Backpropagation

1. Optimization
2. Differentiation
3. Fully-connected
4. Chain Rule
5. Matrix form
6. Convolution

1. Optimization

Consider linear regression model such that $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_k x_k + \varepsilon$ where, $\varepsilon \sim N(0, \sigma_\varepsilon^2)$

Data given $\Rightarrow$ we can write model with data as $Y = X\beta + \varepsilon$ where

$$\left\{ \begin{array}{ll} Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} & X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1k} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{pmatrix} \\ \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix} & \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix} \end{array} \right.$$

Then, we can find out $\hat{\beta}$ by minimizing ε^2.

Because it's obvious that ε^2 is convex, what we need to do is finding $\hat{\beta}$ s.t $\left.\frac{\partial \varepsilon'\varepsilon}{\partial \beta}\right|_{\beta=\hat{\beta}} = 0$

$\therefore \quad \hat{\beta} = (X'X)^{-1}X'Y$

However it's computational complexity is $O(k^3)$ it means too high.

Thus, we use gradient descent.

Although, first method which figures out $\hat{\beta}$ as minimizer of $\epsilon'\epsilon$, is closed form, gradient descent is more efficient.

따라서 $\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \eta \frac{\partial \epsilon'\epsilon}{\partial \hat{\beta}}$ 으로 하여 순차적으로 update 합니다

또한 output = f(input, parameter), loss = h(output, target)

이때 input : constant, parameter : variable 이므로

Optimization : loss를 최소로 만들어 주는 parameter를 찾는 것

2. Differentiation (Gradient)

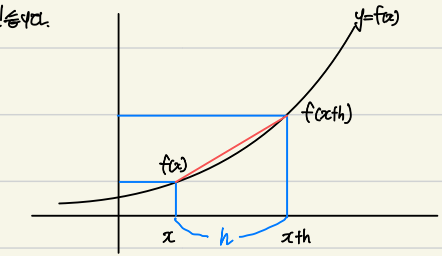
흔히 기울기라고 하는 것은 $\frac{\Delta y}{\Delta x}$, 즉, $\frac{y_2-y_1}{x_2-x_1}$ 입니다.

특히 이를 $\frac{dy}{dx}$ 와 혼용하여 사용되기도 하는데 엄밀 하게는 Δx는 x_2-x_1 이고 dx는 $\forall \varepsilon>0$, $\|x_2-x_1\|<\varepsilon$ 정도로 작은것을 의미합니다

그래서 보통 $\lim_{\Delta x\to 0} \frac{\Delta y}{\Delta x} = \frac{dy}{dx}$ 라고 하며 이를 "미분" or "순간 기울기" 라고 합니다.

이때 $y=f(x)$가 다변수 함수 이면 사실 같은 의미인데 어렵다고 느낄수 있습니다.

다시 일변수 일때 미분을 봅시다.

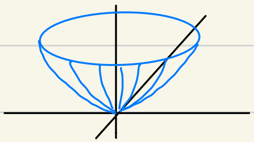


$$f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{(x+h)-(x)}$$ 이며 이를 graph 상에서 나타내보면

빨간색의 직선의 기울기가 $h\to 0$ 이므로 $x+h \to x$ 이고 이는 x 에서의 순간 기울기로 볼수 있습니다

따라서 $f'(x)$는 $f(x)$의 기울기 함수이며 도함수라고 합니다.

여기 이변수 $z = f(x,y) = x^2+y^2$ 을 생각해 봅시다.



와 같은 모양입니다. 그럼 이때 미분은 어떻게 정의할까요?

예를 들어 위의 일변수에서 $f'(a)$ 는 $x=a$ 에서의 $f(a)$ 의 기울기 를 의미합니다.

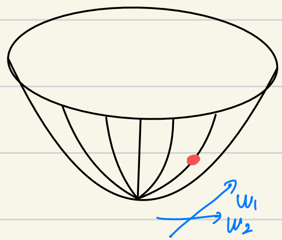
그럼 이 변수에서 $f'(a,b)$ 는 어떤 값을 가져야 하나요?

일 변수는 방향이 x 만 있었으므로 scalar 로, 다변수는 방향이 여러개이므로 각 방향에 대한 기울기를 구해줘야 합니다.

즉, $f'(a,b) = \left(\lim_{h\to 0} \frac{f(a+h,b) - f(a,b)}{h} , \lim_{h\to 0} \frac{f(a,b+h) - f(a,b)}{h} \right)$ 가 되고 이는 각 방향에 대한 기울기가 됩니다.

Thus, we denote $\nabla f = \left(\frac{\partial f_1}{\partial x_1}, \cdots, \frac{\partial f_n}{\partial x_n} \right)$

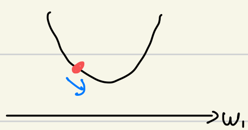
이제 실제 어떻게 update 되는지 보겠습니다.   Remind : 목표는 loss value 를 최소로 만들어 주는 w 를 찾는 것이다.



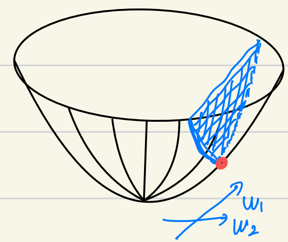
빨간색이 최초의 loss 라면 $\nabla loss|_{\bullet}$ 을 구하고 이를 이용하여 증가되는 반대 방향으로 움직여 주면 됩니다.

이때 vector 이므로 길이만큼 axis가 존재하고 모든 axis에 대해 moving 해야 합니다.

즉, 단면으로 볼때



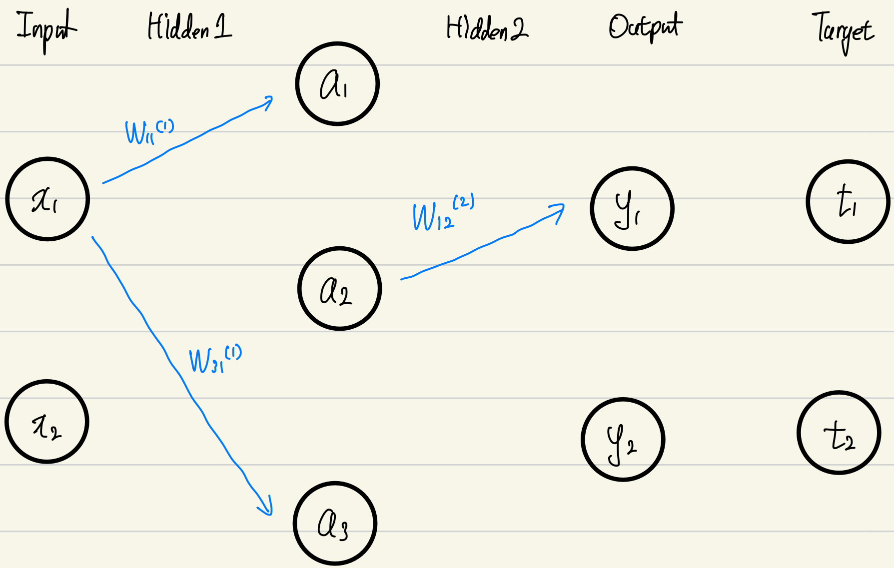
만을 진행한다면



으로 밖에 못 가므로

모든 axis에 대하여 parameter moving을 합니다.

3. Simple fully connected classification model.



input-D = 2, class : 3

no bias

$$\text{sigmoid} = \frac{1}{1+e^{-x}}$$

Cross entropy

$$\text{softmax} = \frac{\exp(y_i)}{\sum \exp(y_i)}$$

Feed forward   let $S(x)$: sigmoid , $P(x)$: softmax

$$a_1 = S(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2) \qquad y_1 = P(W_{11}^{(2)} a_1 + W_{12}^{(2)} a_2 + W_{13}^{(2)} a_3)$$

$$a_2 = S(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2) \qquad y_2 = P(W_{21}^{(2)} a_1 + W_{22}^{(2)} a_2 + W_{23}^{(2)} a_3)$$

$$a_3 = S(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2)$$

$$loss = -\sum_{i \in C} t_i \log(y_i) = -t_1 \log y_1 - t_2 \log y_2$$

또는 binary 이니까 $-t_1 \log x_1 - (1-t_1)\log(1-x_1)$

이제 목표는 loss를 최소화 시켜주는 W를 찾는 것이다.

즉, 이를 $loss = f(W)$ 라고 생각하면 된다. 또한 $W^{(1)}$ $W^{(2)}$ 의 순서를 생각하지 말자. 의미없다.

따라서 모든 $W_{ij}^{(k)}$ 에 대하여 $\nabla loss$ 를 계산하면 됩니다.

gradient. important ☆ : weights 만이 variable 입니다.

$W_{12}^{(2)}$의 gradient 를 구해봅시다.

by definition $$\nabla = \lim_{h \to 0} \frac{loss(w_{12}^{(2)}+h, \cdots\cdots) - loss(w_{12}^{(2)}, \cdots)}{h}$$

실제로 code 에서는 h 를 충분히 작은 수로 할당 해주면 됩니다.

하지만 위 과정은 시간이 상당히 오래 걸립니다. 또한 h에 대한 오차도 있습니다.

또는 도함수를 직접 계산해도 되는데 deep 이면서 nodes가 1000 이면 .....

4. Chain Rule

loss function 은 각 layer, softmax, sigmoid 등이 있으므로 합성함수로 나타낼수 있습니다.

이를 이용하여 gradient 를 쉽게 구할 수 있습니다.

Let $h(x) = f(g(x))$, $y = g(x)$

$$h'(x) = \frac{dh}{dx} = \frac{dh}{dx}\frac{dy}{dy} = \frac{dh}{dy} \times \frac{dy}{dx}$$ 입니다.

즉, $x=a$ 라는 point 에서 gradient 는 $f'(a) = \left.\frac{dh}{dy}\right|_{y=b} \times \left.\frac{dy}{dx}\right|_{x=a}$ where $b = g(a)$

$y = h(x)$

e.g) $f(x) = x^2+2x+1 = g(h(x))$ where $g(x) = x^2$, $h(x) = x+1$

gradient at $x = 2$

i) $f'(x)|_{x=2} = 2(x+1)|_{x=2} = 6$

ii) $f'(x)|_{x=2} = \left.\frac{df}{dx}\right|_{x=2} = \left.\frac{df}{dy}\right|_{y=3} \times \left.\frac{dy}{dx}\right|_{x=2}$

$$= \left.\frac{dy^2}{dy}\right|_{y=3} \times \left.\frac{d(x+1)}{dx}\right|_{x=2} = 2y|_{y=3} \times 1|_{x=2} = 6$$

이를 이용하여 $w_{ij}^{(k)}$ 에서의 gradient를 계산해 보겠습니다.

예를들어 $\frac{\partial Loss}{\partial w_{12}^{(2)}}$ 라면 softmax 를 고려해야 합니다. 따라서 이러한 함수를 미리 미분합니다.

sigmoid and softmax

$$\begin{cases} \text{sigmoid} \quad S(x) = \dfrac{1}{1+\exp(-x)} \\ \text{softmax} \quad P(x_i) = \dfrac{\exp(x_i)}{\sum \exp(x_i)} \end{cases}$$

→ 해당 value 만을 이용

→ 모든 output을 이용

보통 sigmoid는 Activation softmax는 k개의 class 일때 probability를 구하는 용도로 사용

또한 sigmoid는 Binary 이고 이를 k개의 class로 확장하면 softmax가 된다. 이를 이해하기위해서는 logit 부터 알아야 합니다.

Logit = log + odds

odds = 성공/실패 ratio 이때 logit $L = \ln \frac{p}{1-p}$

⇒ 즉, odds는 기준이 1 이고 logit은 기준이 0 이된다. ⇒ logistic regression 이 나옴

Sigmoid model의 output에서 probability p를 구하고 싶다면 logit을 이용하면 된다.

즉, $L = \ln \frac{p}{1-p}$ 에서 $L ==$ output 이라고 하여 p에 대해 정리하면 $p = \frac{1}{1+\exp(-t)}$ 이 된다.

$\therefore$ sigmoid : $f(x) = \frac{1}{1+e^{-x}}$

softmax 이는 binary의 sigmoid를 multi-class로 확장한 개념이다.

Induction

binary case) $e^{t_1} = \frac{P_1}{1-P_1} = \frac{P_1}{P_2}$ where $P_i = P(Y=C_i \mid x)$ x: given data.

then for k classes let $i = 1, 2, \cdots k$ $e^{t_i} := \frac{P_i}{P_k}$ —① , thus $\sum_{i=1}^{k-1} e^{t_i} = \sum_{i=1}^{k-1} \frac{P_i}{P_k} = \frac{1}{P_k} \sum_{i=1}^{k-1} P_i$

and because $\sum_{i=1}^{k} P_i = 1$ $P_k = 1 - \sum_{i=1}^{k-1} P_i$ $\therefore \sum_{i=1}^{k-1} e^{t_i} = \frac{1-P_k}{P_k}$

$\therefore P_k = \frac{1}{\sum_{i=1}^{k-1} e^{t_i} + 1}$, also as we defined $P_i = e^{t_i} P_k = \frac{e^{t_i}}{\sum_{i=1}^{k-1} e^{t_i} + 1}$, because $e^{t_k} = \frac{P_k}{P_k} = 1$

$$\therefore P_i = \frac{e^{t_i}}{\sum_{i=1}^{k} e^{t_i}}$$

i) Sigmoid $\quad S(x) = \dfrac{1}{1+e^{-x}} = \dfrac{e^x}{e^x+1} \Rightarrow \dfrac{\partial S}{\partial x} = \dfrac{e^x(1+e^x) - e^{2x}}{(1+e^x)^2} = \dfrac{e^x}{(1+e^x)^2} = \dfrac{1 \times e^x}{(1+e^x)(1+e^x)} = S(x)\{1-S(x)\}$

ii) Softmax $\quad P(x_j) = \dfrac{\exp(x_j)}{\sum \exp(x_i)} \Rightarrow \dfrac{\partial P}{\partial x_j} = \dfrac{1}{\{\sum \exp(x_i)\}^2} \{\exp(x_j)\}\{\sum \exp(x_i) - \exp(x_j)\}$

다 사용해서 case가 나뉘어.

$$= \frac{\exp(x_j)}{\sum \exp(x_i)} \left\{ \frac{\sum \exp(x_i) - \exp(x_j)}{\sum \exp(x_i)} \right\} = \frac{\exp(x_j)}{\sum \exp(x_i)} \left\{ 1 - \frac{\exp(x_j)}{\sum \exp(x_i)} \right\} = P(x_j)\{1 - P(x_j)\}$$

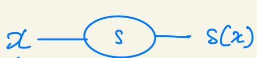
이거 끝이 아닌게 $P(x_j)$를 x_l에 대해서로 미분하는 경우가 있습니다.

$$\frac{\partial P}{\partial x_l} = \frac{\partial}{\partial x_l} \left\{ \frac{\exp(x_j)}{\sum \exp(x_i)} \right\} = \exp(x_j) \frac{\partial}{\partial x_l} \left\{ \frac{1}{\sum \exp(x_i)} \right\} = \exp(x_j) \left\{ \frac{-\exp(x_l)}{\{\sum \exp(x_i)\}^2} \right\}$$

$$= -\frac{\exp(x_j)}{\sum \exp(x_i)} \frac{\exp(x_l)}{\sum \exp(x_i)} = -P(x_j)P(x_l)$$

$$\therefore \frac{\partial P(x_j)}{\partial x_l} = \begin{cases} P(x_j)\{1 - P(x_j)\} & \text{where } j = l \\ -P(x_j)P(x_l) & \text{where } j \neq l \end{cases} \Longleftrightarrow P(x_j)\{I_{(l=j)} - P(x_l)\}$$

여기서 미분값이 의미하는 건



여기 역전파로 아래층에 전달되는 값이다.

iii) Cross-entropy. $\quad -\sum_{i=1}^{c} t_i \log(y_i)$

$$\frac{\partial}{\partial y_j}\left(-\sum_{i=1}^{c} t_i \log(y_i)\right) = \frac{\partial}{\partial y_j}\left(-t_j \log(y_j)\right) = -t_j \frac{1}{y_j}$$

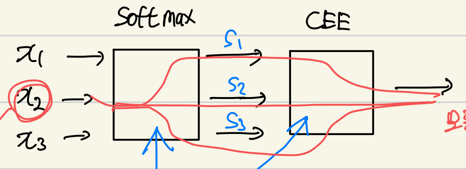
iv) softmax with cross-entropy

둘이 같이 다니면 상당히 쉬워지는데 이를 ii) iii) 의 결과를 이용해보고 또한 직접 합성해서도 구해보겠습니다.

1) 위의 결과 이용 즉, let $f(x_j) = CEE(softmax(x_j))$, let $CEE = L(x)$, $softmax = P(x)$, t_i = target

$$\Rightarrow \frac{\partial}{\partial x_j} f(x_j) = \frac{\partial f(x_j)}{\partial P(x_j)} \times \frac{\partial P(x_j)}{\partial x_j}$$ Wrong임.

⇒ 이렇게 하면 틀린 이유



모든 값에 다 관여했음.

예를 들자면

다 섞임. 즉, x_1에 대한 gradient를 구하려면 s_1, s_2, s_3 를 모두 고려해야 한다.

뒤에서 matrix 이용하면 깔끔하게 해결됨.

그리고 곱의 미 방법을 이용해 보자면 (by Chain Rule)

$$\frac{\partial}{\partial x_j} f(x_j) = \sum_{l=1}^{k} \frac{\partial f(x_j)}{\partial P(x_l)} \times \frac{\partial P(x_l)}{\partial x_j} = -\sum_{l=1}^{k} \frac{t_l}{P(x_l)} \times P(x_l)\left\{ I_{(l=j)} - P(x_j) \right\} = -\sum_{l=1}^{k} t_l \left\{ I_{(l=j)} - P(x_j) \right\}$$

$$= \sum_{l=1}^{k} t_l P(x_j) - \sum_{l=1}^{k} t_l I_{(l=j)} = P(x_j) \sum_{l=1}^{k} t_l - \sum_{l=1}^{k} t_l I_{(l=j)} = P(x_j) - t_j$$

$\therefore \frac{\partial}{\partial x_j} f(x_j) = P(x_j) - t_j = y_j - t_j$ where y_i : output. 보통 이렇게 안하는 이유를 알았습니다.

한번에 하는 방법.

$$a_1 \cdots a_k \rightarrow \boxed{\text{softmax}} \rightarrow y_1 \cdots y_k \rightarrow \boxed{\text{CEE}} \rightarrow L(a_1 \cdots a_k)$$

log 분리

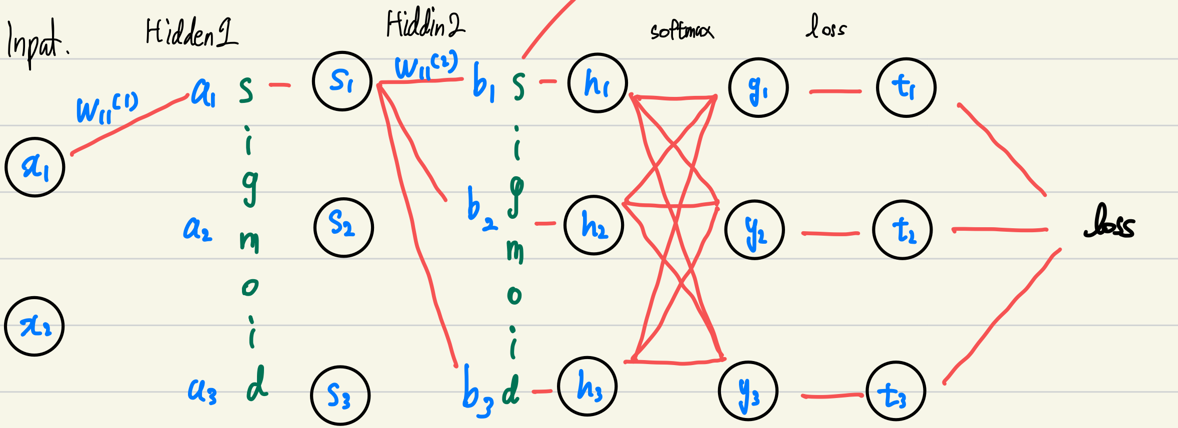
$$L(a_1 \cdots a_k, t_1 \cdots t_k) = -\sum_{i=1}^{k} t_i \log(y_i) = -\sum_{i=1}^{k} t_i \times \log \frac{\exp(a_i)}{\sum_{j=1}^{k} \exp(a_j)} = -\sum_{i=1}^{k} t_i a_i + \underbrace{\sum_{i=1}^{k} t_i}_{1} \log\left\{ \sum_{j=1}^{k} \exp(a_j) \right\}$$

$$\therefore \frac{\partial L}{\partial a_i} = -t_i + \frac{\exp(a_i)}{\sum_{j=1}^{k} \exp(a_j)} = y_i - t_i \qquad \therefore \frac{\partial L}{\partial a} = y - t$$

위에서 구한 값은 중간 과정일 뿐입니다. 이제 진짜 필요한 $W_{11}^{(1)}$을 구해보겠습니다.

다시 처음 그림에서 $W_{11}^{(1)}$의 gradient를 구하기 위한 graph

→ 실제 modeling 할 때는 합쳐서 했다



살짝만 풀어보자면

$$\frac{\partial L}{\partial w_{11}^{(1)}} = \frac{\partial L}{\partial s_1} \times \frac{\partial s_1}{\partial a_1} \times \frac{\partial a_1}{\partial w_{11}^{(1)}} = \frac{\partial L}{\partial s_1} \times \{s_1(1-s_1)\} \times \{x_1\}$$

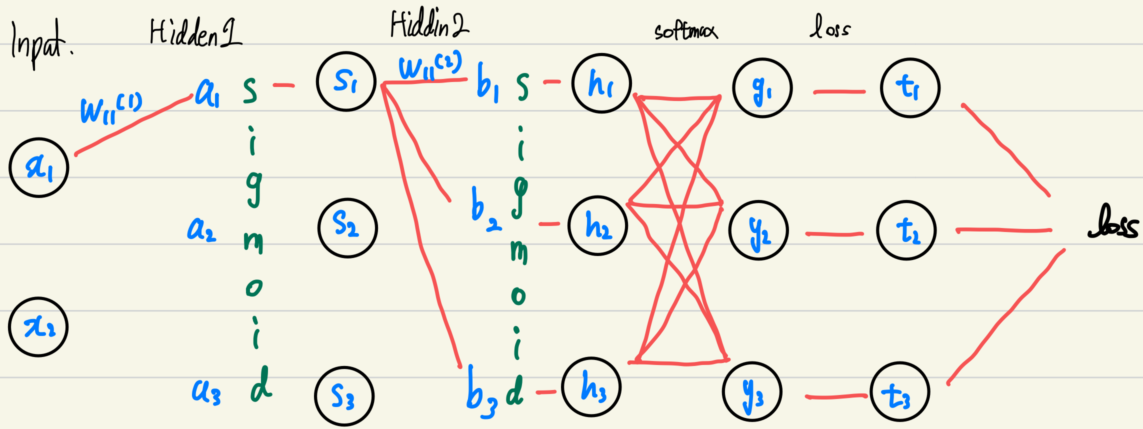
$$\frac{\partial L}{\partial s_1} = \sum_{i=1}^{3} \frac{\partial L}{\partial h_i} \times \frac{\partial h_i}{\partial b_i} \times \frac{\partial b_i}{\partial s_1} = \sum_{i=1}^{3} \frac{\partial L}{\partial h_i} \times \{h_i(1-h_i)\} \times \{w_{i1}^{(2)}\}$$

$$\frac{\partial L}{\partial h_i} = y_i - t_i$$

$$\therefore \frac{\partial L}{\partial w_{11}^{(1)}} = \sum_{i=1}^{3} \{y_i - t_i\} \times \{h_i(1-h_i)\} \times \{w_{i1}^{(2)}\} \times \{s_1(1-s_1)\} \times \{x_1\}$$

5. Matrix form

위에 처럼 하면 하나씩 계산해야 하므로 ⇒ Matrix 로 계산



⟹ by matrix form   이번에는 Linear layer bias == True

$$X \xrightarrow{W_1} a \xrightarrow{\text{sigmoid}} s \xrightarrow{W_2} b \xrightarrow{\text{sigmoid}} h \xrightarrow{\text{softmax}} y \xleftrightarrow{\text{loss}} t$$

$$\downarrow$$

loss value

Feed forward   S: sigmoid   P= softmax

$a = XW_1 + b_1$ matrix는 항상 form 을 생각해야 한다.

$s = S(a)$

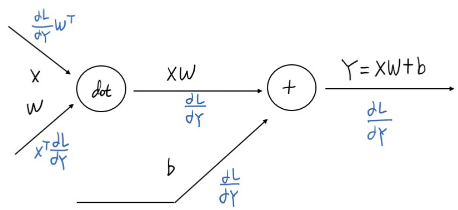
$b = sW_2 + b_2$ $X : 1\times 2$ $W_1 : 2\times 3$ $b_1 : 1\times 3$ $\Longrightarrow$ $a : 1\times 3$

$h = S(b)$ $s : 1\times 3$ $W_2 : 3\times 3$ $b_2 : 1\times 3$ $\Longrightarrow$ $b : 1\times 3$

$y = P(h)$

역전파 (chain Rule)

Affine



i.e) gradient at w,b : $\begin{cases} \frac{\partial L}{\partial w} = X^T\frac{\partial L}{\partial Y} \\ \frac{\partial L}{\partial b} = \frac{\partial L}{\partial Y} \end{cases}$, 이전층으로 들어가는 미분 : $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial Y}W^T$

이 과정, 즉, 중간에 Y를 거치는 과정을

parameterize 라고 생각하면 된다.

Back propagation

위에서 했던 방법과 완전히 같은 방법이다. (e.g) 연립방정식을 Gauss-elimination 으로 푸는것과 같다)

위에서 했던 방법을 다시 생각해 보면

$X \xrightarrow[W_1]{} Y \xrightarrow[W_2]{} Z$ 일때 $\frac{\partial Z}{\partial W_1}$ 를 구하기 위해 Y를 매개변수로 보고 Chain rule 을 이용하였다.

즉 $\frac{\partial Z}{\partial W_1} = \frac{\partial Z}{\partial Y}\frac{\partial Y}{\partial W_1}$ 을 하였다. 이때 W_2 입장에서 보면 미분이 역방향으로 단순히 지나갔을 뿐이다.

이처럼 matrix form 의 미분을 할때에 계산 graph를 이용한다. 또한 크게 두가지의 미분이 계산되는데 첫번째는

해당 layer의 parameter 에 대한 미분 이고 두번째는 이전층으로 흘러들어가는 미분이다.

i) Affine $Y = XW + b$

$$\frac{\partial L}{\partial W} = \frac{\partial Y}{\partial W} \times \frac{\partial L}{\partial Y} = X^T \frac{\partial L}{\partial Y}$$

$$\frac{\partial L}{\partial b} = \frac{\partial Y}{\partial b} \times \frac{\partial L}{\partial Y} = \frac{\partial L}{\partial Y}$$

⇒ 해당 layer에서 거치는 미분이다.

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} W^T$$

⇓

이게 이전층 X로 흘러들어가는 미분이다.