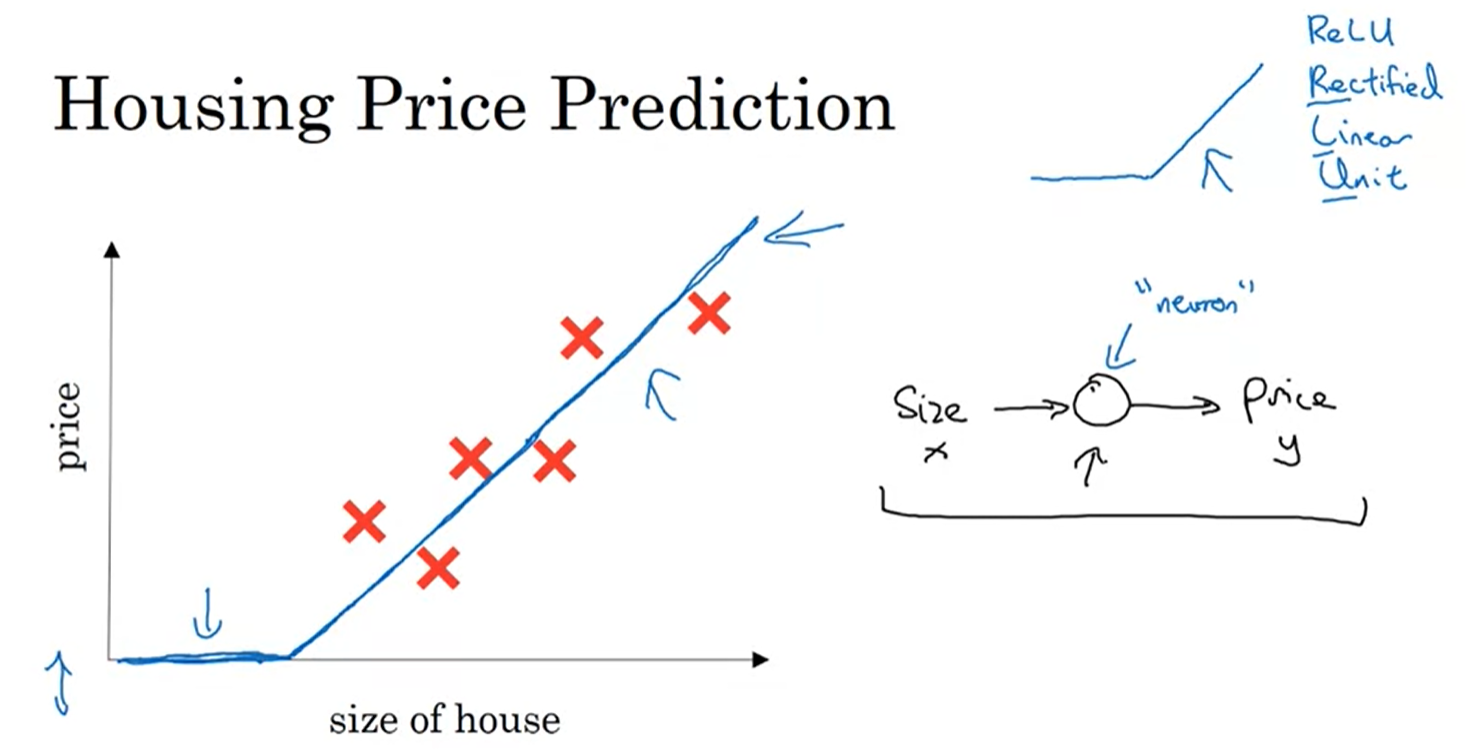
# 神經網路和深度學習

先了解線性代數的計算

匯總[CS\_Neural .docx](file:///D:\doc_office_IT_training_AI\course\TA\CS_Neural%20.docx)

## 深度學習介紹

假設您有一個資料集有六個房子, 您知道這些房子的大小是多少平方呎或者多少平方米 您也知道房價，您想要用一個函數來預測房價



有一個輸入到這個神經網路 房子大小我們稱為x --> 它進入這個節點 --> 然後輸出價格我們稱為y

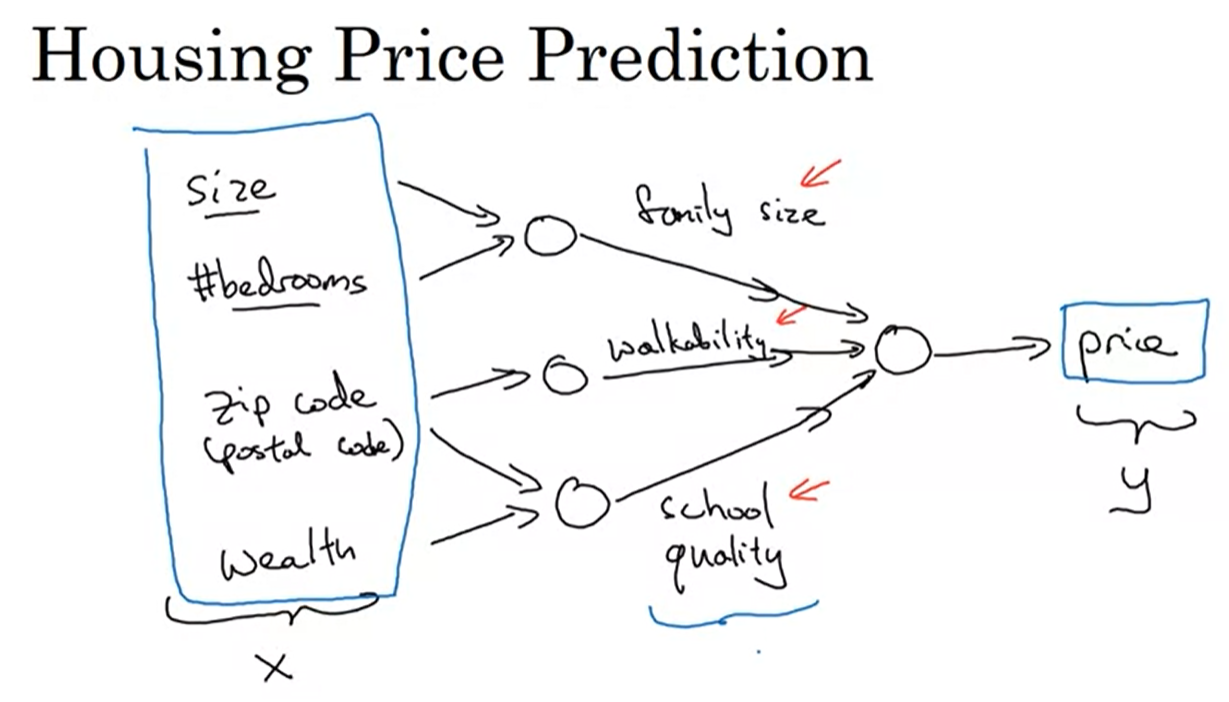
這個小圓圈, 是一個單一神經元在神經網路中建立這個函數 我們畫在左邊

這個神經元做的是輸入大小, 計算這個線性函數取最大值0, 然後輸出預測價格

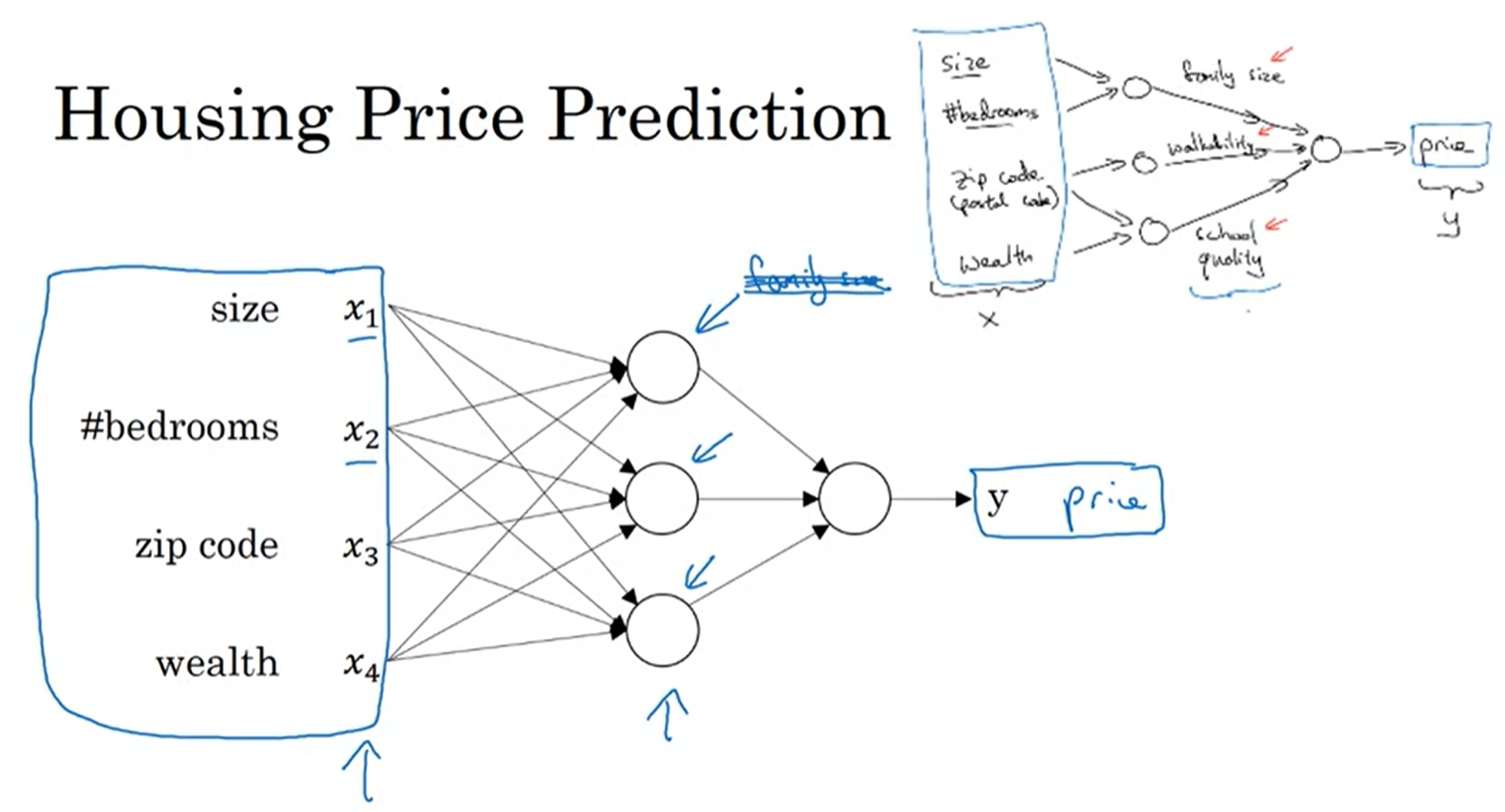
一個大型的神經網路是由很多的單一神經元堆疊在一起

假設有房子其他的特徵，當您建置它時 您需要給一個輸入x 輸出y 一些您訓練集的例子 所有中間的這些它們會自己搞定

這些圓圈稱為隱藏單元



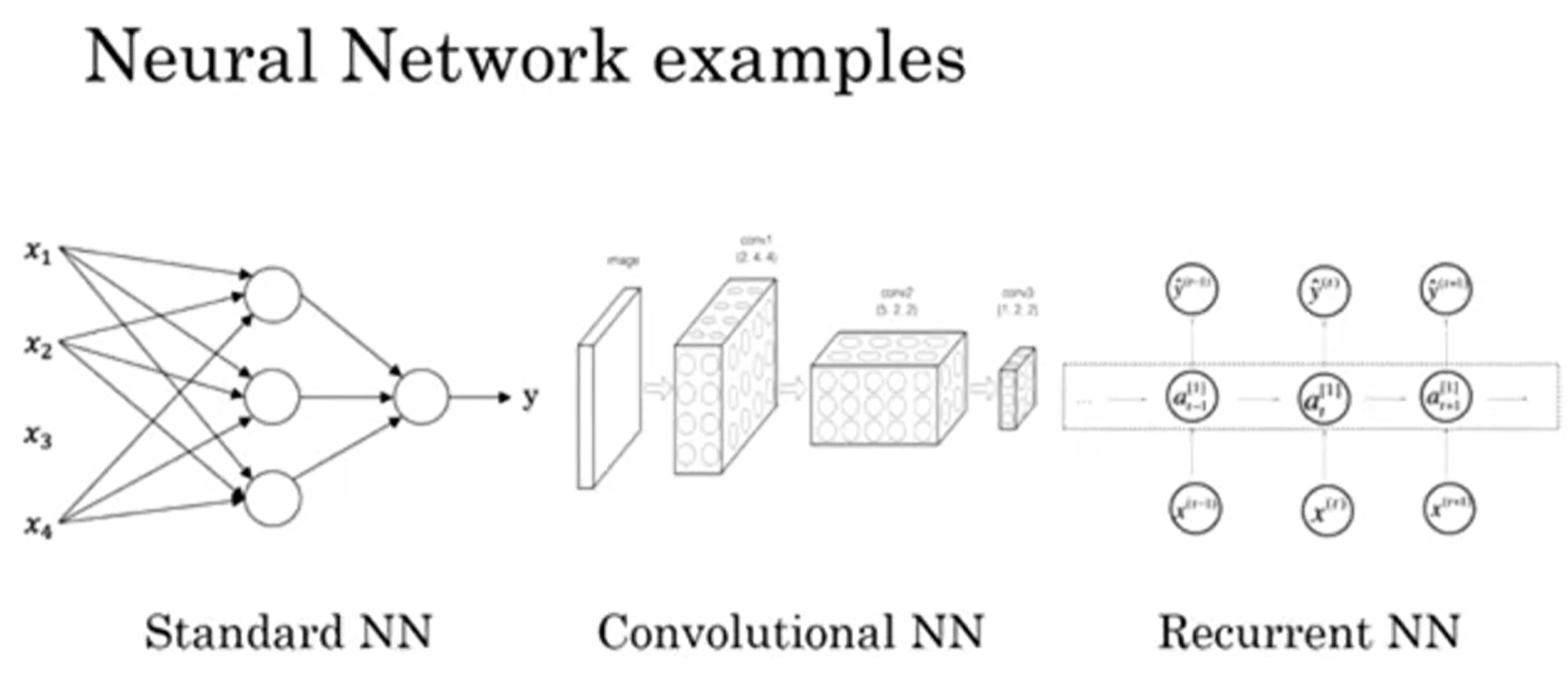
在神經網路, 您決定這些節點是什麼 我們會給您所有四個輸入特徵來計算您想要的 所以我們說這些層是這個輸入層跟這些中間層在神經網路中是緊密連接 因為每一個輸入特徵連接了每一個在中間層的圓圈 而值得注意的是神經網路只要給予足夠的資料x跟y, 給予足夠的訓練例子x跟y, 神經網路會非常好的找出函數來對應從x到y



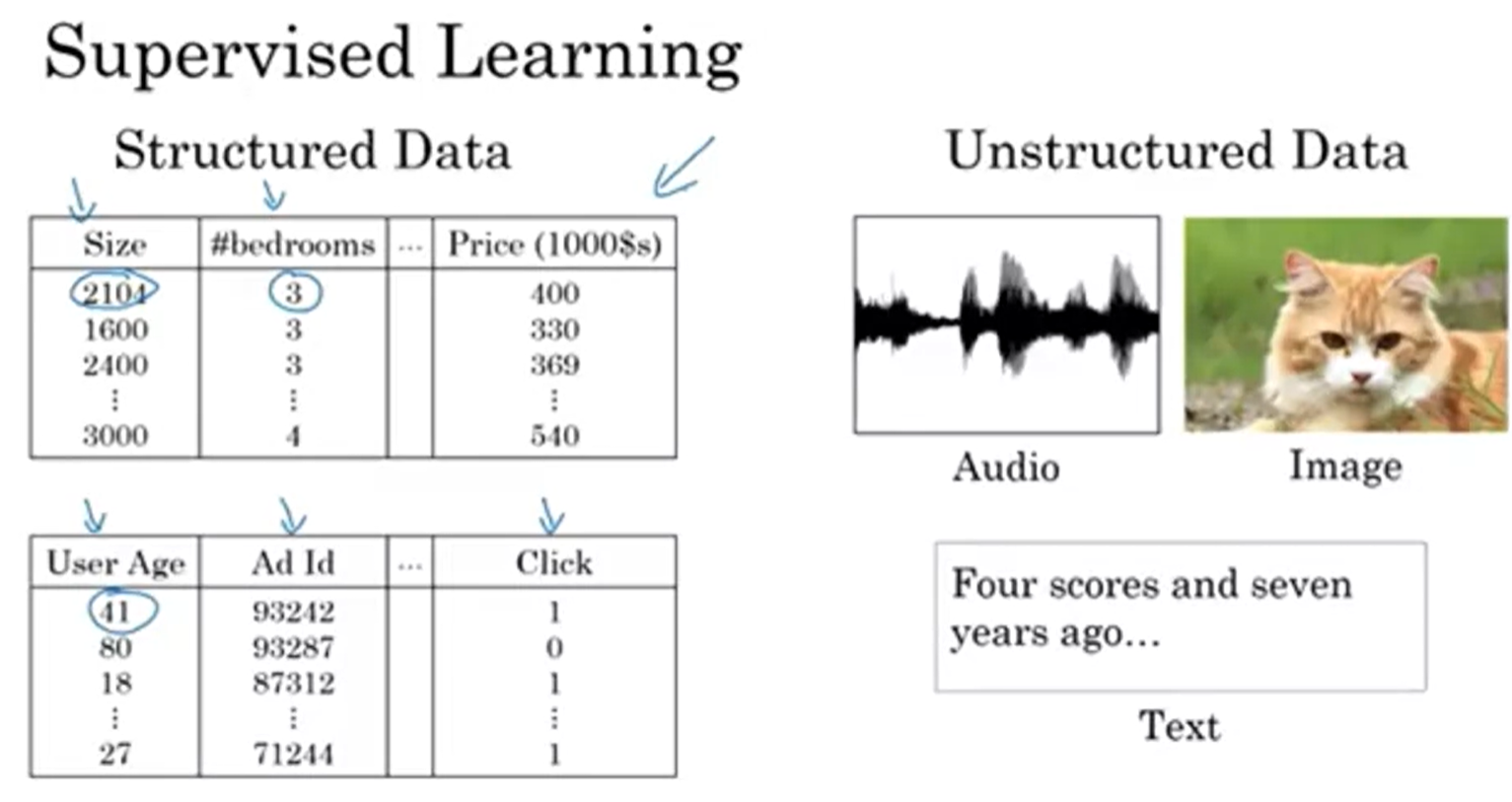
### 監督式學習 在神經網路

電腦視覺過去幾年也取得重大進展 主要來自深度學習

* 對於影像應用我們通常使用卷積神經網路(CNN)
* 對於序列資料(聲音) 通常使用循環神經網路(RNN)

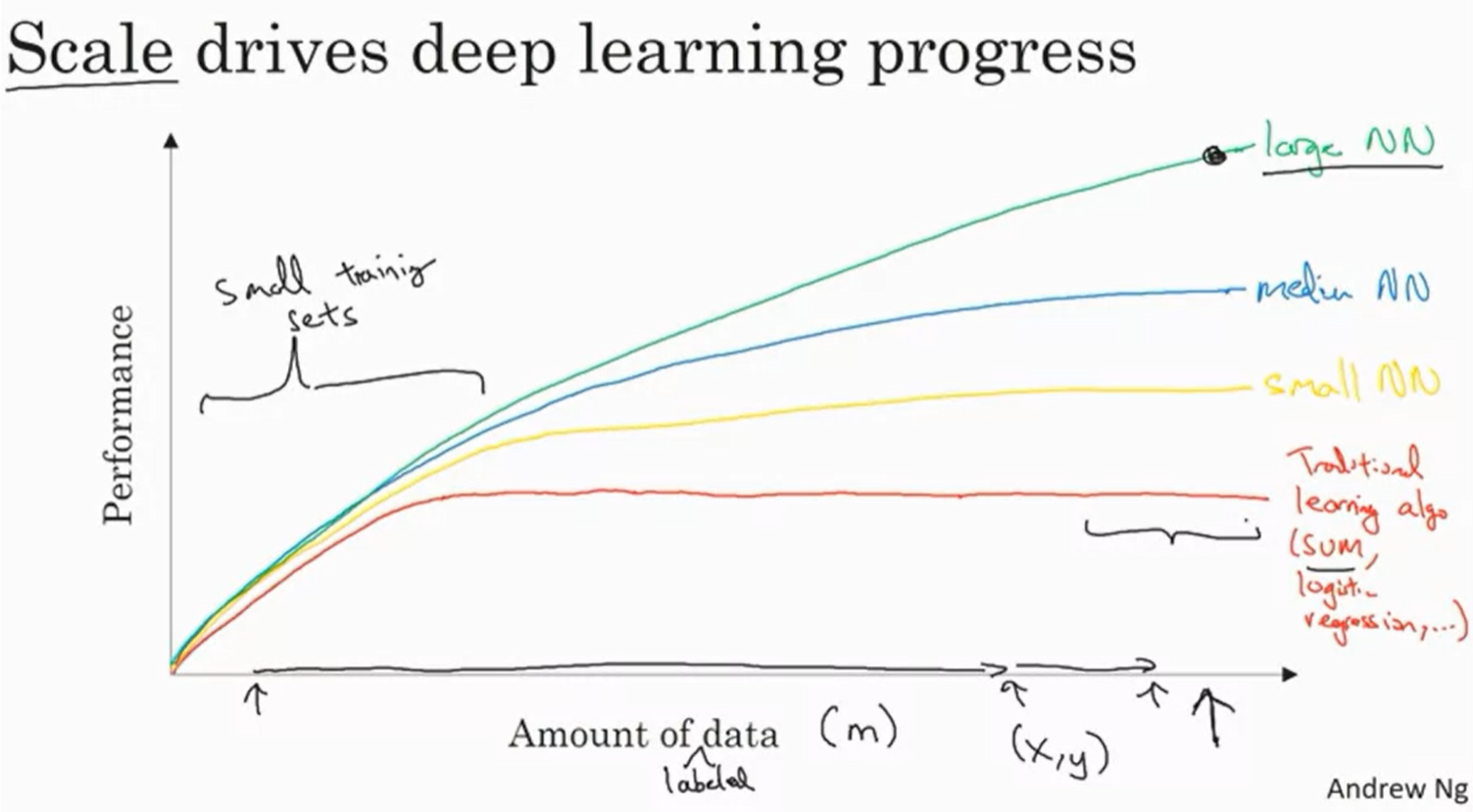


### 結構資料跟非結構資料



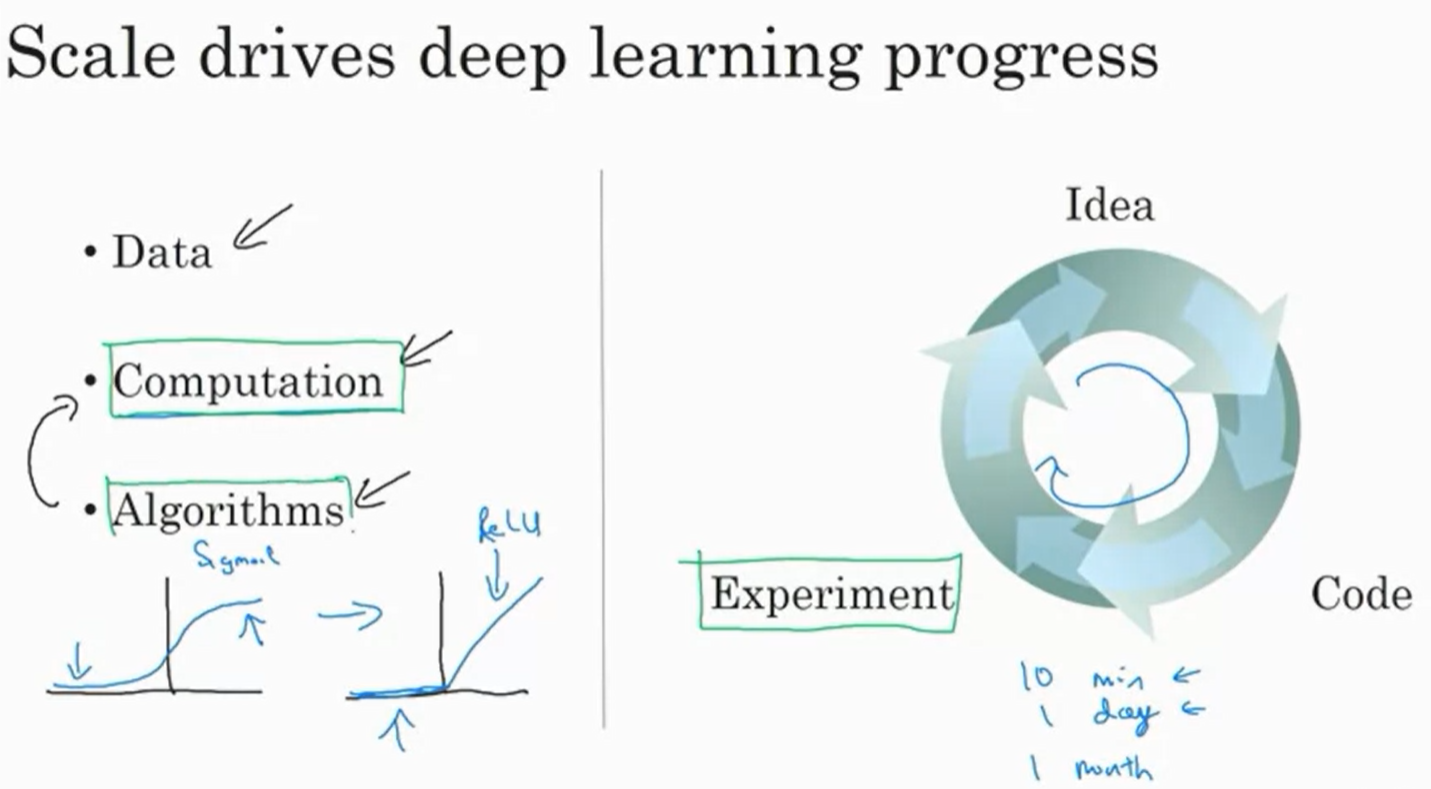
歷史上, 它是比較困難的對於電腦去理解非結構資料相對於結構資料 實際上人類已經進化到非常能了解聲音及影像 而文字是比較現代的發明 但人們非常能解釋非結構資料 所以有關神經網路的崛起 其中最令人興奮的事之一是因為深度學習, 因為神經網路,電腦現在比較能解釋非結構資料

### 為什麼神經網路興起



1. 推動深度學習進步的是規模。一是指神經網路的大小， 意思是神經網路有很多的隱藏單元，很多參數， 很多的連結, 二是資料的規模
2. 一個重大的突破是改變演算法, 因為它允許我們的程式運行的 更快, 這讓我們訓練更大的神經網路或推論,在 合理的時間內

在神經網路是從 從 S型函數(Sigmoid), 看起來是這樣的, 換到 一個線性整流(ReLU)數學函數,



* 啟動函數(Act

(Ref: CS1\_w3、恆逸demo17,18)

Sigmoid ：S型函數

ReLU ：線性整流函數

## 羅吉斯迴歸Logistic Regression as a Neural Network

### 二元分類Binary Classification

羅吉斯迴歸分析是一種演算法使用在二元分類

一個例子有關於二元分類問題 您也許有一個輸入的影像 想要輸出一個標籤來識別這個影像是否是一隻貓 是貓您輸出是1,或者不是貓輸出是0 而您將使用y來記這個輸出標籤



展開這些所有的像素值變成一個輸入特徵向量x

使用nx=12288來表示輸入特徵x的維度

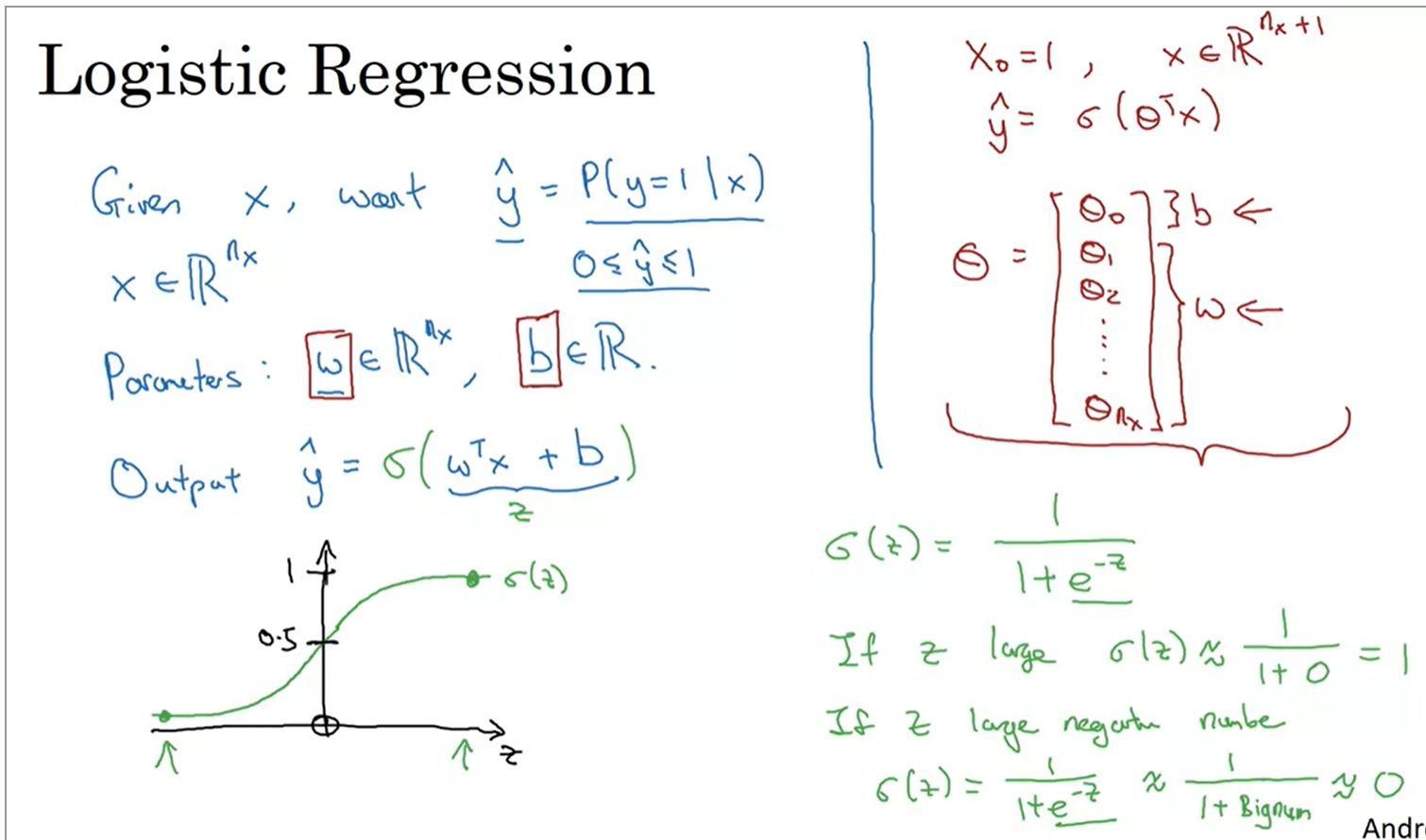
### Logistic Regression

(ŷ)y-hat 就是您的預估y

y-hat是一個機率當輸入x, y等於1的機會

(wTx)+b 類似是輸入x的線性函數

w轉置x加b可能大於1, 或負數 對於概率而言沒意義. 想要它介於0跟1之間. 因此,在羅吉斯迴歸分析中, 輸出將是y-hat 等於S型(sigmoid)函數應用到這個量



ŷ=σ(wTx+b)=σ(z)= ; 0<ŷ<1

### Cost Function

損失函數或者錯誤函數 我們可以用來衡量我們的演算法有多好一種方式是您可以定義損失為當您的演算法輸出y-hat而真正的標籤是y



上標括弧 i 代表資料

定義不一樣的損失函數變得比較容易做優化

損失函數：L(ŷ, y) = -(y log(ŷ) + (1-y)log(1-ŷ))

用這個羅吉斯迴歸分析損失函數我們也會想這要它盡可能小

* 假設y=1 那損失函數(y-hat, y), 就只是第一項也就是負的log y-hat 如果y= 1, 因為如果y=1 那第二項 1-y 會等於0 所以這是說如果 y=1 您希望 負的log y-hat 越小越好 也就是說log y-hat越大越好 越大表示您希望y-hat越大 但因為您知道在S型函數中,它不會大於1 所以,這是說,如果y等於一,你希望y-hat越大越好 但它不會大於1 所以這說明了您希望y-hat盡量接近1

if y=1: L(ŷ, y) = -(1\* log(ŷ) + ~~(1-1)log(1-ŷ))~~ =- log(ŷ) 越小🡺- ŷ越小🡺ŷ越大🡺 ŷ≒1

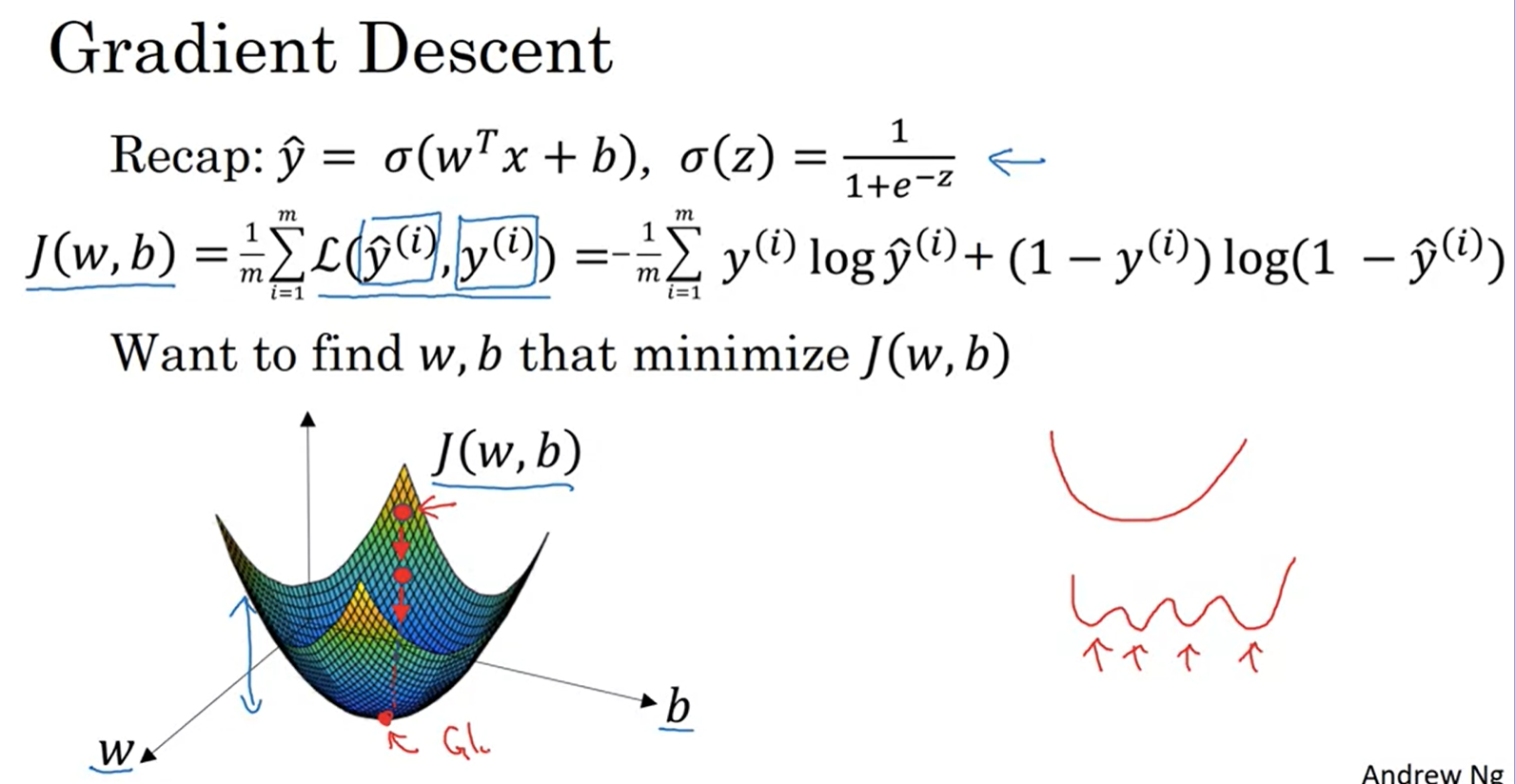
* 如果y=0那損失函數第一項會等於0 因為y是0, 然後第二項定義了損失函數 所以損失變成負的log(1-y-hat) 所以如果在學習過程中您試著讓損失函數最小 也就是您希望log(1- y-hat)最大 因為這裡有一個負號而經過類似的原因您可以得到結論 說這個損失函數試著讓y-hat越小越好 再一次因為y-hat是介於0跟1 這說明了如果y等於0 那您的損失函數會推向那些使得y-hat接近於0的參數

if y=0: L(ŷ, y) = -(~~0\* log(ŷ)~~ + (1-0)log(1-ŷ)) =- log(1-ŷ) 🡺- (1-ŷ)越小🡺ŷ越小🡺 ŷ≒0

成本函數衡量您在整個訓練集作用得好不好 所以**成本函數J**是適用於您的參數 W和B將是平均值與1/m 1/m的損失函數的和應用到每一個訓練例子

羅吉斯迴歸分析可以視為非常非常小的神經網路

### Gradient Descent (梯度下降法)



成本函數J(w,b)是一個曲面在水平軸w跟b上方 這個曲面的高代表J(w,b)的值在一定點 我們要做的是實際上去找到w跟b的值相對於這個成本函數的最小值

梯度下降做法是 它開始於起始點 然後往最陡的下坡方向邁出一步，直到最終收斂到這個全域最佳化或者接近於全域最佳化

