Classification and Representation

classification

examples: spam / not spam ?

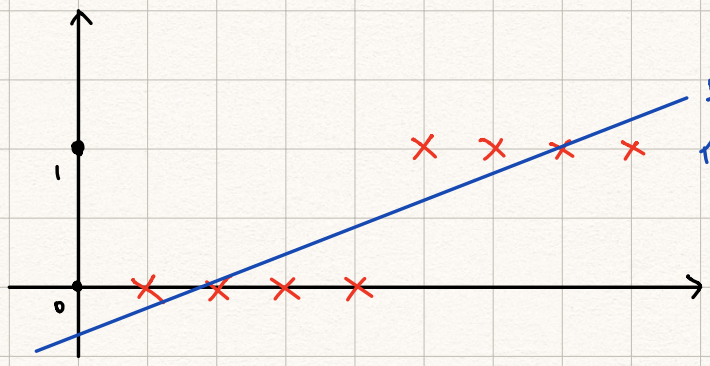
Malignant / Benign ?

0: Negative Class

1: Positive Class

type
- $y \in \{0, 1\}$ — one variable
- $y \in \{0, 1, 2, \cdots, N\}$ — multiple variable

①

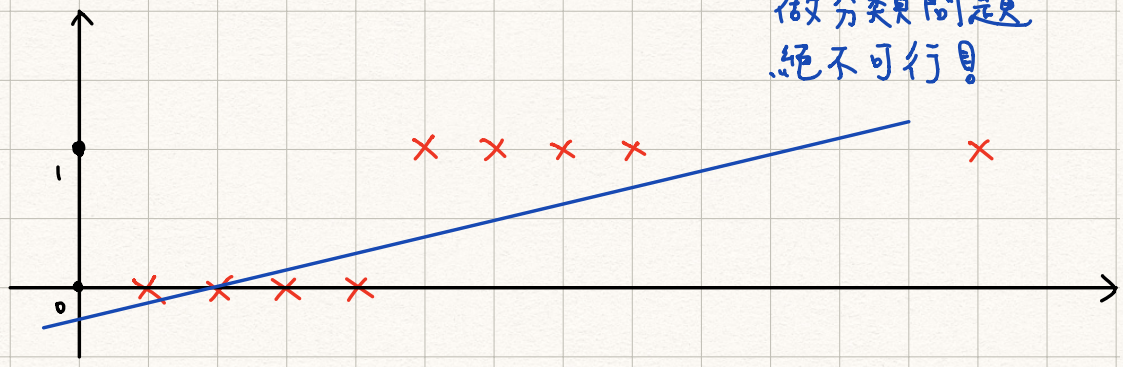


以 linear regression 做分類問題似乎可行

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

- $h_\theta(x) \geq 0.5$, predict $y = 1$
- $h_\theta(x) < 0.5$, predict $y = 0$

②



以 linear regression 做分類問題 絕不可行！

Use what algorithm to classification problem

classification : $y = 0$ 、 1

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

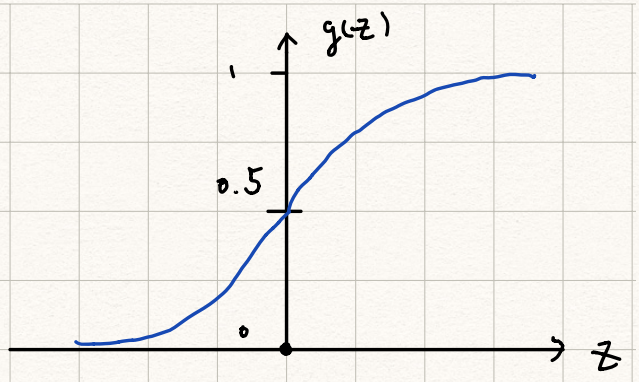
Hypothesis reprentation

want $0 \le h_\theta(x) \le 1$

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

→ sigmoid function

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



$$\hookrightarrow \quad h_\theta(x) = p(y=1 \mid x; \theta)$$

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

$$p(y=0 \mid x; \theta) + p(y=1 \mid x; \theta) = 1$$

Decision boundary

根據 sigmoid function 的圖形

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

$\theta^T x \geq 0$

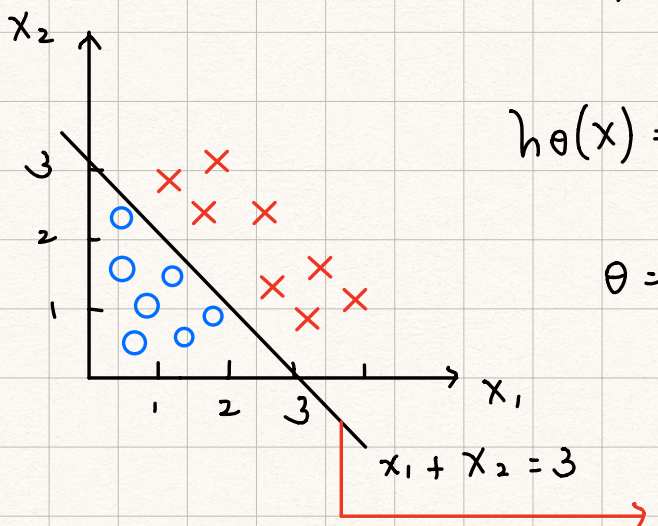
predict "$y=0$" if $h_\theta(x) < 0.5$

$\theta^T x < 0$

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

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

linear decision boundary



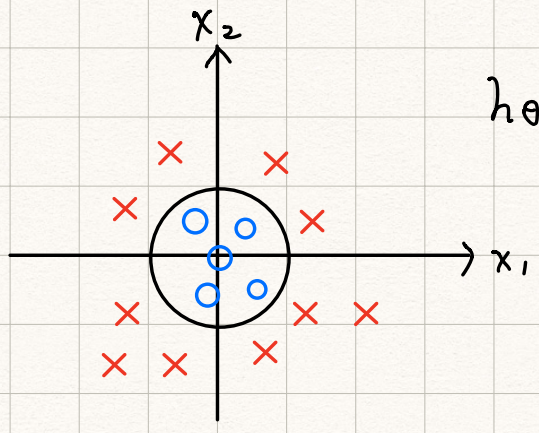
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_3)$$

$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

decision boundary

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

non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_3 \\ \theta_3 x_1^2 + \theta_4 x_2^2)$$

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

"$y=1$" 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 : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples :

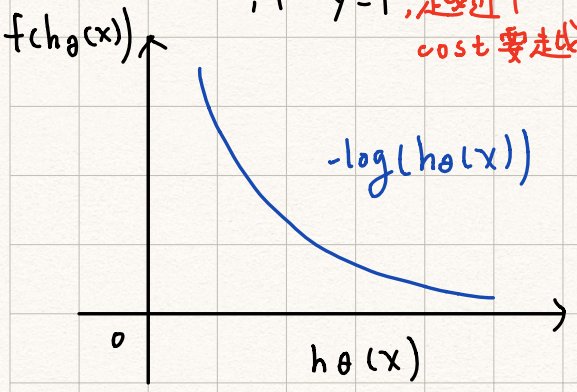
$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad x_0 = 1,\ y \in \{0, 1\}$$

$$x_1 \sim x_n \in \mathbb{R}$$

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

How to choose cost function?

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



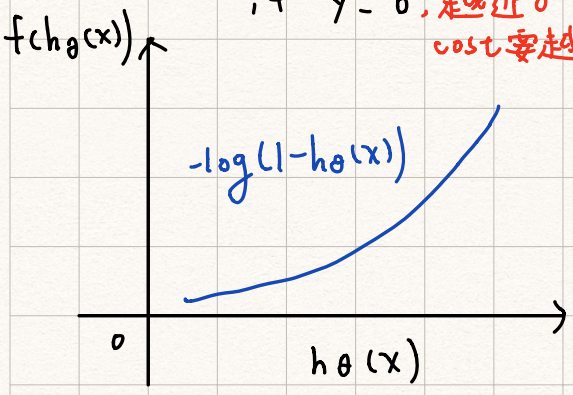
if $y = 1$, 越近1 cost要越小

Cost = 0 if $y = 1$, $h_\theta(x) = 1$

But as $h_\theta(x) \to 0$

$f(h_\theta(x)) \to \infty$

cost $\to \infty$



if $y=0$, 越近0 cost要越小

Cost = 0 if $y=0$, $h_\theta(x)=0$

But as $h_\theta(x) \to 1$, $f(h_\theta(x)) \to \infty$, $cost \to \infty$

Simplified cost function of logistic regression

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

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

simplify ⇒

$$Cost(h_\theta(x), y) = \boxed{-y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))}$$

so, Logistic cost function is

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

Gradient descent of logistic regression

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

(simultaneously update all θ_j)

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

(simultaneously update all θ_j)

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

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)

↓ 維度需 > 2

function [jVal, gradient] = costFunction(theta)

jVal = [code to compute J(θ)]

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

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

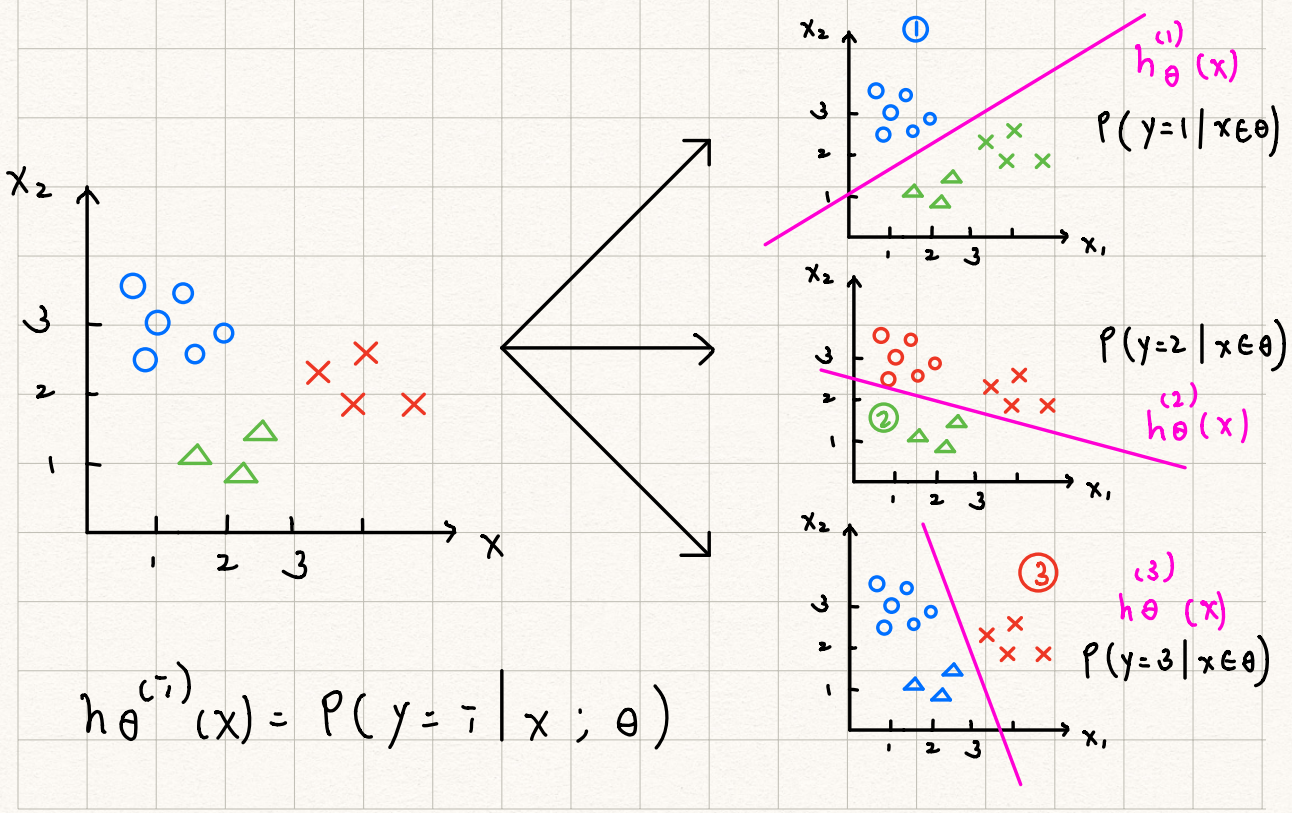
⋮

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

Multi-class classification

Weather: Sunny, Cloudy, Rain, Snow

y =	1	2	3	4
or	0	1	2	3



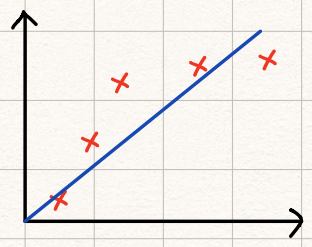
$h_\theta^{(i)}(x) = P(y=i \mid x;\theta)$

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

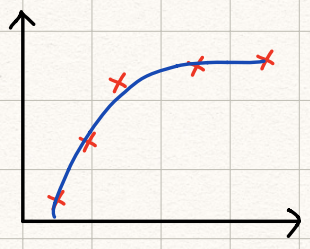
$$\max_i h_\theta^{(i)}(x)$$

Solving the problem of Overfitting

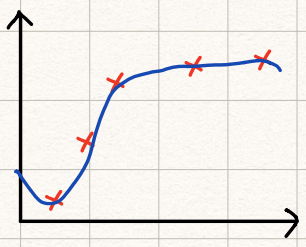
The problem of overfitting



"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証集
→ 驗証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:

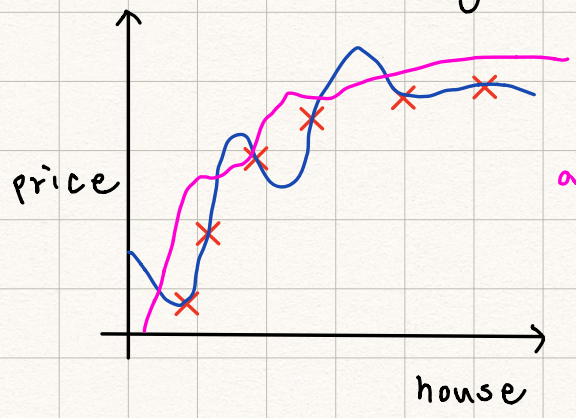
- Features: $x_1, x_2, \dots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

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

↓ regulization parameter

$$\min_\theta J(\theta)$$

linear regression



after using regulization

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

$\lambda \uparrow$, 懲罰 θ_j 的力度越大
$\theta_1 \sim \theta_j$ 越近 zero.

a way to choose regulization value automatically.

Regularized linear regression

Gradient descent

Repeat {

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

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

}

$\rightarrow \frac{\partial}{\partial \theta_j} J(\theta)$ ↓ regularized term

$j = 1, 2, 3, \ldots, n$

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

Normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}_{m \times (n+1)} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

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

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & & & \vdots \\ \vdots & & \ddots & & \\ 0 & \cdots & & 1 & 0 \end{bmatrix} \right)^{-1} X^T y$$

$(n+1)(n+1)$

Non-invertibility (optional/advanced)

Suppose $m \leq n$ $\quad \theta = (X^T X)^{-1} X^T y$

(example : feature)

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

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

Regularized logistic regression

Gradient descent

Repeat {

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

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

$\}$ $\rightarrow \frac{\partial}{\partial \theta_j} J(\theta)$ $\quad j = \theta_1, \theta_2 \dots \theta_n$

↓ regularized term

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

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

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 . . .