Supervised Learning & Linear Regression

Deep learning

Woohwan Jung

한양대학교
HANYANG UNIVERSITY

DSLAB
데이터사이언스 연구실

Topics

- **Supervised learning**
- **Linear regression**
  - Model
  - Cost/Loss function
  - Optimization

Supervised learning

Supervised learning

Goal: generalize the input-output relationship

$x \longrightarrow$ **Model** $\longrightarrow \hat{y} \quad \approx \quad y$

Input (features) **Output (prediction)** **Label (Actual)**

Training?

Building a model to make the model can **predict the labels** by using train data

Dataset

Features Label

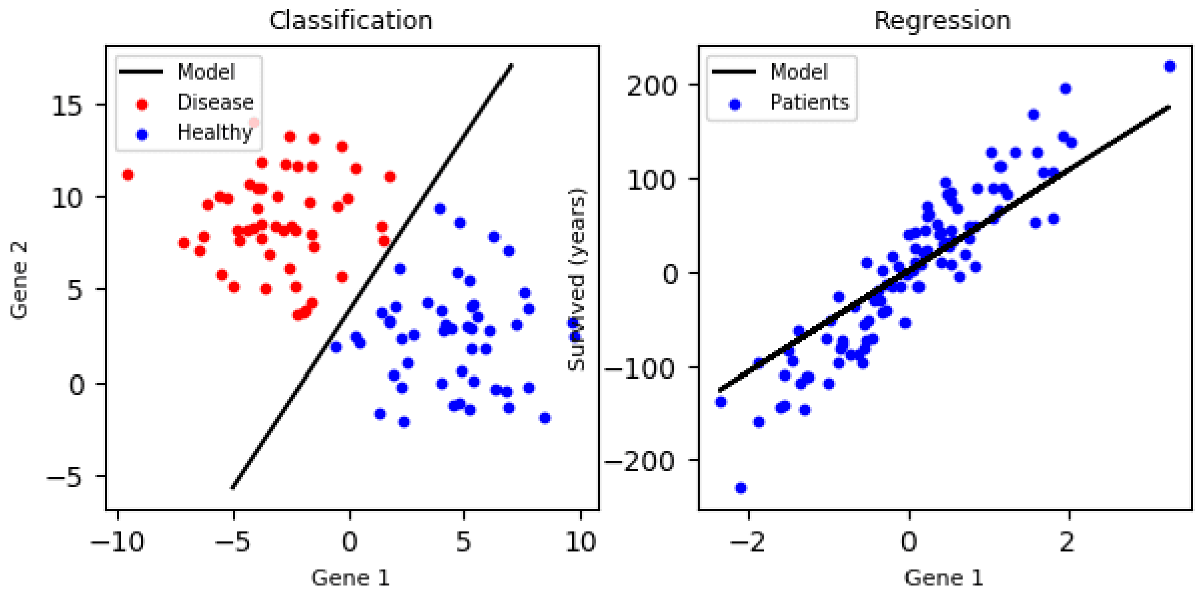
Rows

Size	Beds	Baths	Zip	Price
1100	1	1	64576	1.29
1900	3	1.5	78321	2.14
2800	3	3	98712	3.10
3400	4	3.5	25721	3.75

Columns

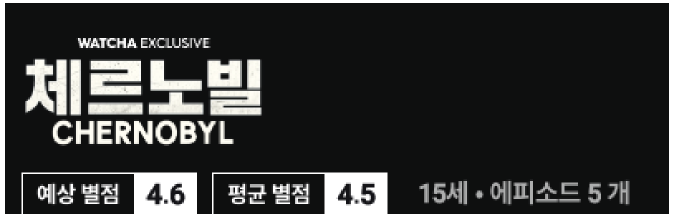
$$D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots (\boldsymbol{x}^{(m)}, y^{(m)})\}$$

Classification vs Regression



Output **Categorical value (Class)** **Numeric value**

Q1. Classification? Regression?



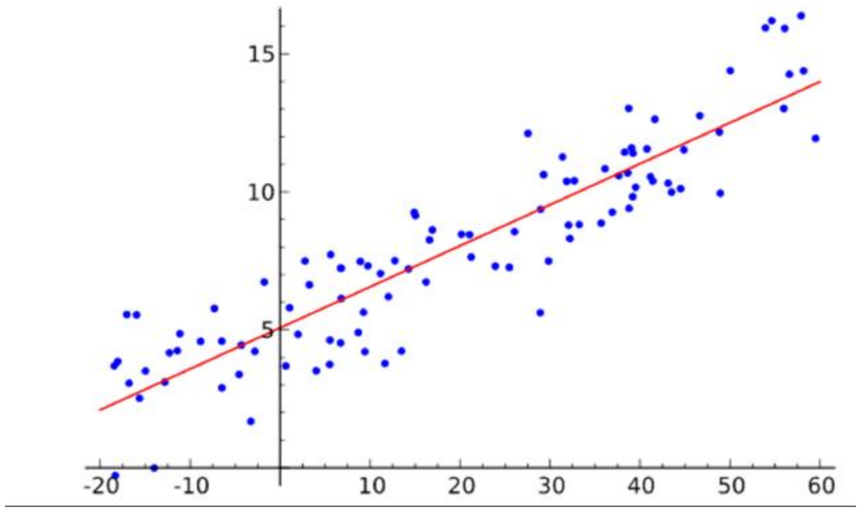
Q2. Classification? Regression?

Cat

Dog

Linear Regression

- Modelling the linear relationship between a scalar response (label) and one or more explanatory variables (features)

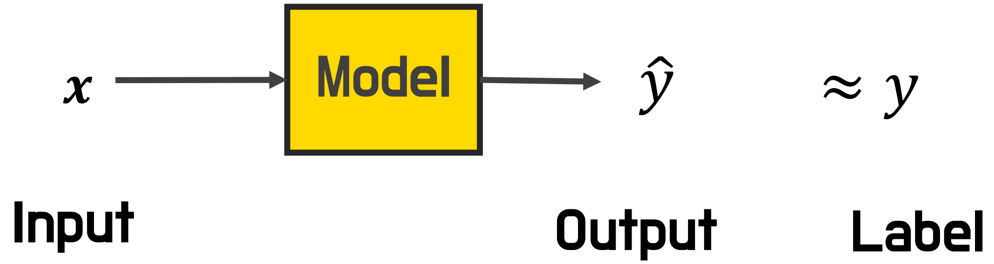


Examples)
Area of house -> house price
# of iPhones sold -> Apple's sales

An example of regression algorithm

Linear Regression

- Data $D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots (\boldsymbol{x}^{(m)}, y^{(m)})\}$
- Model
  - Input: $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$
  - Output $\hat{y}^{(i)} = \boldsymbol{w}^{\top}\boldsymbol{x}^{(i)} + b$
    - Parameters: $\boldsymbol{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$



Training a linear regression model?

Finding the model parameters $\boldsymbol{w}$ and b which make $\hat{y} \approx y$

Measuring the distance between $\hat{y}$ and y

Squared error: $(\hat{y} - y)^2$

Loss function $L(\hat{y}^{(i)}, y^{(i)}) = \left(y^{(i)} - \hat{y}^{(i)}\right)^2$

Cost function $J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$

Training a linear regression model

- Given
  - Training data $D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots (\boldsymbol{x}^{(m)}, y^{(m)})\}$
- Our goal
  - Find $\boldsymbol{w}, b$ that minimizes $J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$

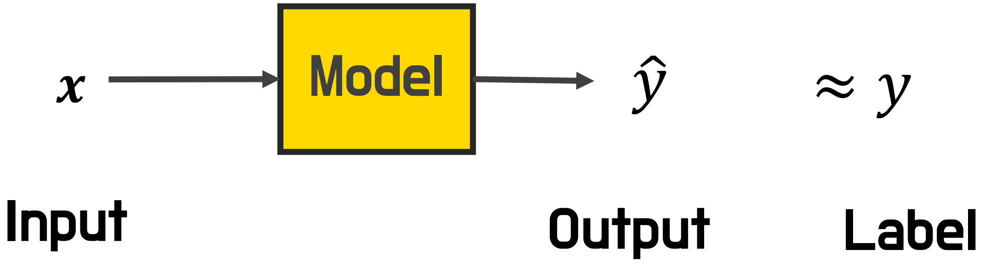
Q. How?

Applicable methods: gradient descent, linear least squares, …

We are going to use gradient descent!

Gradient Descent
:Linear Regression

Optimization



- Linear regression model
  - $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$
- Loss function $L\left(\hat{y}^{(i)}, y^{(i)}\right) = \left(y^{(i)} - \hat{y}^{(i)}\right)^2$
- Cost function

$$J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

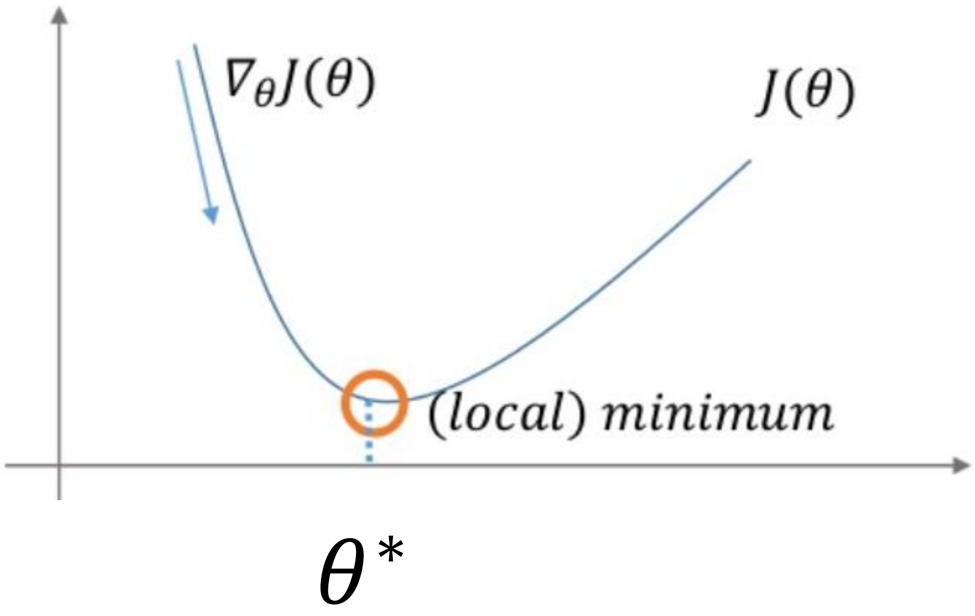
- Our goal
  - Find parameters $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$ that minimize $J(\boldsymbol{w}, b)$
- Gradient Descent!

Gradient Descent

- A popular optimization algorithm to minimize a cost function $J(\theta)$
  - $J(\boldsymbol{\theta})$: cost function
  - $\boldsymbol{\theta}$: model parameters
- Iteratively update model parameters to find the values that result in the lowest $J(\theta)$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

  - η: Learning rate



$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \left[\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} \quad \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2} \ \dots \ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_k}\right]^T \text{where } \boldsymbol{\theta} = [\theta_1 \quad \theta_2 \dots \quad \theta_k]^T$$

By Hakky St

Gradient of L

- Output $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$
- Loss function $L(\hat{y}, y) = (y - \hat{y})^2$

$$\frac{\partial L}{\partial w_k} = -2(y - \hat{y})x_k \qquad \frac{\partial L}{\partial b} = -2(y - \hat{y})$$

Gradient descent on m examples

$$J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

Gradient descent
$$\boldsymbol{\theta} = \boldsymbol{\theta} \text{-}\ \eta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\frac{\partial}{\partial w_k} J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial w_k} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{2}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y\right) x_k$$

$$\frac{\partial}{\partial b} J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial b} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{2}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y\right)$$

Gradient descent for training a Linear Regression model (n = 2)

- Randomly Initialize $\boldsymbol{w}, b$
- lr = 0.1
- For e = 1 to n_{epoch}:
  - d_w1 = 0; d_w2 = 0; d_b=0
  - For i= 1 to m:
    - $a = w_1 x_1^{(i)} + w_2 x_2^{(i)} + b$
    - d_w1 += $2(a - y)x_1^{(i)}$
    - d_w2 += $2(a - y)x_2^{(i)}$
    - d_b += $2(a - y)$
  - $w_1 -= lr * d\_w1/\mathrm{m}$
  - $w_2 -= lr * d\_w2/\mathrm{m}$
  - $b -= lr * d\_b/\mathrm{m}$

Gradient descent
$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

$$\frac{\partial}{\partial w_k} J(\boldsymbol{w}, b) = \frac{2}{m} \sum_{i=1}^{m} (a - y) x_k$$

$$\frac{\partial}{\partial b} J(\boldsymbol{w}, b) = \frac{2}{m} \sum_{i=1}^{m} (a - y)$$