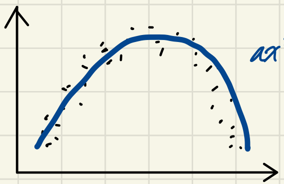
* what is "linear" in Linear Regression?



$$\overbrace{\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix}}^{A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{x} \simeq \overbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix}}^{y}$$

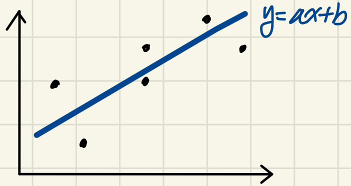
$$A \cdot \underline{x} = \underline{y}$$

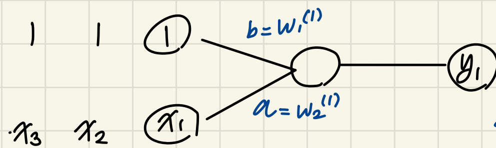
(parameter)

어떤 함수가 y를 만드는 parameter (a, b, c) 에 대해서 linear한가 *.

* Non Linear?

> Neural Network.





$$\text{Cost} = \Sigma \left(y_i - (ax_i+b)\right)^2$$

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \simeq \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix}$$

Linear!

- $y = ax^2+bx+C \rightarrow \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ & \vdots & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \simeq \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix}$
- $y = \dfrac{e^{ax^2+bx+c}}{1+e^{ax^2+bx+c}} \rightarrow \dfrac{y}{1-y} = \dfrac{e^{\sim}}{1+e^{\sim}} \Big/ \dfrac{1}{1+e^{\sim}}$

$$\ln\left(\frac{y}{1-y}\right) = ax^2+bx+c$$

Non Linear!

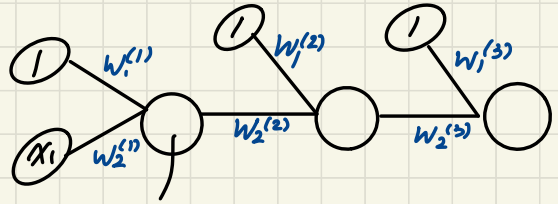
- $y = \dfrac{ax}{b+x}$
-

> Non Linear Reg with NN.

> Deep Neural Network.

↑

Activation function.

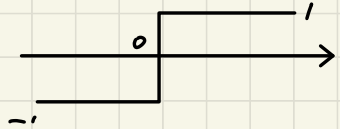


$$(W_1^{(1)} + W_2^{(1)} \cdot x_1) \cdot W_2^{(2)} + W_1^{(2)} + \cdots = y$$

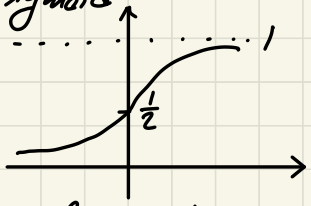
$$= (W_1^{(1)} W_2^{(2)}) \cdot x_1 + (W_1^{(1)} \cdot W_2^{(2)} + W_1^{(2)})$$

Actf → $y = x$. : Linear!

- perceptron.



- sigmoid



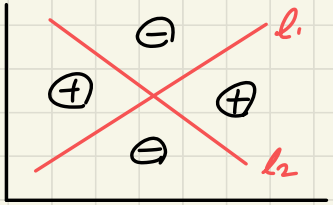
Logistic Reg

- Softmax.

$$\frac{e^{x_i}}{\sum_i e^{x_i}}$$

* Activation function.

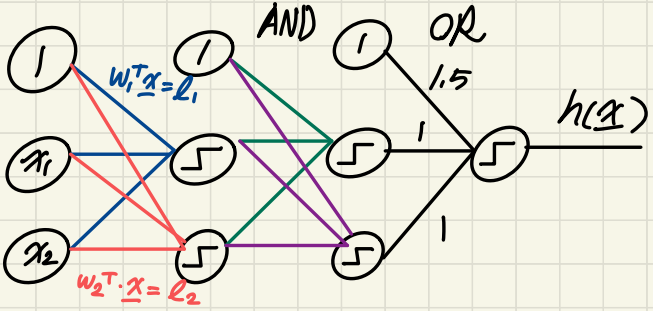
> Neural Net?

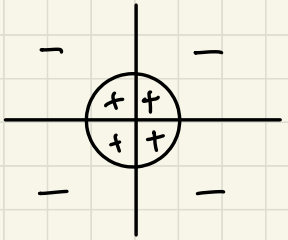


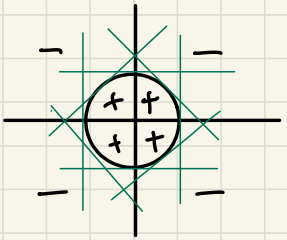
⇒ l_1, l_2 특징의 One으로 OR / AND 조건을 생성해 ⊕, ⊖ 구분

ex) $(l_1 \geq 0 \;\&\; l_2 \geq 0) \mid (l_1 \leq 0 \;\&\; l_2 \leq 0)$

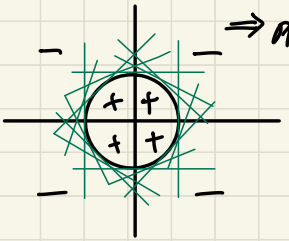
$= \ominus$.







8 perceptron

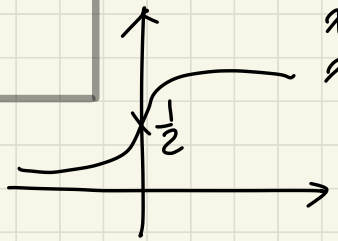


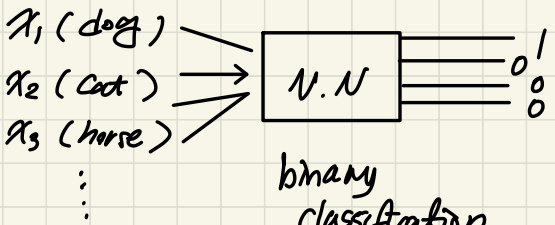
⇒ 여러개의 선 = non linear 한 영역으로 분류가능.

16 perceptron.

* Activation function.
> Sigmoid

$$y = \frac{e^x}{1+e^x}$$





binary classification.

[강아지일 확률
(1 − 강아지일확률) = 강아지 X 확률

$$p(Y_i \mid \underline{x}_i)$$

$$= \begin{bmatrix} sig(f(\underline{x}_i, \underline{w})) \\ 1 - sig(\,) \end{bmatrix}$$

multi class

One-hot encoding

• loss = likelihood.

$$= -\left(\prod_{1}^{n} p(Y_i \mid \underline{x}_i)\right)$$

Θ(Maximize) → minimize

$$= -\sum \log p(Y_i \mid \underline{x}_i)$$

* Multi Class classification.

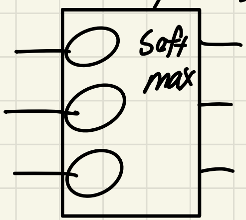
> one-hot-encoding

$\begin{bmatrix}1\\0\\0\end{bmatrix}$ $\begin{bmatrix}0\\1\\0\end{bmatrix}$ $\begin{bmatrix}0\\0\\1\end{bmatrix}$

dog cat horse.

> softmax regression.

output layer.



$$\frac{e^{x_i}}{\sum e^{x_i}}$$

> loss = Cross-Entropy.

$$-\sum p_i \log q_i$$

* Regression vs Classification.

2진 값이 fit하는건 마찬가지.

* Entropy.

- 정보를 표현하는 최소자원량 (bits!). $-\sum p_i \log_2 p_i$

> Cross-entropy.

$$-\sum p_i \log_2 q_i$$

> KL-Divergence

$= CE - E$

> mutual-information. ?

* Scalar를 Vector로 미분.

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = f(x_1, x_2) = (\underline{y} - A\underline{x})^T (\underline{y} - A\underline{x})$$

$$\frac{\partial f}{\partial \underline{x}^T} \triangleq \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right]$$

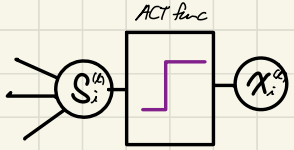
$$\Rightarrow \quad df = f(\underline{x} + d\underline{x}) - f(\underline{x}), \quad d\underline{x} = \begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix}$$

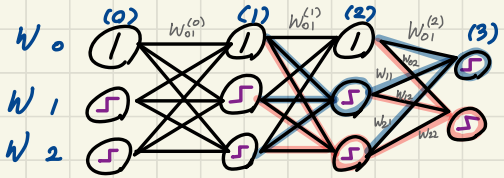
* Deep Neural Net
  > Partial Derivatives.
  > Back Propagation.

$(2x^2+3)^2$ (y)

$\Rightarrow \frac{\partial f}{\partial (2x^2+3)} \cdot \frac{\partial (2x^2+3)}{\partial x}$ — chain rule

$2(2x^2+3) \cdot 4x$ (y)





Act f 통과한 값.

$x_1^{(3)} = g(s_1^{(2)})$,

$x_2^{(3)} = g(s_2^{(2)})$,

2번째 Layer의 weighted sum.

Act f 통과하기 전의 값

$s_1^{(2)} = W_{01}^{(2)} + W_{11}^{(2)} \cdot x_1^{(2)} + W_{21}^{(2)} \cdot x_2^{(2)}$

$s_2^{(2)} = W_{02}^{(2)} + W_{12}^{(2)} \cdot x_1^{(2)} + W_{22}^{(2)} \cdot x_2^{(2)}$

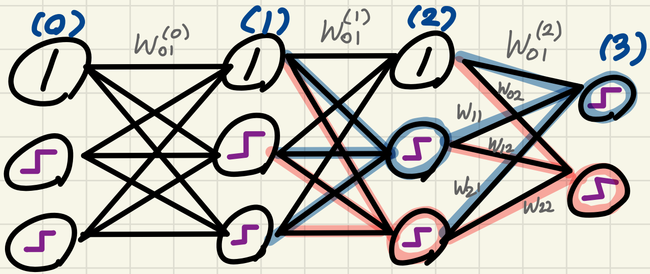
$\mathcal{E} = f = (y_1 - x_1^{(3)})^2 + (y_2 - x_2^{(3)})^2$

① $\frac{\partial \mathcal{E}}{\partial W_{21}^{(2)}} = \frac{\partial f}{\partial W_{21}^{(2)}} = \frac{\partial f}{\partial x_1^{(3)}} \cdot \frac{\partial x_1^{(3)}}{\partial s_1^{(2)}} \cdot \frac{\partial s_1^{(2)}}{\partial W_{21}^{(2)}} = x_2^{(2)}$

② $\frac{\partial f}{\partial W_{12}^{(1)}} = \frac{\partial f}{\partial x_1^{(3)}} \cdot \frac{\partial x_1^{(3)}}{\partial s_1^{(2)}} \cdot \frac{\partial s_1^{(2)}}{\partial x_2^{(2)}} \cdot \frac{\partial x_2^{(2)}}{\partial s_2^{(1)}} \cdot \frac{\partial s_2^{(1)}}{\partial W_{12}^{(1)}}$

⊕ $\frac{\partial f}{\partial x_2^{(3)}} \cdot \frac{\partial x_2^{(3)}}{\partial s_2^{(2)}} \cdot \frac{\partial s_2^{(2)}}{\partial x_2^{(2)}} \cdot \frac{\partial x_2^{(2)}}{\partial s_2^{(1)}} \cdot \frac{\partial s_2^{(1)}}{\partial W_{12}^{(1)}} = x_1^{(1)}$

* Stochastic Gradient Descent



label

$x_1^{(3)} \simeq y_1$

$x_2^{(3)} \simeq y_2$

data1 $f_1 = (x_1^{(3)} - y_1)^2 + (x_2^{(3)} - y_2)^2$

data2 f_2

data3 f_3

⋮

$$\frac{\partial f_1}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} \cdot x_i^{(l)}$$

$f_1 + f_2 + f_3 + \cdots f_n$.

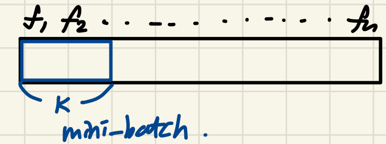
$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - lr \sum_n \frac{\partial f_n}{\partial w_{ij}^{(l)}}$$

(the $\sum_n$ is crossed out)

하나씩 쓰자 = SGD.

초기값에 대해서 update를 해주자.

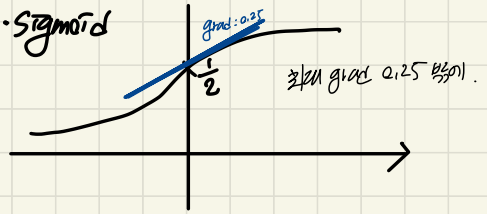
* with mini-batch.



$$w_{ij}^{(l)} - lr \sum_k \frac{\partial f_k}{\partial w_{ij}^{(l)}} \rightarrow w_{ij}^{(l)}$$

* Vanishing Gradient. Problem.

- Sigmoid

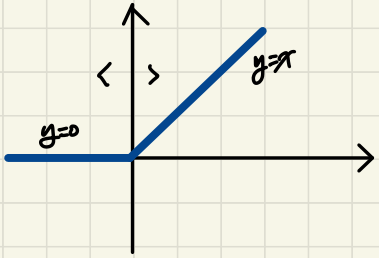


⇒ hidden layer가 깊어질수록 W가 점점 0에 가까워짐.

gradient update가 일어나지 않음

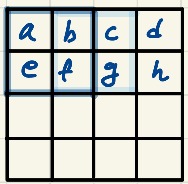
⇓

Grad가 큰 act func을 사용하자 : ReLu.

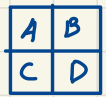


* CNN.

> Convolution?



filter



aA bB
eC fD

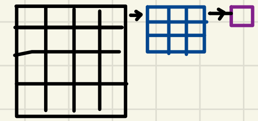
CNN learns filter!

↓

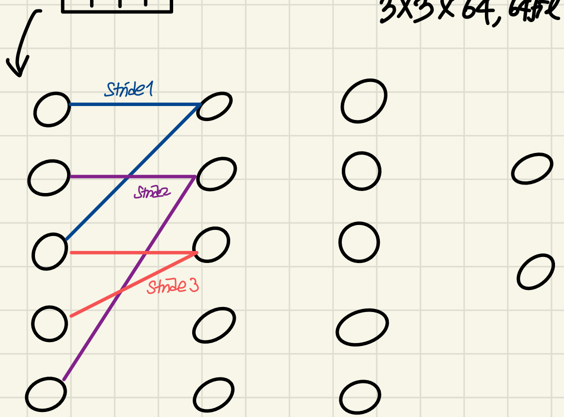
image의 feature를 뽑아내는 것이 특출나다.
overfit이 강해짐.

> stride : filter를 몇칸씩 이동

> pooling.



> Shifting



input → filter (32)

84x84x4 → 8x8x4, stride 4.

↓

3x3x64, 64fil ← 4x4x32, 64fil