# Introduction to Data Science

## (Lecture 12)

**Dr. Mohammad Pourhomayoun**

Assistant Professor

Computer Science Department

California State University, Los Angeles

# Linear Regression with multiple features

# Linear Regression with One Feature

One Feature → Size in feet² (x)

Target → Price (y)

| Size in feet² (x) | Price (y) |
|---|---|
| 1000 | 410K |
| 1200 | 600K |
| 1230 | 620K |
| 1340 | 645K |
| … | … |

# Linear Regression with One Feature

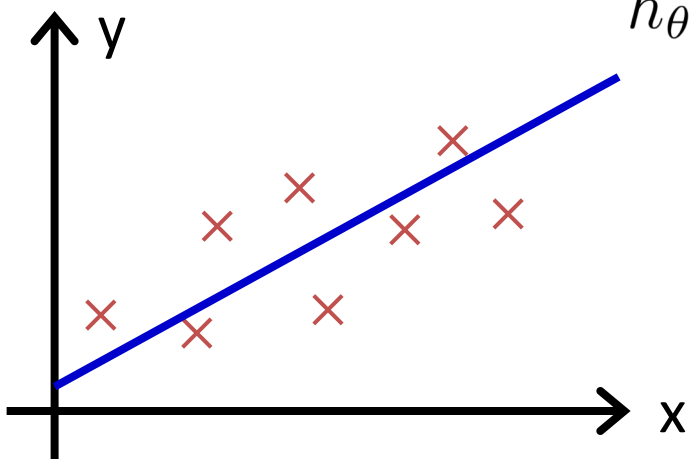
One Feature

Target

| Size in feet$^2$ (x) | Price (y) |
|---|---|
| 1000 | 410K |
| 1200 | 600K |
| 1230 | 620K |
| 1340 | 645K |
| … | … |

$$h_\theta(x) = \theta_0 + \theta_1 x$$

# Linear Regression with One Feature




$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$\boxed{h_\theta(x) = \theta_0 + \theta_1 x}$$

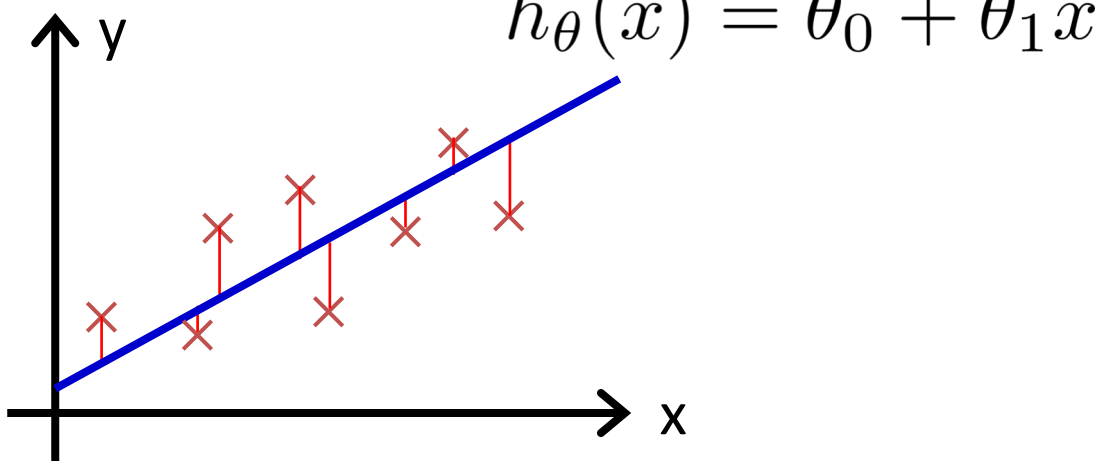
# How to Select the Parameters

**Training Set**: $\{(x^{(1)},y^{(1)}), (x^{(2)},y^{(2)}), \ldots, (x^{(m)},y^{(m)})\}$

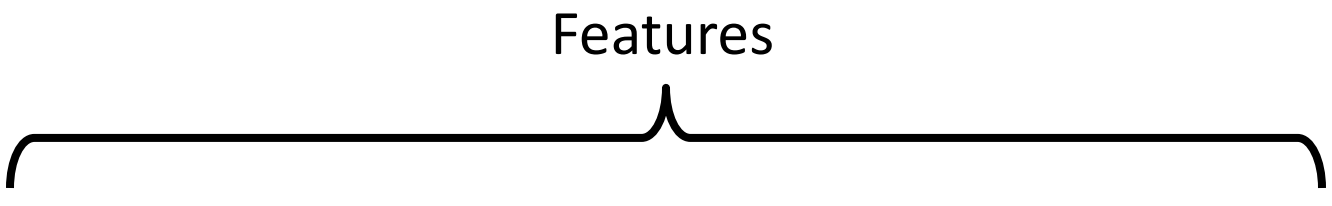
**Cost Function**: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

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

$h_\theta(x) = \theta_0 + \theta_1 x$

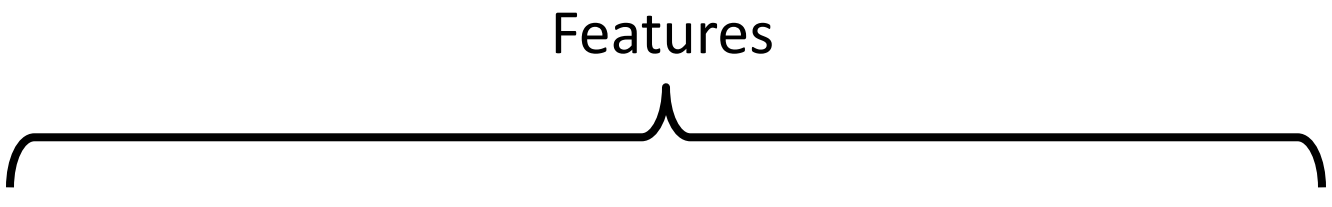


# Linear Regression with Multiple Features




| Size in feet$^2$ ($x_1$) | Number of Bedrooms ($x_2$) | Age of Home ($x_3$) | Price (y) |
|---|---|---|---|
| 1000 | 1 | 21 | 410K |
| 1200 | 2 | 34 | 600K |
| 1230 | 2 | 42 | 620K |
| 1340 | 3 | 17 | 645K |
| … | … | … | … |

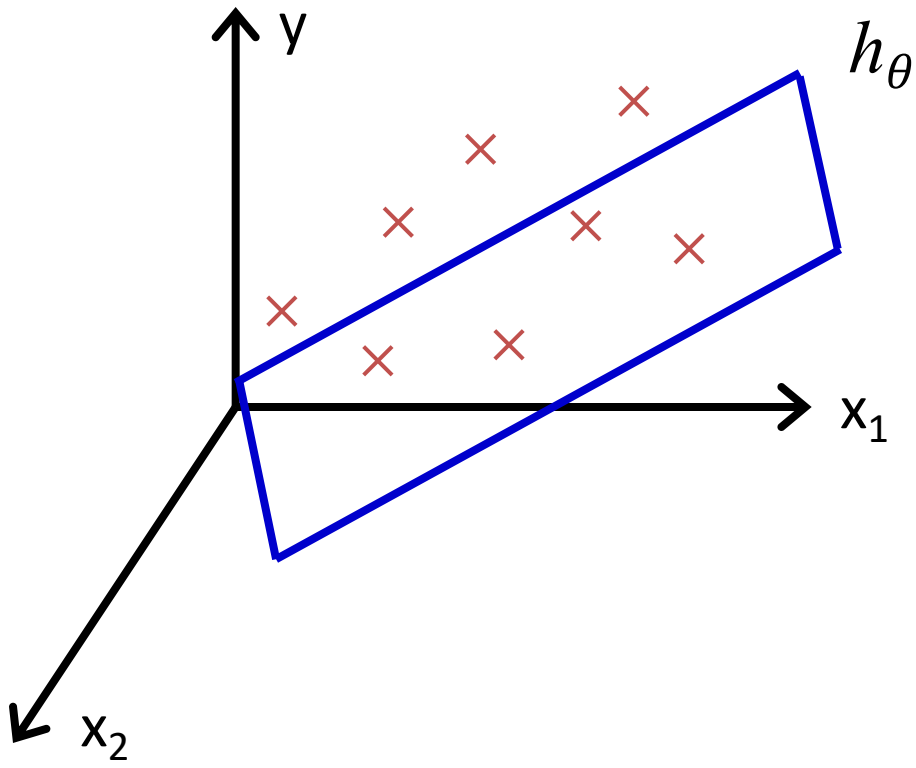
# Linear Regression with Multiple Features




| Size in feet$^2$ ($x_1$) | Number of Bedrooms ($x_2$) | Age of Home ($x_3$) | Price (y) |
|---|---|---|---|
| 1000 | 1 | 21 | 410K |
| 1200 | 2 | 34 | 600K |
| 1230 | 2 | 42 | 620K |
| 1340 | 3 | 17 | 645K |
| … | … | … | … |

$$h_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

# Example: Two Features




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

# Linear Regression with Multiple Features

| Size in feet$^2$ ($x_1$) | Number of Bedrooms ($x_2$) | Age of Home ($x_3$) | Price (y) |
|---|---|---|---|
| 1000 | 1 | 21 | 410K |
| 1200 | 2 | 34 | 600K |
| 1230 | 2 | 42 | 620K |
| 1340 | 3 | 17 | 645K |
| … | … | … | … |

**Notation:**

$n$ = number of features

$m$ = number of training samples

$\boldsymbol{x}^{(i)}$ = feature vector for $i^{th}$ training samples (the entire $i^{th}$ row).

$x_j^{(i)}$ = value of feature $j$ in $i^{th}$ training samples (element $j$ in $i^{th}$ row) .

# Linear Regression with Multiple Features

$$h_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

For simplicity of notation, we define $x_0 = 1$ (add a dummy feature with a constant value for all data samples). Then, we can define:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad ; \quad \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \Rightarrow \quad h_\theta(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$$

# Gradient Descent for Multiple Variables*

- Regression Model:

$$h_\theta(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

- Parameter vector & feature vectors:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

- **Cost function:**

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

* Reference: Andrew Ng, Machine Learning, Stanford University.

# Gradient Descent for Multiple Variables

- Hypothesis: $h_\theta(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$
- Cost function: $J(\boldsymbol{\theta}) = J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})^2$
- **Gradient descent:**

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

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

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

# Gradient Descent for Multiple Features

Previously (n=1):

Repeat $\{$

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

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

(simultaneously update $\theta_0, \theta_1$)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

(simultaneously update $\theta_j$ for $j = 0, \ldots, n$)

$\}$

---

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

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

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

...

# Implementing Gradient Descent in Practice

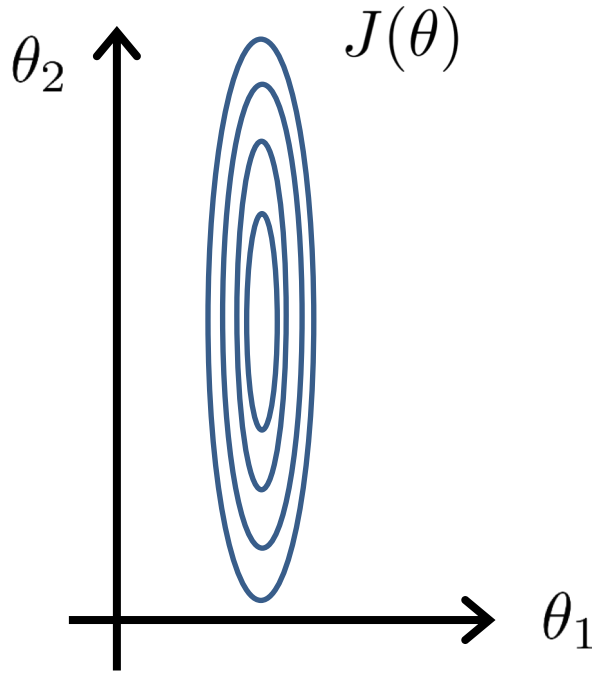
- Make sure to **Normalize** data samples to have all features on a similar scale.

E.g. $x_1$ = size (0-4000 sqr feet$^2$).

$x_2$ = number of bedrooms (1-5).

=> cost contour will be in the form of very tall and skinny ellipses.

=> gradient descent will have a very hard time to find the minimum (convergence will be very slow).



# Example

- Predicting value of a house based on **size of the house** and **number of bedrooms**!

Feature1 = size (0-4000 sqr feet$^2$).

Feature2 = number of bedrooms (1-5).

**An easy way to scale features:**

**F1_Scaled = size / max(size)**

**F2_Scaled = number of bedrooms / max(number of bedrooms )**

**(0 <= F1_Scaled <= 1)**

**(0 <= F2_Scaled <= 1)**

# Example

- Predicting value of a house based on **size of the house** and **number of bedrooms**!

- Before Scaling:

| | Number of Bedrooms | Size (sqr footage) |
|---|---|---|
| 1 | 1 | 1300 |
| 2 | 3 | 2400 |
| 3 | 3 | 2270 |
| 4 | 2 | 1450 |
| 5 | 2 | 1400 |
| 6 | 4 | 2900 |

# Example

- Predicting value of a house based on **size of the house** and **number of bedrooms**!

- After Scaling:

| | Number of Bedrooms | Size (sqr footage) |
|---|---|---|
| 1 | 1/4 | 1300/2900 |
| 2 | 3/4 | 2400/2900 |
| 3 | 3/4 | 2270/2900 |
| 4 | 2/4 | 1450/2900 |
| 5 | 2/4 | 1400/2900 |
| 6 | 4/4 | 2900/2900 |

# Implementing Gradient Descent in Practice

- Make sure to **Normalize** data samples to have all features on a similar scale.

E.g. $x_1$ = size (0-3000 sqr feet$^2$).

$x_2$ = number of bedrooms (1-5).

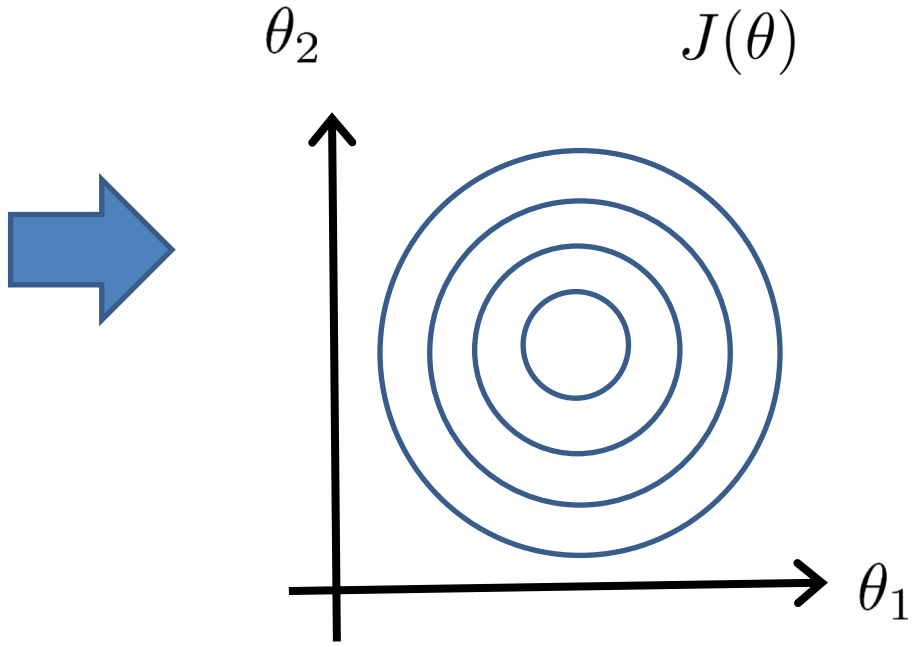
**An easy way to scale features:**

**F1_Scaled = size / max(size)**

**F2_Scaled = number of bedrooms / max(number of bedrooms )**

**(0 <= F1_Scaled <= 1)**

**(0 <= F2_Scaled <= 1)**



# Implementing Gradient Descent in Practice

- A more general approach for normalization is to make each feature **zero mean** with **unit standard deviation** over all data samples (for each feature column). In other word, we should normalize **each column of the feature table**.
- Compute the mean and standard deviation for each feature ($x_{nd}$ is the $d$th feature of $n$th data sample):

$$mean(x_d) = \bar{x}_d = \frac{1}{N}\sum_{n=1}^{N} x_{nd} \qquad std(x_d) = s_d = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_{nd} - \bar{x}_d)^2}$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \left(\frac{x_{nd} - \bar{x}_d}{s_d}\right)$$

# Logistic Regression Classifier

# Review

- **Classification**: Predict a discrete valued output for each item.
  - Labels are discrete (categorical)
  - Labels can be binary (e.g., rainy/sunny, spam/non-spam,) or non-binary (e.g., rainy/sunny/cloudy, Setosa/Versicolor/Virginica)

- **Regression:** Predict a continuous valued output for each item.
  - Labels are continuous (numeric), e.g., stock price, housing price
  - Can define 'closeness' when comparing prediction with true values

# Deriving a Classifier from a Regression Model

1. We learned how to build a linear regression model to predict continuous-valued outputs.

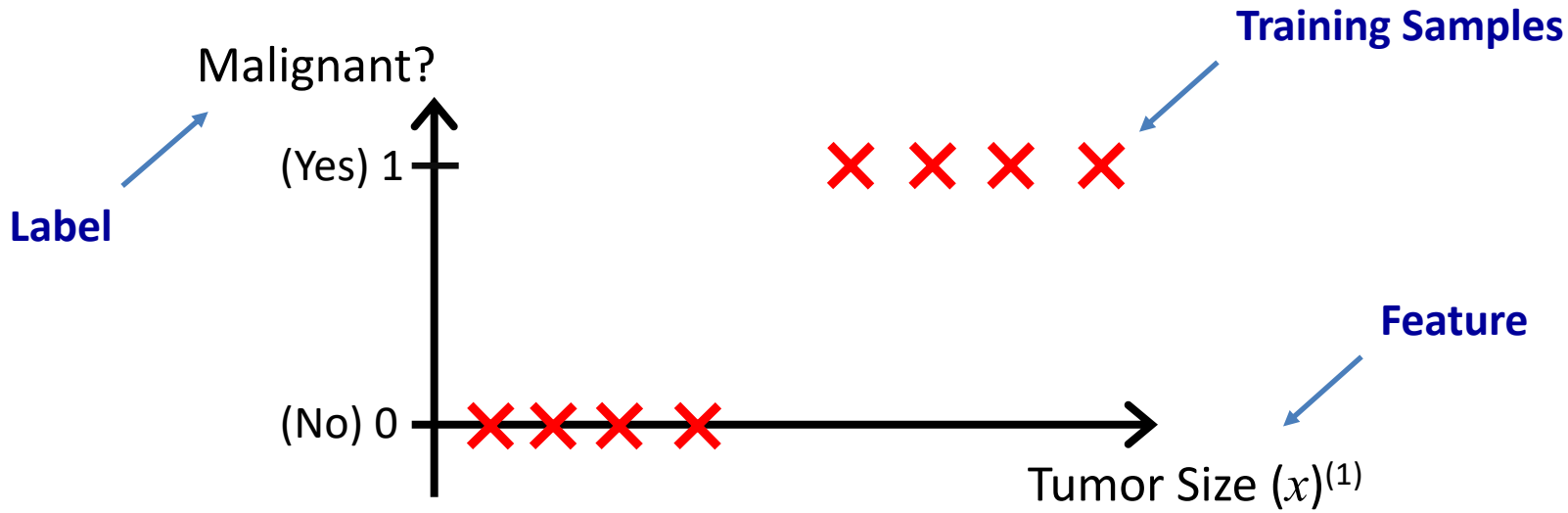
2. Well, in theory we can convert Categorical Labels into Numerical Labels:
   - Sunny → 0 , Rainy → 1  thus: Sunny/Rainy → 0/1
   - Setosa → 0 , Versicolor → 1 , Virginica → 2

3. Thus, Let's take advantage of the linear regression model to develop a Classifier! (We will see that discretizing the output of a linear regression is not always a good idea! So, we need to define new hypothesis model and new cost function)

4. Since it is a classifier built based on linear regression algorithm, we call it: "Logistic Regression"!

# Logistic Regression Classifier with One Feature

**Example**: Predicting if a Tumor is Malignant or Not based on tumor size (so, we need a classifier):

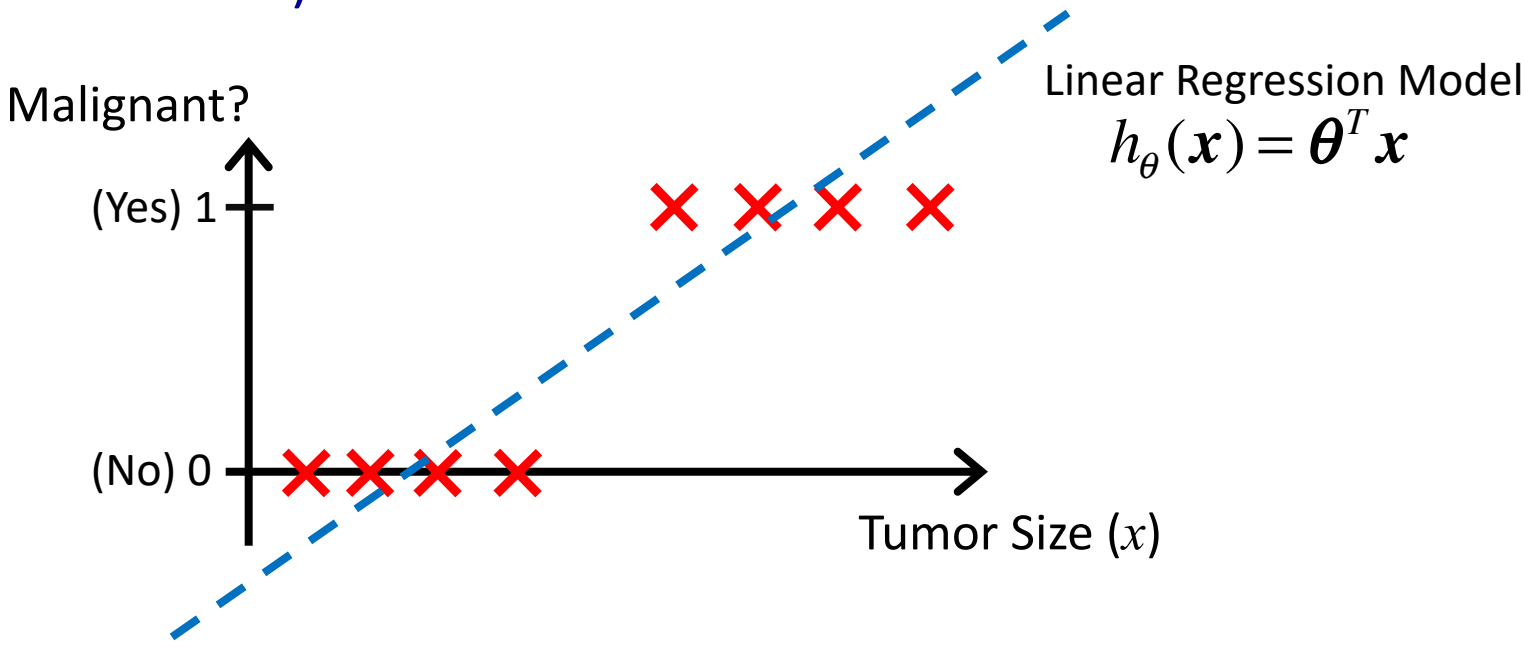



**This is a discrete-valued data. Nonetheless, let's apply our Linear Regression model on it!**

* Example from Andrew Ng, Machine Learning, Stanford University.

# Logistic Regression Classifier with One Feature

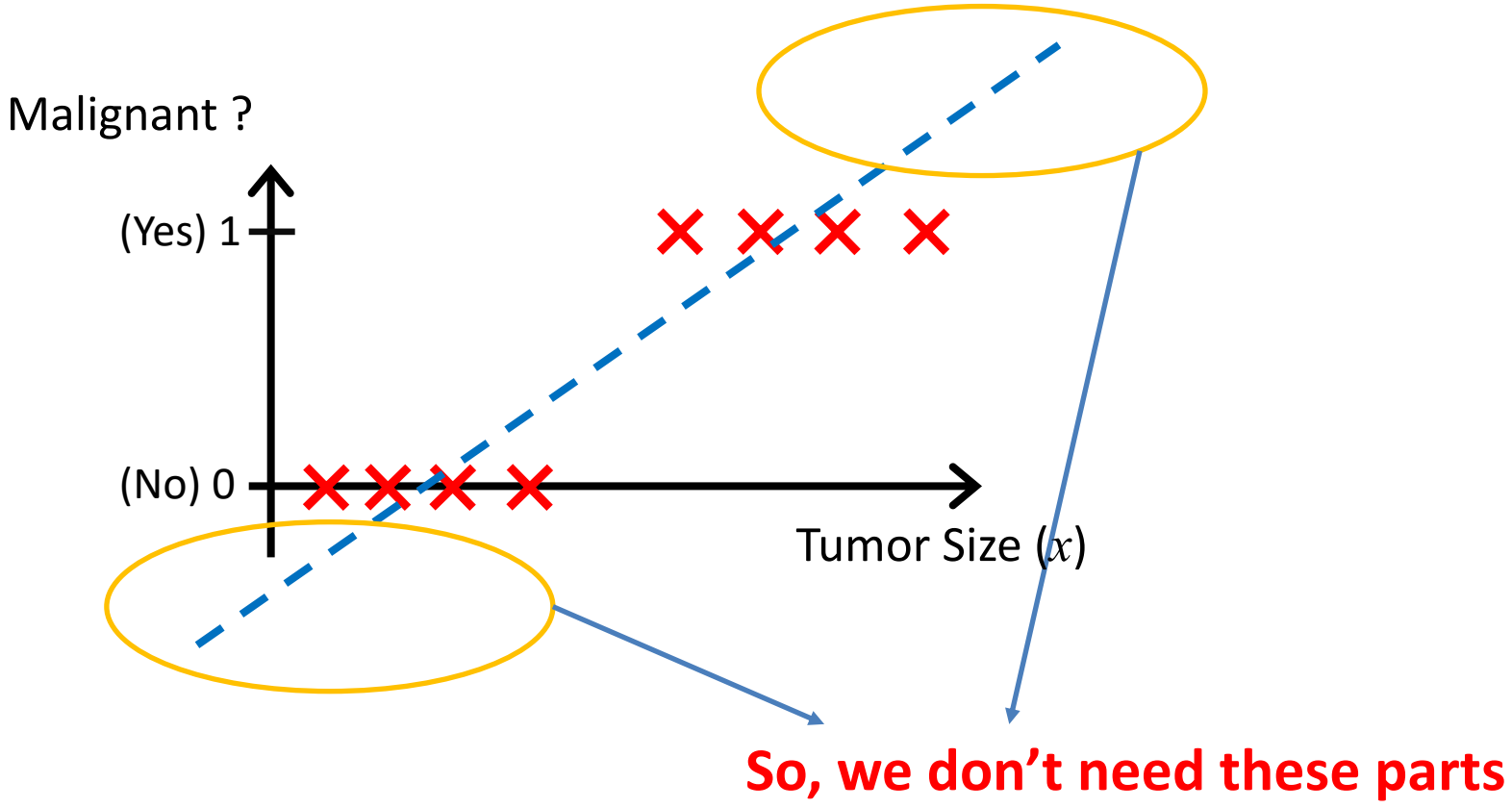
**Example**: Predicting if a Tumor is Malignant or Not based on tumor size (so, we need a classifier):




**This is a discrete-valued data. Nonetheless, let's apply our Linear Regression model on it!**

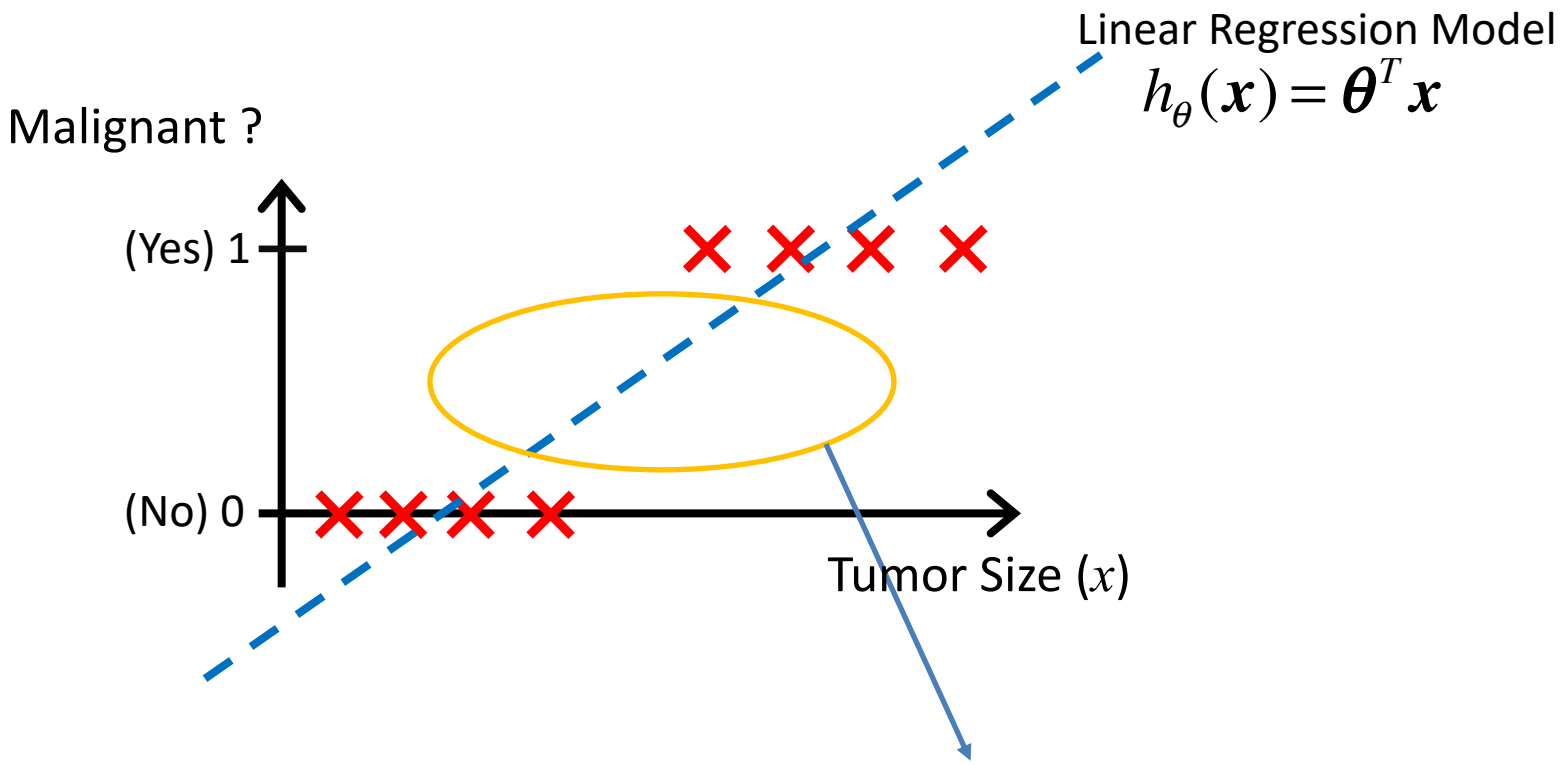
# Logistic Regression Classifier with One Feature

- we know that for our classifier, the **output should be either 1 or 0**!



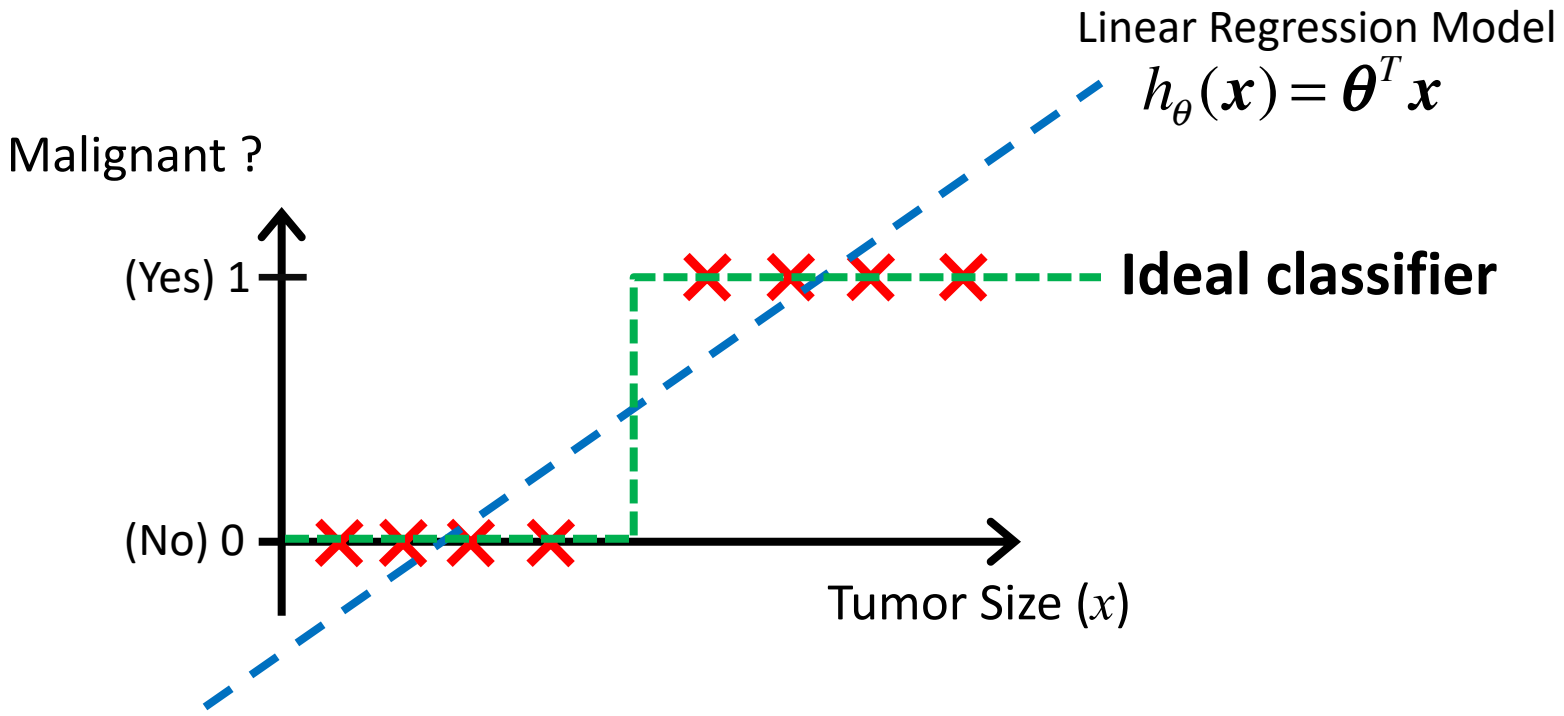
# Logistic Regression Classifier with One Feature

- we know that for our classifier, the **output should be either 1 or 0**!




**So, We also need a sharp transition here**

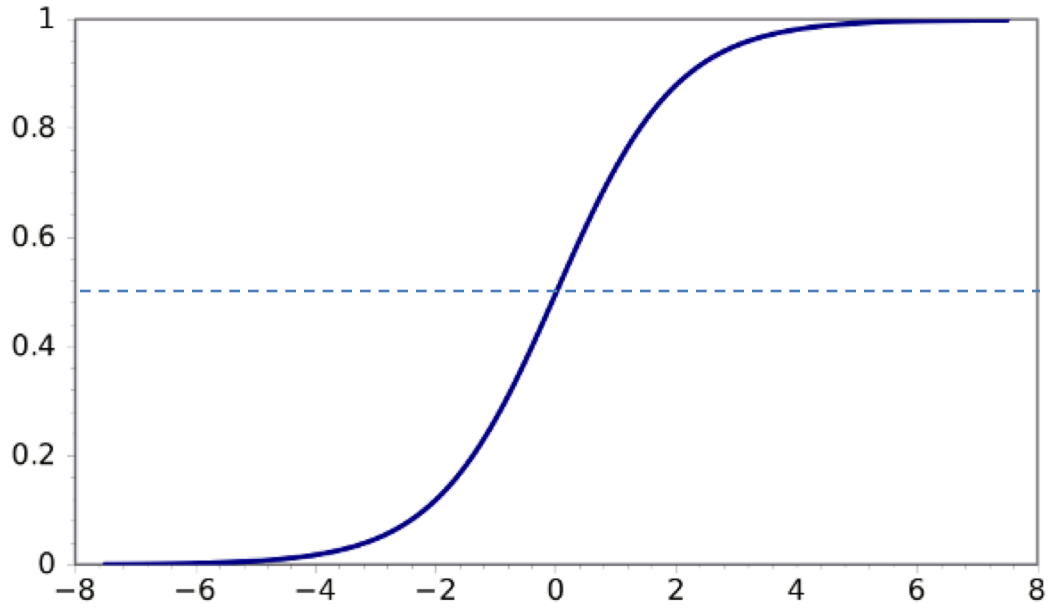
# Logistic Regression Classifier with One Feature



# Sigmoid Function

- Sigmoid Function (Logistic Function):

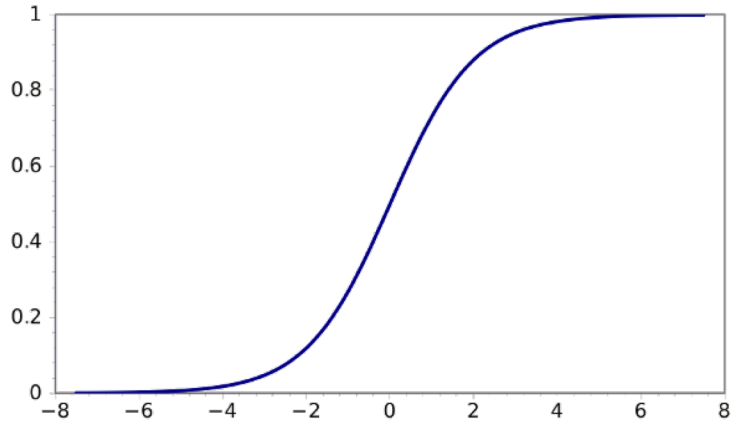
$$g(z) = \frac{1}{1 + e^{-z}}$$



# Logistic Regression Model

- Sigmoid Function (Logistic Function):

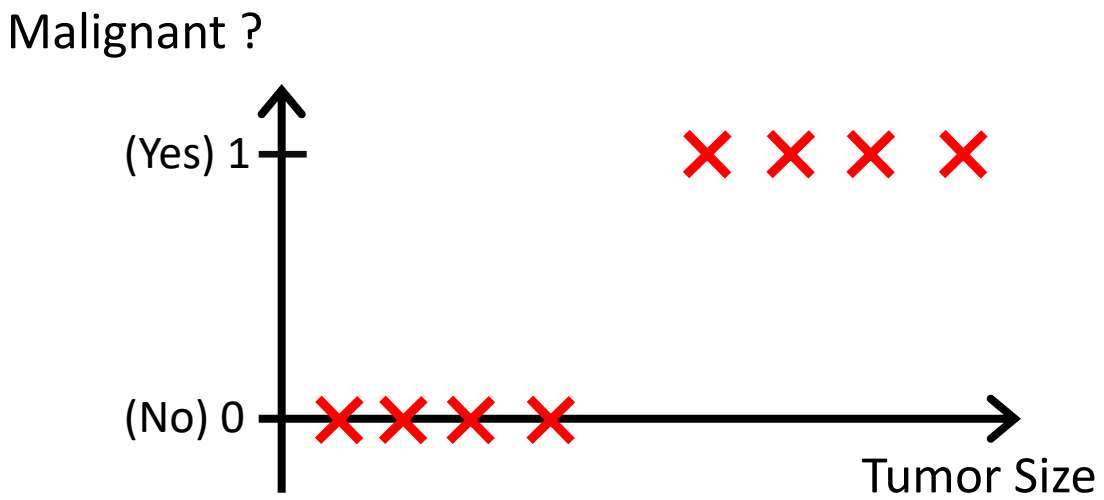
$$g(z) = \frac{1}{1 + e^{-z}}$$




**New approach for output prediction:**

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

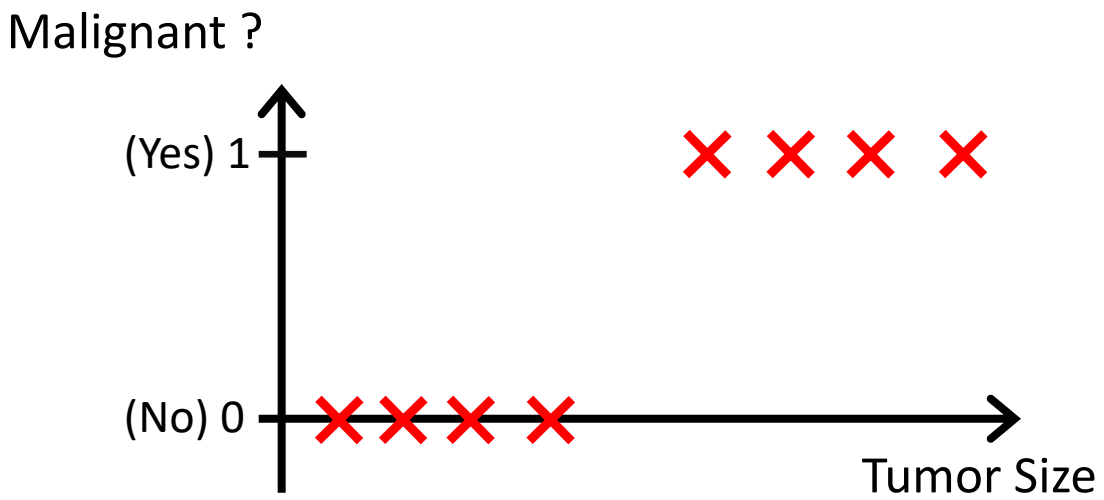
**So, Now the $h_\theta(x)$ is limited to the range of [0,1].**




- New approach: $h_\theta(\boldsymbol{x}) = g(\boldsymbol{\theta}^T \boldsymbol{x}) = \dfrac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$

- After applying the Sigmoid function, $h_\theta(\boldsymbol{x})$ will be limited in the range of [0,1]. But, still can take any value in this range!

- **So, it is like representing the Probability of happening each label!**
  - **E.g. 30% chance of rain, 5% chance of malignant cancer, …**




- New approach: $h_\theta(\boldsymbol{x}) = g(\boldsymbol{\theta}^T \boldsymbol{x}) = \dfrac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$

- After applying the Sigmoid function, $h_\theta(\boldsymbol{x})$ will be limited to the range of [0,1]. But, still can take any value in this range!
- **Now, to generate class output (binary output), we can compare the results with a threshold (e.g. 0.5) to discretize the output:**

$$\begin{cases} \text{predict } \text{“}y = 1\text{“} \text{ if } h_\theta(\boldsymbol{x}) \geq 0.5 \\ \text{predict } \text{“}y = 0\text{“} \text{ if } h_\theta(\boldsymbol{x}) \leq 0.5 \end{cases}$$

# How to Select the Parameters?

Training set (m training samples):

$$\left\{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)})\right\}$$

Feature Vector: $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ $x_0 = 1, y \in \{0, 1\}$

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

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

**How to choose parameters $\theta$?**

# Gradient Descent for Logistic Regression

$$h_\theta(\boldsymbol{x}) = g(\boldsymbol{\theta}^T \boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

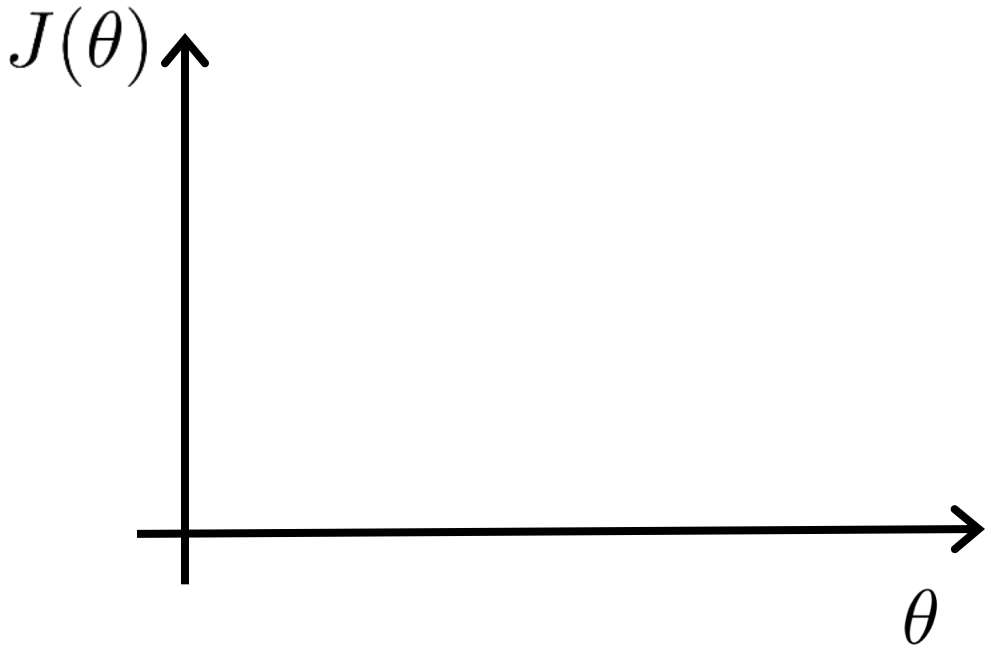
- **Previous definition of Cost function for linear regression:**

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

# Logistic Regression Cost Function

- Previous definition of Cost function for linear regression:

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




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

**This Cost Function is not Convex for Logistic Regression (It has many local minimums)!**

- Since the Cost Function with previous definition **is not convex** for logistic regression, there is **no guarantee** for gradient descent to find the **global minimum** for error. Thus, we take advantage of a "**log**" function to define an alternative **convex** cost function (The mathematical details why *log* is a good option is beyond the scope of the class! But, an intuitive idea is available in next page!).

- **New Cost function:**

$$J(\boldsymbol{\theta}) = J(\theta_0, \theta_1, \ldots, \theta_n)$$

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

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

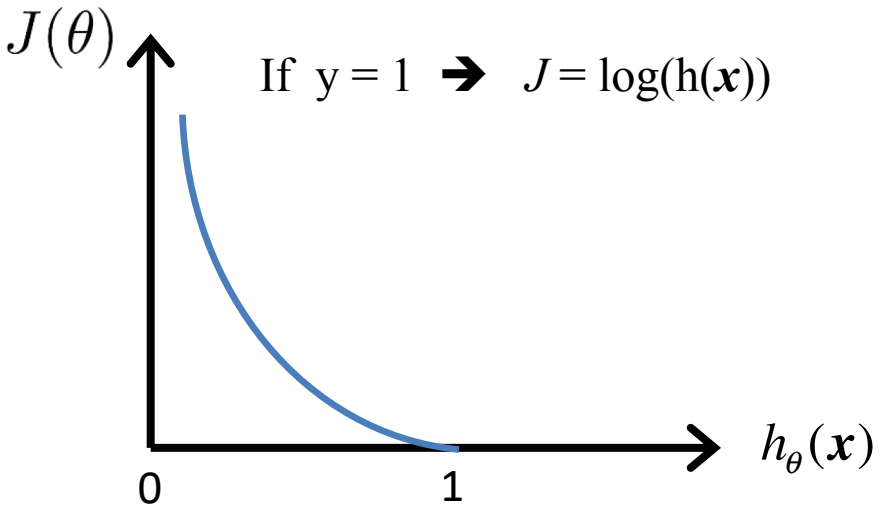
# Optional

- **Why does this Cost function work?**

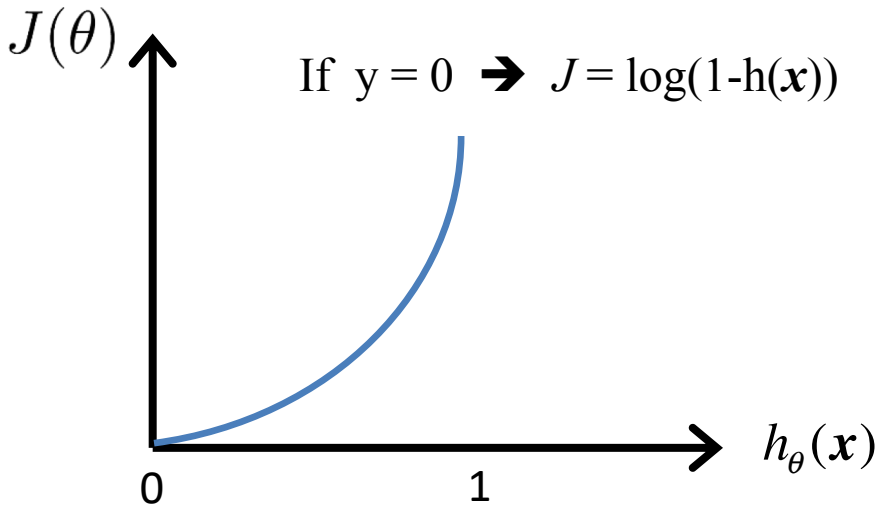
$$J(\boldsymbol{\theta}) = J(\theta_0, \theta_1, \ldots, \theta_n)$$

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

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




- when y=1, and h(x) is close to 1, the cost will be very small, and when y=1, and h(x) is close to 0, the cost will be too high.




- when y=0, and h(x) is close to 0, the cost will be very small, and when y=0, and h(x) is close to 1, the cost will be too high.

# Logistic Regression Cost Function

Cost Function:

$$J(\boldsymbol{\theta}) = J(\theta_0, \theta_1, \ldots, \theta_n)$$

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

To found the best parameters $\theta$ :

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

# Gradient Descent for Multiple Variables

- Output (Probability): $h_\theta(\boldsymbol{x}) = \dfrac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$
- Want $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ to find $\boldsymbol{\theta}$
- Cost function: $J(\boldsymbol{\theta}) = -\dfrac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log h_\theta(\boldsymbol{x}^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(\boldsymbol{x}^{(i)}))\right]$
- **Gradient descent:**

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

$\}$

(simultaneously update for every $j = 0, \ldots, n$ )

# Gradient Descent for Multiple Variables

- Output (probability): $h_\theta(\boldsymbol{x}) = \dfrac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$
- Cost function: $J(\boldsymbol{\theta}) = -\dfrac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log h_\theta(\boldsymbol{x}^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(\boldsymbol{x}^{(i)}))\right]$
- **Gradient descent:**

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

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

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

- **Algorithm looks identical to linear regression (except for the definition of $h(x)$ )!!!**

# Gradient Descent for Multiple Variables

- Output (probability): $h_\theta(\boldsymbol{x}) = \dfrac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$
- Cost function: $J(\boldsymbol{\theta}) = -\dfrac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log h_\theta(\boldsymbol{x}^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(\boldsymbol{x}^{(i)}))\right]$
- **Gradient descent:**

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

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

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

- **Algorithm looks identical to linear regression (except for the definition of $h(x)$ )!!!**

*Thank You!*

**Questions?**