[Week-1] Machine Learning

Introduction / Linear Regression with One Variable / Linear Algebra

Joohyung Kang

Contents

Introduction

- **What is Machine Learning? (1/3)**
  - ✓ **Machine Learning Definition**
    - Machine Learning은 ***인공 지능***의 한 분야로, 컴퓨터가 ***학습 (Learning)***할 수 있도록 하는 알고리즘과 기술을 개발하는 분야
    - 즉, 명시적인 명령(Logic)없이도 컴퓨터가 데이터를 ***분석 및 학습***하여 스스로 문제를 해결하도록 하는 것 (ex. Spam mail filter)

❖ **주어진 문제에 대해 데이터를 분석 및 학습하여 컴퓨터 스스로 문제를 해결할 수 있도록 알고리즘과 기술을 개발하는 인공지능 연구분야**

Introduction

- **What is Machine Learning? (2/3)**
  - ✓ **Machine Learning Definition – Tom Mitchell(1998)**
    - **E** ➔ Experience datasets for learning
    - **T** ➔ Task(Problem)
    - **P** ➔ Performance measure of **T**

    ❖ Experience **E**를 사용하여(Learning) Task **T**의 Performance **P**가 개선되도록 하는 알고리즘

  - ✓ **Ex. SPAM Mail Filter**
    - **E** ➔ 기존에 받았던 SPAM Mail (데이터)
    - **T** ➔ SPAM Mail 판단 문제 (풀어야 할 문제)
    - **P** ➔ SAPM Mail을 얼마나 잘 분류하였는가? (결과)

Introduction

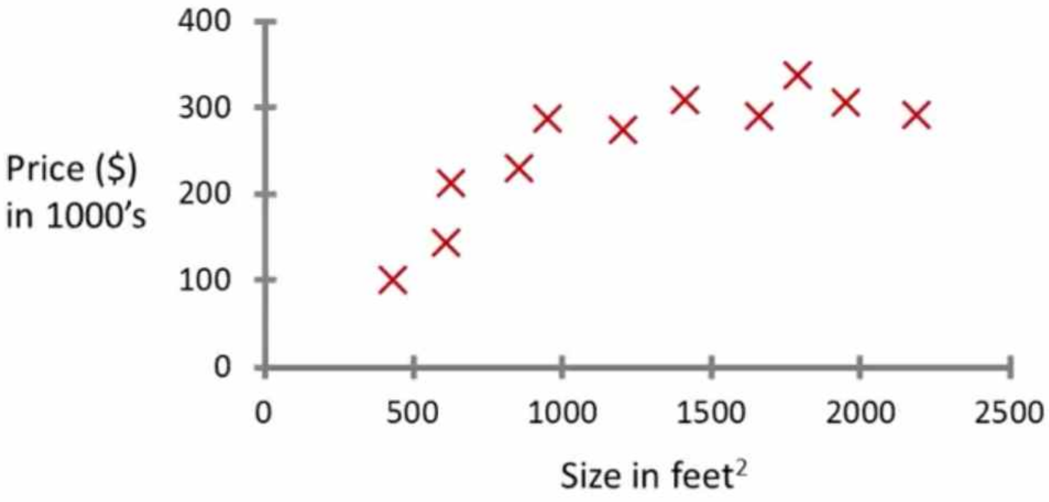
- **What is Machine Learning? (3/3)**
  - ✓ **Machine Learning Algorithm**
    - **Supervised Learning**
    - **Unsupervised Learning**
    - **Others: Reinforcement Learning, Recommender Systems**

❖ Why Machine Learning?

- ✓ **Complex Problem**
  - 복잡한 문제를 사람이 하는 것보다 빠르게 해결할 수 있음
  - 또한, 많은 양의 데이터를 처리하는데 용이함 (Big Data)
    - ※ Machine Learning은 많은 양의 데이터를 분석하기 때문에 Big Data 분야에서 방대한 데이터의 문제 해결에 사용됨

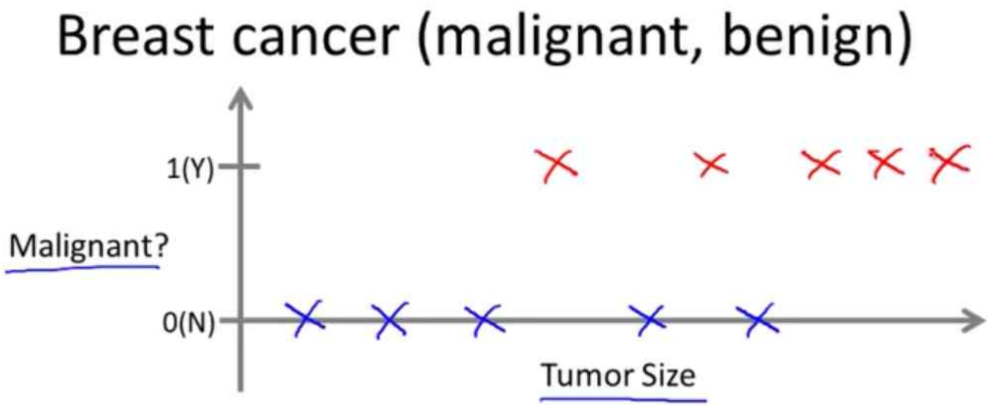
Introduction

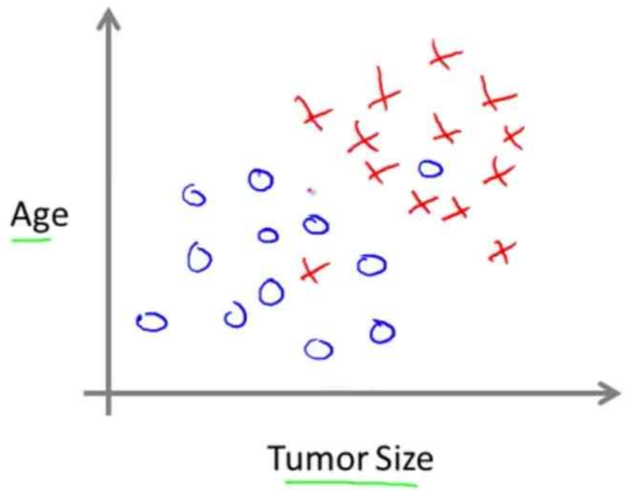
- **Supervised Learning (1/5)**
  - ✓ **Given** ***'Right answer'***
    - (학습 데이터) **Input X**에 대한 **Output Y**의 정보가 주어지는 경우
    - **Ex. Housing price prediction**
      - 집의 크기(Input X)에 따른 가격(Output X)을 예측하는 문제
      - 주어진 학습 데이터가 문제의 **입력**과 **출력**에 대한 값을 인지하고 있음



Introduction

- **Supervised Learning (2/5)**
  - ✓ **Given** ***'Right answer'***
    - (학습 데이터) **Input X**에 대한 **Output Y**의 정보가 주어지는 경우
    - **Ex. Breast cancer prediction**
      - 종양의 크기(Input X)에 따른 악성 종양(Output X)을 예측하는 문제
      - 주어진 학습 데이터가 문제의 **입력**과 **출력**에 대한 값을 인지하고 있음





Introduction

- **Supervised Learning (3/5)**
  - ✓ **Regression Problem vs Classification Problem**
    - ▪ **Problem Type**
      - **Continuous?**
      - **Discrete?**

Example 2:

Continuous {1, 2, 3, 4, … }

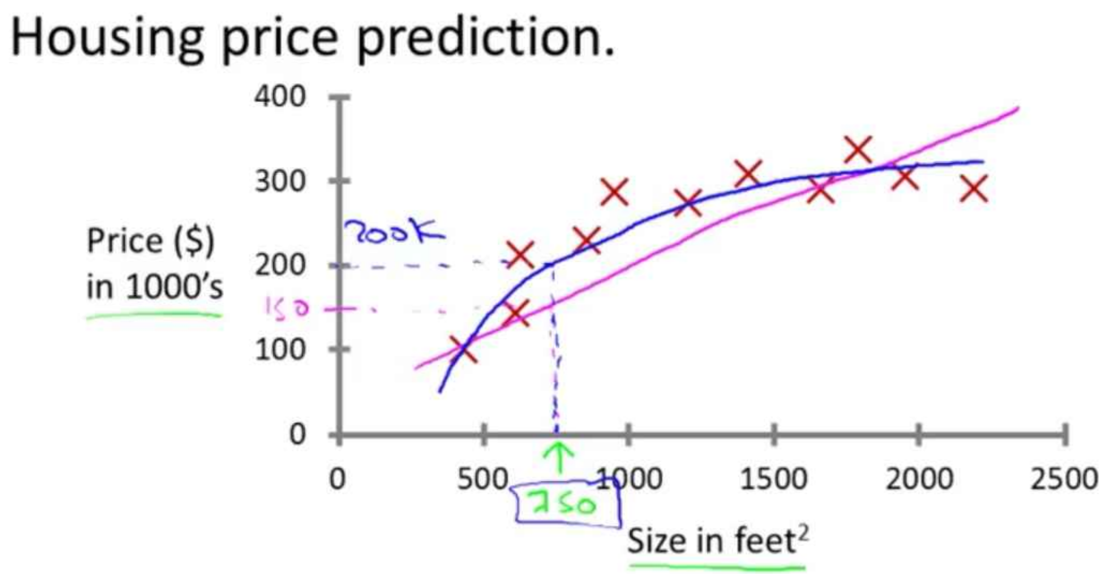
(a) Regression - Given a picture of a person, we have to predict their age on the basis of the given picture

(b) Classification - Given a patient with a tumor, we have to predict whether the tumor is malignant or benign.

Discrete {Yes or No}, {Type 1 or 2 … }

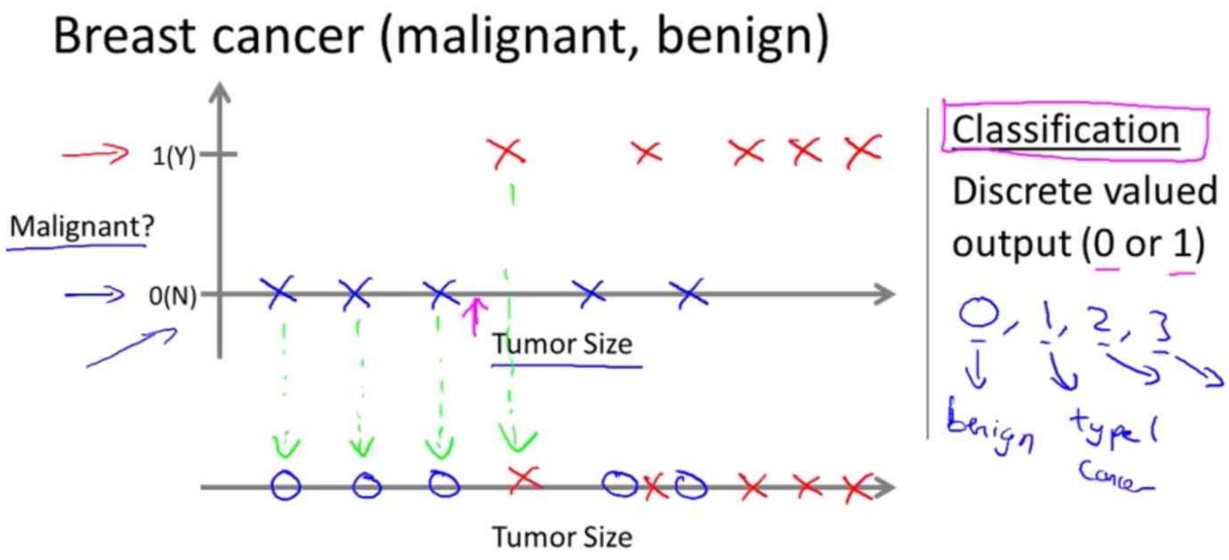
Introduction

- **Supervised Learning (4/5)**
  - ✓ **Regression Problem: Continuous Output**
    - 학습 데이터를 대표하는 **'Model'**을 만들고 미래의 사건을 예측
    - **Ex. Linear Regression**
      - Input Data ➔ Mapping ➔ ***Continuous*** Output



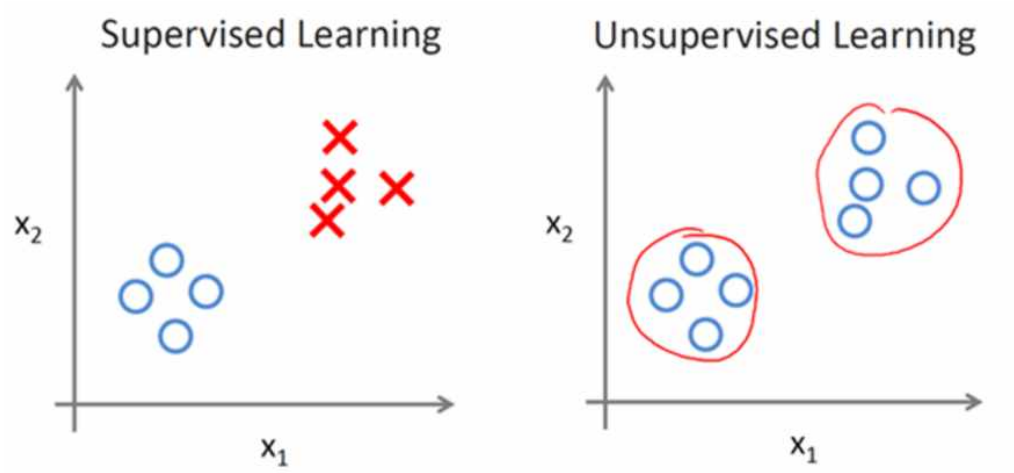
Introduction

- **Supervised Learning (5/5)**
  - ✓ **Classification Problem: Discrete Output**
    - 기존에 학습된 데이터를 근거로 새로운 데이터를 분류하는 문제
    - **Ex. K-NN, Support Vector Machine**
      - Input Data ➔ Classification ➔ ***Discrete*** Output (0 or 1)



Introduction

- **Unsupervised Learning (1/2)**
  - ✓ **Unknown** ***'Right answer'***
    - 학습 데이터를 구분할 수 있는 정보**(Label)**가 주어지지 않는 경우
    - **Classification of Training Data**
      - Supervised Learning: X ***(Training data is labeled)***
      - Unsupervised Learning ➔ **Need!**

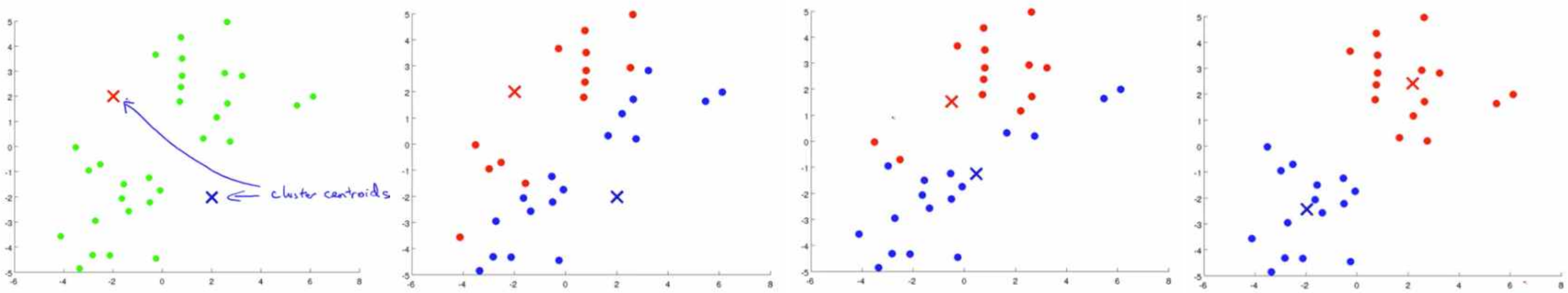


Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$.

Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Introduction

- **Unsupervised Learning (2/2)**
  - ✓ **Cluster(Grouping) Analysis**
    - ▪ Training Data에서 비슷한 특징들을 군집하는 것
    - ▪ **Goal:** 데이터가 어떻게 구성되어 있는지 알아내는 것
      - K-Means, Expectation Maximization(EM)
      - Ex. K-Means Algorithm



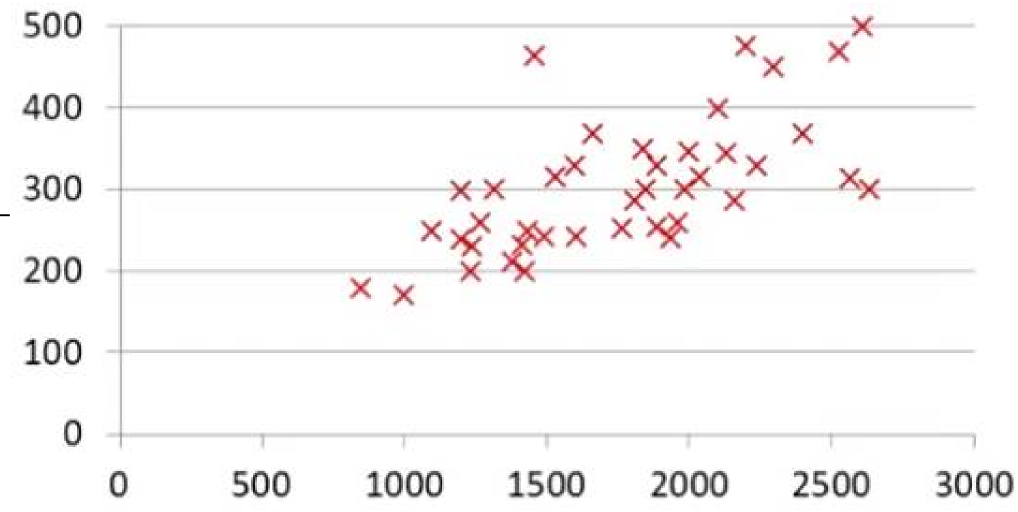
Model Representation

Linear Regression with One Variable

Linear Regression with One Variable

- **Model Representation (1/4)**
  - ✓ **Model of Training Data in Supervised Learning**

Output Prediction
<Y-Axis Data>

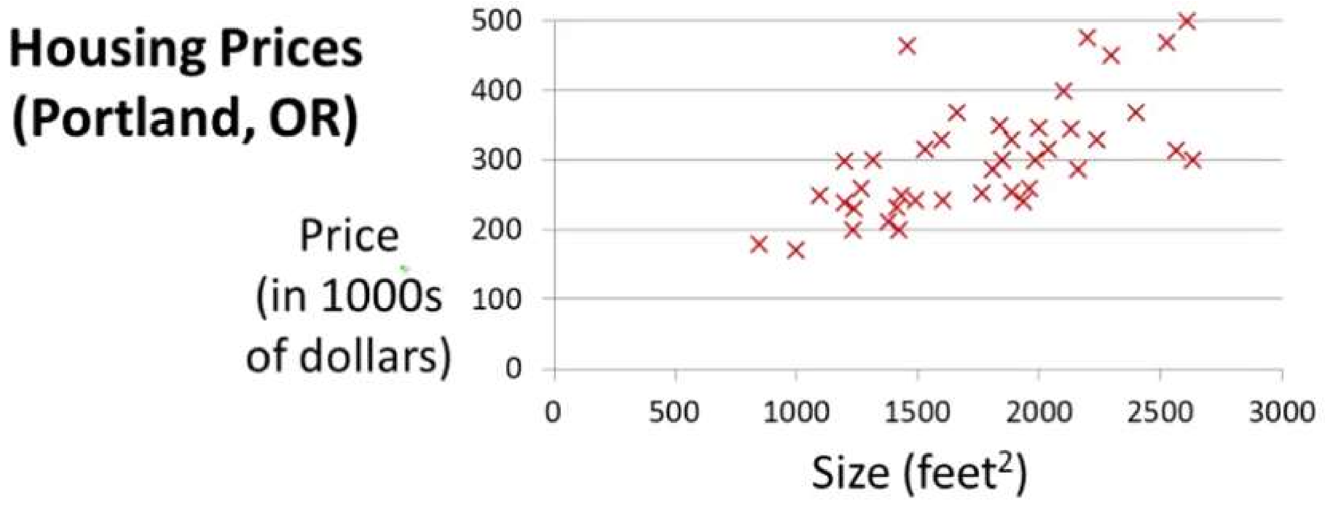


How to predictive output variable?

Input Feature
<X-Axis Data>

Linear Regression with One Variable

- **Model Representation (2/4)**
  - ✓ **Model of Training Data in Supervised Learning**
    - **Input(Feature)**에 대한 **Output**을 예측할 수 있는 ***판단 기준*** 필요
    - **What is Model?**
      - 학습 데이터의 특성을 묘사한 함수 형태의 객체
      - 즉, 학습 데이터를 대표할 수 있는 함수 ➔ ***Object Function***



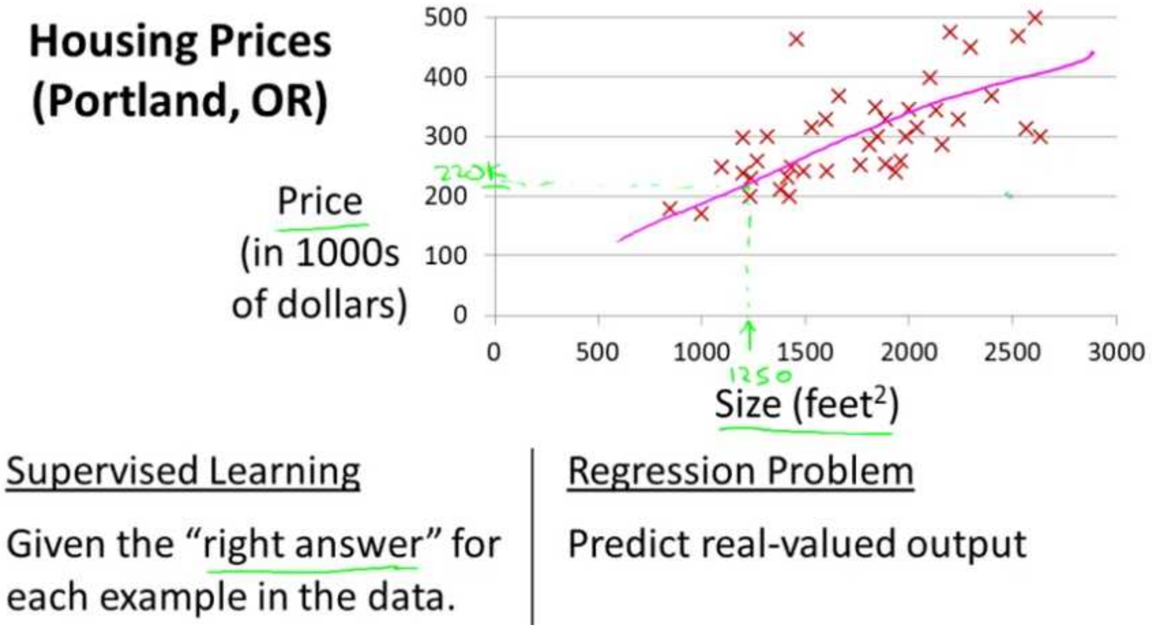
Linear Regression with One Variable

- **Model Representation (3/4)**

  ✓ **Ex. Linear Regression**

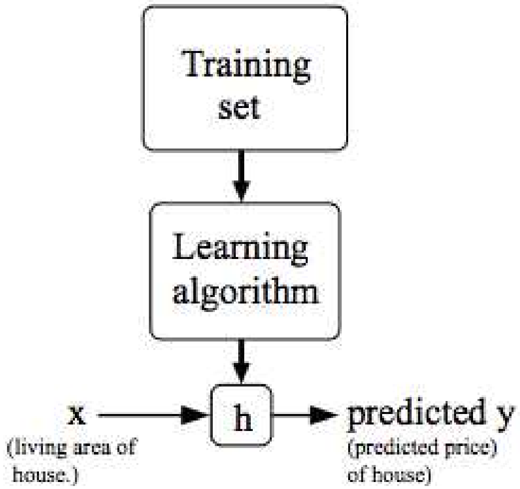
  - 학습 데이터를 대표하는 '**Model**'을 만들고 미래의 사건을 예측

  ※**Training Data ➔ *<u>Linear Model</u>***



Linear Regression with One Variable

- **Model Representation (4/4)**
  - ✓ **Model of Training Data in Supervised Learning**
    - ① 주어진 **'Training Set'**을 토대로
    - ② **Algorithm**을 통해 데이터를 **'Learning'**하여
    - ③ **Training Set**을 대표하는 **"Model"**을 생성하고
    - ④ 이를 통해 입력에 대한 출력을 ***예측*****(Prediction)**

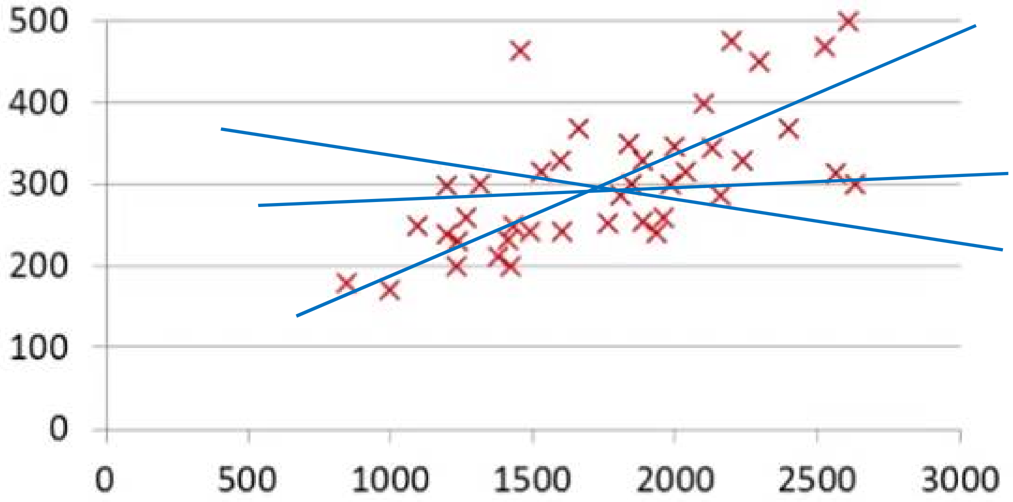


Cost Function

Linear Regression with One Variable

Linear Regression with One Variable

- **Cost Function (1/5)**
  - ✓ **Model of Training Data in Supervised Learning**



Q. 어떤 가설 *h*가 Training Data에 적합한 모델이라고 할 수 있는가?

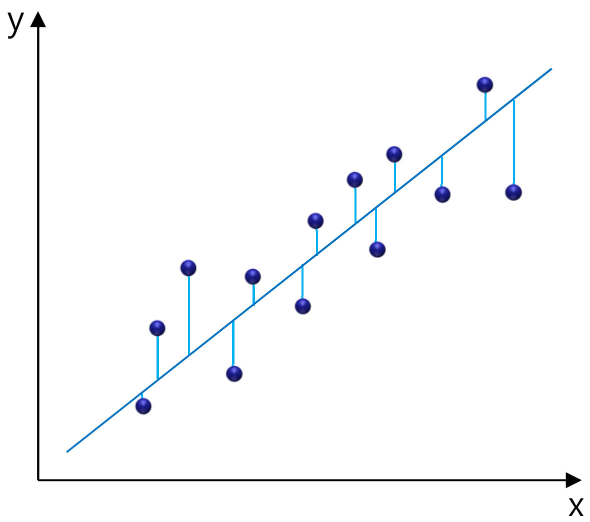
Linear Regression with One Variable

- ## Cost Function (2/5)

  ✓ **Definition**

  - **Training Data**와 가설 ***h***간의 적합성에 대한 척도: **비용(Cost)**
  - 적합도를 얻는 방법의 성능을 정량화
    - 적합도가 높을수록 낮은 비용 발생 ➔ **Best**
    - 적합도가 낮을수록 높은 비용 발생 ➔ **Worst**

    ※ ***<u>Cost Function을 최소화하면 최적의 추정치(Parameter)를 구할 수 있음</u>***



※ **Cost =** $\dfrac{| + | + | + | + \ \ldots + |}{\textbf{Number of data}}$

즉, Training Data와의 Cost가 가장 적은 *h*를 찾는 것!
<u>(Minimized Problem)</u>

Linear Regression with One Variable

- **Cost Function (3/5)**
  - ✓ **Definition of Mathematic: Linear Regression**

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

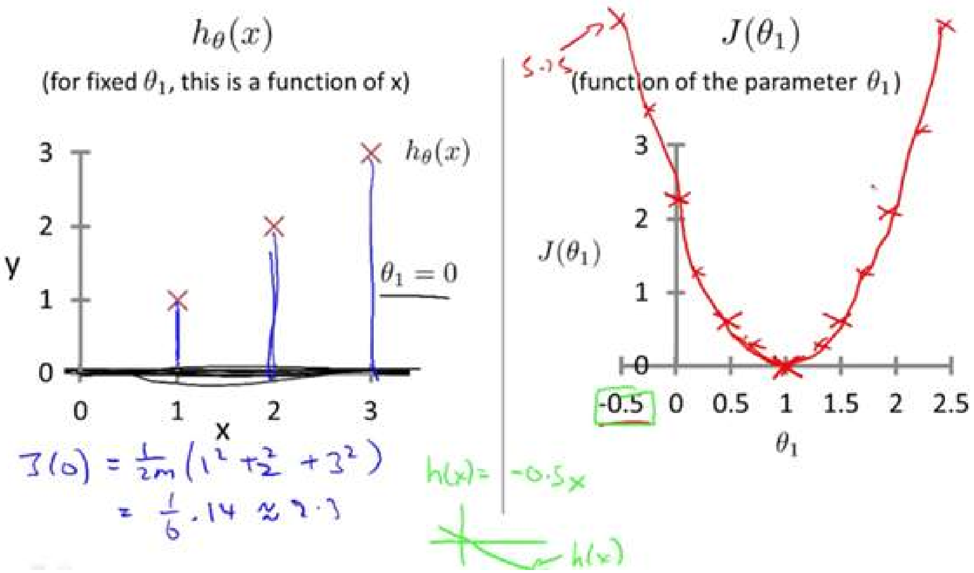
Goal: $\underset{\theta_0,\theta_1}{minimize} = J(\theta_0, \theta_1)$

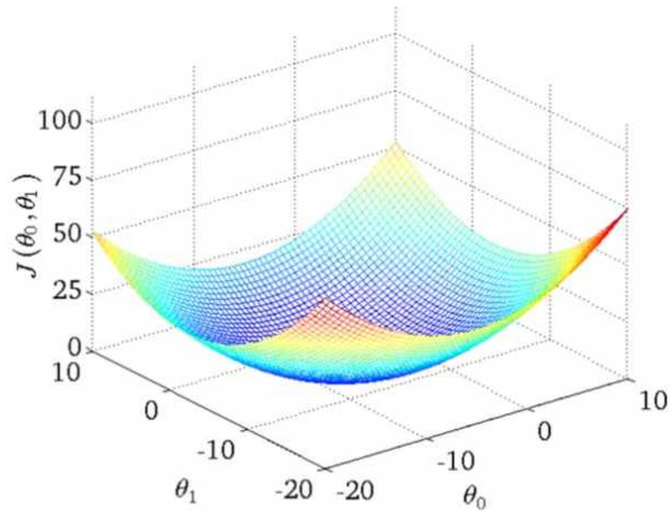
Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

Linear Regression with One Variable

- **Cost Function (4/5)**





Parameters: θ_1

Parameters: θ_0, θ_1

Linear Regression with One Variable

- **Cost Function (5/5)**
  - ✓ **Goal: Cost**가 가장 적은 최적의 추정치 ***h***를 찾음
    - ※ 즉, 가설(추론) ***h***에 대한 최적의 **Parameter** θ_0, θ_1 를 구함

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

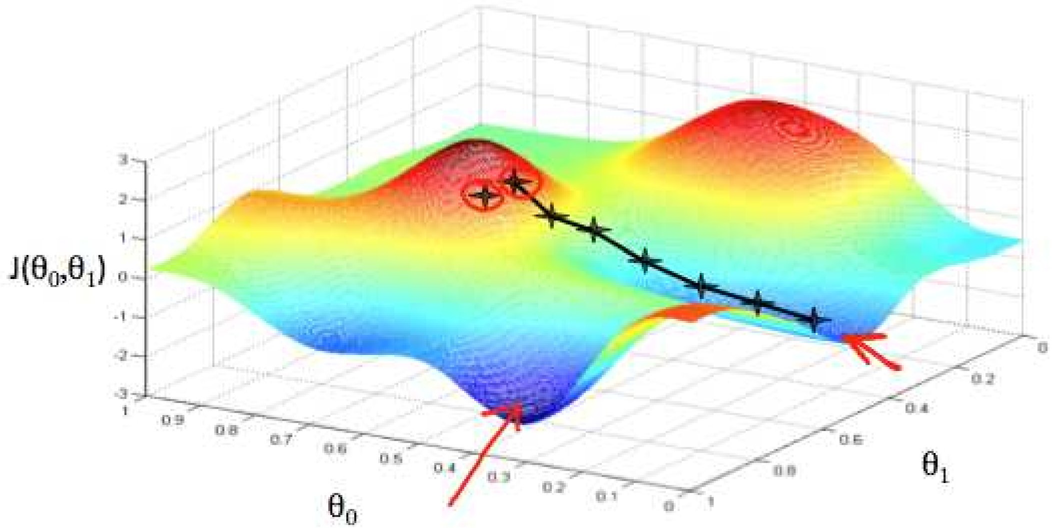
Gradient Descent

Linear Regression with One Variable

Parameter Learning

- **Gradient Descent**
  - ✓ **Definition**
    - ▪ *Function Optimization Method* (함수 최적화)
    - ▪ **Local Minimum ➔** 극소점
      - • <u>미분의 개념</u>을 최적화 문제에 적용 ➔ 경사면을 따라 함수의 극소를 찾음

      ※ ***<u>Gradient의 특성을 이용하여 비용함수에 대한 최적의 파라메터를 구함</u>***



❖ **시작점에 따라 Global Minimum을 찾지 못할 경우도 발생**

Parameter Learning

- ## Gradient Descent

  ✓ **Definition of Mathematatic: Linear Regression**

  - *Repeat until convergence*

$$\theta_j := \theta_j - \underline{\alpha} \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (for \;\; j = 0 \; and \;\; j = 1)$$

Learning Rate

  - **Update Rule**

$$temp0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

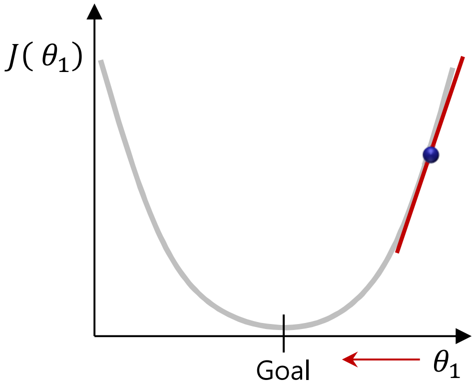
$$temp1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := temp0$$

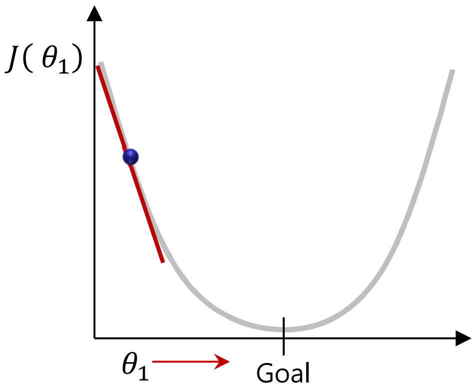
$$\theta_1 := temp1$$

Parameter Learning

- **Gradient Descent**
  - ✓ **Definition of Mathematic: Linear Regression**
    - $\theta_0 = 0, \quad J(\theta_1)$



$$\theta_1 := \theta_1 - \alpha \boxed{\frac{\partial}{\partial \theta_1} J(\theta_1)} \quad \geq 0$$



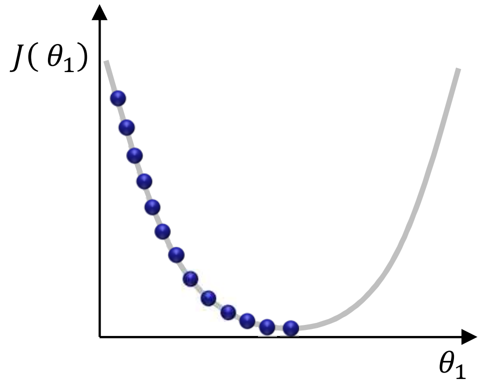
$$\theta_1 := \theta_1 - \alpha \boxed{\frac{\partial}{\partial \theta_1} J(\theta_1)} \quad \leq 0$$

Parameter Learning

- **Gradient Descent**

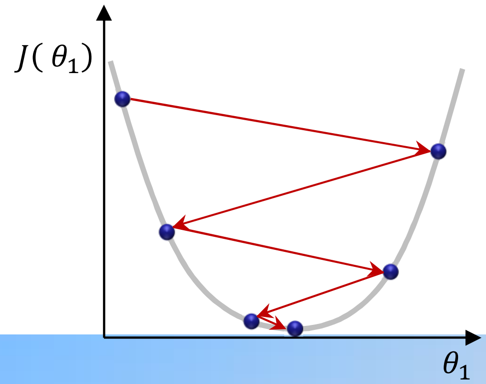
  ✓ **Definition of Mathematic: Linear Regression**

  - ***If Learning rate is too small, gradient descent can be slow.***



$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

  - ***If Learning rate is too large, gradient decent can overshoot the minimum.***



$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

Parameter Learning

- **Gradient Descent For Linear Regression**

  ✓ **Gradient Descent of Cost Function**

  - Gradient 특성을 이용하여 Cost Function에 대한 ***최적의 파라메터***를 구함
  - *Repeat until convergence*

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (for\ \ j = 0\ and\ \ j = 1)$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)^2$$

$$J = 0 : \quad \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)$$

$$J = 1 : \quad \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i) x_i$$

Parameter Learning

- ## Gradient Descent For Linear Regression

✓ Gradient Descent of Cost Function

- Gradient 특성을 이용하여 Cost Function에 대한 ***최적의 파라메터***를 구함
- *Repeat until convergence*

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (for\ \ j = 0\ and\ \ j = 1)$$

$$\left.\begin{aligned} \theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i) \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i) x_i \end{aligned}\right\}$$

❖ **Update** θ_0 **and** θ_1

Parameter Learning

- **Gradient Descent For Linear Regression**

✓ **Goal: Cost**가 가장 적은 최적의 추정치 ***h***를 찾음

※ 즉, 가설(추론) ***h***에 대한 최적의 **Parameter** θ_0, θ_1 를 구함

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Parameter Learning - Experiment

Question 2

Many substances that can burn (such as gasoline and alcohol) have a chemical structure based on carbon atoms; for this reason they are called hydrocarbons. A chemist wants to understand how the number of carbon atoms in a molecule affects how much energy is released when that molecule combusts (meaning that it is burned). The chemists obtains the dataset below. In the column on the right, "kJ/mol" is the unit measuring the amount of energy released. examples.

Name of Molecule	Number of Carbon Atoms per Molecule (x)	Heat Released when Burned (y) (kJ/mol)
methane	1	-890
ethene	2	-1411
ethane	3	-1560
propane	4	-2220
cyclopropane	5	-2091
butane	6	-2878
pentane	7	-3537
benzene	8	-3268
cycloexane	9	-3920
hexane	10	-4163
octane	11	-5471
napthalene	12	-5157

You would like to use linear regression (hθ(x)=θ_0+θ_1x) to estimate the amount of energy released (y) as a function of the number of carbon atoms (x). Which of the following do you think will be the values you obtain for θ_0 and θ_1? You should be able to select the right answer without actually implementing linear regression.

Parameter Learning - Experiment

- **Gradient Descent For Linear Regression**
  - ✓ **Experiment**
    - **Program Language** ➔ Python, Matlab
    - Learning rate: 0.002
    - Stop Condition: Smaller steps(0.3)

Parameter Learning - Experiment

- **Cost Function for Linear Regression**

```python
def CalcLinearCost(a, b, x, y, m, d = 0):
    # @param a: parameter (Linear model)
    # @param b: parameter (Linear model)
    # @param x: training data (x-axis)
    # @param y: training data (y-axis)
    # @param m: size of training sets
    # @param d: flag for parameter
    # @return : Cost
    sum = 0
    for i in range(m):
        if d == 0: sum += ((a * x[i] + b) - y[i]) * x[i]     # 'a' Parameter
        if d == 1: sum += ((a * x[i] + b) - y[i])            # 'b' Parameter
    # end for

    return sum / m
# end function
```

Parameter Learning - Experiment

- **Main Function**

```python
from CalculateCost import *

# Training Sets
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
y = [-890, -1411, -1560, -2220, -2091, -2878, -3537, -3268, -3920, -4163, -5471, -5157]
if len(x) != len(y):
    print('The size of two arrays is different\n')
    exit()

# Size of training sets
m = len(x)

# Parameter settings
a            = 0.9
b            = 0.9
learningRate = 0.002
threshold    = 0.3

# Parameters learning
while True:
    # Calculate parameters for update
    tempA = a - learningRate * CalcLinearCost(a, b, x, y, m)
    tempB = b - learningRate * CalcLinearCost(a, b, x, y, m, 1)

    # Learing stop condition
    if abs(tempA - a) <= threshold and abs(tempB - b) <= threshold:
        break

    # Update
    a = tempA
    b = tempB
# end while

# Results
print("Parameter 'a': ", a)
print("Parameter 'b': ", b)
```

Parameter Learning - Experiment

- **Result of Experiment**