

# IT5005 Artificial Intelligence

## Introduction to Learning

Sirigina Rajendra Prasad

Slide Credit: Prof. Ben Leong



Picture → **Function** → Word/number

Picture → **Function** → Chihuahua  
/muffin

Email → **Function** → Spam  
/not spam

Black  
& white  
photo → **Function** → Coloured  
photo

↑  
How do we write this function?

# Types of Feedback

- Supervised
  - Correct answer given for each example
  - e.g. image of “A” and its unicode 0041
- Unsupervised
  - No answers given
  - e.g. are there patterns in the data?

# Types of Feedback

- Weakly supervised
  - Correct answer given, but not precise
  - e.g. This slide contains a face (but not exact location)
- Reinforcement
  - Occasional rewards given
  - e.g. robot navigating a maze

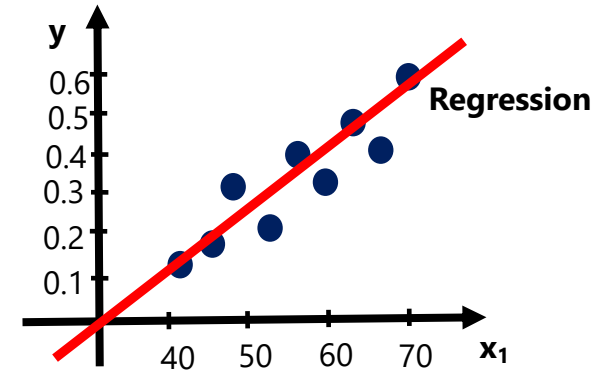
# Supervised Learning

Given a data set, we know what we are looking for and we have some idea of a relationship between input and output

# Supervised Learning

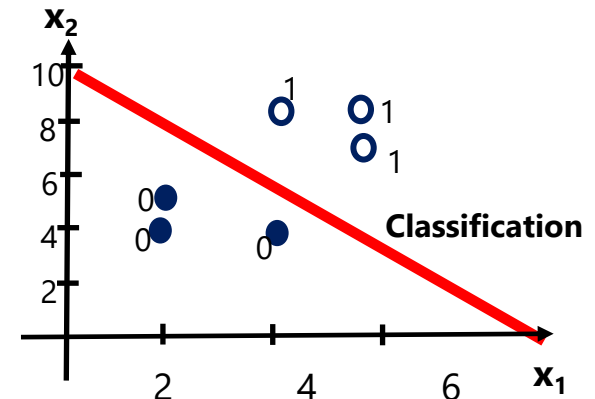
## 1. Regression

- Predict results within a continuous output, map input variables to some continuous function.

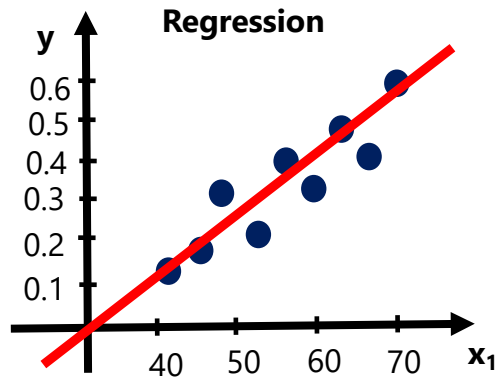


## 2. Classification

- Predict results in a discrete output, map input variables into discrete categories.



# Regression vs Classification



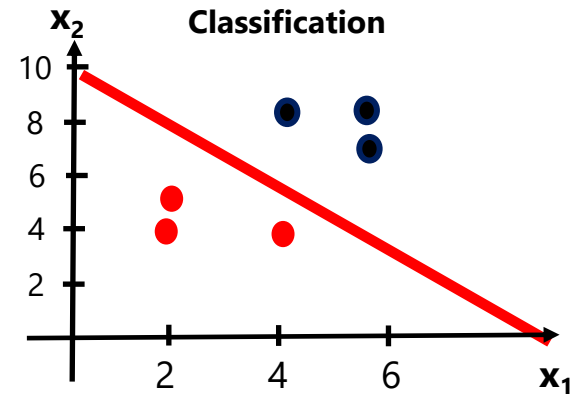
Data

$x_1$	$y$
41	0.13
45	0.18
50	0.3
55	0.2
60	0.3
59	0.35

$$y = h_r(x_1)$$

Regression function

*h: hypothesis*



Data

$x_1$	$x_2$	$y$
2	4	1
2	6	1
4	4	1
4	9	0
6	7	0
6	9	0

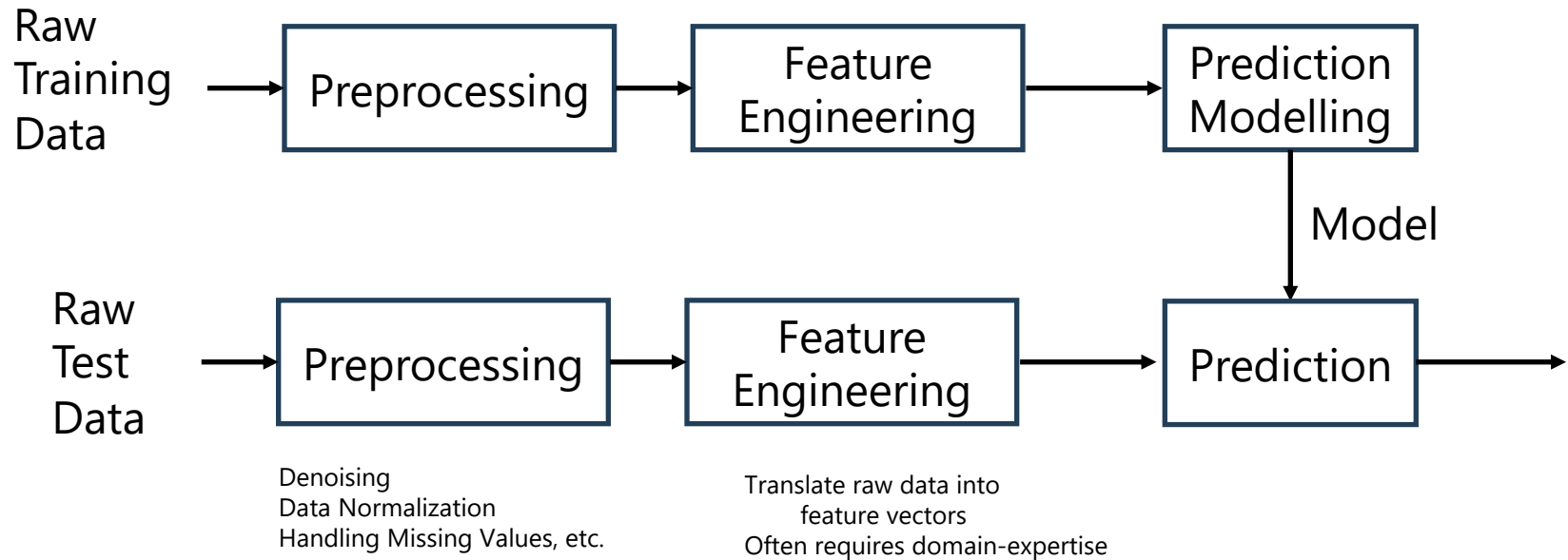
$$y = h_c(x_1, x_2)$$

Classification function

# Typical ML Pipeline

S. No	b	$x_1$	...	$x_n$	$y$
1	1	$x_1^{(1)}$	...	$x_n^{(1)}$	$y^{(1)}$
2	1	$x_1^{(2)}$	...	$x_n^{(2)}$	$y^{(2)}$
$\vdots$					
$m$	1	$x_1^{(m)}$	...	$x_n^{(m)}$	$y^{(m)}$

feature vector  $\mathbf{x}^{(2)}$



# Machine Learning Pipeline

1. Data collection
2. Data extraction (Feature engineering)
3. Data understanding (with Visualization)
4. Data pre-processing
5. Model choice / design
6. Model training
7. Model validation (Evaluation)
8. Model understanding (Visualization / Explainability)
9. Model deployment

# Machine Learning Models

- Linear Models
  - Linear Regression
  - Perceptron Learning Algorithm
  - Logistic Regression
  - Support Vector Machine, etc.
- Nonlinear Models
  - Multilayer Perceptron (MLP)
  - Convolutional Neural Networks (CNN)
  - Sequence Models
    - Recurrent Neural Networks (RNN)
    - LSTMs, GRUs
    - Transformers

# Linear Models



## Objective:

Predict the label  $\mathbf{y}^{(j)}$  using the feature vector  $\mathbf{x}^{(j)} = \begin{bmatrix} 1 \\ x_1^{(j)} \\ \vdots \\ x_n^{(j)} \end{bmatrix}$  input for bias term

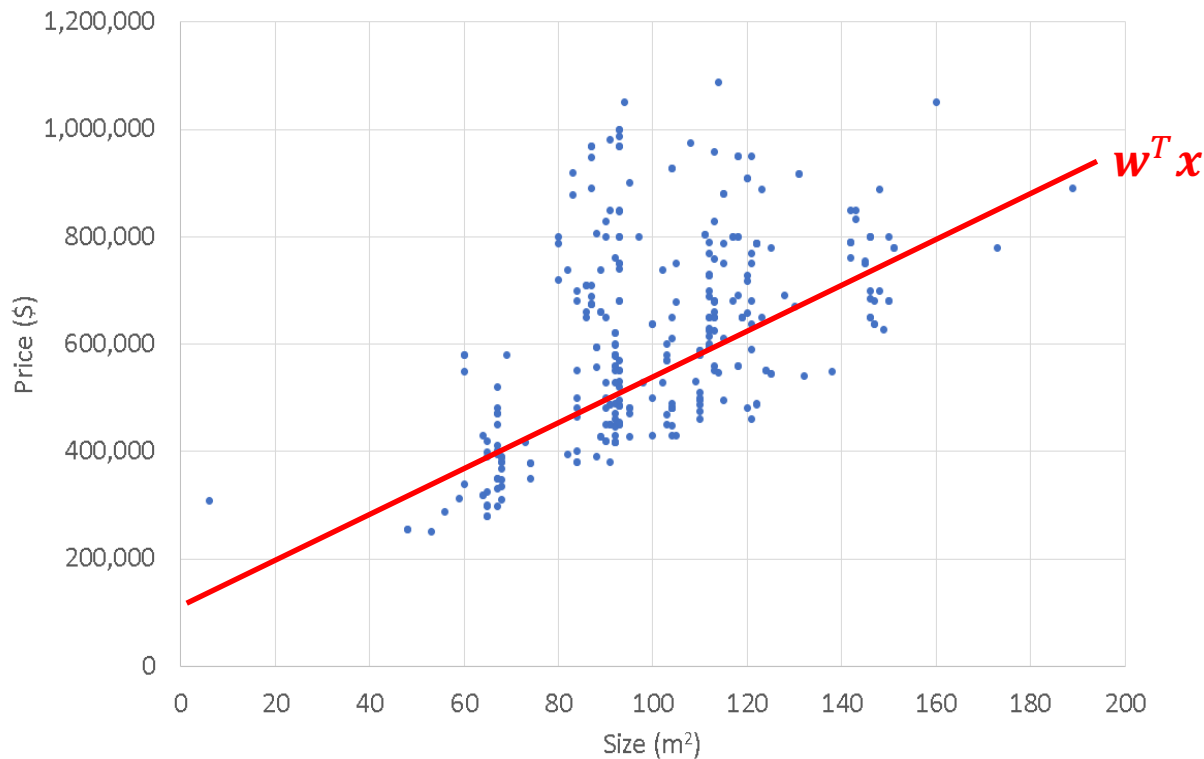
**Prediction**  $\hat{y}_j$  :

$\hat{y}_j = g(\mathbf{w}^T \mathbf{x}^{(j)})$  Linear model

Where  $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$  weight for bias term

# Linear Regression

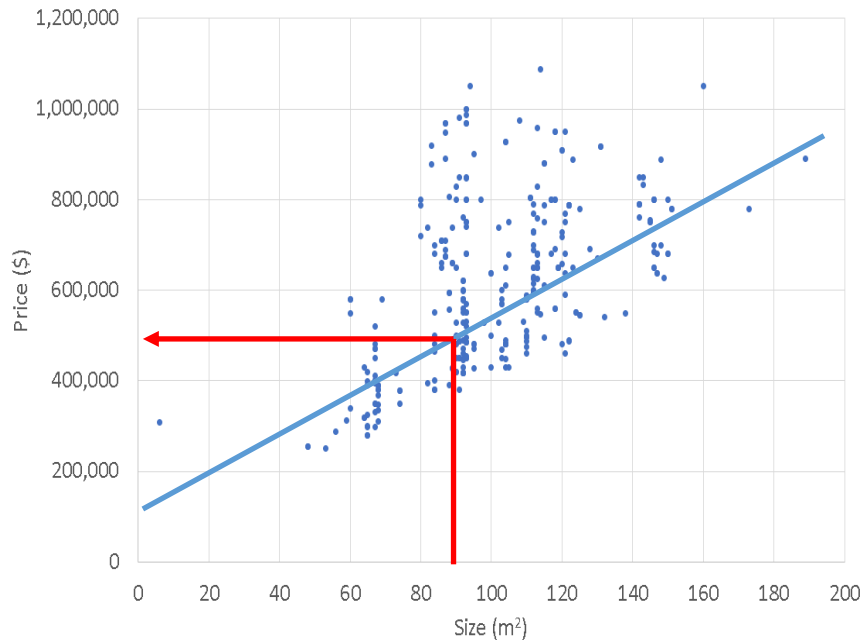
# HDB Prices



$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \end{bmatrix}$$

$$\mathbf{w}^T \mathbf{x} = w_0 + w_1 x_1$$

# HDB Prices



Regression  
problem

Given size:  
predict price

90 m<sup>2</sup> → ~\$500K

# HDB Prices

Training set of  
HDB prices from  
SRX

Size (m <sup>2</sup> ) ( $x$ )	Price (\$) ( $y$ )
113	560,000
102	739,000
100	430,000
84	698,000
112	688,888
68	390,000
121	768,000
...	...

Notation:

- $m$  = number of training examples
- $x$  = “input” variables/features
- $y$  = “output” variables/ “target” variables

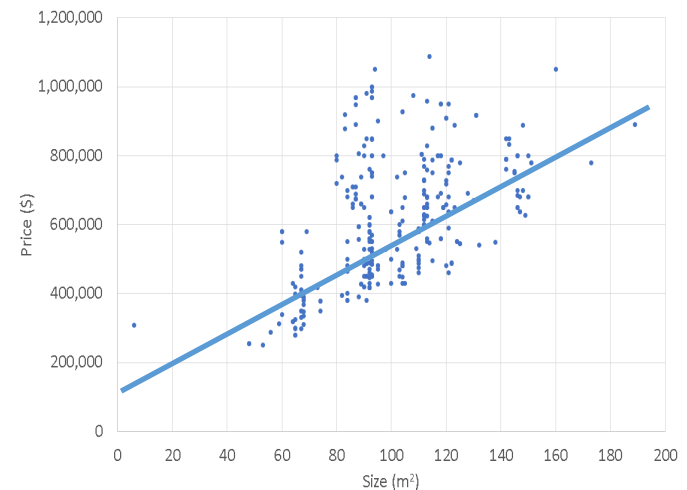
# Linear Regression

$$h_w(x): \mathbf{w}^T \mathbf{x}$$

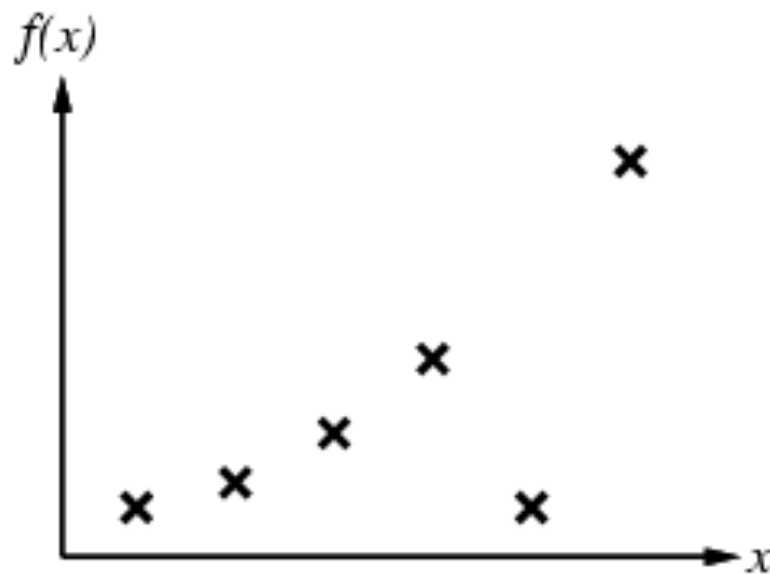
where  $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$  and  $\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \end{bmatrix}$

How do we determine  $\mathbf{w}$  ?

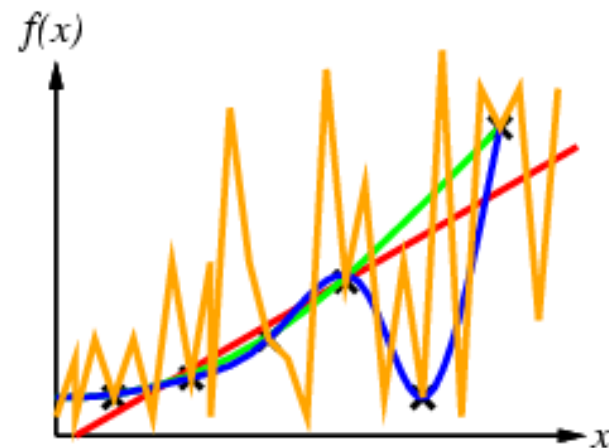
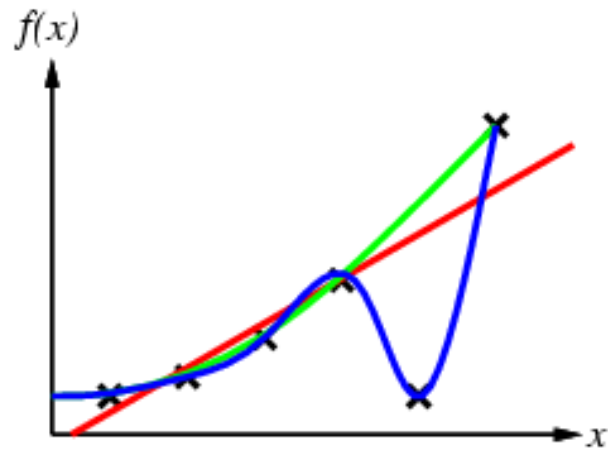
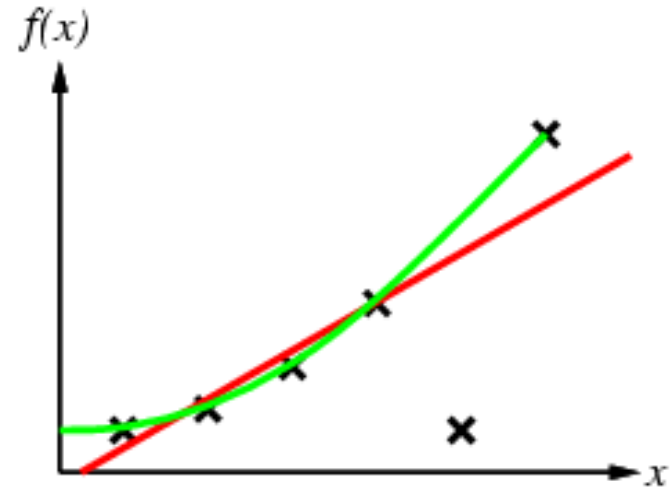
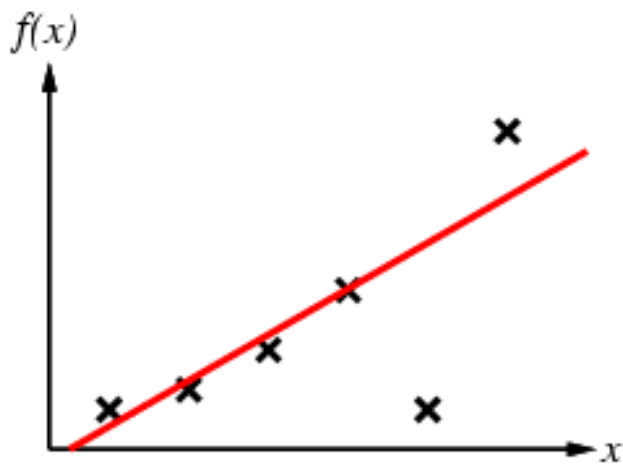
We want  $h_w$  so that “fits data well”



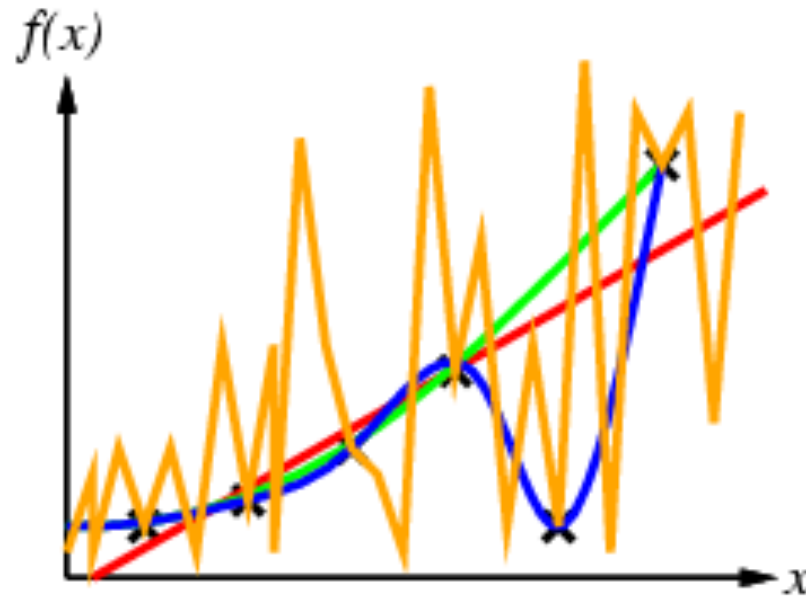
# What do we mean by “fits the data well”?



# Curve Fitting



# Curve Fitting



**Occam's razor:** prefer the simplest hypothesis consistent with data

Key Idea: Choose  $w$  so that  
 $h_w(x)$  is “close” to the  
training examples

→ Cost function

# How to Select Cost Functions

## Convexity

- Global minimum is local minimum

## Differentiable

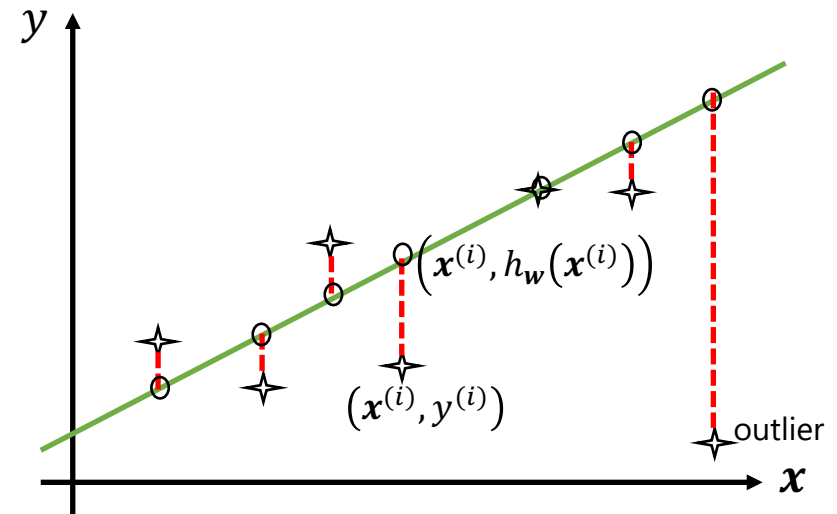
- Enables gradient descent

## Robustness

- To outliers

# Cost Function: Mean Squared Error (MSE)

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2m} \sum_{i=1}^m J_i(\mathbf{w}) \\ &= \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)})^2 \end{aligned}$$



Also known as  $L_2$ -Loss:  $J(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(m)})^T \end{bmatrix}$$

# Gradient Descent: Recap

$$\mathbf{w} = \mathbf{w} - \alpha \nabla J(\mathbf{w})$$

where,

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$\nabla J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1} \end{bmatrix}$$

$J(\mathbf{w})$  is a scalar  
 $\mathbf{w}$  is a vector

# Gradient Descent for Linear Regression

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)})^2$$

$$\Rightarrow \frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{1}{m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)})$$

$$\Rightarrow \frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{1}{m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)}) \cdot x^{(i)}$$

Chain rule:

$$h(x) = f(g(x))$$

$$h'(x) = f'(g(x))g'(x)$$

Refer 'Linear Regression Gradient Descent.ipynb' for a demo

# Variants of Gradient Descent

- (Batch) Gradient Descent
  - Consider all training examples

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m J_i(\mathbf{w})$$

- Stochastic Gradient Descent
  - Consider one randomly selected data point at a time
  - Cheaper (faster)
  - More randomness – might escape local minima
- Mini-Batch Gradient Descent

# HDB Prices (Single Feature)

Training set of  
HDB prices from  
SRX

$x$ Size (m <sup>2</sup> )	$y$ Price (\$)
113	560,000
102	739,000
100	430,000
84	698,000
112	688,888
68	390,000
121	768,000
...	...

Notation:

- $m$  = number of training examples
- $x$  = "input" variables/features
- $y$  = "output" variables/ "target" variables

# HDB Prices ( $n = 4$ )

$x_1$ Year	$x_2$ Size (m <sup>2</sup> )	$x_3$ #bedrooms	$x_4$ #bathrooms	$y$ Price (\$)
2016	113	4	2	560,000
1998	102	3	2	739,000
1997	100	3	0	430,000
2014	84	3	2	698,000
2016	112	3	0	688,888
1979	68	2	2	390,000
1969	53	2	1	250,000
1986	122	3	2	788,000
1985	150	3	3	680,000
2009	90	3	2	828,000

Notation:

- $n$  = number of features
- $x^{(i)}$  = input features of the  $i$ th training example
- $x_j^{(i)}$  = value of feature  $j$  in  $i$ th training example

Hypothesis ( $n = 4$ ):

$$h_{\mathbf{w}}(\mathbf{x}): \mathbf{w}^T \mathbf{x}, \text{ where } \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \text{ and } \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Cost Function:

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)})^2$$

Gradient Descent:

$$\mathbf{w} := \mathbf{w} - \alpha \nabla J(\mathbf{w})$$

$$\nabla J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial w_4} \end{bmatrix}$$

$$h_{\mathbf{w}}(\mathbf{x}): w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

## Previously ( $n=1$ )

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m (h_w(\mathbf{x}^{(i)}) - y^{(i)})^2$$

Repeat {

$$w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_w(\mathbf{x}^{(i)}) - y^{(i)})$$

$$w_1 := w_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_w(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

}

## Generalized ( $n > 1$ )

$$J(w) = \frac{1}{2m} \sum_{i=1}^m (h_w(\mathbf{x}^{(i)}) - y^{(i)})^2$$

Repeat {

$$w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_w(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$w_1 := w_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_w(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$\vdots$

$$w_n := w_n - \alpha \frac{1}{m} \sum_{i=1}^m (h_w(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_n^{(i)}$$

}

# Feature Scaling

Gradient Descent doesn't work very well if the features have significantly different scales

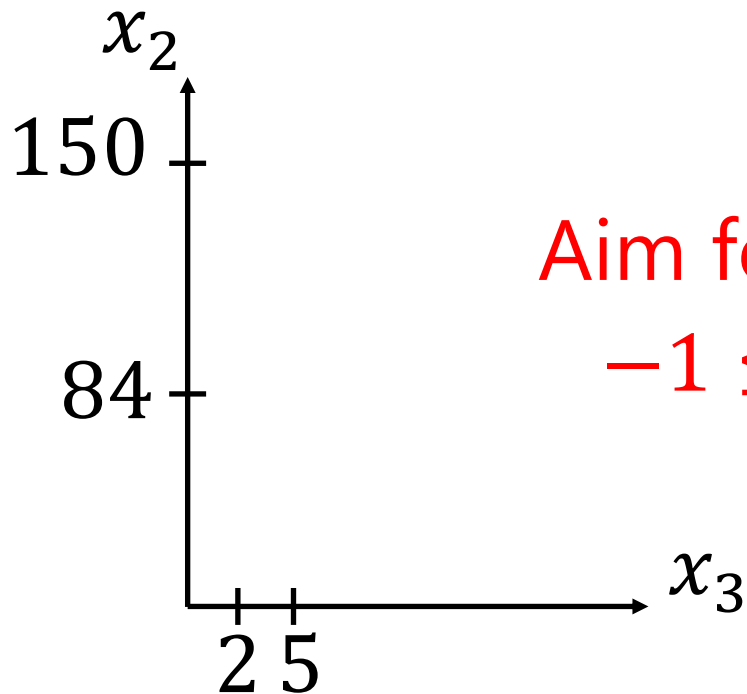
Key Idea: Scale features so that they vary roughly the same scales

$$84 \leq x_2 \leq 150$$

$$2 \leq x_3 \leq 5$$

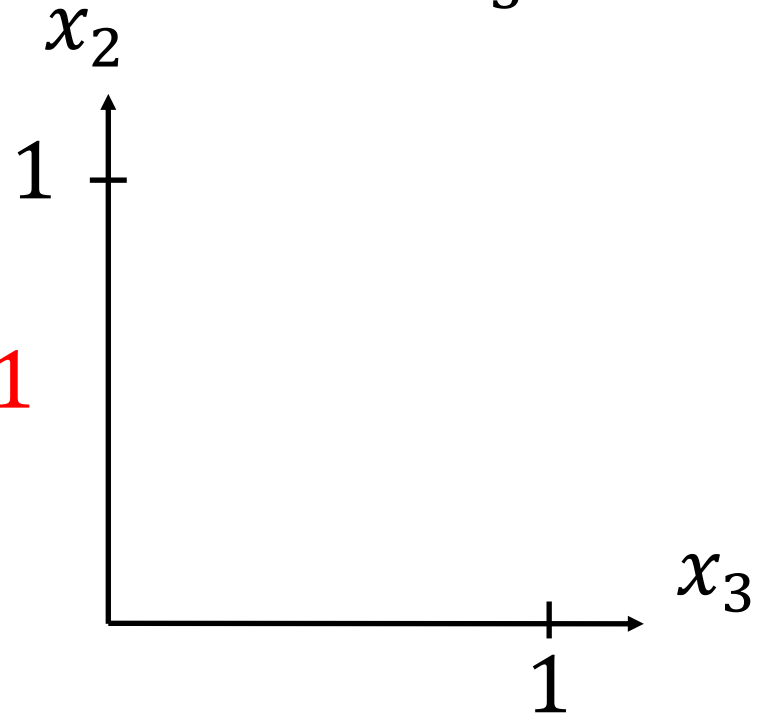
$$x_2 = \frac{\text{size} - 84}{150 - 84}$$

$$x_3 = \frac{\text{rooms} - 2}{5 - 2}$$



Aim for:

$$-1 \leq x_i \leq 1$$



# Mean normalization

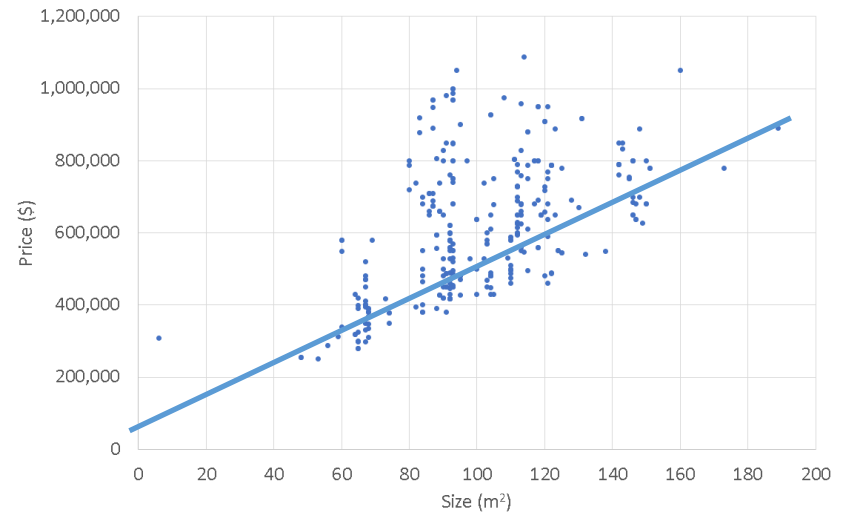
Replace  $x_i$  with  $x_i - \mu_i$  so that all features have approximately zero mean

$$x_i \leftarrow \frac{x_i - \mu_i}{\sigma_i}$$

std dev

# Polynomial Regression

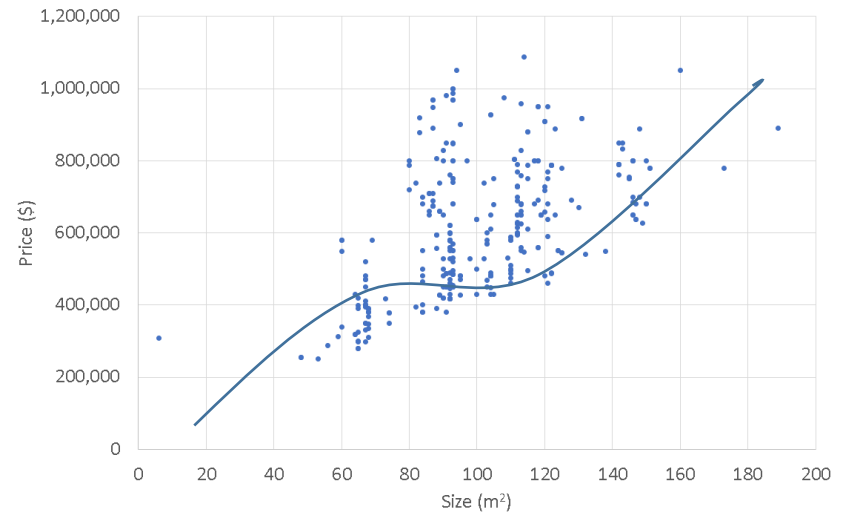
Size (m <sup>2</sup> ) ( $x$ )	Price (\$) ( $y$ )
113	560,000
102	739,000
100	430,000
84	698,000
112	688,888
68	390,000
121	768,000
...	...



No good reason for the relationship to be linear

# Polynomial Regression

Size (m <sup>2</sup> ) ( $x$ )	Price (\$) ( $y$ )
113	560,000
102	739,000
100	430,000
84	698,000
112	688,888
68	390,000
121	768,000
...	...



Why not:

$$h_w(\mathbf{x}): w_0 + w_1x + w_2x^2 + w_3x^3$$

# Polynomial Regression

$$\begin{array}{ll} x_1 \leftarrow x & 84 \leq x \leq 150 \\ x_2 \leftarrow x^2 & 7,000 \leq x_2 \leq 22,500 \\ x_3 \leftarrow x^3 & 560\text{K} \leq x_3 \leq 3.4 \times 10^6 \end{array}$$

Need to do feature scaling!

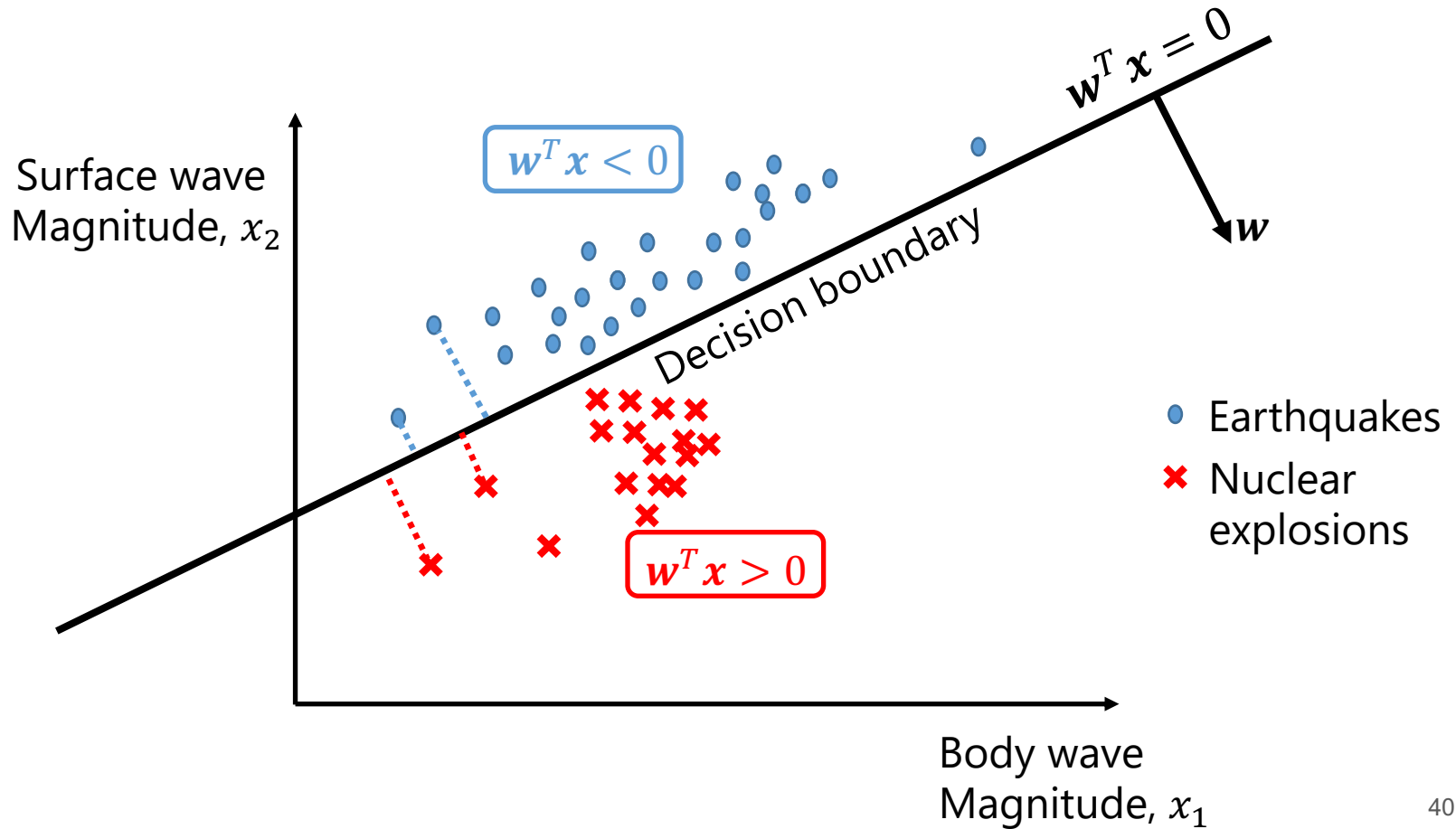
Need not be  $x^i$ , can be  $\sqrt{x}$

i.e.  $h_w(\mathbf{x}): w_0 + w_1 x + w_2 \sqrt{x}$

# Logistic Regression

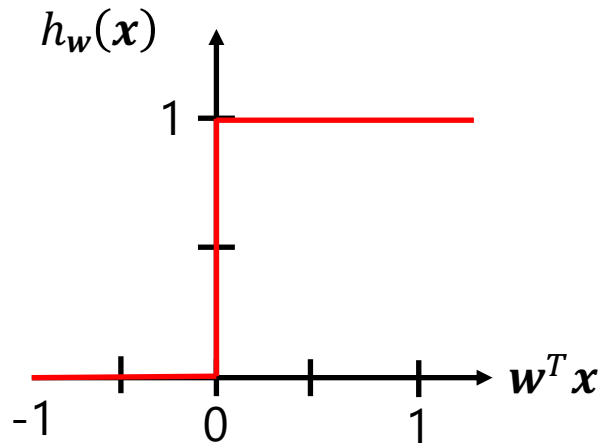
Classification with Continuous Inputs

# Classification with Continuous Inputs



$$\mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}$$

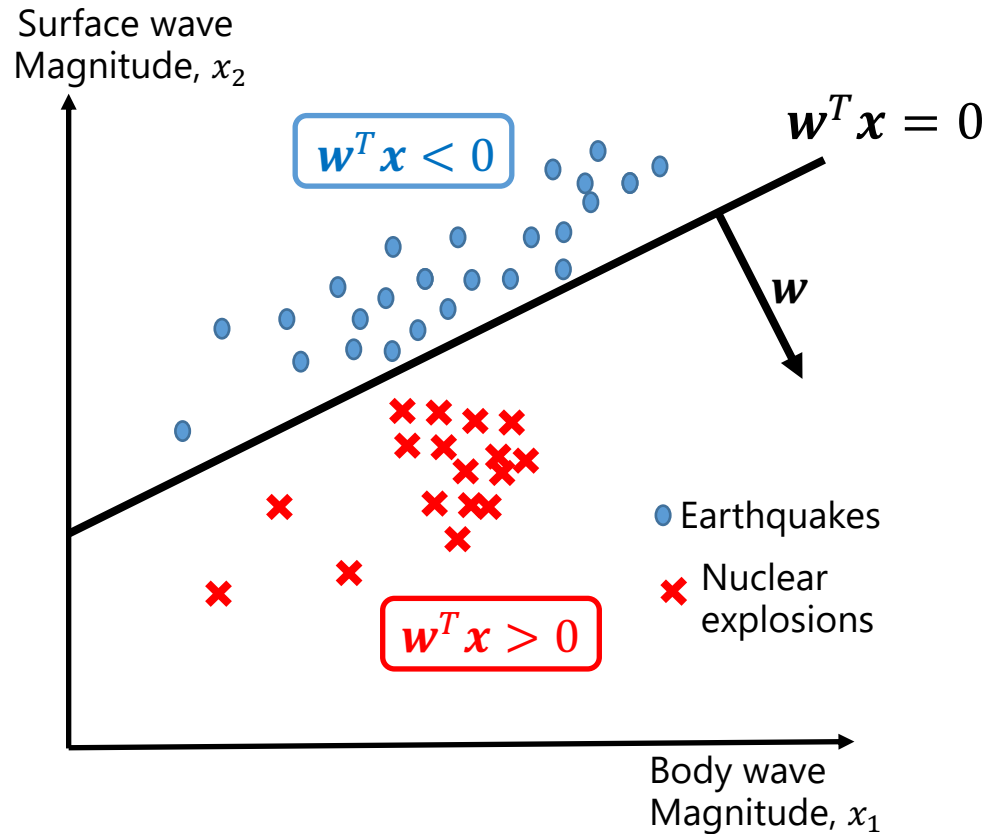
# Threshold Function



$$h_w(\mathbf{x}) = \begin{cases} 1, & \text{if } w^T \mathbf{x} > 0 \\ 0, & \text{Otherwise} \end{cases}$$

positive class  
(nuclear explosions)

negative class  
(Earthquake)



$$\mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}$$

# Threshold Function

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{w}^T \mathbf{x} > 0 \\ 0, & \text{Otherwise} \end{cases} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}$$

Loss Function: Mean Square Error (MSE)

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)})^2$$

**Issue:**  $h_{\mathbf{w}}(\mathbf{x}^{(i)})$  is not differentiable

**Solution:** Sigmoid/Logistic function

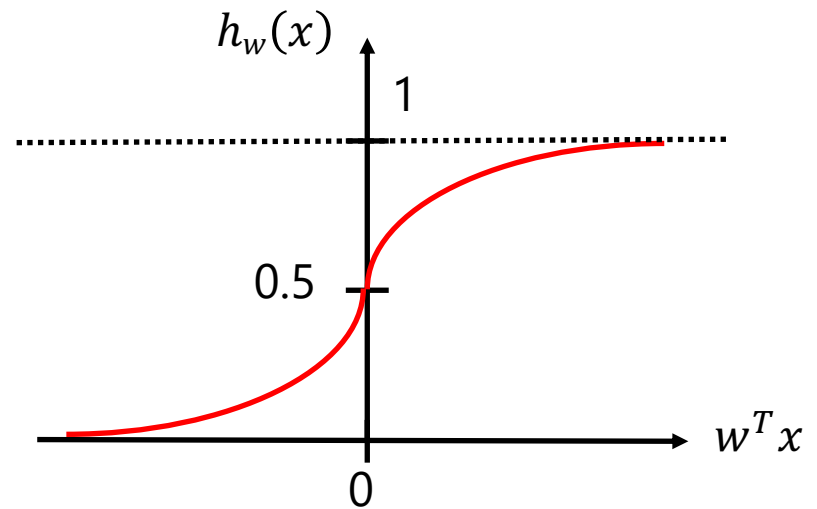
# Logistic Regression

Hypothesis:

$$h_w(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$

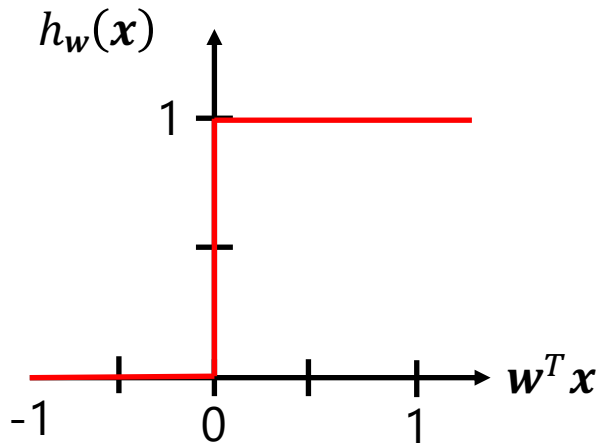
$$g(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid function  
or  
Logistic function



“logits”

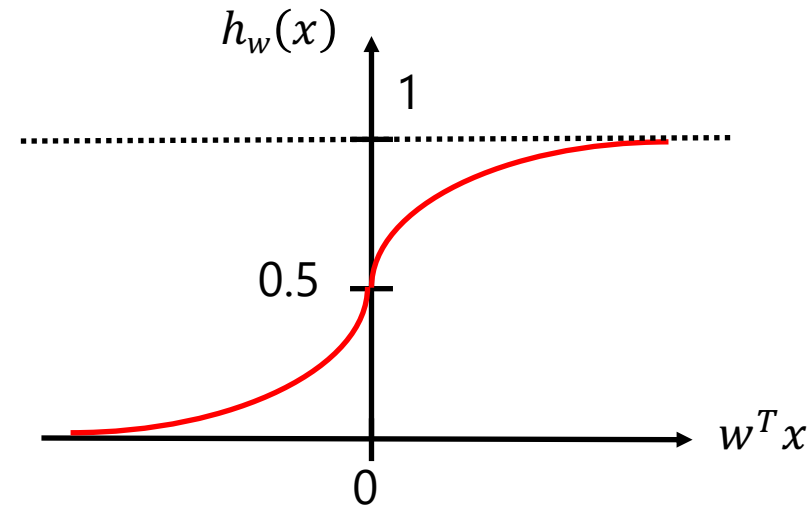
# Threshold Vs Logistic Function



$$h_w(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{w}^T \mathbf{x} > 0 \\ 0, & \text{Otherwise} \end{cases}$$

positive class  
(nuclear explosions)

negative class  
(Earthquake)



$$h_w(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$
$$g(z) = \frac{1}{1 + e^{-z}}$$

# Logistic Regression: Why not MSE Loss?

Mean Square Error (MSE) Loss:

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)})^2$$

$$\text{where } h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T \mathbf{x}}}$$

# Logistic Regression: Cross-Entropy

**Entropy:** Measure of randomness in a random variable

Let  $X$  be a discrete random variable with  $\text{dom}(X) = \{0, 1, \dots, C - 1\}$

with probability distribution  $\mathbf{P}(X) = \begin{bmatrix} P(X = 0) \\ P(X = 1) \\ \vdots \\ P(X = C - 1) \end{bmatrix}$ , then entropy  $H(\mathbf{P}(X))$  of the random variable  $X$  is defined as

$$H(\mathbf{P}(X)) = - \sum_{x \in \text{dom}(X)} P(X = x) \log P(X = x)$$

# Cross-Entropy: Definition

Let  $X$  and  $Y$  be two discrete random variables with  $\text{dom}(X) = \text{dom}(Y) = \{0, 1, \dots, C - 1\}$

Let  $\mathbf{P}(X)$  and  $\mathbf{P}(Y)$  be the probability distributions as shown below:

$$\mathbf{P}(X) = \begin{bmatrix} P(X = 0) \\ P(X = 1) \\ \vdots \\ P(X = C - 1) \end{bmatrix}$$

$$\mathbf{P}(Y) = \begin{bmatrix} P(Y = 0) \\ P(Y = 1) \\ \vdots \\ P(Y = C - 1) \end{bmatrix}$$

**Cross-Entropy**  $H(\mathbf{P}(X), \mathbf{P}(Y))$ :

$$H(\mathbf{P}(X), \mathbf{P}(Y)) = - \sum_{c \in \{0, 1, \dots, C - 1\}} P(X = c) \log P(Y = c)$$

# Binary Cross-Entropy (BCE): Definition

Let  $X$  and  $Y$  be two binary random variables with  $\text{dom}(X) = \text{dom}(Y) = \{0,1\}$

Let  $\mathbf{P}(X)$  and  $\mathbf{P}(Y)$  be the probability distributions as shown below:

$$\mathbf{P}(X) = \begin{bmatrix} P(X=0) \\ P(X=1) \end{bmatrix}$$

$$\mathbf{P}(Y) = \begin{bmatrix} P(Y=0) \\ P(Y=1) \end{bmatrix}$$

**Binary Cross-Entropy**  $H(\mathbf{P}(X), \mathbf{P}(Y))$ :

$$H(\mathbf{P}(X), \mathbf{P}(Y)) = - \sum_{c \in \{0,1\}} P(X=c) \log P(Y=c)$$

$$H(\mathbf{P}(X), \mathbf{P}(Y)) = -P(X=1) \log P(Y=1) - P(X=0) \log P(Y=0)$$

# Logistic Regression: BCE Loss

$$J(\mathbf{w}) = -\frac{1}{m} \sum_{i=1}^m J_i(\mathbf{w})$$

Cross-Entropy of  $\mathbf{P}(y^{(i)})$  and  $\mathbf{P}(\hat{y}^{(i)})$

where

$$J_i(\mathbf{w}) = H(\mathbf{P}(y^{(i)}), \mathbf{P}(\hat{y}^{(i)}))$$

$$= -\sum_{c \in \{0,1\}} \mathbf{P}(y^{(i)} = c) \log \mathbf{P}(\hat{y}^{(i)} = c)$$

$$= -\mathbf{P}(y^{(i)}=1) \log(\mathbf{P}(\hat{y}^{(i)}=1)) - \mathbf{P}(y^{(i)}=0) \log(\mathbf{P}(\hat{y}^{(i)}=0))$$

$$= -y^{(i)} \log h_w(\mathbf{x}^{(i)}) - (1 - y^{(i)}) \log (1 - h_w(\mathbf{x}^{(i)}))$$

Distribution of ground truth

$$\mathbf{P}(y^{(i)}) = \begin{bmatrix} P(y^{(i)} = 0) \\ P(y^{(i)} = 1) \end{bmatrix}$$

Distribution of output

$$\mathbf{P}(\hat{y}^{(i)}) = \begin{bmatrix} P(\hat{y}^{(i)} = 0) \\ P(\hat{y}^{(i)} = 1) \end{bmatrix}$$

Negative Log-Likelihood  $\equiv$  Binary Cross Entropy

Question: What are other measures?

# Logistic Regression: BCE Loss

$$J_i(\mathbf{w}) = H(\mathbf{P}(\mathbf{y}^{(i)}), \mathbf{P}(\hat{\mathbf{y}}^{(i)})) = - \sum_{c \in \{0,1\}} \mathbf{P}(\mathbf{y}^{(i)} = c) \log(\mathbf{P}(\hat{\mathbf{y}}^{(i)} = c))$$

Ground Truth		Prediction
Label	Probability Distribution $\mathbf{P}(\mathbf{y}^{(i)})$	Output Probability Distribution $\mathbf{P}(\hat{\mathbf{y}}^{(i)})$
$y^{(i)} = 1$	$\begin{bmatrix} P(y^{(i)} = 1) \\ P(y^{(i)} = 0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} P(\hat{y}^{(i)} = 1) \\ P(\hat{y}^{(i)} = 0) \end{bmatrix} = \begin{bmatrix} h_w(\mathbf{x}^{(i)}) \\ (1 - h_w(\mathbf{x}^{(i)})) \end{bmatrix}$
$y^{(i)} = 0$	$\begin{bmatrix} P(y^{(i)} = 1) \\ P(y^{(i)} = 0) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	

# Logistic Regression: BCE Loss

Often we see cross-entropy loss in the following form

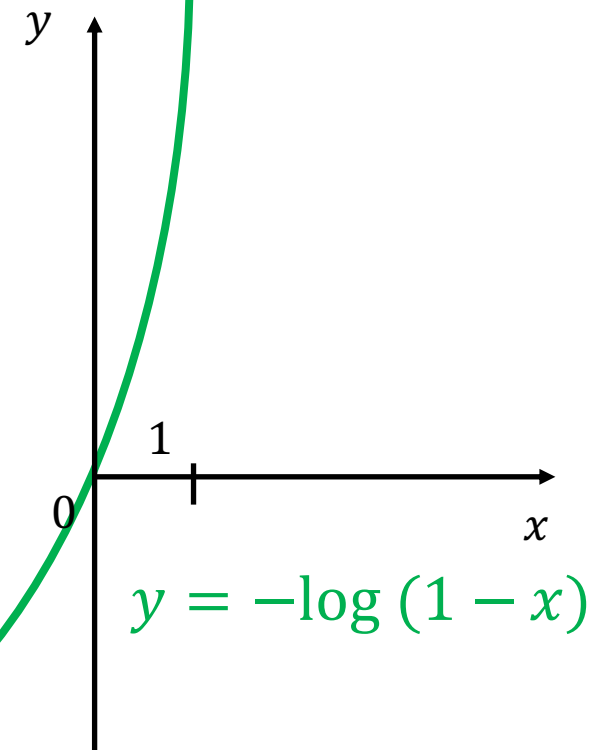
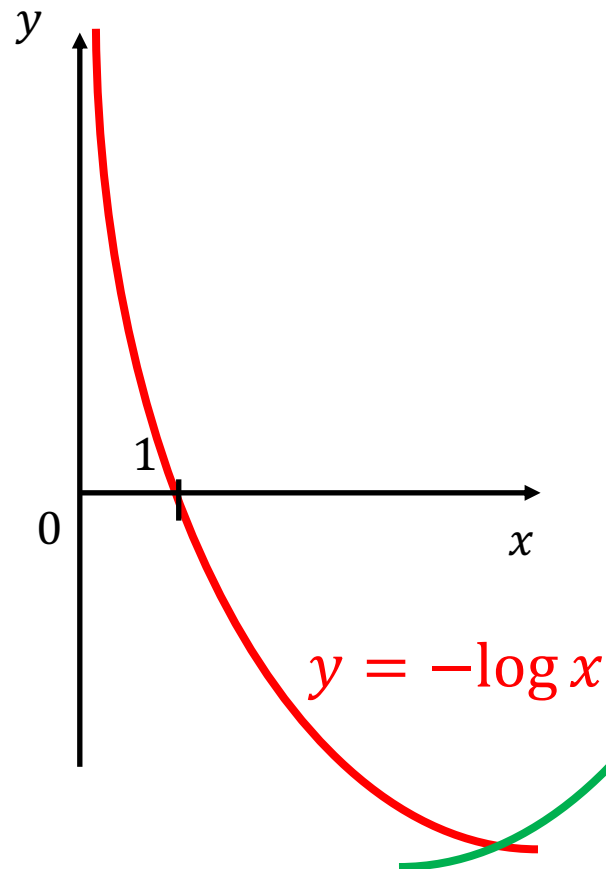
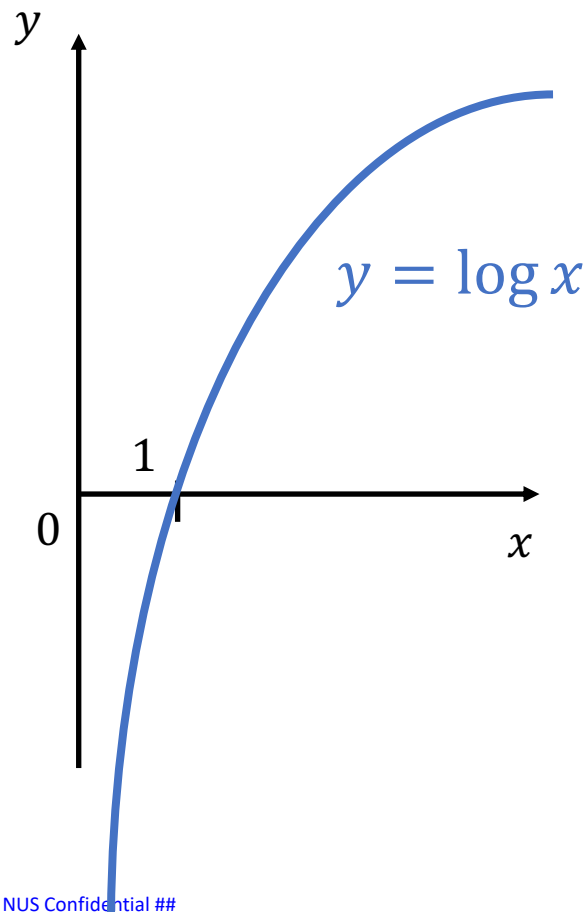
$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\mathbf{w}}(\mathbf{x}), y^{(i)})$$

where

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

# Understanding the Cost Function

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



# Cost Function

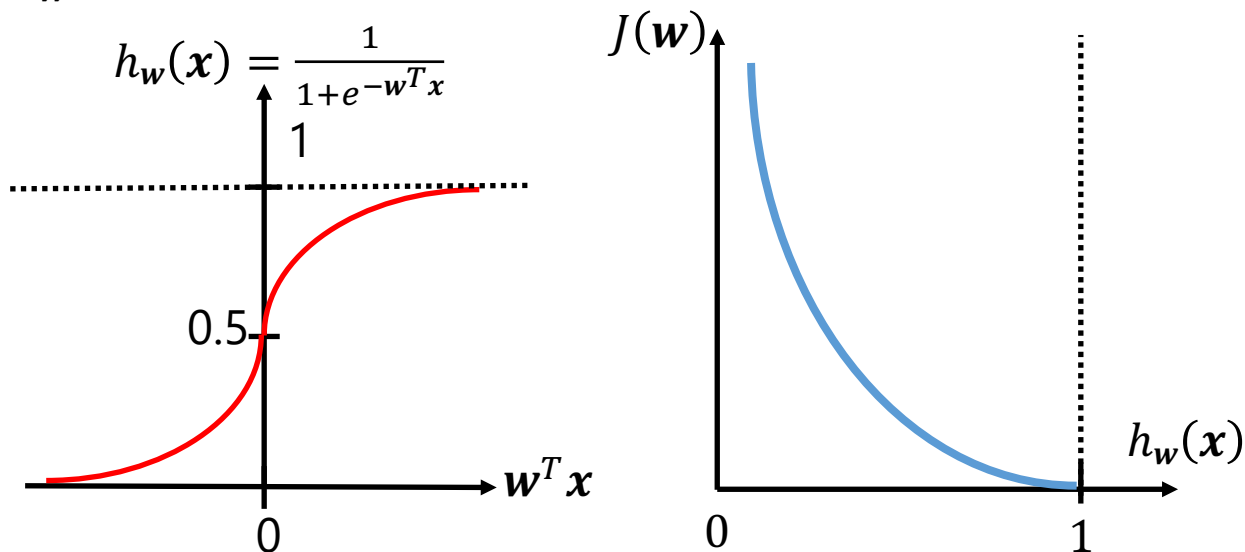
$$\text{Cost}(h_{\mathbf{w}}(\mathbf{x}), y) = \begin{cases} -\log h_{\mathbf{w}}(\mathbf{x}) & \text{if } y = 1 \\ -\log(1 - h_{\mathbf{w}}(\mathbf{x})) & \text{if } y = 0 \end{cases}$$

$$= -y \log h_{\mathbf{w}}(\mathbf{x}) - (1 - y) \log(1 - h_{\mathbf{w}}(\mathbf{x}))$$

Consider  $y = 1$ :

$$h_{\mathbf{w}}(\mathbf{x}) \rightarrow 0, \quad J(\mathbf{w}) \rightarrow \infty$$

$$h_{\mathbf{w}}(\mathbf{x}) \rightarrow 1, \quad J(\mathbf{w}) \rightarrow 0$$



# Logistic Regression: Minimizing BCE Loss

Cost Function:

$$J(\mathbf{w}) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log h_{\mathbf{w}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\mathbf{w}}(\mathbf{x}^{(i)}))$$

Gradient Descent:

$$\mathbf{w} := \mathbf{w} - \alpha \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

where

$$w_n := w_n - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_n^{(i)}$$

# Interpreting $h_w(\mathbf{x})$

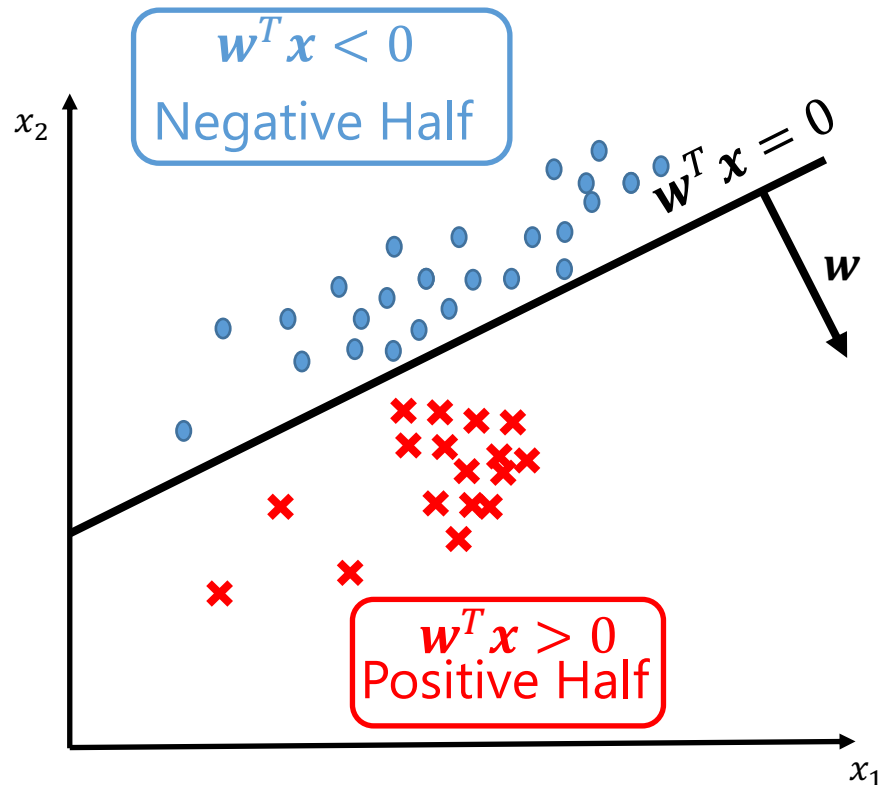
Hypothesis:  $h_w(\mathbf{x}) = \frac{1}{1 + e^{-w^T \mathbf{x}}}$

$h_w(\mathbf{x})$  = estimated probability that  
 $y = 1$  on input  $\mathbf{x}$ .

Why not  $(\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$   
as loss function for  
classification?

$(\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$  is the Widrow-Hoff loss

# Logistic Regression



Suppose  $\mathbf{x}$  is a **positive data sample**

- If  $\mathbf{x}$  is far away from decision surface in the positive half, the loss should be zero

(or)

If  $\mathbf{w}^T \mathbf{x} > 0$  and  $|\mathbf{w}^T \mathbf{x}|$  is large, loss should be zero

- If  $\mathbf{x}$  is far away from decision surface in the negative half, the loss should be high

(or)

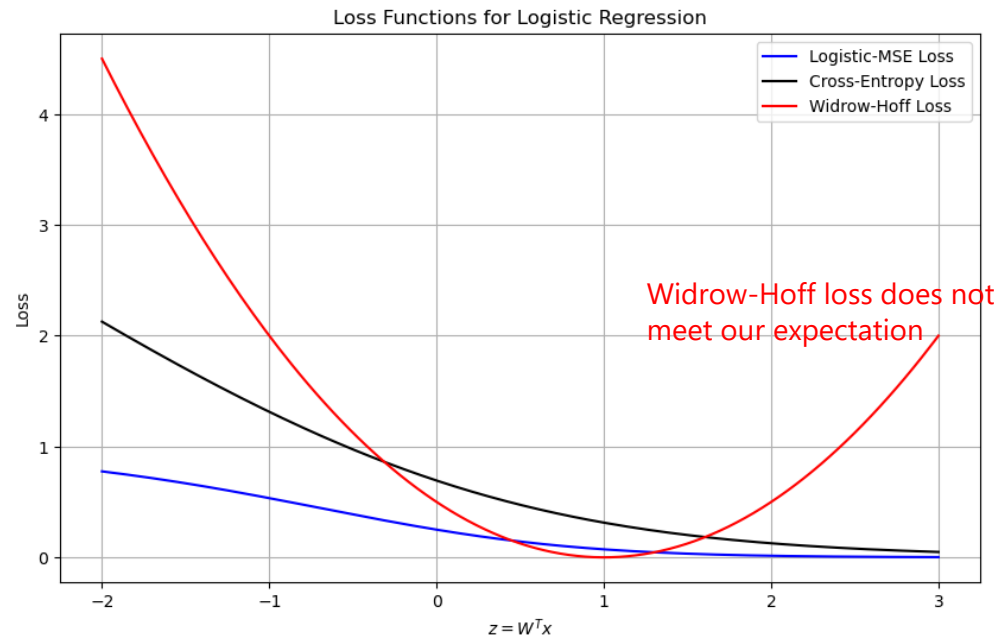
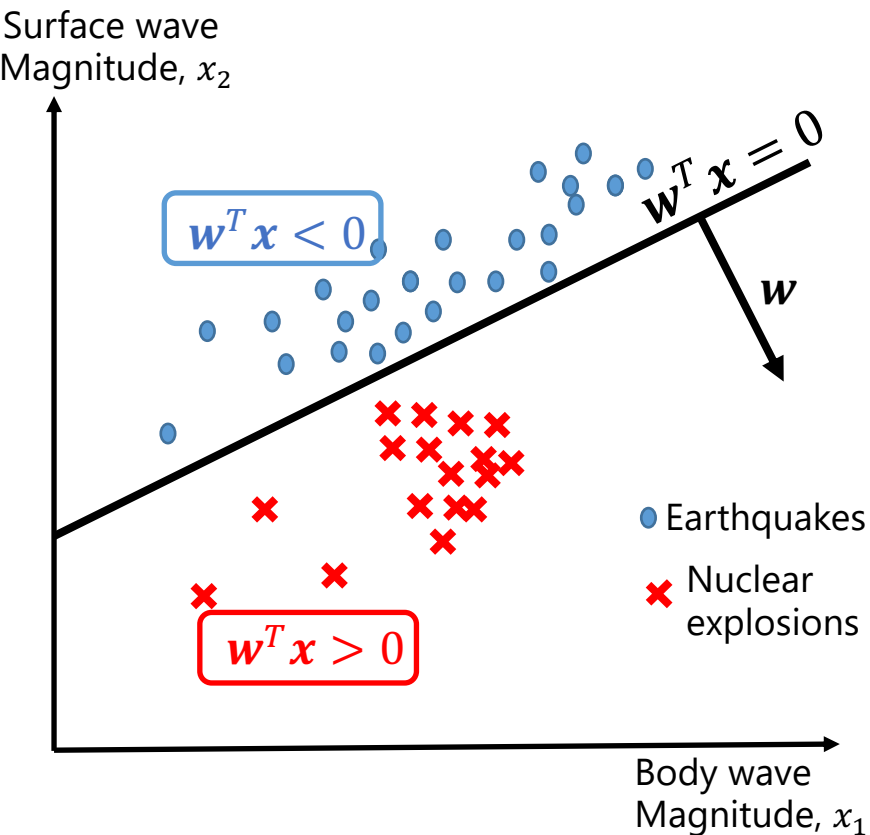
If  $\mathbf{w}^T \mathbf{x} < 0$  and  $|\mathbf{w}^T \mathbf{x}|$  is large, loss should be high

Distance between a data point  $\mathbf{x}$  and decision surface:  $\frac{|\mathbf{w}^T \mathbf{x}|}{\|\mathbf{w}\|}$

# Logistic- MSE vs Cross-Entropy Vs Widrow-Hoff

- What we want?

- If  $\mathbf{w}^T \mathbf{x} > 0$  and  $|\mathbf{w}^T \mathbf{x}|$  is large, loss should be zero
- If  $\mathbf{w}^T \mathbf{x} < 0$  and  $|\mathbf{w}^T \mathbf{x}|$  is large, loss should be high



# Iris Flower Dataset

## Measurements:

Sepal Length  
Sepal Width  
Petal Length  
Petal Width

Given the measurements, objective is to classify the data into one the three categories

virginica



versicolor

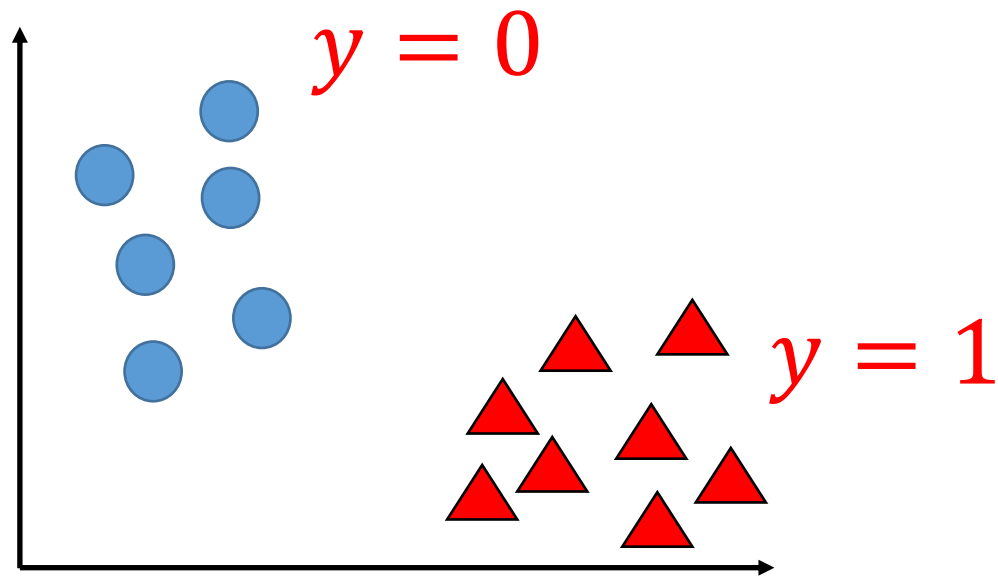


setosa



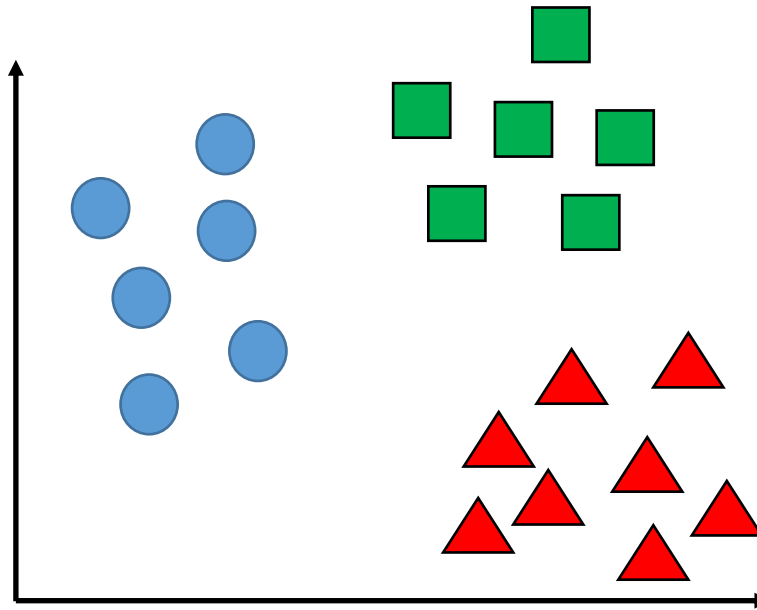
<https://academic.oup.com/jrssig/article/18/6/26/7038520?login=false>

# Binary classification



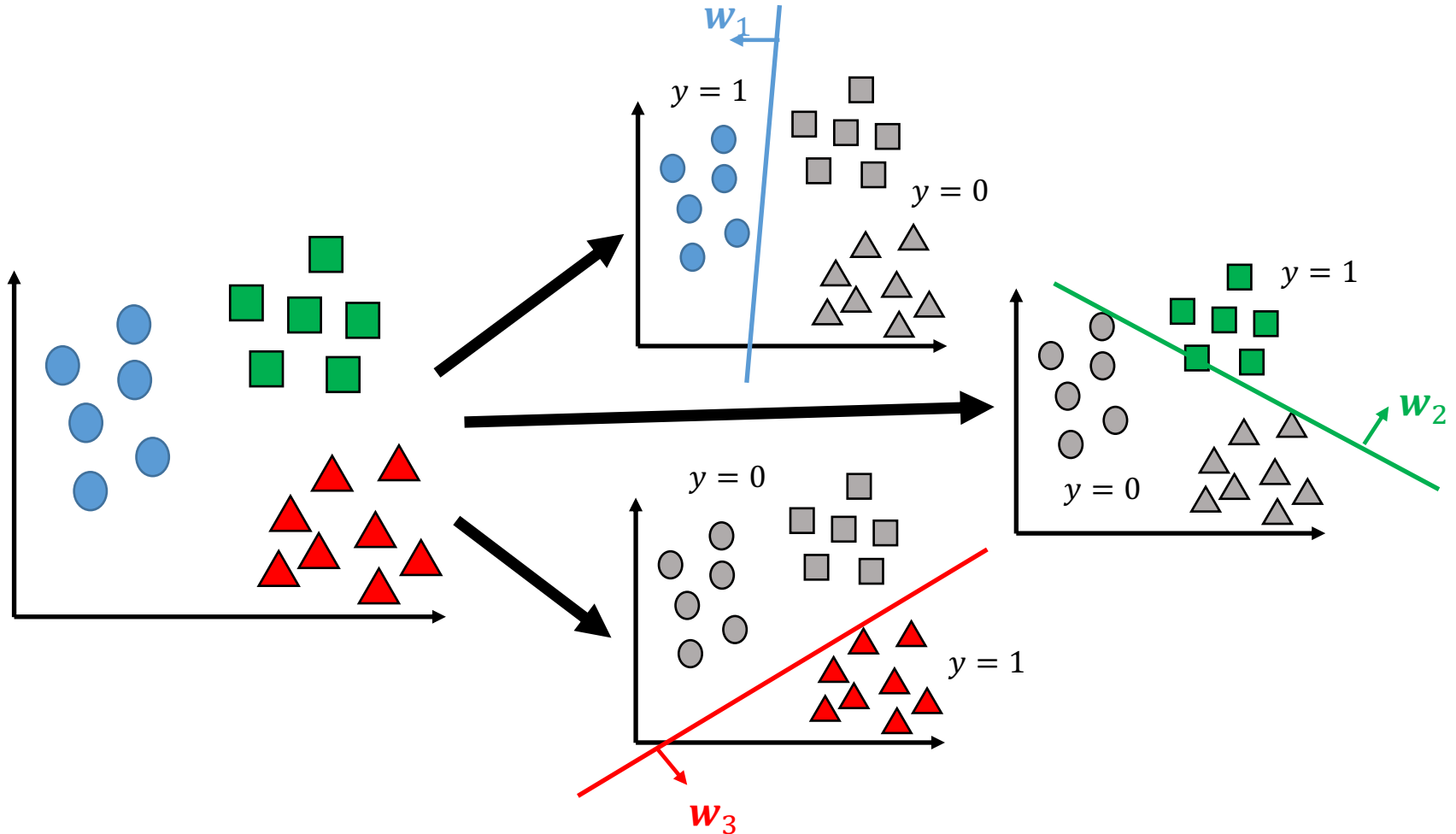
Logistic regression works!

# Multi-class classification



Can we still use logistic regression?

# Multi-class classification



# Multi-class classification

Train a logistic classifier  $h_w^{(c)}(\mathbf{x})$  for each class  $c$  to predict that  $y = c$

For each input  $x$ , pick the class  $c$  that maximizes:

$$\max_c h_w^{(c)}(\mathbf{x})$$

one-vs-all  
one-vs-rest

$h_w(x)$  = estimated probability that  $y = 1$  on input  $x$ .

# Multi-Class Cross-Entropy Loss

**Cross-Entropy**  $H(\mathbf{P}(X), \mathbf{P}(Y))$ :

$$H(\mathbf{P}(X), \mathbf{P}(Y)) = - \sum_{c \in \{1, 2, \dots, C\}} P(X = c) \log P(Y = c)$$

$$J_i(\mathbf{w}) = - \sum_{c=1}^C P(\mathbf{y}^{(i)} = c) \log(P(\hat{\mathbf{y}}^{(i)} = c))$$

$\mathbf{P}(\mathbf{y}^{(i)})$  is the one-hot encoding of label

$$\mathbf{P}(\mathbf{y}^{(i)}) = \begin{bmatrix} P(\mathbf{y}^{(i)} = 1) \\ P(\mathbf{y}^{(i)} = 2) \\ \vdots \\ P(\mathbf{y}^{(i)} = C) \end{bmatrix}$$

Ground Truth	
Label	Probability Distribution $\mathbf{P}(\mathbf{y}^{(i)})$
$\mathbf{y}^{(i)} = 1$	$\mathbf{P}(\mathbf{y}^{(i)}) = [1, 0, \dots, 0]^T$
$\mathbf{y}^{(i)} = 2$	$\mathbf{P}(\mathbf{y}^{(i)}) = [0, 1, \dots, 0]^T$
$\vdots$	$\vdots$
$\mathbf{y}^{(i)} = C$	$\mathbf{P}(\mathbf{y}^{(i)}) = [0, 0, \dots, 1]^T$

Is  $\mathbf{P}(\mathbf{y}^{(i)})$  a probability distribution?

# Multi-Class Cross-Entropy Loss

Is  $\mathbf{P}(\hat{y}^{(i)}) = \begin{bmatrix} P(\hat{y}^{(i)} = 1) \\ P(\hat{y}^{(i)} = 2) \\ \vdots \\ P(\hat{y}^{(i)} = C) \end{bmatrix} = \begin{bmatrix} h_w^{(1)}(x^{(i)}) \\ h_w^{(2)}(x^{(i)}) \\ \vdots \\ h_w^{(C)}(x^{(i)}) \end{bmatrix}$ , a probability distribution?

No

where  $h_w^{(c)}(x^{(i)}) = \sigma(\mathbf{w}_c^T \mathbf{x})$

How to make it a probability distribution?

Normalization!

$$\mathbf{P}(\hat{y}^{(i)}) = \begin{bmatrix} \hat{y}_1^{(i)} \\ \hat{y}_2^{(i)} \\ \vdots \\ \hat{y}_C^{(i)} \end{bmatrix} = \text{Softmax} \left( \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \\ \vdots \\ z_C^{(i)} \end{bmatrix} \right) = \begin{bmatrix} \frac{e^{z_1^{(i)}}}{e^{z_1^{(i)}} + e^{z_2^{(i)}} + \dots + e^{z_C^{(i)}}(x)} \\ \frac{e^{z_2^{(i)}}}{e^{z_1^{(i)}} + e^{z_2^{(i)}} + \dots + e^{z_C^{(i)}}(x)} \\ \vdots \\ \frac{e^{z_C^{(i)}}}{e^{z_1^{(i)}} + e^{z_2^{(i)}} + \dots + e^{z_C^{(i)}}(x)} \end{bmatrix},$$

where  $z_c^{(i)} = \mathbf{w}_c^T \mathbf{x}$  and  $\mathbf{w}_c$  is the weight vector for  $c$ -th class

# Loss function for Multiclass Classification

$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m J_i(\mathbf{w})$$

where

$$J_i(\mathbf{w}) = - \sum_{c=1}^C P(y^{(i)} = c) \log(P(\hat{y}^{(i)} = c))$$

# Model Evaluation

Only for your reading

# Training/testing procedure for linear regression

1. Learn  $\mathbf{w}$  from training set by minimizing  $J(\mathbf{w})$  :

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)})^2$$

2. Compute test set error

$$J_{test}(\mathbf{w}) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_{\mathbf{w}}(\mathbf{x}_{test}^{(i)}) - y_{test}^{(i)})^2$$

# Training/testing procedure for logistic regression

1. Learn  $\mathbf{w}$  from training set by minimizing  $J(\mathbf{w})$
2. Compute test set error

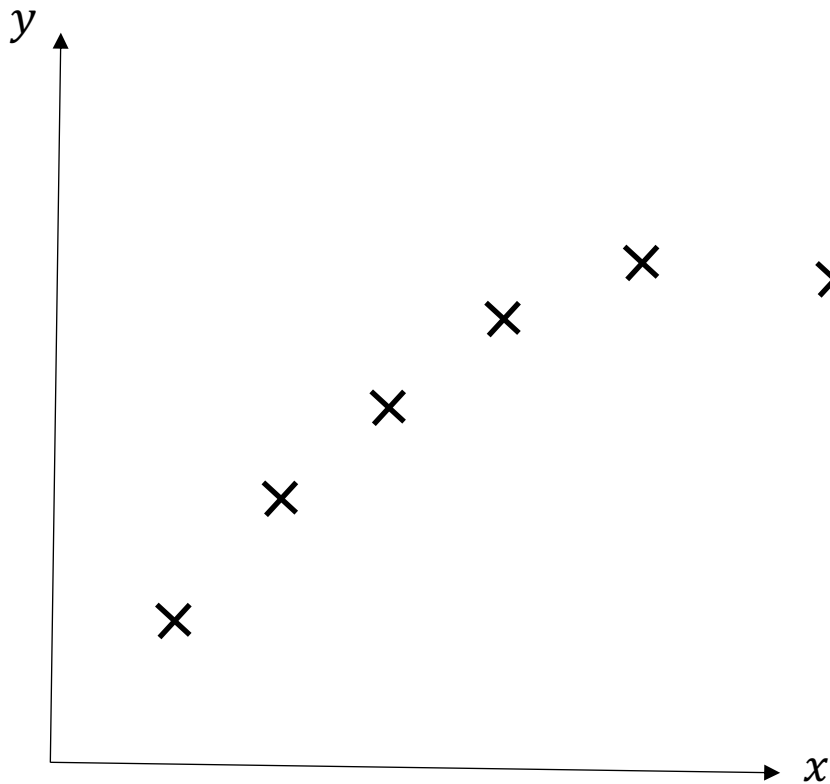
$$J_{test}(\mathbf{w}) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_{\mathbf{w}}(\mathbf{x}_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log(1 - h_{\mathbf{w}}(\mathbf{x}_{test}^{(i)}))$$

# Training/testing procedure for logistic regression

## 3. Misclassification Error

$$error(h_{\mathbf{w}}(\mathbf{x}), y) = \begin{cases} 1, & \text{if } h_{\mathbf{w}}(\mathbf{x}) \leq 0.5 \text{ and } y = 1 \\ & \text{or } h_{\mathbf{w}}(\mathbf{x}) > 0.5 \text{ and } y = 0 \\ 0, & \text{otherwise} \end{cases}$$

$$test\ error = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} error(h_{\mathbf{w}}(\mathbf{x}_{test}^{(i)}), y_{test}^{(i)})$$



Which hypothesis should we pick?

1.  $y = w_0 + w_1x$
2.  $y = w_0 + w_1x + w_2x^2$
3.  $y = w_0 + w_1x + w_2x^2 + w_3x^3$
4.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$
5.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5$
6.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$

# Model Selection

1.  $y = w_0 + w_1x$

2.  $y = w_0 + w_1x + w_2x^2$

3.  $y = w_0 + w_1x + w_2x^2 + w_3x^3$

4.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$

5.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5$

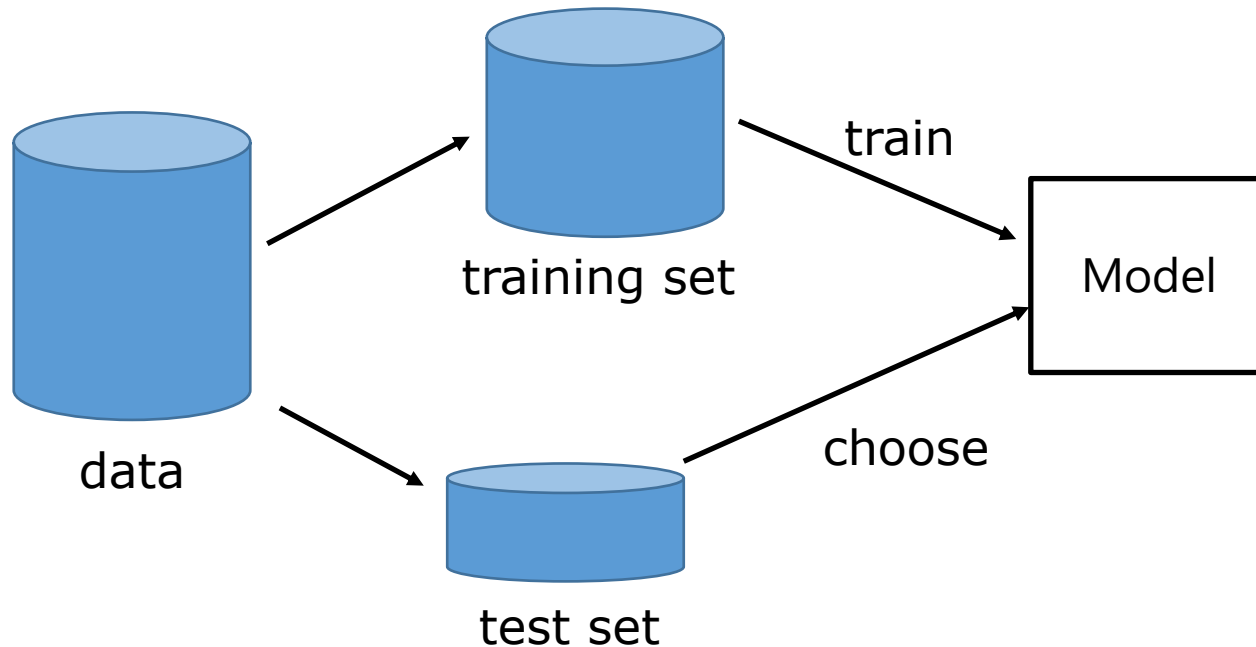
6.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$

Train each model

Compute  
 $J_{test}(\mathbf{w})$

Pick model with the lowest  $J_{test}(\mathbf{w})$ !

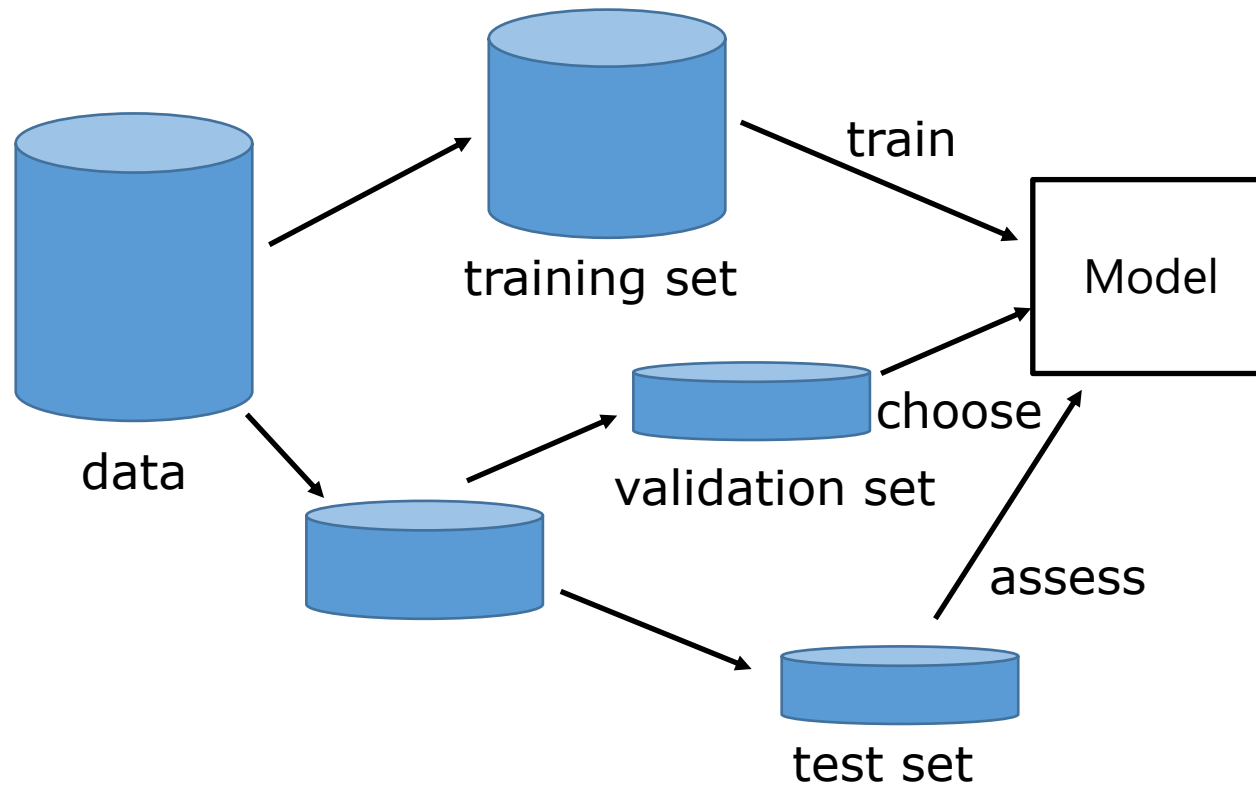
# Evaluating the hypothesis



Can we report the goodness of  
our model with  $J_{test}(w)$ ?

Model might be biased!

# Evaluating the hypothesis



# Measuring Goodness

## 1. Training Error

$$J_{train}(\mathbf{w}) = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} (h_{\mathbf{w}}(\mathbf{x}_{train}^{(i)}) - y_{train}^{(i)})^2$$

## 2. Cross Validation Error

$$J_{cv}(\mathbf{w}) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_{\mathbf{w}}(\mathbf{x}_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

## 3. Test Error

$$J_{test}(\mathbf{w}) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_{\mathbf{w}}(\mathbf{x}_{test}^{(i)}) - y_{test}^{(i)})^2$$

# Model Selection

1.  $y = w_0 + w_1x$

2.  $y = w_0 + w_1x + w_2x^2$

3.  $y = w_0 + w_1x + w_2x^2 + w_3x^3$

4.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$

5.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5$

6.  $y = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$

Train each model

Compute

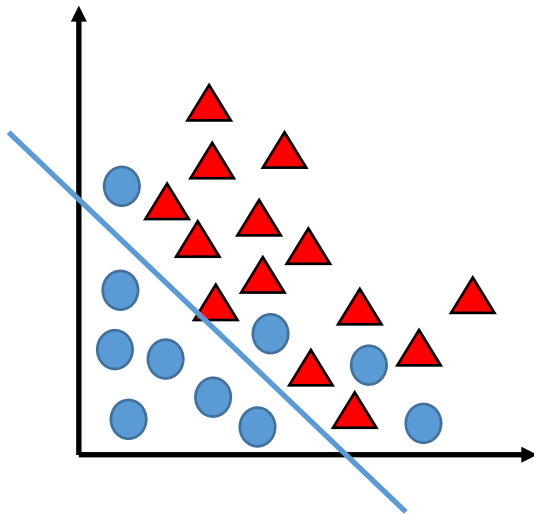
$J_{cv}(\mathbf{w})$

Pick model with  
the lowest

$J_{cv}(\mathbf{w})!$

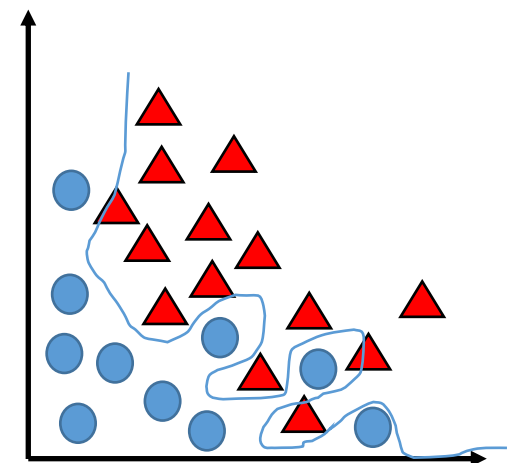
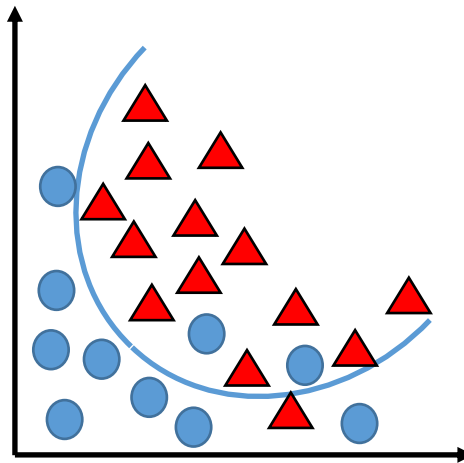
Use  $J_{test}(\mathbf{w})$  to estimate performance on  
unseen samples

# Bias and Variance



Underfit

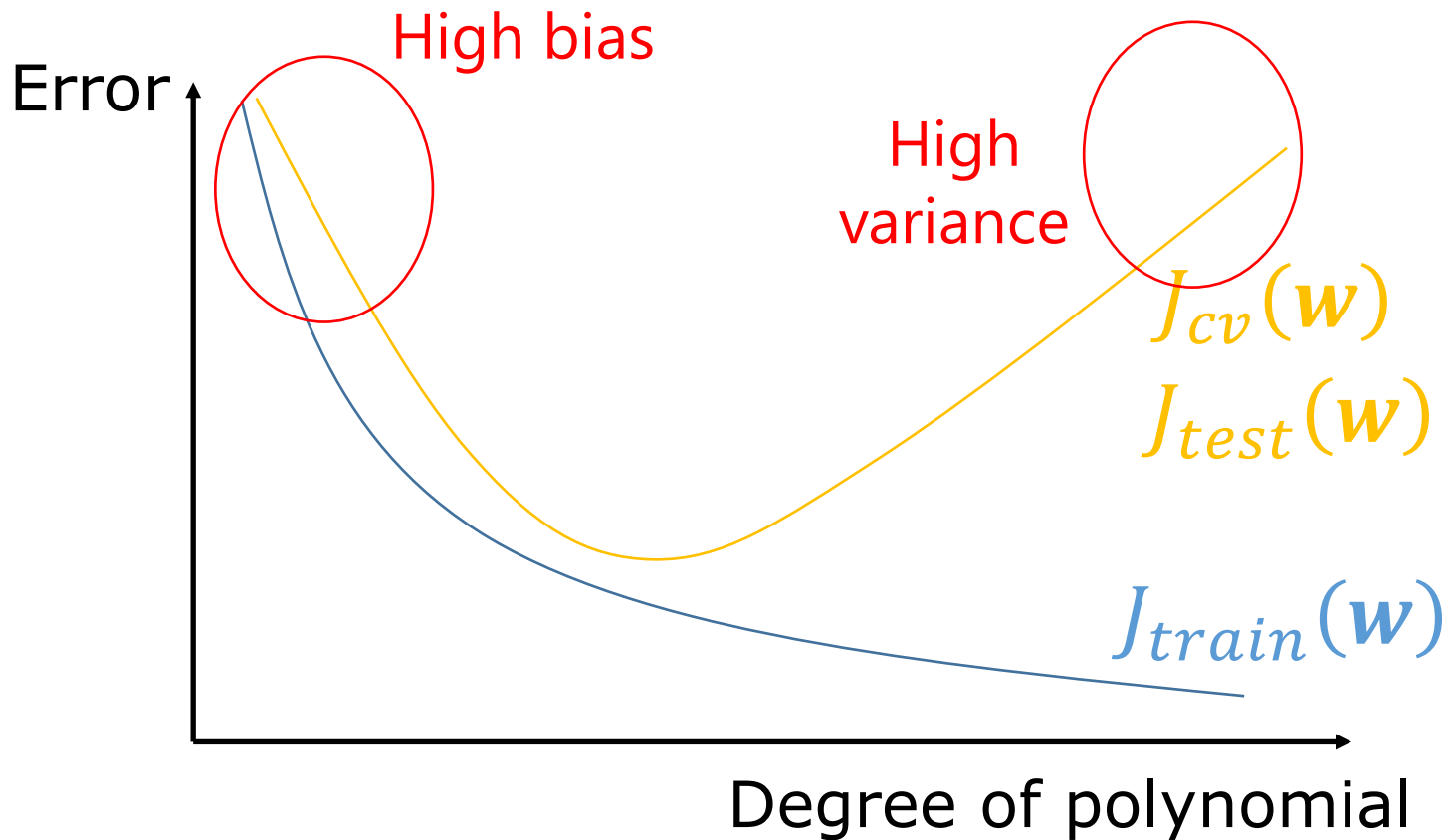
High bias



Overfit

High variance

# Bias and Variance



# Diagnosing Bias vs Variance

- Bias (underfit):

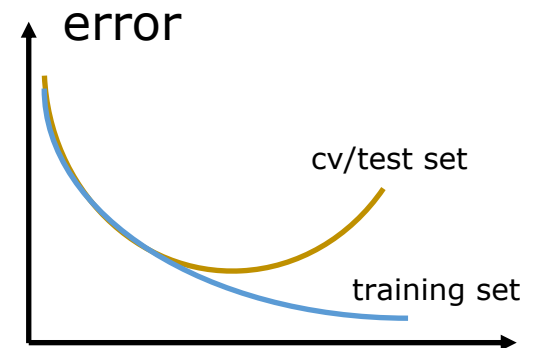
$J_{train}(\mathbf{w})$  will be high

$$J_{train}(\mathbf{w}) \approx J_{cv}(\mathbf{w})$$

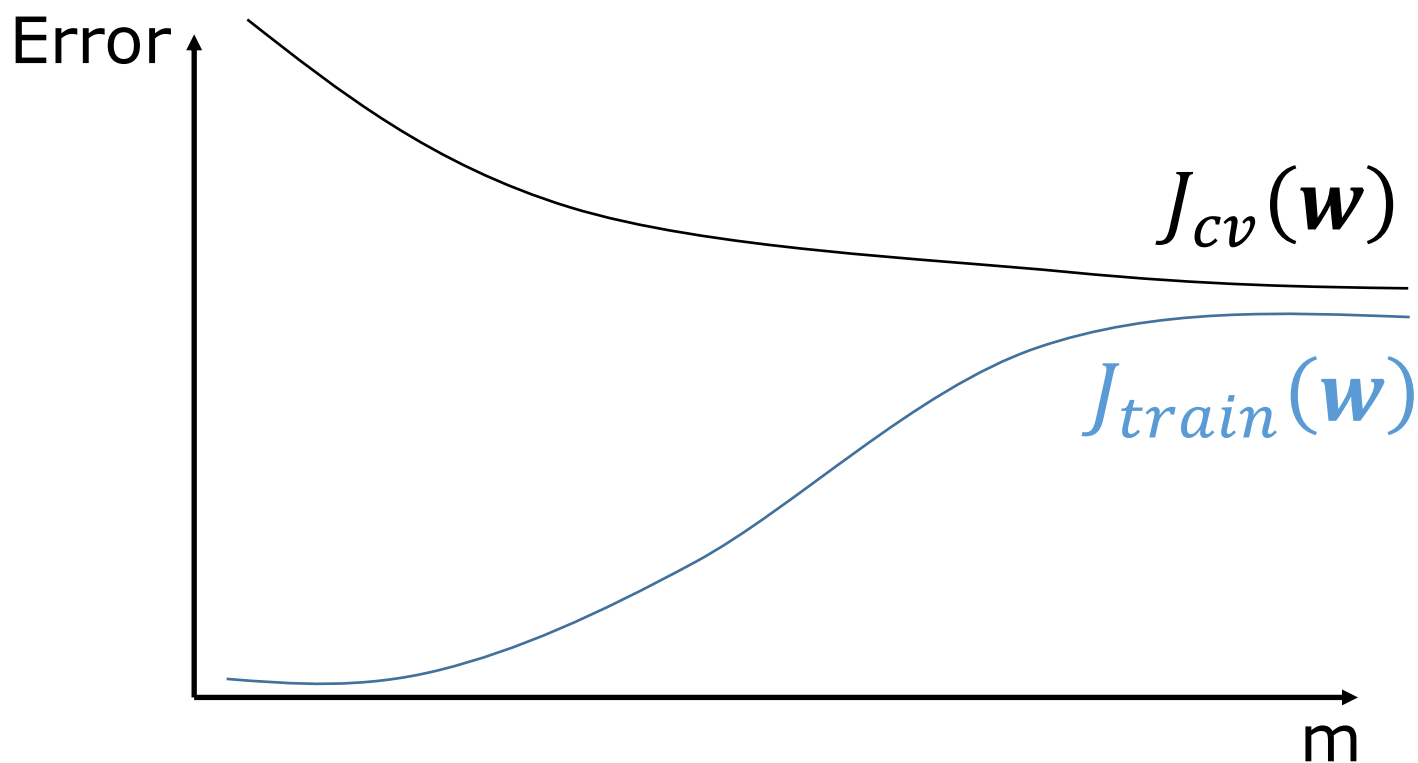
- Variance (overfit):

$J_{train}(\mathbf{w})$  will be low

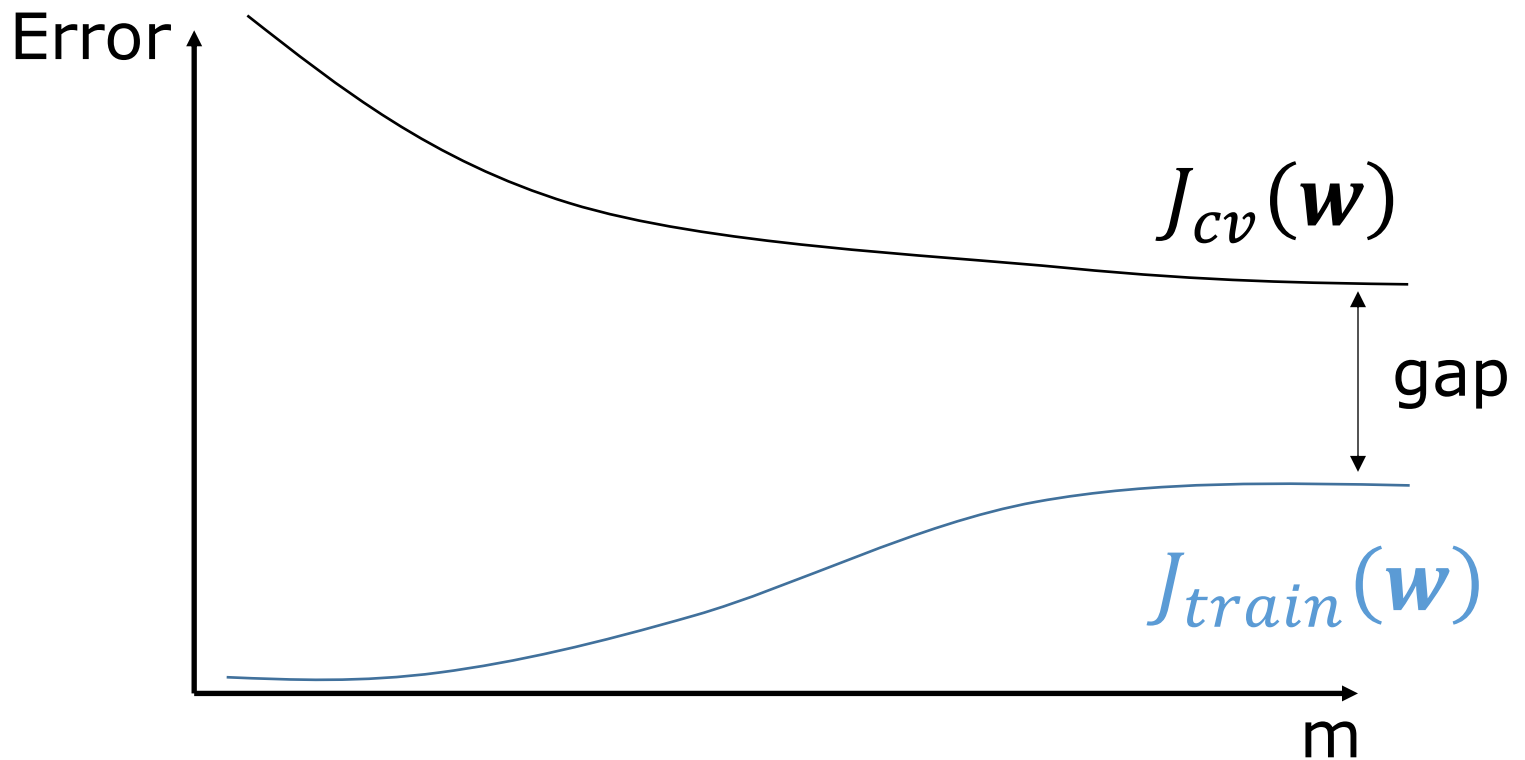
$$J_{cv}(\mathbf{w}) \gg J_{train}(\mathbf{w})$$



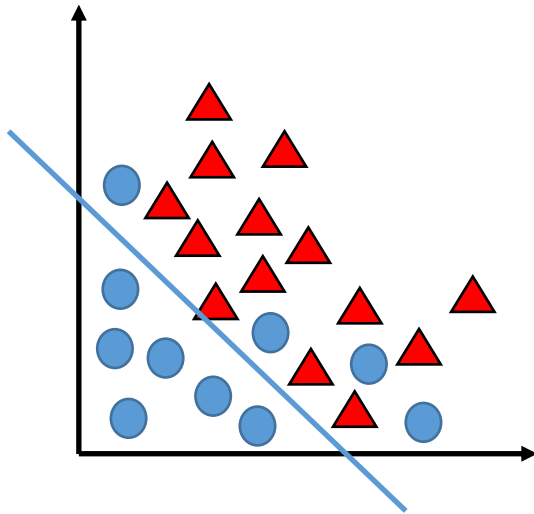
# Impact of $m$ : High Bias



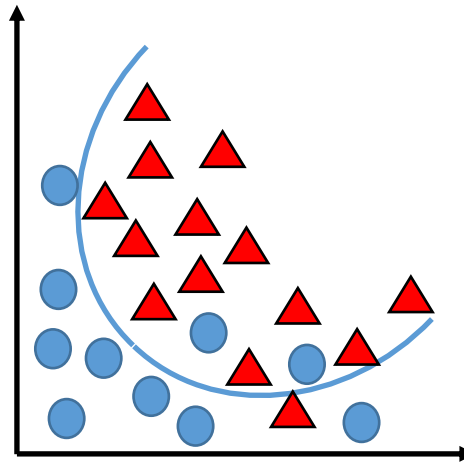
# Impact of $m$ : High Variance



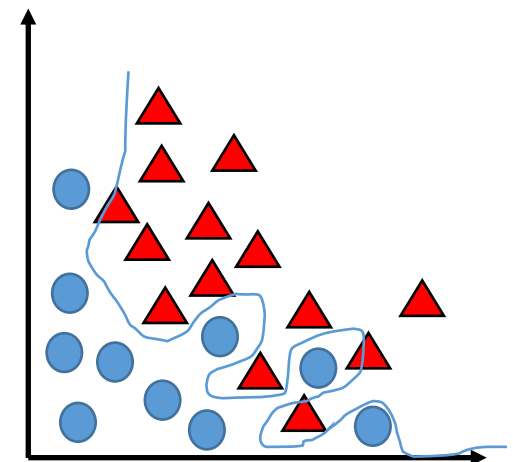
# More about Overfitting



$h_w(x) = g(w_0 + w_1x_1 + w_2x_2)$   
"under-fitting"



$h_w(x) = g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2)$



$h_w(x) = g(w_0 + w_1x_1 + w_2x_1^2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + \dots)$   
"over-fitting"

# Addressing Overfitting

If we have a lot of features, we can certainly fit the training set very well.....

... but the hypothesis might end up making poor predictions

# Addressing Overfitting

1. Reduce the number of features
2. Regularization
  - Keep all features but reduce the magnitude of the parameters  $w_j$

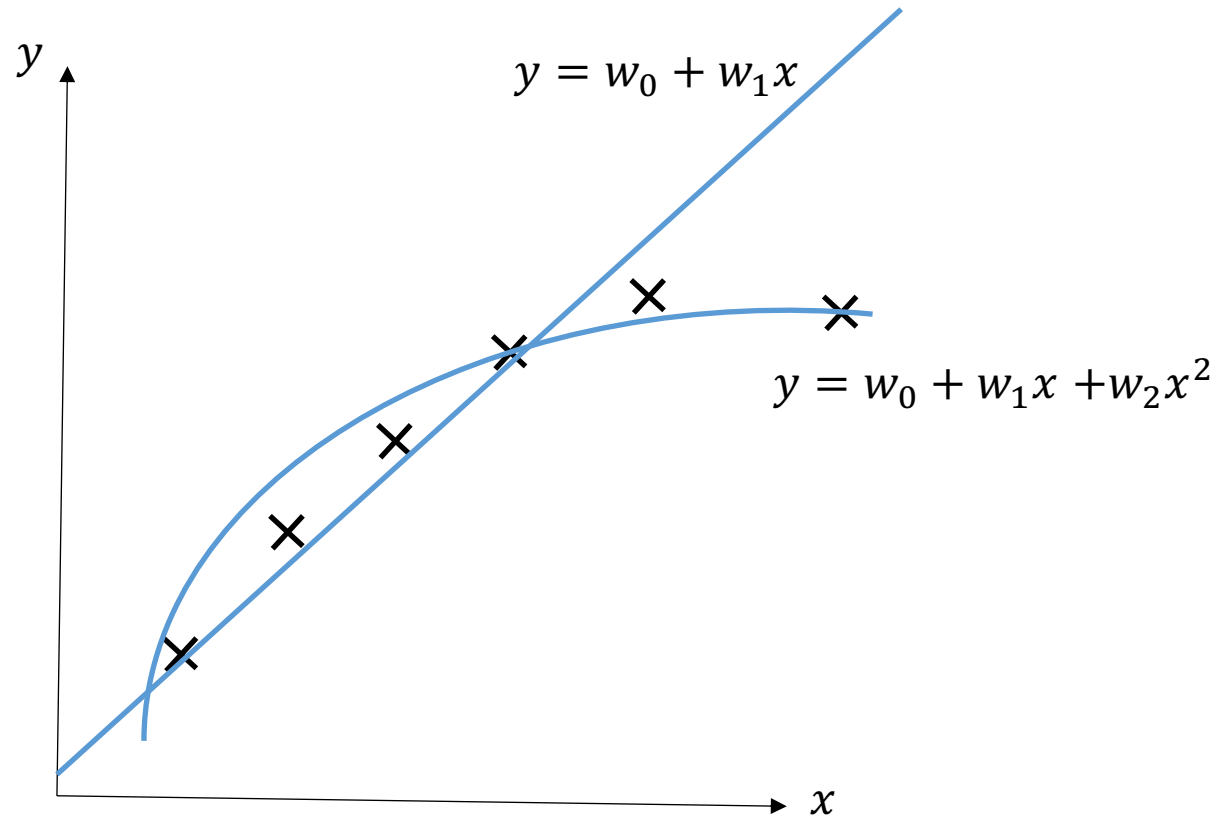
# Regularization: Linear Regression

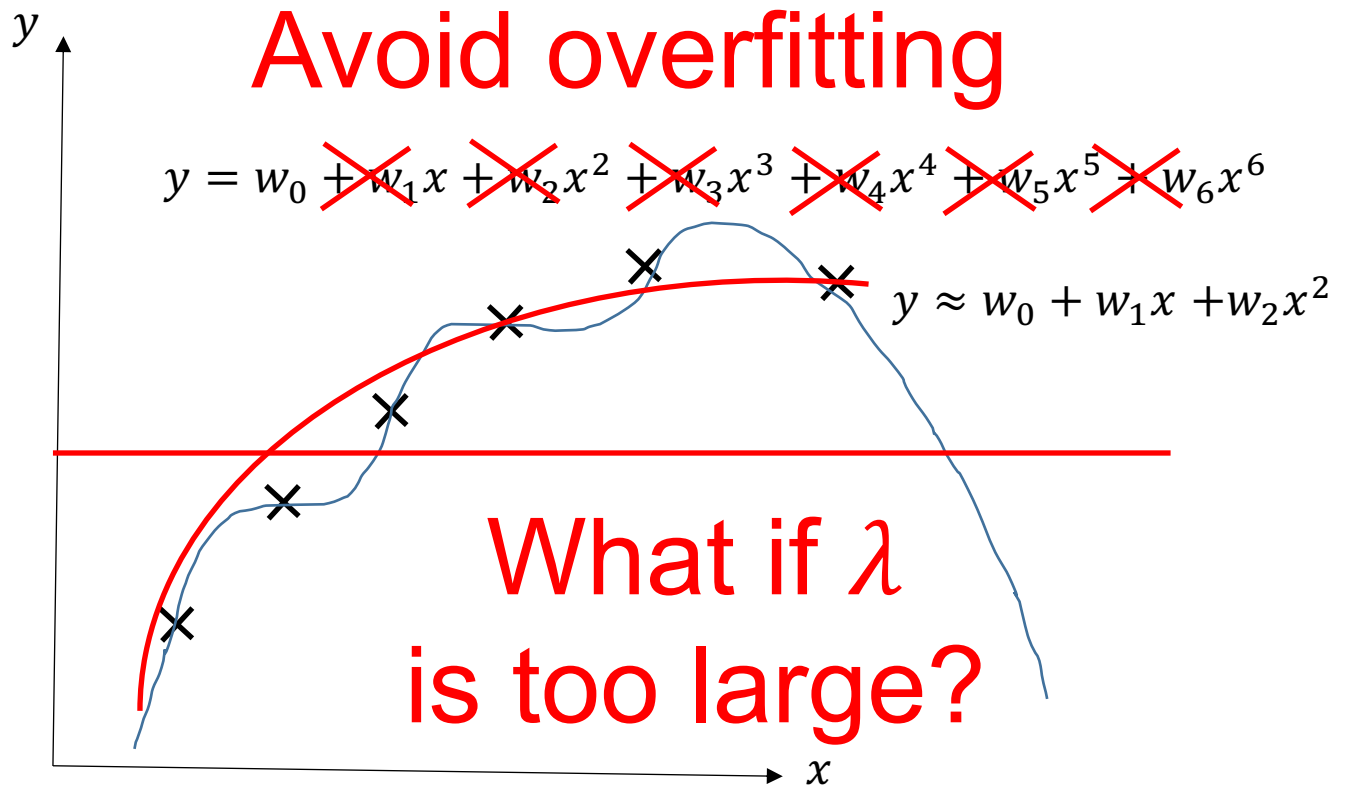
Hypothesis:

$$h_{\mathbf{w}}(\mathbf{x}): \mathbf{w}^T \mathbf{x}$$

Cost Function: **Regularization**

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





Regularization will make  $w_i$ ,  $i = 1, \dots, 6$  small(er)

# Linear Regression with Regularization

Hypothesis:

$$h_{\mathbf{w}}(x): \mathbf{w}^T \mathbf{x}$$

Cost Function:

$$J(\mathbf{w}) = \frac{1}{2m} \left[ \underbrace{\sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)})^2}_{\text{fitting data "well"}} + \underbrace{\lambda \sum_{i=1}^n w_i^2}_{\text{avoid "over-fitting"}} \right]$$

regularization  
parameter

# Gradient Descent for Linear Regression with Regularization

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

Repeat {

$$w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$w_1 := w_1 - \alpha \frac{1}{m} \left[ \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_1^{(i)} + \lambda w_1 \right]$$

$$\vdots$$
$$w_n := w_n - \alpha \frac{1}{m} \left[ \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_n^{(i)} + \lambda w_n \right]$$

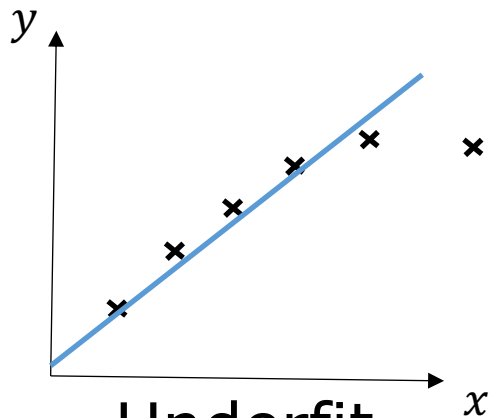
}

# Gradient Descent for Linear Regression with Regularization

$$\begin{aligned} w_n &:= w_n - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_n^{(i)} - \alpha \frac{\lambda}{m} w_n \\ &:= \underbrace{\left(1 - \frac{\alpha\lambda}{m}\right)}_{\text{slightly less than 1}} w_n - \underbrace{\alpha \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_n^{(i)}}_{\text{“regular” gradient descent}} \end{aligned}$$

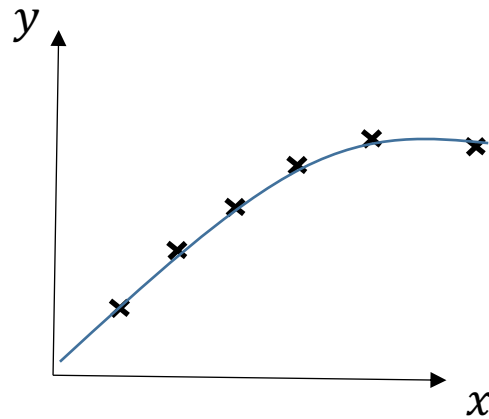
# Another example: linear regression with regularization

$$J(w) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^n w_i^2 \right]$$

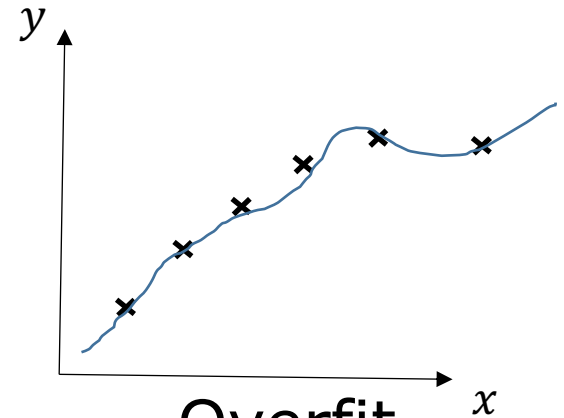


Underfit

large  $\lambda$



moderate  $\lambda$



Overfit

$\lambda = 0$

# Model Selection

1. Try  $\lambda = 0$
2. Try  $\lambda = 0.01$
3. Try  $\lambda = 0.02$
4. Try  $\lambda = 0.04$
5. Try  $\lambda = 0.08$
6. Try  $\lambda = 0.16$
- ...

Train each model

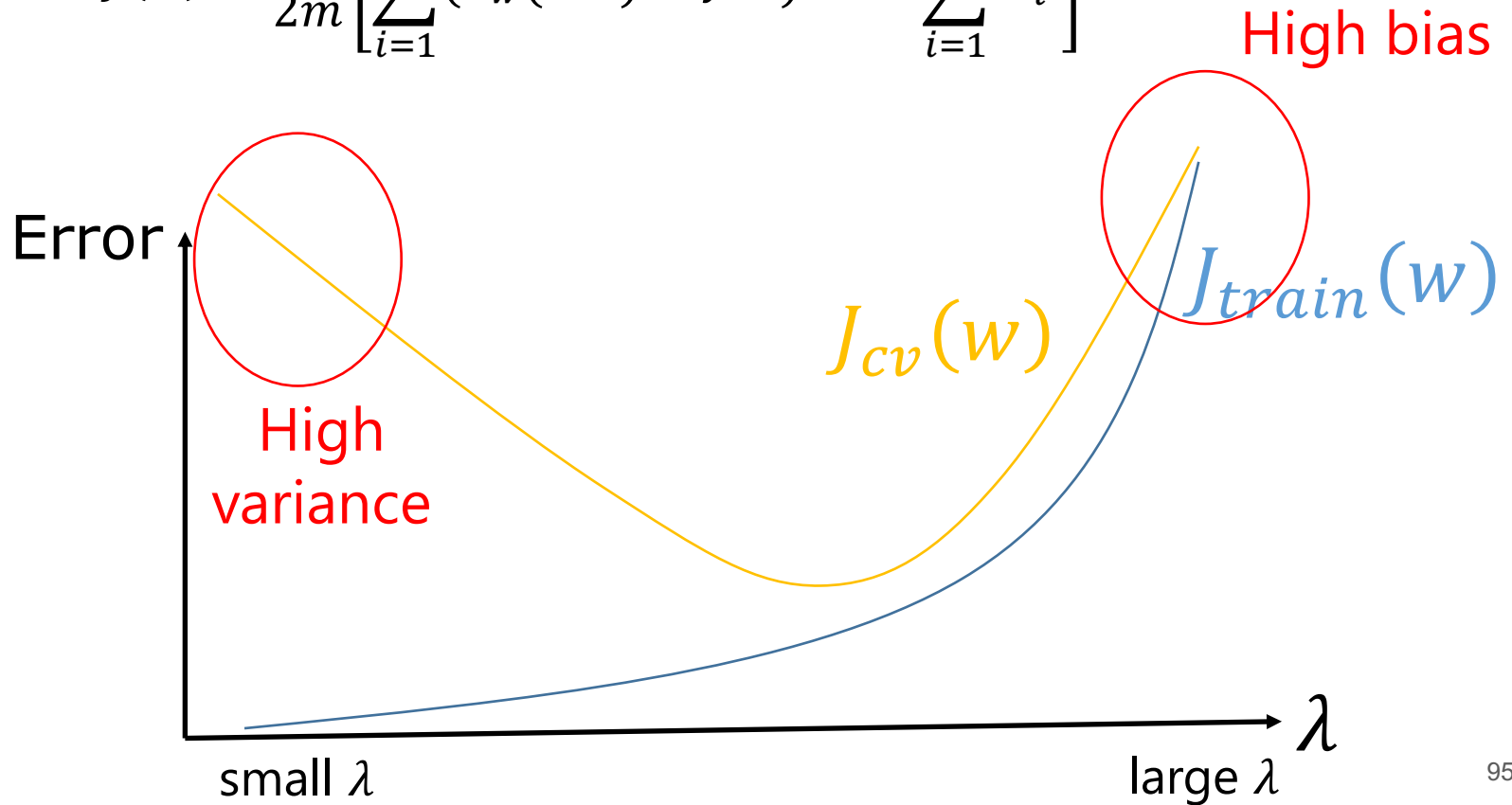
Compute  $J_{cv}(w)$

Pick model with the  
lowest  $J_{cv}(w)$ !

Use  $J_{test}(w)$  to estimate performance on  
unseen samples

# Another example: linear regression with regularization

$$J(w) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_w(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^n w_i^2 \right]$$



# Logistic Regression with Regularization

Hypothesis:

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

Cost Function:

$$J(\mathbf{w}) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log h_{\mathbf{w}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\mathbf{w}}(\mathbf{x}^{(i)})) + \frac{\lambda}{2m} \sum_{i=1}^n w_i^2$$

Gradient Descent:

$$w_n := w_n - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_n^{(i)} - \alpha \frac{\lambda}{m} w_n$$