Recurrent Neural Network

By INFOR 28th李睦樂、洪啟勳、王冠人

Abstract

近年來人工類神經網路(Artificial Neural Network)因硬體的進步(如GPU)而成為Machine Learning中熱門的話題，本文將以介紹Machine Learning中的Logistic Regression、Neural Network以及Recurrent Neural Network為主，並且從中講述一些有關於此種演算法的相關技術以及數學概念。

1. Introduction

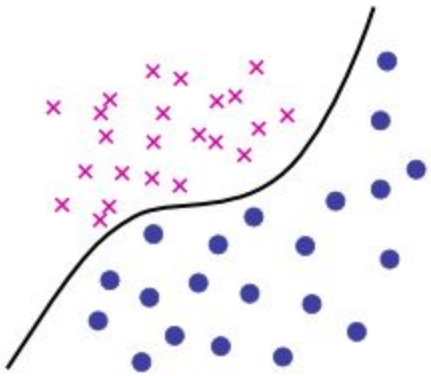
1.1. What is Machine Learning ?

Machine Learning，顧名思義，就是讓機器學習。針對特定的任務進行學習，進而在日後達成任務，這就是「學習」的意涵。Machine Learning 的過程可粗略分為兩部分：Training和 Applying 。

1.2. Basic Machine Learning Categories

i. Supervised Learning

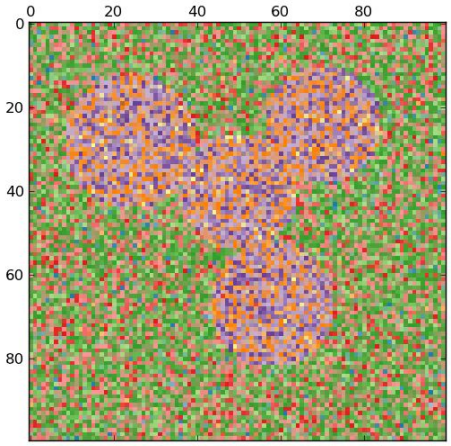
Supervised Learning 所處理的問題有"標準答案"，常見的有 Regression, Classification 等。舉 Classification 為例，當我們將電子郵件分為垃圾/非垃圾兩類時，每封郵件必為其中一類，而這一類即為這封郵件的標準答案。Supervised Learning 的一大特徵是訓練資料有 labeled(訓練資料包含"標準答案")，故我們可由此判斷是否為 Supervised。

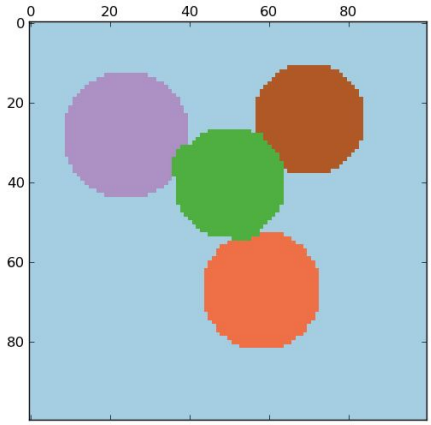


supervised learning----classification

ii. Unsupervised Learning

相對於 Supervised Learning， Unsupervised Learning 處理的問題沒有標準答案，意即結果可能有多種，常見的問題有 Clustering 等。舉例來說，公司在進行客戶分析時，可能希望將客戶分成幾群，但分群的結果並不會把客戶分為特定幾類，因為我們並沒有一開始就設定好類別，而是讓機器代替我們找出好的分法，也因此不會有標準答案。另外，相較於 Supervised Learning， Unsupervised Learning 的資料是沒有 labeled 的。





before clustering　　after clustering

iii. Reinforcement Learning

Reinforcement Learning 所要學習的是在哪些"狀態"下，要採取哪一種"動作"，而使"獎勵"能夠最高。例如讓機器學習如何下棋，棋盤上棋子分布情形即為"狀態"，要下哪個位置為"動作"，而輸贏程度(或下棋時優劣程度)則為"獎勵"。此例中，Reinforcement Learning 會試圖找出最好的贏法。像這種過程中有利害關係的任務，就常常會使用到 Reinforcement Learning 。

1.3. How to do Machine Learning ?

Supervised Learning 機器學習個過程中，最簡單的比喻就是希望找到一個函數，可以將資料丟進去，而可以預測結果，也就是說，最主要的目標就是找出這個函數的係數。

在尋找係數的過程中，要將預測出來的結果與正確答案之間的誤差降到最低，因此對於每一種算法，都會有一個Cost function，此function表示了誤差值，當此誤差直降到最低時，也就完成學習。

2. Previous Work

2.1. Linear Regression

線性回歸是研究單一依變項與一個或多個自變項之間的關係，主要用途在於預測(prediction)以及因果分析(causal analysis)。例如現在有許多數據坐落在座標圖上，利用最小平方法(least squares)在座標平面上找出一條線，使其跟數據集的吻合度最高。從這條線可以預測數據的趨勢，像是房價的預測等等。

2.2. Logistic Regression

i. 簡介

邏輯回歸使用於數據為離散的值時，例如「將電子郵件分成垃圾郵件或是非垃圾郵件」這種分類問題，不過就實際

上來說，當然不只能做到二元分類，例如要判斷這個人是哪個國家的人，就有很多類別。

ii. 實作

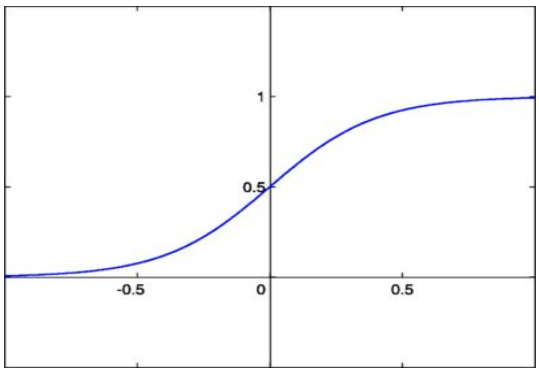
因為是分類，希望輸出的值介於0到1之間，以代表屬於某類的機率為何。所以使用sigmoid function當作我們的hypothesis。而我們的找出Cost function，然後再minimize，即可找出最佳解。

2.2.ii.1. sigmoid function

logistic regression主要目的是分類，因此每次運算都是要分成兩邊，但若直接以0.5為基準直接分為True or False，並不利於Cost function的使用，因此採用一種兼具線性的sigmoid function

sigmoid function: $g(z) = \frac{1}{1+e^{-z}}$

其函數圖形為



2.2.ii.2. Cost function

因為如果對sigmoid function使用最小平方法，會造成其Cost function是non-convex，所以可能沒辦法找出全局最佳解。故我們使用以此式：

$J(\theta) = -\frac{1}{m}[\Sigma(i=1,\ m)\ y^{(i)} log\ h_\theta(x^{(i)}) + (1-y^{(i)}) log\ (1-h_\theta(x^{(i)}))]$

作為我們的Cost function，使其為convex function。

2.2.ii.3. minimize

找出Cost function後，我們需要找出在該函數圖上最低的點，使誤差最低，最簡單的方法是使用梯度下降法(Gradient Descent)更新找出最佳解，更新表達式如下：

$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta j}J(\theta) \quad (for\ j = 0,\ 1,\ ...,\ n)$

Gradient Descent的觀念是，在欲處理函數圖型上選一個起始點，然後求導，若還該點(θ_j)不是最低點，那就更新其數值(注意，必須同時更新所有θ)，也就是該點(θ_j)減掉該點偏導數($\frac{\partial}{\partial\theta j}J(\theta)$)乘上更新速率(learning rate:$\alpha$)；直到收斂(convergence)，那麼我們就可以順利找到全局最低點。

除了Gradient Descent，還有許多不同的minimize方法，這些方法會自動更新learning rate，使得

convergence變快，但相對也比較複雜。

- Gradient Descent
- Conjugate Descent
- BFGS
- L-BFGS

2.3. NN(Neural Network)

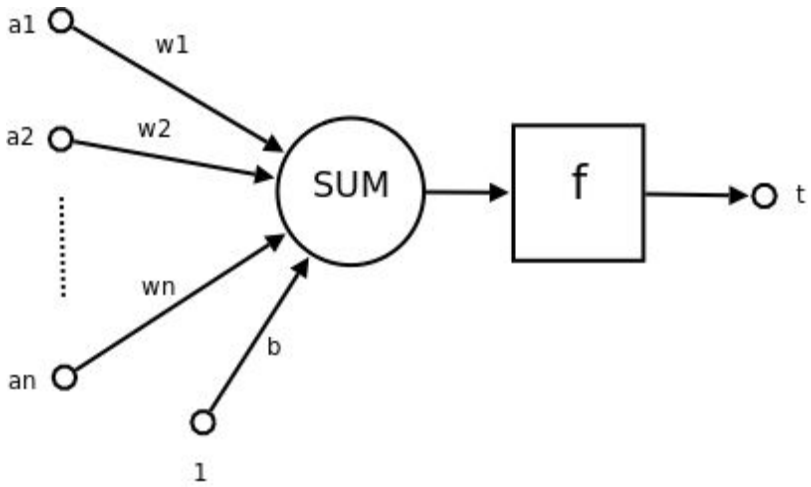
i. What is NN?

類神經網路的靈感是來自於人類大腦的神經網路，大腦有很多神經元，而NN的每個神經元(perceptron)也是參考大腦神經元：有多輸入、單輸出的構造。這項技術廣泛被使用在例如語音辨識系統、手寫辨識系統等等。

ii. 神經元

每個神經元都又輸入以及輸出，每個輸入相當於神經上的突觸，上面都帶有該筆資料的權值(weight)，將會與輸入相乘之後加總到神經元中。

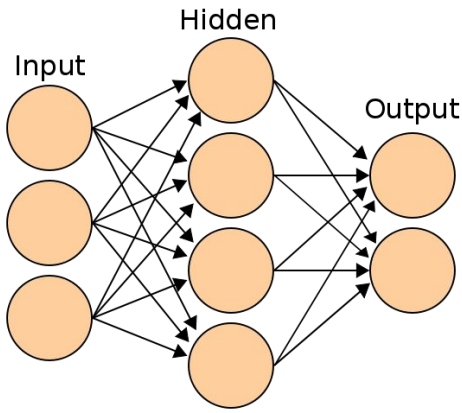
a_1 ～ a_n 為輸入
w_1 ～ w_n 為神經元各個突觸的權值(weight)
b 為常數(bias)
f 為傳遞函數，通常有tanh, sigmoid, etc.
t 為輸出



若將 a_1 ～ a_n 轉為一矩陣 $a = [1, a_1, a_2, ..., a_n]$
若將 w_1 ～ w_2 轉為一矩陣 $w = [1, w_1, w_2, ..., w_n]$
令 $\varphi = f$
則可表示為 $t = aw^T\varphi$

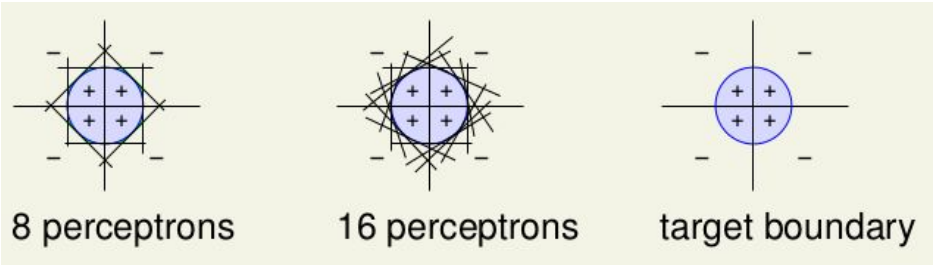
iii. 架構

以神經元組成下面的架構：



一般多層結構的前饋網絡分成三個部份：
輸入層(Input layer)：接受大量的輸入訊息，數目以輸入為準。
隱藏層(Hidden layer)：數目不定，愈多愈可以表現不同分類特徵。
輸出層(Output layer)：分析運算資料，並且輸出。

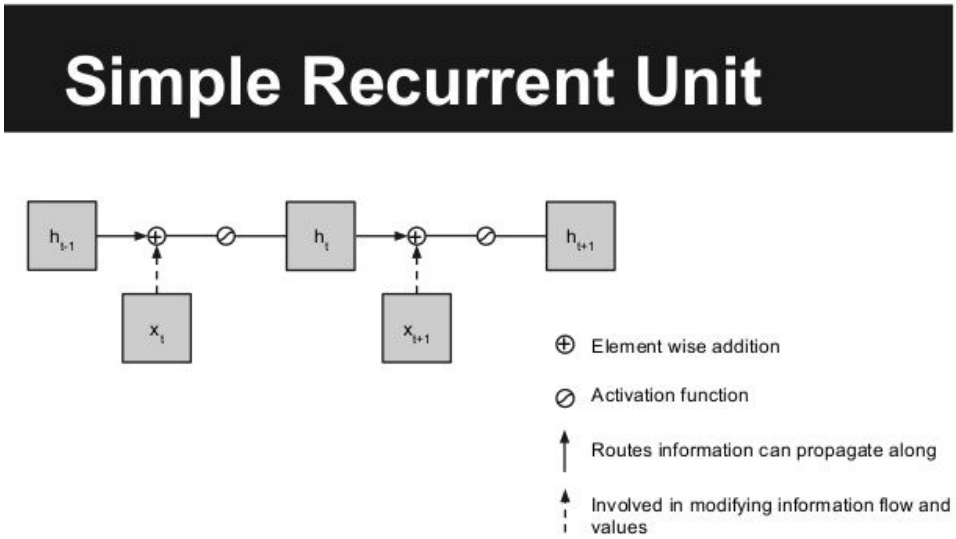
NN最重要的就是隱藏層，隱藏層中的perceptron愈多，愈可以完整的分類。



3. Model
   3.1. Recurrent Neural Network
      i. From NN to RNN
      NN可以分析比較複雜的數據，但是在使用NN的時，會遇到一種問題，比如說：
      現在想要使用NN來辨別語句，分析語句"I am a boy." 以及"I am not a boy." 時，依照NN的運作原理，會將此兩者認定為很相近的句子，但事實上，這兩者是不同的。
      對於這種feature有順序性質的問題，解決的方法其中之一就是讓前次output去影響這次的output (Sequential Learning)，也就是說，計算當次結果時，也將上次的結果加進這次的運算中。

      ii. How a RNN works?
      遞迴神經網路(Recurrent Neural Network)，是以NN為基本架構，但在隱藏層中，其節點運算過後一方面將output輸出，一方便也將output保留下來，傳回到下次的節點中計算。
      如此一來，每一個節點都會接收到該次的輸入，以及上次的輸出，如此即可分析具有順序性質的資料。

iii. A simple recurrent neural network



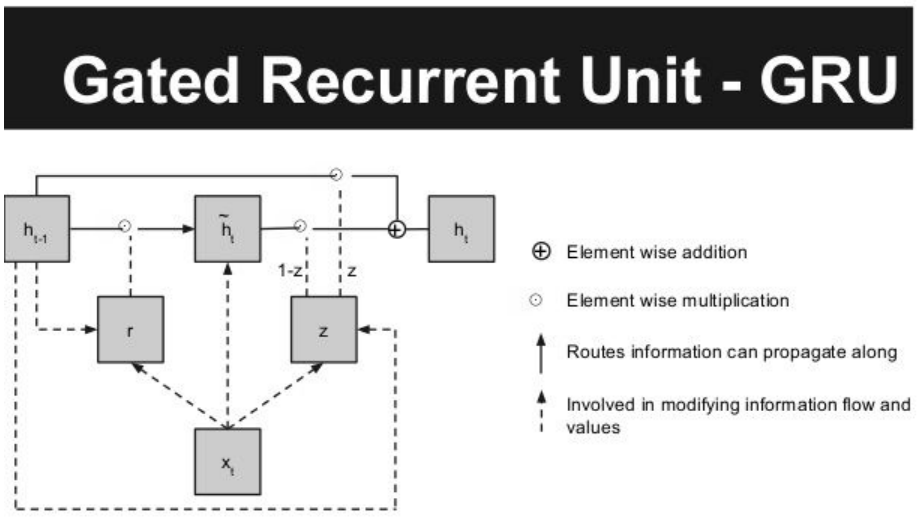
接下來先以一個簡單的方式建構出RNN：

定義：
x ：輸入層(Input layer)
h ：隱藏層(Hidden layer)
y ：輸出層(Output layer)

$$h_t = \theta\varphi(h_{t-1}) + \theta x_t$$
$$y_t = \theta_y\varphi(h_t)$$

x_t 第t次的輸入乘上 θ 權重(weight)之後，加上上次hidden layer的output(h_{t-1})乘以 φ (某函數)，以及權重 θ，就成為這次hidden layer的output。
而各次輸出層再將各hidden layer 的output乘以 φ (某函數)，及權重 θ，成為最終的輸出。

3.2. Gated Recurrent Unit
Gated Recurrent Unit(GRU)，是一個RNN的Model，總共有兩個gate，一個用來決定上一次output的有多少要影響這次cell，另一個決定上一次結果的和這一次的cell分別要佔這次的output的多少。



gates:

- r(reset gate)：分析x_t與h_{t-1}並丟進sigmoid function，這個gate是用來決定多少上一次的結果要送入，此gate的值將會介於0和1之間。
- z(dynamics)：再一次分析x_t與h_{t-1}，這個gate是用來決定這次算出來的值與上次算出來的要佔輸出多少，而此gate的值也會介於0和1之間。

$\overline{h_t}$(cell)：由h_{t-1}乘以r 與x_t取得。

$$h_t = \overline{h_t}(1-z) + h_{t-1}z \; \overline{h_t}(1-z) + h_{t-1}z$$

3.3. momentum

如果想要加快速度，可以改變原本梯度下降時更新weight的方法，讓training rate α可以式以一個線性的方式逐次改變，也就是說，一開始的 α很大，之後 α就愈來愈小於是將原本的更新方法改成：

$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t)$$
$$W_{t+1} = W_t + V_{t+1}$$

並先決定一個μ(介於0到1)

4. Discussion

4.1. sigmoid function

在用來分類的logistic regression中，通常人們很直觀的會認為，把結果丟進Step function即可，雖然這樣很有道理，但是計算Cost function時，會無法產生下降的趨勢，因此使用sigmoid function，即可解決這個問題又保留原本分類的問題。

4.2. Regularization

若是過度的train，將會造成overfitting，當所建構出的模型的參數個數相對的大於數據的個數的時候，就會發生overfitting，也就是說weight過度的接近train data，這會造成真正預測的時候反而會有落差，這時候就要使用一些Regularization的方法，也就是對於過大的weight懲罰。

以下是幾種Regularization的方法，在此就不一一解釋。

L2-Regularization

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \mathbf{w}))^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

L1-Regularization

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \mathbf{w}))^2 + \lambda|\mathbf{w}|_1$$

L0-Regularization

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \mathbf{w}))^2 + \lambda\sum_{n=0}^{N-1}\delta(w_n \neq 0)$$

5. Reference

5.1. Machine Learning: 2014-2015 Course materials from: https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning

5.2. Alec Radford, 2015, General Sequence Learning using Recurrent Neural Networks from:https://www.youtube.com/watch?v=VINCQghQRuM

5.3. Andrew Ng, Machine Learning, Stanford, Coursera https://www.coursera.org/course/ml

5.4. Machine learning, Wikipedia http://en.wikipedia.org/wiki/Machine_learning

5.5. Solver, Caffe by the BVLC http://caffe.berkeleyvision.org/tutorial/solver.html

5.6. Volodymyr Mnih; Nicolas Heess; Alex Graves; Koray Kavukcuoglu. (2014) Recurrent Models of Visual Attention, Google DeepMind. http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf