Lecture 6 : Back Propagation

This will be your least favorite lecture, since it requires the most tedious derivations of the whole course. 하하하하

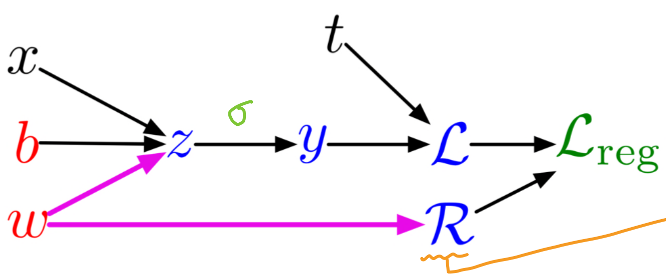
※ lecture 6 , Lecture 8 둘다 BP에 관한 챕터이며
6은 수학적 접근을 통한 자세한 이해를 위한 챕터,
8은 이를 굳이 매번 직접 계산하지 않아도 되게끔 하는 automatic differentiation engine
구현에 대하여 설명한다.

1 Introduction

↳ BP란 신경망에서 손실함수에 대한 각 파라미터(w)의 편미분을
구하는 방식으로 이전에 Linear Regression, Logistic Regression 에서
진행한 것과 동일하게 이 편미분 값을 Gradient descent에 활용한다.

2 The Chain Rule revisited

예시로 든 모델 설정 : 단일 변수, 단일 출력, 단일 샘플 (x, t)



Regularization term
이 강의에서는 크게 중요하지 않으니 이런게 있구나 정도만.
(뒤에 다른 챕터에서 다룸)

$$z = wx + b \quad (1)$$

$$y = \sigma(z) \quad (2)$$

$$\mathcal{L} = \frac{1}{2}(y-t)^2 \quad (3)$$

$$\mathcal{R} = \frac{1}{2}w^2 \quad (4)$$

$$\mathcal{L}_{\text{reg}} = \mathcal{L} + \lambda\mathcal{R}. \quad (5)$$

이 상황에서 학습 = 성능개선 을 하고자 한다면

우리가 바뀌야 하는 값 : w , b

⇒ 즉 우리는 $\mathcal{L}_{reg}$ 에 대한 w의 미분값 $\frac{\partial L_{reg}}{\partial w}$, b 의 미분값 $\frac{\partial L_{reg}}{\partial b}$ 가 필요

2.1 How you would have done it in calculus class

↳ 너네가 1~2학년때 미적분학에서 이걸 어떻게 풀었냐면 ...

$$
\begin{aligned}
\mathcal{L}_{\text{reg}} &= \frac{1}{2}(\sigma(wx+b)-t)^2 + \frac{\lambda}{2}w^2 \\
\frac{\partial \mathcal{L}_{\text{reg}}}{\partial w} &= \frac{\partial}{\partial w}\left[\frac{1}{2}(\sigma(wx+b)-t)^2 + \frac{\lambda}{2}w^2\right] \\
&= \frac{1}{2}\frac{\partial}{\partial w}(\sigma(wx+b)-t)^2 + \frac{\lambda}{2}\frac{\partial}{\partial w}w^2 \\
&= (\sigma(wx+b)-t)\frac{\partial}{\partial w}(\sigma(wx+b)-t) + \lambda w \\
&= (\sigma(wx+b)-t)\sigma'(wx+b)\frac{\partial}{\partial w}(wx+b) + \lambda w \\
&= (\sigma(wx+b)-t)\sigma'(wx+b)x + \lambda w \\
\frac{\partial \mathcal{L}_{\text{reg}}}{\partial b} &= \frac{\partial}{\partial b}\left[\frac{1}{2}(\sigma(wx+b)-t)^2 + \frac{\lambda}{2}w^2\right] \\
&= \frac{1}{2}\frac{\partial}{\partial b}(\sigma(wx+b)-t)^2 + \frac{\lambda}{2}\frac{\partial}{\partial b}w^2 \\
&= (\sigma(wx+b)-t)\frac{\partial}{\partial b}(\sigma(wx+b)-t) + 0 \\
&= (\sigma(wx+b)-t)\sigma'(wx+b)\frac{\partial}{\partial b}(wx+b) \\
&= (\sigma(wx+b)-t)\sigma'(wx+b)
\end{aligned}
$$

← 싹다 전개해서 풀었단다 .... PTSD ..

이게 Chain Rule !!

☆ 이러한 방식의 문제점

1. 계산식이 미궁에 갔다. + 실수를 하기 아주 쉽다 (위 식에서도 틀린게 있대요 찾아보아요)
2. 중복되는 term이 너무많다. (wx+b) term은 값 도출까지 4번이나 적힌다.

↓

BP 의 기본 아이디어는 이러한 중복되는 term을 하나의 모듈처럼 생각하여 공유하는 것이다 !

2.2 Multivariable chain rule: the easy case

↳ 다변수 일때는 Chain Rule 을 어떻게 적용할까?

review

단일 변수 일때 Chain Rule 은 다음과 같은 식이었다. $\left(f(g(x))\right)' = f(g(x))' \cdot g(x)' = \frac{df}{dg} \cdot \frac{dg}{dx}$

그럼 $\left(f(g(x), h(x))\right)' = ?$

Changing x slightly has two effects: it changes g slightly, and it changes h slightly. Each of these effects causes a slight change to f. For infinitesimal changes, these effects combine additively.

↳ 강의에서는 다음과 같이 설명함. x 값을 조정하면 g, h 값이 달라져 f 값이 바뀌는데, 만약 그 변화량이 아주 작다면 f 에 대한 미분 계산이 각 변수의 chain Rule 결과를 합하는 것으로 계산한다.

$$\therefore \left(f(g(x), h(x))\right)' = \frac{\partial f}{\partial g} \cdot \frac{dg}{dx} + \frac{\partial f}{\partial h} \cdot \frac{dh}{dx}$$

이게 핵심

2.3 An alternative notation

↳ 위에서 한걸 좀 세련되고 눈에 들어오는 notation 으로 바꾸자!
(공식 표기법은 아니나, 이어지는 강의 notation 이해를 위해 필수)

함수 L 을 이루는 여러 변수 중 어떤 한 변수 V 에 대해 $\quad \bar{v} \triangleq \frac{\partial L}{\partial v}$

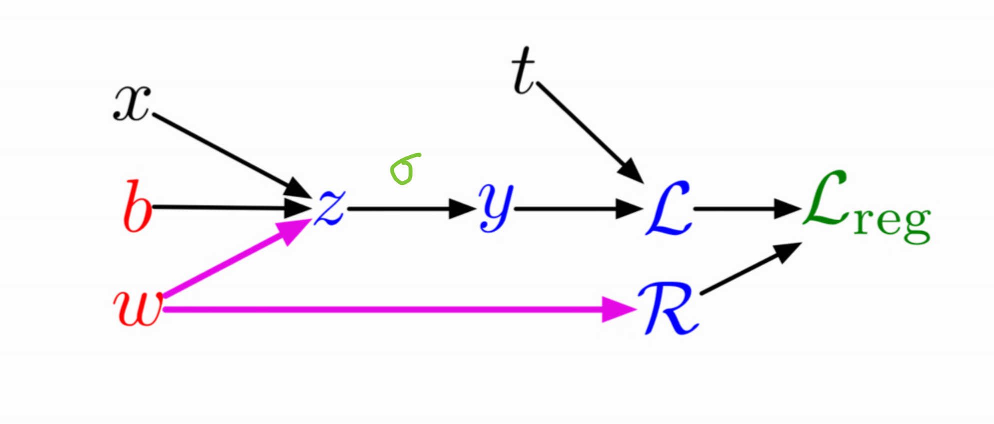
→ 신경망과 관련 지어 생각하자면,

- L = 편미분하고자 하는 가장 상위 함수 = 오차 함수
- V = 업데이트 하고자 하는 파라미터 = 특정 가중치

2.2 결과를 이 표기대로 바꾸면,

$$\left(f(g(x), h(x))\right)' = \frac{\partial f}{\partial g} \cdot \frac{dg}{dx} + \frac{\partial f}{\partial h} \cdot \frac{dh}{dx} = \bar{g} \cdot \frac{dg}{dx} + \bar{h} \cdot \frac{dh}{dx}$$

2.4 Using the computation graph



BP in pseudo code

위 연산 흐름 그래프에서 각 지점을 v, 해당 위치 에서 편미분 값을 $\overline{v}$ 라고 할때,

For $i = 1, \ldots, N$

Compute v_i as a function of $\mathrm{Pa}(v_i)$ ⇒ Forward pass

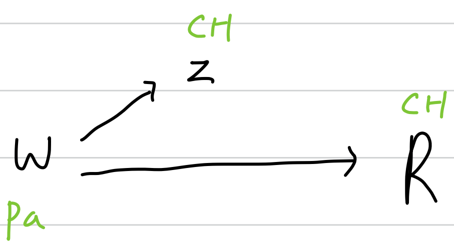
---

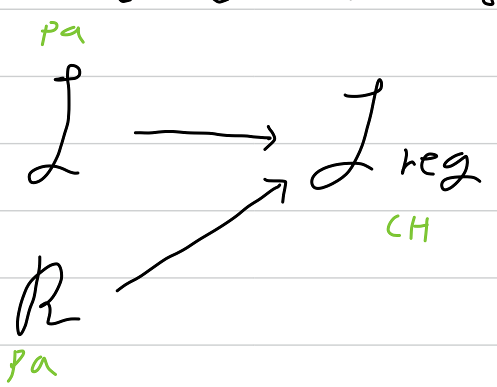
$v_N = 1$

For $i = N-1, \ldots, 1$ ⇒ Backward Pass

$\overline{v_i} = \sum_{j \in \mathrm{Ch}(v_i)} \overline{v_j} \frac{\partial v_j}{\partial v_i}$

Pa() = parent , ch( ) = children 을 의미하며
연산 흐름 그래프에서 부모 자식 관계는 다음과 같다.





수도 코드 흐름대로 $\frac{\partial L_{reg}}{\partial w}(\overline{w})$, $\frac{\partial L_{reg}}{\partial b}(\overline{b})$ 를 계산해보자.

└ 2.1과 비교해보자.

$$
\begin{aligned}
\overline{\mathcal{L}_{\text{reg}}} &= 1 \\
\overline{\mathcal{R}} &= \overline{\mathcal{L}_{\text{reg}}} \frac{\mathrm{d}\mathcal{L}_{\text{reg}}}{\mathrm{d}\mathcal{R}} \\
&= \overline{\mathcal{L}_{\text{reg}}}\, \lambda \\
\overline{\mathcal{L}} &= \overline{\mathcal{L}_{\text{reg}}} \frac{\mathrm{d}\mathcal{L}_{\text{reg}}}{\mathrm{d}\mathcal{L}} \\
&= \overline{\mathcal{L}_{\text{reg}}} \\
\overline{y} &= \overline{\mathcal{L}} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y} \\
&= \overline{\mathcal{L}}\,(y - t) \\
\overline{z} &= \overline{y} \frac{\mathrm{d}y}{\mathrm{d}z} \\
&= \overline{y}\, \sigma'(z) \\
\overline{w} &= \overline{z} \frac{\partial z}{\partial w} + \overline{\mathcal{R}} \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}w} \\
&= \overline{z}\, x + \overline{\mathcal{R}}\, w \\
\overline{b} &= \overline{z} \frac{\partial z}{\partial b} \\
&= \overline{z}
\end{aligned}
$$

최종정리

$$
\begin{aligned}
\overline{\mathcal{L}_{\text{reg}}} &= 1 \\
\overline{\mathcal{R}} &= \overline{\mathcal{L}_{\text{reg}}}\, \lambda \\
\overline{\mathcal{L}} &= \overline{\mathcal{L}_{\text{reg}}} \\
\overline{y} &= \overline{\mathcal{L}}\,(y - t) \\
\overline{z} &= \overline{y}\, \sigma'(z) \\
\overline{w} &= \overline{z}\, x + \overline{\mathcal{R}}\, w \\
\overline{b} &= \overline{z}
\end{aligned}
$$

☆

- 중복 연산이 없다.
- 각 $\overline{v}$ 가 마치 하나의 모듈처럼 처리되고 공유된다.
  - ↳ ex) 만약 다른 Loss Function 을 쓰려면 전체를 다 계산할 필요 없이 $\overline{y}$ 만 바꾸면 된다.

3 Backprop on a multilayer net

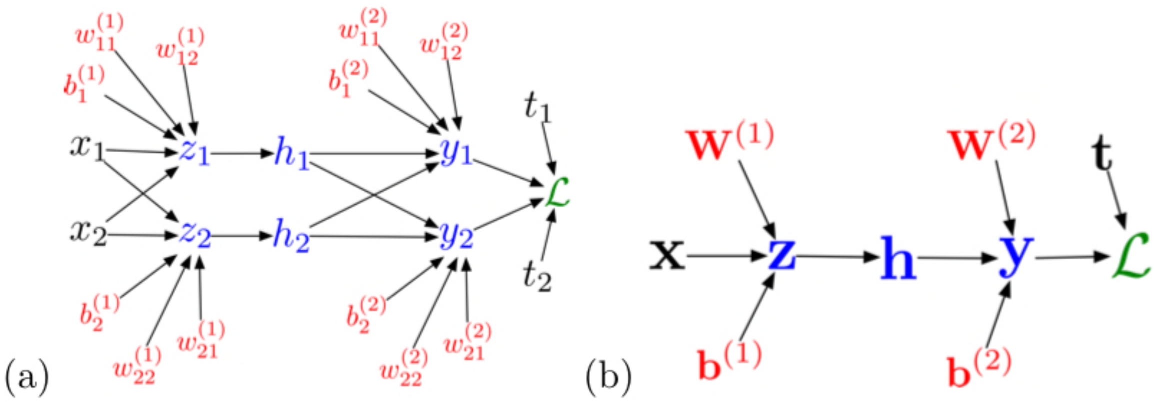


Figure 2: **(a)** Full computation graph for the loss computation in a multi-layer neural net. **(b)** Vectorized form of the computation graph.

↳ 강의 자료 왈 : "식이 조금 지저분해질 뿐, 똑같다",
"$\bar{h}$ 만 조금 달라지는데 multivariate Chain Rule 이 적용된다."

Forward - Pass

$$z_i = \sum_j w_{ij}^{(1)} x_j + b_i^{(1)}$$

$$h_i = \sigma(z_i)$$

$$y_k = \sum_i w_{ki}^{(2)} h_i + b_k^{(2)}$$

$$\mathcal{L} = \frac{1}{2} \sum_k (y_k - t_k)^2$$

Backward Pass

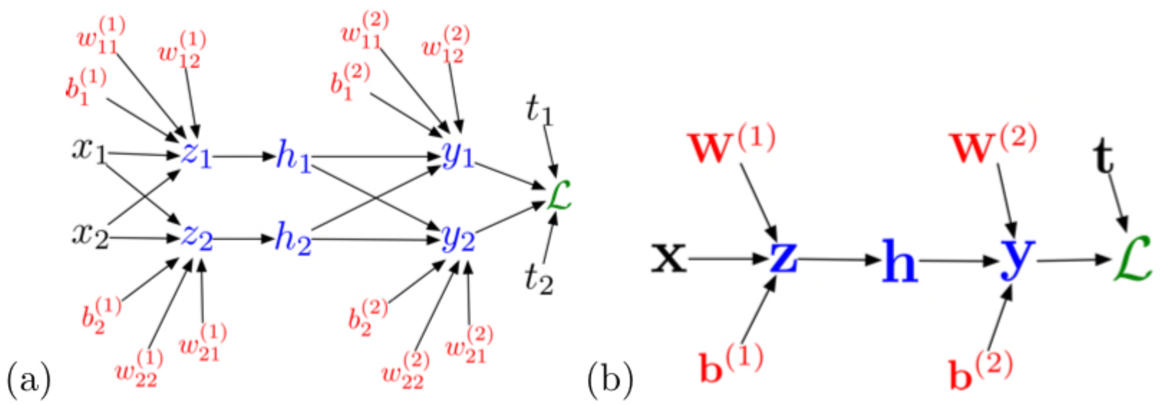


Figure 2: **(a)** Full computation graph for the loss computation in a multi-layer neural net. **(b)** Vectorized form of the computation graph.

$$\overline{\mathcal{L}} = 1$$

$$\overline{y_k} = \overline{\mathcal{L}}\,(y_k - t_k)$$

$$\overline{w_{ki}^{(2)}} = \overline{y_k}\,h_i$$

$$\overline{b_k^{(2)}} = \overline{y_k}$$

$$\overline{h_i} = \sum_k \overline{y_k} w_{ki}^{(2)}$$

⇒ 화살표가 두우우우개

$$\overline{z_i} = \overline{h_i}\,\sigma'(z_i)$$

$$\overline{w_{ij}^{(1)}} = \overline{z_i}\,x_j$$

$$\overline{b_i^{(1)}} = \overline{z_i}$$

식이 혹시 헷갈리면

$\overline{V_1} = \overline{V_2} \cdot \square$ 꼴의

식에서 $V_1 \leftrightarrow V_2$ 사이

관계식을 Forward pass

에서 찾아서 V_1으로

미분 해보자.

$$
\begin{aligned}
\overline{\mathcal{L}} &= 1 \\
\overline{\mathbf{y}} &= \overline{\mathcal{L}}\,(\mathbf{y} - \mathbf{t}) \\
\overline{\mathbf{W}^{(2)}} &= \overline{\mathbf{y}}\mathbf{h}^\top \\
\overline{\mathbf{b}^{(2)}} &= \overline{\mathbf{y}} \\
\overline{\mathbf{h}} &= \mathbf{W}^{(2)\top}\overline{\mathbf{y}} \\
\overline{\mathbf{z}} &= \overline{\mathbf{h}} \cdot \sigma'(\mathbf{z}) \\
\overline{\mathbf{W}^{(1)}} &= \overline{\mathbf{z}}\mathbf{x}^\top \\
\overline{\mathbf{b}^{(1)}} &= \overline{\mathbf{z}}
\end{aligned}
$$

⇒ Vectorized Notation

텀원 반응보고 skip 결정.

Note: $\overline{W^{(2)}}$ = 2 X 2 matrix = $\begin{pmatrix} \overline{W_{11}} & \overline{W_{12}} \\ \overline{W_{21}} & \overline{W_{22}} \end{pmatrix}$

$\overline{y}$ = 2x1 matrix = $\begin{pmatrix} \overline{y_1} \\ \overline{y_2} \end{pmatrix}$

h^T = 1x2 matrix = (h_1, h_2)

4 Appendix: why the weird notation?

↳ not important, why $\overline{V}$?

+ Lecture 8 이 뭐하는지 그냥 그림만 한번 훑기