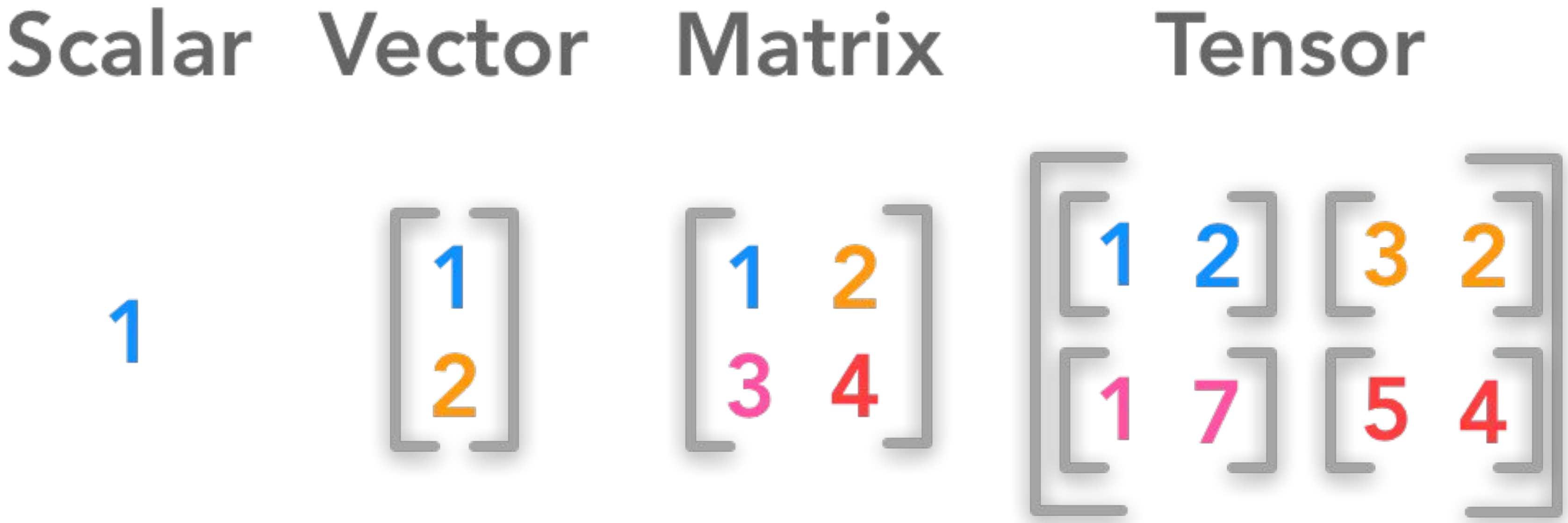
FinanceData.KR

머신러닝 딥러닝 핵심 개념

이승준

FinanceData.KR 2021-2024

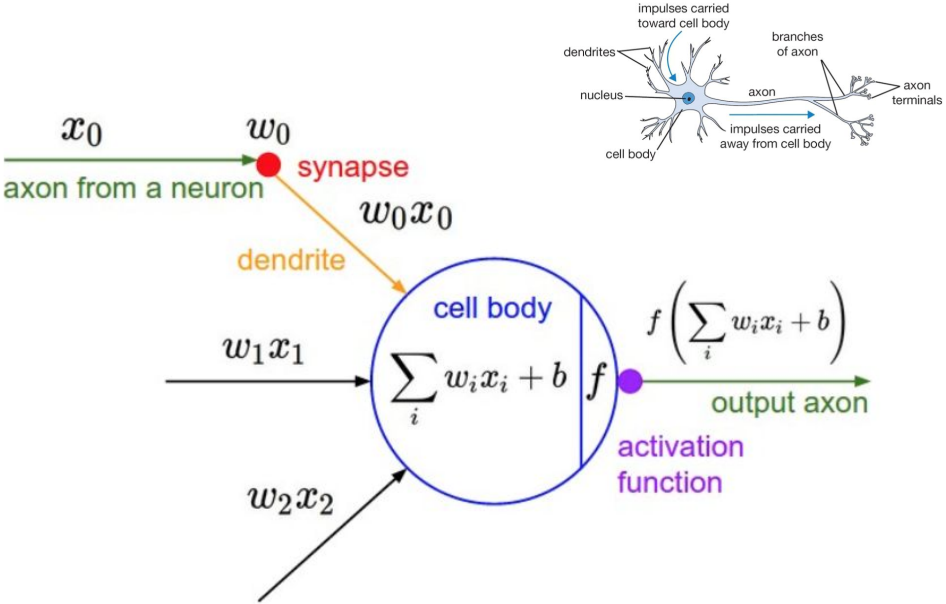


Scalar Vector Matrix Tensor

$$1 \qquad \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$$

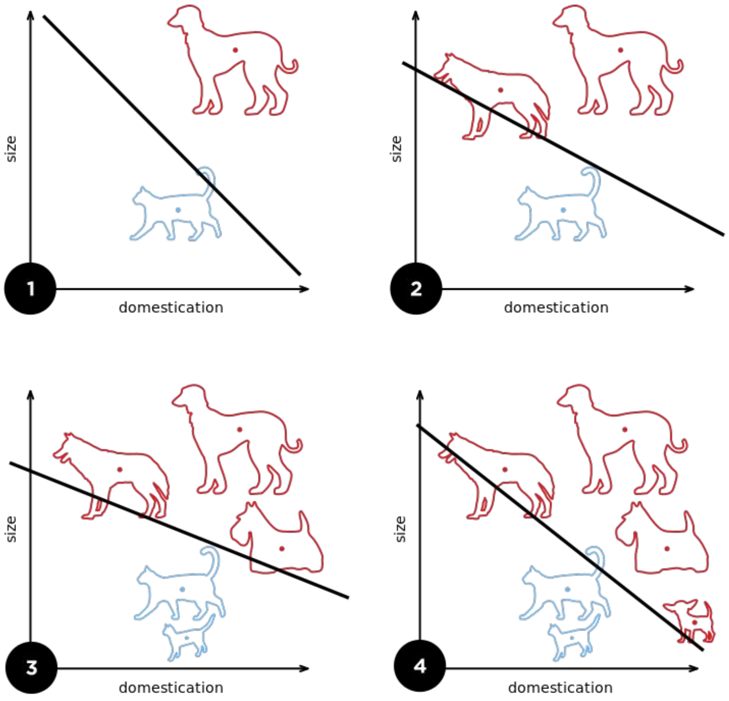
https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.1-Scalars-Vectors-Matrices-and-Tensors/

Perceptron



Update weights: $\vec{w} := \vec{w} + y^{(t)} \cdot \vec{x}^{(t)}$

http://bit.ly/2Zaijqz

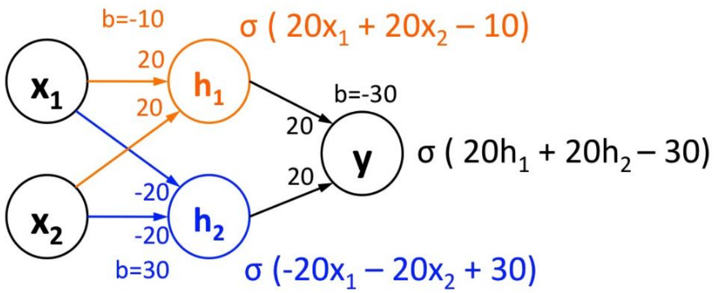


https://en.wikipedia.org/wiki/Perceptron

XOR Problem

https://youtu.be/kNPGXgzxoHw

MLP can solve XOR



$\sigma(20x_1 + 20x_2 - 10)$

$\sigma(20h_1 + 20h_2 - 30)$

$\sigma(-20x_1 - 20x_2 + 30)$

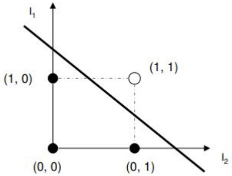
$\sigma(20*0 + 20*0 - 10) \approx 0$
$\sigma(20*1 + 20*1 - 10) \approx 1$
$\sigma(20*0 + 20*1 - 10) \approx 1$
$\sigma(20*1 + 20*0 - 10) \approx 1$

$\sigma(-20*0 - 20*0 + 30) \approx 1$
$\sigma(-20*1 - 20*1 + 30) \approx 0$
$\sigma(-20*0 - 20*1 + 30) \approx 1$
$\sigma(-20*1 - 20*0 + 30) \approx 1$

$\sigma(20*0 + 20*1 - 30) \approx 0$
$\sigma(20*1 + 20*0 - 30) \approx 0$
$\sigma(20*1 + 20*1 - 30) \approx 1$
$\sigma(20*1 + 20*1 - 30) \approx 1$

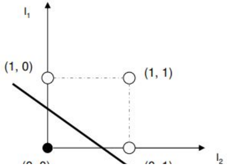
AND

I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



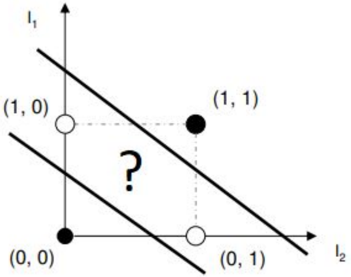
OR

I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1

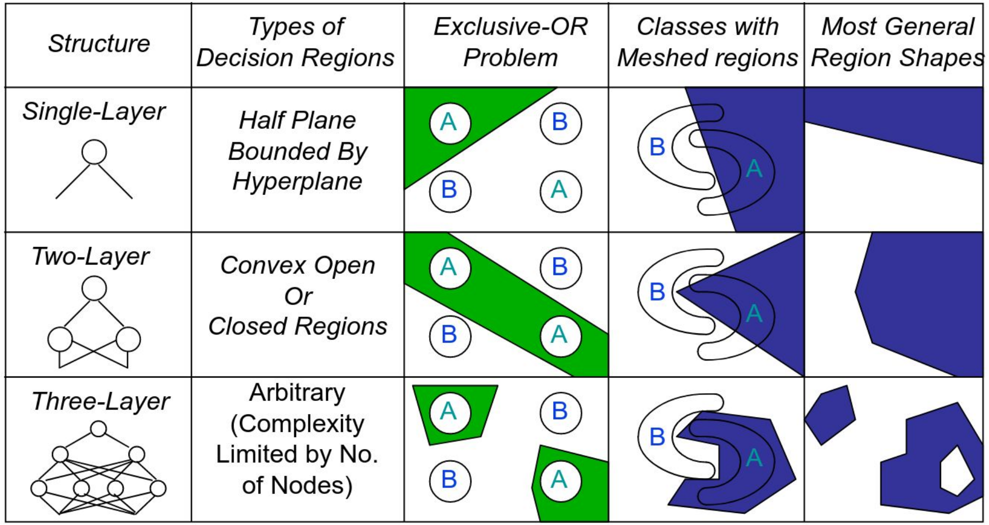


XOR

I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



http://bit.ly/2PLnzhi

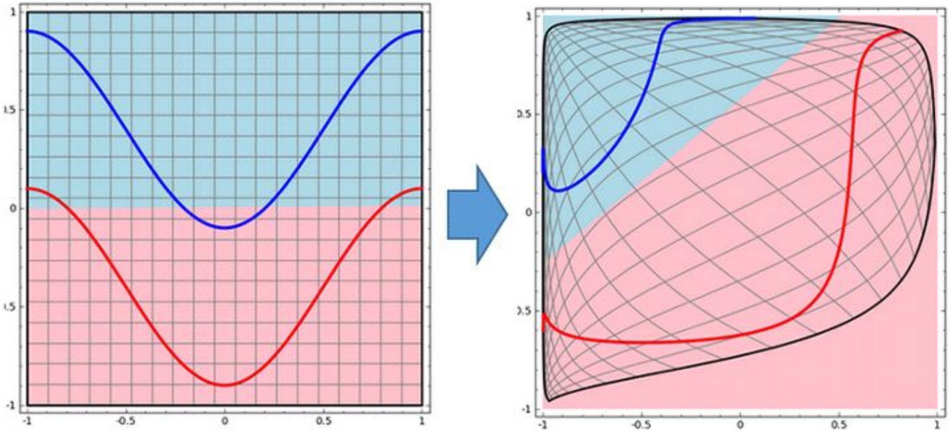


Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer	Half Plane Bounded By Hyperplane			
Two-Layer	Convex Open Or Closed Regions			
Three-Layer	Arbitrary (Complexity Limited by No. of Nodes)			

http://bit.ly/2MhOyi2

Representation Learning

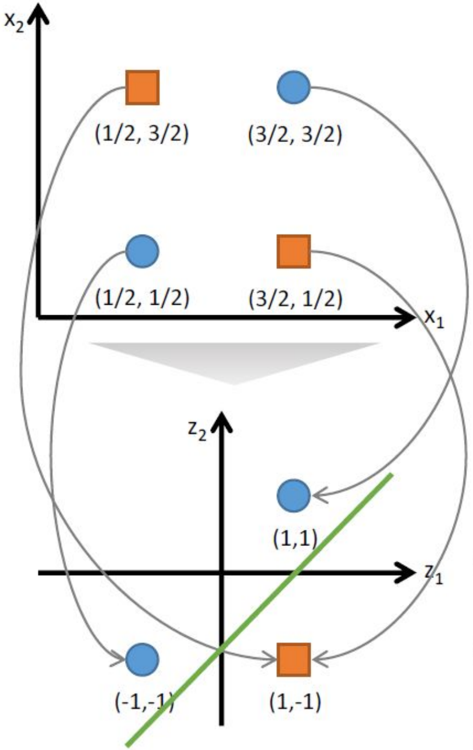
discover the representations needed for feature detection or classification from raw data (=**Feature learning**)



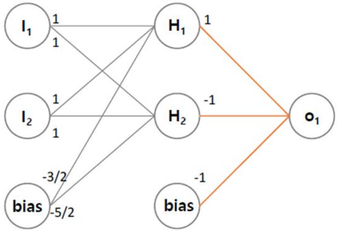
뉴럴네트워크 (= representation learner)

선형으로 분리할 수 없는 데이터를 선형 분리가 가능하게끔 데이터가 변형

http://bit.ly/2EJrOUi

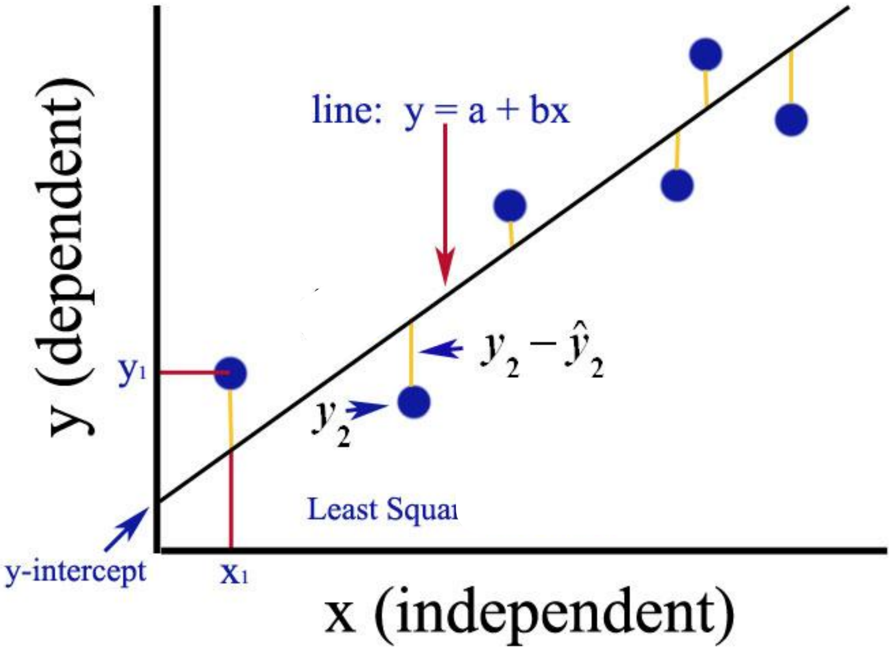


http://bit.ly/2Qcycsg



	x_1	x_2	h_1	z_1	h_2	z_2	o_1	z
●	1/2	1/2	-1/2	-1	-3/2	-1	-1	-1
🟧	3/2	1/2	1/2	1	-1/2	-1	1	1
🟧	1/2	3/2	1/2	1	-1/2	-1	1	1
●	3/2	3/2	3/2	1	1/2	1	-1	-1

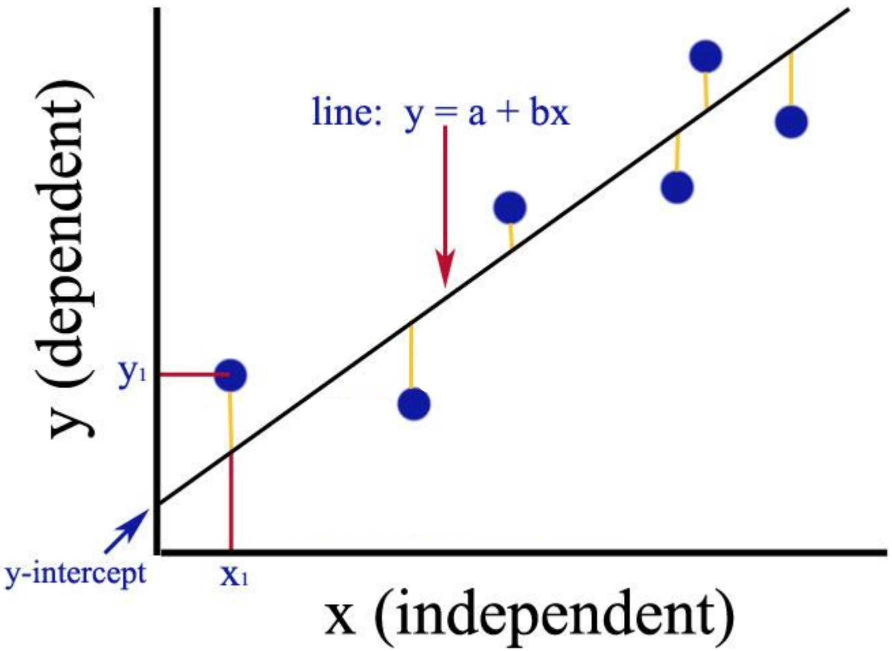
Loss, Error, Cost



- y: 실제값, $\hat{y}$: 예측치
- $(y-\hat{y})$: 실제값과 예측치의 차이

- Error - population
- Residual - sample

Model

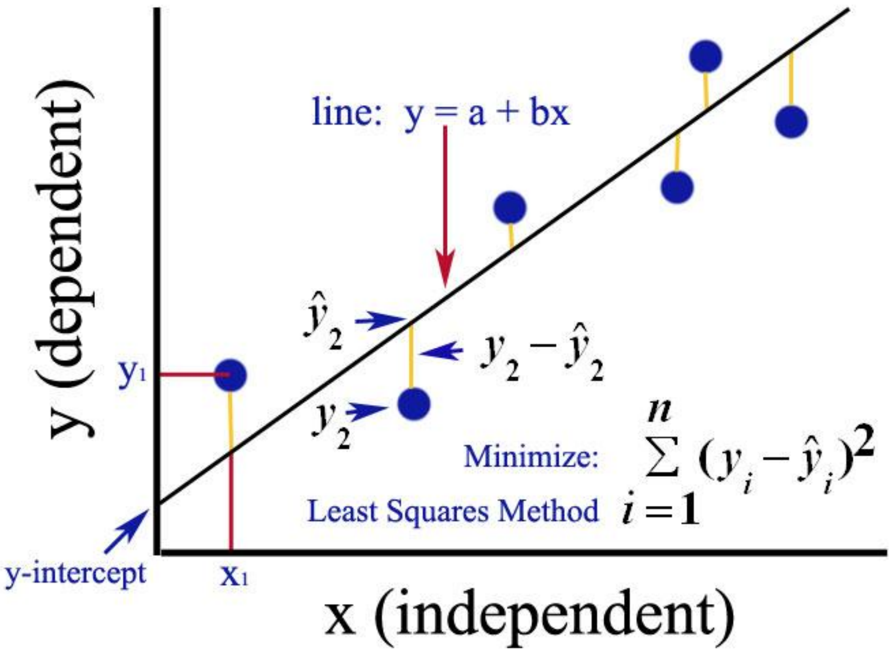


- $y = a + bx$: 우리의 모델 (a,b: 파라미터)

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

$$H(x) = Wx + b$$

Cost function



$$H(x) = Wx + b$$

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

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

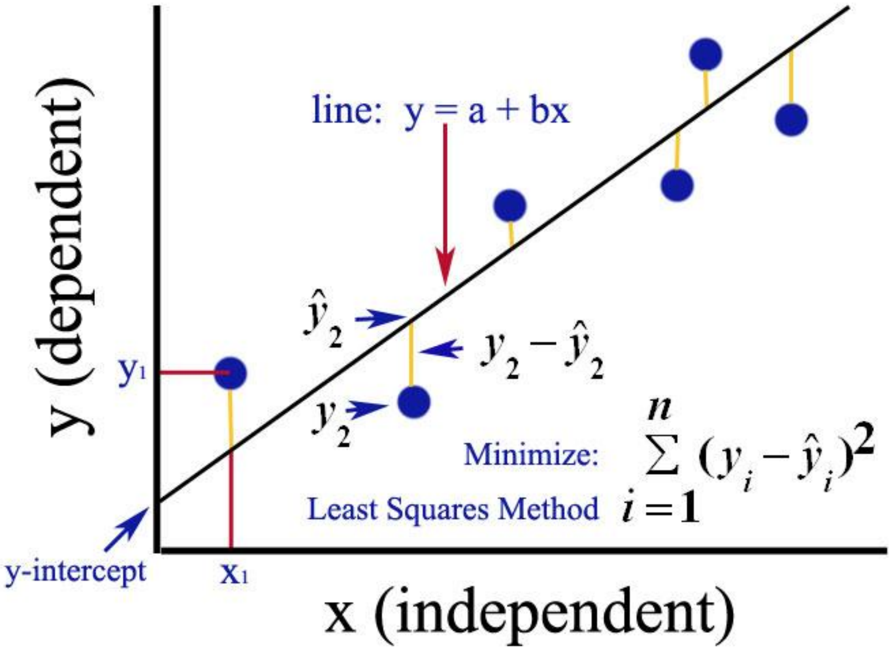
Parameters: $\underline{\theta_0, \theta_1}$

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Linear Regression

"Regression toward the mean"

Sir Francis Galton (1822 ~ 1911)



$$H(x) = Wx + b$$

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

Goal: $\underset{W,b}{minimize}\, cost(W, b)$

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

Parameters: $\underline{\theta_0, \theta_1}$

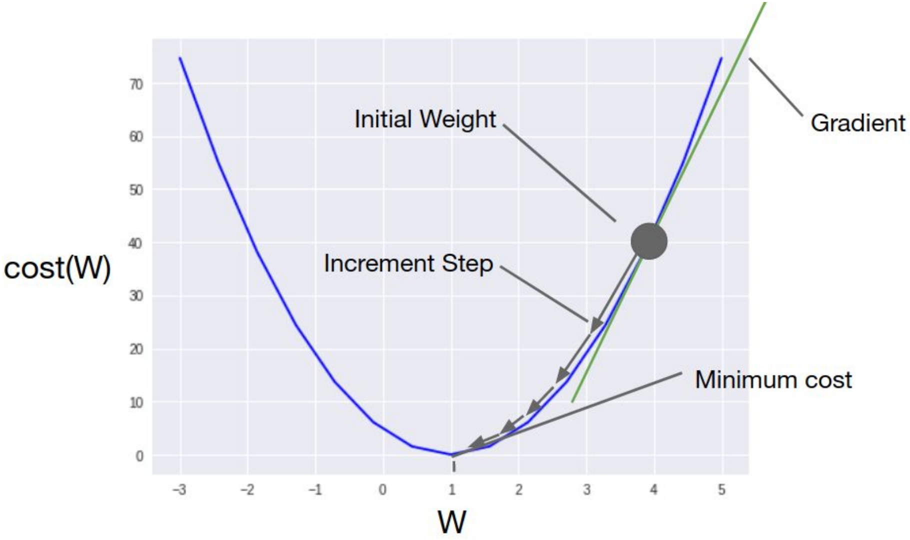
Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

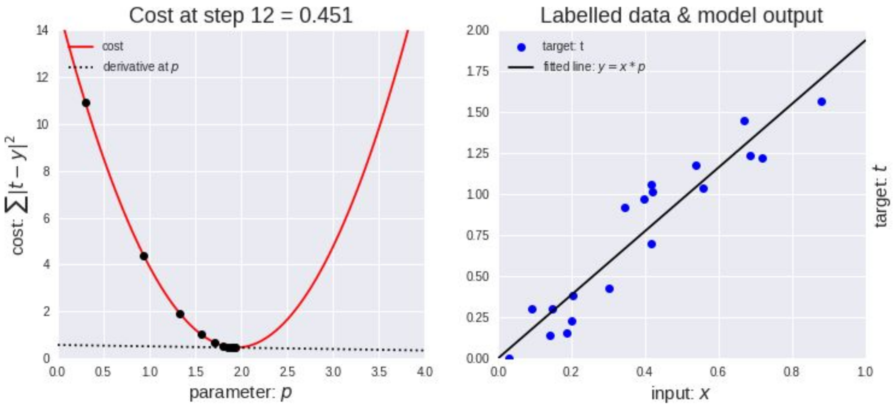
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\, J(\theta_0, \theta_1)$

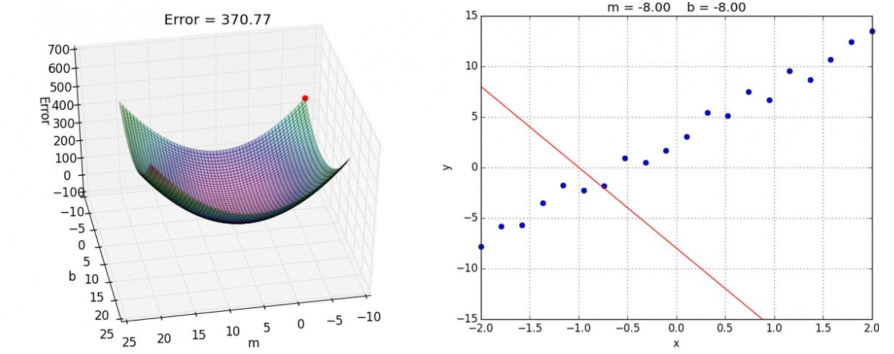
Gradient descent

$$cost(W,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(H(x_i) - y_i\right)^2$$

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

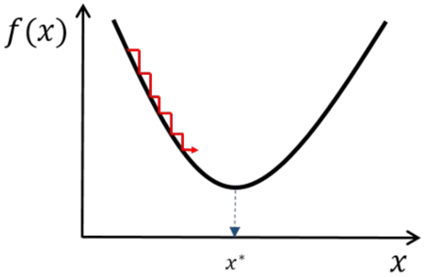




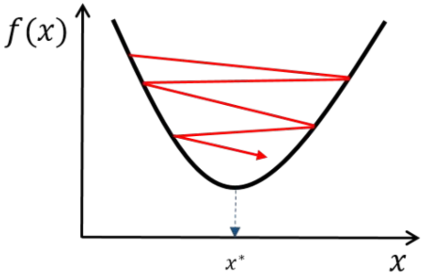


https://alykhantejani.github.io/a-brief-introduction-to-gradient-descent/

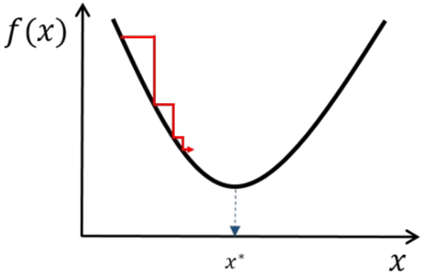
Learning rate



Too small: converge very slowly

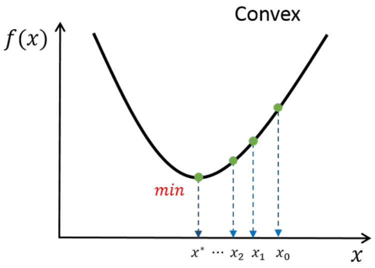


Too big: overshoot and even diverge

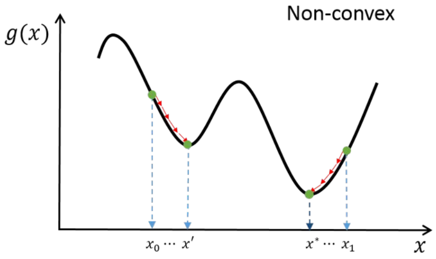


Reduce size over time

http://bit.ly/2QAKrPK


Convex

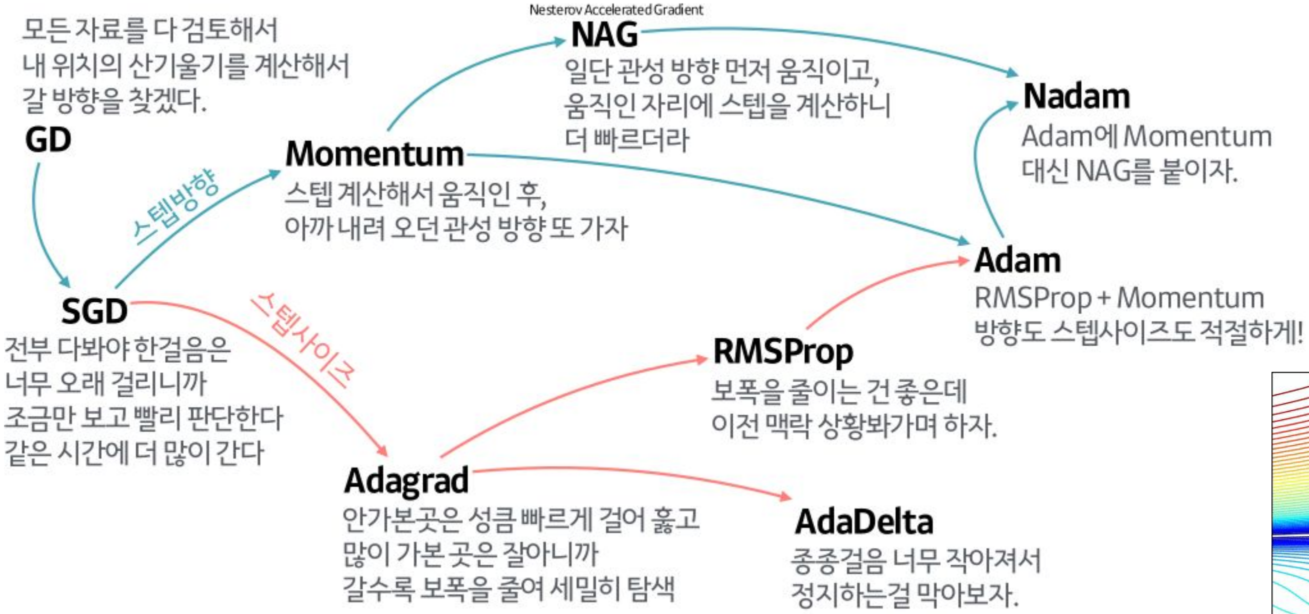
Any local minimum is a global minimum


Non-convex

Multiple local minima may exist

Optimizer

최저점을 찾기 위한 스텝의 방향과 크기를 결정하는 방법들



GD
모든 자료를 다 검토해서
내 위치의 산기울기를 계산해서
갈 방향을 찾겠다.

SGD
전부 다봐야 한걸음은
너무 오래 걸리니까
조금만 보고 빨리 판단한다
같은 시간에 더 많이 간다

Momentum
스텝 계산해서 움직인 후,
아까 내려 오던 관성 방향 또 가자

Nesterov Accelerated Gradient
NAG
일단 관성 방향 먼저 움직이고,
움직인 자리에 스텝을 계산하니
더 빠르더라

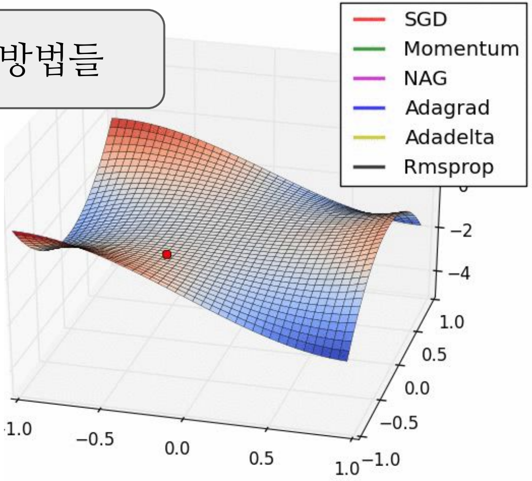
Nadam
Adam에 Momentum
대신 NAG를 붙이자.

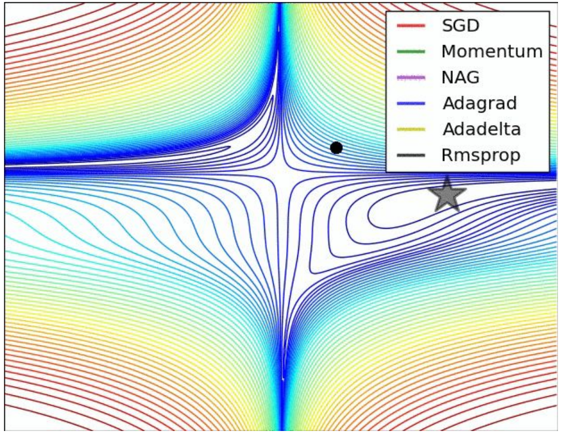
Adam
RMSProp + Momentum
방향도 스텝사이즈도 적절하게!

RMSProp
보폭을 줄이는 건 좋은데
이전 맥락 상황봐가며 하자.

Adagrad
안가본곳은 성큼 빠르게 걸어 훑고
많이 가본 곳은 잘아니까
갈수록 보폭을 줄여 세밀히 탐색

AdaDelta
종종걸음 너무 작아져서
정지하는걸 막아보자.

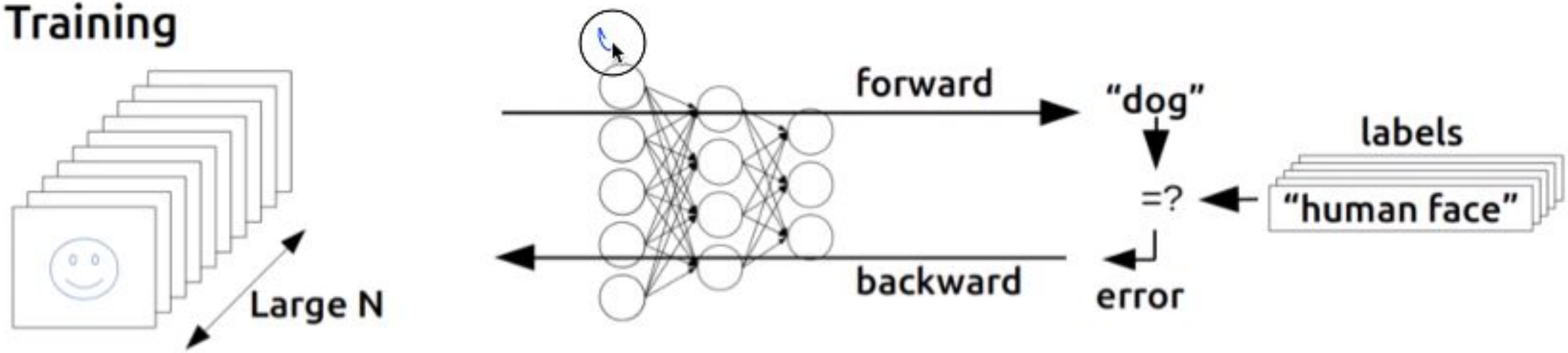




https://www.slideshare.net/yongho/ss-79607172

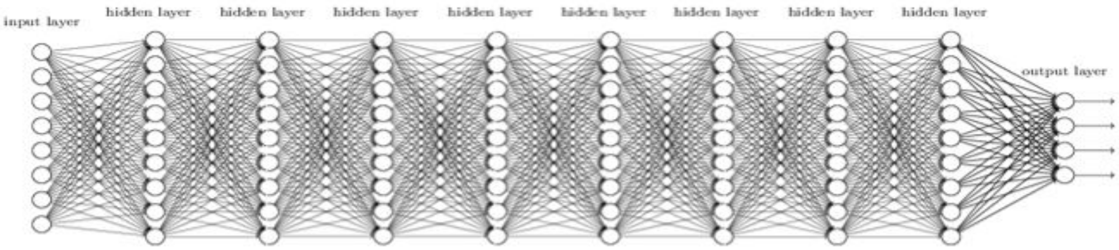
https://ruder.io/optimizing-gradient-descent/

Back propagation

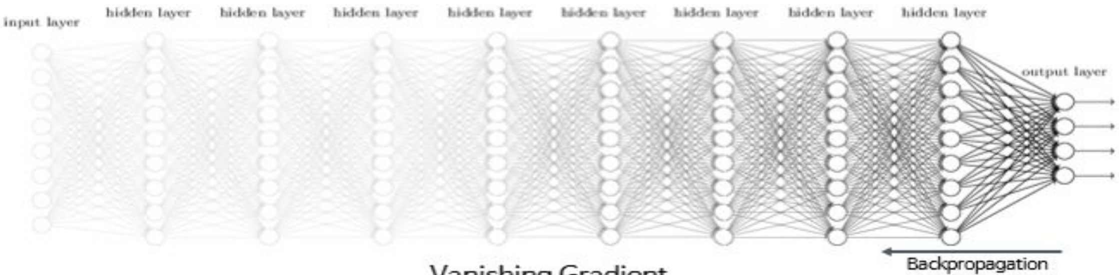


(1974, 1982 by Paul Werbos, 1986 by Hinton)

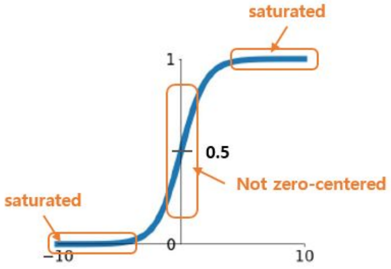
Vanishing Gradient



Deep Neural Network

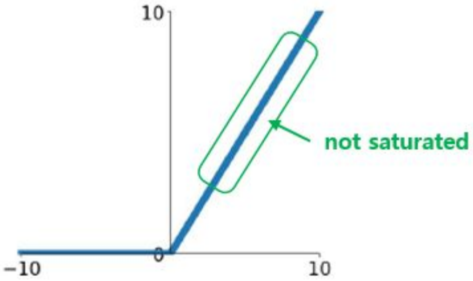


Vanishing Gradient



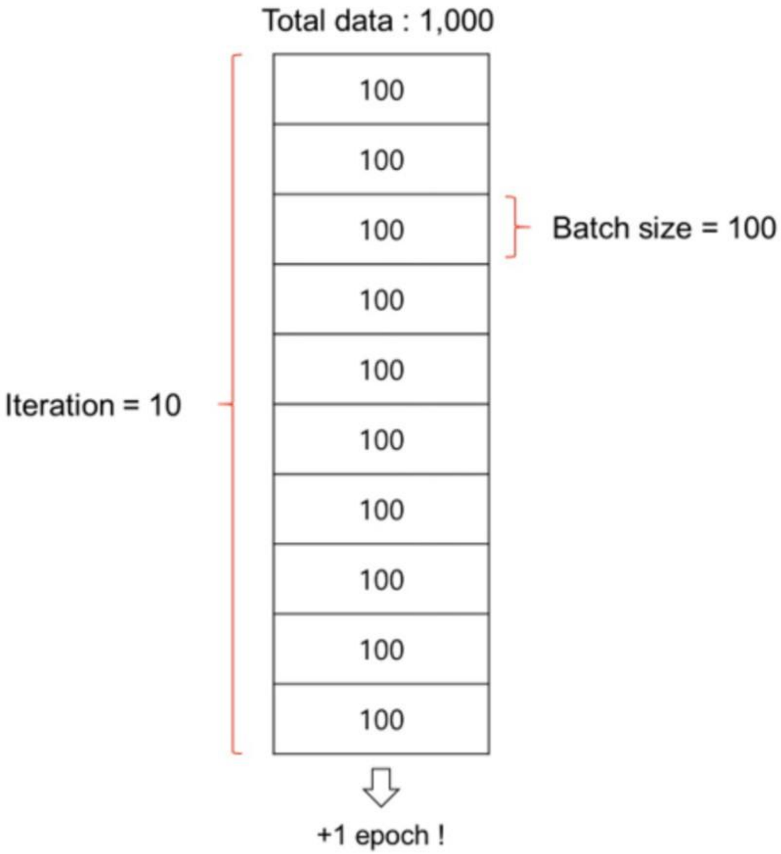
Sigmoid Problems

1. **saturated**: Gradient Kill
2. **Not zero-centered**: Slow Performance



ReLU (Rectified Linear Unit)

Epoch, Batch size, Iterations



Total data : 1,000

100

100

100

100

100

100

100

100

100

100

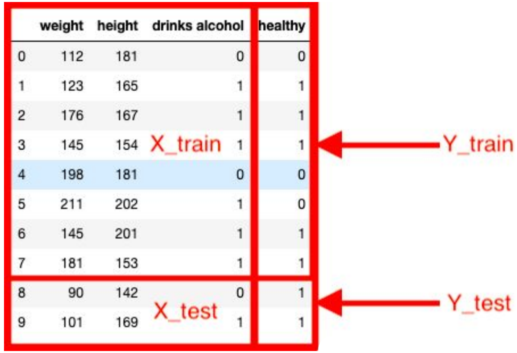
Batch size = 100

Iteration = 10

+1 epoch !

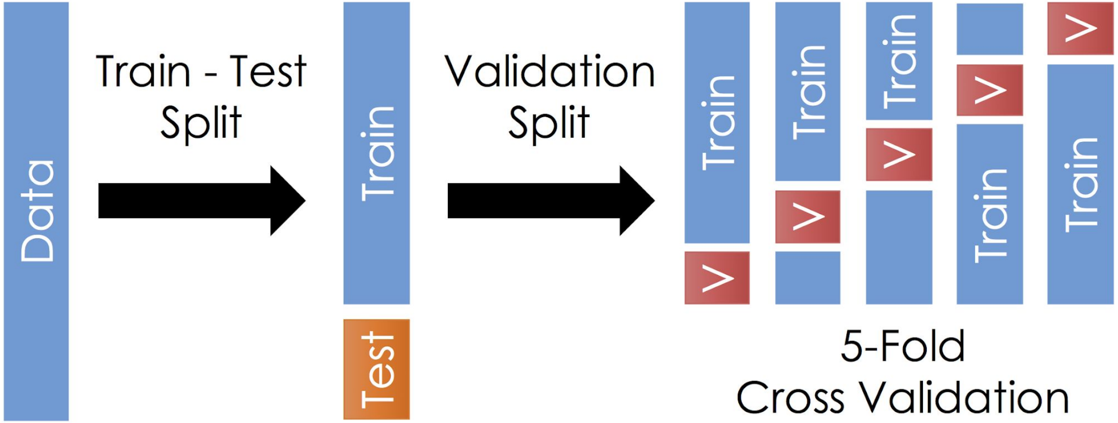
Train, Test, Validation

Train-Test Split



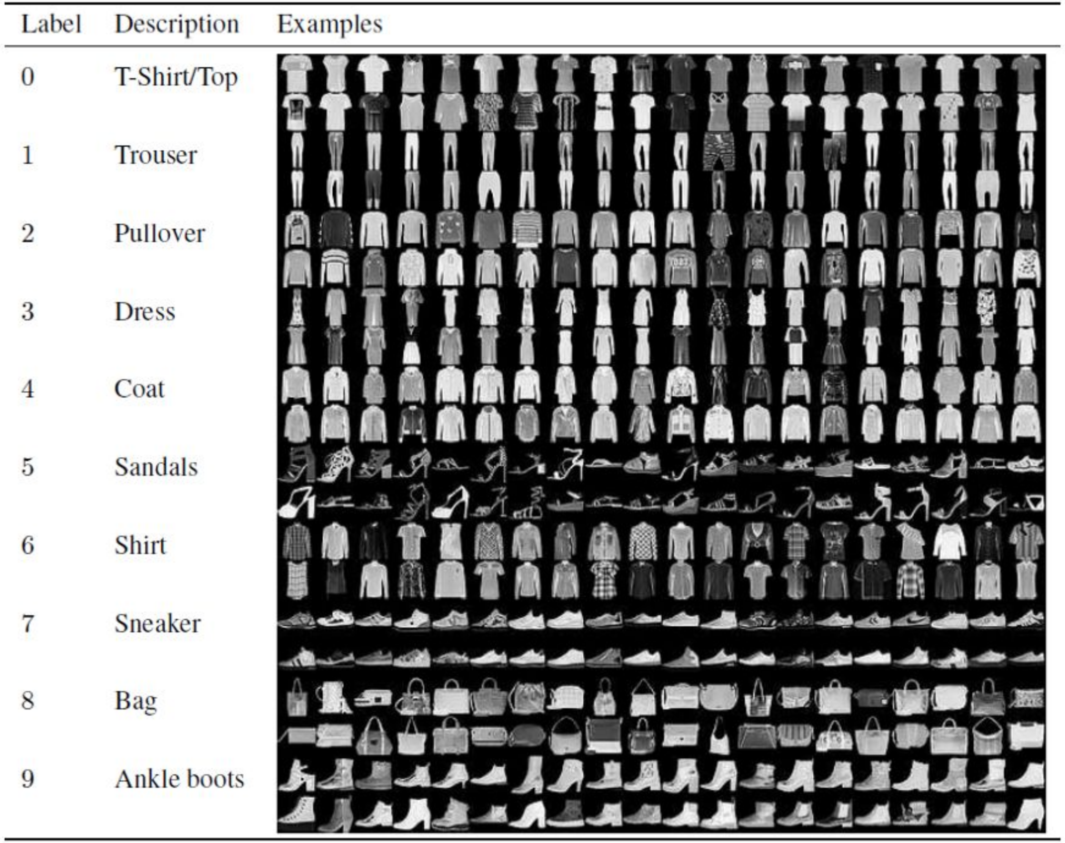
	weight	height	drinks alcohol	healthy
0	112	181	0	0
1	123	165	1	1
2	176	167	1	1
3	145	154	1	1
4	198	181	0	0
5	211	202	1	0
6	145	201	1	1
7	181	153	1	1
8	90	142	0	1
9	101	169	1	1

Validation Split

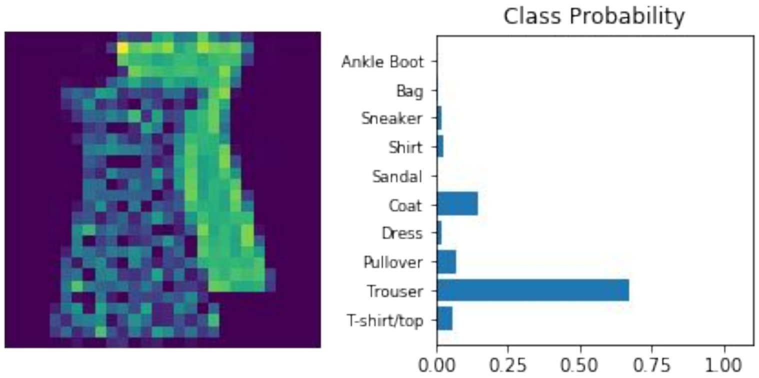


5-Fold
Cross Validation

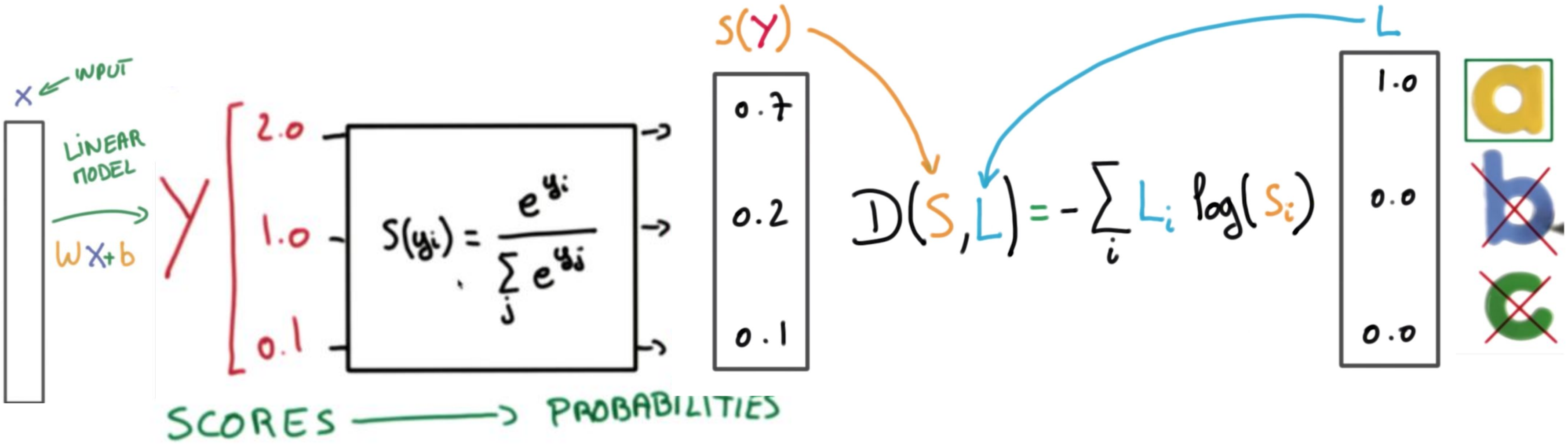
Label and Class Probability



Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

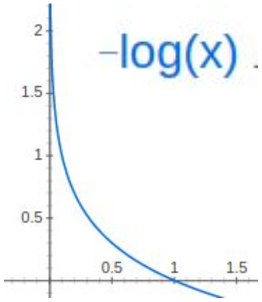


Softmax Cross-Entropy



Softmax

Cross-Entropy



Thanks

FinanceData.KR 2020