Neural network

A gentle introduction: Neural network for supervised learning

Khiem Nguyen

Email	khiem.nguyen@glasgow.ac.uk
MS Teams	khiem.nguyen@glasgow.ac.uk
Whatsapp	+44 7729 532071 (Emergency only)

May 18, 2025

University of Glasgow

Table of Contents

1. Backpropagation
   - Computation Graph
   - Back propagation for logistic regression
2. Deep neural network with more than one hidden layer
   - Forward propagation
   - Back propagation
3. Implementation in PyTorch
   - Building blocks of a deep neural network
   - Build a simple neural network in PyTorch

Chain rule in differentiation

- We review the chain rule. Assume that

$$\begin{aligned} f &= f(a) \\ a &= a(\theta) \end{aligned}$$

- Then, the chain rule says

$$\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial \theta}$$

Computation Graph

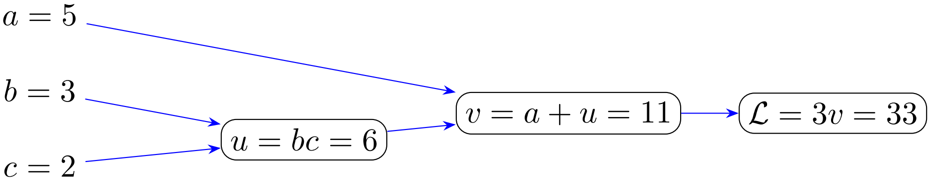
$$\underbrace{3(\underbrace{a + \underbrace{bc}_{u}}_{v})}_{\mathcal{L}} \quad \mathcal{L}(a, b, c) = 3(a + bc)$$

$$u = bc \quad \rightarrow \quad v = a + u \quad \rightarrow \quad \mathcal{L} = 3v \tag{1}$$

Computation Graph

$$\mathcal{L}(a,b,c) = 3\underbrace{(\underbrace{a+\underbrace{bc}_{u}}_{v})}_{\mathcal{L}}$$

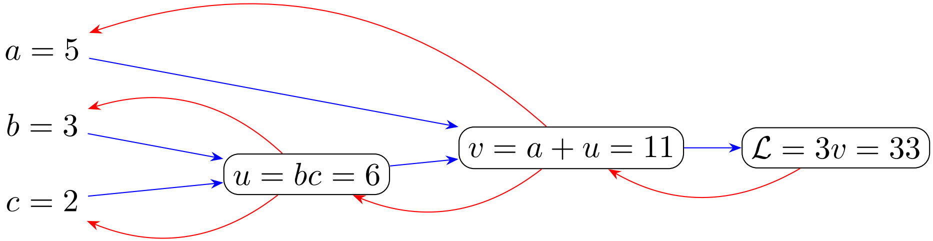
$$u = bc \quad \rightarrow \quad v = a + u \quad \rightarrow \quad \mathcal{L} = 3v \tag{1}$$



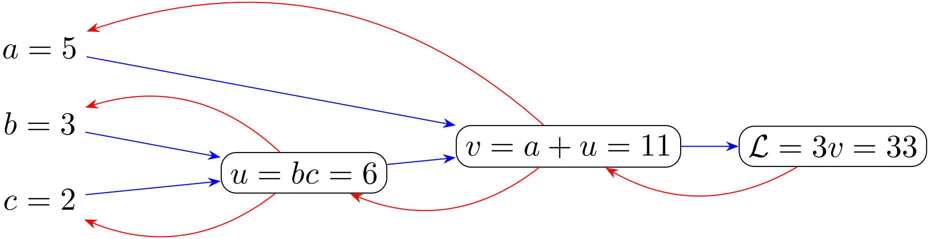
Computation Graph

$$\mathcal{L}(a,b,c) = \underbrace{3(\underbrace{a+\underbrace{bc}_{u}}_{v})}_{\mathcal{L}}$$

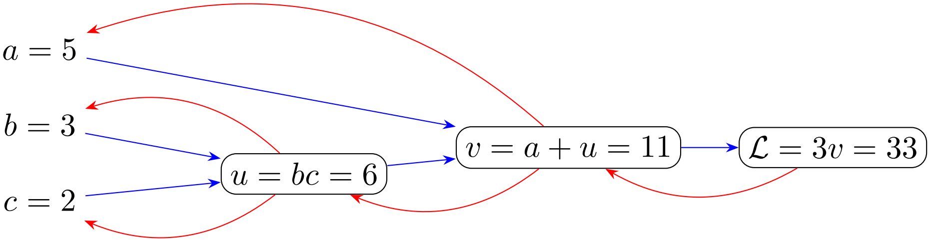
$$u = bc \quad \rightarrow \quad v = a + u \quad \rightarrow \quad \mathcal{L} = 3v \tag{1}$$



Derivatives with a Computation Graph



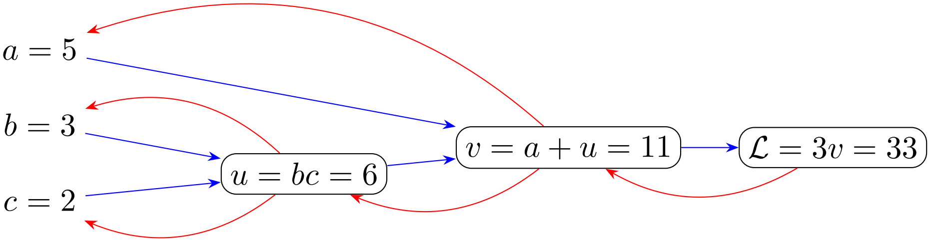
Derivatives with a Computation Graph



$$\frac{\partial \mathcal{L}}{\partial v} = 3$$

$$\frac{\partial \mathcal{L}}{\partial a} = \underbrace{\frac{\partial \mathcal{L}}{\partial v}}_{=3} \cdot \underbrace{\frac{\partial v}{\partial a}}_{=1} = 3 \cdot 1 = 3$$

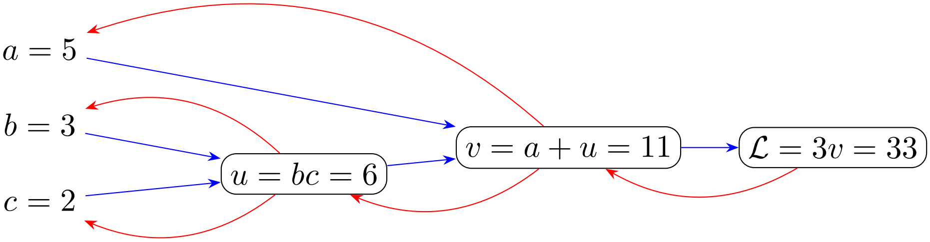
Derivatives with a Computation Graph



$$\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial \mathcal{L}}{\partial v} \cdot \frac{\mathrm{d}v}{\mathrm{d}u} = 3 \cdot 1 = 3$$

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}b} = \underbrace{\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}u}}_{=3} \cdot \underbrace{\frac{\mathrm{d}u}{\partial b}}_{=2} = 3 \cdot 2 = 6$$

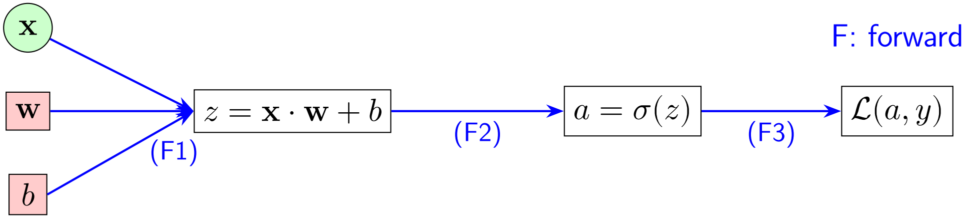
Derivatives with a Computation Graph



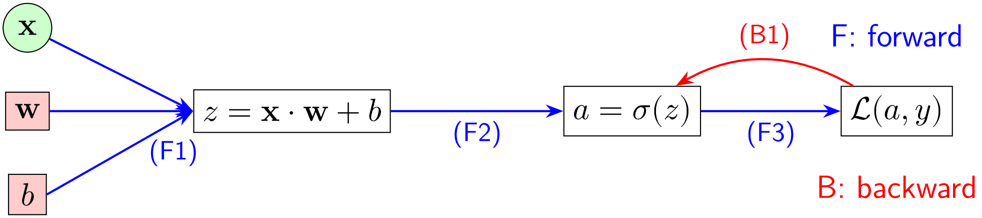
$$\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial \mathcal{L}}{\partial v} \cdot \frac{\partial v}{\partial u} = 3 \cdot 1 = 3$$

$$\frac{\partial \mathcal{L}}{\partial c} = \underbrace{\frac{\partial \mathcal{L}}{\partial u}}_{=3} \cdot \underbrace{\frac{\partial u}{\partial c}}_{=3} = 3 \cdot 3 = 9$$

Back propagation for Logistic Regression

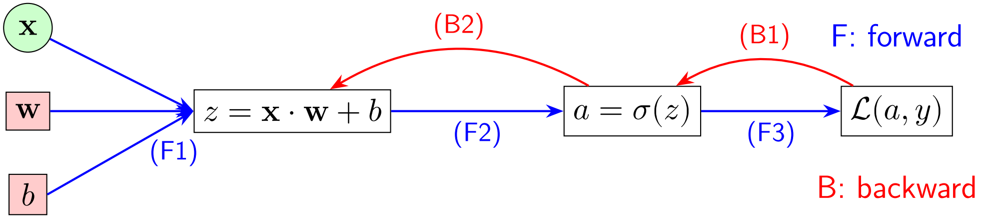


Back propagation for Logistic Regression



(B1): $\dfrac{\partial \mathcal{L}}{\partial a} = \dfrac{\partial}{\partial a}\big[-y\log(a) - (1-y)\log(1-a)\big] = -\dfrac{y}{a} + \dfrac{1-y}{1-a}$

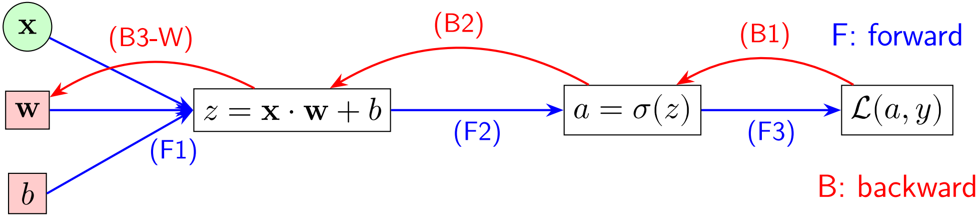
Back propagation for Logistic Regression



(B1): $$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial}{\partial a}\big[-y\log(a) - (1-y)\log(1-a)\big] = -\frac{y}{a} + \frac{1-y}{1-a}$$

(B2): $$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial z} = \frac{\partial \mathcal{L}}{\partial a}\sigma'(z) = \frac{\partial \mathcal{L}}{\partial a}\underbrace{\sigma(z)}_{a}(1-\underbrace{\sigma(z)}_{a}) = \frac{\partial \mathcal{L}}{\partial a}a(1-a) = a - y$$

Back propagation for Logistic Regression

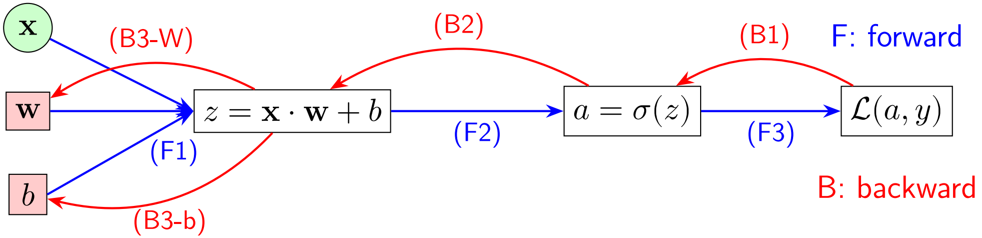


(B1): $$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial}{\partial a}\Big[-y\log(a) - (1-y)\log(1-a)\Big] = -\frac{y}{a} + \frac{1-y}{1-a}$$

(B2): $$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial z} = \frac{\partial \mathcal{L}}{\partial a}\sigma'(z) = \frac{\partial \mathcal{L}}{\partial a}\underbrace{\sigma(z)}_{a}(1-\underbrace{\sigma(z)}_{a}) = \frac{\partial \mathcal{L}}{\partial a}a(1-a) = a - y$$

(B3-W): $$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial b} = \frac{\partial \mathcal{L}}{\partial z} = a - y$$

Back propagation for Logistic Regression



(B1): $$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial}{\partial a}\left[-y\log(a) - (1-y)\log(1-a)\right] = -\frac{y}{a} + \frac{1-y}{1-a}$$

(B2): $$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial z} = \frac{\partial \mathcal{L}}{\partial a}\sigma'(z) = \frac{\partial \mathcal{L}}{\partial a}\underbrace{\sigma(z)}_{a}(1-\underbrace{\sigma(z)}_{a}) = \frac{\partial \mathcal{L}}{\partial a}a(1-a) = a - y$$

(B3-W): $$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial b} = \frac{\partial \mathcal{L}}{\partial z} = a - y$$

(B3-b): $$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial z}\mathbf{x} = (a-y)\mathbf{x}$$

Gradient of loss function: Notation

- By introducing the notation for the partial derivative of the loss function

$$\partial\Theta = \frac{\partial\mathcal{L}}{\partial\Theta},$$

we can write the above formulation for the back propagation as follows

Gradient of loss function: Notation

- By introducing the notation for the partial derivative of the loss function

$$\partial\Theta = \frac{\partial\mathcal{L}}{\partial\Theta},$$

we can write the above formulation for the back propagation as follows

$$\text{(B1)}: \quad \mathrm{d}a = -\frac{y}{a} + \frac{1-y}{1-a}$$

Gradient of loss function: Notation

- By introducing the notation for the partial derivative of the loss function

$$\partial\Theta = \frac{\partial\mathcal{L}}{\partial\Theta},$$

we can write the above formulation for the back propagation as follows

$$\text{(B1)}: \quad \mathrm{d}a = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$\text{(B2)}: \quad \mathrm{d}z = \mathrm{d}a\,\frac{\partial a}{\partial z} = \mathrm{d}a\,\sigma'(z) = \mathrm{d}a \cdot a(1-a) = a - y$$

Gradient of loss function: Notation

- By introducing the notation for the partial derivative of the loss function

$$\partial\Theta = \frac{\partial\mathcal{L}}{\partial\Theta},$$

  we can write the above formulation for the back propagation as follows

$$\text{(B1)}: \quad \mathrm{d}a = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$\text{(B2)}: \quad \mathrm{d}z = \mathrm{d}a\,\frac{\partial a}{\partial z} = \mathrm{d}a\,\sigma'(z) = \mathrm{d}a \cdot a(1-a) = a - y$$

$$\text{(B3-W)}: \quad \mathrm{d}\mathbf{w} = \mathrm{d}z\,\frac{\partial z}{\partial \mathbf{w}} = \mathrm{d}z\,\mathbf{x} = (a-y)\mathbf{x}$$

$$\text{(B3-b)}: \quad \mathrm{d}\mathbf{b} = \mathrm{d}z\,\frac{\partial z}{\partial \mathbf{b}} = \mathrm{d}z = a - y$$

- It turns out that we only focus on computing *easy derivatives* in the back propagation:

$$\frac{\partial a}{\partial z}, \quad \frac{\partial z}{\partial \mathbf{w}}, \quad \frac{\partial z}{\partial b}$$

Reasoning for back propagation

- Let us consider one neural network with single hidden layer

$$\mathbf{z}^{[1]} = \mathbf{a}^{[2]}\mathbf{W}^{[1]T} + \mathbf{b}^{[1]}$$
$$\mathbf{a}^{[1]} = g(\mathbf{z}^{[1]})$$
$$\mathbf{z}^{[2]} = \mathbf{a}^{[1]}\mathbf{W}^{[2]T} + \mathbf{b}^{[2]}$$
$$\mathbf{a}^{[2]} = g^{\mathbf{z}^{[2]}}$$

- We have the following series of mappings

$$\mathbf{a}^{[0]} \mapsto \text{Linear Mapping} \mapsto \mathbf{z}^{[1]} \mapsto \text{Activation Function} \mapsto \mathbf{a}^{[1]}$$
$$\mapsto \text{Linear Mapping} \mapsto \mathbf{z}^{[2]} \mapsto \text{Activation Function} \mapsto \mathbf{a}^{[2]} \mapsto \mathcal{L}(\mathbf{a}^{[2]}; y)$$

- Our goal: ompute the gradient of the loss function w.r.t. the learnable/model paremeters

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1]}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1]}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[2]}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[2]}}$$

Computation of gradient of loss function

- Let us start with

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}} = \sum_k \frac{\partial \mathcal{L}}{\partial z_k^{[1]}} \frac{\partial z_k^{[1]}}{\partial \mathbf{W}^{[1]}}$$

  The true formulation is more complex due to:

  - $\mathbf{W}^{[1]}$ is a matrix, so the derivative is done w.r.t. each component $W_{ij}^{[1]}$
  - $\mathbf{z}^{[1]}$ is a vector, so the derivative is done w.r.t. each component $z_k^{[1]}$
  - The chain rule in differentiation is actually carried out in the above formula.

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{[1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}} = \sum_k \frac{\partial \mathcal{L}}{\partial z_k^{[1]}} \frac{\partial z_k^{[1]}}{\partial W_{ij}^{[1]}}$$

  for all i and j running in the set of parameters in $\mathbf{W}^{[1]}$

- We shall focus on the "reduced" writing notation

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1]}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}}}_{\text{NOT EASY}} \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}}}_{\text{easy}}$$

Computation of gradient of loss function

- We shall try to compute the derivatives $\partial\mathcal{L}/\partial\mathbf{W}^{[1]}$ by chain rule

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}^{[1]}} = \underbrace{\frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[1]}}}_{\text{NOT EASY}} \underbrace{\frac{\partial\mathbf{z}^{[1]}}{\partial\mathbf{W}^{[1]}}}_{\text{easy}} \qquad \frac{\partial\mathcal{L}}{\partial\mathbf{a}^{[1]}} = \underbrace{\frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[2]}}}_{\text{NOT EASY}} \underbrace{\frac{\partial\mathbf{z}^{[2]}}{\partial\mathbf{a}^{[1]}}}_{\text{easy}}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[1]}} = \underbrace{\frac{\partial\mathcal{L}}{\partial\mathbf{a}^{[1]}}}_{\text{NOT EASY}} \underbrace{\frac{\partial\mathbf{a}^{[1]}}{\partial\mathbf{z}^{[1]}}}_{\text{easy}} \qquad \frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[2]}} = \underbrace{\frac{\partial\mathcal{L}}{\partial\mathbf{a}^{[2]}}}_{\text{FINALLY EASY}} \underbrace{\frac{\partial\mathbf{a}^{[2]}}{\partial\mathbf{z}^{[2]}}}_{\text{easy}}$$

- In summary, we compute the derivative in the order

$$\frac{\partial\mathcal{L}}{\partial\mathbf{a}^{[2]}} \longrightarrow \frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial\mathbf{a}^{[2]}}{\partial\mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial\mathcal{L}}{\partial\mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial\mathbf{z}^{[2]}}{\partial\mathbf{a}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial\mathbf{a}^{[1]}}{\partial\mathbf{z}^{[1]}}}_{\text{easy}}$$

$$\to \frac{\partial\mathcal{L}}{\partial\mathbf{W}^{[1]}} \text{ using } \underbrace{\frac{\partial\mathbf{z}^{[1]}}{\partial\mathbf{W}^{[1]}}}_{\text{easy}}$$

Computation of gradient of loss function

- Similarly, replacing $\dfrac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}}$ with $\dfrac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}}$ $\partial \mathcal{L}/\partial \mathbf{b}^{[1]}$, we arrive at

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1]}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}}}_{\text{NOT EASY}} \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}}}_{\text{easy}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}}}_{\text{NOT EASY}} \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}}}_{\text{NOT EASY}} \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}}}_{\text{FINALLY EASY}} \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}}$$

- In summary, we compute the derivative in the order

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}}$$

$$\longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}}}_{\text{easy}}$$

Computation of gradient of loss function

- Let us now compute the derivative $\partial\mathcal{L}/\partial\mathbf{W}^{[2]}$ by chain rule

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}^{[2]}} = \underbrace{\frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[2]}}}_{\textbf{NOT EASY}} \underbrace{\frac{\partial\mathbf{z}^{[2]}}{\partial\mathbf{W}^{[2]}}}_{\text{easy}}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[2]}} = \underbrace{\frac{\partial\mathcal{L}}{\partial\mathbf{a}^{[2]}}}_{\textbf{ALSO EASY}} \underbrace{\frac{\partial\mathbf{a}^{[2]}}{\partial\mathbf{z}^{[2]}}}_{\text{easy}}$$

- In summary, we compute the derive in the order

$$\frac{\partial\mathcal{L}}{\partial\mathbf{a}^{[2]}} \longrightarrow \frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial\mathbf{a}^{[2]}}{\partial\mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial\mathcal{L}}{\partial\mathbf{W}^{[2]}} \text{ using } \underbrace{\frac{\partial\mathbf{z}^{[2]}}{\partial\mathbf{W}^{[2]}}}_{\text{easy}}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{a}^{[2]}} \longrightarrow \frac{\partial\mathcal{L}}{\partial\mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial\mathbf{a}^{[2]}}{\partial\mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial\mathcal{L}}{\partial\mathbf{b}^{[2]}} \text{ using } \underbrace{\frac{\partial\mathbf{z}^{[2]}}{\partial\mathbf{b}^{[2]}}}_{\text{easy}}$$

Computation of gradient of loss function

Back propagation process

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}}}_{\text{easy}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}}}_{\text{easy}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{W}^{[2]}}}_{\text{easy}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{b}^{[2]}}}_{\text{easy}}$$

Back propagation process

- Repeat the formulation above

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \Theta^{[1]}}}_{\text{easy}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \Theta^{[2]}}}_{\text{easy}}$$

where Θ can play the role of $\mathbf{W}$ or $\mathbf{b}$

- To compute the derivative $\frac{\partial \mathcal{L}}{\Theta^{[1]}}$, we need to go through the exactly same process for computing $\frac{\partial \mathcal{L}}{\Theta^{[2]}}$ but then continue one layer more.

Back propagation process

- **Repeat the formulation above**

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \Theta^{[1]}}}_{\text{easy}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \Theta^{[2]}}}_{\text{easy}}$$

- **Backpropagation process**: Let us combinet two equations in one as follows

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \Theta^{[1]}}}_{\text{easy}}$$

$$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$$

$$\frac{\partial \mathcal{L}}{\partial \Theta^{[2]}} \text{ using } \frac{\partial \mathbf{z}^{[2]}}{\partial \Theta^{[2]}}$$

Backpropagation process

- To compute the derivative $\dfrac{\partial \mathcal{L}}{\partial \Theta^{[1]}}$, we need to go through the exactly same process for computing $\dfrac{\partial \mathcal{L}}{\partial \Theta^{[2]}}$ but then continue one layer more.
- What to expect?
  To compute the derivative $\dfrac{\partial \mathcal{L}}{\partial \Theta^{[l-1]}}$, we need to go through the exactly same process for computing $\dfrac{\partial \mathcal{L}}{\partial \Theta^{[l]}}$ but then continue one layer more.

Summary of gradient descent

- For one training example:

$$\frac{\partial \mathcal{L}}{\partial z^{[2]}} = a^{[2]} - y$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[2]}} = \left(\frac{\partial \mathcal{L}}{\partial z^{[2]}}\right)^T \mathbf{a}^{[1]}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[2]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} = \frac{\partial \mathcal{L}}{\partial z^{[2]}} \mathbf{W}^{[2]} \odot \sigma'(\mathbf{a}^{[1]})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1]}} = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}}\right)^T \mathbf{a}^{[0]}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}}$$

- For one training mini-batch (m examples):

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{[2]}} = \mathbf{A}^{[2]} - \mathbf{Y},$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[2]}} = \frac{1}{m}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{[2]}}\right)^T \mathbf{A}^{[1]}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[2]}} = \frac{1}{m}\texttt{np.sum}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{[2]}}, \texttt{axis=0}\right)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{[1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{[2]}} \mathbf{W}^{[2]} \odot \sigma'(\mathbf{A}^{[1]})$$

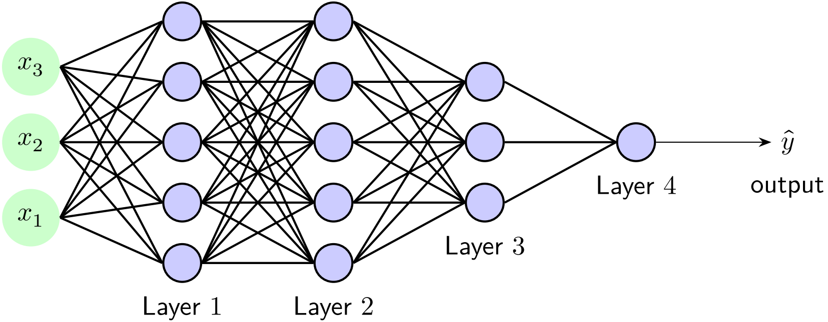
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1]}} = \frac{1}{m}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{[1]}}\right)^T \mathbf{A}^{[0]}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1]}} = \frac{1}{m}\texttt{np.sum}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{[1]}}, \texttt{axis=0}\right)$$

Table of Contents

1. Backpropagation
   - Computation Graph
   - Back propagation for logistic regression
2. Deep neural network with more than one hidden layer
   - Forward propagation
   - Back propagation
3. Implementation in PyTorch
   - Building blocks of a deep neural network
   - Build a simple neural network in PyTorch

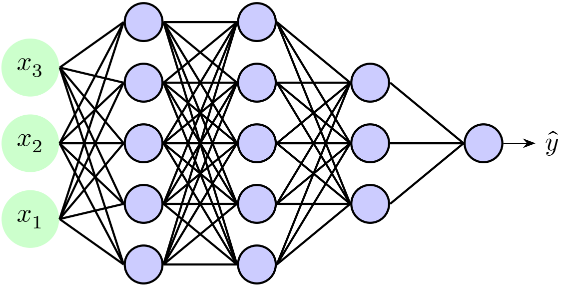
Deep neural network notation



- Number of layers: $L = 4$
- Number of units/neurons in layer l: $n^{[l]}$
- Activations at layer l: $\mathbf{a}^{[l]} = (a_1^{[l]}, \dots, a_{n^{[l]}}^{[l]})$
- Activations at layer l: $\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})$
- Weights and biases for $\mathbf{z}[l]$: $\mathbf{W}^{[l]}$, $\mathbf{b}^{[l]}$
- Activation function for layer l: $g^{[l]}(\cdot)$

In this example, we have: $n^{[0]} = n_{\mathbf{x}} = 3, \quad n^{[1]} = 5, \quad n^{[2]} = 5, \quad n^{[3]} = 3, \quad n^{[4]} = n^{[L]} = 1.$

Forward propagation in a deep neural network



For one training example:

$$\mathbf{z}^{[1]} = \mathbf{a}^{[0]}\mathbf{W}^{[1]T} + \mathbf{b}^{[1]}$$

$$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$$

$$\cdots = \cdots$$

$$\mathbf{z}^{[4]} = \mathbf{a}^{[3]}\mathbf{W}^{[4]T} + \mathbf{b}^{[4]}$$

$$\mathbf{a}^{[4]} = g^{[4]}(\mathbf{z}^{[4]})$$

Generalization:

$$\mathbf{z}^{[l]} = \mathbf{a}^{[l]}(\mathbf{W}^{[l]})^T + \mathbf{b}^{[l]}$$

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})$$

Vectorization:

$$\mathbf{Z}^{[l]} = \mathbf{A}^{[l]}(\mathbf{W}^{[l]})^T + \mathbf{b}^{[l]}$$

$$\mathbf{A}^{[l]} = g^{[l]}(\mathbf{Z}^{[l]})$$

Backpropagaton for L-layer network

Let us repeat the backpropagation for a shallow neural network

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \Theta^{[1]}}}_{\text{easy}}$$

$$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$$

$$\frac{\partial \mathcal{L}}{\partial \Theta^{[2]}} \text{ using } \frac{\partial \mathbf{z}^{[2]}}{\partial \Theta^{[2]}}$$

Backpropagaton for L-layer network

By computing carefully for 4-layer network, we arrive at

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[4]}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[4]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[4]}}{\partial \mathbf{z}^{[4]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[4]}}{\partial \mathbf{a}^{[3]}}}_{\text{easy}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[4]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[4]}}{\partial \Theta^{[4]}}}_{\text{easy}} \Longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[4]}}$$

$$\longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{z}^{[3]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{a}^{[2]}}}_{\text{easy}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[3]}}{\partial \Theta^{[3]}}}_{\text{easy}} \Longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[3]}}$$

$$\longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \Theta^{[2]}}}_{\text{easy}} \Longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[2]}}$$

$$\longrightarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \Theta^{[1]}}}_{\text{easy}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \Theta^{[1]}}}_{\text{easy}} \Longrightarrow \frac{\partial \mathcal{L}}{\partial \Theta^{[1]}}$$

Backpropagaton for L-layer network

Introducing the notation dvariable $= \frac{\partial \mathcal{L}}{\partial \text{variable}}$

$$\mathrm{d}\mathbf{a}^{[4]} \longrightarrow \mathrm{d}\mathbf{z}^{[4]} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[4]}}{\partial \mathbf{z}^{[4]}}}_{\text{easy}} \longrightarrow \mathrm{d}\mathbf{a}^{[3]} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[4]}}{\partial \mathbf{a}^{[3]}}}_{\text{easy}}$$

$$\longrightarrow \mathrm{d}\mathbf{z}^{[3]} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{z}^{[3]}}}_{\text{easy}} \longrightarrow \mathrm{d}\mathbf{a}^{[2]} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{a}^{[2]}}}_{\text{easy}}$$

$$\longrightarrow \mathrm{d}\mathbf{z}^{[2]} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}}_{\text{easy}} \longrightarrow \mathrm{d}\mathbf{a}^{[1]} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{\text{easy}}$$

$$\longrightarrow \mathrm{d}\mathbf{z}^{[1]} \text{ using } \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{\text{easy}} \longrightarrow \mathrm{d}\Theta^{[1]} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \Theta^{[1]}}}_{\text{easy}}$$

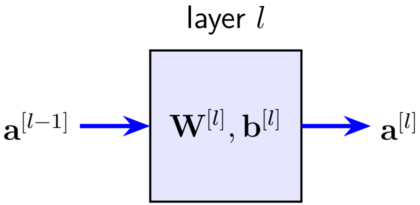
$$\mathrm{d}\mathbf{z}^{[4]} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[4]}}{\partial \Theta^{[4]}}}_{\text{easy}} \Longrightarrow \mathrm{d}\Theta^{[4]}$$

$$\mathrm{d}\mathbf{z}^{[3]} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[3]}}{\partial \Theta^{[3]}}}_{\text{easy}} \Longrightarrow \mathrm{d}\Theta^{[3]}$$

$$\mathrm{d}\mathbf{z}^{[2]} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \Theta^{[2]}}}_{\text{easy}} \Longrightarrow \mathrm{d}\Theta^{[2]}$$

$$\mathrm{d}\mathbf{z}^{[1]} \text{ using } \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial \Theta^{[1]}}}_{\text{easy}} \Longrightarrow \mathrm{d}\Theta^{[1]}$$

Forward versus backward propagation



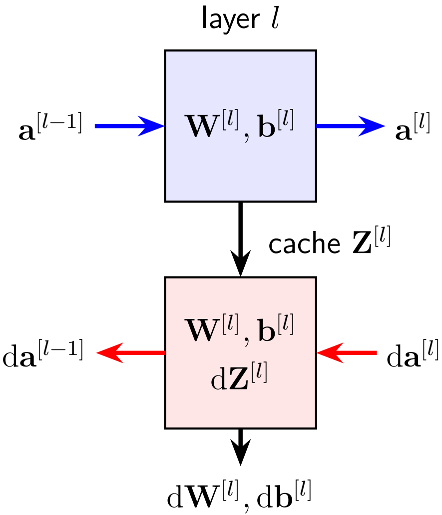
Forward Input $\mathbf{a}^{[l-1]}$, output $\mathbf{a}^{[l]}$ ($\mathbf{W}^{[l]}, \mathbf{b}^{[l]}$ given)

Cache: $\mathbf{Z}^{[l]}$

$$\mathbf{Z}^{[l]} = \mathbf{a}^{[l-1]}(\mathbf{W}^{[l]})^T + \mathbf{b}^{[l-1]}$$

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})$$

Forward versus backward propagation



Forward Input $\mathbf{a}^{[l-1]}$, output $\mathbf{a}^{[l]}$ ($\mathbf{W}^{[l]}, \mathbf{b}^{[l]}$ given)

Cache: $\mathbf{Z}^{[l]}$

$$\mathbf{Z}^{[l]} = \mathbf{a}^{[l-1]}(\mathbf{W}^{[l]})^T + \mathbf{b}^{[l-1]}$$

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})$$

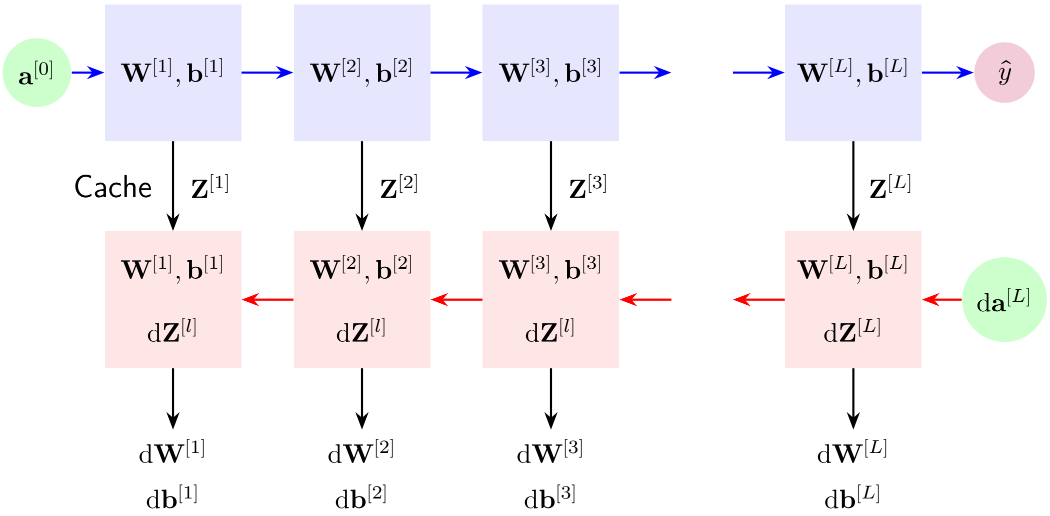
Backward Input $\mathrm{d}\mathbf{a}^{[l]}$, output $\mathrm{d}\mathbf{a}^{[l-1]}, \mathrm{d}\mathbf{W}^{[l]}, \mathrm{d}\mathbf{b}^{[l]}$

(using cache $\mathbf{Z}^{[l]}$)

Logics behind backward propagation

See the hand-written note

Forward and backward propagation



Forward propagation for layer l

- Input $\mathbf{a}^{[l-1]}$
- Output $\mathbf{a}^{[l]}$, cache $\mathbf{z}^{[l]}$

For one training example:

$$\mathbf{z}^{[l]} = \mathbf{a}^{[l-1]}(\mathbf{W}^{[l]})^T + \mathbf{b}^{[l]}$$

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})$$

For a mini-batch of examples:

$$\mathbf{Z}^{[l]} = \mathbf{A}^{[l-1]}(\mathbf{W}^{[l]})^T + \mathbf{b}^{[l]}$$

$$\mathbf{A}^{[l]} = g^{[l]}(\mathbf{Z}^{[l]})$$

Backward propagation for layer l

- Input $\mathrm{d}\mathbf{a}^{[l]}$
- Output $\mathrm{d}\mathbf{a}^{[l-1]}, \mathrm{d}\mathbf{W}^{[l]}, \mathrm{d}\mathbf{b}^{[l]}$

For one training example:

$$\mathrm{d}\mathbf{z}^{[l]} = \mathrm{d}\mathbf{a}^{[l]} \odot g^{[l]\prime}(\mathbf{z}^{[l]})$$

$$\mathrm{d}\mathbf{W}^{[l]} = (\mathrm{d}\mathbf{Z}^{[l]})^T \mathrm{d}\mathbf{a}^{[l-1]}$$

$$\mathrm{d}\mathbf{b}^{[l]} = \mathrm{d}\mathbf{z}^{[l]}$$

$$\mathrm{d}\mathbf{a}^{[l-1]} = \mathrm{d}\mathbf{z}^{[l]} \mathbf{W}^{[l]}$$

For a mini-batch of examples:

$$\mathrm{d}\mathbf{Z}^{[l]} = \mathrm{d}\mathbf{A}^{[l]} \odot g^{[l]\prime}(\mathbf{Z}^{[l]})$$

$$\mathrm{d}\mathbf{W}^{[l]} = \frac{1}{m}(\mathrm{d}\mathbf{Z}^{[l]})^T \mathrm{d}\mathbf{A}^{[l-1]}$$

$$\mathrm{d}\mathbf{b}^{[l]} = \frac{1}{m}\texttt{np.sum}\left(\mathrm{d}\mathbf{Z}^{[l]}, \texttt{axis=0}\right)$$

$$\mathrm{d}\mathbf{A}^{[l-1]} = \mathrm{d}\mathbf{Z}^{[l]} \mathbf{W}^{[l]}$$

Table of Contents

1. Backpropagation
   - Computation Graph
   - Back propagation for logistic regression
2. Deep neural network with more than one hidden layer
   - Forward propagation
   - Back propagation
3. Implementation in PyTorch
   - Building blocks of a deep neural network
   - Build a simple neural network in PyTorch

A module in PyTorch

- Base class for all neural network modules.
- Your models should also subclass this class.
- Modules can also contain other Modules, allowing to nest them in a tree structure. You can assign the submodules as regular attributes:

A module in PyTorch

Module

CLASS torch.nn.Module(**args*, ***kwargs*) [SOURCE]

Base class for all neural network modules.

Your models should also subclass this class.

Modules can also contain other Modules, allowing to nest them in a tree structure. You can assign the submodules as regular attributes:

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self) -> None:
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

`nn.Module` Click on Me!

Implementation in PyTorch: `nn.Linear`

Linear

CLASS torch.nn.Linear(*in_features*, *out_features*, *bias=True*, *device=None*, *dtype=None*) [SOURCE]

Applies an affine linear transformation to the incoming data: $y = xA^T + b$.

This module supports TensorFloat32.

On certain ROCm devices, when using float16 inputs this module will use different precision for backward.

Parameters

- **in_features** (*int*) – size of each input sample
- **out_features** (*int*) – size of each output sample
- **bias** (*bool*) – If set to `False`, the layer will not learn an additive bias. Default: `True`

Shape:

- Input: $(*, H_{in})$ where $*$ means any number of dimensions including none and $H_{in} = \text{in\_features}$.
- Output: $(*, H_{out})$ where all but the last dimension are the same shape as the input and $H_{out} = \text{out\_features}$.

`nn.Linear` Click on Me!

Sigmoid

CLASS torch.nn.Sigmoid(*args, **kwargs) [SOURCE]

Applies the Sigmoid function element-wise.

$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1 + \exp(-x)}$$

Shape:

- Input: $(*)$, where $*$ means any number of dimensions.
- Output: $(*)$, same shape as the input.

`nn.Sigmoid` Click on Me!

Implementation in PyTorch: `nn.Tanh`

Tanh

CLASS torch.nn.Tanh(*args, **kwargs) [SOURCE]

Applies the Hyperbolic Tangent (Tanh) function element-wise.

Tanh is defined as:

$$\text{Tanh}(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Shape:

- Input: $(*)$, where $*$ means any number of dimensions.
- Output: $(*)$, same shape as the input.

`nn.Tanh` Click on Me!

Implementation in PyTorch: `nn.ReLU`

ReLU

CLASS torch.nn.ReLU(*inplace=False*) [SOURCE]

Applies the rectified linear unit function element-wise.

$$\mathrm{ReLU}(x) = (x)^+ = \max(0, x)$$

Parameters

inplace (*bool*) – can optionally do the operation in-place. Default: `False`

Shape:

- Input: $(*)$, where $*$ means any number of dimensions.
- Output: $(*)$, same shape as the input.

`nn.ReLU` Click on Me!

Implementation in PyTorch: `nn.functional.sigmoid`

torch.nn.functional.sigmoid

torch.nn.functional.sigmoid(*input*) → Tensor [SOURCE]

Applies the element-wise function $\text{Sigmoid}(x) = \frac{1}{1+\exp(-x)}$

See `Sigmoid` for more details.

`nn.Sigmoid` Click on Me!

Implementation in PyTorch: `nn.functional.tanh`

torch.nn.functional.tanh

torch.nn.functional.tanh(*input*) → Tensor [SOURCE]

Applies element-wise, $\mathrm{Tanh}(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$

See Tanh for more details.

`nn.Sigmoid` Click on Me!

torch.nn.functional.relu

`torch.nn.functional.relu(input, inplace=False) → Tensor` [SOURCE]

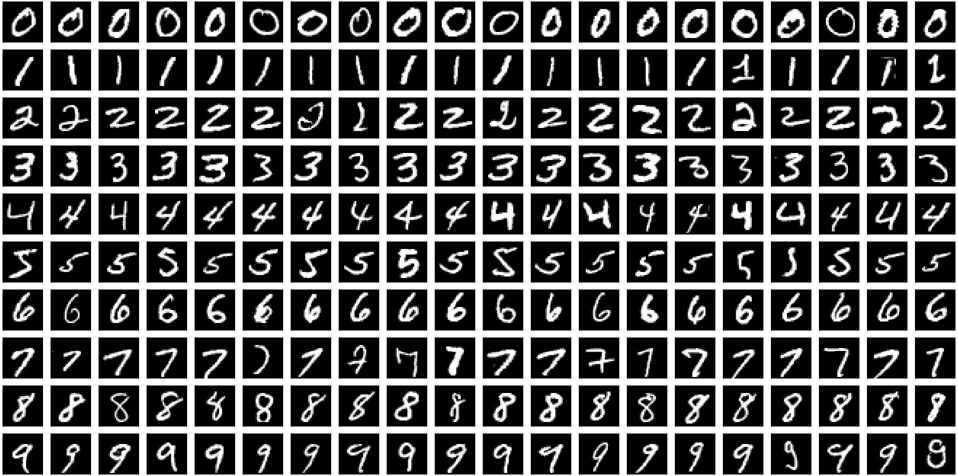
Applies the rectified linear unit function element-wise. See `ReLU` for more details.

Return type

Tensor

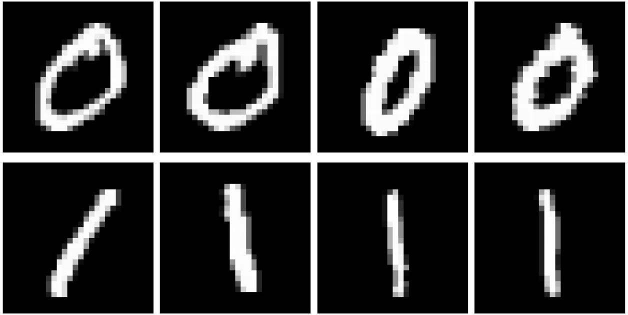
`nn.functional.relu` Click on Me!

A neural network for classification of hand-written digits



MNIST database (Modified National Institute of Standards and Technology database)

A neural network for classification of hand-written digits



Each image is gray-scale image of size $(28, 28)$.

- $\mathbf{x}^{(i)}$ is an image of size $(28, 28)$
- Flatten $\mathbf{x}^{(i)}$ into a vector of dimension 784 $(= 24 \cdot 24)$

A neural network for classification of hand-written digits

We want to build a neural network with the following architecture

$$\text{Linear}[784, 512] \longrightarrow \text{ReLU} \longrightarrow \text{Linear}[512, 512] \longrightarrow \text{ReLU} \longrightarrow \text{Linear}[512, 10]$$

Questions

A neural network for classification of hand-written digits

We want to build a neural network with the following architecture

$$\color{red}{\text{Linear}}[784, 512] \longrightarrow \color{blue}{\text{ReLU}} \longrightarrow \color{red}{\text{Linear}}[512, 512] \longrightarrow \color{blue}{\text{ReLU}} \longrightarrow \color{red}{\text{Linear}}[512, 10]$$

Questions

- Which component contains learnable parameters?

A neural network for classification of hand-written digits

We want to build a neural network with the following architecture

$$\text{Linear}[784, 512] \longrightarrow \text{ReLU} \longrightarrow \text{Linear}[512, 512] \longrightarrow \text{ReLU} \longrightarrow \text{Linear}[512, 10]$$

Questions

- Which component contains learnable parameters?
- How many learnable parameters does the model have? (assume all layers have biases)

A neural network for classification of hand-written digits

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

model = NeuralNetwork().to(device)      # device = 'cpu' or device = 'cuda'
```