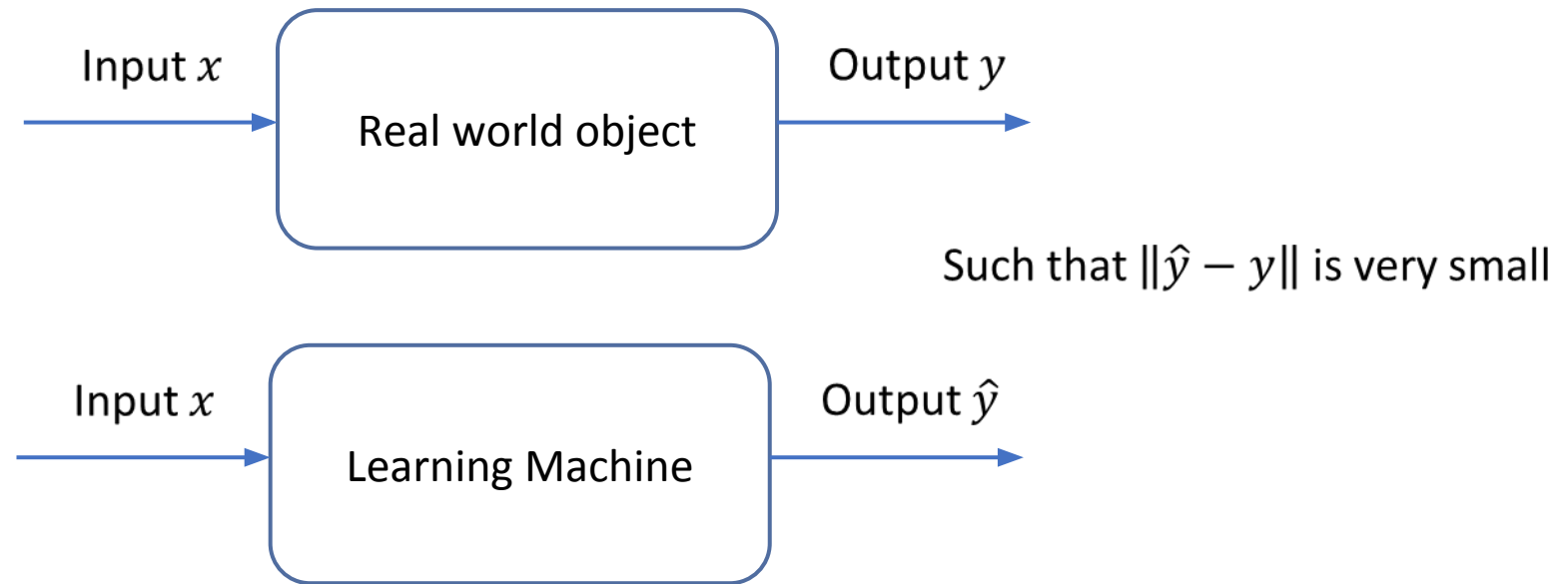


# Introduction to Machine Learning

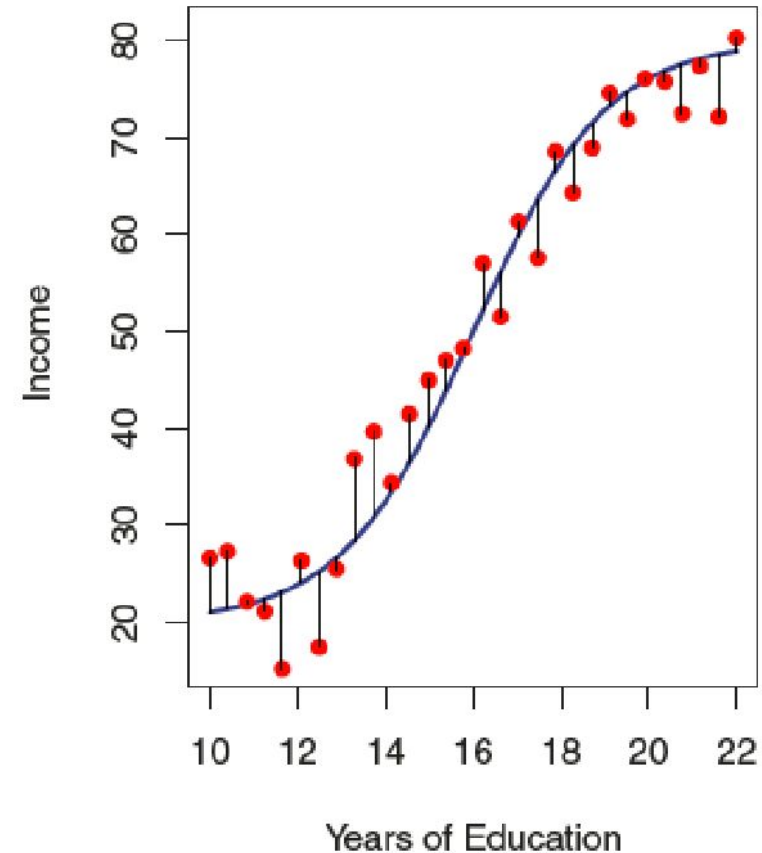
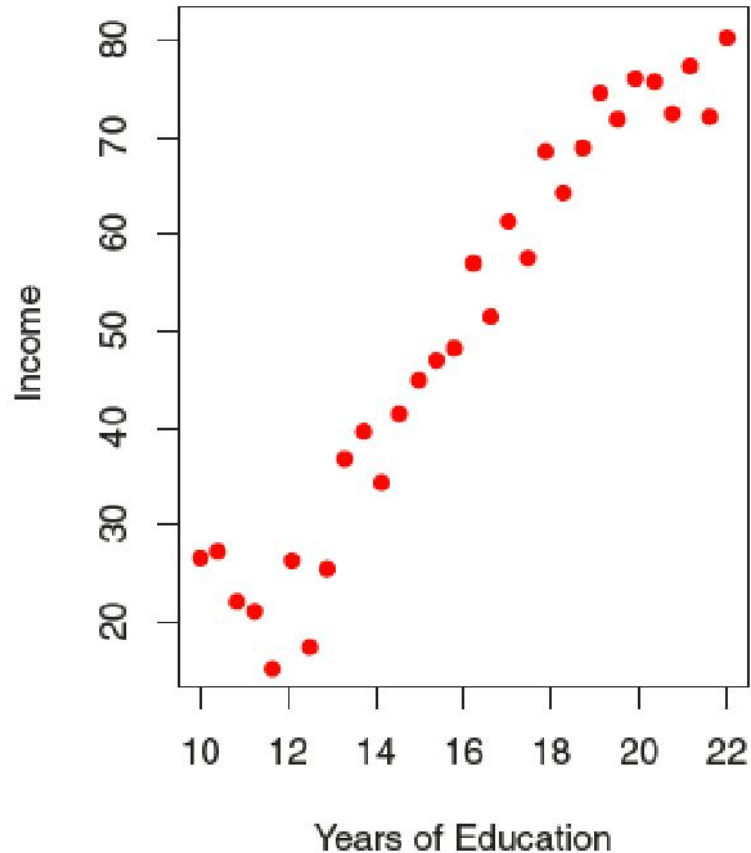
Dai Bui

# What is Machine Learning?



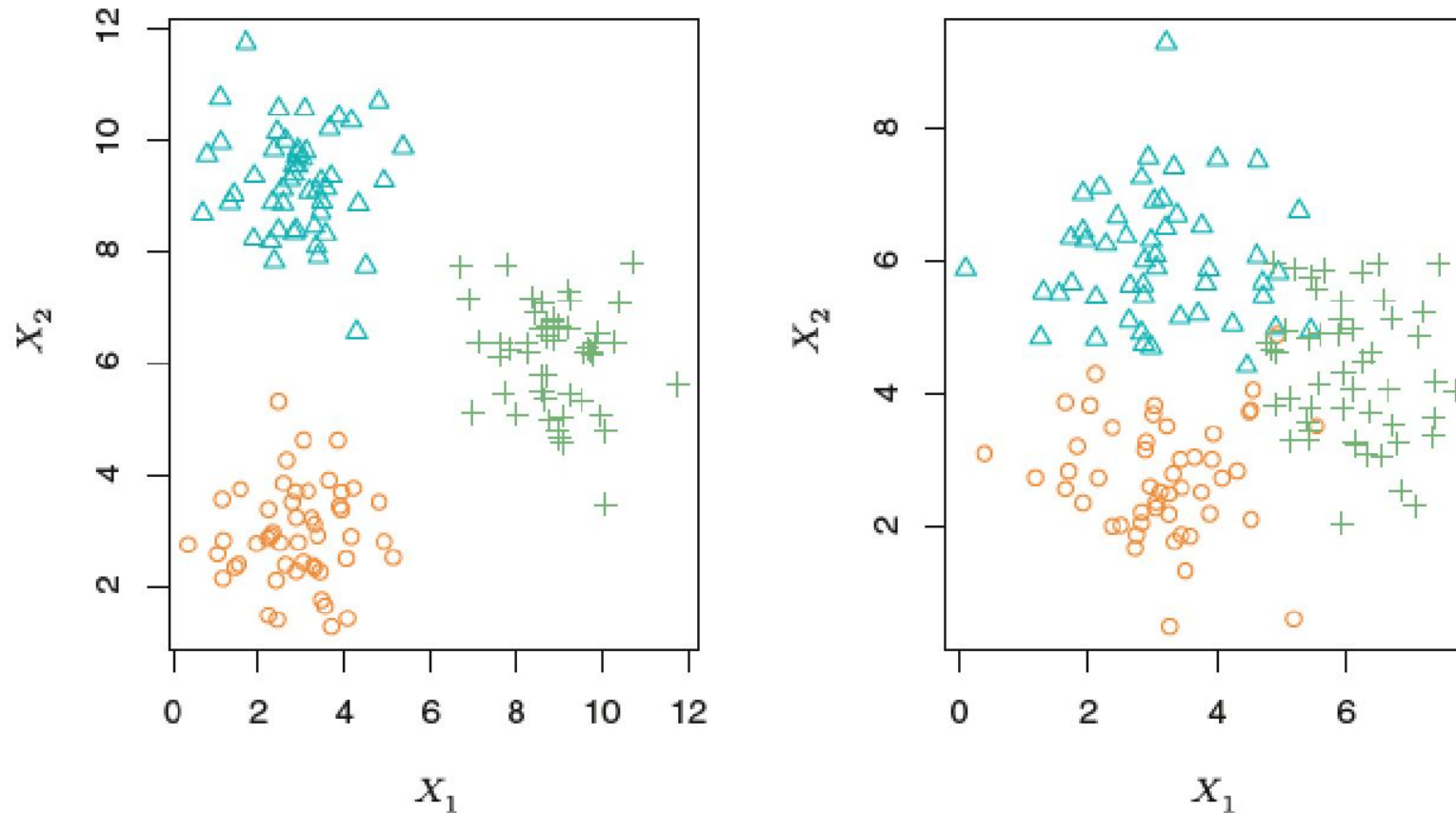
This is called **supervised learning**  
The Machine learns the internal  
(invisible) states of the real world  
object through the observation of  
the behaviors of the object

# What is Machine Learning?



Why do we need to find the function?

# What is Machine Learning?

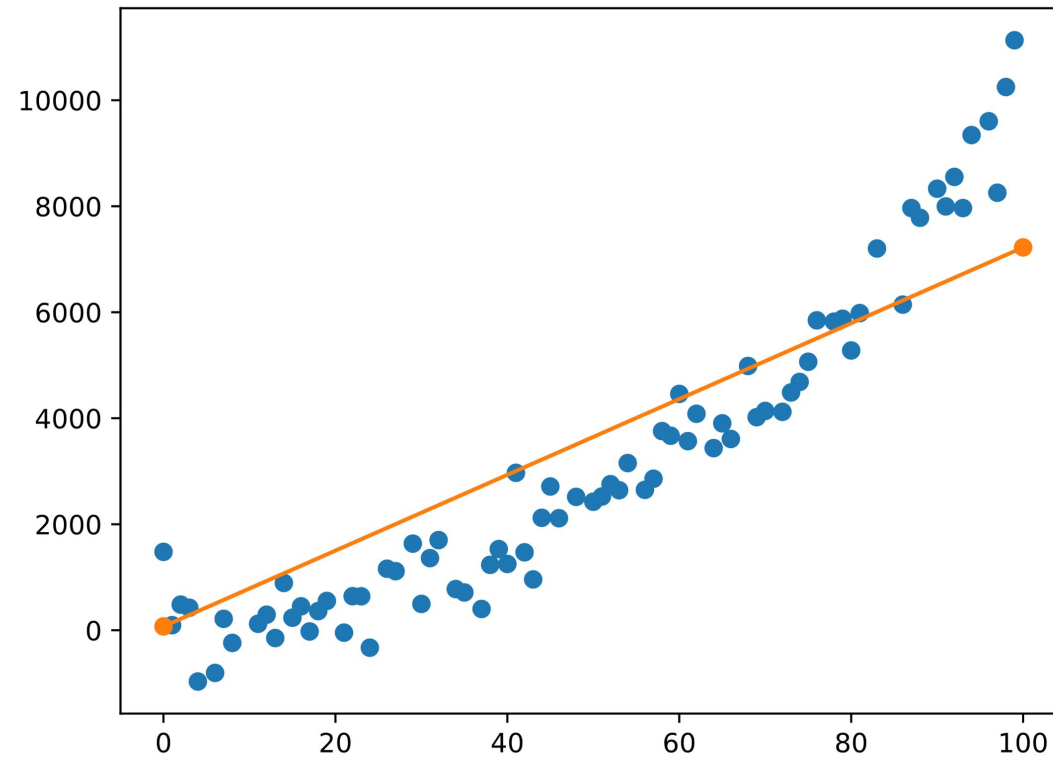


No previous outcome is given. The machine needs to learn the structure by themselves: **Unsupervised learning**

# Machine Learning Phases

- It is composed of two phases
  - Training
  - Validation
- Training
  - Inputs are fed into the learning machine
  - The parameters of the learning machine are tune so that the predicted outputs are close to the target outputs
  - Similar to homework practicing for students
- Validation
  - Some of the data samples are retained and not used during training
  - Use the retained samples to test if the learned machine can predict close to the target values of the retained samples when fed with the retained input samples
  - Can be thought as exams for students

# Linear Regression



# Linear Regression

- Suppose that we have the following linear hypothesis function:

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

suppose that  $x_0 = 1$ , we can rewrite the function as:

$$h_{\theta}(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x$$

$$\text{with } x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \text{ and } \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

# Loss Function

- Suppose that we have  $m$  samples  $x^j$  with  $j = \{1 \rightarrow m\}$
- Now we compute the predicted  $h_\theta$  value for each  $x^j$
- We want this value  $\sum_{j=1}^m (h_\theta(x^j) - y^j)^2$  to be minimized
  - What can we change so the value becomes minimized?
- Because  $x^j$  and  $y^j$  are collected data, so they are **given fact**. As a result, they cannot be changed
  - We can only change  $\theta$



# Loss Function

- We define the following **loss** function as a function of  $\theta$ :

$$L(\theta) = \sum_{j=1}^m (h_{\theta}(x^j) - y^j)^2$$

- We will find  $\theta^*$  such that  $L(\theta^*)$  is smallest:

$$\theta^* = \arg \min_{\theta} L(\theta)$$

- How do we find such  $\theta^*$ ?

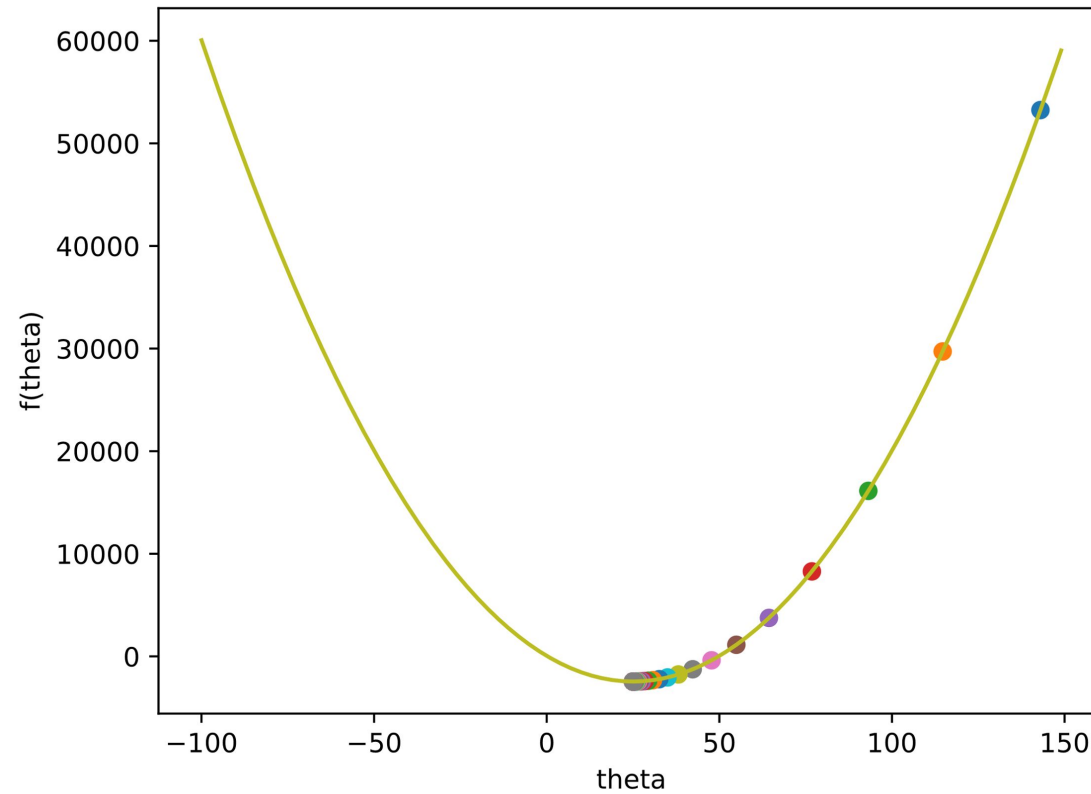
# Gradient Descent

- Let us take an example: Find  $\theta$  such that  $f(\theta) = 4\theta^2 - 200\theta + 50$  is smallest:

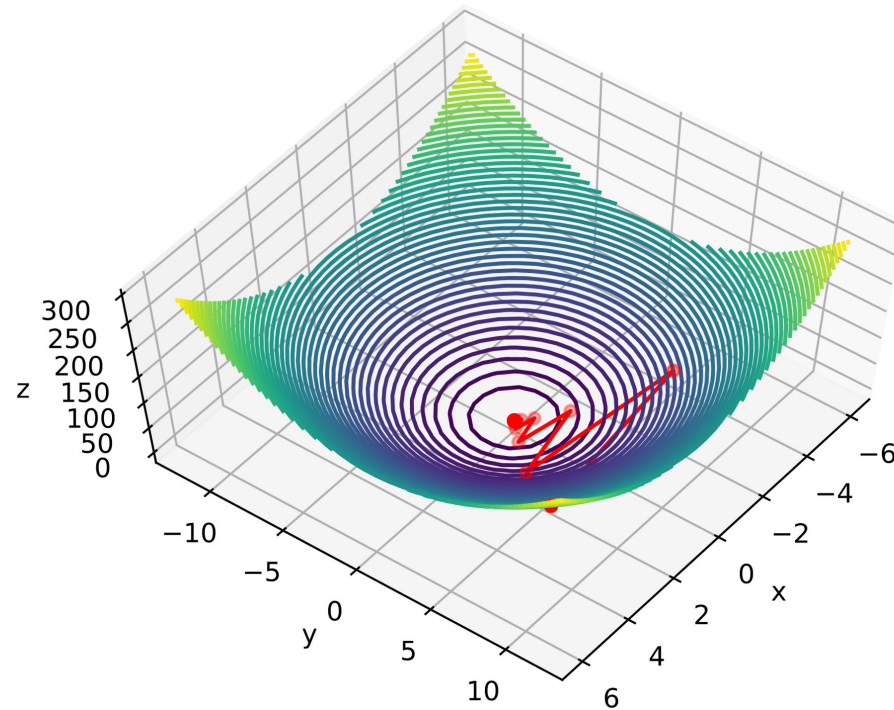
$$\begin{aligned}\nabla f(\theta) &= 8\theta - 200 \\ \theta &= \theta - \alpha * \nabla f(\theta)\end{aligned}$$

- $\alpha$  is called the **learning rate**
- When do we stop?

# Gradient Descent Coding



# Multivariate Gradient Descent



# Gradient Descent for Linear Regression Loss Function

- $$\nabla L(\theta) = \frac{\partial \sum_{j=1}^m (h_{\theta}(x^j) - y^j)^2}{\partial \theta}$$
- Because this is a function of  $\theta$ , we have:
$$\nabla L(\theta) = \sum_{j=1}^m \frac{\partial (h_{\theta}(x^j) - y^j)^2}{\partial \theta}$$
- According to the matrix derivation we have:

$$\frac{\partial (h_{\theta}(x^j) - y^j)^2}{\partial \theta} = \begin{bmatrix} \frac{\partial (h_{\theta}(x^j) - y^j)^2}{\partial \theta_0} \\ \frac{\partial (h_{\theta}(x^j) - y^j)^2}{\partial \theta_1} \\ \vdots \\ \frac{\partial (h_{\theta}(x^j) - y^j)^2}{\partial \theta_n} \end{bmatrix}$$

# Gradient Descent for Linear Regression Loss Function

$$\begin{aligned}\frac{\partial (h_{\theta}(x^j) - y^j)^2}{\partial \theta_i} &= 2 * (h_{\theta}(x^j) - y^j) * \frac{\partial (h_{\theta}(x^j) - y^j)}{\partial \theta_i} \\ &= 2 * (h_{\theta}(x^j) - y^j) * \frac{\partial \sum_{k=0}^n \theta_k x_k^j}{\partial \theta_i} \\ &= 2 * (h_{\theta}(x^j) - y^j) * x_i^j\end{aligned}$$

- So we have:

$$\frac{\partial (h_{\theta}(x^j) - y^j)^2}{\partial \theta} = 2 * \begin{bmatrix} (h_{\theta}(x^j) - y^j) * x_0^j \\ (h_{\theta}(x^j) - y^j) * x_1^j \\ \vdots \\ (h_{\theta}(x^j) - y^j) * x_n^j \end{bmatrix} = 2 * (h_{\theta}(x^j) - y^j) * \begin{bmatrix} x_0^j \\ x_1^j \\ \vdots \\ x_n^j \end{bmatrix}$$

# Gradient Descent for Linear Regression Loss Function

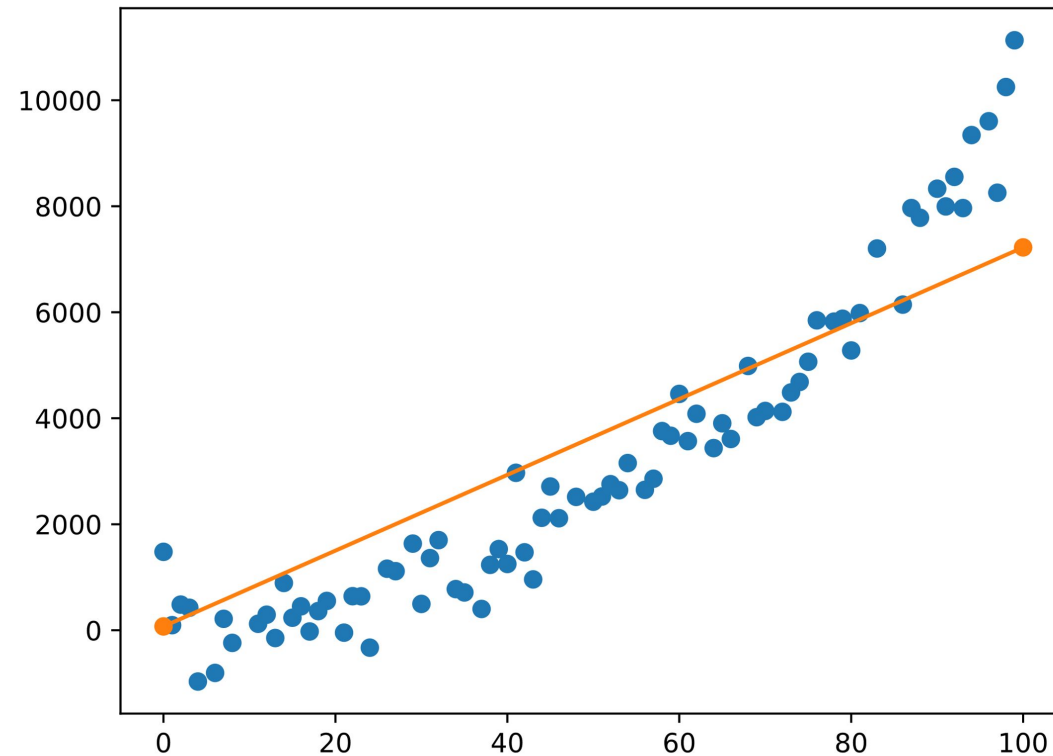
- Finally

$$\nabla L(\theta) = \frac{2}{m} * \sum_{j=1}^m (h_{\theta}(x^j) - y^j) * \begin{bmatrix} x_0^j \\ x_1^j \\ \vdots \\ x_n^j \end{bmatrix}$$

- We then use the same update method:

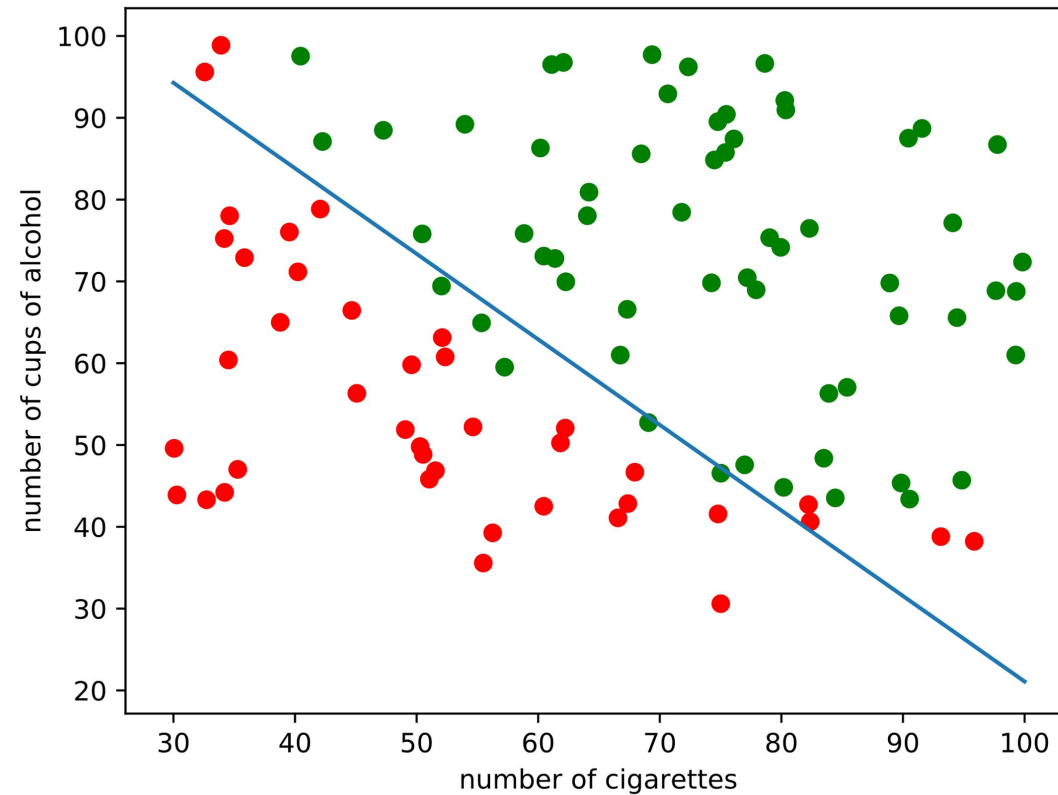
$$\theta = \theta - \alpha \nabla L(\theta)$$

# Linear Regression Coding





# Binary Classification



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

# Logistic Regression

- Now  $y \in \{0,1\}$ , we need to modify hypothesis function  $h_{\theta}(x)$  so that it takes values between 1 and 0
- We choose the following function

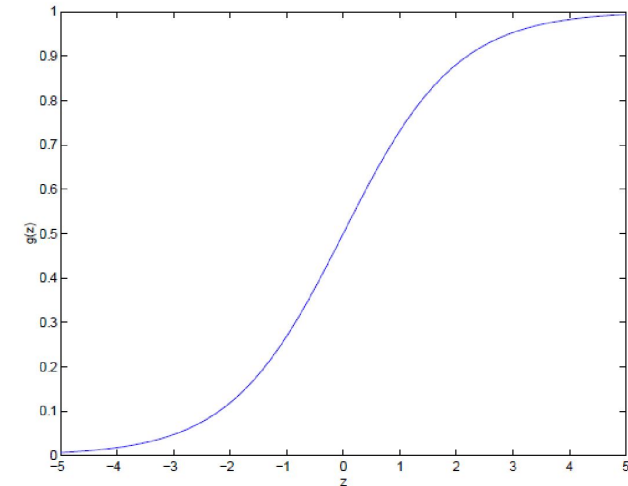
$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

where

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

Is called **logistic function** or **sigmoid function**

- $g(z)$  takes value from 0 to 1



# Logistic Regression

- Because now  $h_{\theta}(x)$  takes value from 0 to 1, we **assume** that it is the probability of  $y$  equal to 1

$$\begin{aligned}P(y = 1|x; \theta) &= h_{\theta}(x) \\P(y = 0|x; \theta) &= 1 - h_{\theta}(x)\end{aligned}$$

equivalently

$$p(y|x; \theta) = (h_{\theta}(x))^y(1 - h_{\theta}(x))^{1-y}$$

- Suppose that we have  $m$  training examples  $\{\vec{x}, \vec{y}\}$  generated independently, we then have the likelihood function:

$$\begin{aligned}L(\theta) &= p(\vec{y}|\vec{x}; \theta) \\&= \prod_{j=1}^m p(y^j|x^j; \theta) \\&= \prod_{j=1}^m (h_{\theta}(x^j))^{y^j} (1 - h_{\theta}(x^j))^{1-y^j}\end{aligned}$$

# Logistic Regression

- To avoid roundoff error of multiplication, we take the log of the likelihood function:

$$\begin{aligned}l(\theta) &= L(\theta) \\ &= \sum_{j=1}^m y^j \log h_{\theta}(x^j) + (1 - y^j) \log(1 - h_{\theta}(x^j))\end{aligned}$$

- How do we maximize the likelihood function? We use the same gradient descent method to find  $\theta$ .

# Gradient Descent of Logistic Regression

$$\begin{aligned}
 \frac{\partial l(\theta)}{\partial \theta_i} &= \frac{\partial \sum_{j=1}^m y^j \log h_{\theta}(x^j) + (1 - y^j) \log(1 - h_{\theta}(x^j))}{\partial \theta_i} \\
 &= \sum_{j=1}^m \frac{\partial (y^j \log h_{\theta}(x^j) + (1 - y^j) \log(1 - h_{\theta}(x^j)))}{\partial \theta_i} \\
 &= \sum_{j=1}^m y^j \frac{\partial (\log h_{\theta}(x^j))}{\partial \theta_i} + (1 - y^j) \frac{\partial (\log(1 - h_{\theta}(x^j)))}{\partial \theta_i} \\
 &= \sum_{j=1}^m y^j \frac{1}{h_{\theta}(x^j)} \frac{\partial (h_{\theta}(x^j))}{\partial \theta_i} + (1 - y^j) \frac{1}{1 - h_{\theta}(x^j)} \frac{\partial (1 - h_{\theta}(x^j))}{\partial \theta_i} \\
 &= \sum_{j=1}^m \left( y^j \frac{1}{h_{\theta}(x^j)} - (1 - y^j) \frac{1}{1 - h_{\theta}(x^j)} \right) \frac{\partial (g(\theta x^j))}{\partial \theta_i} \\
 &= \sum_{j=1}^m \left( y^j \frac{1}{h_{\theta}(x^j)} - (1 - y^j) \frac{1}{1 - h_{\theta}(x^j)} \right) \frac{\partial (g(\theta x^j))}{\partial (\theta x^j)} \frac{\partial (\theta x^j)}{\partial \theta_i} \\
 &= \sum_{j=1}^m \left( y^j \frac{1}{h_{\theta}(x^j)} - (1 - y^j) \frac{1}{1 - h_{\theta}(x^j)} \right) \frac{\partial (g(\theta x^j))}{\partial (\theta x^j)} x_i^j
 \end{aligned}$$

# Gradient Descent of Logistic Regression

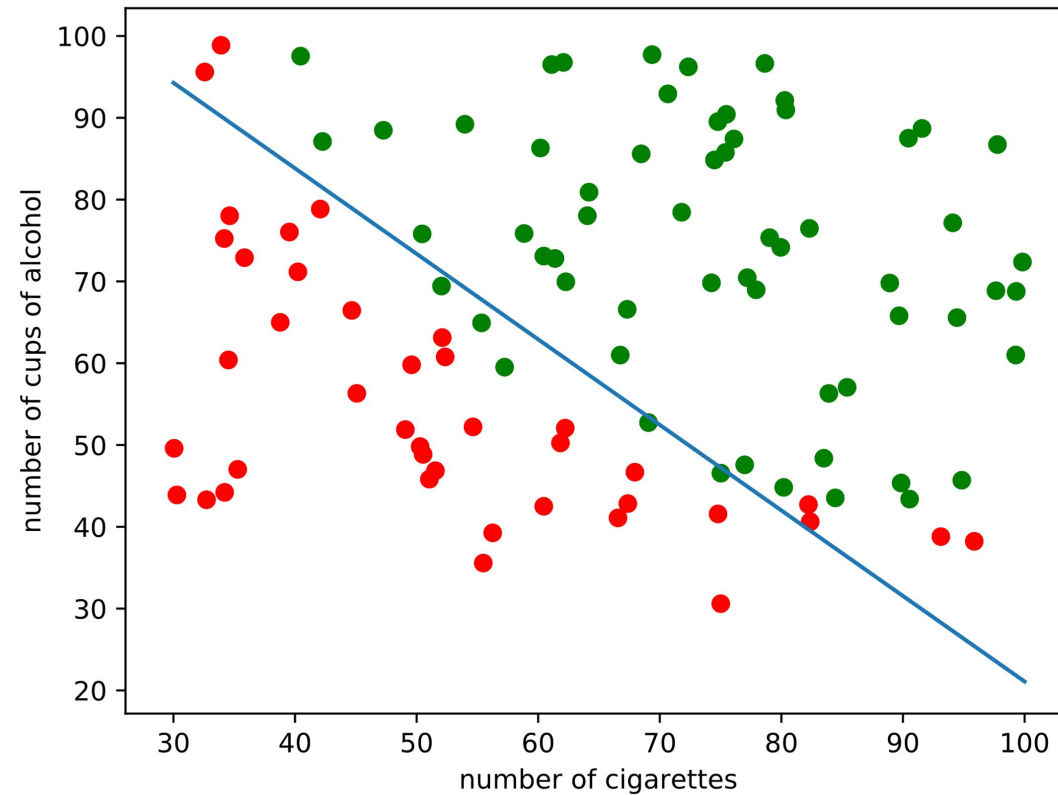
- Note that

$$\begin{aligned}\frac{\partial g(z)}{\partial z} &= \frac{1}{(1 + e^{-z})^2} e^{-z} \\ &= \frac{1}{(1 + e^{-z})} \left( 1 - \frac{1}{(1 + e^{-z})} \right) \\ &= g(z)(1 - g(z))\end{aligned}$$

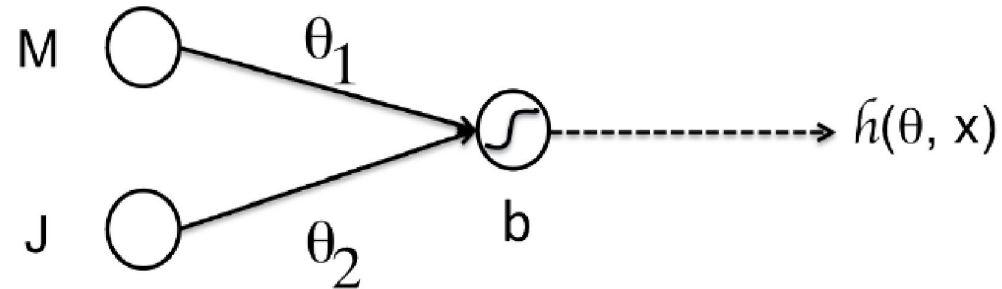
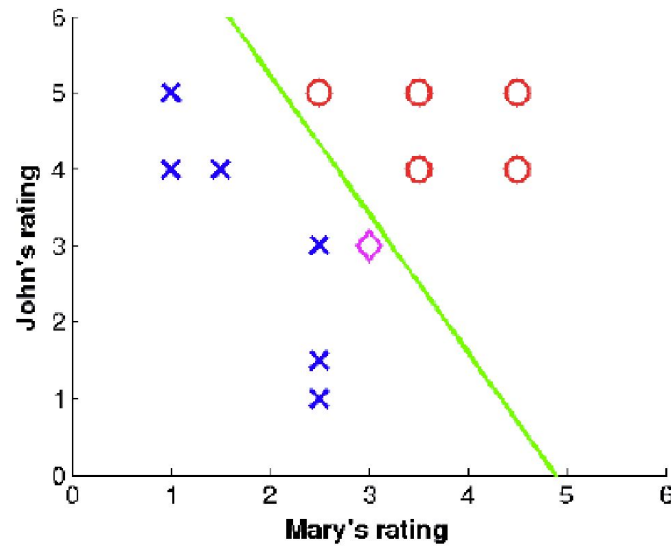
- So we have

$$\begin{aligned}\frac{\partial l(\theta)}{\partial \theta_i} &= \sum_{j=1}^m \left( y^j \frac{1}{h_{\theta}(x^j)} - (1 - y^j) \frac{1}{1 - h_{\theta}(x^j)} \right) (h_{\theta}(x^j)) (1 - h_{\theta}(x^j)) x_i^j \\ &= \sum_{j=1}^m (y^j - h_{\theta}(x^j)) x_i^j\end{aligned}$$

# Logistic Regression Coding

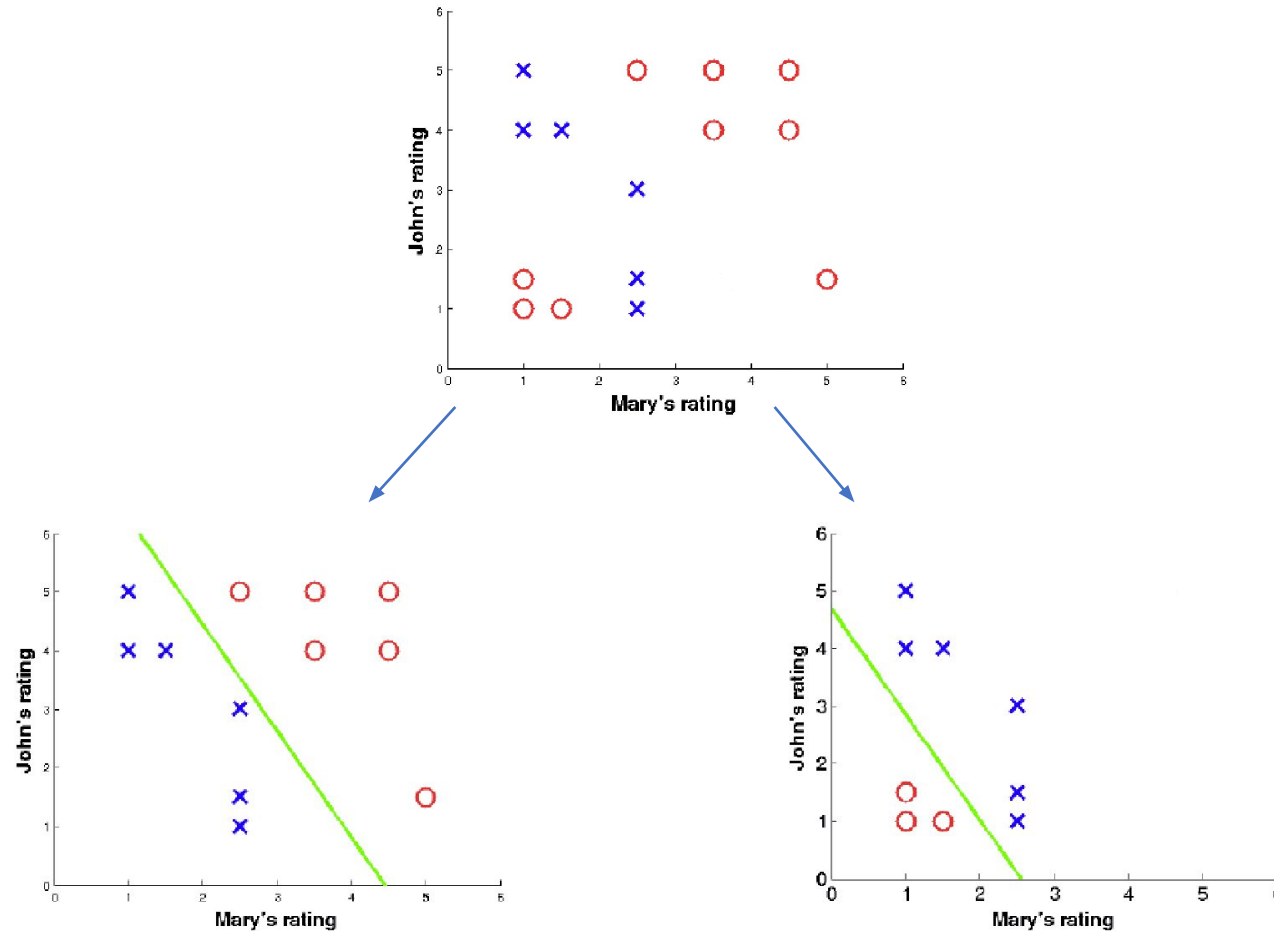


# Logistic Regression

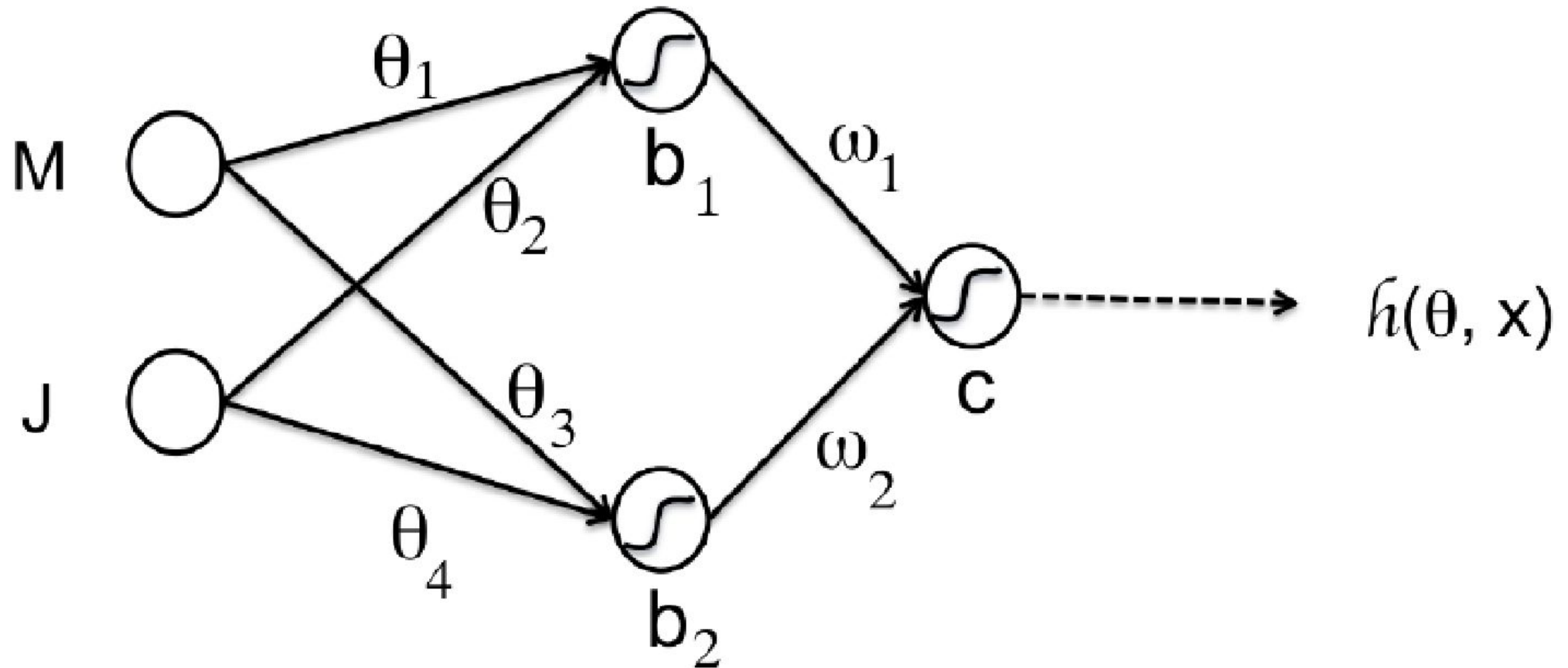




# Limitation of Logistic Regression



# Neural Networks



# Next Class: Neural Networks Basics