

Lecture 5~8

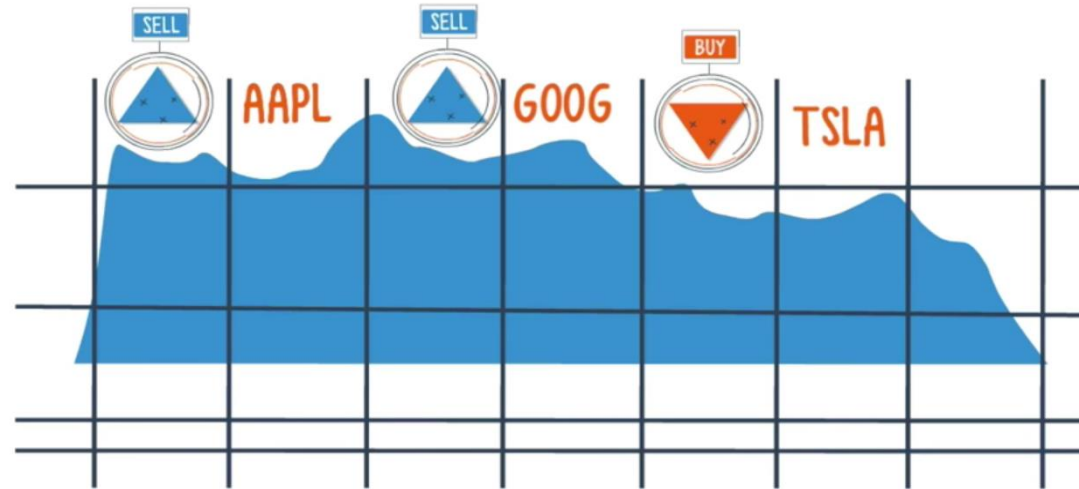
Lecture 5 – Logistic Regression

Classification

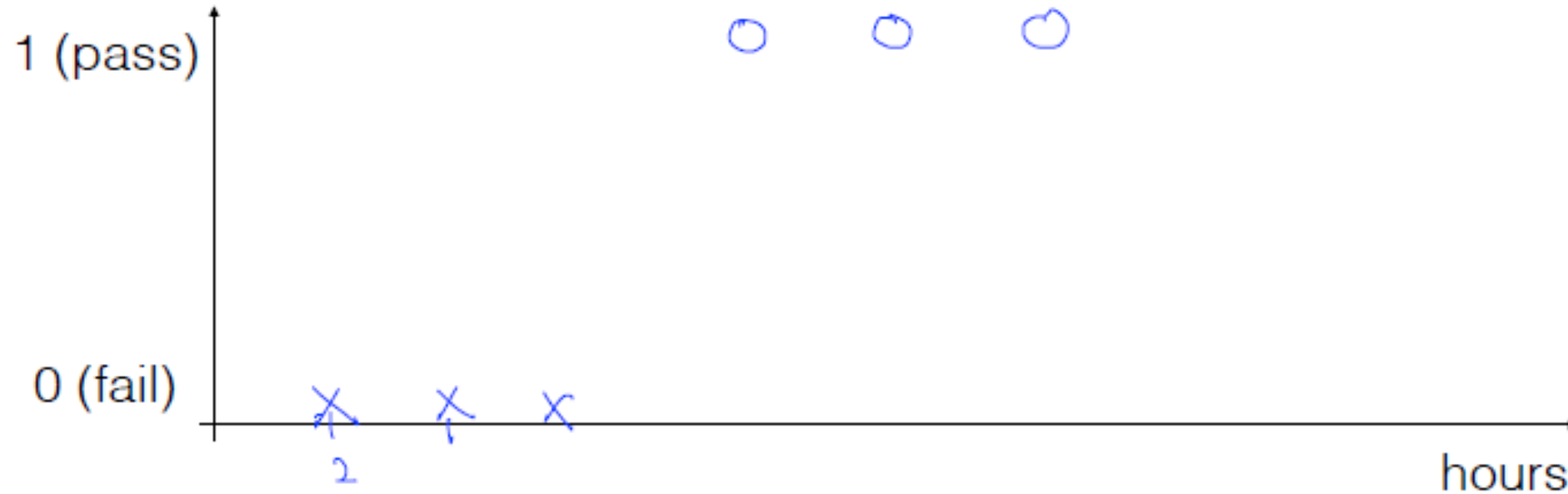
- Spam Detection: Spam(1) or Ham(0)
- Facebook feed: show(1) or hide(0)
- Credit Card Fraudulent Transaction detection: legitimate/fraud

Finance

DWJI	17,499.10	▼
SP500	2,025.51	▼
NASDAQ	4,976.9	▲
AAPL	107.71	▲
GOOG	750.06	▲
TSLA	234.24	▼



Pass(1)/Fail(0) based on study hours



In linear regression...

- We know Y is 0 or 1

$$\underline{H(x) = Wx + b}$$

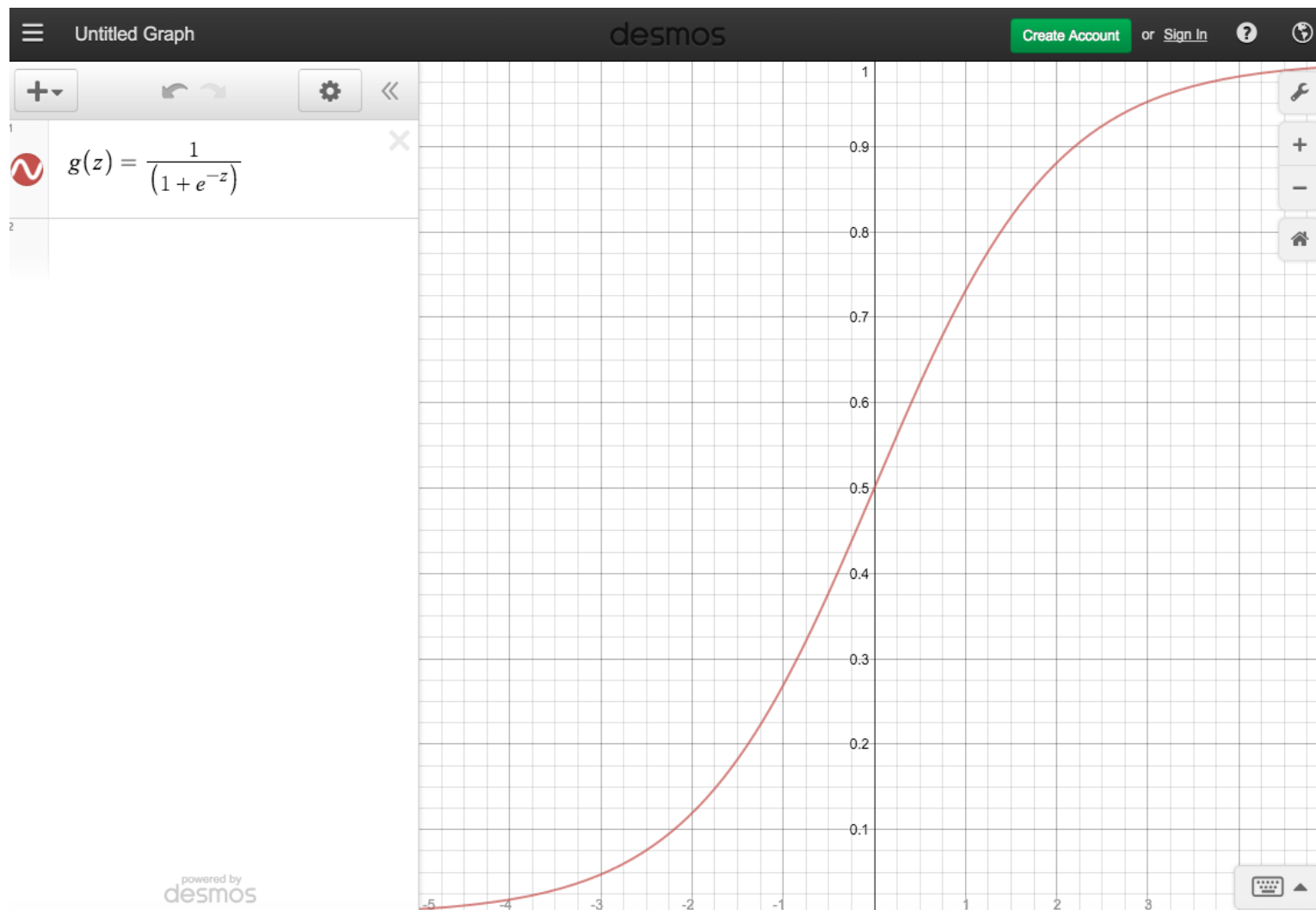
- Hypothesis can give values large than 1 or less than 0

$$x \begin{cases} 5 \\ 10 \\ 11 \end{cases}$$

$$w \approx 0.5, \quad b = 0$$

$$0 < \sim 4$$

—



Logistic Hypothesis

$$H(X) = \frac{1}{1 + e^{-(w^T X)}}$$

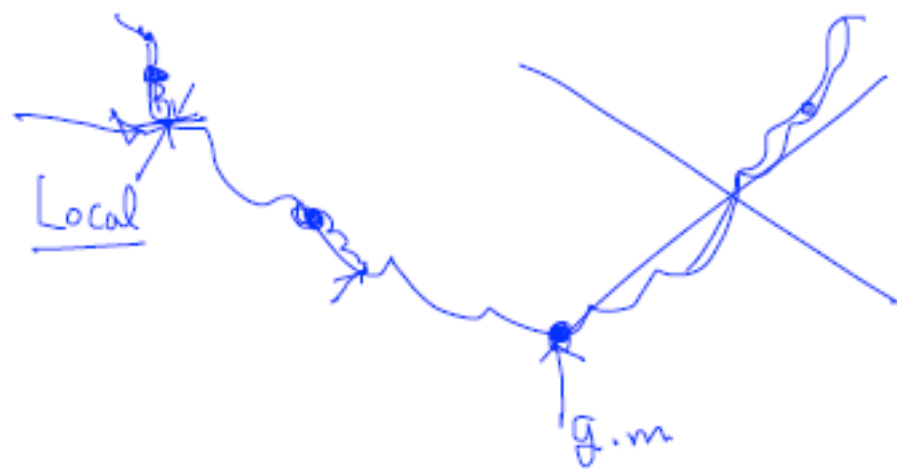
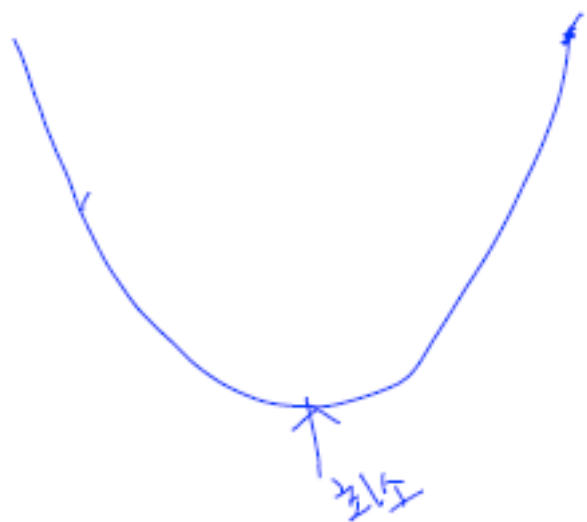
Cost function

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

$0 < \sim < 1$

$$\underline{H(x) = Wx + b} \quad //$$

$$\underline{H(X) = \frac{1}{1 + e^{-W^T X}}}$$



New cost function for logistic

$$\underline{cost(W)} = \frac{1}{m} \sum \quad \underline{\epsilon(H(x), y)}$$

$$\underline{c(H(x), y)} = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1 - H(x)) & : y = 0 \end{cases}$$

Cost function

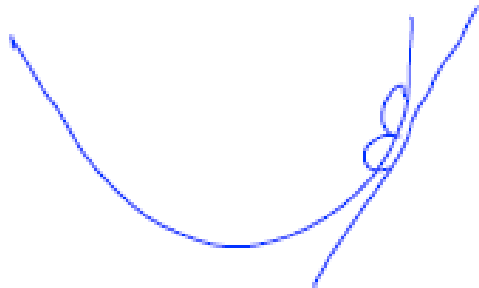
$$\text{cost}(W) = \frac{1}{m} \sum c(H(x), y)$$

$$C(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1 - H(x)) & : y = 0 \end{cases}$$

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

Minimize cost - Gradient decent algorithm

$$\underline{cost(W)} = -\frac{1}{m} \sum y \log(H(x)) + (1 - y) \log(1 - H(x))$$



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

Binary Classification

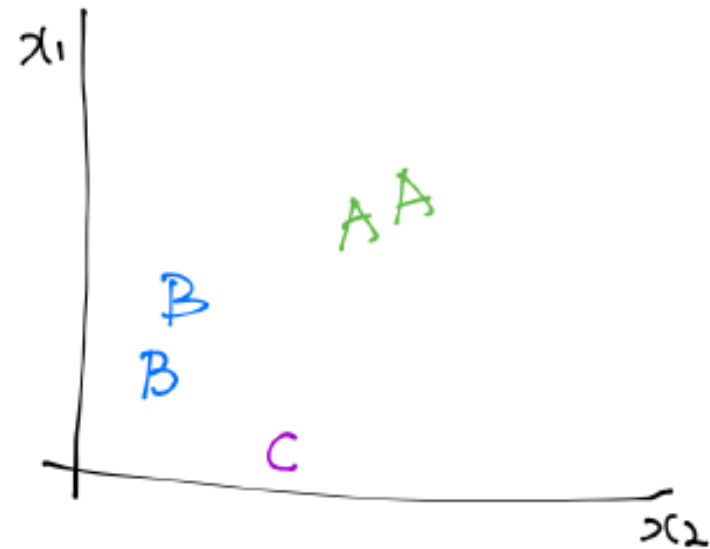
Label	Yes	No
Model 이 예측 결과 (\hat{y})	0.7	0.3
실제 값 (y)	1	0
	$-1 \cdot \log(0.7)$	$-0 \cdot \log(0.3)$

두 확률 분포의 차이
= cross entropy

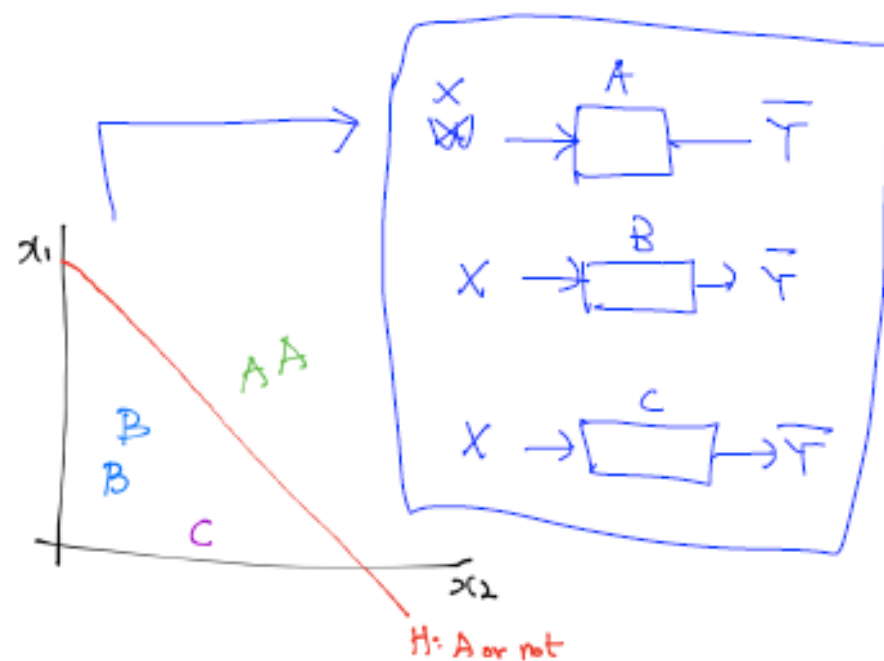
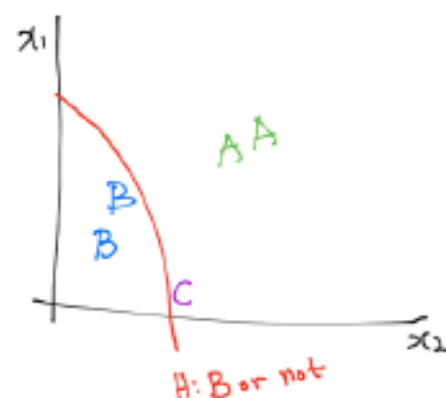
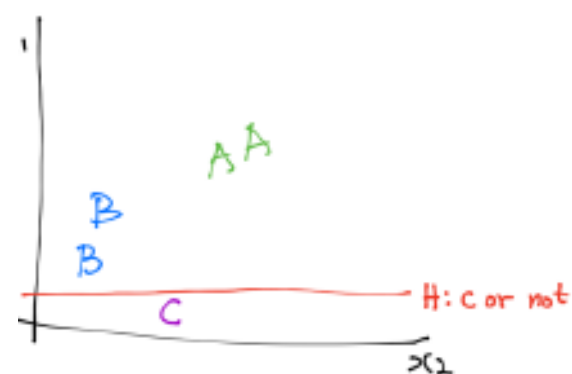
Lecture 6 – Softmax classification

Multinomial classification

x_1 (hours)	x_2 (attendance)	y (grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C



Multinomial classification

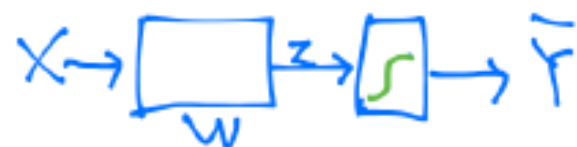
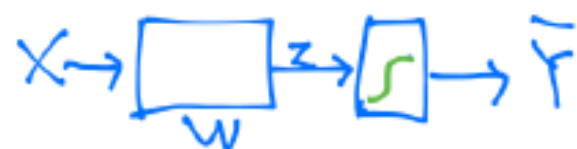
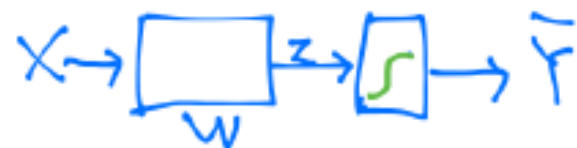


Multinomial classification

$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_2 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_2]$$

$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$



Multinomial classification

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$



$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} =$$



LOGISTIC CLASSIFIER

$$W X + b = Y$$



A

$\begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$

$\rightarrow p = 0.7$



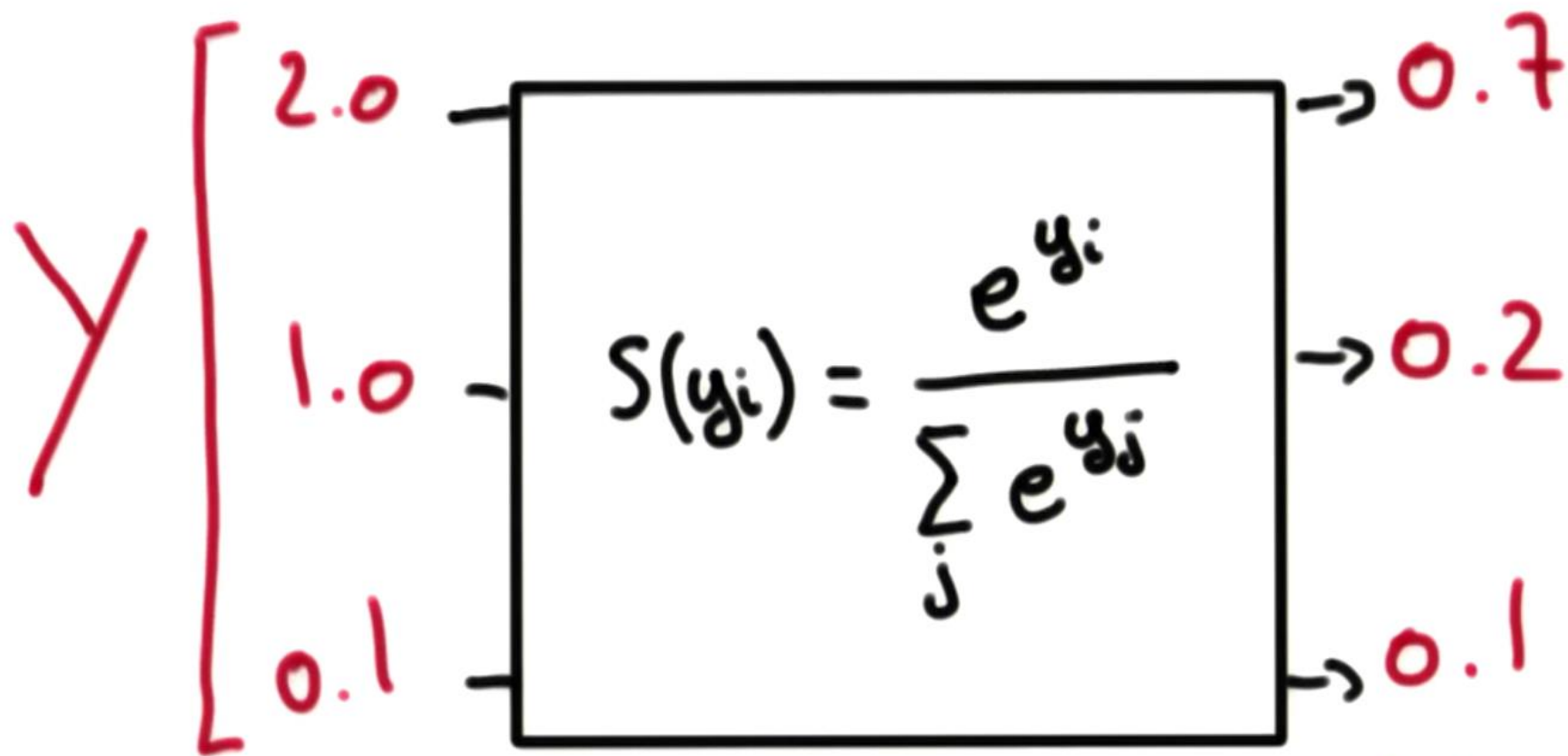
$\rightarrow p = 0.2$



$\rightarrow p = 0.1$



SOFTMAX



'LOGITS'



SCORES



PROBABILITIES

CROSS-ENTROPY

$S(Y)$

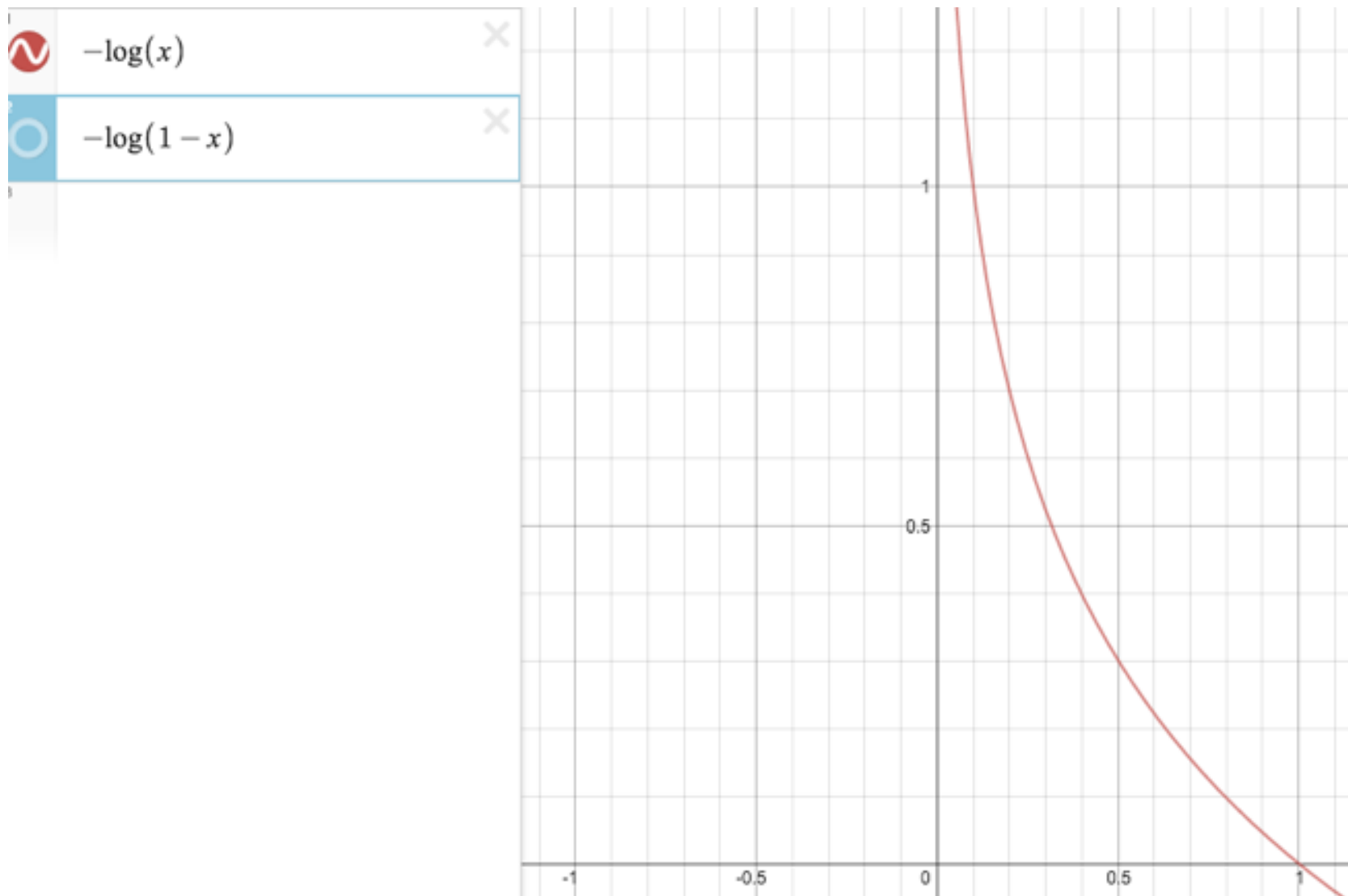
0.7
0.2
0.1

L

1.0
0.0
0.0

$$D(S, L) = - \sum_i L_i \log(S_i)$$

그래서 cross entropy가 MSE보다 좋은게 뭔데?



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

$$\mathcal{D}(S, L) = - \sum_i L_i \log(S_i)$$

// ?

LOSS

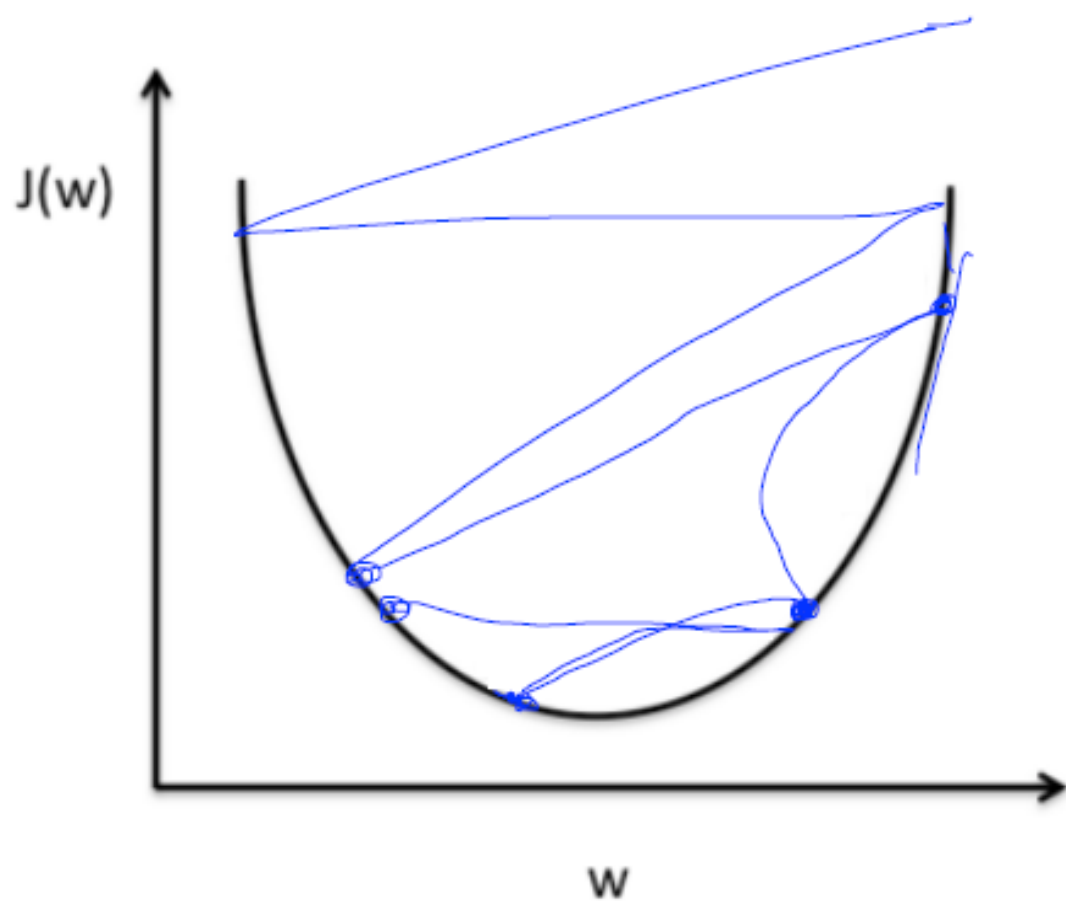
$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(S(w x_i + b), L_i)$$

TRAINING SET

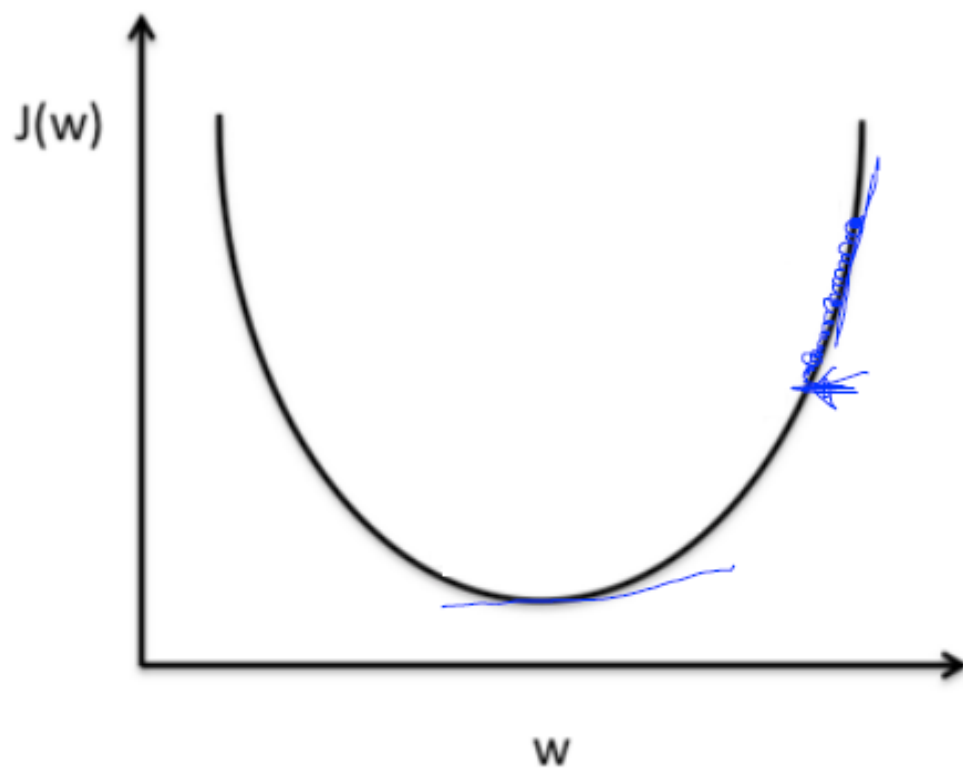
Classification 정리

- MLP의 결과를 Probability distribution 으로 만들기 위해
- Softmax 사용
- 정답 Label의 probability distribution와 softmax 결과의 probability distribution가 얼마나 다른지?
- Cross Entropy Loss 사용
- Binary classification이든 multinomial classification이든 똑같음.

Large learning rate: overshooting



Small learning rate:
takes too long, stops at local minimum



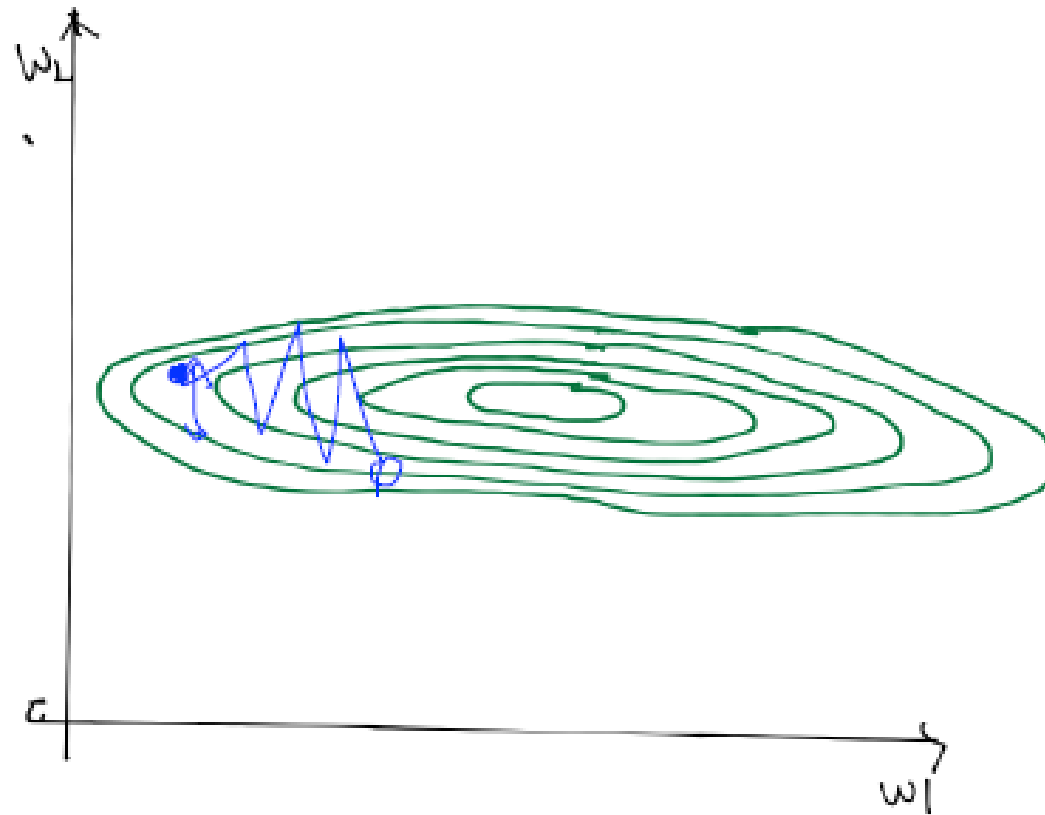
Try several learning rates

0.01

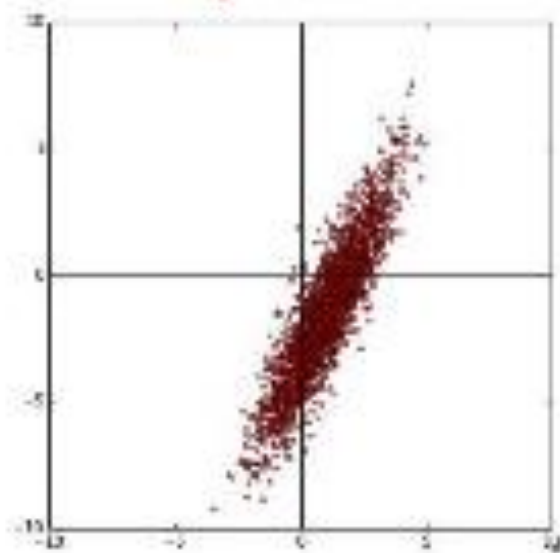
- Observe the cost function
- Check it goes down in a reasonable rate

Data (X) preprocessing for gradient descent

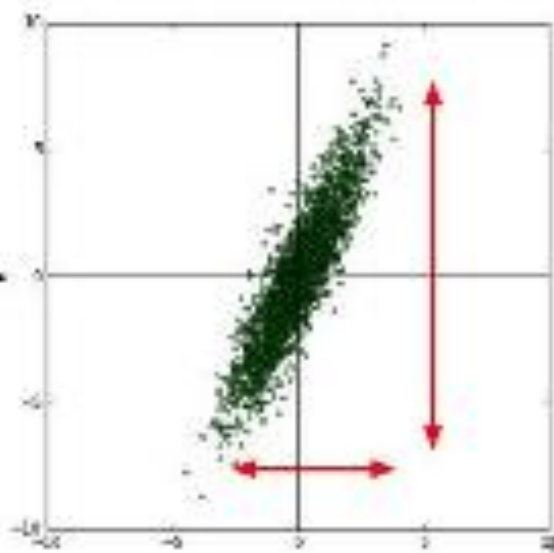
x1	x2	y
1	9000	A
2	-5000	A
4	-2000	B
6	8000	B
9	9000	C



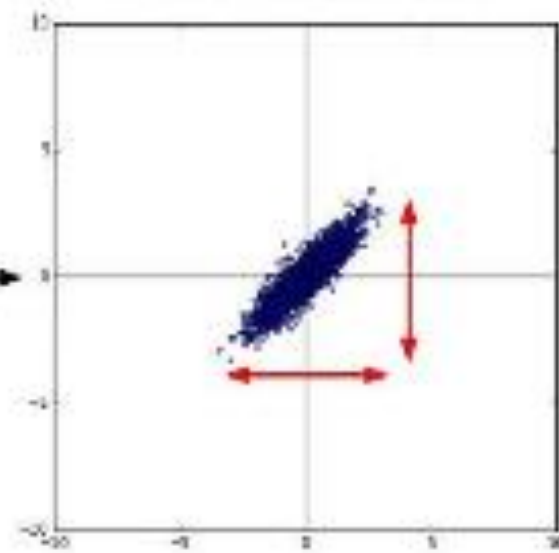
original data



zero-centered data



normalized data



Standardization

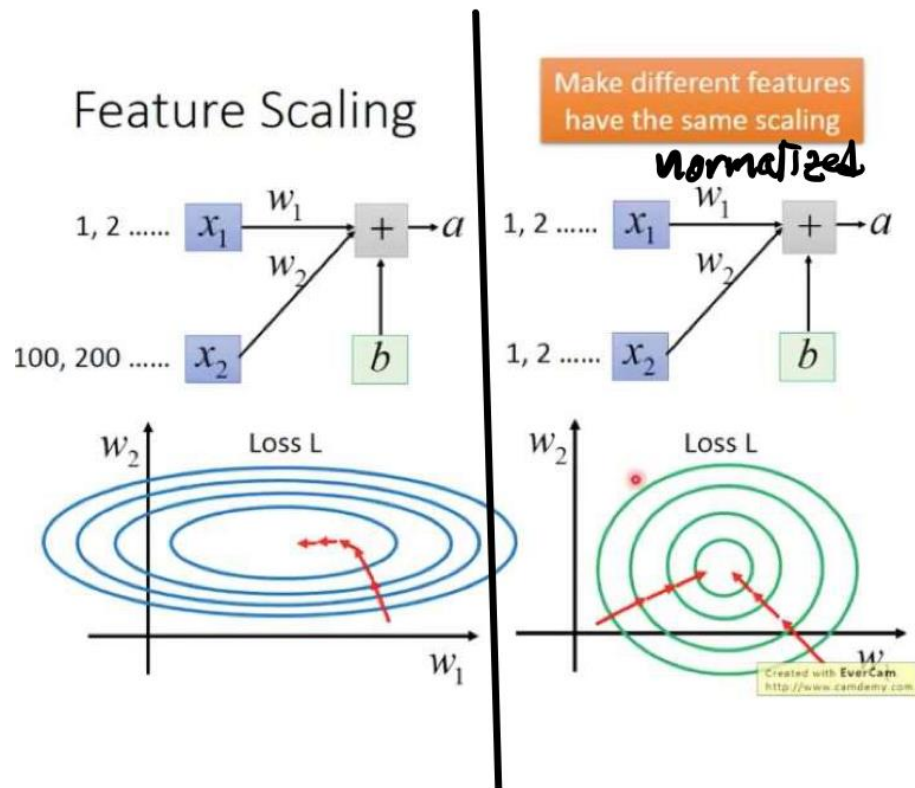
$$\boxed{x'_j = \frac{x_j - \mu_j}{\sigma_j}} \star$$

```
X_std[:,0] = (X[:,0] - X[:,0].mean()) / X[:,0].std()
```

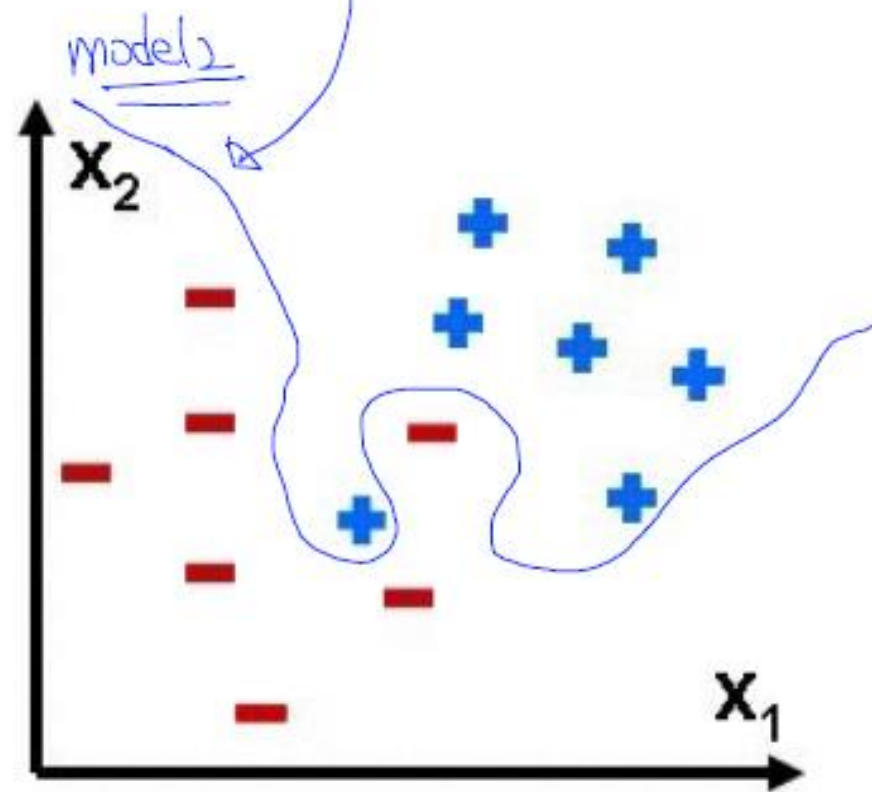
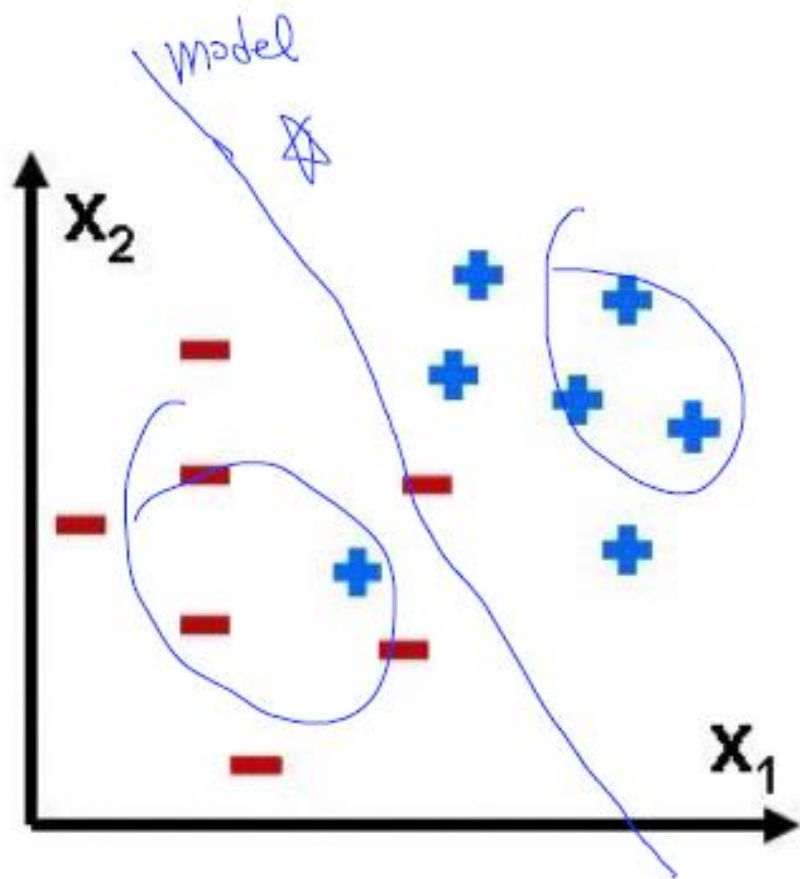
http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Data Preprocessing: 왜함?

- 모델이 더 빠르게 학습할 수 있다.
- 모델이 학습하기 더 쉬워진다.



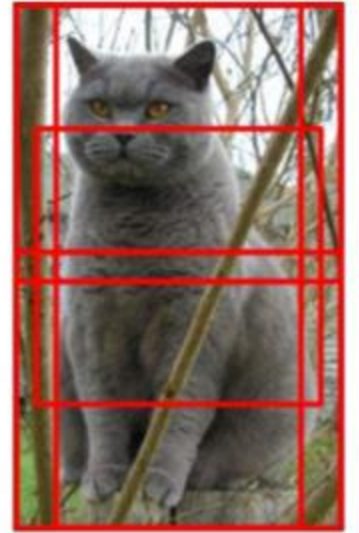
Overfitting



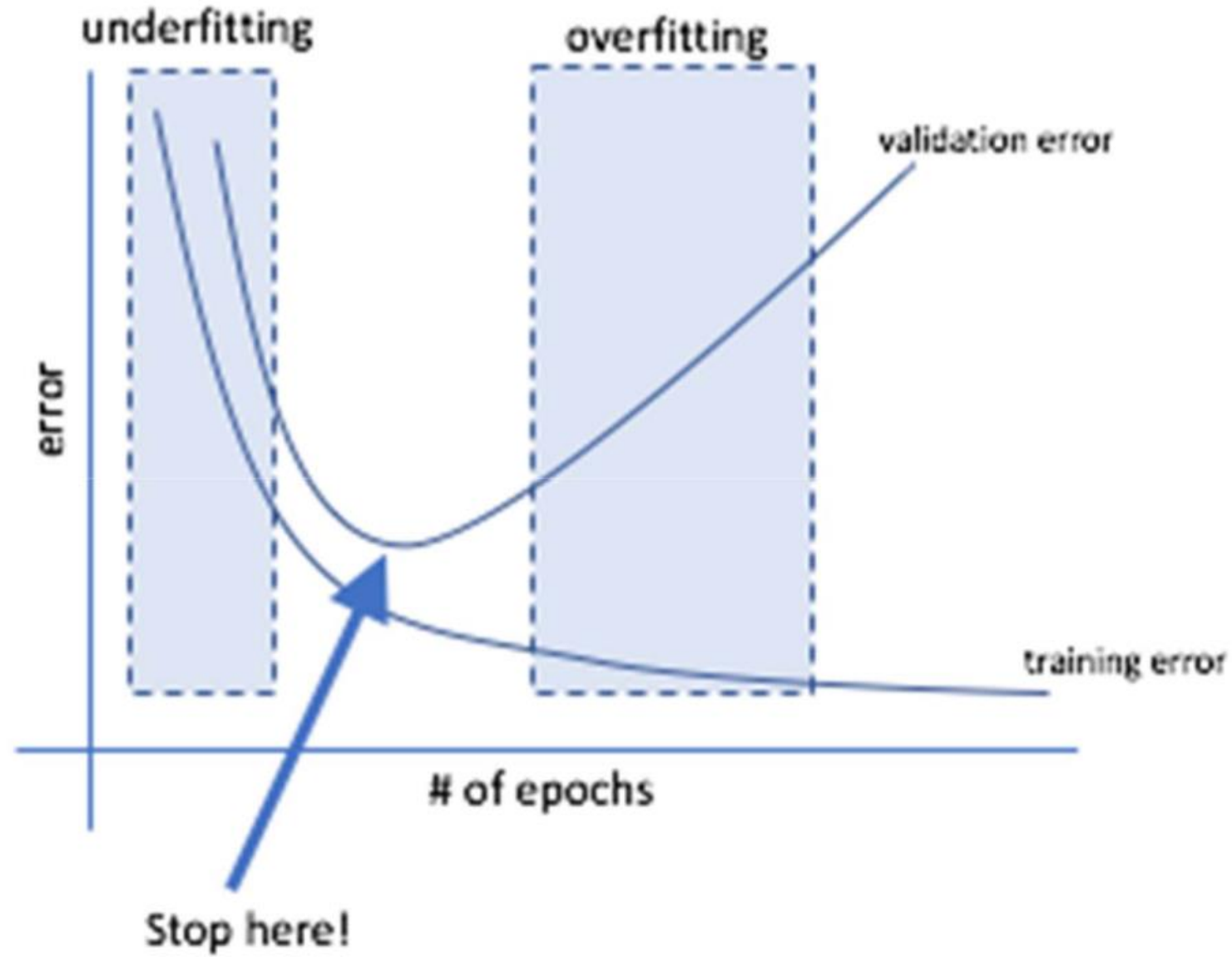
Solutions for overfitting

- More training data!
- Reduce the number of features
- Regularization - L1, L2, Batch norm, Dropout
 - + Data Augmentation
 - + Early Stopping
 - + Ensembles

Data Augmentation?



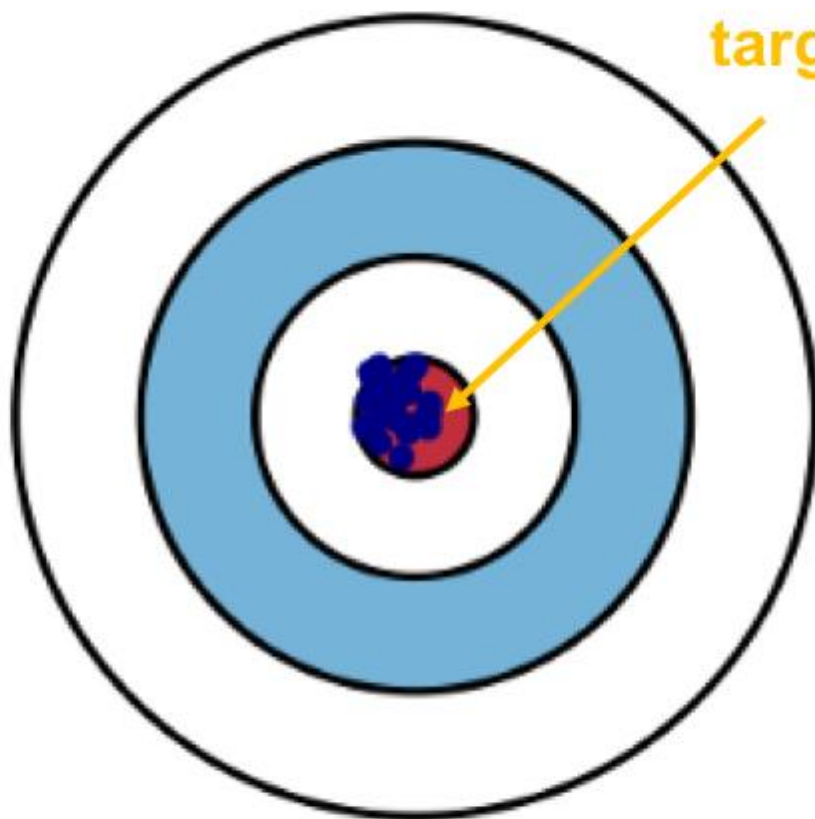
Early Stopping?



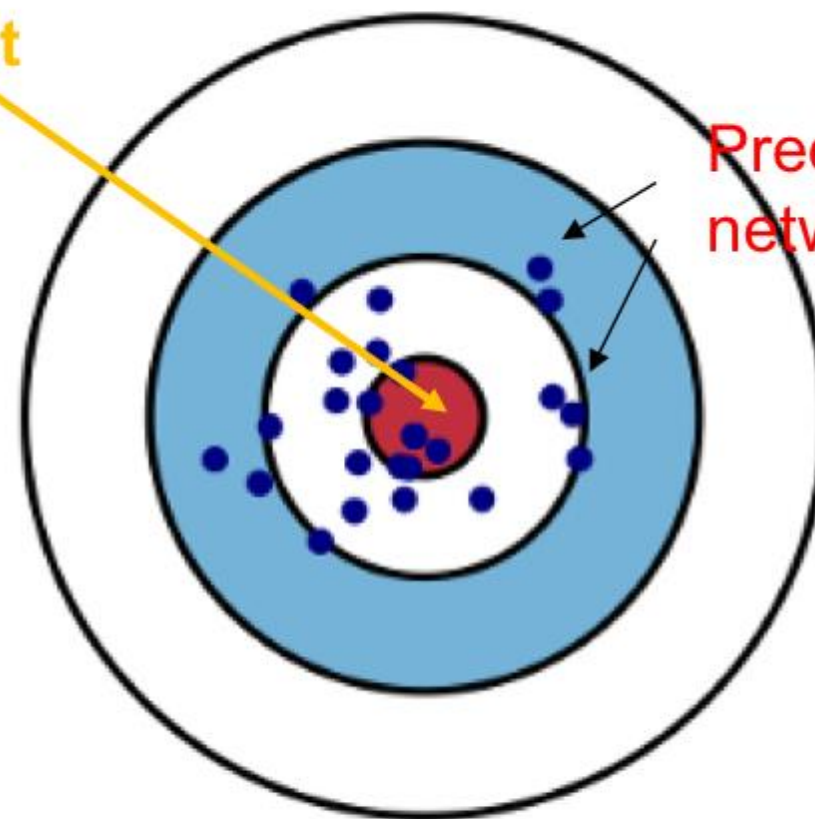
Low Bias

Low Variance

High Variance



target



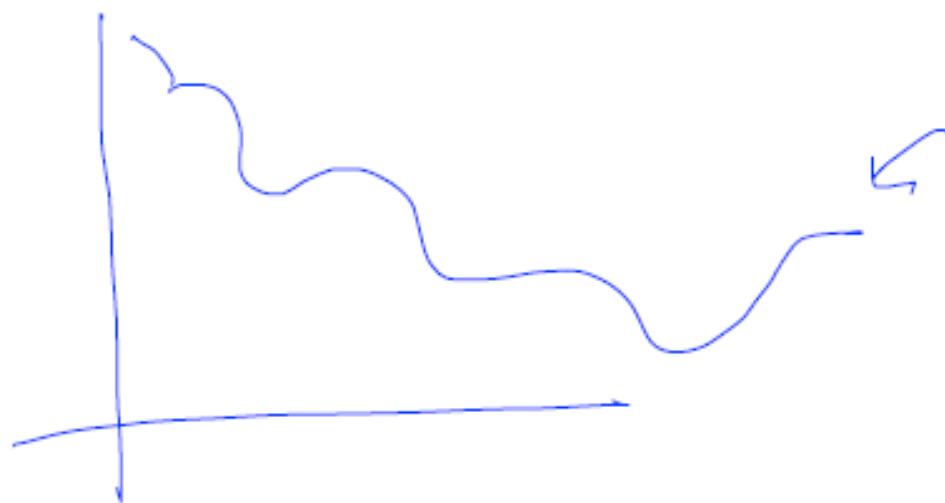
Predictions from N
networks

Well Generalized

Overfitting

Regularization

- Let's not have too big numbers in the weight



Weight가 크면 왜 안좋은데?

Regularization

- Let's not have too big numbers in the weight

Diagram illustrating the components of the loss function and regularization:

LOSS (indicated by an arrow pointing to the loss function)

l2reg = `0.001 * tf.reduce_sum(tf.square(W))` (indicated by an arrow pointing to the regularization term in the code)

TRAINING SET (indicated by an arrow pointing to the index i in the loss function)

The loss function is defined as:

$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i) + \lambda \sum W^2$$

Annotations:

- \mathcal{L} is the total loss.
- $\frac{1}{N}$ is the inverse of the number of samples.
- \sum_i is the summation over the training set.
- $\mathcal{D}(s(w x_i + b), L_i)$ is the divergence (loss) between the predicted output $s(w x_i + b)$ and the target output L_i .
- λ is the regularization strength.
- $\sum W^2$ is the sum of the squared weights, representing the regularization term.

Evaluation using training set?

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

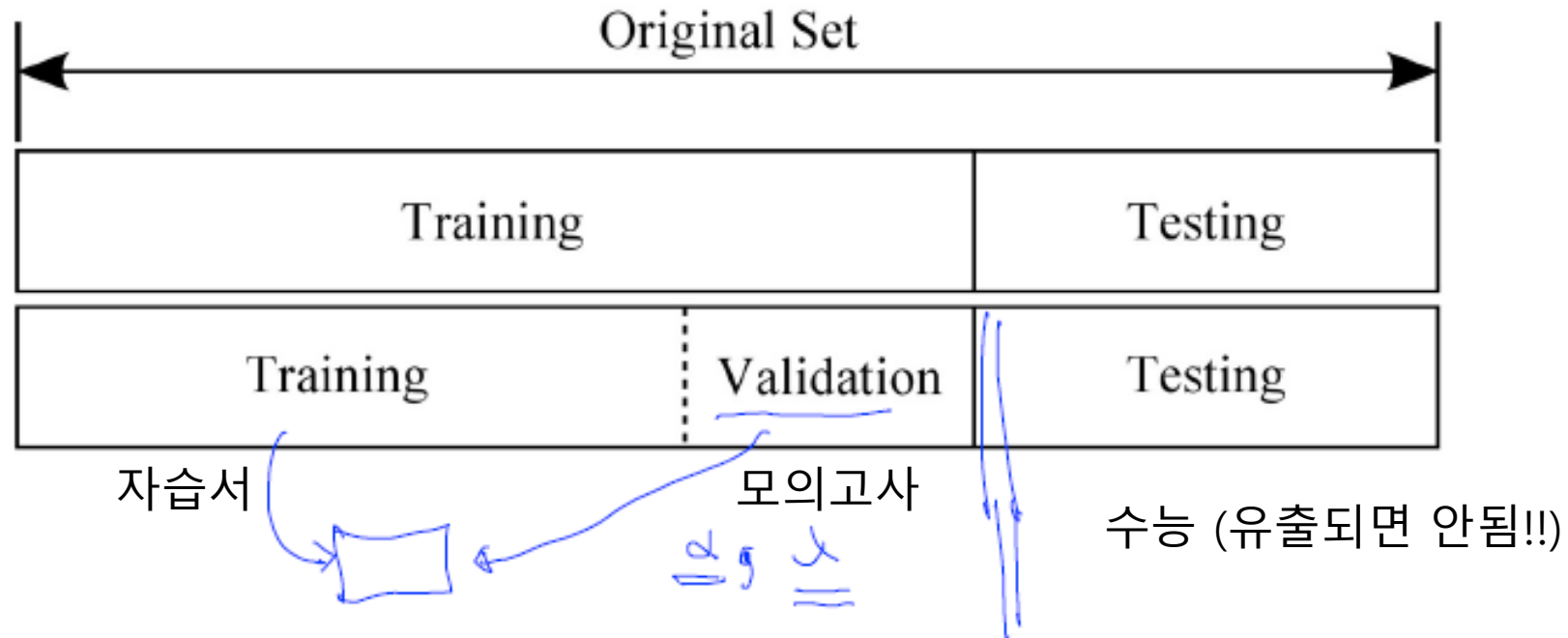
training set



- 100% correct (accuracy)
- Can memorize

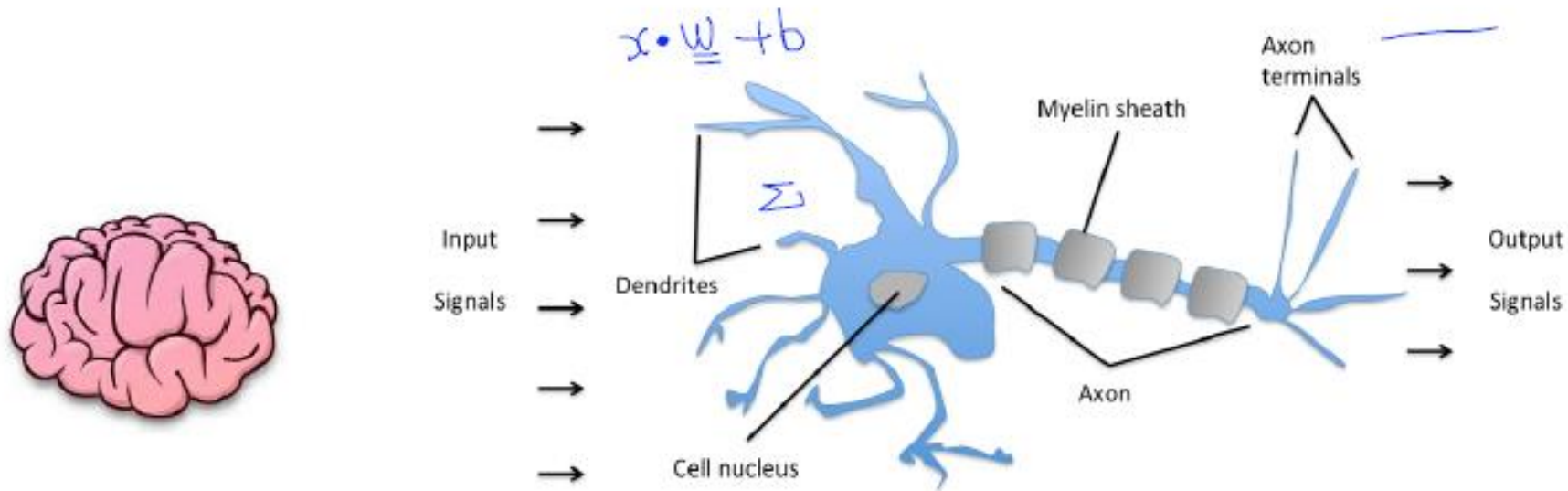
족보 그대로 시험문제가 나옴!! 제대로 평가할 수 없음.

Training, validation and test sets



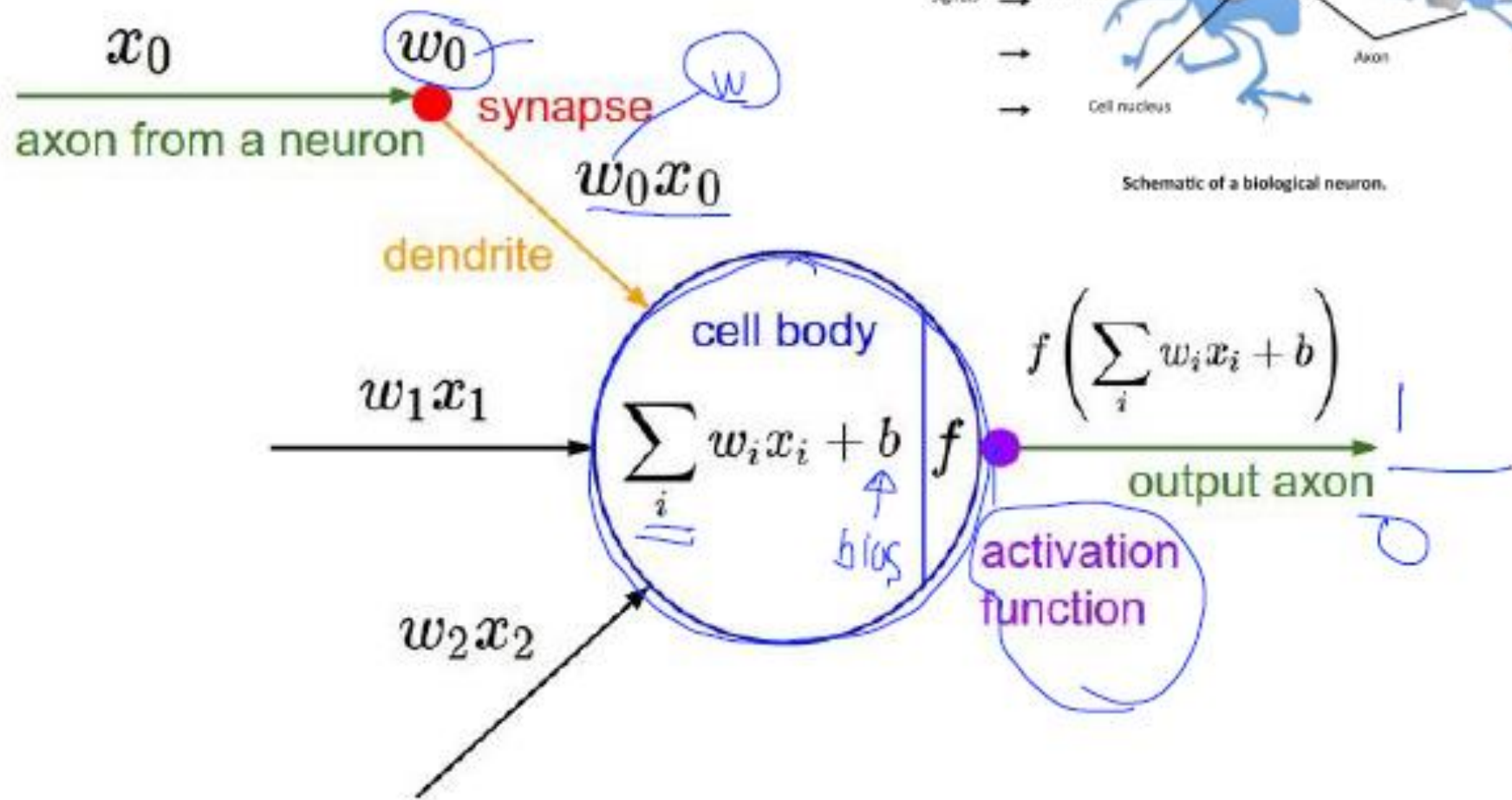
Lecture 8

Ultimate dream: thinking machine



Schematic of a biological neuron.

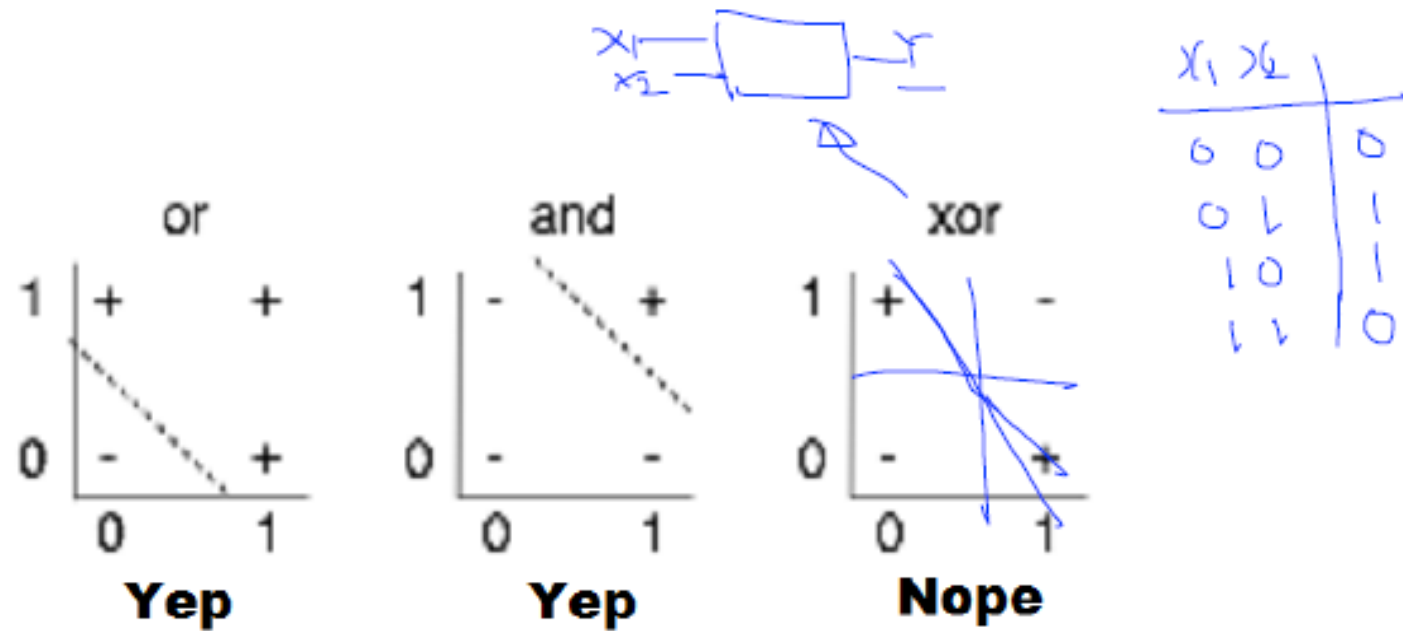
Activation Functions



Perceptron도 문제가 있다?

XOR Problem

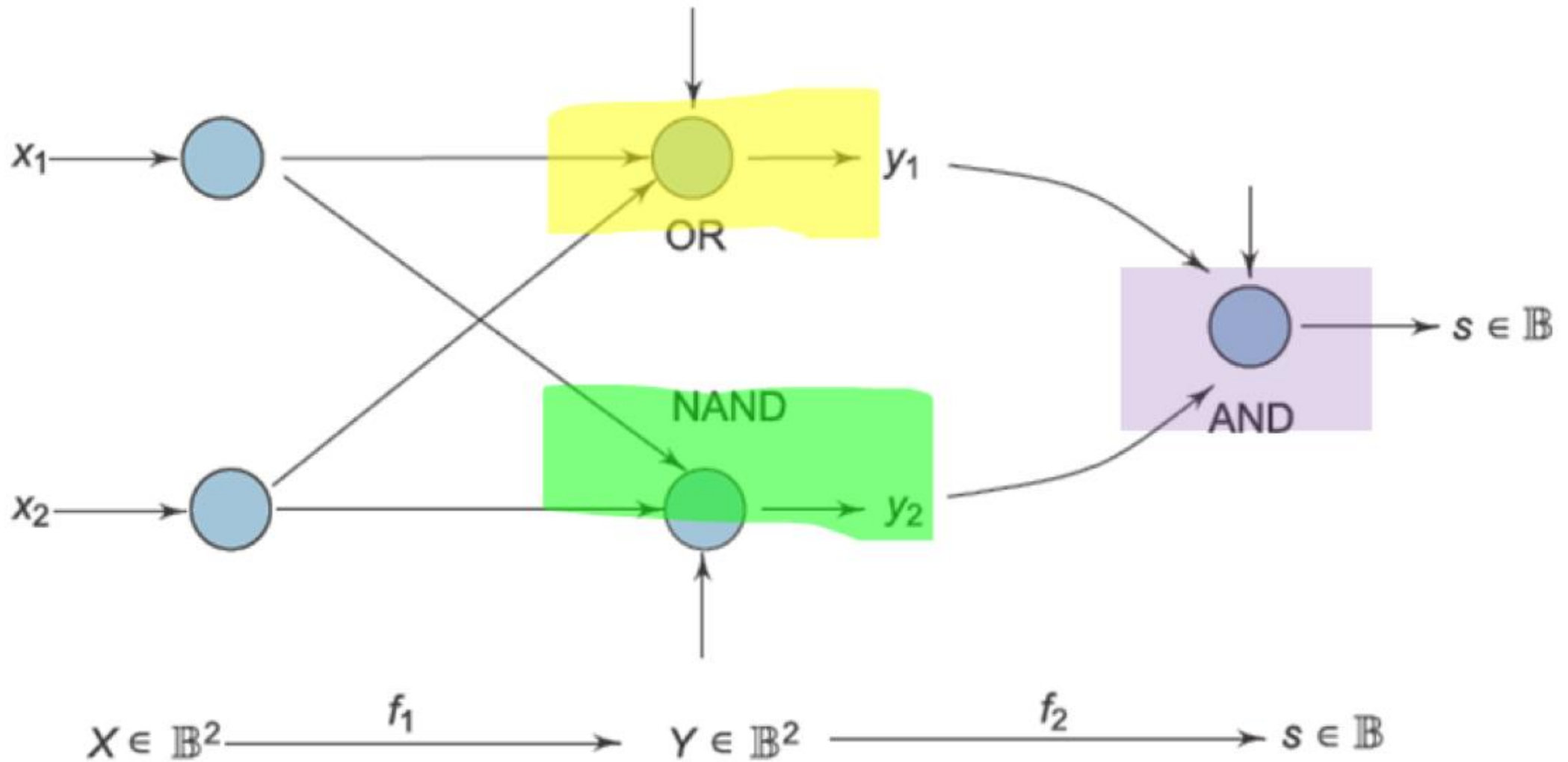
(Simple) XOR problem: linearly separable?



어떻게 해결하지?

- Perceptron은 Linearly separable 한 문제만 풀 수 있다.
 - 그럼 문제를 linearly separable하게 만들면 되지 않을까?
-
- 1. instance 의 숫자를 줄인다.
 - 2. feature의 차원을 높인다.

Instance의 개수를 줄인다?



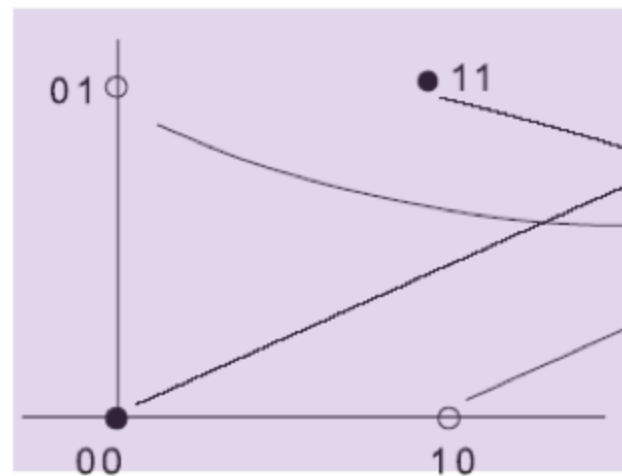
$$y_1 = x_1 + x_2; y_2 = \overline{x_1 x_2}$$

x_1	x_2		y_1	y_2	s
0	0	→	0	1	0
0	1	→	1	1	1
1	0	→	1	1	1
1	1	→	1	0	0

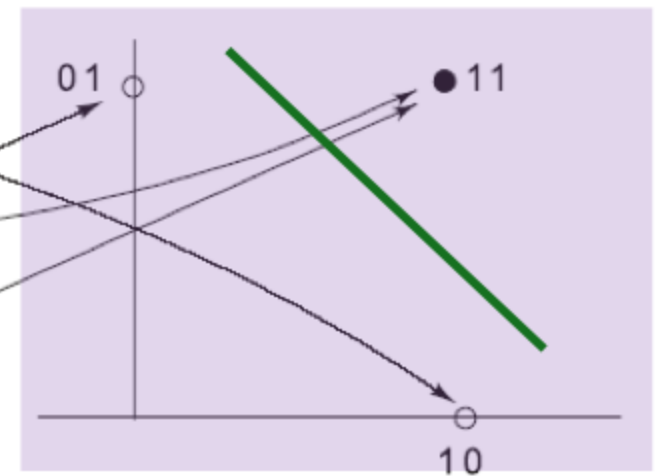
Original
input

New
input

Output

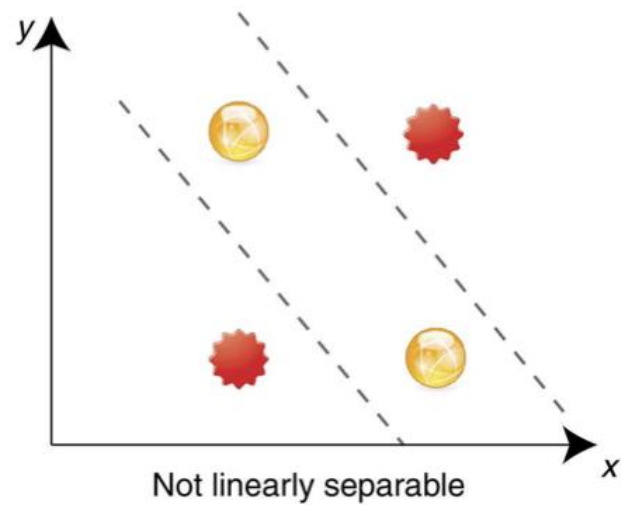


Linearly non-separable

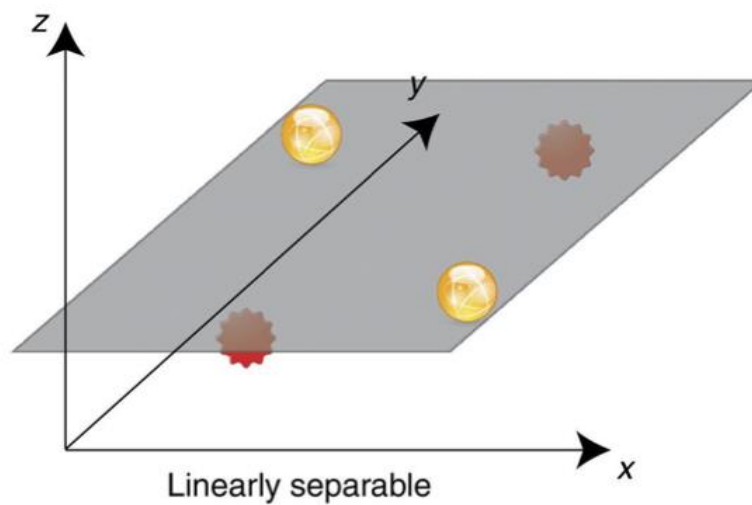


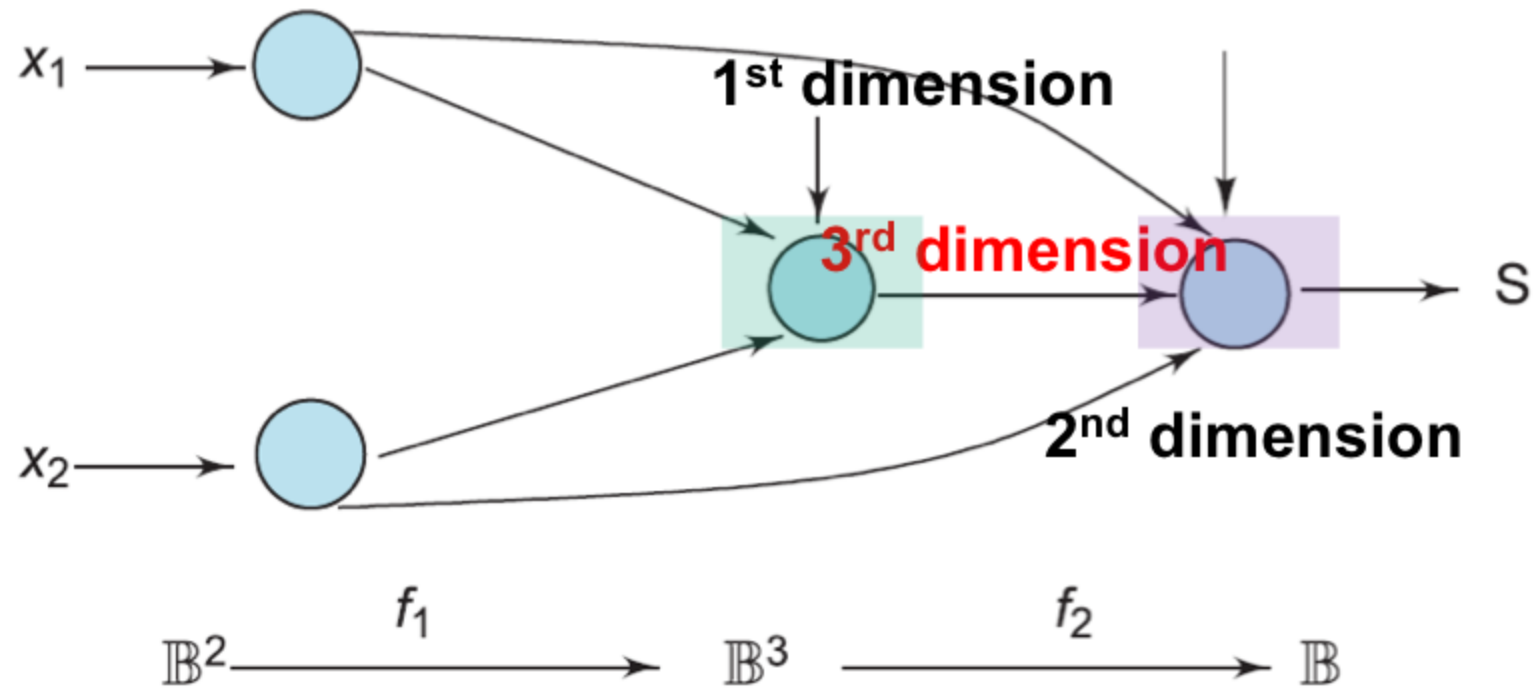
Linearly separable

Feature 의 차원을 높인다?

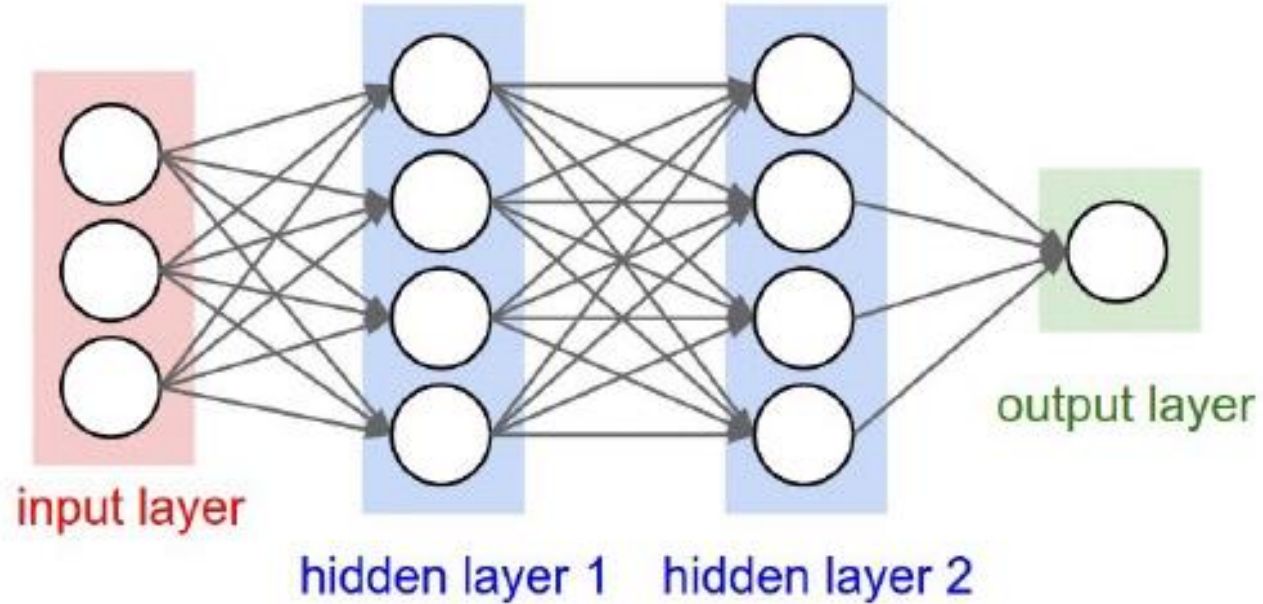


depth






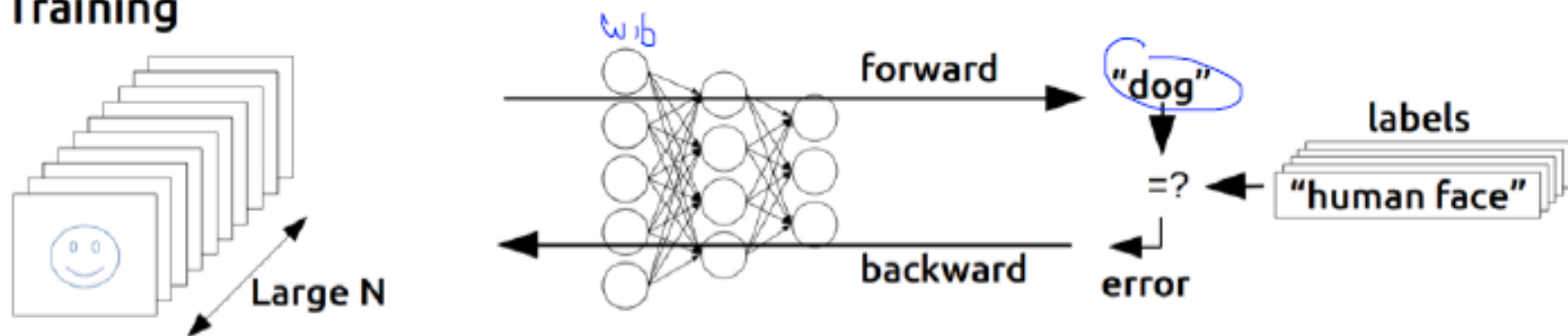
“No one on earth had found a viable way to train*”



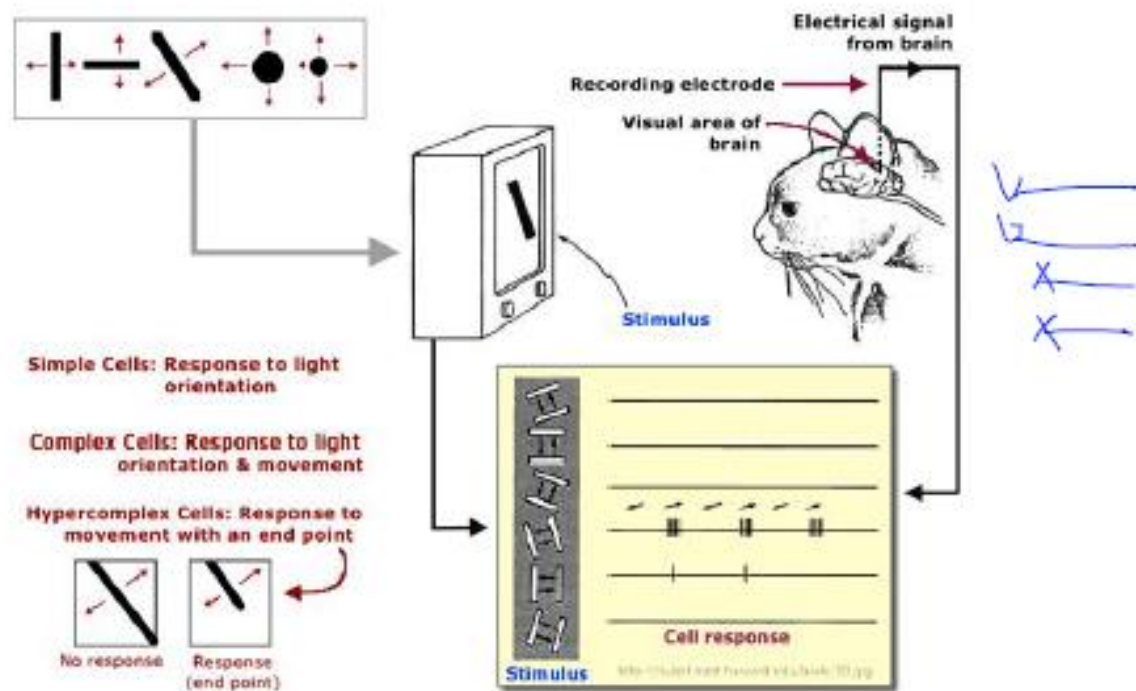
Backpropagation

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

Training



Convolutional Neural Networks

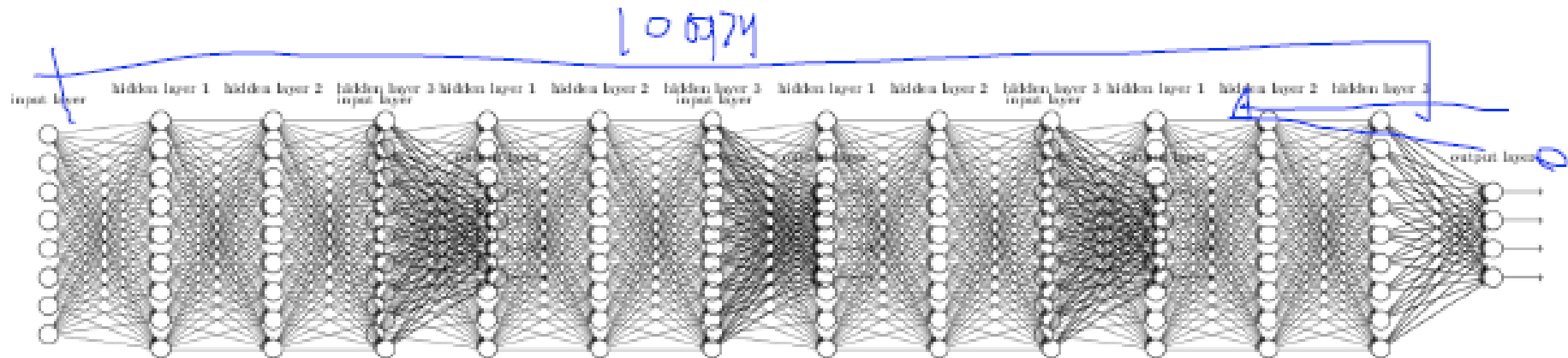


Hubel & Wiesel, 1959

A BIG problem

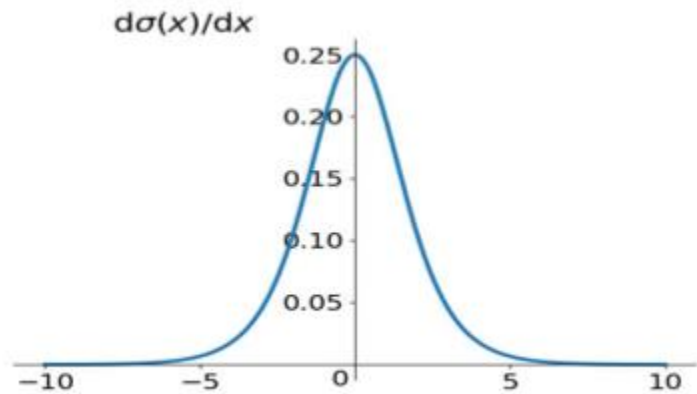


- Backpropagation just did not work well for normal neural nets with many layers
- Other rising machine learning algorithms: SVM, RandomForest, etc.
- 1995 “Comparison of Learning Algorithms For Handwritten Digit Recognition” by LeCun et al. found that this new approach worked better



Gradient Vanishing이 일어나는 이유?

- 비 효율적인 Activation function을 썼기 때문.
- 당시 널리 쓰이던 sigmoid로 인한 문제.



Breakthrough

in 2006 and 2007 by Hinton and Bengio

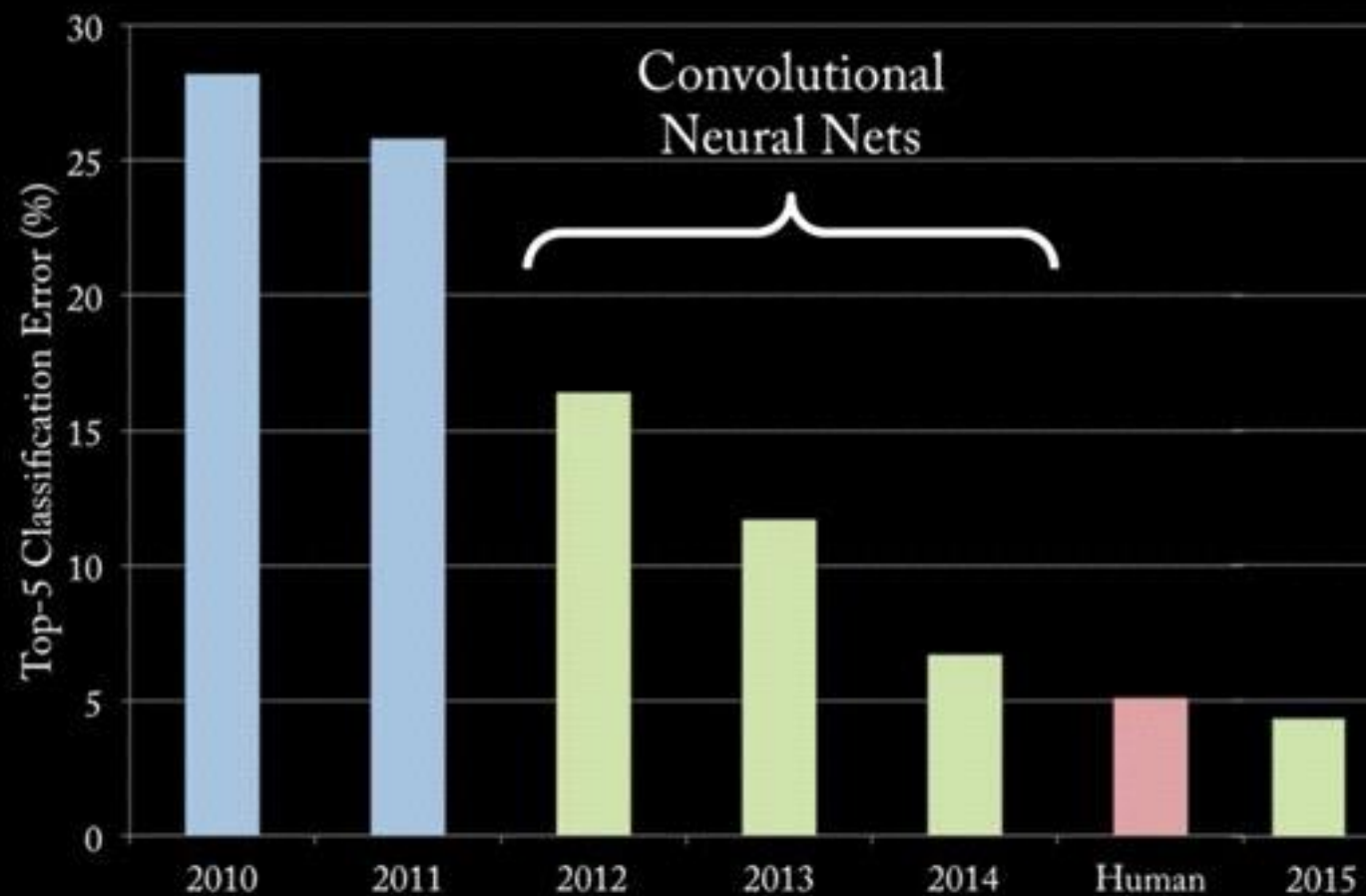


- Neural networks with many layers really could be trained well, if the weights are initialized in a clever way rather than randomly.
- Deep machine learning methods are more efficient for difficult problems than shallow methods.
- Rebranding to Deep Nets, Deep Learning

Initialization이 영향을 많이 미치나?

- 작게 초기화 했을 경우 => 학습이 너무 느림
- 크게 초기화 했을 경우 => bad local minima에 갇힘
- Xavier, He initializer으로 개선.

ImageNet Classification (2010 – 2015)



끝