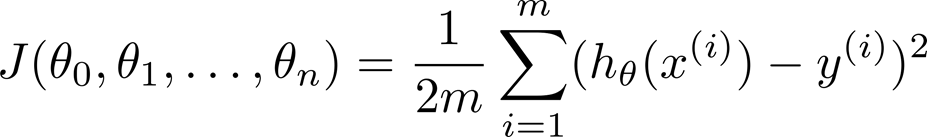
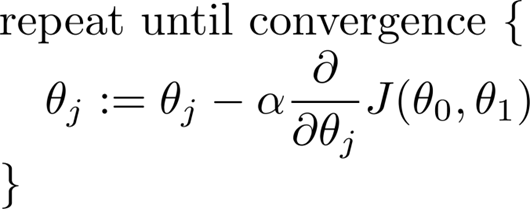
**Linear Regression**的應用在於判別預測有規律的線性關係，像是房價跟土地大小，房間關係，股票等等

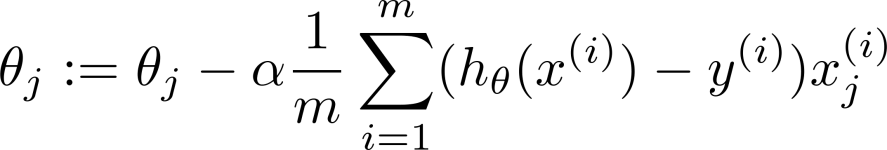
Linear Regression Gradient Descent 的cost function=



Gradient Descent j(theta)就是cost function需對其偏微分

即等於



其中

所有的Gradient Descent可以用

**Feature Scaling** : Make sure features are on a similar scale.

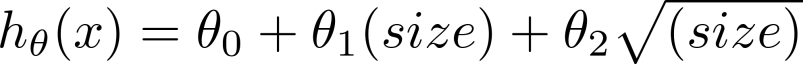
如 X是在0~2000中🡪 X除與2000使其0~1

Y是在0~5中🡪 X除與5使其0~1

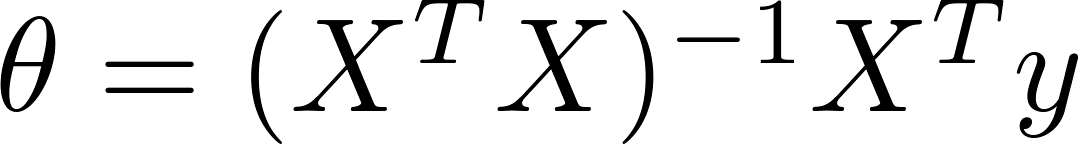
盡量使參數落在0~1中，像是-3~3 或是 -1/3~1/3 都很可以

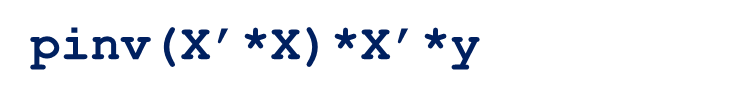
Declare convergence if J(Theta) decreases by less than 0.001 in one iteration.

choose 學習效率lanta建議是以3倍為垮度，從0.001~0.03~0.1~0.3~1~3

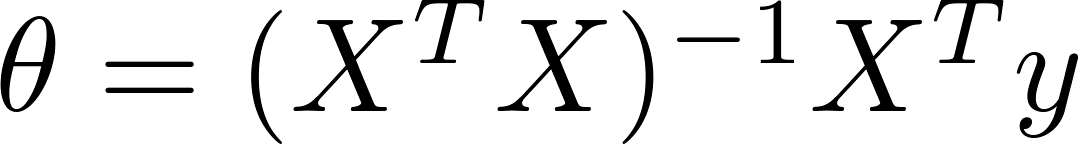
對於預測線性的regression，如果使用二次方程式當假設，當x軸數字過大時會下滑對於一些情況不符，故建議假設改成如下即可讓X軸數字很大仍是預測上升

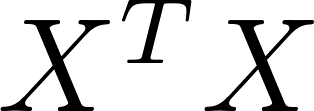
**Normal equation (公式解**



OCTAVE

如果參數數量是在10000以內用normal equation比較快，而大於10000數量建議使用梯度下降更快更穩定

Normal equation 不用選擇學習率，不用跌代次數但須計算，其中X是所有參數的矩陣， Y是數字結果。

有可能non-invertible?無法有逆解 因為singular/ degenerate

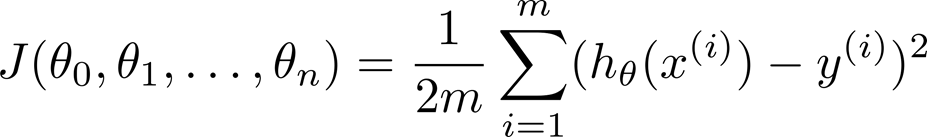
解決方法 1.把相關參數併成一項使其單位相同 2.刪除一些多於且不重要的參數或是用regularization

Gradient Descent 需要選擇學習率，需要跌代次數，當參數很多時使用梯度下降更快更穩定

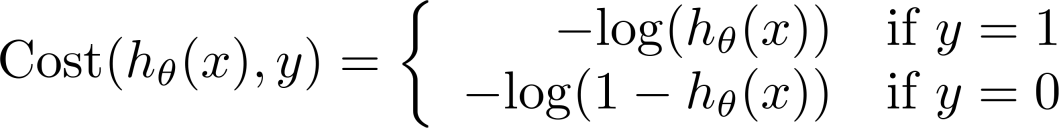
**Logistic Regression**的應用在於判別多種聚類分布，像是分類很多種不同事物

Logistic Regression的Gradient Descent的

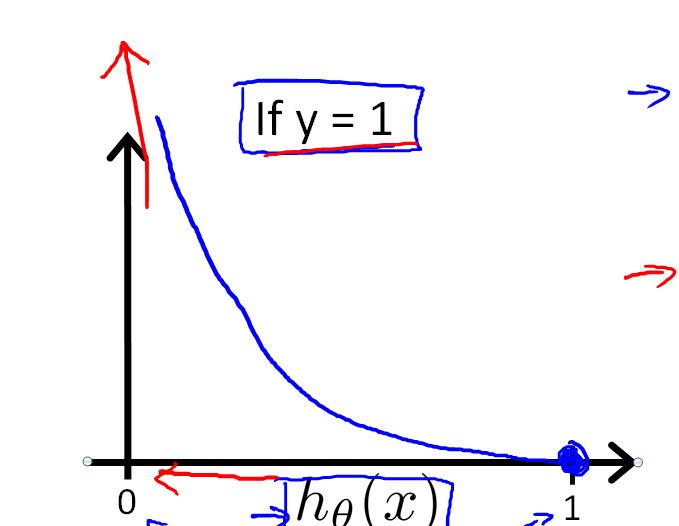
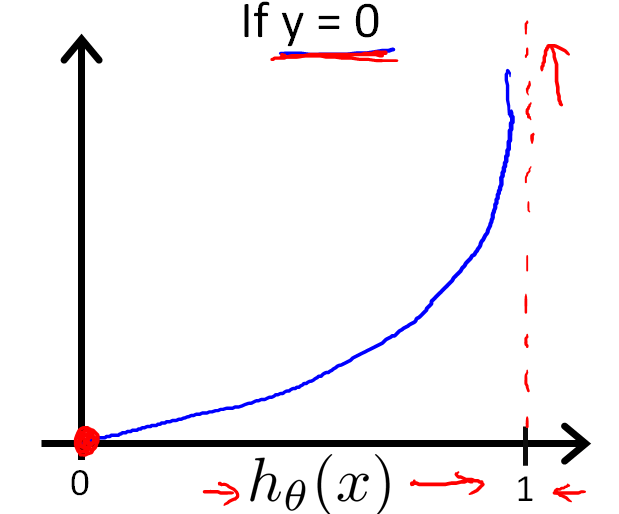
cost function =



但是其會造成線是non-convex 會是凹凸不平的曲線，所以需要改成

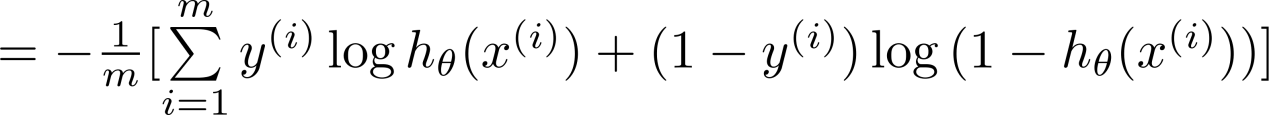


其中

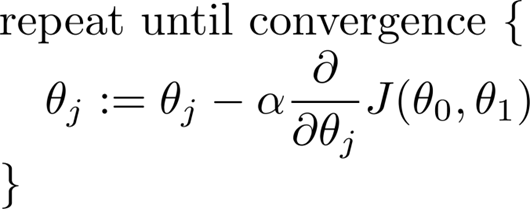
注意: Y永遠只會是0或是1

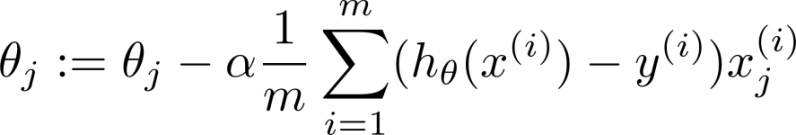
Cost Function合併成



去求 

Gradient Descent j(theta)就是cost function 需對其偏微分

即等於 

**Logistic Regression 的 Advanced optimization (進階**

**Optimization algorithm** 算法有像是 1.Conjugate gradient 2.BFGSL 3.BFGS

好處 1.不用人為去更換學習率，算法會自己去找最佳的學習率

2.比一般梯度下降更快收斂

壞處 1.算法比較複雜

建議機械學習都將數據用 向量化才能減少code以及用別人的開源庫

**function [jVal, gradient]   
 = costFunction(theta)**

**jVal = (theta(1)-5)^2 + ... (theta(2)-5)^2;**

**gradient = zeros(2,1);**

**gradient(1) = 2\*(theta(1)-5);  
gradient(2) = 2\*(theta(2)-5);**

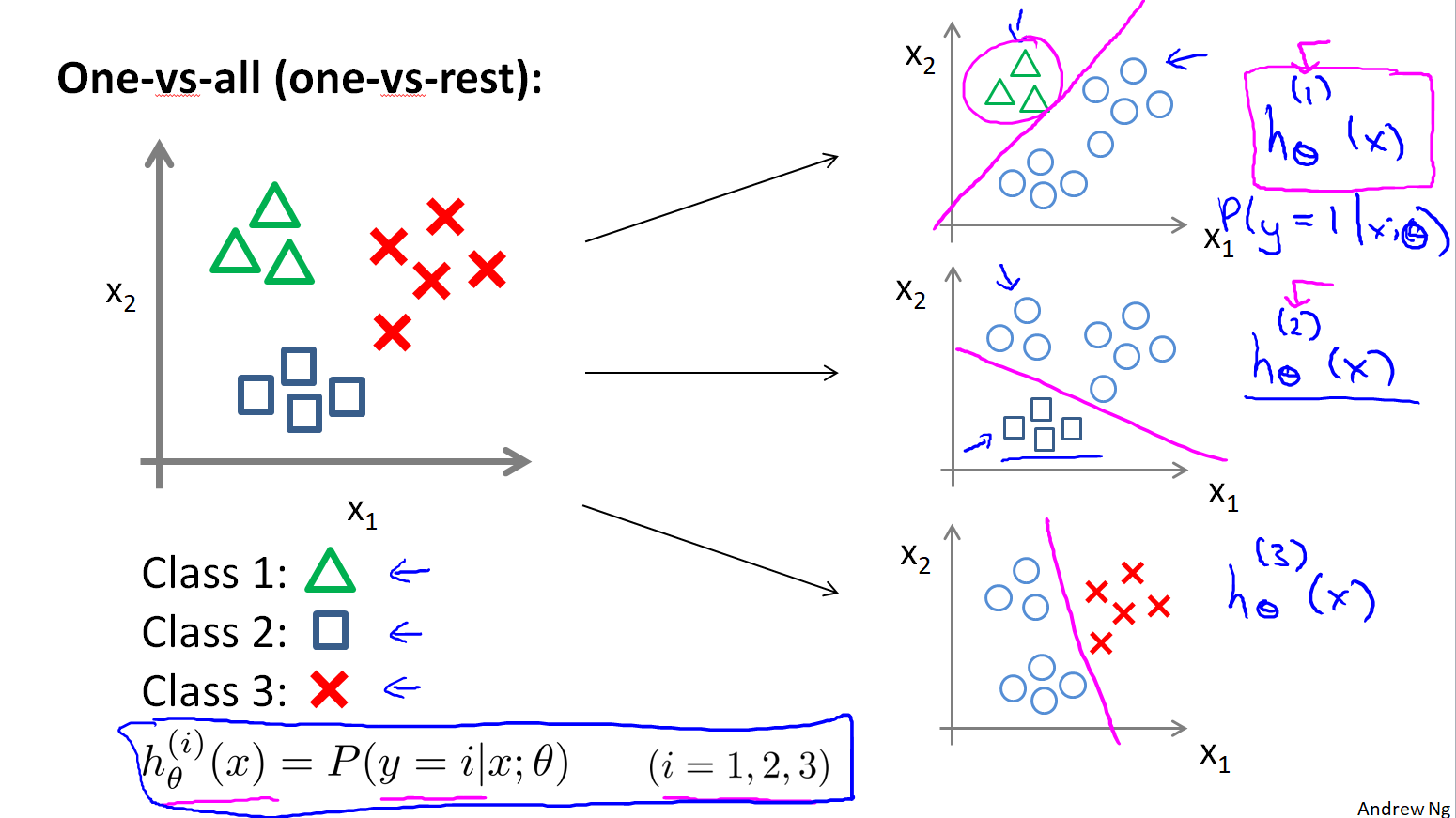
**options = optimset(‘GradObj’, ‘on’, ‘MaxIter’, ‘100’);**

**initialTheta = zeros(2,1);  
[optTheta, functionVal, exitFlag] ...**

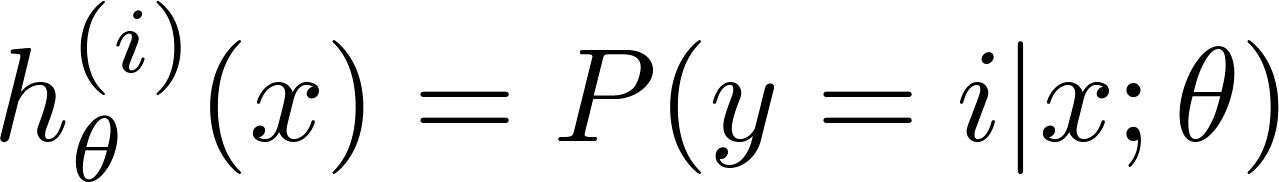
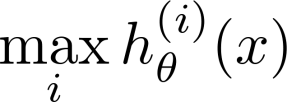
**= fminunc(@costFunction, initialTheta, options);**

**# GradObj是讓其自動去調整學習率，100是去試100組**

**Multi-class classification: One-vs-all (多重分類方法)**



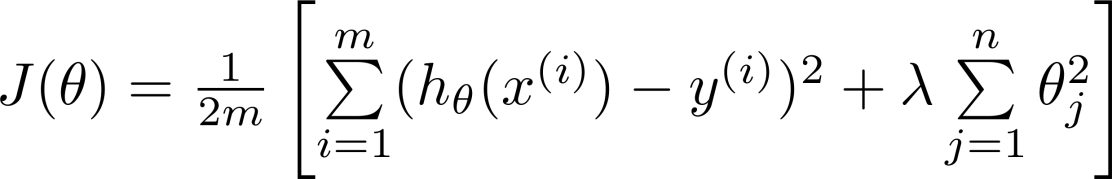
假如分三種就畫三條線更自分類成三種，最後再將其假設合併，取其最大值h(theta)即可以知道在該點是屬於哪一類

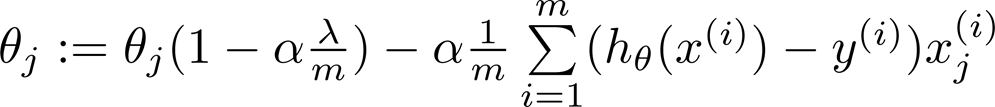
  

**Regularized 正規化(進階**

Regularized 正規化使用原理是將overfitting的多項式中後面J(theta)在1~n 中的項目做懲罰使之參數平方或次方等過大但是影響小的參數變更小，才不會overfit

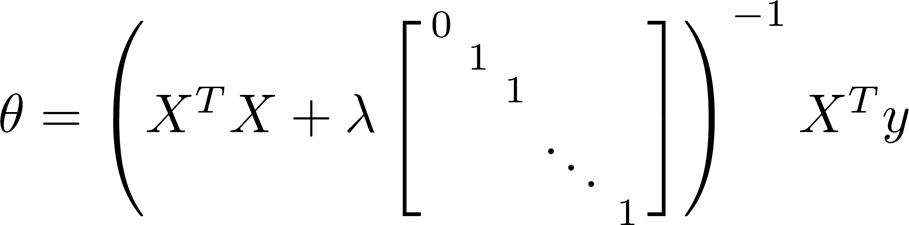
**Regularized linear regression 梯度下降**





Theta j後的參數會小於1但Normal 則不同

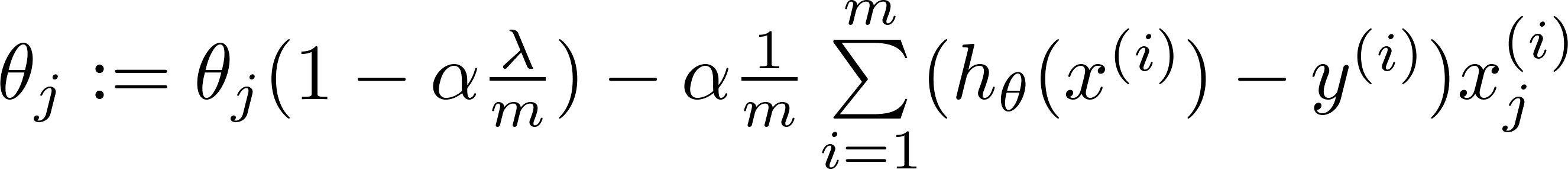
**Regularized linear regression Normal equation**



Almbta加上n+1項矩陣如果是大於0，注意有n個參數而矩陣是n+1項矩陣，而且矩陣第一項是0，因為正規化就是只對theta 1~theta n 懲罰

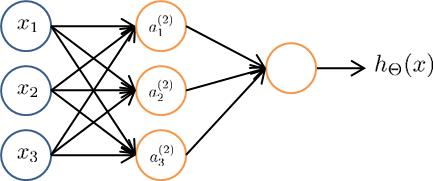
Almbta矩陣可以確保用正規化的normal equation 不會singulate一定有逆矩陣，不會degenerate無法求解

Logistic Regression的梯度下降

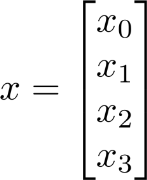
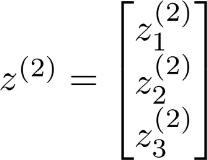


但 跟linear regression的h(theta)不同

**Neural Networks神經網路**



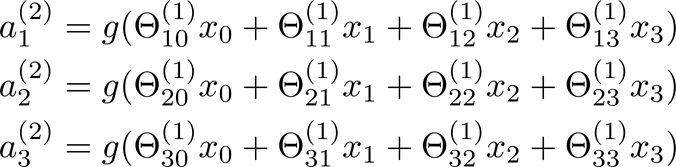
其中上圖的X1 X2 X3可以寫成一個向量並用 表示

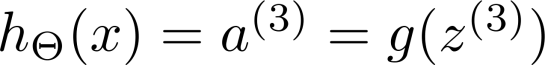
可加上 即變成bias unit或不加上(需要去試試哪種較好)

其中的 表示第一個hidden layer，在上圖也就是第二層。

而第一個隱藏層的各參數可以用以下來表示



而綠色的圈圈其實output layer也就是

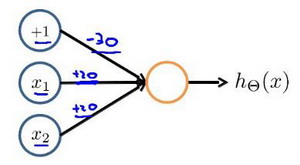
假設即是 其中

可以用隱藏層直接輸入X1 X2 X3，也可以試試直接把X1 X2 X3用多項式表示輸入可能會求出更好的解，可以把上圖的隱藏層與OUTPUT層看成是一個邏輯回歸

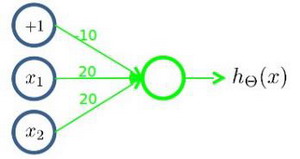
Logistic Regression

Neural Networks 直覺表示如何合成

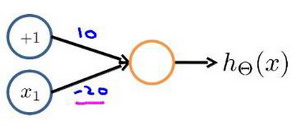
下圖的神經元（三個權重分別為-30，20，20）可以被視為作用同於邏輯與（AND）：



下圖的神經元（三個權重分別為-10，20，20）可以被視為作用等同於邏輯或（OR）：



下圖的神經元（兩個權重分別為 10，-20）可以被視為作用等同於邏輯非（NOT）：

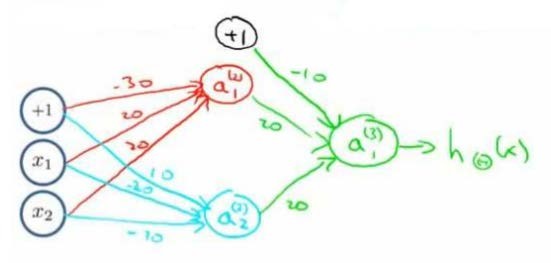


神經元可以將不同的組合變成更為複雜多元的運算，像是 XNOR 功能（輸入的兩個值必須一樣，均為1或均為0

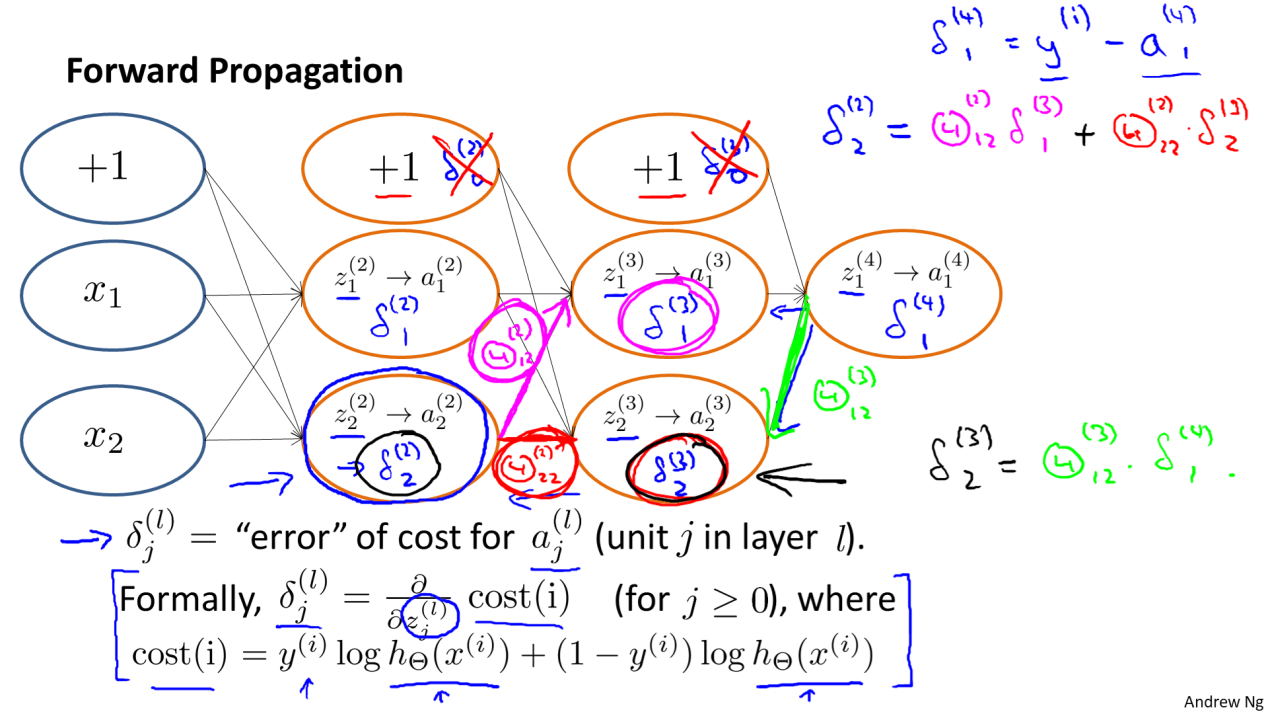
0

可以透過第一個參數用and 第二個參數用or合併，如紅色是and而淺藍是or

而最後是，變成XNOR 功能



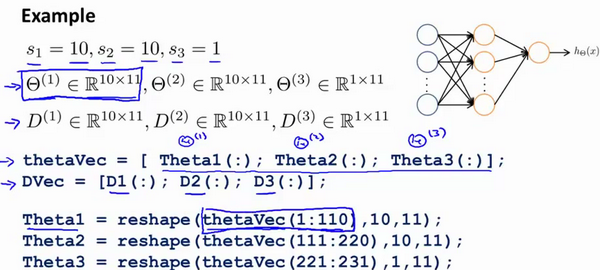
**Backpropagation反向傳播算法**



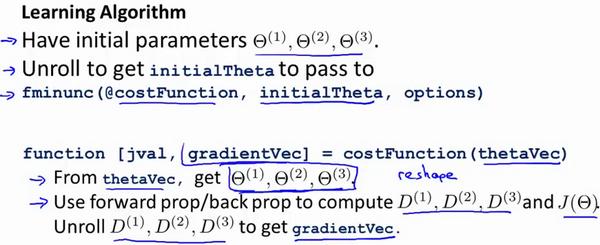
利用前向傳播求出假設即是而是輸出利用y(實際數據)-可求出參數，再利用來從後往前運算出各個隱藏層，而bias項(+1)可不列入，因為微分後沒有影響

**Unrolling Parameters (矩陣與向量互換)**

當參數是矩陣時有一大好處是對於深度學習來說可以輕易的運用前項傳播與後項傳播，而之後再將參數變成是向量可以對於一些更高階的優化程式都會需要變成一長列的向量才能執行



thetaVec是一個很長的向量而Theta1,2,3則是代表各假設的矩陣



利用矩陣運用前項傳播與後項傳播再去用向量去求的假設方程式

**Gradient Checking(梯度檢驗)**

當我們對一個較為複雜的模型（例如神經網絡）使用梯度下降算法時，可能會存在一些不容易察覺的錯誤，意味著，雖然代價看上去在不斷減小，但最終的結果可能並不是最優解。Gradient Checking 適用在100%複雜的模型，採用的方法是在代價函數上沿著切線的方向選擇離兩個非常近的點然後計算兩個點的平均值用以估計梯度。即對於某個特定的θ，我們計算出在θ-ε ​​處和θ+ε 的代價值（ε是一個非常小的值，通常選取0.001），然後求兩個代價的平均，用以估計在θ處的代價值

gradApprox = (J(theta + eps) – J(theta - eps)) / (2\*eps)

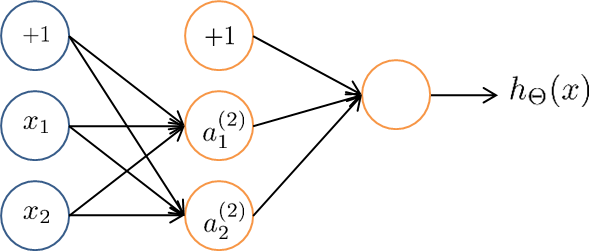
計算出每個假設參數的θ，舉例

第一先使用反向傳播計算求出各神經元的矩陣，再使用gradApprox來梯度檢驗，再確定各神經元的θ矩陣(即為線之間連的數字)，與gradApprox來梯度檢驗的結果相近後(可能差個小數點幾位數)，最後最重要是要把gradApprox來梯度檢驗迴圈關掉來只用反向傳播數據來學習，若不關掉會很慢，而且gradApprox來梯度檢驗只是為了要知道反向傳播數據是否正確

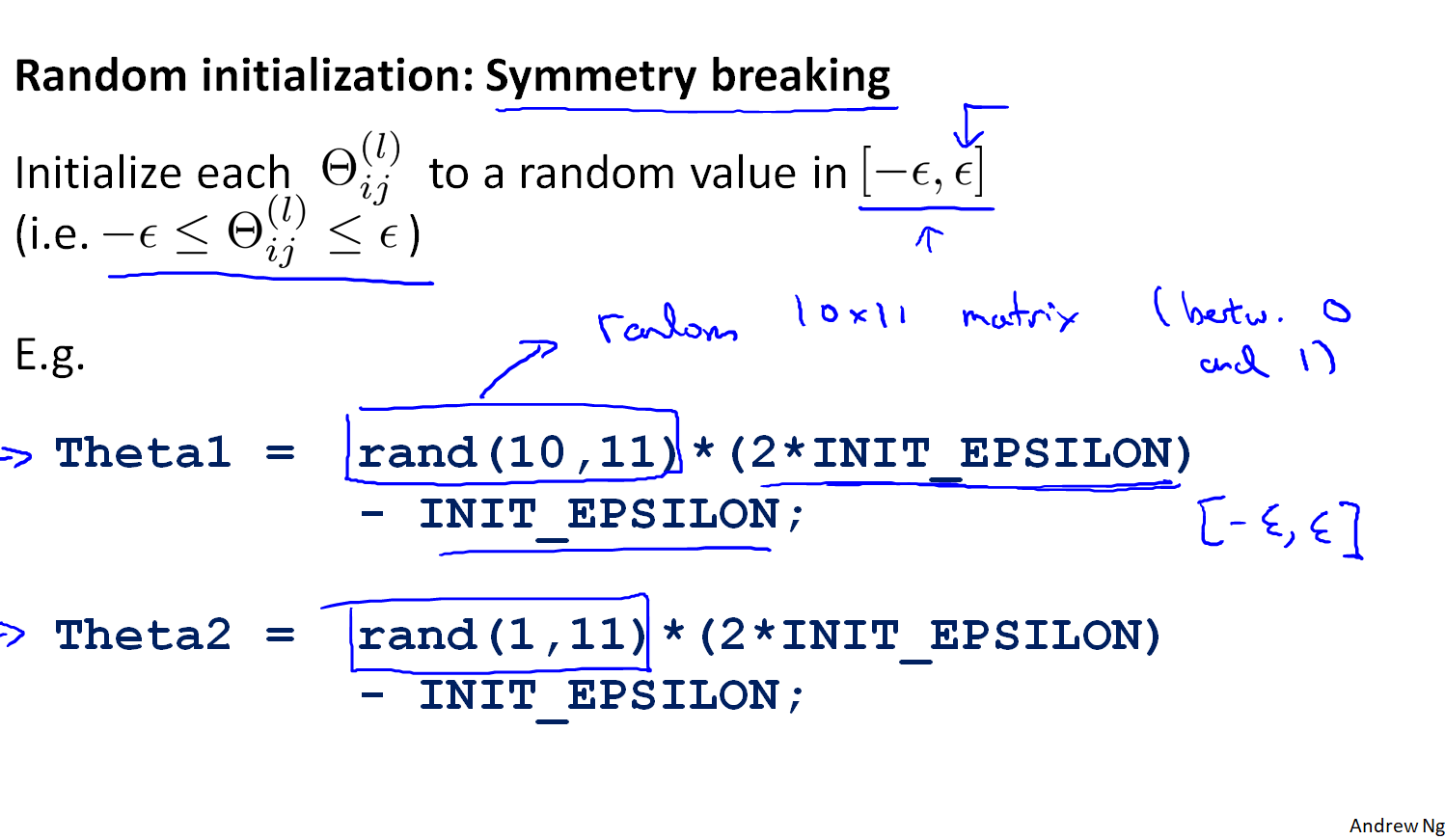
**Random initialization**

Logistic Regressiond可以使用zero initialization 但deep learning不可以用zero initialization，若是使用了，其各神經元的θ矩陣(即為線之間連的數字)會兩兩相同圖所示 1到a(2)1 和a(2)2的θ一樣而2到a(2)1 和a(2)2的θ一樣所以會讓

1到a(2)1 和a(2)2值都一樣



所以要用Random Initialization 用symmetry breaking 將初始θ介於[– eps，eps]



Theta1 = rand(10, 11) \* (2\*eps) – eps

而這個eps與前面的Gradient Checking(梯度檢驗)值不同，所以Gradient Checking(梯度檢驗)需要額外加值大多是在0.001之內

**總結deep learning**

使用神經網絡時的步驟：

網絡結構：第一件要做的事是選擇網絡結構，即決定選擇多少層以及決定每層分別有多少個單元。(每個隱藏層的unit是相同的而且大概多於input參數)

第一層的單元數即我們訓練集的特徵數量。(input parameter)

最後一層的單元數是我們訓練集的結果的類的數量。(output )

如果隱藏層數大於1，確保每個隱藏層的單元個數相同，通常情況下隱藏層單元的個數越多越好。真正要決定的是隱藏層的層數和每個中間層的單元數。

訓練神經網絡：

參數的隨機初始化(要用Random Initialization設定初始的θ使其介於[– eps，eps])

利用正向傳播方法計算所有的 

編寫計算代價函數 J 的代碼 

利用反向傳播方法計算所有偏導數(再用Backpropagation求出θ)

利用數值檢驗方法檢驗這些偏導數(再使用Gradient Checking把θ額外多增減0.001來測試是否是正確的)

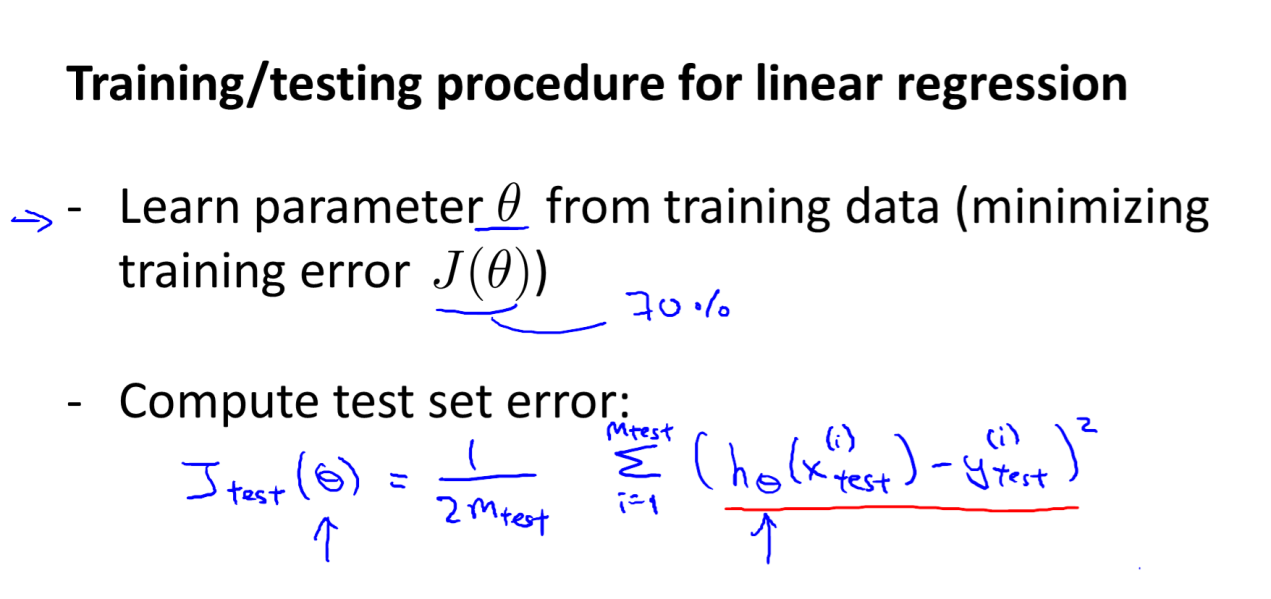
關掉再用Backpropagation求出θ來學習去訓練

使用優化算法來最小化代價函數(可使用梯度下降或是其他高級算法求解)

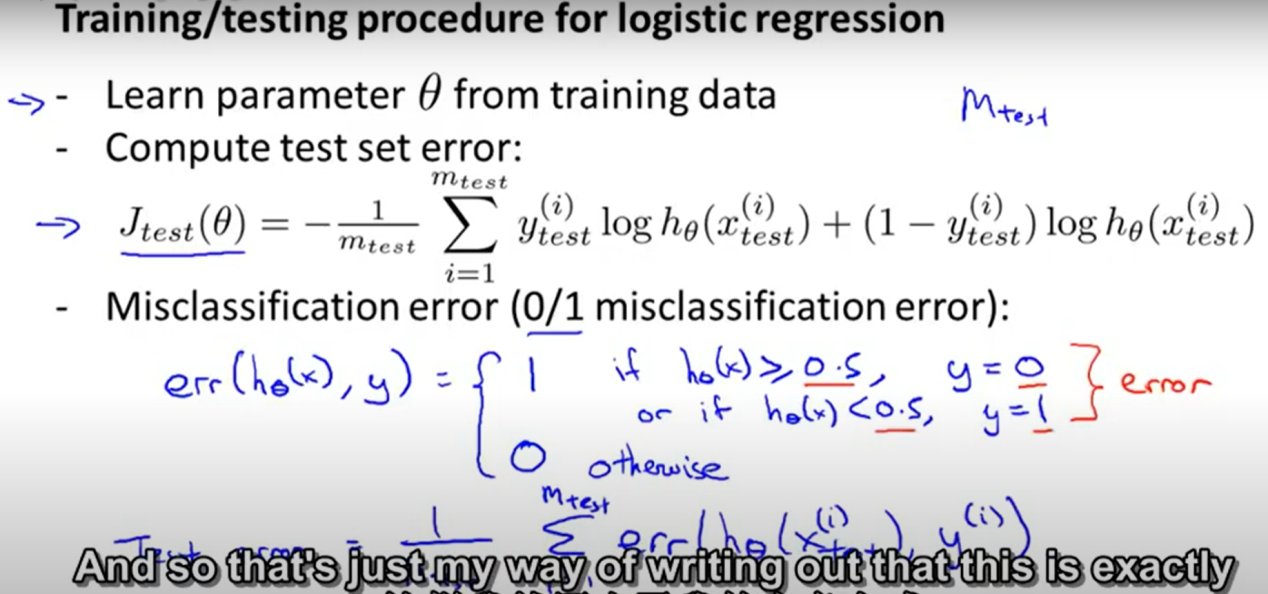
因為神經網絡的cost function是no convex function所以最優解只會是local mini但是已經會很接近最優解

**Evaluating a Hypothesis**

如何在有限的資料中驗證假設是否欠擬合或過擬合，可以利用將70%的資料來訓練當train data再將剩下30%資料當test data來驗證是否欠擬合或過擬合，再去試試其test error大小



此為 Linear regression



此為 Logistic regression

**Model Selection and Train\_Validation\_Test Sets**

首先先列舉不同次方(1-10次方)的方程式來確立假設方程式，將資料分成60%train set，20% validation set，20 % test set，將60%用在訓練出1-10次方的方程式，將20% validation set 的帶入10種方程式來取的哪種假設有最小的誤差，再用那種次方的方程式來輸入 test set 求出generalize error.即可避免overfit

1. 使用訓練集訓練出10個模型

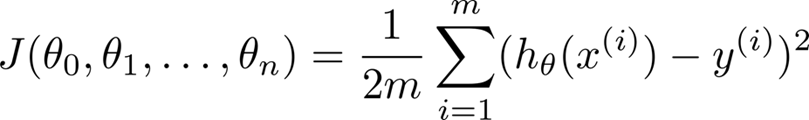
2. 用10個模型分別對交叉驗證集計算得出交叉驗證誤差（代價函數的值）

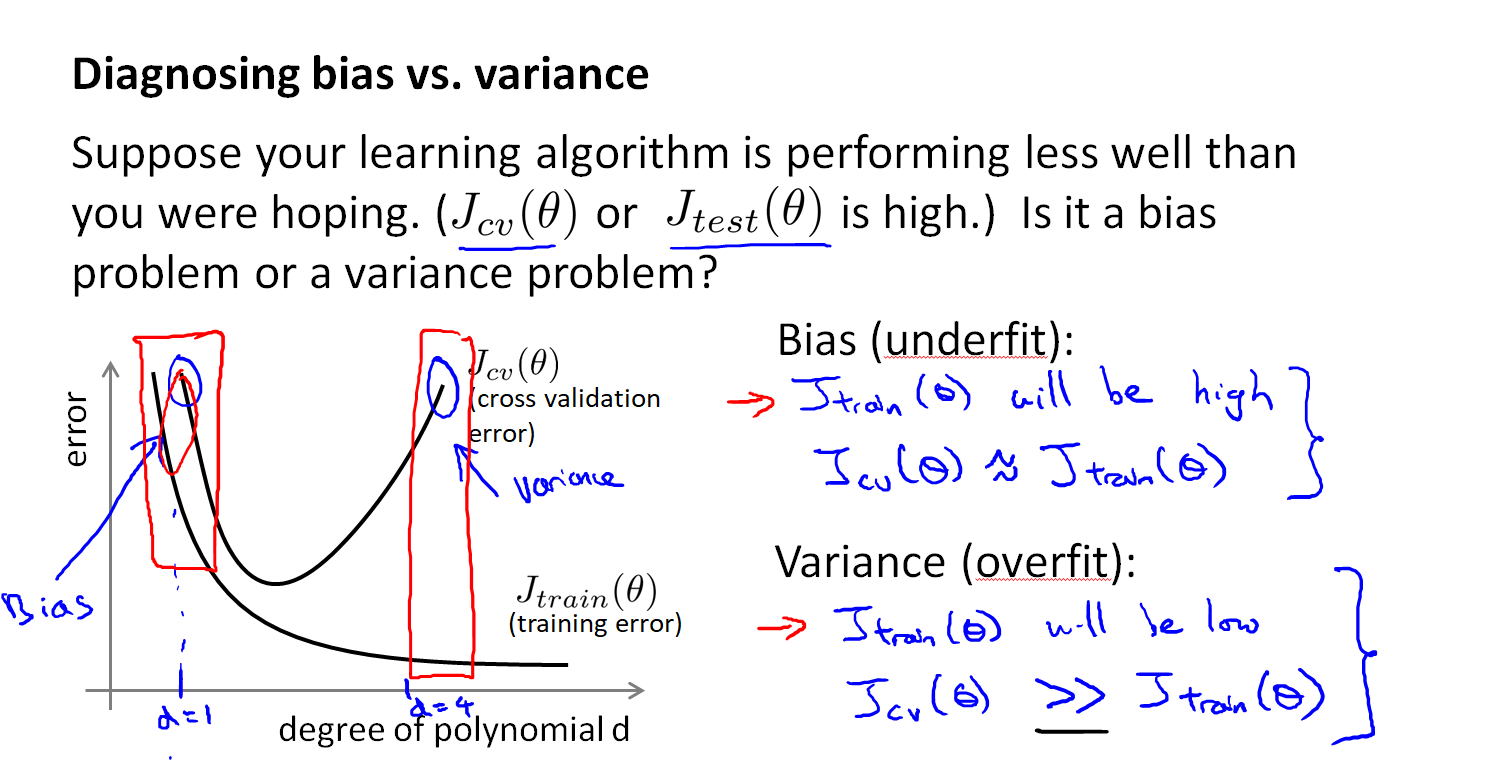
3. 選取代價函數值最小的模型

4. 用步驟3中選出的模型對測試集計算得出推廣誤差（代價函數的值）

**Diagnosing Bias vs. Variance**

如何判定欠擬合或過擬合，如下圖所示當cross validation set error 以及train set error都很大時，就是high bias是欠擬合.而當cross validation set error很高而train set error很低時則代表low bias and high variance是過擬合(其cost function如下)





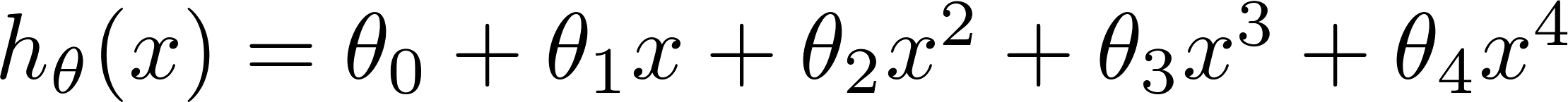
Bias ==偏差 Variance==方差，故要取中間兩者皆小的值才能generalize

**Regularization and bias/variance**

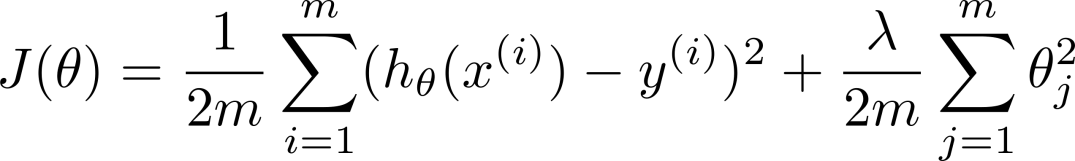
在訓練模型的過程中，一般會使用一些正則化方法來防止過擬合，而需選取λ 值，通常是 0-10之間的呈現2倍關係的值

（如：0,0.01,0.02,0.04,0.08,0.15,0.32,0.64,1.28,2.56,5.12,10.24 等共12個）

然後多項式如下



cost function如下



同樣把數據分成train set(60%) cross validation set (20%) test set (20%)

但train set，cross validation set，test set的cost function是沒有計算λ/2m只有假設的cost function有

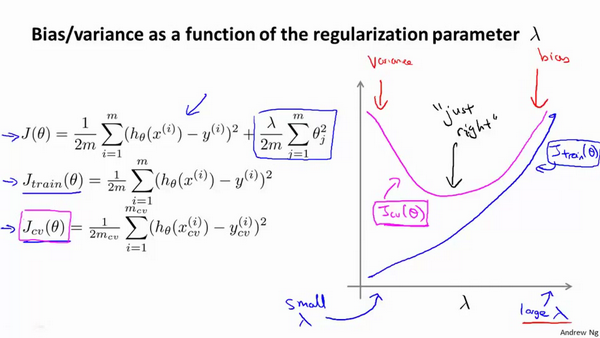
選擇λ的方法為：

1.使用訓練集train set(60%)訓練出12個不同程度正則化的模型

2.用12個模型分別對交叉驗證cross validation set集計算的出交叉驗證誤差

3.選擇得出交叉cross validation set驗證誤差最小的模型

4.運用步驟3中選出模型對測試集計算得出推廣誤差，我們也可以同時將訓練集和交叉驗證集模型的代價函數誤差與λ的值繪製在一張圖表上：



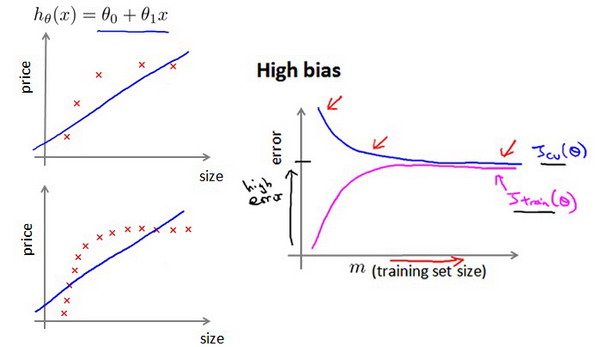
如圖所示

當λ過小時則正規化沒效果還是過擬合(當 λ 較小時，訓練集誤差較小（過擬合）而交叉驗證集誤差較大)

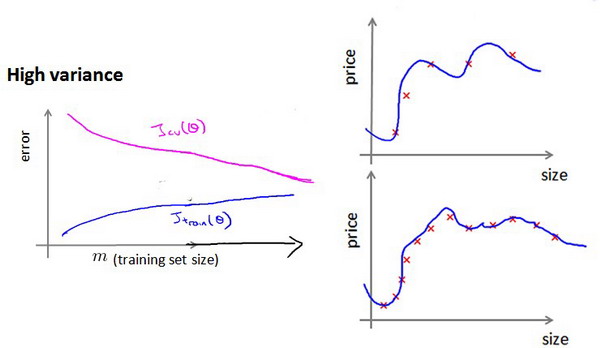
而λ過大時則正規化沒效果還是欠擬合.(隨著 λ 的增加，訓練集誤差不斷增加（欠擬合），而交叉驗證集誤差則是先減小後增加)

Learning Curves

學習函數就是將train set，cross validation set的誤差做訓練例子數目m的函數曲線圖，可運用來判斷是否過擬合或是欠擬合



在高偏差下增加訓練集的數量並不能幫助，還是欠擬合



而在高方差且正则化非常小當交叉驗證集誤差遠大於訓練集誤差情況下增加訓練集數目可以一定程度的幫助過擬合

**Deciding What to Do Next Revisited**

1. 獲得更多的訓練實例——解決高方差

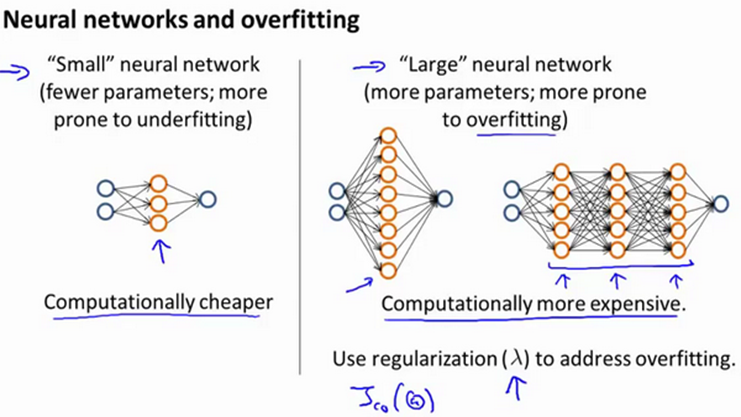
2. 嘗試減少特徵的數量——解決高方差

3. 嘗試獲得更多的特徵——解決高偏差

4. 嘗試增加多項式特徵——解決高偏差

5. 嘗試減少正則化程度λ——解決高偏差

6. 嘗試增加正則化程度λ——解決高方差



使用少層hidden layer易高偏差且欠擬合，使用多層hidden layer易高方差且過擬合且算力過高昂，所以用多層hidden layer配合正規化最適合

一般多用先用單層隱藏層再慢慢往上加隱藏層，然後用train set 訓練完後將cross validation set 帶入取最小cost function將其設為假設

**Prioritizing What to Work On**

舉例來說算法用以分類是否是垃圾信件，像是從信的標題提取特徵，信的正文提取特徵，增加data，把刻意探測刻意的拼寫錯誤（把watch 寫成w4tch）開發複雜的算法

**Error Analysis**

對於許多工程師來說很少人使用Error Analysis，這是一個非常重要的技巧可以幫助省下好幾個月的時間，朝著正確方向前進，所以首先，應該快速的實行很多簡易且不太好的演算法針對不同特徵feature再利用cross validation來看出此特徵對於這類問題的效益，畫出learning curve驗誤差，看是否有高偏差或高誤差的問題在這樣分析之後，再來決定用更多的數據訓練，或者加入更多的特徵變量是否有用，因為剛接觸問題時不知道要不要負責的特徵，或是更多數據，所以很難知道要改進算法哪部分，開始時快速的實行很多簡易且不太好的演算法可以提供證據來指引演算法優化方向

必須人工去看那些類型的郵件被錯誤的分類就能啟發你找到新的特徵變量，或者告訴你：現在這個系統的短處，然後如何去改善它

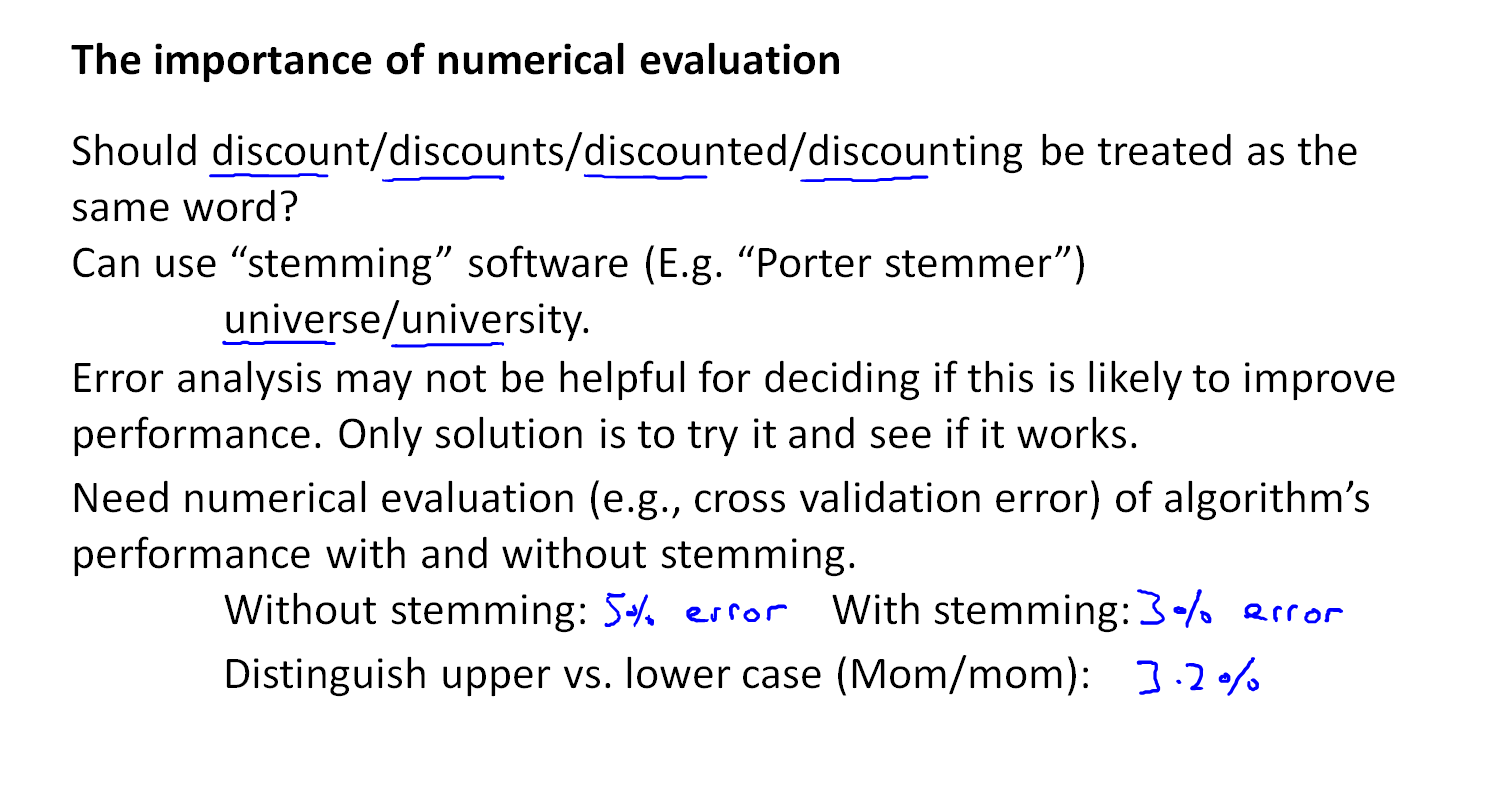
1. 從一個簡單的能快速實現的算法開始，實現該算法並用交叉驗證集數據測試這個算法

2. 繪製學習曲線，決定是增加更多數據，或者添加更多特徵，還是其他選擇

3. 進行誤差分析：人工檢查交叉驗證集中我們算法中產生預測誤差的實例，看看這些實例是否有某種系統化的趨勢

以垃圾郵件過濾器為例，誤差分析要做的既是檢驗交叉驗證集中我們的算法產生錯誤預測的所有郵件，看：是否能將這些郵件按照類分組。例如醫藥品垃圾郵件，仿冒品垃圾郵件或者密碼竊取郵件等。然後看分類器對哪一組郵件的預測誤差最大，並著手優化。

思考怎樣能改進分類器。例如，發現是否缺少某些特徵，記下這些特徵出現的次數。例如記錄下錯誤拼寫出現了多少次，異常的郵件路由情況出現了多少次等等，然後從出現次數最多的情況開始著手優化。誤差分析並不總能幫助我們判斷應該採取怎樣的行動。有時我們需要嘗試不同的模型，然後進行比較，在模型比較時，用數值來判斷哪一個模型更好更有效，通常我們是看交叉驗證集的誤差。



在我們的垃圾郵件分類器例子中，對於“我們是否應該將discount/discounts/discounted/discounting處理成同一個詞？”如果這樣做可以改善我們算法，可用一些stemming softwarm(截詞軟件)。誤差分析去嘗試採用和不採用截詞軟件這兩種不同方案或是到底是否使用詞幹提取，是否區分大小寫。，然後根據數值檢驗的結果來判斷哪一種更好。

所以需化量化的數值評估，因為去嘗試很多新的想法要手動地檢測這些例子，去看看是表現差還是表現好，用量化才能依據事實評估，這會大大提高你實踐算法時的速度。所以要在交叉驗證集上來實施誤差分析，而不是在測試集上。

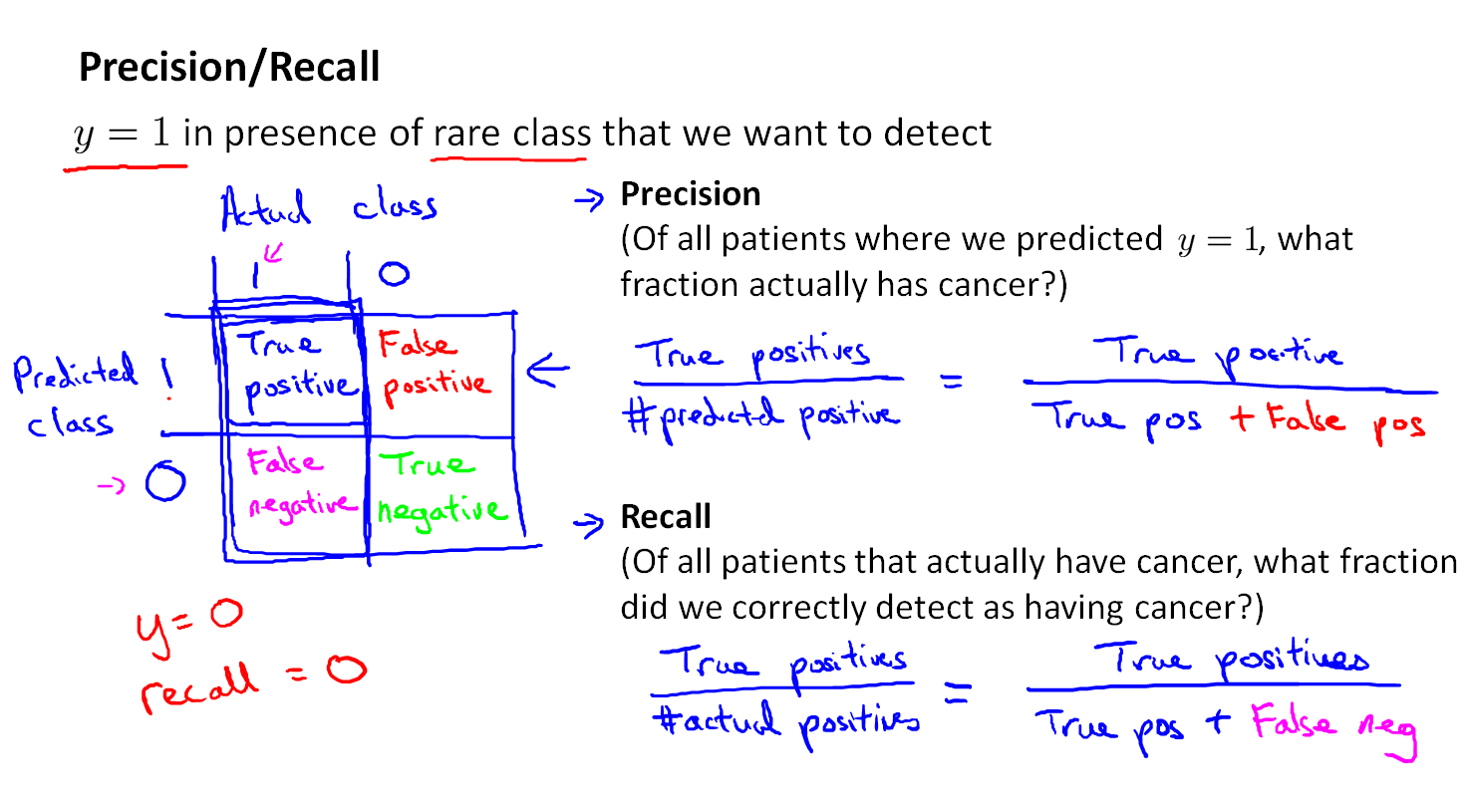
當在研究一個新的機器學習問題時許多人花費大量的時間在構造算法上，構造他們以為的簡單的方法。但實際上需要先建立快速、即便不是那麼完美的算法，來幫助你決定下一步的做法。

透過看看算法造成的錯誤，通過誤差分析，然後來決定優化的方式。此外也能會幫助你嘗試新的想法，快速地發現你嘗試的這些想法是否能夠提高算法的表現，從而你會更快地做出決定，在算法中放棄什麼，吸收什麼誤差分析可以幫助我們系統化地選擇該做什麼。

**Error metrics for skewed classes**

skewed classes 是指在一個數據集中，如果某一類樣本數量遠遠小於另一類樣本數量，那麼將這種情況稱為偏斜類問題。

在skewed classes下，一些評價指標會出現問題（例如分類誤差或分類準確率）。例如，在癌症鑑別任務中，測試集中只有0.5%的患者真正得了癌症，那麼如果我們的分類模型得到1%的錯誤率（看起來還不錯），這其實就是有嚴重問題的，因為我們的誤報比實際高出整整兩倍。進一步說，如果我們設計一個算法讓輸出=0（也就是總是預測人沒有得癌症），那麼在上述測試集中，模型就可以得到0.5%的錯誤率看起來跟複雜的算法誤差一樣 ，但這會是毀滅性的、會造成嚴重診斷失誤的0.5%！

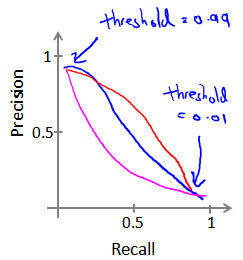


把y=1設成例子較少的要尋找項目，像是癌症研究，y=1就是多少人的癌症例子

所以要透過precision 跟 recall 來驗證，precision是在算法中預測有得癌症的例子中有幾個真的預測有得癌症且真的得癌症比例，recall是在所有有得癌症的例子中有幾個真的預測有得癌症且真的得癌症比例，所以算法要越高的precision或是recall才是好的算法

**Trading off precision and recall**

查准率和查全率的取捨，precision是在算法中預測有得癌症的例子中有幾個真的預測有得癌症且真的得癌症比例，recall是在所有有得癌症的例子中有幾個真的預測有得癌症且真的得癌症比例，而precision和recall不太可能都會很高，兩者會有魚與熊掌不可兼得如圖下



所以會採用個F1(SCORE)

取F1值最高的當假設方程式

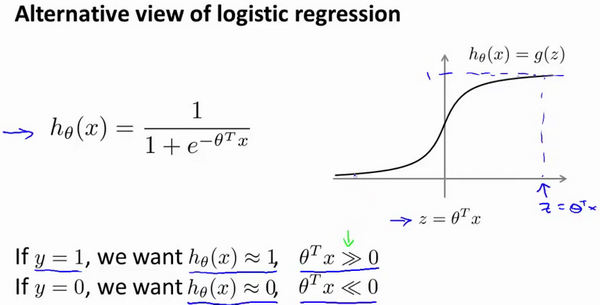
**Data For Machine Learning**

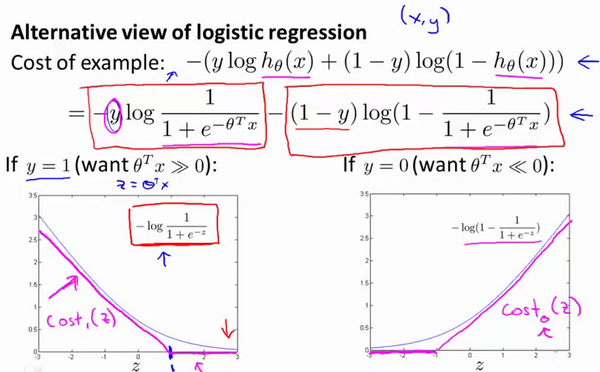
使用大量數據在何時能夠有效的幫助算法更加精確，第一是要讓bias下降，所以算法的參數要夠多就不會欠擬合，這樣train set cost會下降然後透過大量數據就能降低varience使之不會過擬合，test set cost就會跟train set cost接近，業就會讓test set cost也很小

所以要做假設時要問問自己，如果一個專家看到了特徵值 x，能有信心的預測出y值嗎？ 若y可以根據特徵值x被準確地預測出來代表X是很有用的參數。其次，能得到一組龐大的訓練集，並且在訓練集中訓練出一個有很多參數的學習算法嗎？ 如果都可以才是個有強大的算法

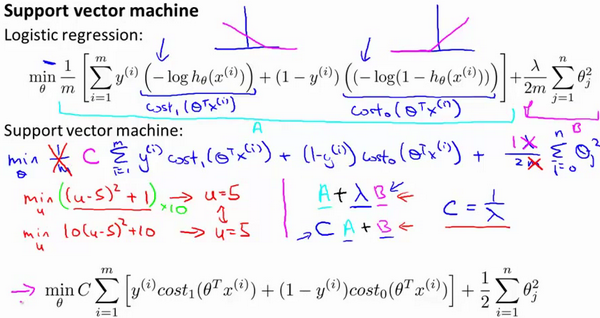
**SVM Optimization Objective**

Support Vector Machine支持向量機，對於一般的邏輯回歸如下圖所示





E是數學常數2.7182可以將判斷是y=1的在z軸 z=1做轉折點分成兩個斜線故需要Z>=1，再將判斷是y=0的在z軸 z=-1 做轉折點分成兩個斜線故需要Z<=-1



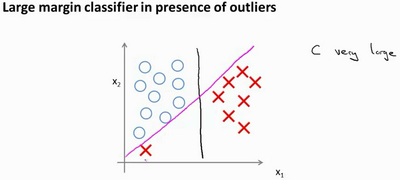
SVM相當於把Logistic regression 在於其cost funtion同除M(因為是常數而不影響)，而差別在 Logistic regression的cost function = (A+λb) 後面有正規化(λ是學習率 B是regularzation)，其主旨是不改變A而是控制λ來求最小函數，而SVM則是將不改變B而是控制C來求最小函數，其中C就是控制A的參數可以將其視為1/λ，因為函數的乘除不會影響求其最小值

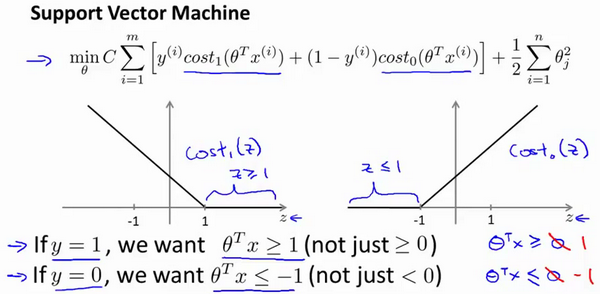
**Large Margin Intuition**

SVM costfunction 是 CA+B所以當設定C很大時也就是C=1/λ(λ很小)會容易受到極端值影響，會從黑線變成粉線，而當C很小時也就是C=1/λ(λ很大)不易受到極端值影響，會維持在黑線

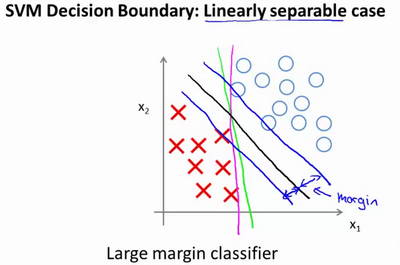
C 較大時，相當於 λ 較小，可能會導致過擬合，高方差。

C 較小時，相當於 λ 較大，可能會導致低擬合，高偏差。

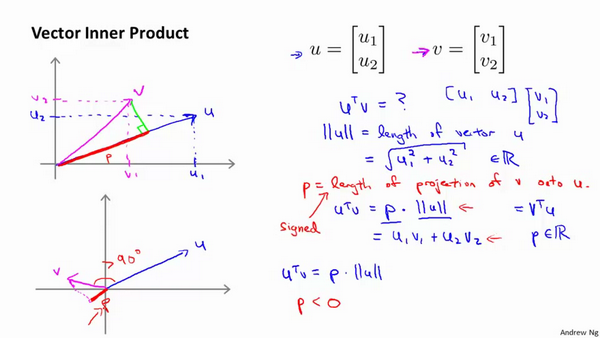




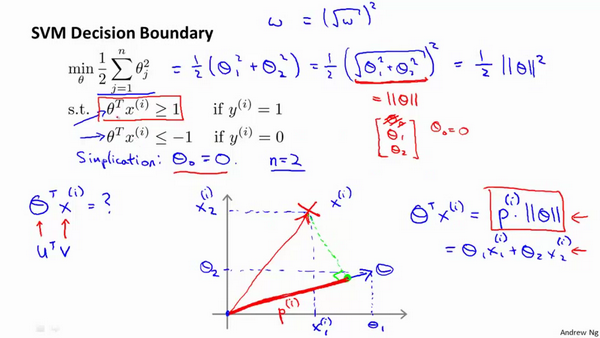
而SVM又叫做大間距分類器Large margin classifier，其中如下圖兩個藍線的間距就叫做Margin，其應該是如上圖當y=1的但z軸Z<=1加上y=0但z軸Z>=-1 時的Margin

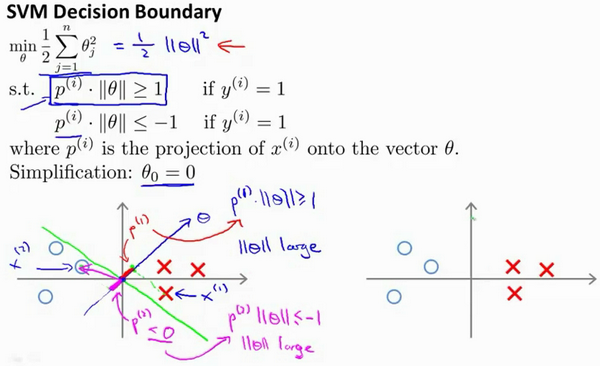


**Mathematics Behind Large Margin Classification**



=是V在U上面的投影

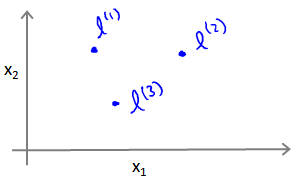




P(i)是向量的投影，而圖左可看出若其越靠近邊界線，則p(i)數字很小那||theta||就必須很大，那就不是好的SVM算法，而圖右可看出若其越遠離邊界線，則p(i)數字很大(至少大於一)那||theta||就很小，那就是很好的SVM算法

**Kernels**

假設方程式是…後令 )的近似程度來選取新的特徵。



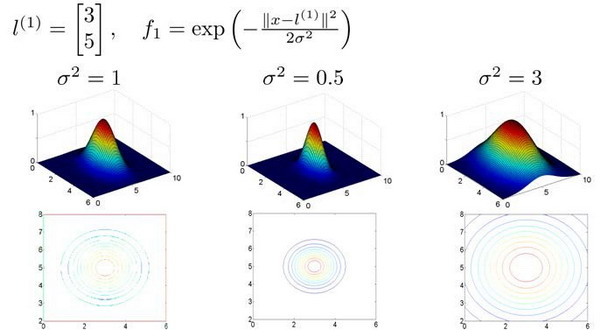
假設使用一種高斯核函數：

為實例x中所有特徵與地標之間的距離的和。上例中的similarity(x,l^((1)))就是核函數，具體而言，這裡是一個高斯核函數(Gaussian Kernel)。注：這個函數與正態分佈沒什麼實際上的關係，只是看上去像而已。

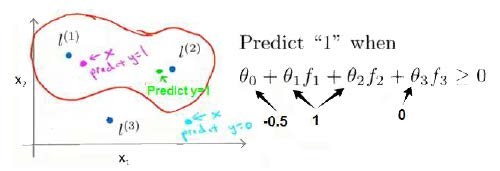
如果一個訓練實例x與地標L之間的距離近似於0，則新特徵f近似於e^(-0)=1，

如果訓練實例x與地標L之間距離較遠，則f近似於e^ (-(一個較大的數))=0。

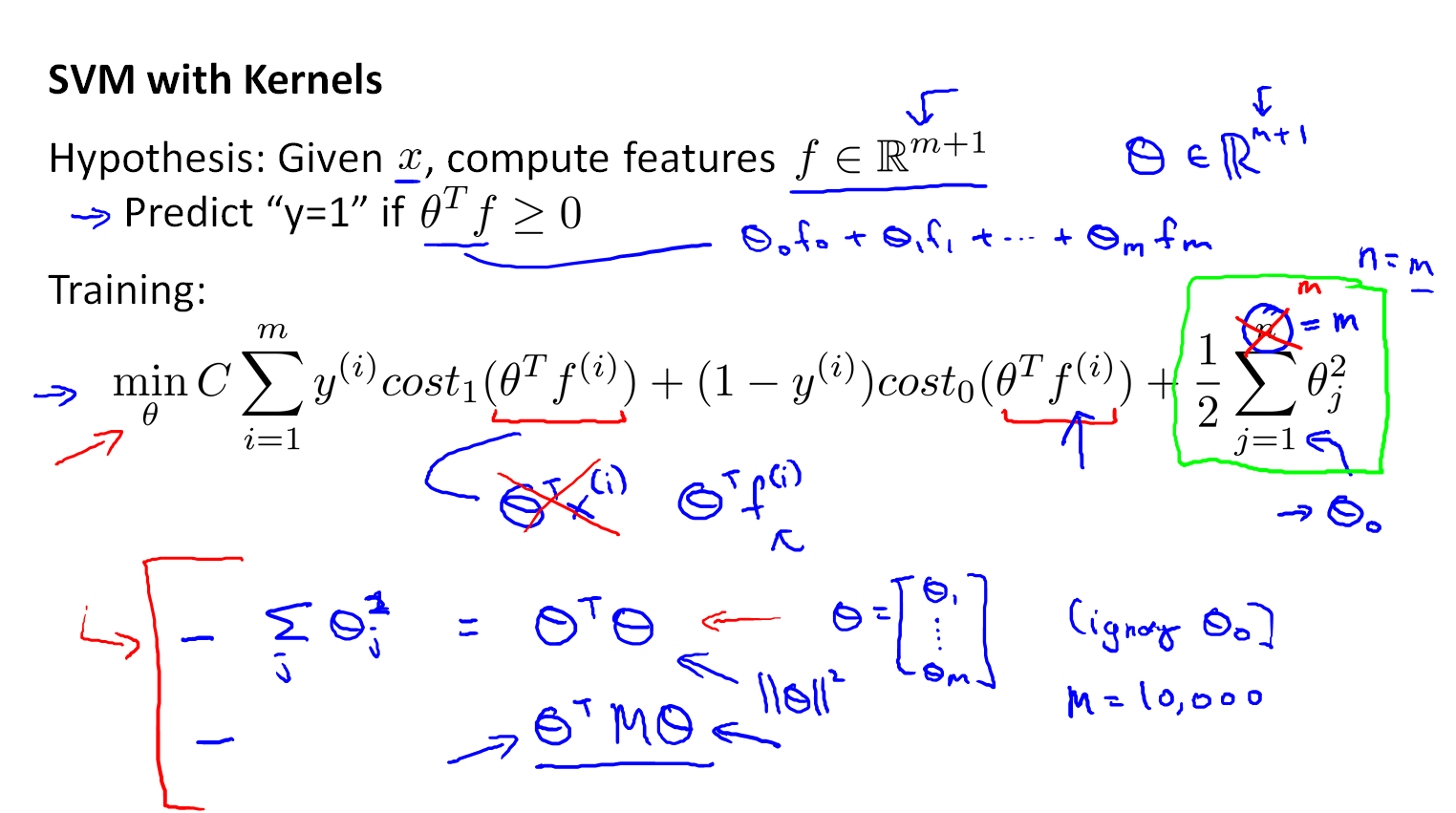
假設我們的訓練實例含有兩個特徵[x\_1 x\_2]，給定地標l^((1))與不同的σ值，見下圖：



圖中坐標為 x1，x2而垂直坐標軸代表f。當x1,x2與重合時f才具有最大值。隨著x的改變f值改變的速率受到σ^2的控制，σ^2越大變化越慢，σ^2越小變化越快



而當粉紫色的點離f1進，而遠離f2 f3 遠時，近的部分就會是1而遠離的兩點會算成是0所以就能將假設的方程式算出此時是大於0，但對淡藍色的點三點都距離其很遠所以都是0帶入後是負的所以在邊界外面。透過這種算法數字可以得出其邊界，在預測時，我們採用的特徵不是訓練實例本身的特徵，而是通過核函數計算出的新特徵f1,f2,f3



最後面的正規化沒有從0是因為theta0是常數不用正規化，可將其從一到m個例子，建議將參數很多像是破萬時(具體化技術embodied)，把向量改成矩陣可減少運算量，此外Kernel核運算可用在其他邏輯回歸等等但是不好使用會很慢所以不建議

如何對SVM的參數和做取捨

大時則，也就是正規化較小，所以是過擬合，高方差低偏差

小時則，也就是正規化較大，所以是欠擬合，低方差高偏差

較大时，也就是正規化較大，所以是欠擬合，低方差高偏差

較小時，也就是正規化較小，所以是過擬合，高方差低偏差