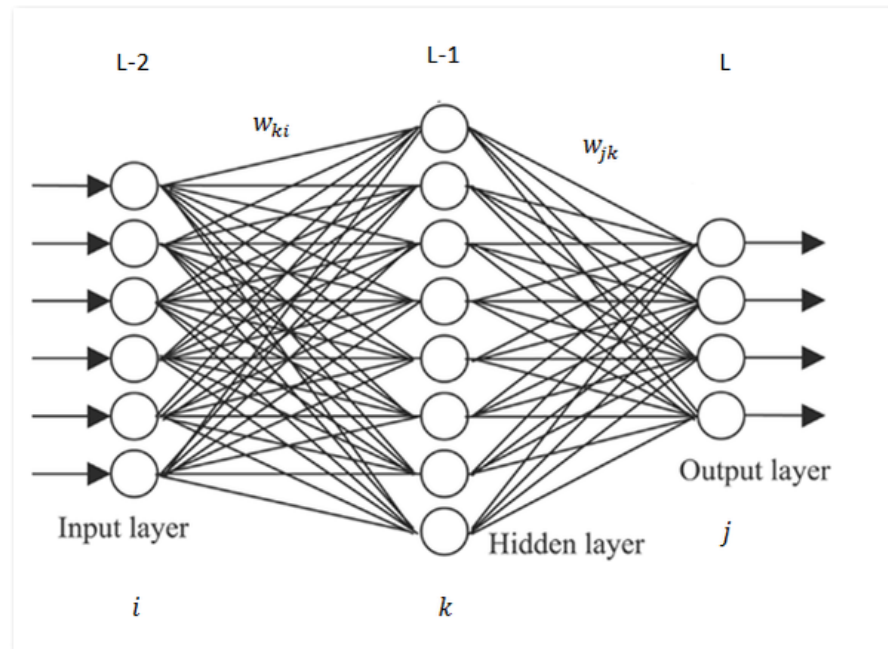


圖 6-5

$$a_j^L = f(\sum_k w_{jk}^L a_k^{L-1} + b_j^L) = f(z_j^L) \quad (6-2-1)$$

首先我們先定義各項參數， a_j^L 代表在第L層中，第j科神經元的輸出， w_{jk}^L 代表在模型中第L層和L-1層中的權重值，

b_j^L 代表在模型中第L層和L-1層中的偏值，k代表L-1層的神經元索引值。

$$E = \frac{1}{2} \sum_n [y_j - a_j^L]^2 \quad (6-2-2)$$

其中，L代表最後一層， a_j^L 代表在輸出層中第j科神經元的輸出， y_j 是代表第j個輸出的樣本，n代表最後一層神經元個數，我們將輸出層所有神經元誤差都加總起來，並加上平方項目以消除正負號所帶來的影響，最後除以2來當作整個神經網路的總體誤差。

我們首先求得總體誤差對輸出層權重質的微分，根據微積分的交換率我們可以改寫如以下公式(6-2-3)，我們分別對各個偏微分項做運算得到(6-2-4)(6-2-5)(6-2-6)

$$\frac{\partial E}{\partial w_{jk}^L} = \frac{\partial E}{\partial a_j^L} * \frac{\partial a_j^L}{\partial z_j^L} * \frac{\partial z_j^L}{\partial w_{jk}^L} \quad (6-2-3)$$

$$\frac{\partial E}{\partial a_j^L} = -(y_j - a_j^L) \quad (6-2-4)$$

$$\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial f(z_j^L)}{\partial z_j^L} = f'(z_j^L) \quad (6-2-5)$$

$$\frac{\partial z_j^L}{\partial w_{jk}^L} = \frac{\partial \sum_k w_{jk}^L a_k^{L-1} + b_j^L}{\partial w_{jk}^L} = a_k^{L-1} \quad (6-2-6)$$

將(6-2-4)(6-2-5)(6-2-6)組裝再一起我們可以得到(6-2-8)，其中 δ_j^L 稱作在L層第j顆神經元的敏感度，式子如(6-2-9)所示。

$$\frac{\partial E}{\partial w_{jk}^L} = a_k^{L-1} (a_j^L - y_j) f'(z_j^L) = a_k^{L-1} \delta_j^L \quad (6-2-8)$$

$$\delta_j^L = (a_j^L - y_j) f'(z_j^L) \quad (6-2-9)$$

再來我們求得總體誤差對隱藏層權重值的偏導數，我們一樣可以按照連鎖律將式子拆成如(6-2-10)所示，依照剛剛的方法分別運算可以求得式子(6-2-11)(6-2-12)(6-2-13)

$$\frac{\partial E}{\partial w_{ki}^{L-1}} = \frac{\partial E}{\partial a_k^{L-1}} * \frac{\partial a_k^{L-1}}{\partial z_k^{L-1}} * \frac{\partial z_k^{L-1}}{\partial w_{ki}^{L-1}} \quad (6-2-10)$$

$$\frac{\partial E}{\partial a_k^{L-1}} = \sum_j \frac{\partial E}{\partial a_j^L} * \frac{\partial a_j^L}{\partial z_j^L} * \frac{\partial z_j^L}{\partial a_k^{L-1}} \quad (6-2-11)$$

類神經網路與模糊控制理論 入門與應用 王進德 編著
<http://neuron.csie.ntust.edu.tw/homework/93/NN/homework2/M9304302/welcome.htm>
<http://iamtrask.github.io/2015/07/12/basic-python-network/>
[https://en.wikipedia.org/wiki/Hadamard_product_\(matrices\)](https://en.wikipedia.org/wiki/Hadamard_product_(matrices))
<http://neuralnetworksanddeeplearning.com/chap2.html>
<https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>
<http://cs231n.github.io/optimization-2/>

文章標籤

[backpropagation](#) [MLP](#) [neural network](#)

創作者介紹



Darwin的AI天地

我的小小AI 天地



Darwin的AI天地 發表在 痞客邦 PIXNET 留言(12)

E-mail轉寄



· 留言列表 (12)

發表留言

悄悄話

#2 MagicJackTing · 於 2017/01/16 10:16

Word 轉部落格的方法:

1. 現成的轉檔工具好像都無法完成, 因為轉檔工具不知道圖片檔上傳之後的 ID (URL 的一部份).
2. 有試過把文章貼在 Windows Live Writer 上, 再上傳: 圖片檔是可以上傳的, 不過有些東西的格式會跑掉, 例如: 有合併儲格的表格. 所以, 如果妳的文章轉貼 (copy-paste) 到 WLW 上看起來是 OK 的話, 可以用 WLW. 雖然操作上麻煩一些.
3. Pixnet 有提供 API 可以上傳圖片檔, API 格式中的圖片ID 是可以自行指定的. 所以也許可以利用 PandDoc 轉檔, 再寫一個小程序找出所有圖片檔的 ID, 並使用該 ID 把圖片上傳 (當然, 本文檔也可以一併完成).

找出圖片ID基本上pandoc就有內建了, 他抽取出圖片時就會自動按照順序命名圖片, 但問題是我需要前面的網址都不變只有後面名稱改變, 這樣我就可以很容易地從html代換程式碼更改, 但找很久都找不到這種圖床. 請問API名稱是? 或許這個可以試看看^.^

Darwin的AI天地 於 2017/01/16 13:19 回覆

#3 MagicJackTing · 於 2017/01/16 11:32

Word 2007 (含) 以後的版本改經把 WLW 的功能包含在內了.
以 Word 2010 為例:

3. 選擇 新增，部落格供應商 選擇 wordpress，下一步

4. 部落格文章 URL 修改為 "http://api.pixnet.cc/blog/xmlrpc"

5. 使用者名稱/密碼，自己在 pixnet 的帳號/密碼。

6. 圖片選項，選 "我的部落格提供者"。

7. 完成。

這樣就可以直接在 word 編修 部落格文章了 (舊貼文也可以拉回來改)

謝謝~~但這方法我測試過了，不包含公式可行，公式轉換會全部變空白

Darwin的AI天地 於 2017/01/16 13:17 回覆

#4  Renton · 於 2017/01/24 08:35

妳好,我是一個機器學習的初學者

在查找資料時看見妳寫的部落格,真的很驚訝南部有這樣的高手,也有些問題想要請教

我過去是3D的特效師,沒有任何數學背景,對數學的記憶大概只停留在國中的簡單代數

而當我開始學習機器學習最大的問題大概就在於如何看懂複雜的數學符號例如妳上面所列出的公式:

$$\delta L - 1k = f'(zL - 1k) * \sum_j (aLj - yj) * f'(zLj) * wLjk \delta kL - 1 = f'(zkL - 1) * \sum_j (ajL - yj) * f'(zjL) * wjkL$$

(6 -2-16)

我就完全看不懂這公式的含意

目前我挑的ML書籍多呈宣稱不用太多數學基礎的,但是我想一邊準備必要的數學知識以在未來進入深一步的領域時能夠準備好

若以看懂ML數學為前提的話,妳可以給我一些建議或是書單嗎?

非常謝謝!

您好喔，謝謝您的讚美，我也是初學者阿~跟國外那些大牛比起來 小編只是一介小民，那個解釋我在6.2.15的下文有提到喔，是敏感度，基本上只要會微積分基本概念就可以從無到有學會囉，我標題列的那本書就很棒囉 他有程式代 慢慢看一定能懂的

Darwin的AI天地 於 2017/09/22 13:56 回覆

#5  Renton · 於 2017/02/11 06:10

謝謝,下次發問我會記得 XD

我沒有數學基礎只好從代數開始複習

恩恩 有問題隨時問我喔 ^.^

其實呢只要搞懂代號的意義，代數其實很簡單的，不要被符號嚇到了我可以舉一個很簡單的例子說明代數

法學院的女生比男生多，在下學期的數學期末考試中，法學院不及格的學生超過了一半。由此可見

...

女學生不及格的比男生及格的多。

你看的出來嗎? 至少我是看不出來拉

但是你把它寫成代數後就可以推出來了

假設pb:及格男生 pg:及格女生 fb:不及格男生 fg:不及格女生

根據題目定義可以得到

$$pg + fg > pb + fb$$

$$fg + fb > pg + pb$$

因為a>b,c>d

所以a-d>b-c

套用上去

$$pg + fg - pg - pb > pb + fb - fg - fb$$

相同的削去後可以得到

$$fg - pb > pb - fg$$

所以 fg>pb

這題運用的技巧只是不等式數學技巧，而類神經推導的是運用微積分技巧，如此罷了 ^.^

Darwin的AI天地 於 2017/02/11 10:59 回覆

#6  大玩家闖天涯 · 於 2017/02/14 10:44

NN簡介重點整理很清楚，
加上中文翻譯讓更多台灣人有機會學習，
而不是中文網站關於深度學習大都是大陸博客，謝謝分享!
我也在做deep learning影像相關研究，
FCN和RNN、LSTM架構不是很模糊不清，
應該是說要親自實作比較會有感覺，
但前期在架設caffe時就遇到不少問題，
tensorflow也是未來會接觸的部分，
希望不吝賜教有問題會多多向您請教，多討論多交流!

p.s. 想不到你也有使用希平方學習平台，哈哈

恩恩 你的部落格也很棒啊XD
我對RNN也很有興趣
改天應該也會去探索研究
不敢當不敢當~~~有問題可以隨時討論 ^.^

Darwin的AI天地 於 2017/02/14 15:57 回覆

#7  鈞 於 2017/04/25 13:07

您好 我想請問一下關於sigmoid函數的微分
不是應該為 $y(x) = 1/(1+e^{(-x)})$
 $dy/dx = (1+e^{(-x)})^{(-2)} * e^{(-x)}$
這樣才對嗎?
為什麼在上面簡單版的變成
 $dy/dx = y*(1-y)@@?$
是我微分算錯嗎?

#8  鈞 於 2017/04/25 13:11

阿
將y疊代回去就是我算的沒錯XD
一時轉不過來
不好意思

恩恩 對對~~XD
沒關係~~有問題可以再討論喔~~

Darwin的AI天地 於 2017/04/26 00:12 回覆

#9  alanyuwenche 於 2017/08/06 11:26

感謝大大的分享.

圖6.2下方,yh1的計算值似乎有誤, 應為
 $yh1 = f(0.15*0.05+0.25*0.1+0.35) = f(0.3825)$
我是在檢查程式時發現這個問題.

喔喔 好喔 ~謝謝 ~~


Darwin的AI天地 於 2017/08/19 22:10 回覆

#10  playhigh(From CoCo) 於 2017/10/11 10:49

拜讀大大這篇筆記，簡要而清晰。有助讀者豁然領略奧妙。
在這裡 (<http://www.coco-in.net/forum.php?mod=viewthread&tid=53517&page=1#pid741583>) 有一文關於用BPN 做預測的實務上的疑問，
盼望大大瀏覽，敬請指教。

關於用類神經網路做預測的問題我想應該早就有前人做過了
類似的技術不只侷限於股票預測
且近年來深度學習技術蓬勃發展
用CNN LSTM做預測的也大有人在
從理論上來說都確實是可實現的

Darwin的AI天地 於 2017/10/11 19:38 回覆



!!!,

請問 i1 and i2 default 值 0.05 and 0.1, 是從頭到尾都用這組值去 train? 都不需要輸入新的值? 有些人說要越多資料樣本不是越準?

這個只是用簡單的例子解說喔


目的只是幫助理解神經網路的運算原理

真正在應用的時候是以這個為基礎再擴展的

是的，真正在算的時候i會有很多顆，取決於input的大小

Darwin的AI天地 於 2017/11/23 15:02 回覆

#12  無名小子 於 2017/11/24 15:18



淺顯易懂，獲益良多。很希望筆者能多寫一些這種細詳說明數學公式的AI相關課程。讓我們這種數學邊緣人能邊看公式邊看文字說明，了解AI的內部構造。萬分感謝您~~

不會~~很高興幫助到您XD

Darwin的AI天地 於 2017/11/25 19:41 回覆

推1

您尚未登入，將以訪客身份留言。亦可[登入](#)留言

您的暱稱 ...

留個言吧 ...

悄悄話

其他選項

送出留言

[回到首頁](#) [回到主文](#) [免費註冊](#) [客服中心](#) [痞客邦首頁](#) © 2003 - 2018 PIXNET

http://darren1231.pixnet.net/blog/post/338810666-%E9%A1%9E%E7%A5%9E%E7%B6%93%E7%B6%B2%E8%B7%AF%28backpropagation... 16/16