倒傳遞神經網路(Backpropagation Network, BPN)

# 寫在前頭

你可以從這篇文章得到：

1. BPN基本概念。
2. 如何實作BPN，並應用在遊戲裡。
3. 提供一個使用BPN決策的DEMO NPC。

# BPN簡介

BPN的網路架構為多層感知器(multilayer perceptron, MLP)，就是輸入層與輸出層之間多一層隱藏層(hidden layer)，使用誤差倒傳遞演算法(Error Back Propagation, EBP)為學習演算法，屬於多層前饋式網路，以監督式學習方式，處理輸入輸出之間非線性映射關係。

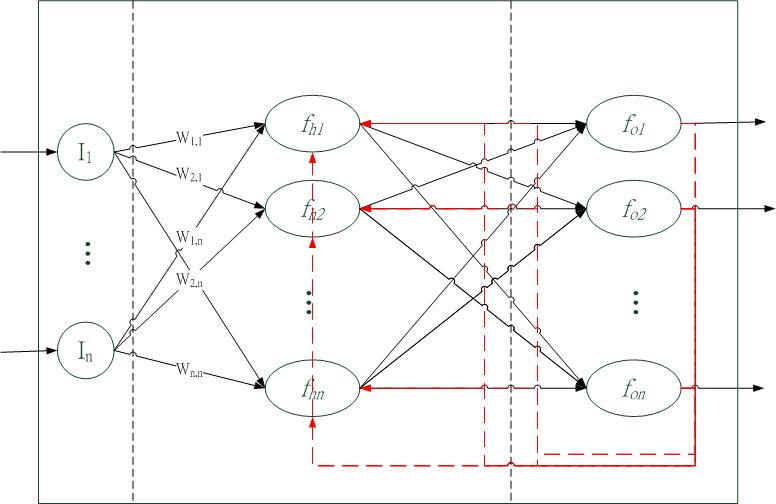


圖 1 BP網路圖

簡單說，BPN是由向前傳遞(forward pass)及向後傳遞(backward pass)兩個部份組成，向前傳遞(如圖1 由左至右黑線)是先將訓練資料丟進網路去跑，在計算出輸出結果與對應目標之間誤差，而向後傳遞(如圖1 由右至左紅線)是依誤差值去調整網路權重，經過這樣多次訓練後，就會將網路修正到誤差極小範圍內的輸出結果。其主要特性如下：

1. 學習精確度高
2. 回想速度快
3. 可處理非線性問題

提一下BPN會用到的數學公式。

* 向前傳遞

1. 使用sigmoid為激活函數：
2. 以均平方誤差(mean-squared error, MSE)為計算誤差值方法：

* 向後傳遞

1. 計算輸出層及隱藏層誤差值：
2. 調整隱藏到輸出層和隱藏層到輸入層的權重：

看了一大推數學算式，也不知道該怎麼實作。最後，可以參考**AI Application Programming, 2ed**這本書提供的倒傳遞演算法例子。

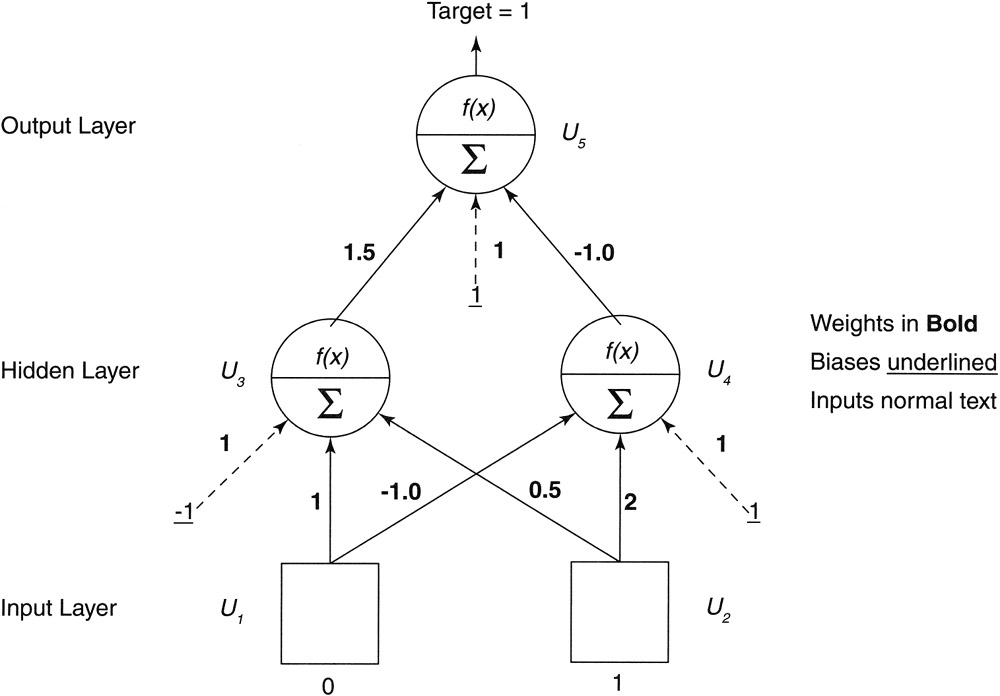


圖 2 BP實例

* The Feed-Forward Pass

計算各隱藏層的值和MSE。

* The Error Backward-Propagation Pass

計算出輸出層和隱藏層誤差值，在調整輸出層和隱藏各權重值及偏差值。(學習率為

經過調整後，再跑一次向前傳遞，重新計算MSE有明顯變低。

# BPN如何應用在NPC決策行為

首先，容許我說明一下遊戲設定，在遊戲中會有一位NPC和玩家對抗，基本上，NPC會有下列屬性：

1. 生命值：範圍會介於0~2之間，如果~生命值=0時，NPC就會消失不見。
2. 小刀：NPC必須要擁有小刀，才能與玩家進行近距離攻擊，其數值為0或1。
3. 手槍：除了要佩帶手槍外，NPC還要有足夠的子彈數，就可以射擊玩家，其數值為0或1。
4. 子彈數：範圍會介於0~2之間，每發射一次，就會耗損一枚子彈，若子彈數為0時，就不能發射，必須要填充子彈後，才可以繼續射擊。

當賦予NPC這些屬性後，會在遊戲中與玩家的互動過程中，會產生一連串決策行為，其下列是NPC設定之行為：

1. 射擊：當NPC有佩帶手槍且子彈數夠(大於0)，就會發動射擊玩家，若玩家被射中，就會損失一點生命值。
2. 小力攻擊：當沒有手槍可以攻擊，若NPC身上還有小刀時，NPC會選擇用小刀來攻擊玩家，直到NPC生命值很低，就會選擇逃跑模式。
3. 裝彈：當NPC身上佩帶手槍的子彈用盡時，就會到裝彈的地方補充彈藥，繼續攻擊玩家。
4. 逃跑：如果NPC無計可施，就會採最壞打算「逃跑模式」，直到自己被消滅為止。

將上述邏輯轉成M╳N的數值矩陣(如表1所示)，前四欄為輸入屬性，而後四欄是輸出目標，即為NPC對應的行為，當輸出目標的其中之一欄為1時，NPC就會去執行該動作。

表 1 訓練資料

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 生命值 | 小刀 | 手槍 | 子彈數 | 射擊 | 小刀攻擊 | 裝彈 | 逃跑 |
| 2.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| 2.0 | 0.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 2.0 | 0.0 | 1.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 2.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 2.0 | 1.0 | 0.0 | 2.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 2.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 2.0 | 1.0 | 1.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| 1.0 | 0.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 1.0 | 0.0 | 1.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 |

從這些NPC訓練資料中， BPN會去找出它們之間的Pattern，在利用這些訓練好的Pattern去完成決策。

# 略談遊戲程式碼

這個DEMO中所用的程式碼，如下列所示：

|  |  |
| --- | --- |
| 名稱 | 說明 |
| Backpropagation.cs | 為BPN運作核心，先將訓練資料讀入，並把網路權重訓練好，傳入參數至action方法，得到對應指令。 |
| BulletCollision.cs | 當子彈碰撞時，會產生爆炸效果，若子彈是和Player或NPC發生碰撞，並計算扣血值。 |
| Detection.cs | 當NPC跑到補充點，就會補充NPC手槍彈藥。 |
| FireBullet.cs | Player或NPC發射子彈時，會丟出剛建立的子彈物件。 |
| KnifeCollision.cs | NPC拿小刀攻擊時，如果發生碰撞，就會傷害Player。 |
| MainGUI.cs | 顯示Player的生命值。 |
| NPC.cs | 定義NPC所有屬性和對應行為。 |
| PlayerScript.cs | 定義Player所有屬性和對應行為。 |

# 總結

這一篇主要核心是利用BPN作分類器，由訓練資料集去劃分四個對應動作類別，在有限訓練資料筆數中，其分類準確度極高，但缺點是若輸入參數超出訓練資料集的數值範圍，就會造成誤判的問題，故必須要時時擴充訓練資料集，若能將所有發生的可能都放進去，就可避免此情形發生。

# 參考資料

1. M. Tim Jones, “AI Application Programming”, Second Edition, 2005
2. Brian Schwab, “AI Game Engine Programming”, Second Edition, 2009
3. [Neural Networks - An Introduction](http://www.tek271.com/documents/others/into-to-neural-networks)

# 備註

[Unity3D Package下載](http://goo.gl/wZaDI)