MORE ON
LINEAR REGRESSION

Recap: Linear Regression

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$
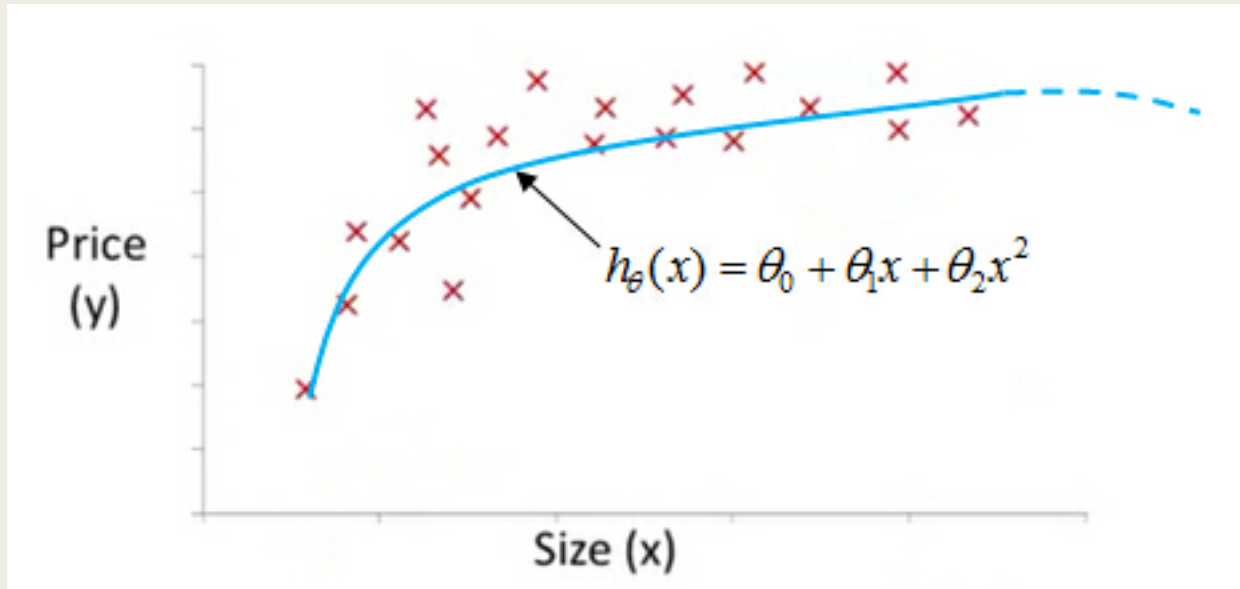
Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}} \ J(\theta_0, \theta_1)$

Polynomial Regression

- A polynomial function may fit the data better than a linear function

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Set

- $x_1 = x$
- $x_2 = x^2$

Model representation: $h(x) = \boldsymbol{\theta_0} + \boldsymbol{\theta_1} x_1 + \boldsymbol{\theta_1} x_2$

Polynomial Regression

Model representation: $h(x) = \theta_0 + \theta_1 x_1 + \theta_1 x_2$

Cost Function

$$J(\theta_0, \theta_1, \theta_2) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right)^2$$

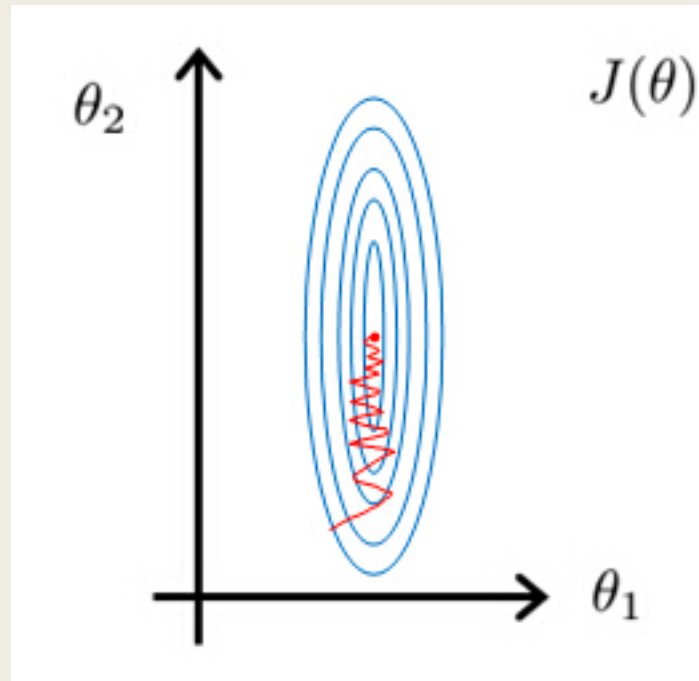## Gradient Descent (Parameters Learning)

repeat until convergence: {

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

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

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

...

}

Feature Scaling

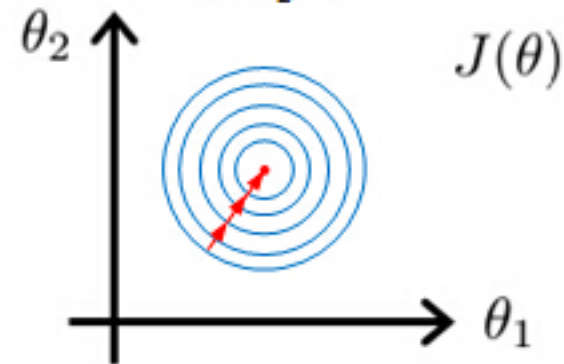Idea: make sure features are on a similar scale

x_1 = size (0-2000 feet^2)
x_2 = number of bedrooms (1-5)

$$x_1 = \frac{size(feet^2)}{2000}$$

$$x_2 = \frac{number\ of\ bedrooms}{5}$$

$$0 \leq x_1 \leq 1$$

$$0 \leq x_2 \leq 1$$

Feature Scaling

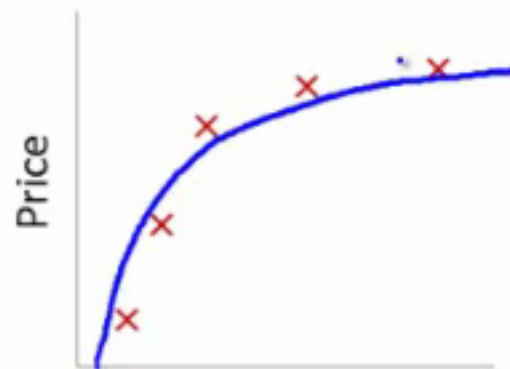Mean Normalization

$$x_i \leftarrow \frac{x_i - mean}{std}$$

So all features would have an average of zero and similar range
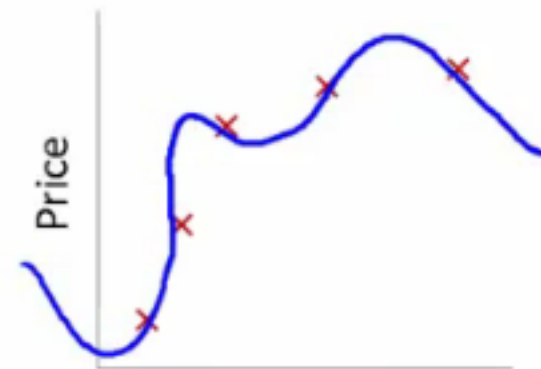
The problem of overfitting

High bias (underfit): $\theta_0 + \theta_1 x$

"Just right": $\theta_0 + \theta_1 x + \theta_2 x^2$

High variance (overfit): $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Addressing overfitting

1) <u>Reduce number of features</u> x automation

- Manually select which features to keep
- Model selection algorithms are discussed later (good for reducing number of features)
- But, in reducing the number of features we lose some information

2) <u>Regularization</u> this method is better
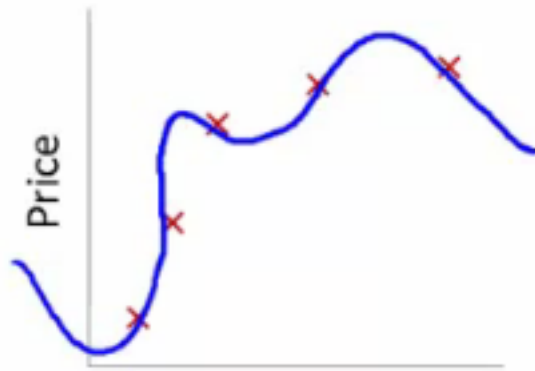
- Keep all features, but reduce magnitude of parameters $\boldsymbol{\theta}$
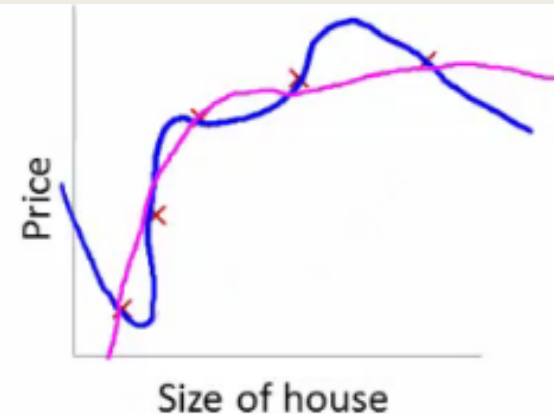- Works well when we have a lot of features, each of which contributes a bit to predicting y

Regularization

Intuition: making ==theta so small== that is almost ==equivalent to zero==

Penalize and make some of the θ parameters really ==small== e.g. here θ_3 and θ_4

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \boxed{\theta_3 x^3 + \theta_4 x^4}$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

minimize this fn

Ridge regression (L2 Regularization)

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

- Small values for parameters corresponds to a simpler hypothesis (you effectively get rid of some of the terms)

- A simpler hypothesis is less prone to overfitting

เลือก lambda ให้ถูกต้อง กราฟจะได้ไม่พัง
- λ is the **regularization parameter**
  Controls a trade off between our two goals
  1. Want to fit the training set well
  2. Want to keep parameters small

This is what we call "Hyper-parameters"

Lasso regression (L1 Regularization)

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(x^i) - y^i\right)^2 + \lambda \sum_{j=1}^{n} |\boldsymbol{\theta}_j|$$
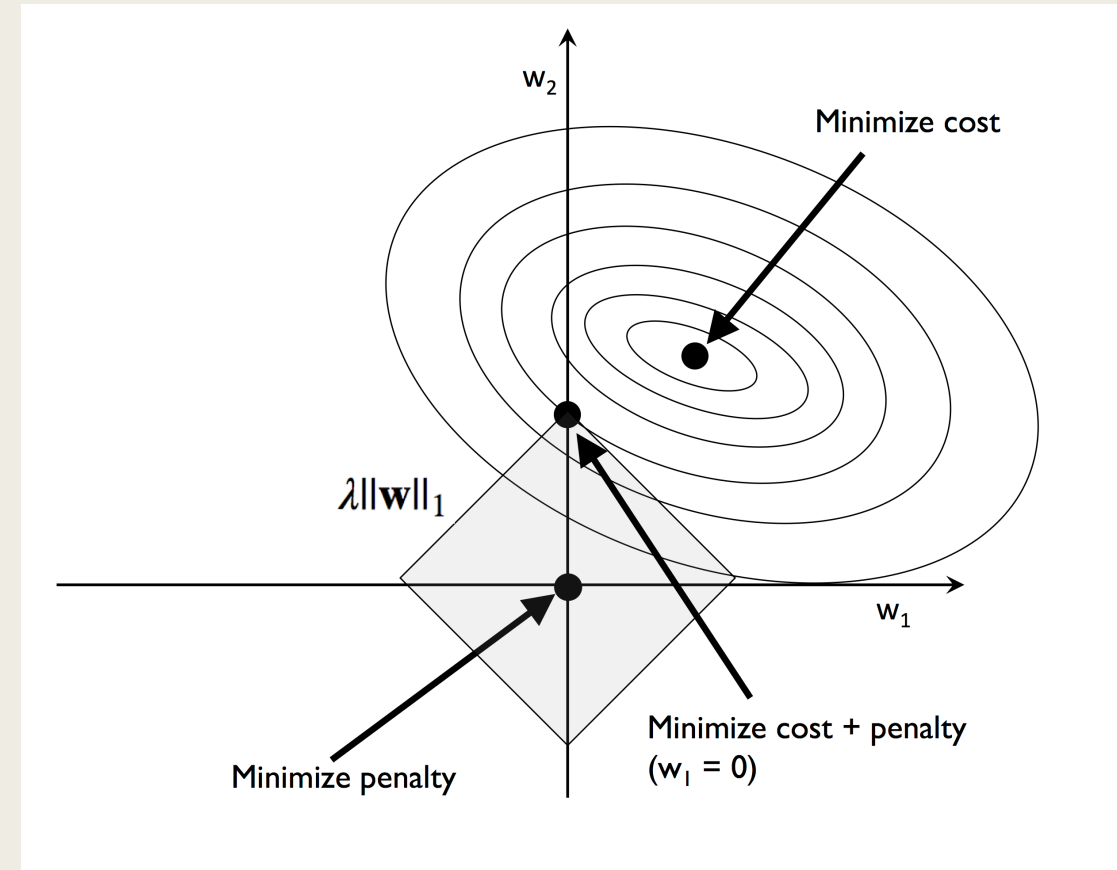
- Very similar to L2 regularization.

- However, instead of a quadratic penalty term as in L2, we penalize the model by the absolute weight coefficients

- This is a subtle, but important change. Some of the coefficients may be shrunk exactly to zero.
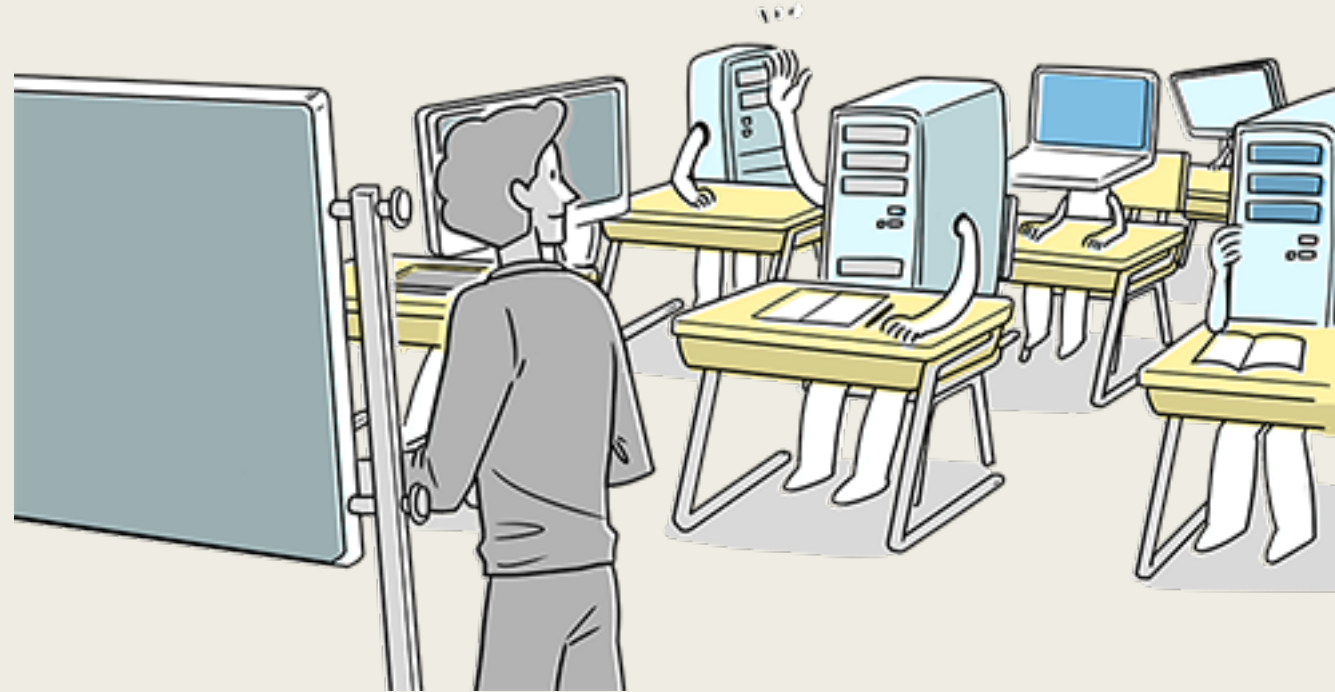
- Perform both "Feature selection" and "Regularization"

Ridge (L2) vs Lasso (L1)
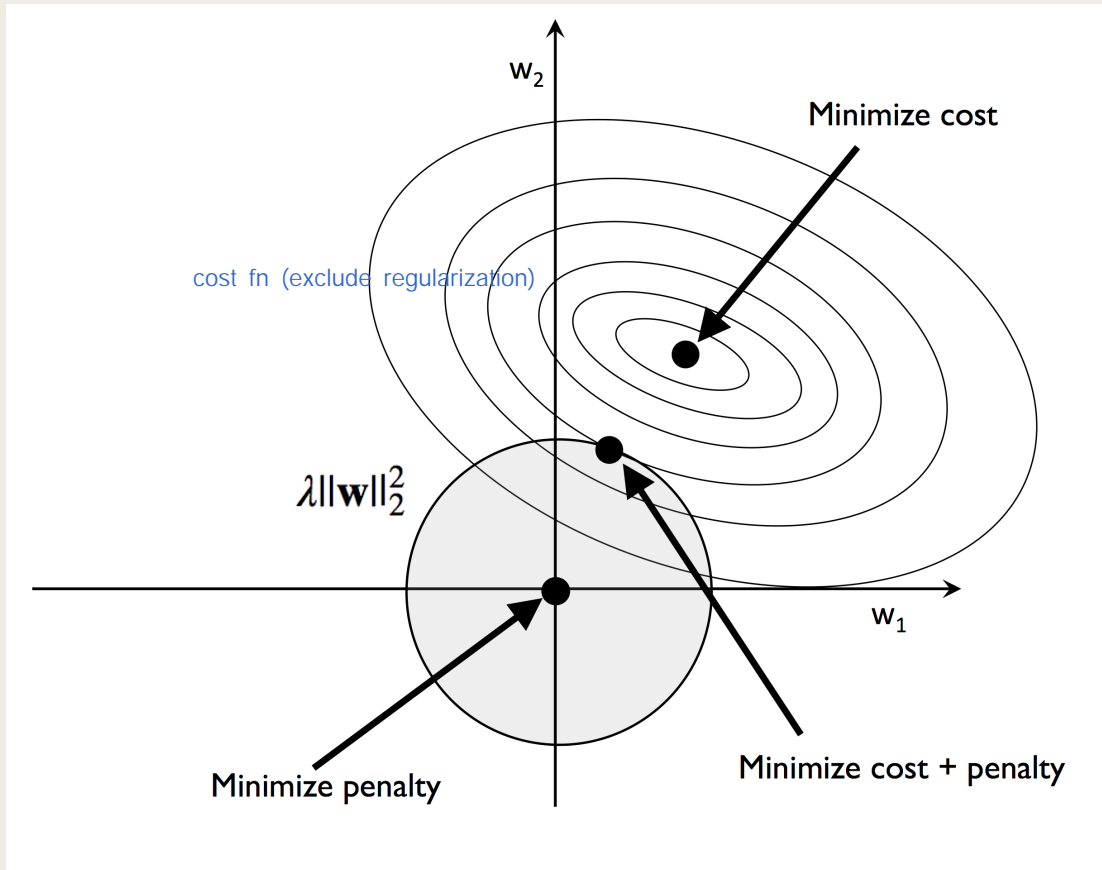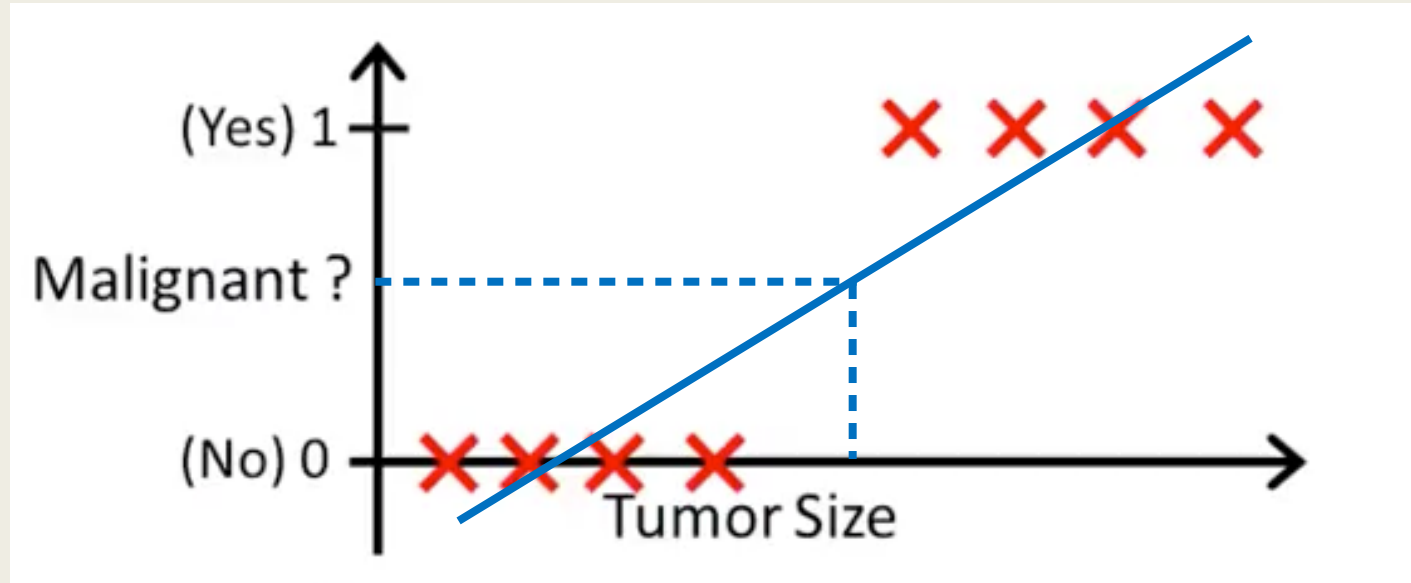
CLASSIFICATION

Classification

- Email: Spam/ Not Spam ?

- Online Transactions: Fraudulent (Yes/No) ?

- Tumor: Malignant/ Benign?

$$y \in \{0, 1\}$$

0: "Negative Class" (e.g., benign tumor)
1: "Positive Class" (e.g., malignant tumor)
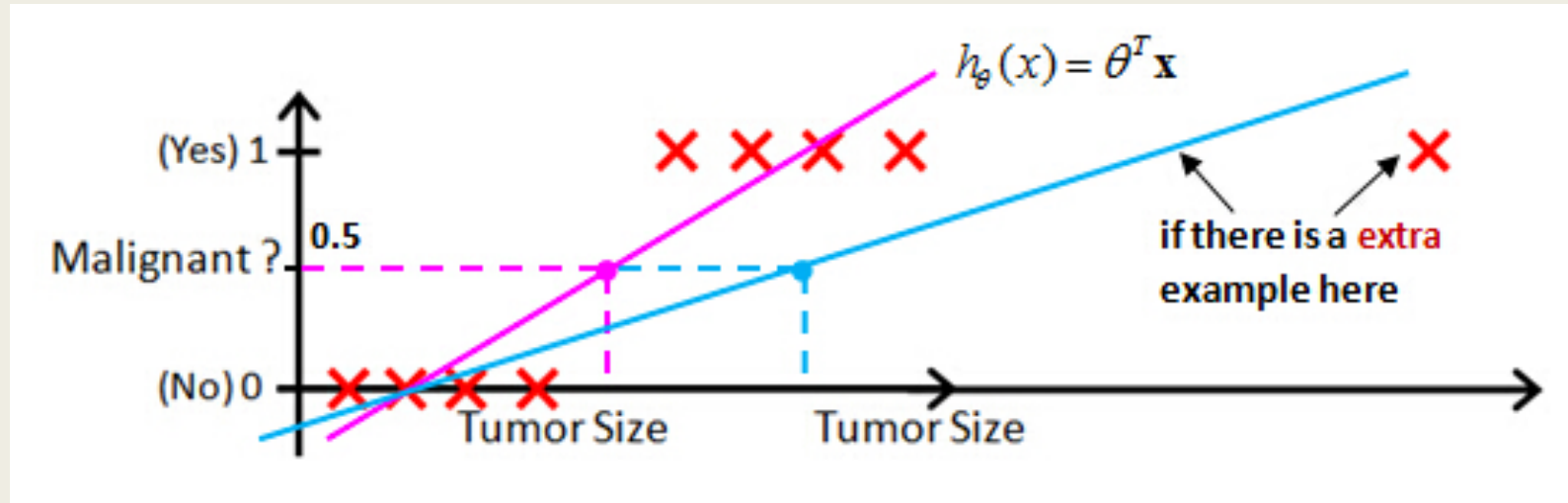
How to develop a ==classification== algorithm?

Can we use linear regression ? $h(x) = \boldsymbol{\theta_0} + \boldsymbol{\theta_1} x_1$

If $h(x) \geq 0.5$, predict "y = 1"
If $h(x) < 0.5$, predict "y = 0"

How to ==develop== a classification ==algorithm==?

Linear regression to a classification problem often isn't a great idea

Classification: y = 1 or 0

But for linear regression: $h(x)$ can be > 1 or < 0

We want $0 \leq h(x) \leq 1$

Logistic Regression Model

We want: $0 \leq h(x) \leq 1$
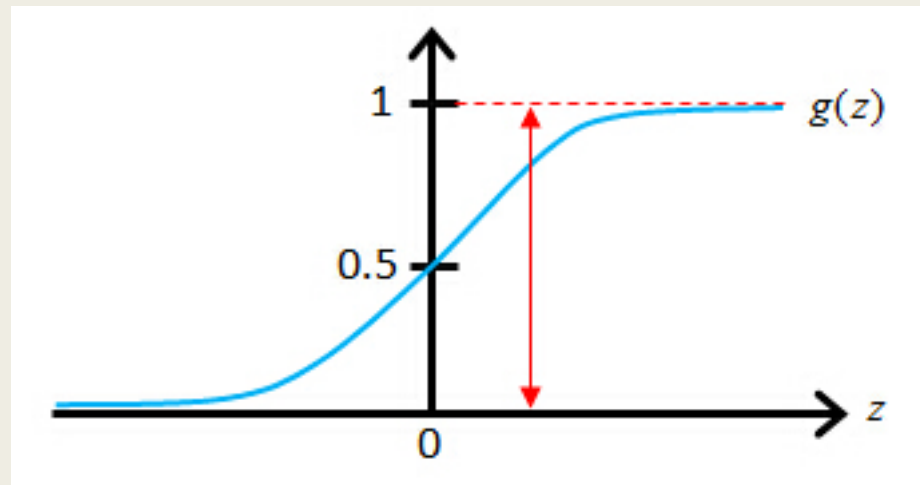
$h(x) = g(\boldsymbol{\theta_0} + \boldsymbol{\theta_1} x_1)$

$g(z) = \dfrac{1}{1 + e^{-z}}$

$\Longrightarrow$

$h(x) = \dfrac{1}{1 + e^{-(\boldsymbol{\theta_0} + \boldsymbol{\theta_1} x_1)}}$ < 1&& != 0

The sigmoid/logistic function g(z)

Interpretation of Hypothesis Output

$$\mathbf{h}(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 \mathbf{X}_1)}}$$
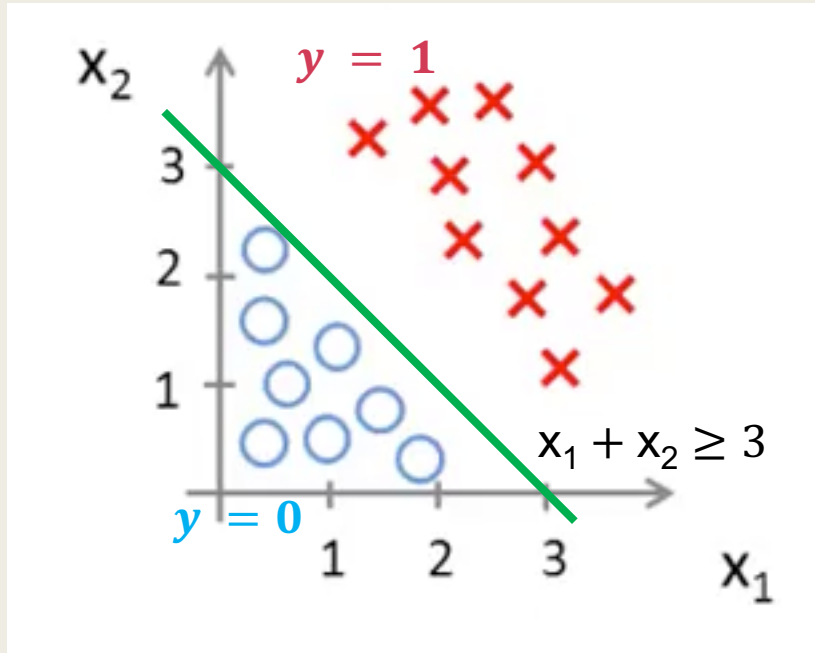
$h(x)$ = estimated probability that y = 1 on input x

$h(x) = P(y = 1 | x)$

e.g. P(email is spam | number of word "discount" in the email)

Decision Boundary <mark>always be linear</mark>
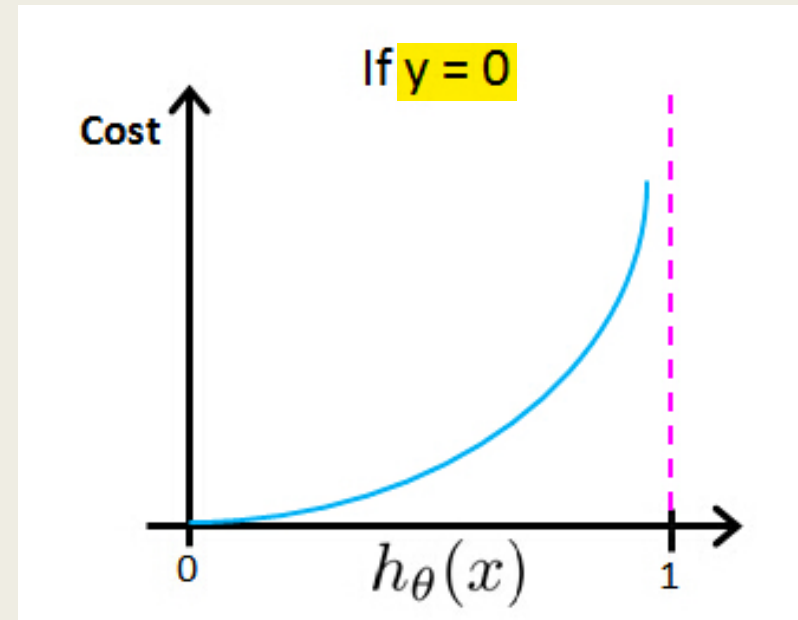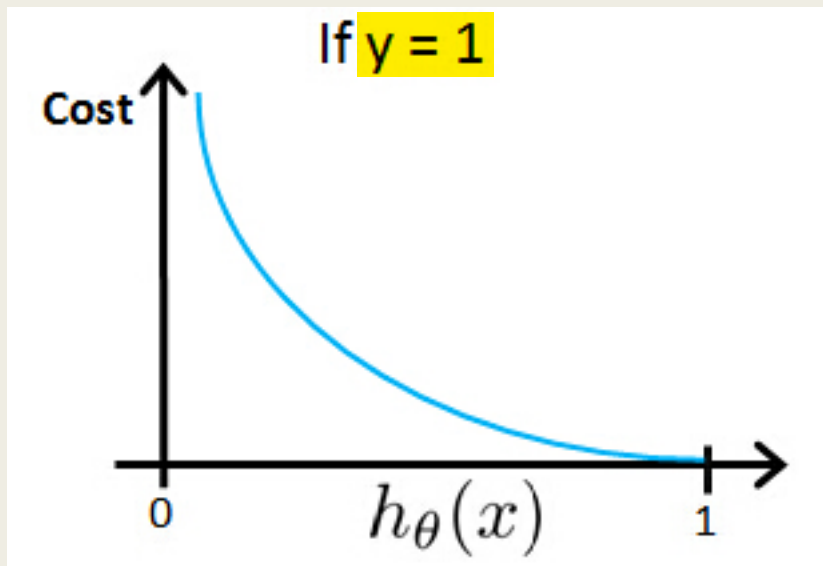

$$h(x) = \frac{1}{1 + e^{-(-3 + 1X_1 + 1X_2)}}$$

Predict $y = 1$ if $h(x) \geq 0.5$

$equivilent\ to\ -3 + x_1 + x_2 \geq 0$

Logistic Regression Cost function

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

น้อย

มาก

If y = 1

If y = 0

Parameter Learning with Gradient Descent

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

Want $\min_\theta J(\theta)$:

Repeat {

Algorithm looks <mark>identical</mark> to <mark>linear</mark> regression!!

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

(simultaneously update all θ_j)

}