# 〈Part 2〉 데이터 사이언스 필수 수학

## [Ch 01.] 딥러닝 네트워크의 연산

## [CH03.] 01,02

Lecture.3
Sigmoid and Softmax

# <Lecture.3 Sigmoid and Softmax>

## CH03_01.[이론정리]_Logit_and_Sigmoid.

딥러닝 학습방법

- Loss : 데이터셋과 모델이 만든 아웃풋이 서로 얼마나 다른지 측정
  → 차이 줄어들게 하는 것이 DL의 학습의 개념.
- sigmoid, Softmax → 이미지 분류에 많이 쓰임. image classification

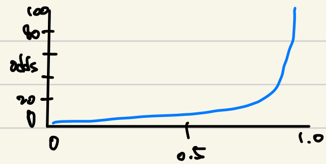
- DL → CV, NLP
  ↳ RNN

- **odds** : 확률을 표현하는 방법 : 내가 원하는 사건이 일어날 확률 / 안 일어날 확률

  $O = \frac{p}{1-p}$ 　 $p$ : 앞면 나올 확률 / $1-p$ : 뒷면 나올 확률

  ex) 앞쪽으로 휘어진 동전 $P(\text{앞}:H) = 0.7$ 　 $P(\text{뒤}:T) = 0.3$

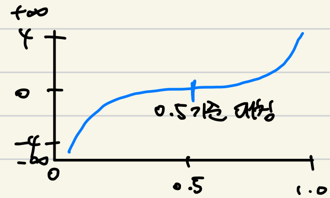
  → $O$가 1보다 크면 앞면 나올 확률이 더 큰 것.




- <logit>

$$l = \log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$
$$= \log p - \log(1-p)$$

ex) $P = 0.5 \rightarrow l = 0$




★ $-\infty < l < \infty$, 　 $p = 0.5$ 대칭.

- <Logit and Sigmoid>

$$l = \log\left(\frac{p}{1-p}\right)$$
$$e^l = \frac{p}{1-p}$$
$$\frac{1}{e^l} = \frac{1-p}{p} = \frac{1}{p} - 1$$
$$\frac{1}{e^l} + 1 = \frac{1}{p}$$

$$\rightarrow \frac{1+e^l}{e^l} = \frac{1}{p}$$
$$p = \frac{e^l}{1+e^l}$$
$$\therefore p = \frac{1}{1+e^{-l}}$$

$$p = \sigma(l) = \frac{1}{1+e^{-l}}$$

시그모이드 $\sigma(z) = \frac{1}{1+e^{-z}}$와 같음.

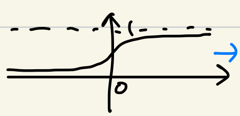
역함수

$p \rightarrow l = \log\left(\frac{p}{1-p}\right) \rightarrow l$ : logit 함수 (odds를 입력받아 logit으로 바꿈)

$l \rightarrow p = \frac{1}{1+e^{-l}} \rightarrow p$ : 시그모이드 (logit을 입력받아 확률로 바꿈)

↳ $-\infty < l < \infty$

$-\infty < (x)^T w + b < \infty$




→ 0 ~ 1 사이값
→ 확률로써 사용.

• z값을 logit으로 사용 + sigmoid → 확률로서 표현 가능.
→ 이 모델이 예측하는 확률 P가 output이 됨.

$$P$$
$$\uparrow$$
$$\boxed{p = \frac{1}{1+e^{-l}}}$$
$$\uparrow$$
$$-\infty < l < \infty$$
$$-\infty < (\vec{x})^T \vec{w} + b < \infty$$
$$\parallel$$
$$z$$

☆ sigmoid는 z값을 logit으로 사용해 P를 output으로 나타내는 함수이다.

- binary classification 에서 T/F의 확률, 앞/뒤의 확률.

$$P$$
$$\uparrow$$
$$\sigma(z)$$
$$\uparrow$$
$$(\vec{x})^T \vec{w} + b$$
$$\uparrow$$
$$(\vec{x})$$

☆ $\text{logit} \xrightarrow[\text{sigmoid}]{} P$ ☆

- <from Logit to Probability>

$$\uparrow \quad 0 \le p \le 1$$
$$p = \frac{1}{1+e^{-z}}$$
$$\uparrow \quad -\infty < z < \infty$$
$$z = (\vec{x})^T \vec{w} + b$$
$$\uparrow$$
$$(\vec{x})$$

⇒ logistic regression


dense layer의 마지막 한 뉴런에 activation = sigmoid
→ p를 output

$$\vec{p} \in \mathbb{R}^{N \times 1}$$

sigmoid $\boxed{\vec{p} = \frac{1}{1+e^{-\vec{z}^{[L]}}}}$

$$\vec{z}^{[L]} \in \mathbb{R}^{N \times 1}$$

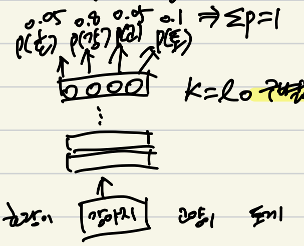
마지막 layer 1개 뉴런: $\boxed{\vec{z}^{[L]} = X^T \vec{w}^{[L]} + b^{[L]}}$   $\vec{w}^{[L]} \in \mathbb{R}^{l_I}$, $b^{[L]} \in \mathbb{R}$

$$\underset{N \times l_I}{X^T} = \underbrace{\begin{pmatrix} \leftarrow & (\vec{x}^{(1)})^T & \rightarrow \\ \leftarrow & (\vec{x}^{(2)})^T & \rightarrow \\ & \vdots & \\ \leftarrow & (\vec{x}^{(N)})^T & \rightarrow \end{pmatrix}}_{l_I}$$

output layer 직전 결과

# CH03-02. [이론강의]_Softmax_Layer

- <Softmax Layer>




$k = l_O$ 구별해야하는 class 수 = 마지막 layer 뉴런수

- 마지막 layer

$(\vec p)^T$ : $p_1 \quad p_2 \quad \cdots \quad p_K$

$p_i = p(C_i),\ 1 \le i \le k$

$\rightarrow \sum_{i=1}^{k} p_i = 1$

softmax: $S_j((\vec l)^T) = p_j = \dfrac{e^{l_j}}{\sum_{k=1}^{K}[e^{l_k}]}$

$(\vec l)^T$ : $l_1 \quad l_2 \quad \cdots \quad l_K$

logit1 logit2 logitK

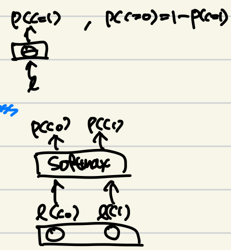
$-\infty < z_1 < \infty \quad z_2 \quad \cdots \quad z_K$

① ② ⋯ Ⓚ

k : class 수.

sigmoid : $l \rightarrow p$ : binary

softmax : $\vec l \rightarrow \vec p$ : 여러개 class




$p_1 \quad p_2 \quad \cdots \quad p_K$ $\quad (\vec p)^T \in R^{1\times K}$

$\vec P^T = \begin{pmatrix} \leftarrow \vec p^{(1)T} \rightarrow \\ \leftarrow \vec p^{(2)T} \rightarrow \\ \vdots \\ \leftarrow \vec p^{(N)T} \rightarrow \end{pmatrix} \in R^{N\times K}$

첫번째 sample이 강아지일 P, 호랑이일 P, …

softmax $\boxed{S_j((\vec z^{[O]})^T) = p_j = \dfrac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}}$

logit으로 출력

마지막 layer 뉴런수 = 구별해야할 class 수(K)

$(\vec z^{[O]})^T \in R^{1\times l_O}$, $l_O = K$

$(\vec Z^{[O]})^T = \begin{pmatrix} \leftarrow (\vec z^{[O](1)})^T \rightarrow \\ \leftarrow (\vec z^{[O](2)})^T \rightarrow \\ \vdots \\ \leftarrow (\vec z^{[O](N)})^T \rightarrow \end{pmatrix} \in R^{N\times l_O},\ l_O = K$

첫번째 sample이 강아지일 logit, 호랑이일 logit…

$\boxed{(\vec z^{[O]})^T = (\vec a^{[O-1]})^T W^{[O]} + (\vec b^{[O]})^T}$

$(\vec a^{[O-1]})^T$ 마지막 layer 전단계 output