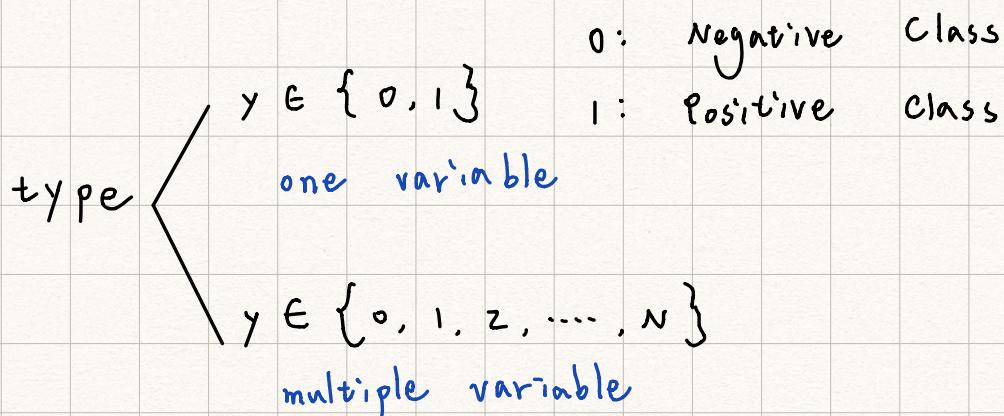


Classification and Representation

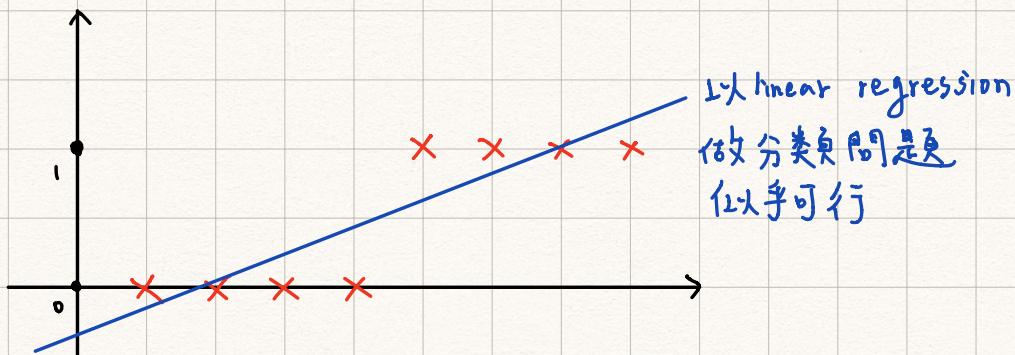
classification

examples: spam / not spam?

Malignant / Benign?



①



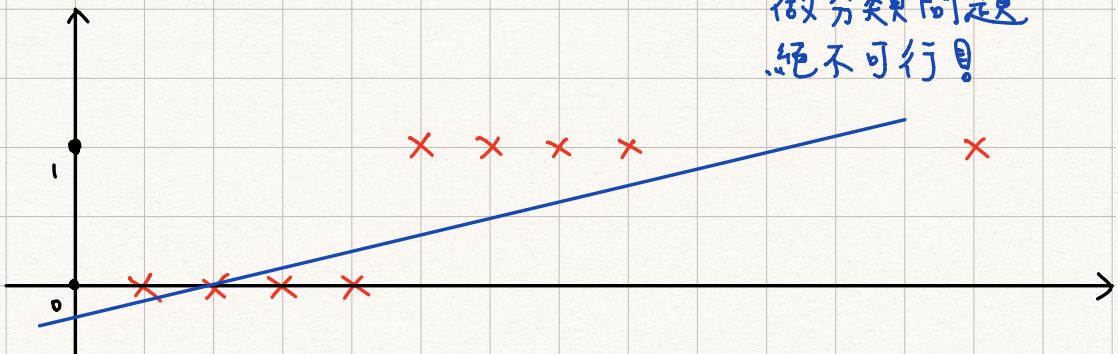
$$h_{\theta}(x) = \theta^T x$$

(linear regression)

$h_{\theta}(x) \geq 0.5$, predict $y=1$

$h_{\theta}(x) < 0.5$, predict $y=0$

②



由 linear regression
做 分類 題
絕不可行！

Use what algorithm to classification problem

classification : $y = 0 \vee 1$

{ linear regression : $h_{\theta}(x)$ can be > 1 or < 0
logistic regression : $0 \leq h_{\theta}(x) \leq 1$ ✓
for classification

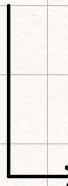
Hypothesis representation

want $0 \leq h_{\theta}(x) \leq 1$

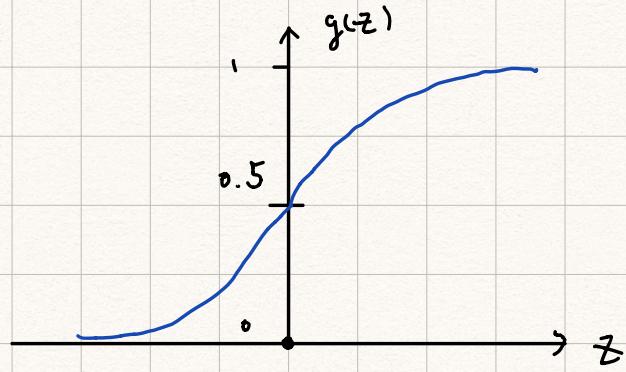
$$h_{\theta}(x) = g(\theta^T x), \quad g(z) = \frac{1}{1 + e^{-z}}$$

→ sigmoid function

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$



$$h_{\theta}(x) = P(y=1 | x; \theta)$$



probability that $y=1$, given x , parameterize by θ .

$$P(y=0 | x; \theta) + P(y=1 | x; \theta) = 1$$

Decision boundary

根據 sigmoid function 的圖形

Suppose predict " $y=1$ " if $h_{\theta}(x) \geq 0.5$

$$\theta^T x \geq 0$$

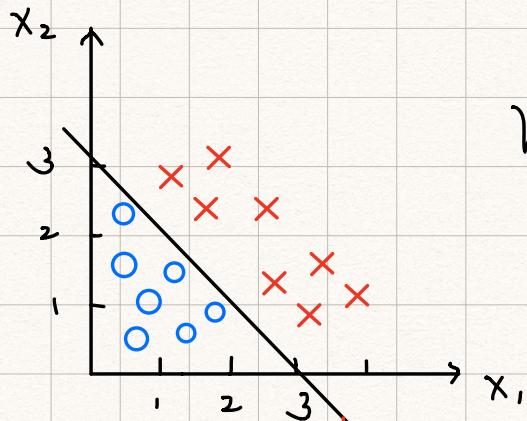
predict " $y=0$ " if $h_{\theta}(x) < 0.5$

$$\theta^T x < 0$$

in sigmoid function, $g(z) \geq 0.5$ when $z \geq 0$

$g(z) < 0.5$ when $z < 0$

linear decision boundary



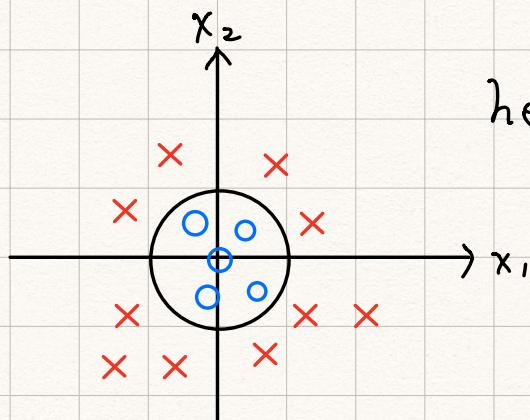
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$\theta = \begin{bmatrix} -3 \\ ; \\ ; \end{bmatrix}$$

$x_1 + x_2 = 3$ decision boundary

"y=1" if $-3 + x_1 + x_2 \geq 0$, $x_1 + x_2 \geq 3$

non-linear decision boundaries



$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ ; \end{bmatrix}$$

"y=1" if $-1 + x_1^2 + x_2^2 \geq 0$, $x_1^2 + x_2^2 \geq 1$

Logistic Regression Model

Cost function of logistic regression

training set : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(n)})\}$

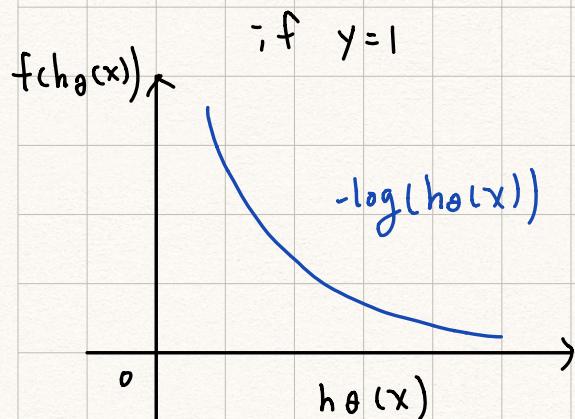
m examples :

$$x \in \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \quad x_0 = 1, y \in \{0, 1\}$$
$$x_1, \dots, x_n \in \mathbb{R}$$

$$h_{\theta}(x) : \frac{1}{1 + e^{-\theta^T x}}$$

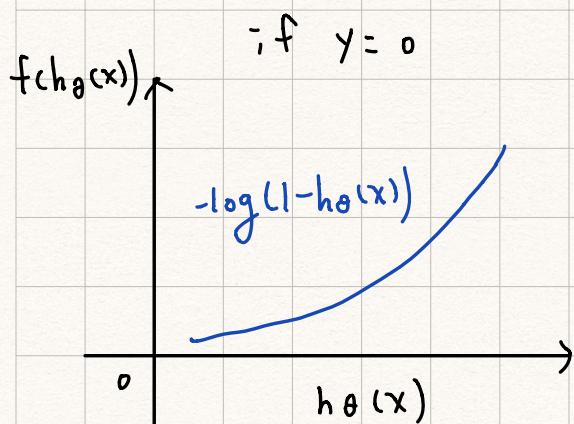
How to choose cost function?

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y=1 \\ -\log(1-h_{\theta}(x)) & \text{if } y=0 \end{cases}$$



$$\text{Cost} = 0 \quad \text{if } y=1 \quad h_{\theta}(x)=1$$

But as $h_{\theta}(x) \rightarrow 0$
 $f(h_{\theta}(x)) \rightarrow \infty$
 $\text{cost} \rightarrow \infty$



$J(\theta) = 0 \text{ if } y = 0$
 $J(\theta) = \infty \text{ if } y = 1$

But as $h_\theta(x) \rightarrow 1$
 $J(h_\theta(x)) \rightarrow \infty$
 $J(\theta) \rightarrow \infty$

Simplified cost function of logistic regression

Note: $y = 0$ or $y = 1$ always

$$J(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Simplify \Rightarrow

$$J(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$$

So, Logistic cost function is

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

**

Gradient descent of logistic regression

$$\text{repeat } \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

(simultaneously update all θ_j)

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \text{ for } i=0 \text{ to } n$$

linear regression : $h_\theta(x) = \theta^T x$

logistic regression : $h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$

Advanced Optimization

加速學習速度，適用更多特徵的大型問題

Optimization algorithms :

- Gradient descent

- Conjugate gradient
 - BFGS
 - L-BFGS
- } • No need to manually pick α .
- faster than GD
- more complex

Octave function for advanced optimization

`costFunction . optimset . fminunc`

`optimset ('GradObj', 'on', 'MaxIter', '100')`

`[optTheta, functionVal, exitFlag]`

= `fminunc (@costFunction, initialTheta, options)`

\downarrow 維度需 > 2

`function [JVal, gradient] = costFunction(theta)`

`JVal = [code to compute J(theta)]`

gradient (1) = [code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$]

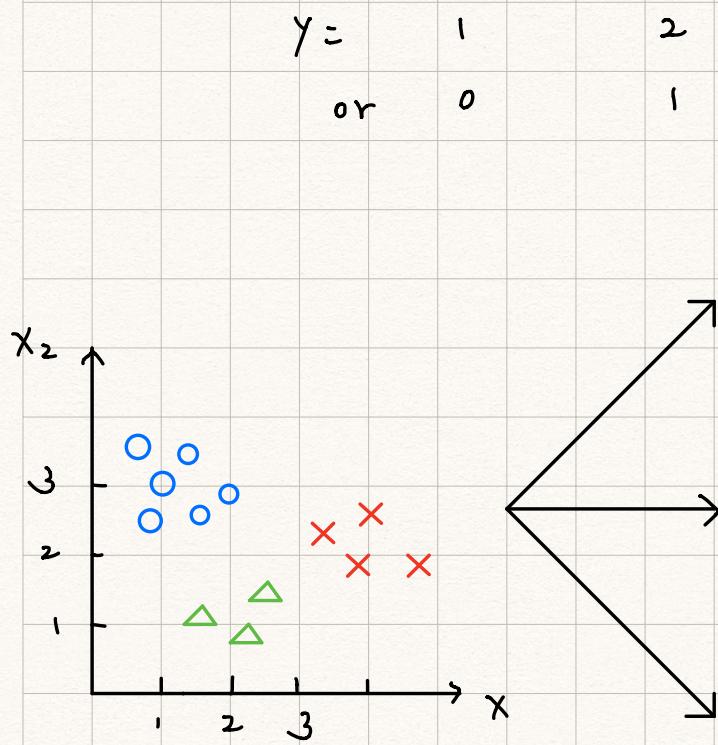
gradient (2) = [code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$]

⋮

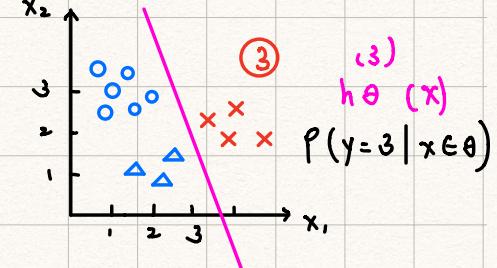
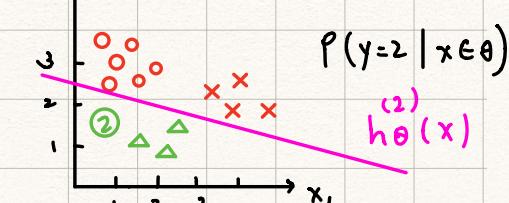
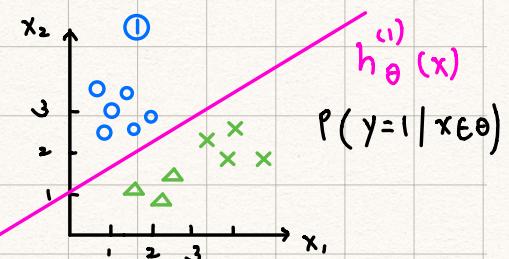
gradient (n+1) = [code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$]

Multi-class classification

Weather: Sunny . Cloudy , Rain , Snow



$$h_{\theta}^{(i)}(x) = P(y=i \mid x; \theta)$$

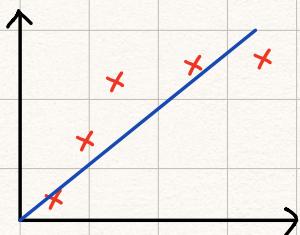


On a new input x , to make a prediction, pick the class i that maximizes

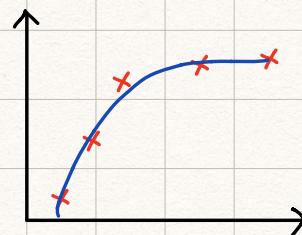
$$\max_i h_{\theta}^{(i)}(x)$$

Solving the problem of Overfitting

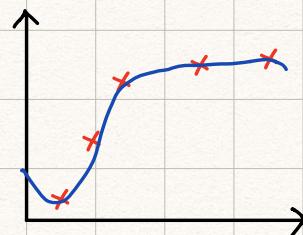
The problem of overfitting



"underfit"
"high bias"



"just right"



"overfit"
"high variance"

Overfitting: if we have too many features, the learned hypothesis may fit the training set very well, but fail to new examples (predict prices on new examples).

→ 訓練集的下降速度遠大於驗證集

→ 驗證集的損失過段時間後反而上升

Addressing overfitting

1. Reduce number of features.

- Manually select which features to keep.
- Model selection algorithm.

2. Regularization

- keep all the features, but reduce magnitude or values of parameters θ_j .
- works well when we have a lot of features, each of which contributes a bit to predicting y .

Regularization

Small values for parameters $\theta_0, \theta_1, \dots, \theta_n$:

- "Simpler" hypothesis
- Less prone to overfitting

Housing:

- Features: x_1, x_2, \dots, x_{100}

- Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

$$\min_{\theta} J(\theta)$$

regulation parameter



- if λ (regularization parameter) is extremely large, then the algorithm results in underfitting.

$\lambda \uparrow$, 應該 θ_j 的力度越大
 θ_j 越近 zero.

a way to choose regularization value automatically.

Regularized linear regression

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$$\underbrace{\quad}_{\frac{\partial}{\partial \theta_j} J(\theta)} \quad j = 1, 2, 3, \dots, n$$

regualrized term

$$\theta_j := \theta_j \underbrace{\left(1 - \alpha \frac{\lambda}{m}\right)}_{< 1, \text{ like } 0.99} - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Normal equation

$$X = \begin{bmatrix} (x^{(1)})^\top \\ \vdots \\ (x^{(m)})^\top \end{bmatrix} \quad m \times (n+1)$$

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad I \mathbb{R}^m$$

$$\min_{\theta} J(\theta)$$

$$\Theta = \left(X^T X + \lambda \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & & & \\ \vdots & & \ddots & & \vdots \\ 0 & & \dots & 1 & 0 \end{pmatrix} \right)^{-1} X^T y$$

(n+1)(n+1)

Non-invertibility (optional / advanced)

Suppose $m \leq n$

:	:
example	feature

$$\Theta = (X^T X)^{-1} X^T y$$

If $\lambda > 0$, this matrix is invertible

$$\Theta = \left(X^T X + \lambda \begin{pmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \right)^{-1} X^T y$$

Regularized logistic regression

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

}

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

$j = \theta_1, \theta_2, \dots, \theta_n$

regularized term

$$- \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j$$

$$= - \frac{1}{m} \sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

programming *

function [jVal, gradient] = costFunction(theta)

jVal = {code to compute $J(\theta)$ }

gradient(1) = {code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$ }

$$\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

gradient(2) = {code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$ }

$$\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} + \frac{\lambda}{m} \theta_1$$

gradient(n+1) = {code to compute $\frac{\partial}{\partial \theta_2} J(\theta)$ }

- - - - -

fminunc ...