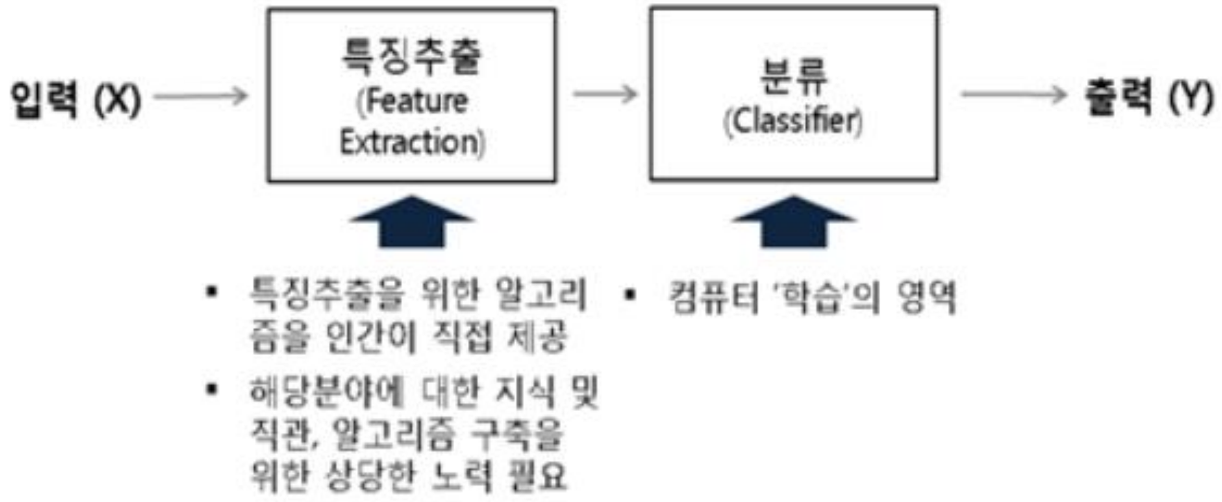
Linear & Logistic regression

자연어처리연구실
김균엽

Machine Learning

- machine learning process
  - **특징이 추출된 정제된 입력**을 **지정된 특정 알고리즘**을 이용하여 원하는 결과를 예측하는 방법론



Machine Learning

- machine learning process
  - 사람이 직접 추출한 입력 데이터의 feature를 기반으로 정답을 예측하는 것
  - 입력된 feature과 정답 사이의 pattern을 사람이 설정한 식(알고리즘)을 통해 표현하고자 하는 것
  - ex)
    - 학습한 시간과 성적간의 상관관계를 가장 잘 나타내는 식(알고리즘) 찾기
    - 자연어에서 추출된 TF-IDF를 SVM을 통해 문장의 감정 분류

Machine Learning

- machine learning의 특징
  - 정해진 알고리즘(식)을 통해 결과를 도출한다.
  - 정제된 입력을 이용하여 결과를 도출한다.
  - 정제된 입력을 기반으로 파라미터를 학습한다

Linear Regression

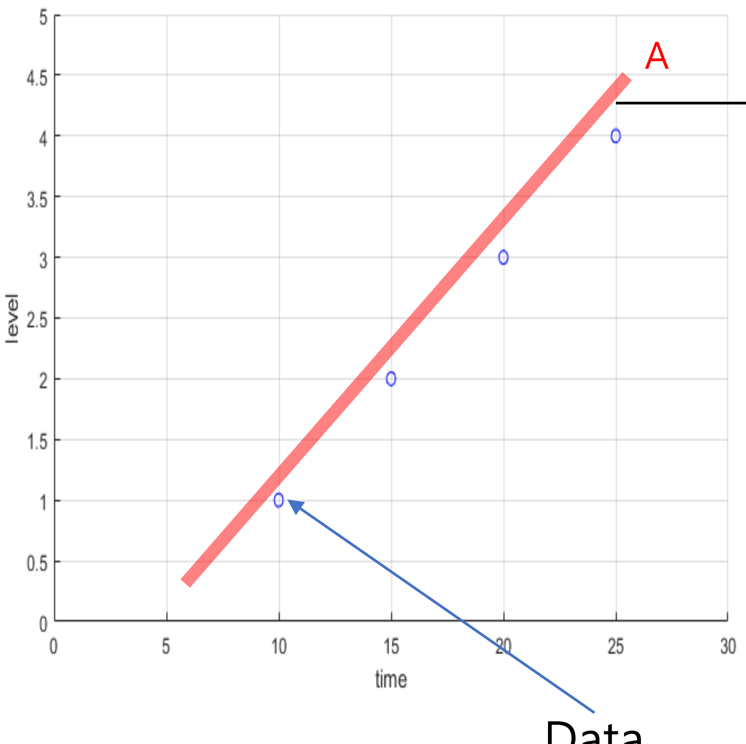
- Regression
  - 머신러닝 알고리즘을 통해 유사한 값이 나오도록 하는 task
  - 연속된 숫자 중 실제 값과 유사한 값을 예측
  - Ex)공부한 시간을 기반으로한 점수 예측
- Classification
  - 머신러닝 알고리즘을 통해 정답 class가 나오도록 예측
  - Binary classification에서는 0/1과같이 각 class에 해당하는 boolean을 예측
  - Ex)공부한 시간을 기반으로한 통과 여부 예측

Linear Regression

- linear regression
  - 어떠한 입력들이 주어졌을때 그 입력들을 대표할 수 있는 직선을 찾는 알고리즘
- 그래프상에서 어떠한 직선을 찾는 알고리즘이기에 1차 함수를 사용
  - hypothesis: y = ax + b
- x,y를 포함한 데이터셋을 가장 잘 표현하는 a,b를 찾는 과정
  - x,y = data, target
  - a,b = parameter
- 가장 최적의 a,b를 찾는것이 목표

Linear Regression

- Linear Regression
  - 선별된 1개의 값을 1차함수에 입력하여 정답을 예측하는 머신러닝 알고리즘



학습할 파라미터

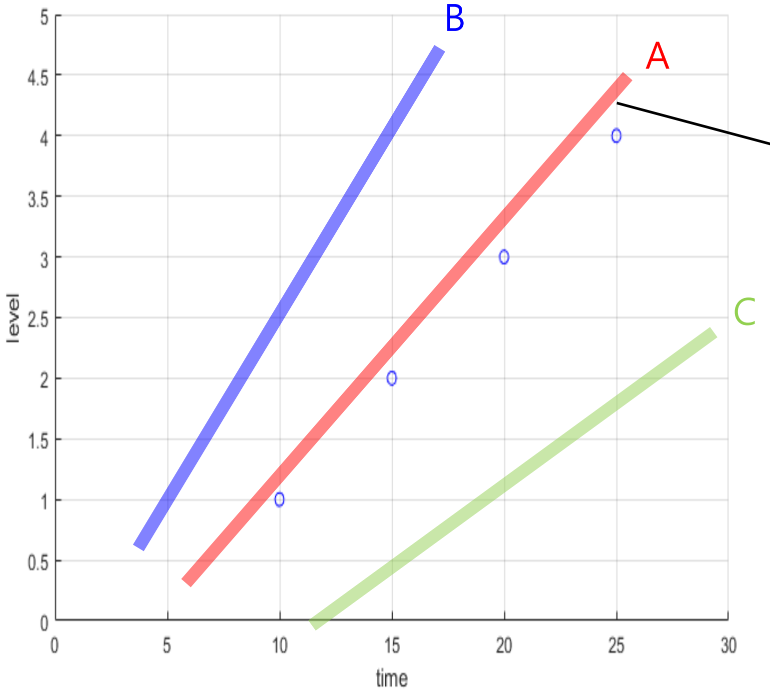
Level = a * time + b

variable(data)
선별된 입력

Hypothesis
알고리즘

Linear Regression

- linear regression
  - A,B,C중에 또는 그릴수있는 모든 직선중 경우의수 중 가장 데이터를 잘 표현할 수 있는 선이 무엇인지 찾는 과정
  - ex)학습한 시간과 성적의 관계를 가장 잘 나타내는 1차함수 찾기



parameters

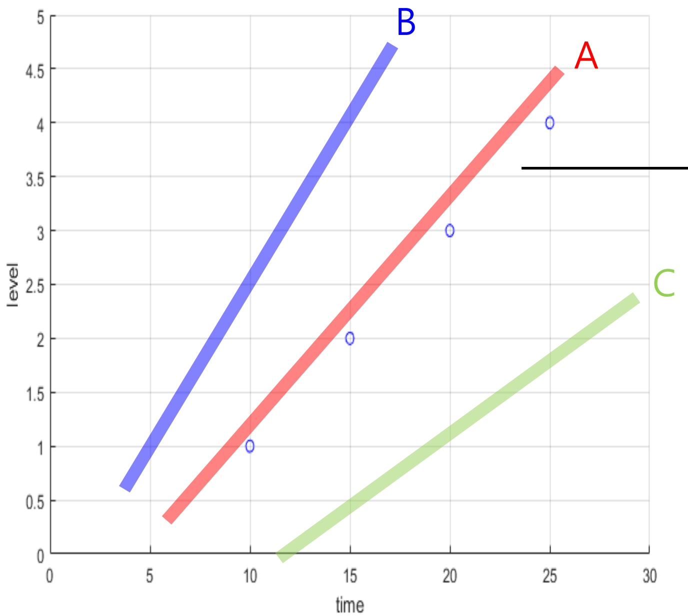
Level = a * time + b

variable(data)

hypothesis

Linear Regression

- linear regression
  - A,B,C중 데이터를 가장 잘 나타낼 수 있는 직선은? A
  - 이유는 A와 거리가 가장 가깝기 때문
  - 이러한 이유를 정량적으로 나타내기 위해 데이터와 직선의 거리를 error라고 표현
  - error의 제곱의 합이 적을수록 데이터를 가장 잘 나타내는 직선임



parameters

Level = a * time + b

variable(data)

hypothesis

Linear Regression

- MSE
  - 예측값과의 거리를 정량적으로 나타내기 위한 식
  - 예측값과 실제 정답간의 거리의 제곱값
  - 예측값과 실제 정답의 괴리가 클수록 큰값이 나오고 정확할수록 작은 값이 나옴

MSE = $\frac{1}{N}\sum_{i=1}^{N}(predict\ - ground\ thruth)^2$

C(a) = $\frac{1}{N}\sum_{i=1}^{N}(a\ * time + b\ - ground\ thruth)^2$

Method of Least Squares

- 최소제곱법
  - linear regression과 같이 적은 수의 parameter을 가진 식의 경우 데이터와 정답을 찾는 수식을 이용해서 error가 가장 적은 x절편과 y절편을 구할 수 있음
  - 지금 가진 정보가 x 값과 y 값일 때 이를 이용해 기울기 a를 구하는 방법은 다음식과 같다
  - 해당식을 이용하면 가장 적은 MSE가 나오는 a,b가 도출됨

$$a = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$b = \bar{y} - a\bar{x}$$

x가 한 개일 때만 성립

Method of Least Squares

- linear regression
  - example - 사용한 시간에 따른 level에 대한 데이터

시간(X)	등급(Y)
10	1
15	2
20	3
25	4

$$a = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

- mean X = 17.5
- mean Y = 2.5

$$a = \frac{(10-17.5)(1-2.5)+(15-17.5)(2-2.5)+(20-17.5)(3-2.5)+(25-17.5)(4-2.5)}{(10-17.5)^2+(15-17.5)^2+(20-17.5)^2+(25-17.5)^2}$$

$$a = \frac{(-7.5)*(-1.5)+(-2.5)(-0.5)+(2.5)(0.5)+(7.5)(1.5)}{(-7.5)^2+(-2.5)^2+(2.5)^2+(7.5)^2}$$

$$a = \frac{25}{125} = 0.5$$

Method of Least Squares

- linear regression
  - example - 사용한 시간에 따른 level에 대한 데이터

시간(X)	등급(Y)
10	1
15	2
20	3
25	4

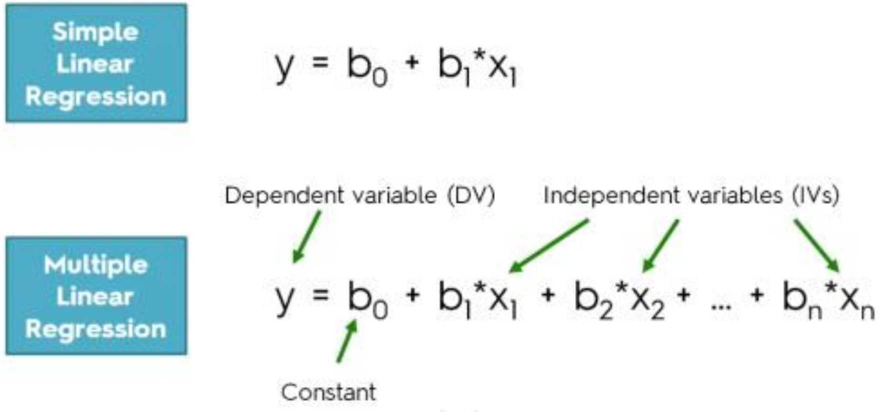
- mean X = 17.5
- mean Y = 2.5
- A = 0.2
- B = -1

$$b = \bar{y} - a\bar{x}$$

$$b = 2.5 - 0.2 * 17.5 = 2.5 - 3.4 = -1$$

Method of Least Squares

- problem linear regression
  - 입력할 수 있는 feature가 1개가 아니면 1차함수를 통해 나타낼 수 없음
    - Multiple linear regression



  - 1차함수가 아닌 곡선형태를 나타낼 수 없음
    - polynomial regression

- Quadratic – 2nd order

$$\hat{Y} = b_0 + b_1\,x_1 + b_2(\,x_1)^2$$

- Cubic – 3rd order

$$\hat{Y} = b_0 + b_1\,x_1 + b_2(\,x_1)^2 + b_3(\,x_1)^3$$

- Higher order

$$\hat{Y} = b_0 + b_1\,x_1 + b_2(\,x_1)^2 + b_3(\,x_1)^3 + ..$$

Method of Least Squares

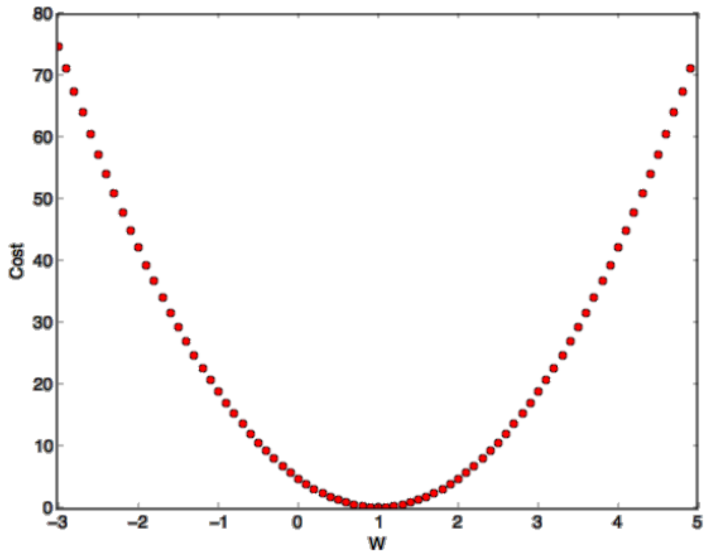
- problem linear regression
  - multiple linear regression과 polynomial regression은 더 많은 parameter를 가짐
  - linear regression처럼 parameter갯수가 정해져있지 않고 많기 때문에 모든 case에 대해 **최소제곱법처럼 최적의 parameter를 구할 수 있는 식**을 만들 수 없음
    - -> 경사하강법(Gradient descent)

minimize MSE function

Hypo: Level = a * time + b

- Gradient descent in linear regression
  - b = 0 이라 가정, parameter은 a = W하나라 가정
    - 방법1) 최대한 많은 W에 대해서 값을 구해봄
    - 모든 case에 대해서 값을 구하면 최저일때의 W값을 확인할 수 있음
    - **하지만 이 방법은 너무 많은 자원 소모**

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$



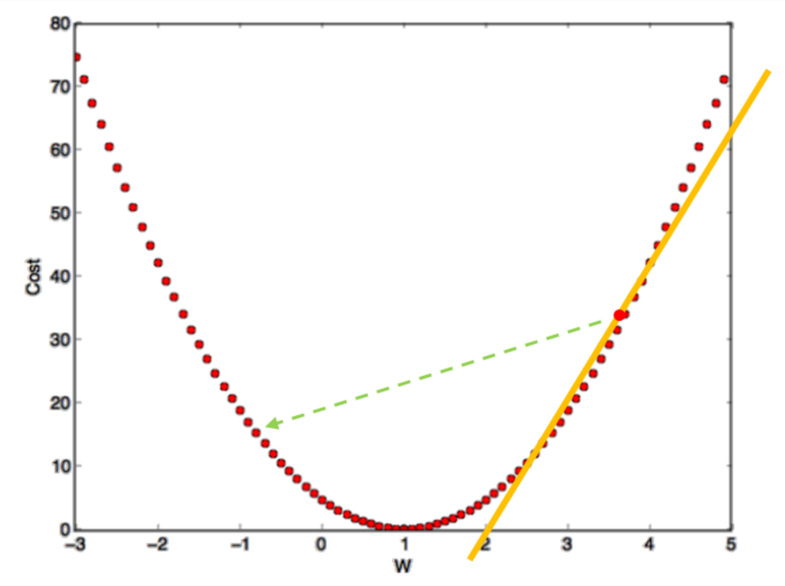
minimize MSE function

Hypo: Level = a * time + b

- Gradient descent in linear regression
  - b = 0 이라 가정, parameter은 a = W하나라 가정
    - 방법2) gradient descent
    - 가장 먼저 random한 값의 parameter에서 시작
    - 현재 기울기를 기반으로 최소값을 찾아갈 수 있음
    - 매번 기울기를 빼서 기울기가 0인곳으로 점점 이동

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$



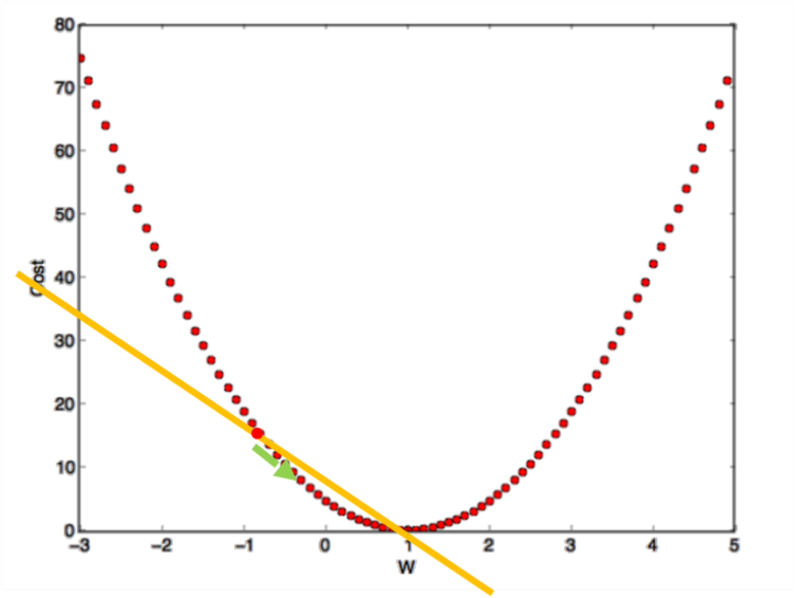
minimize MSE function

Hypo: Level = a * time + b

- Gradient descent in linear regression
  - b = 0 이라 가정, parameter은 a = W하나라 가정
    - 방법2) gradient descent
    - 가장 먼저 random한 값의 parameter에서 시작
    - 현재 기울기를 기반으로 최소값을 찾아갈 수 있음
    - 매번 기울기를 빼서 기울기가 0인곳으로 점점 이동

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$



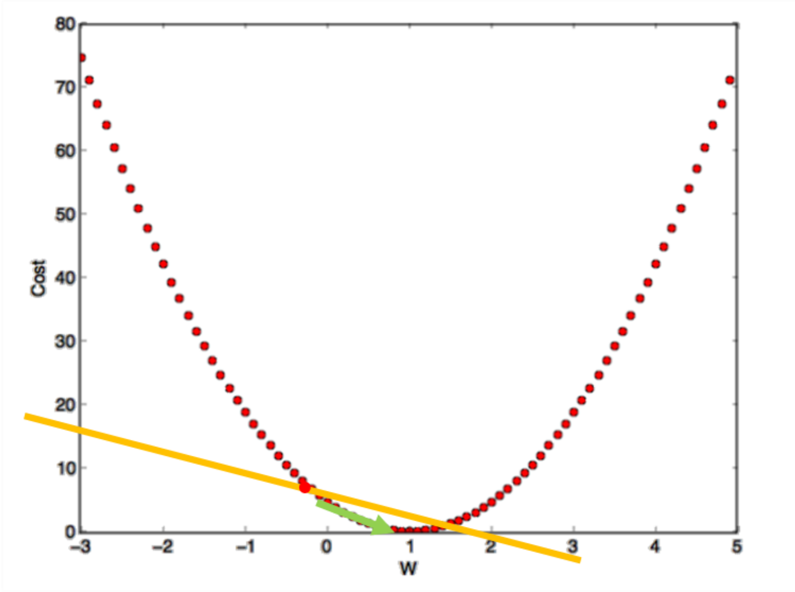
minimize MSE function

Hypo: Level = a * time + b

- Gradient descent in linear regression
  - b = 0 이라 가정, parameter은 a = W하나라 가정
    - 방법2) gradient descent
    - 가장 먼저 random한 값의 parameter에서 시작
    - 현재 기울기를 기반으로 최소값을 찾아갈 수 있음
    - **매번 기울기를 빼서 기울기가 0인곳으로 점점 이동**

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (W x^{(i)} - y^{(i)})^2$$



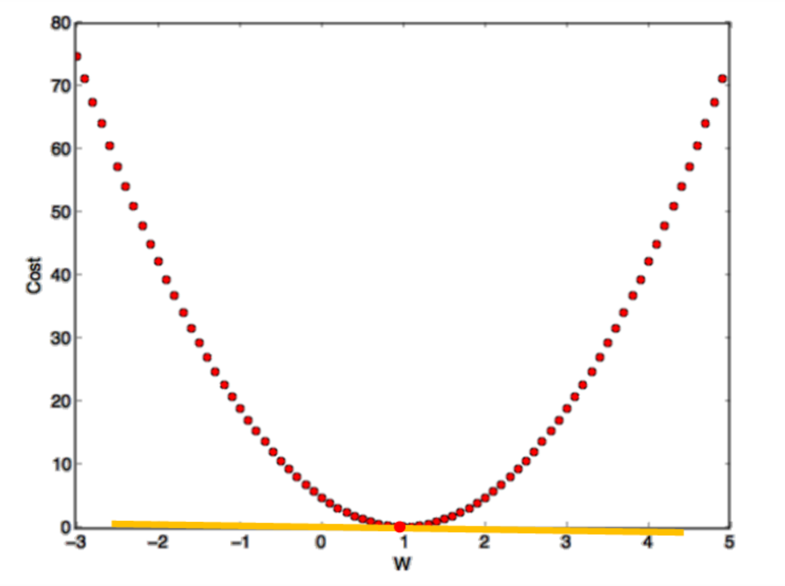
minimize MSE function

Hypo: Level = a * time + b

- Gradient descent in linear regression
  - b = 0 이라 가정, parameter은 a = W하나라 가정
    - 방법2) gradient descent
    - 가장 먼저 random한 값의 parameter에서 시작
    - 현재 기울기를 기반으로 최소값을 찾아갈 수 있음
    - 매번 기울기를 빼서 기울기가 0인곳으로 점점 이동

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$



minimize MSE function

Hypo: Level = a * time + b

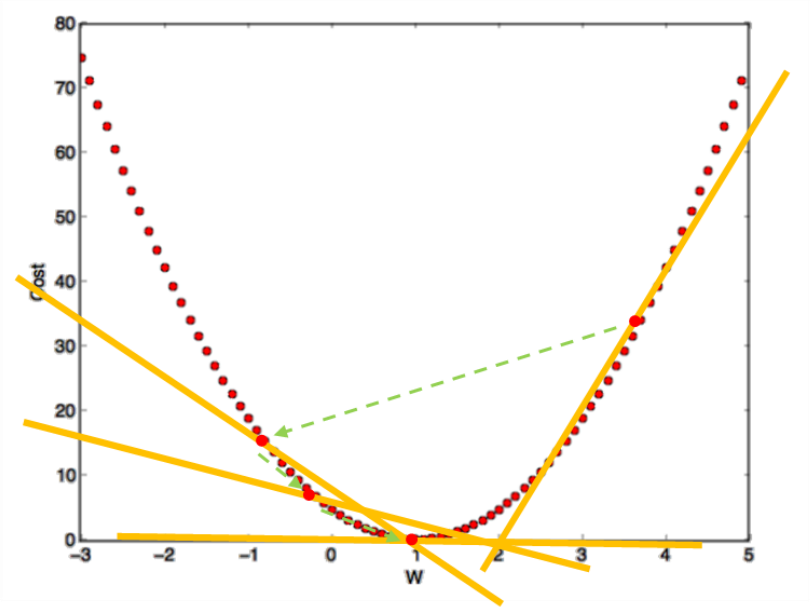
식과 학습 데이터들이 존재하면 미분값 계산가능

미분한 값을 기반으로 기울기가 0인부분을 탐색

이를 통해 gradient descent는
cost function이 줄어는 방향으로 학습함

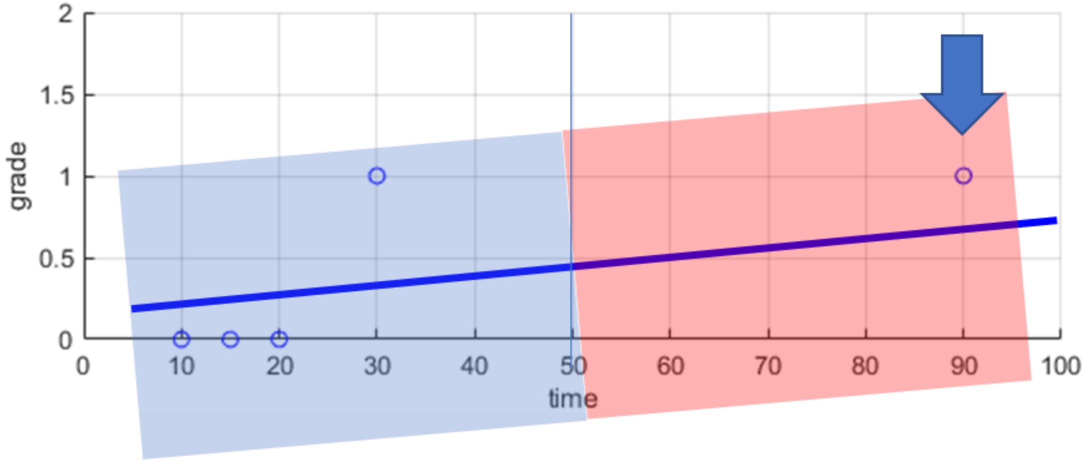
$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$



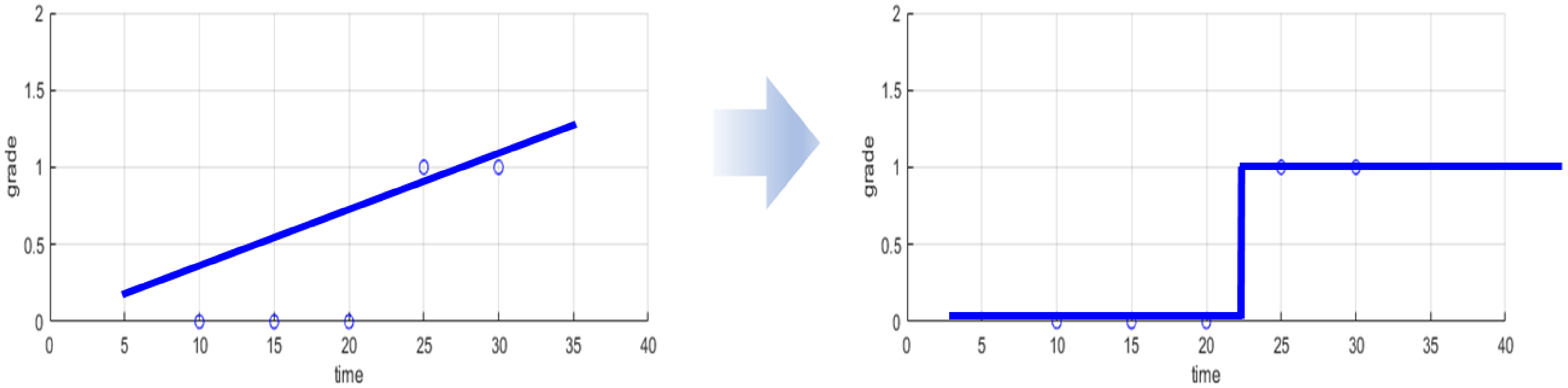
logistic regression

- logistic regression
  - linear regression을 classification 문제에서 사용하기 위한 방법론
  - 0/1같은 classification문제에서는 linear regression을 통해 최적의 함수를 찾을 수 없음
  - linear regression in classification
    - 시간에 따른 pass(1)/fail(0)에 대한 그래프
    - 아래 사진과 같이 30~90점이 모두 pass이면 정확한 error을 줄여도 정확한 함수를 그릴 수 없음



logistic regression

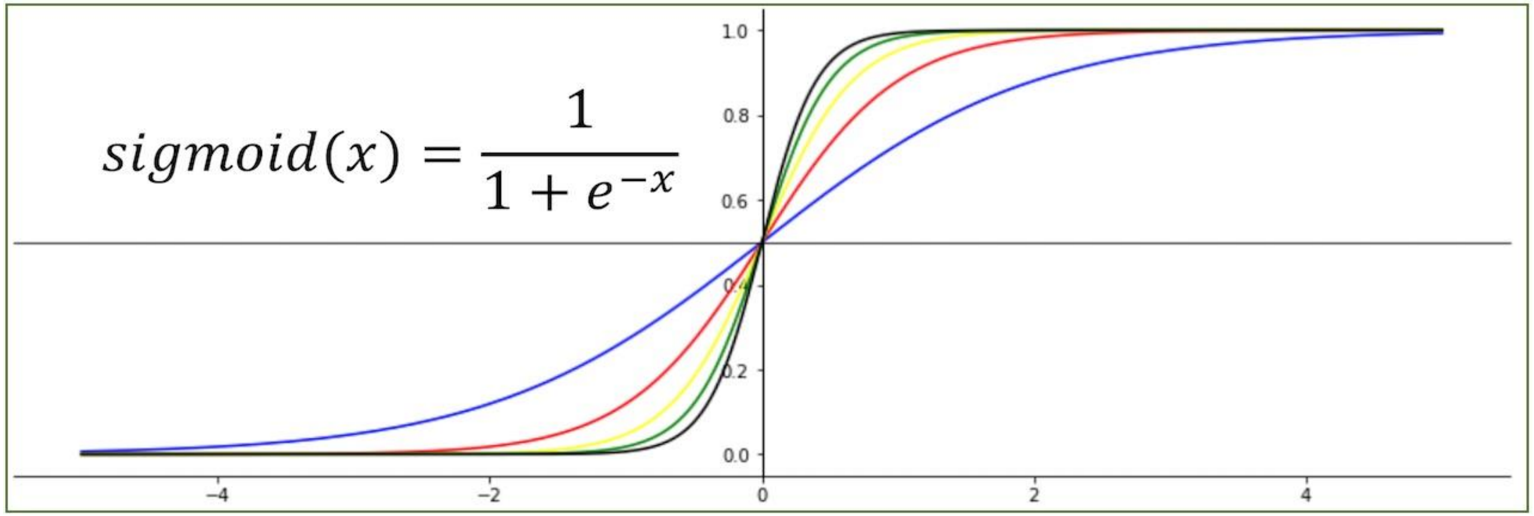
- logistic regression
  - classification을 위해 기존의 1차함수가 아니라 classification의 데이터 모형과 비슷한 모양을 가지는 함수를 채택
  - 다음과같이 1차함수가 어떠한 threshold를 넘으면 1 아니면 0이되게 설정할 수도 있다.



logistic regression

- logistic regression
  - logistic regression에서는 sigmoid function을 이용하여 나타냄
  - sigmoid function을 이용하여 classification에서도 비슷한 모양의 함수를 사용할 수 있음

$$y = \ sigmoid\,(Wx + b) \quad == \quad y = \frac{1}{1 - e^{-(WX+b)}}$$



logistic regression

- Cross-entropy loss
  - 기존 regression과 다르게 해당 class를 맞췄냐에 따른 cost function 사용
  - 정답이 1일때는 예측값(h(x))이 1에 가까워지도록
    0일때는 0에가까워지도록 하는 cost function사용(로그함수 참조)
  - cost function이 0에 가까워지면 더 많은 정답을 맞췄음을 의미

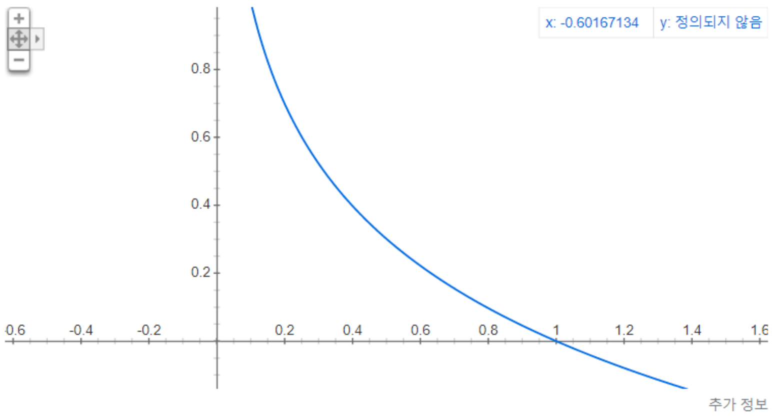
Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

Note: $y = 0$ or 1 always

-log(x) 그래프



logistic regression

Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

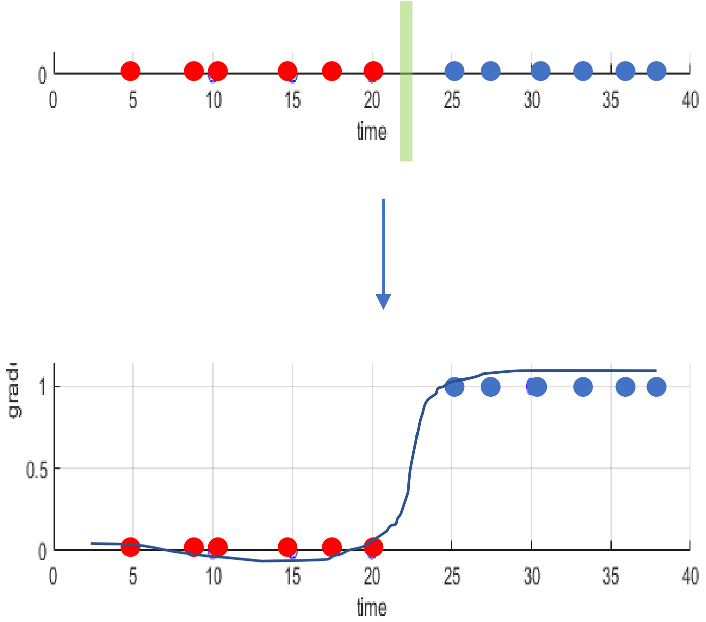
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

$||$

$$C(H(x), y) = -y \log(H(x)) - (1 - y) \log(1 - H(x))$$

logistic regression



- Hypothesis

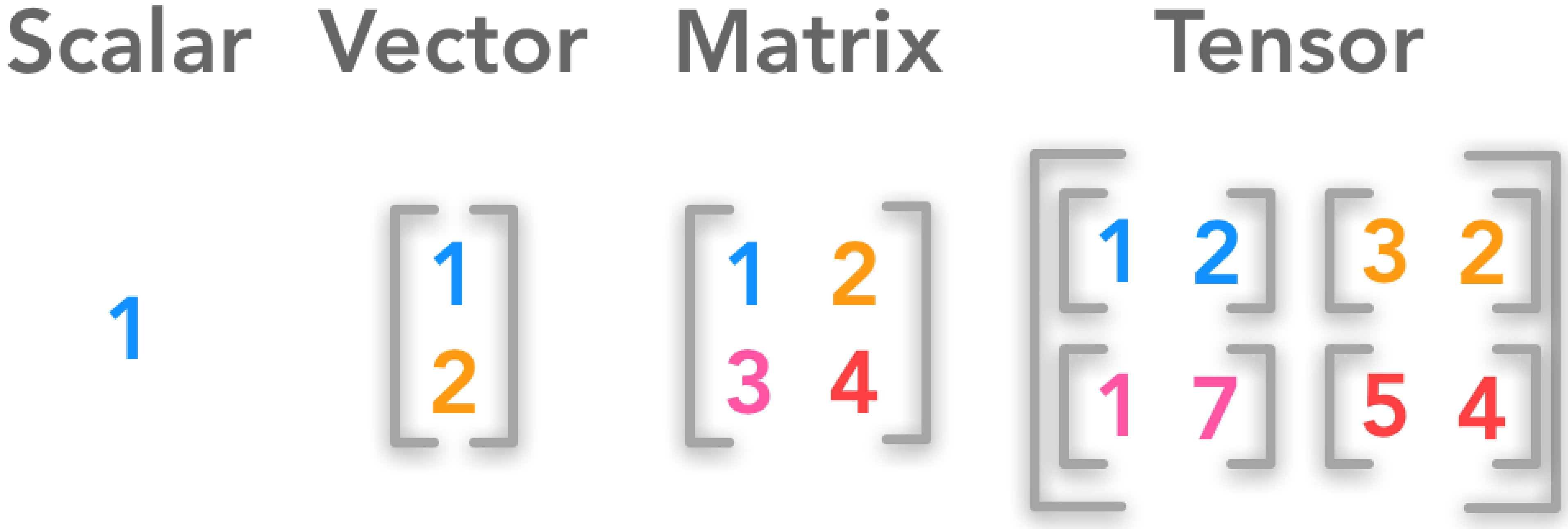
$$y = \ sigmoid\ (Wx + b) \quad == \quad y = \frac{1}{1 - e^{-(WX+b)}}$$

- Cost function

$$C(H(x), y) = \text{-y} \log\big(H(x)\big) - (1 - y)\log\big(1 - H(x)\big)$$

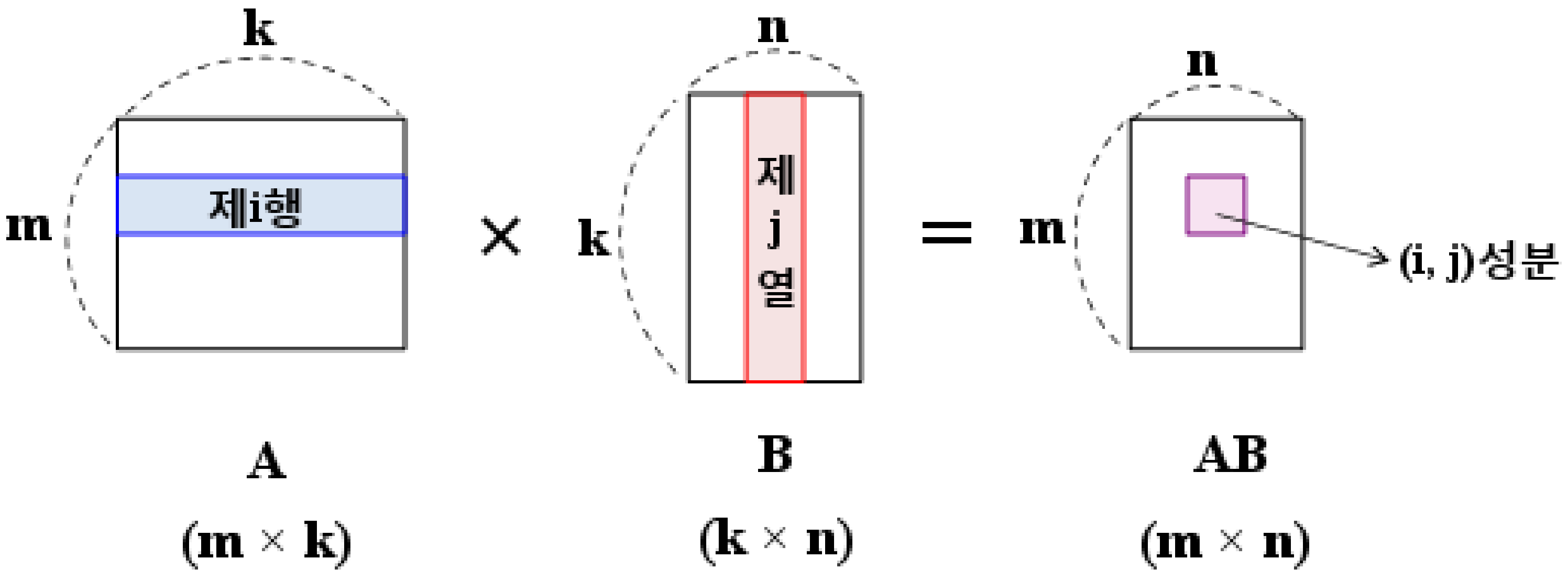
Tensor

- Tensor
  - 3차원 이상의 행렬
  - 딥러닝에서는 통상적으로 다 텐서라고 부름



Tensor

- 행렬곱



곱해지는 텐서들의 뒤(k),앞(k)의 차원이 같아야함　　정답의 차원은 뒤(k),앞(k)를 제외한 차원을 붙임

$$\underset{\text{Row}}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}} \cdot \underset{\text{Column}}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} = \begin{bmatrix} 6 \\ 15 \\ 24 \end{bmatrix}$$

1*1 + 2*1 + 3*1 = 6

Linear Regression with Tensor

- 행렬곱
  - Hypo: y = a * x + b * 1

[[x,1]] (1,2)   [[a,b]] (1,2)   [[y]] (1,1)

Transpose [a,b]

[[x,1]] (1,2) • [[a, b]] (2.1) = [[y]] (1,1)

Linear Regression with Tensor

- Batch processing
  - Hypo: y = a * x + b * 1
  - 여러 개의 데이터를 한번의 연산으로 처리하는 방법
  - Tensor를 이용해서 여러 개의 데이터를 한번에 처리할 수 있음
  - 메모리를 load하고 flush하는 과정에서 많은 cost가 들기에
    하나의 tensor load에 여러 개의 데이터를 처리할 수 있어서 효율적임
  - Data: x1,x2,x3 / target: y1,y2,y3

[[x1,1],
 [x2,1],
 [x3,1]]

(3,2)

•

[[a],
 [b]]

(2.1)

=

[[y]]

(3,1)

Linear Regression with Tensor

- Batch processing
  - Hypo: y = a * x + b * 1
  - 여러 개의 데이터를 한번의 연산으로 처리하는 방법
  - Tensor를 이용해서 여러 개의 데이터를 한번에 처리할 수 있음
  - 메모리를 load하고 flush하는 과정에서 많은 cost가 들기에
    하나의 tensor load에 여러 개의 데이터를 처리할 수 있어서 효율적임
  - 3개의 데이터 Data: x1,x2,x3 / target: y1,y2,y3

$$\underset{(3,2)}{\begin{matrix}[[x1,1], \\ [x2,1], \\ [x3,1]]\end{matrix}} \bullet \underset{(2.1)}{\begin{matrix}[[a], \\ [b]]\end{matrix}} = \underset{(3,1)}{\begin{matrix}[[y1], \\ [y2], \\ [y3]]\end{matrix}}$$

x1*a + 1*b = y1

Linear Regression with Tensor

- Batch processing
  - Hypo: y = a * x + b * 1
  - 여러 개의 데이터를 한번의 연산으로 처리하는 방법
  - Tensor를 이용해서 여러 개의 데이터를 한번에 처리할 수 있음
  - 메모리를 load하고 flush하는 과정에서 많은 cost가 들기에
    하나의 tensor load에 여러 개의 데이터를 처리할 수 있어서 효율적임
  - 3개의 데이터 Data: x1,x2,x3 / target: y1,y2,y3

[[x1,1],
[x2,1],
[x3,1]]
(3,2)

•

[[a],
[b]]
(2.1)

=

[[y1],
[y2],
[y3]]
(3,1)

x2*a + 1*b = y2

Linear Regression with Tensor

- Batch processing
  - Hypo: y = a * x + b * 1
  - 여러 개의 데이터를 한번의 연산으로 처리하는 방법
  - Tensor를 이용해서 여러 개의 데이터를 한번에 처리할 수 있음
  - 메모리를 load하고 flush하는 과정에서 많은 cost가 들기에
하나의 tensor load에 여러 개의 데이터를 처리할 수 있어서 효율적임
  - 3개의 데이터 Data: x1,x2,x3 / target: y1,y2,y3

$$\underset{(3,2)}{\begin{bmatrix} x1, 1 \\ x2, 1 \\ x3, 1 \end{bmatrix}} \bullet \underset{(2.1)}{\begin{bmatrix} a \\ b \end{bmatrix}} = \underset{(3,1)}{\begin{bmatrix} y1 \\ y2 \\ y3 \end{bmatrix}}$$

x3*a + 1*b = y3