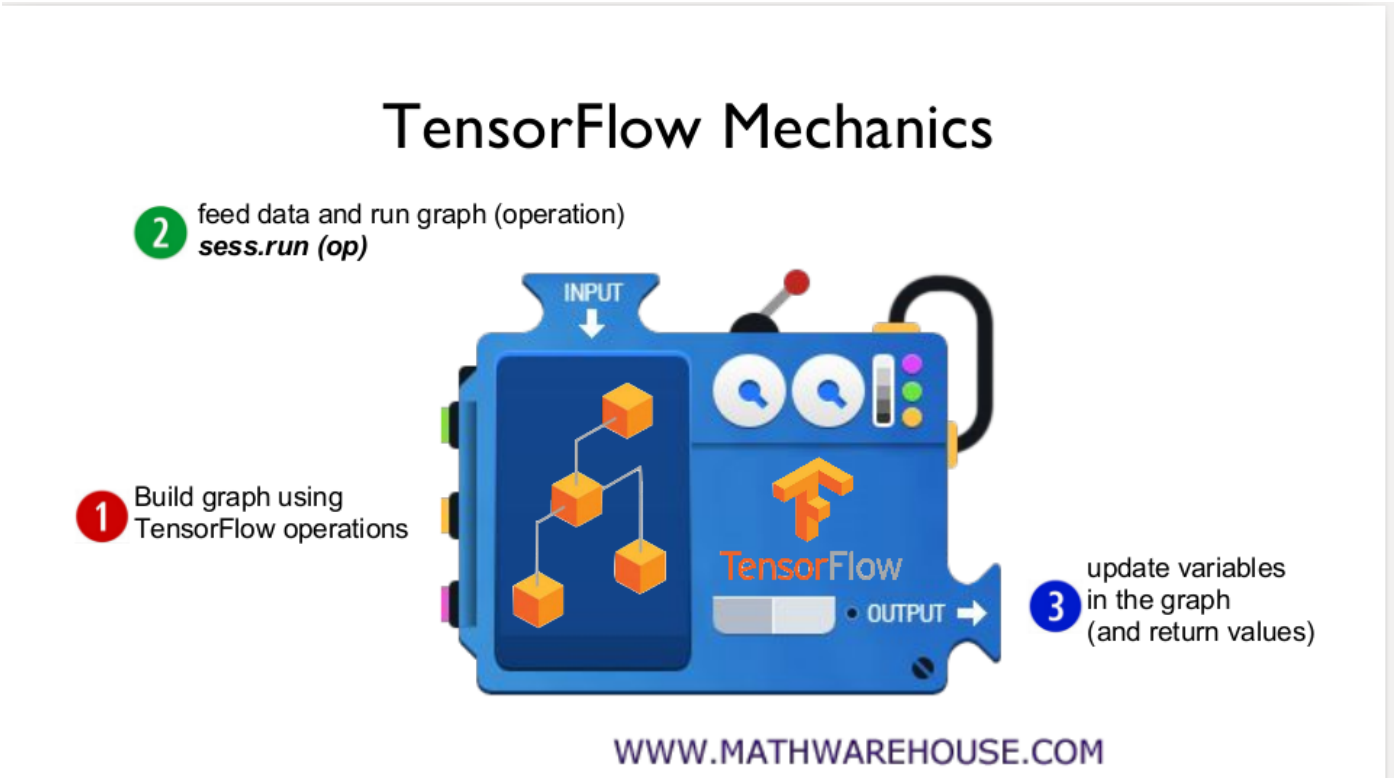
Deep Learning / Machine Learning Seminar

- Basic to MNIST -

박성현

How TensorFlow Works?

Build Graph → Feed Data → Run Graph → Update Variables

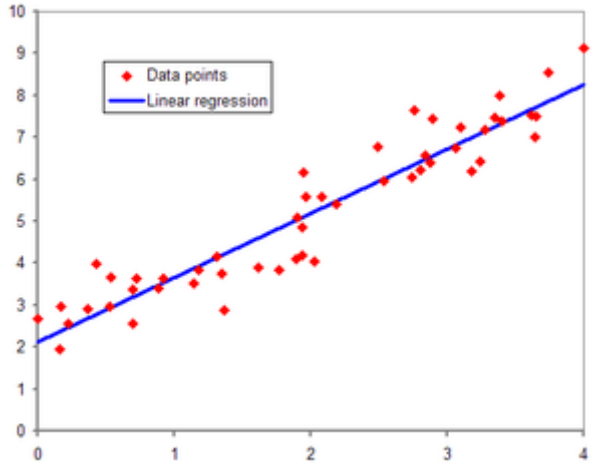


Basic TensorFlow

- Session
- constant
- Variable
- add / mul
- matmul
- initialize_all_variables

How to Predict Next Value?

- 기존의 데이터에서 다음값을 “어떻게” 추측할 것인가
- 가장 간단하게 하는 법 : linear regression [선형 회귀분석]
- $H(x) = W \cdot x + b$ // W : weight, b : bias
- 예측은 했는데 그대로 써도 될까?
- 실제값과의 차이를 나타내는 함수를 쓰자
- $C(x) = H(x) - y$
- $\text{Cost} = \frac{1}{m} \sum_{\square}^{last} (H(x) - y)^2$



How to Predict Next Value?

- Goal : minimize “cost value”
- cost(W,b) = $\frac{1}{m}\sum_{\square}^{last}(\mathrm{H}(x)-y)^2$

- MINIMIZE cost by changing Weight and bias

How It Works?

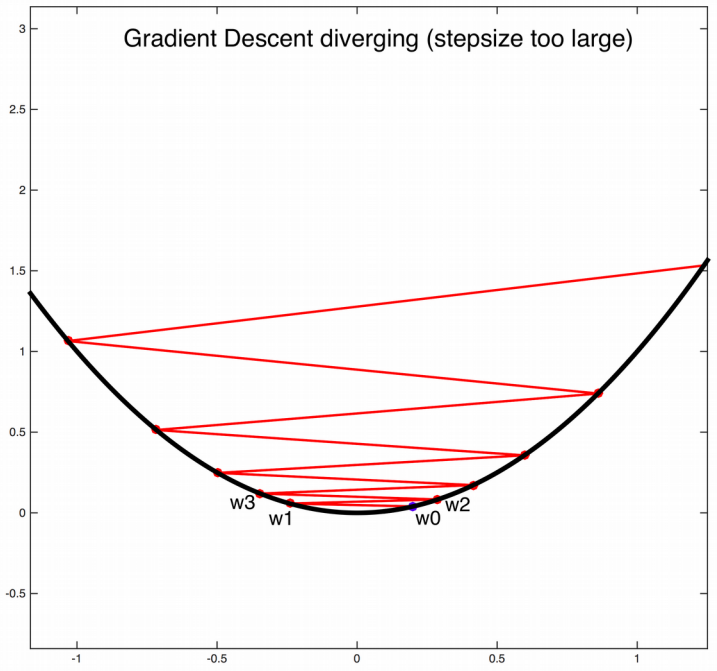
Gradient descent algorithm

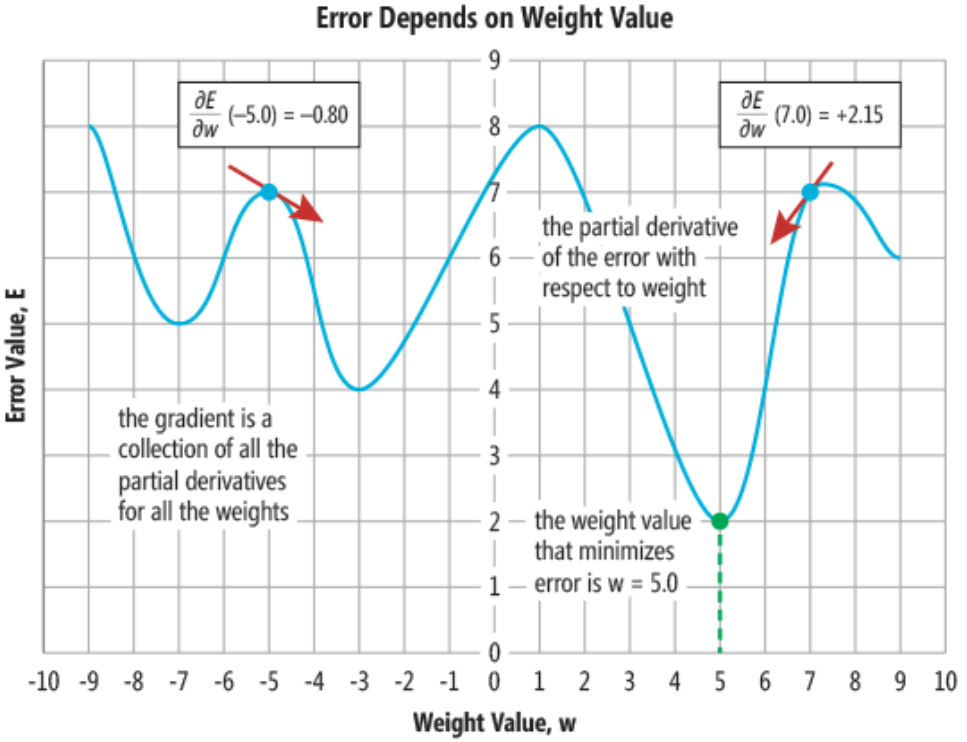
repeat until convergence {

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

(for $j = 1$ and $j = 0$)

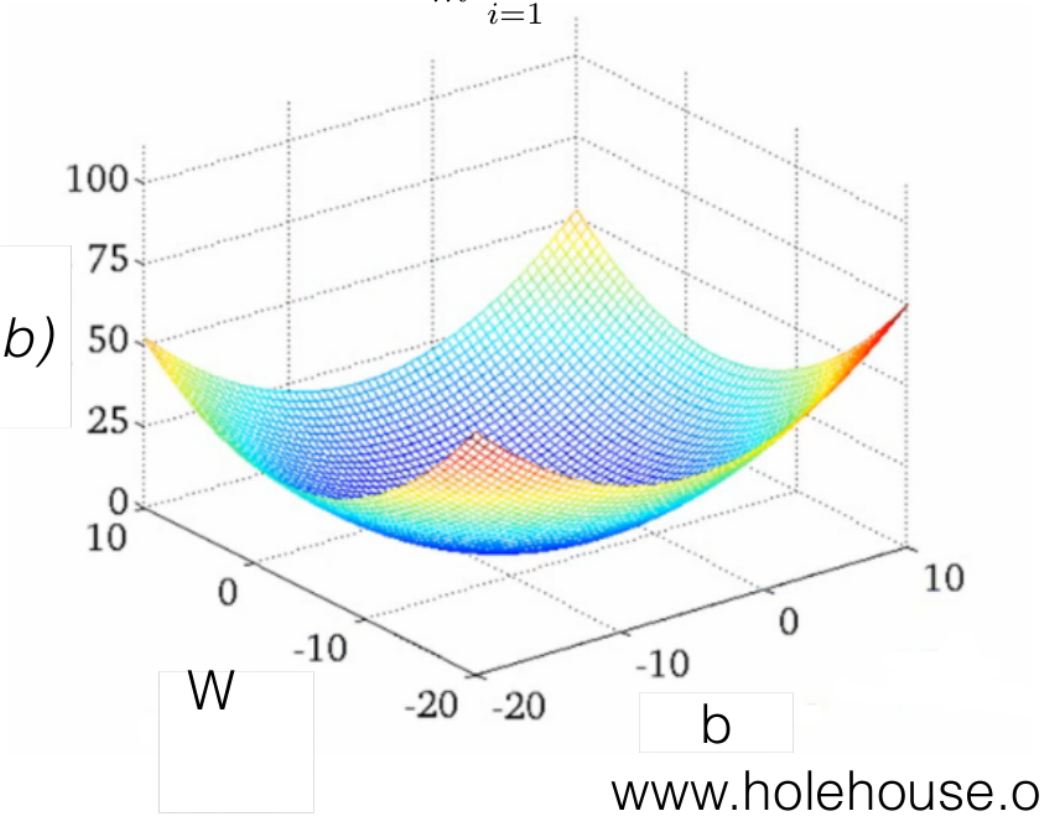
}





How It Works?

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



www.holehouse.org/mlclass/

Multi-Variable Linear Regression

- 말 그대로 “여러개”의 linear regression을 적용하는 것
- 왜?

  → 여러 개의 기준을 적용할 때 !!
- 그냥 하기엔 힘드니까, “행렬 연산”을 통해 간단하게 하자

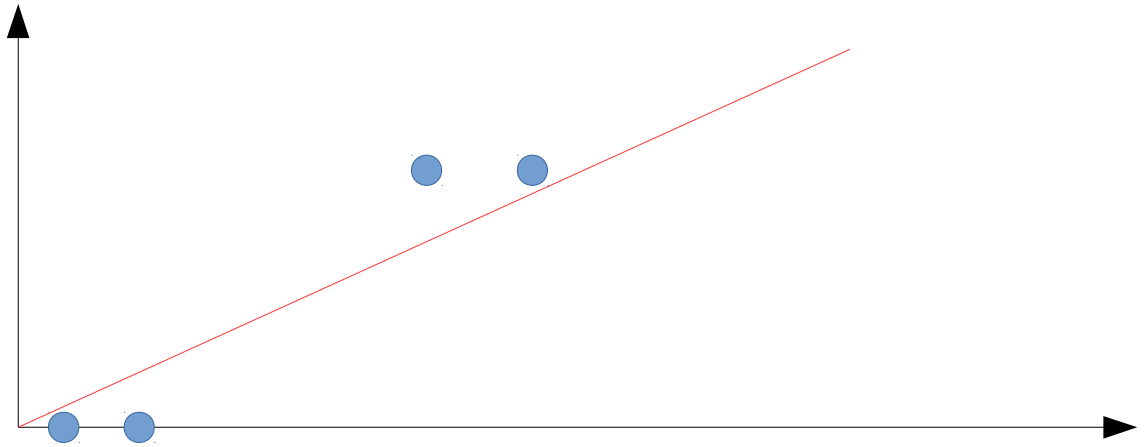
$$H(x_1, x_2, x_3, ..., x_n) = w_1x_1 + w_2x_2 + w_3x_3 + ... + w_nx_n + b$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1w_1 + x_2w_2 + x_3w_3 \end{pmatrix}$$

$$H(X) = XW$$

소올직히 linear 쓰레기 ㅇㅈ?ㅇㅇㅈ~

- 겨우 선형회귀분석하려고 TensorFlow를 쓸까 "NO"
- 컴퓨터에서 보통 상태를 0, 1로 표현을 함
- 이런 경우엔 어떻게 해야 되는가



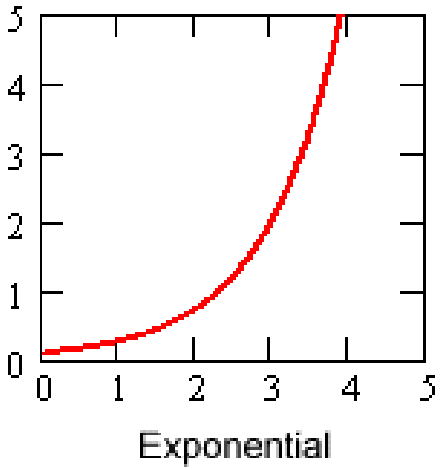
답은 “Logistic Regression” 이다

Logistic Function : $\frac{1}{1+e^{Wx+b}}$

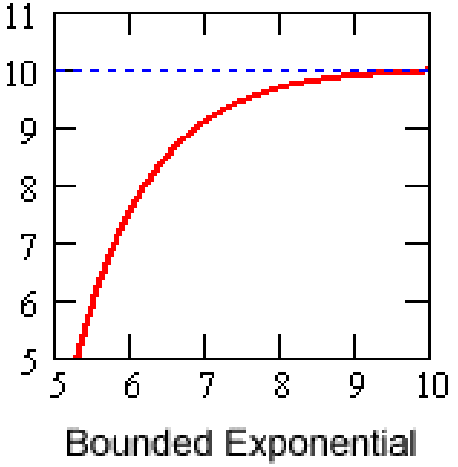
Cost Function $\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$

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

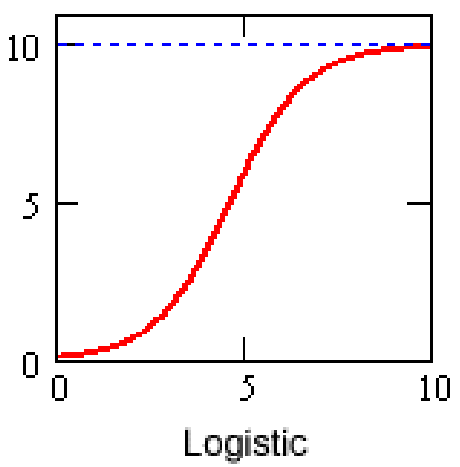
$L1(x) := \frac{1}{10} \cdot e^{x}$ $L2(x) := 10 \cdot \left(1 - 100 \cdot e^{-x}\right)$ $L(x) := \frac{10}{1 + 100 \cdot e^{-x}}$



+

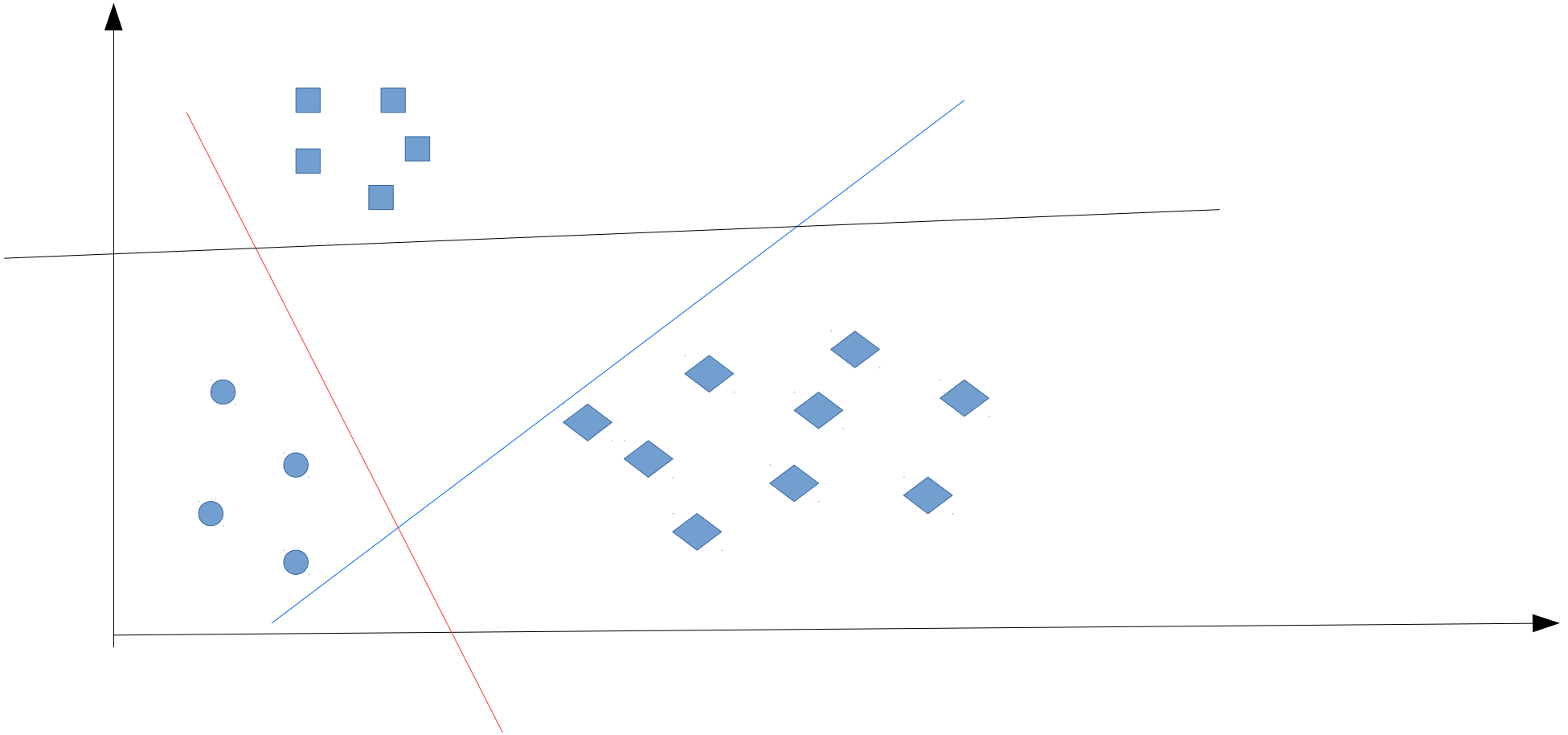


=



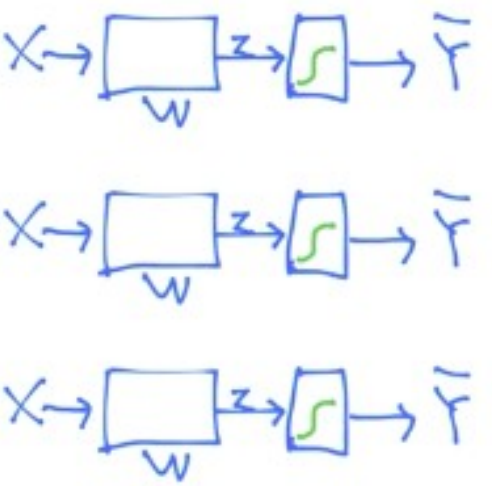
Multinomial Classification

- Only one boundary?
- No



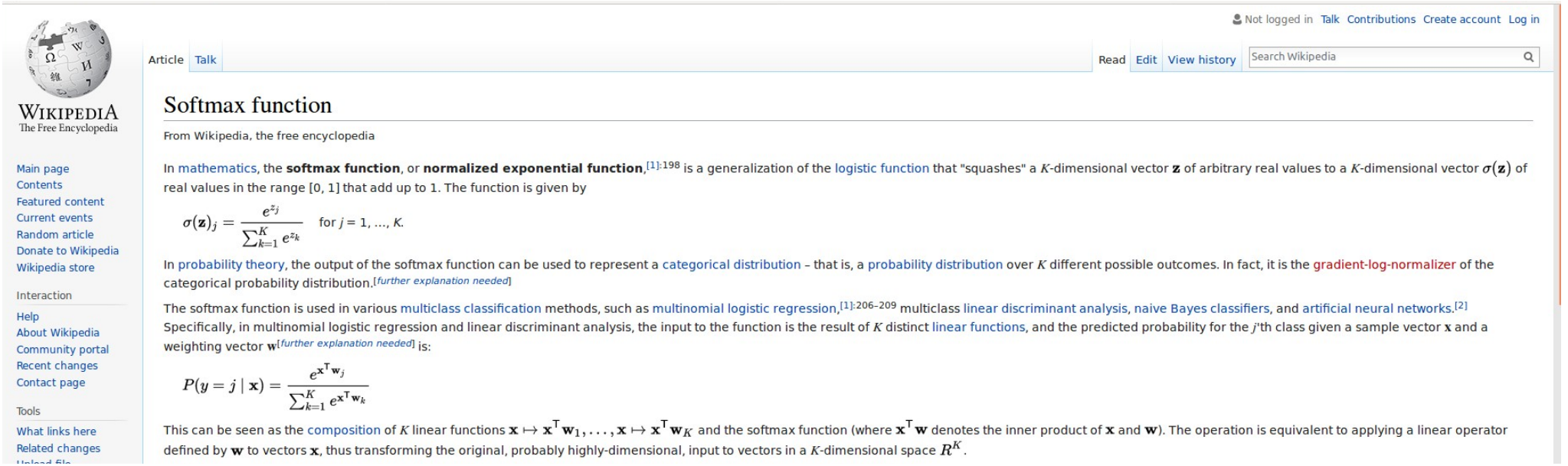
Multinomial Classification

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \bar{y}_A \\ \bar{y}_B \\ \bar{y}_C \end{bmatrix}$$



Softmax Function

- 0~1의 확률로 변환하는 함수



Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes

Softmax function

From Wikipedia, the free encyclopedia

In mathematics, the **softmax function**, or **normalized exponential function**,[1]:198 is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range [0, 1] that add up to 1. The function is given by

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

In probability theory, the output of the softmax function can be used to represent a categorical distribution – that is, a probability distribution over K different possible outcomes. In fact, it is the gradient-log-normalizer of the categorical probability distribution.[*further explanation needed*]

The softmax function is used in various multiclass classification methods, such as multinomial logistic regression,[1]:206–209 multiclass linear discriminant analysis, naive Bayes classifiers, and artificial neural networks.[2] Specifically, in multinomial logistic regression and linear discriminant analysis, the input to the function is the result of K distinct linear functions, and the predicted probability for the j'th class given a sample vector $\mathbf{x}$ and a weighting vector $\mathbf{w}$[*further explanation needed*] is:

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

This can be seen as the composition of K linear functions $\mathbf{x} \mapsto \mathbf{x}^\mathsf{T}\mathbf{w}_1, \ldots, \mathbf{x} \mapsto \mathbf{x}^\mathsf{T}\mathbf{w}_K$ and the softmax function (where $\mathbf{x}^\mathsf{T}\mathbf{w}$ denotes the inner product of $\mathbf{x}$ and $\mathbf{w}$). The operation is equivalent to applying a linear operator defined by $\mathbf{w}$ to vectors $\mathbf{x}$, thus transforming the original, probably highly-dimensional, input to vectors in a K-dimensional space R^K.

One-Hot Encoding

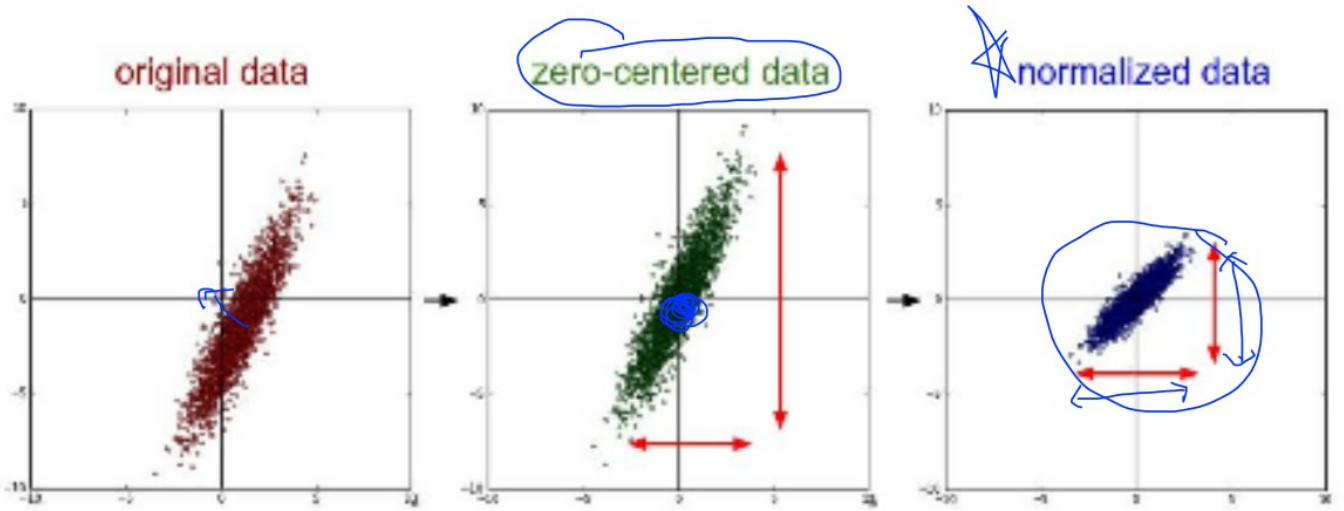
- Why?
- Logistic function or Linear Regression → ex] 2.5, 3.4, … 그냥 숫자
- 의도하지 않은 우위 [priority]를 부여하는 경우가 생길 수도 있음
- One-Hot Encoding : ex] [1,0,0,0] [0,1,0,0]

Practice Lab

- 1. Basic TensorFlow
- 2. Linear / Logistic Regression
- 3. XOR solve

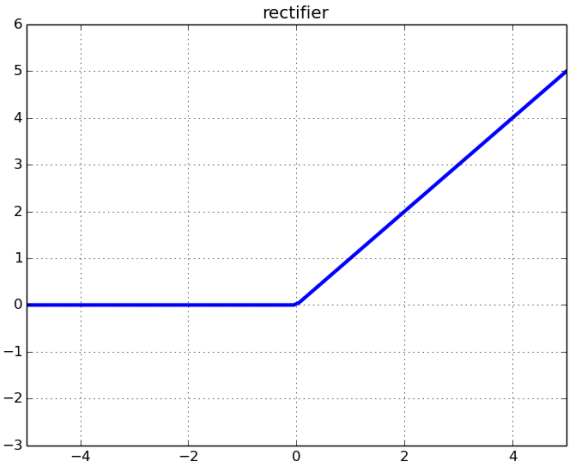
Data Preprocessing

Data (X) preprocessing for gradient descent



ReLU

- ReLU = Rectified Linear Unit
- Why ReLU?
- To solve XOR problem !



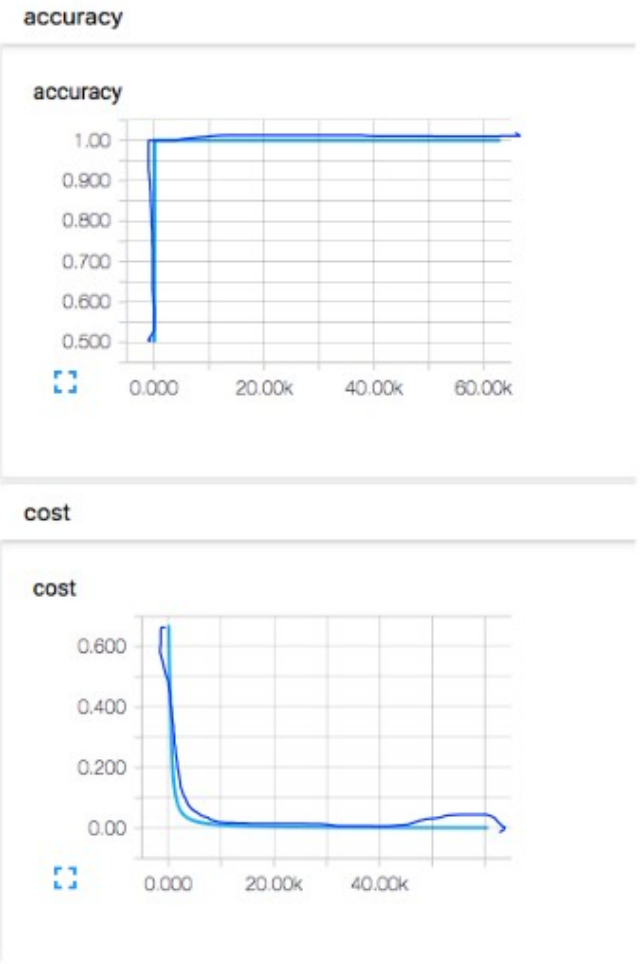
- Result of 9 layers of sigmoid for XOR

Tensorboard
Cost &
Accuracy

ReLU

- ReLU = Rectified Linear Unit
- Why ReLU?
- To solve XOR problem !

- Result of 9 layers of “ReLU” for XOR



Practice Lab

- 1. Basic TensorFlow
- 2. Linear / Logistic Regression
- 3. XOR solve