The Backpropagation algorithm

Thomas Gerald

March 14, 2025

Laboratoire Interdisciplinaire des Sciences du Numérique – LISN, CNRS

The lecture

Content of the lecture:

1. Neural Network: Forward Pass

   This section is a reminder of the previous course. We will unroll neural networks from input space to the prediction or logits, computing the functions is called the froward pass. We will also define the calculus graph use in deep-learning algorithm

2. Reminder of training algorithm

   In this part we will present the gradient descent algorithm

3. The backpropagation algorithm

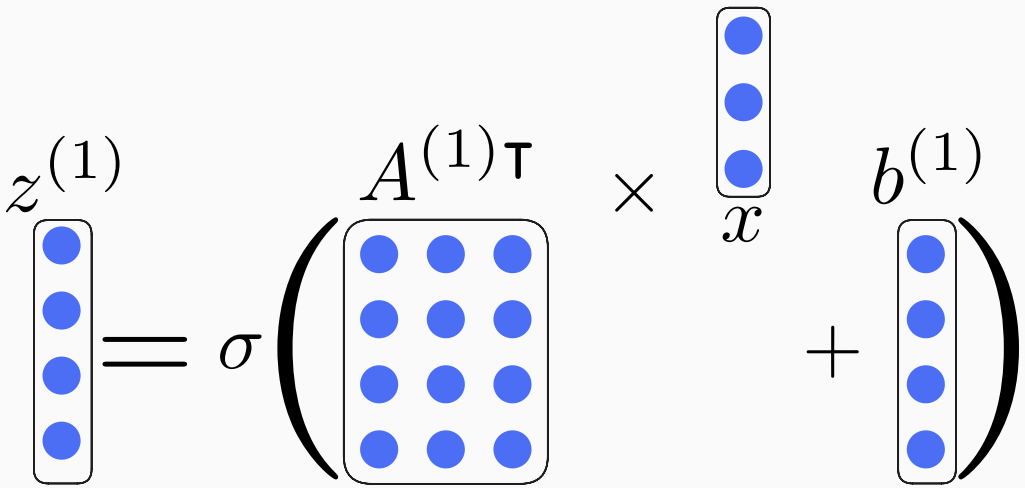
   In the section we will explain how the gradient can be processed. Thanks to the calculus graph we can visit this graph to process the gradient by part. We will present both mathematical approach and language oriented approach.

4. Optimisation approaches (**reported to next week**)

   This last part is dedicated to the optimisation of non-convex function. We will present the different gradient-descent alternatives.

Neural Network: Forward Pass

Notations



Notation reminder

- x: input features
- $z^{(i)}$: hidden representation
- o: output logits
- $\Theta = \{A^{(1)}, b^{(1)}, A^{(2)}, b^{(2)}, \ldots, A^{(L)}, b^{(L)}\}$: parameters
- $\sigma^{(i)}$: non-linear activation

Muti-layer perceptron and Feed-forward Neural Network (1/2)

Feed-forward Neural Network

We show that a neural networks is a complex parametrized function f_Θ.

$$\begin{aligned} f_\Theta : \mathcal{X} &\to \mathcal{Y} \\ x &\mapsto (g_{\Theta_L} \circ \dots g_{\Theta_2} \circ g_{\Theta_1})(x) \end{aligned}$$

Those architectures are also called feed-forward Neural Network

MLP (Multi-Layer Perceptron):

$\to$ The i^{th} **hidden representation** is the output of $(g_{\Theta_i} \circ g_{\Theta_{i-1}} \circ \dots \circ g_{\Theta_1})(x)$
$\to$ The function g_{Θ_i} is defined by:

$$\begin{aligned} g_{\Theta_i} : \mathbb{R}^{n_{i-1}} &\to \mathbb{R}^{n_i} \\ z^{(i)} &\mapsto \sigma(A^{(i)\intercal} z^{(i-1)} + b^{(i)}) \end{aligned}$$

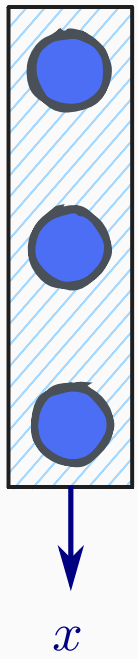
Muti-layer perceptron and Feed-forward Neural Network (2/2)

Lets unroll a 3 hidden layer perceptron

Muti-layer perceptron and Feed-forward Neural Network (2/2)

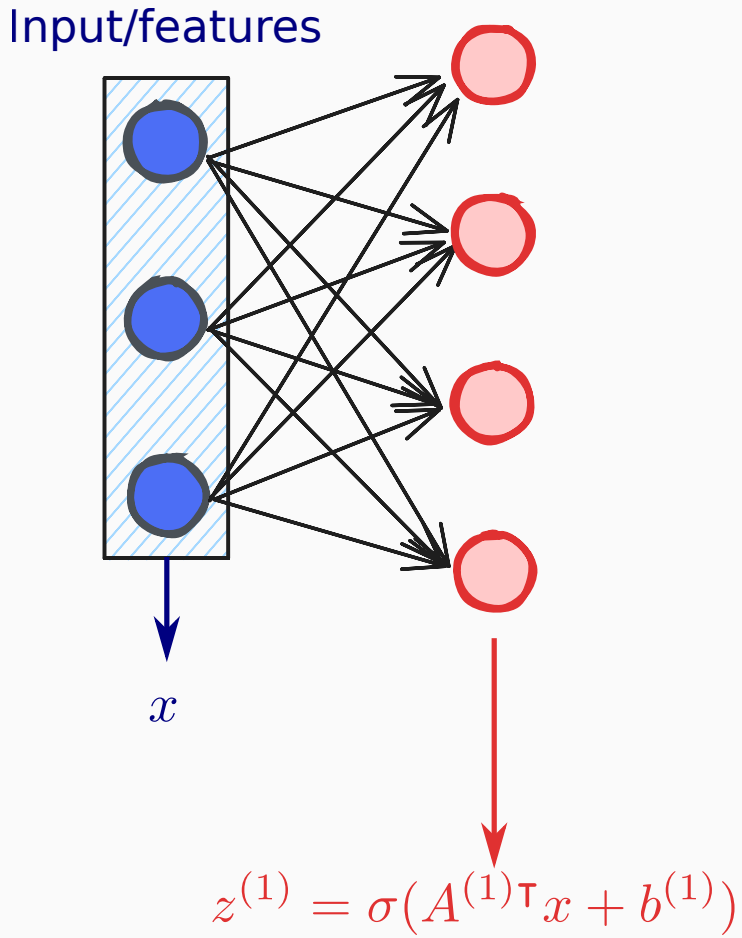
Lets unroll a 3 hidden layer perceptron

Input/features



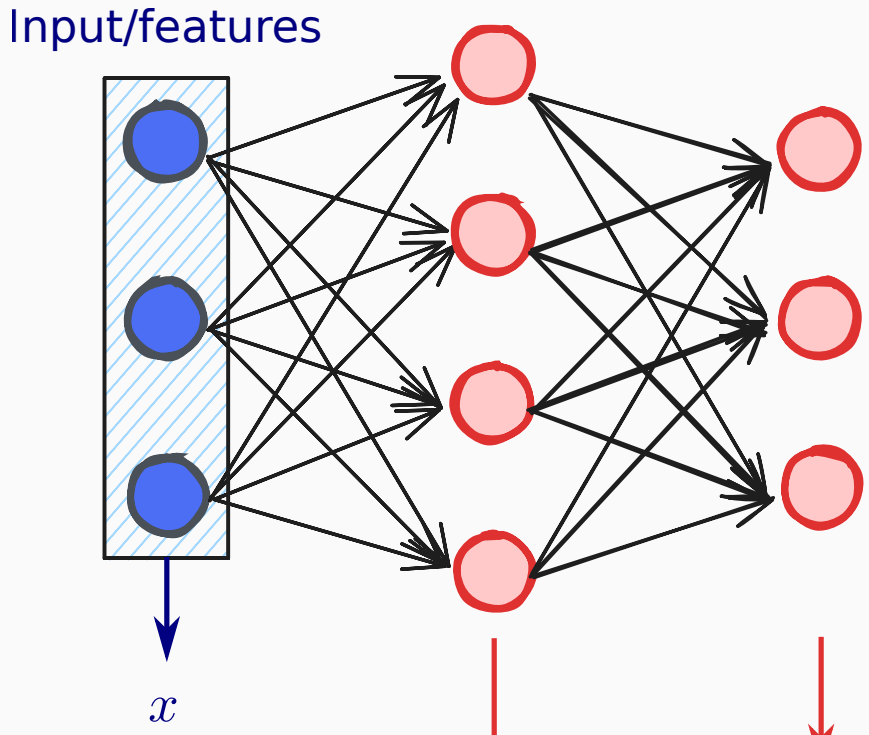
Muti-layer perceptron and Feed-forward Neural Network (2/2)

Lets unroll a 3 hidden layer perceptron

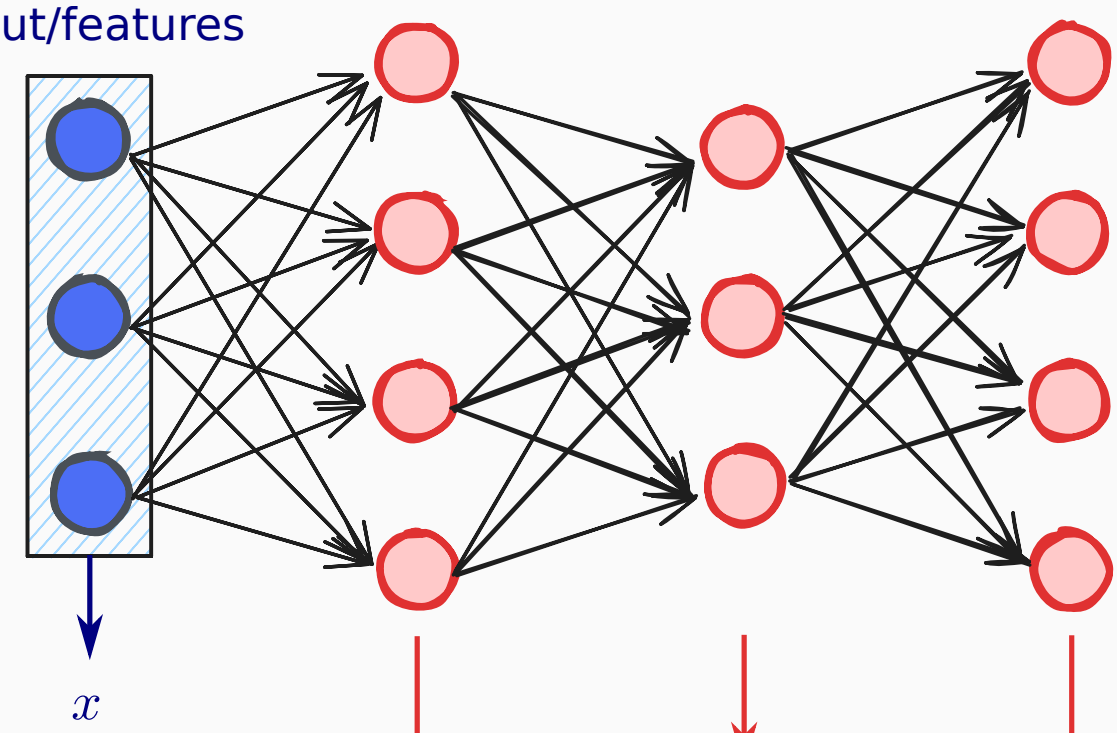


Muti-layer perceptron and Feed-forward Neural Network (2/2)

Lets unroll a 3 hidden layer perceptron



Lets unroll a 3 hidden layer perceptron



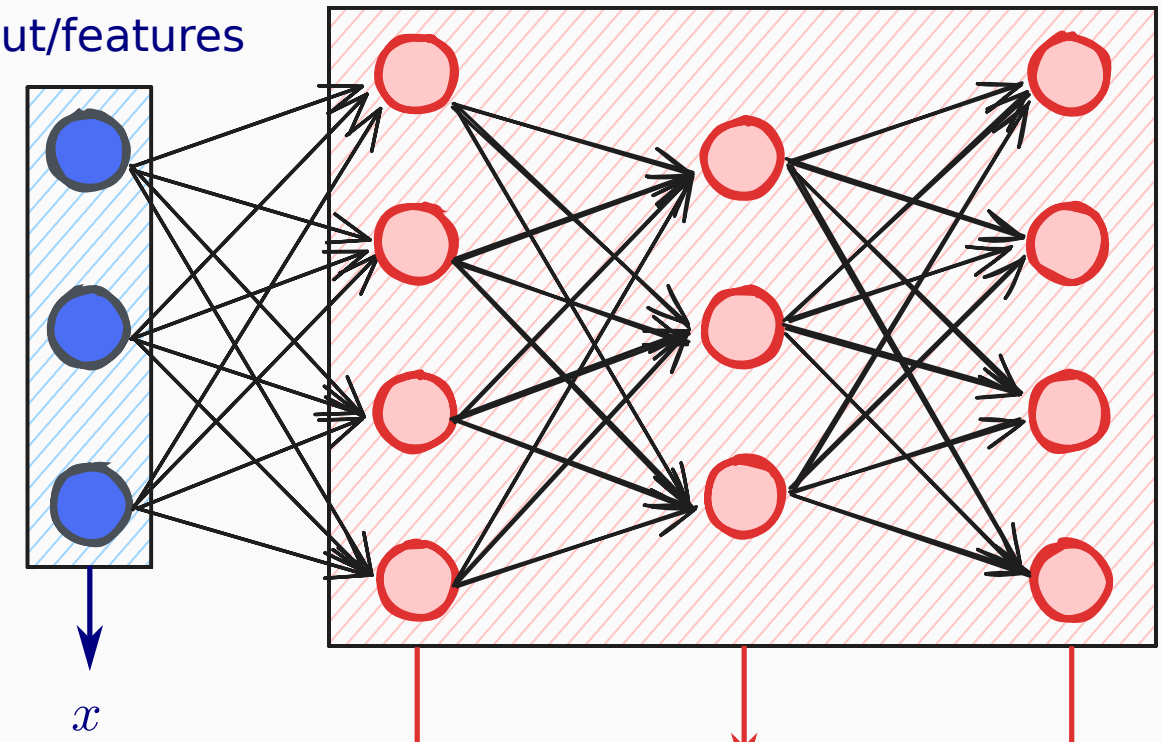
$$z^{(1)} = \sigma(A^{(1)\intercal} x + b^{(1)})$$

$$z^{(2)} = \sigma(A^{(2)\intercal} z^{(1)} + b^{(2)})$$

$$z^{(3)} = \sigma(A^{(3)\intercal} z^{(2)} + b^{(3)})$$

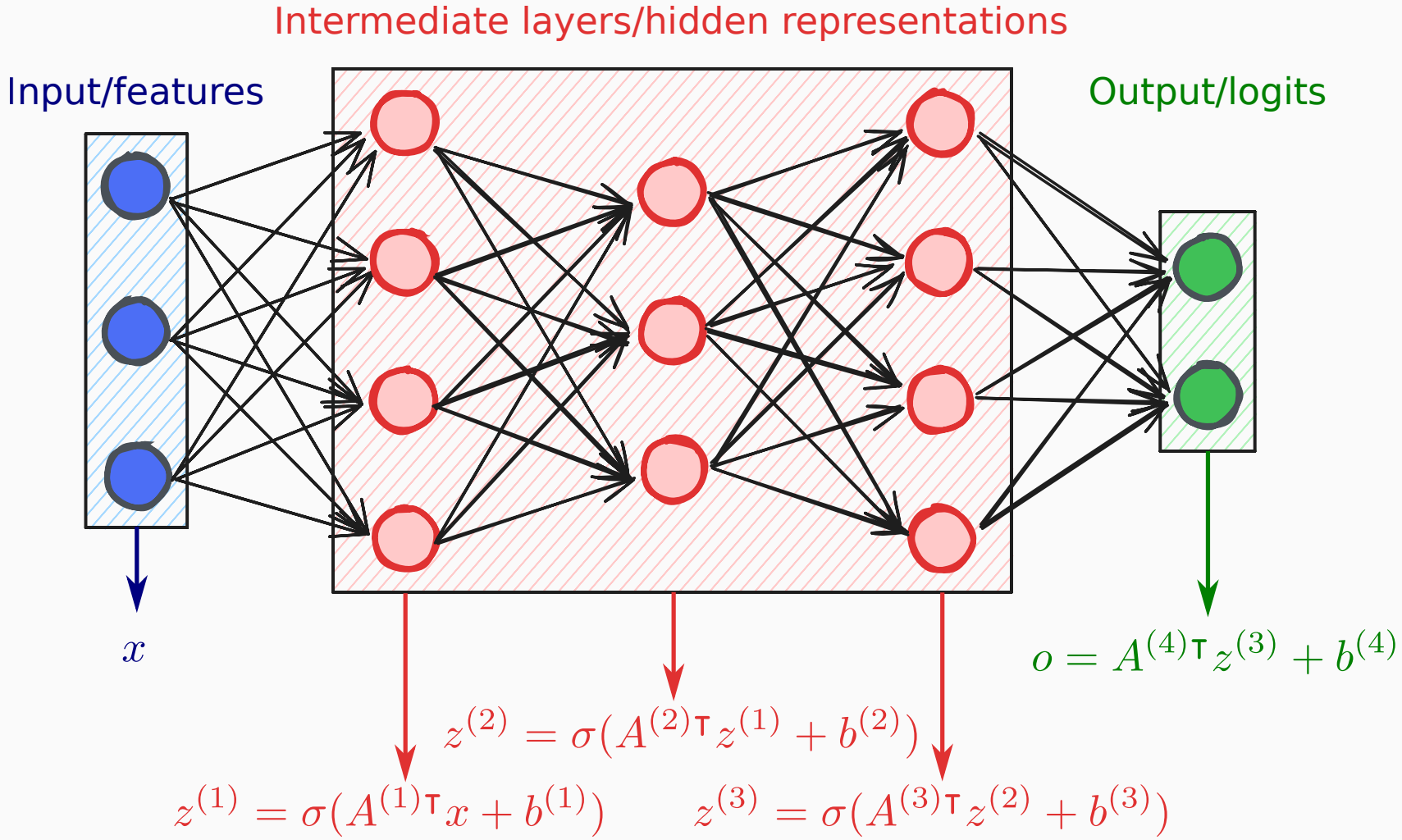
Lets unroll a 3 hidden layer perceptron

Intermediate layers/hidden representations



Muti-layer perceptron and Feed-forward Neural Network (2/2)

Lets unroll a 3 hidden layer perceptron



Feed-forward Neural Network: blocks and modules

We can decompose the calculus of the output into several **blocks**

Feed-forward Neural Network: blocks and modules

We can decompose the calculus of the output into several **blocks**

$\rightarrow$ In programmation we call it a module (pytorch as a class `nn.Module`)

Feed-forward Neural Network: blocks and modules

We can decompose the calculus of the output into several **blocks**

$\rightarrow$ In programmation we call it a module (pytorch as a class `nn.Module`)

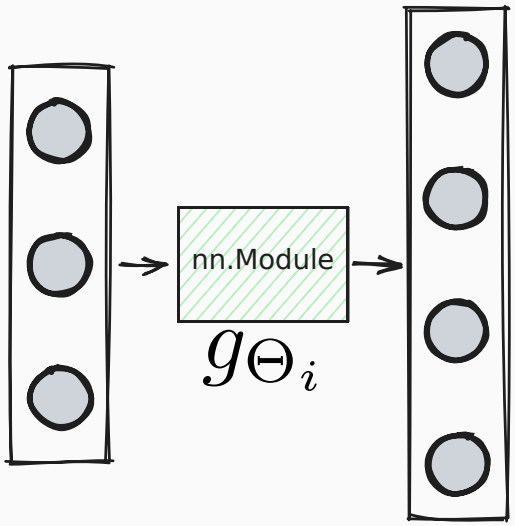
$\rightarrow$ Computing sequentially the output is called the forward pass (each module implement a forward method: `my_module.forward(x)`)

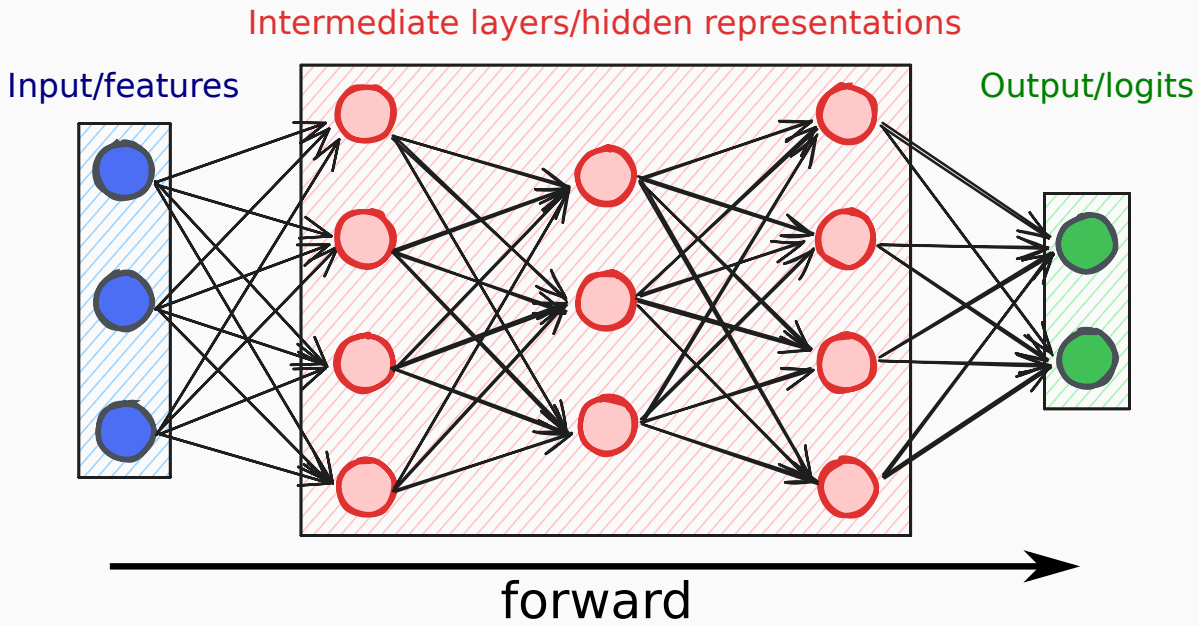
Feed-forward Neural Network: blocks and modules

We can decompose the calculus of the output into several **blocks**

$\rightarrow$ In programmation we call it a module (pytorch as a class `nn.Module`)

$\rightarrow$ Computing sequentially the output is called the forward pass (each module implement a forward method: `my_module.forward(x)`)





Calculus graph

A Feed forward algorithm can seen as a graph where node is function:

- A node can be the function g_{Θ_i}
- g_{Θ_i} can be decomposed into two function $g_{\Theta_i} = \sigma \circ l_{\Theta_i}$
  - A node for l_{Θ_i} (linear function)

$$l_{\Theta_i} : \mathbb{R}^{n_i - 1} \to \mathbb{R}^{n_i}$$
$$x \mapsto A^{(i)\intercal} x + b^{(i)}$$

  - A node for σ (activation function) ⚠ Applied piecewise

$$\sigma : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$$
$$x \mapsto \sigma(x)$$

Gradient Descent

The gradient descent algorithm: Vanilla GD

Stochastic Gradient Descent (GD):

Update the weight according to the gradient of all examples

The gradient descent algorithm: Vanilla GD

Stochastic Gradient Descent (GD):

Update the weight according to the gradient of all examples

Let f_Θ be the neural network

The gradient descent algorithm: Vanilla GD

Stochastic Gradient Descent (GD):

Update the weight according to the gradient of all examples

Let f_Θ be the neural network

1. Apply f on each data i.e $o^{(i)} = f_\Theta(x^{(i)})$

The gradient descent algorithm: Vanilla GD

Stochastic Gradient Descent (GD):

Update the weight according to the gradient of all examples

Let f_Θ be the neural network

1. Apply f on each data i.e $o^{(i)} = f_\Theta(x^{(i)})$
2. Update weights $\Theta_{t+1} = \Theta_t - \lambda \nabla_\Theta \frac{1}{|\mathcal{D}_{train}|} \sum_{(x^{(i)}, y^{(i)}) \in \mathcal{D}_{train}} \mathcal{L}(o^{(i)}, y^{(i)})$

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

Let f_Θ be the neural network

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

Let f_Θ be the neural network

1. Sample (x, y) from $\mathcal{D}_{train}$

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

Let f_Θ be the neural network

1. Sample (x, y) from $\mathcal{D}_{train}$
2. Apply f on sampled data i.e $o = f_\Theta(x)$

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

Let f_Θ be the neural network

1. Sample (x, y) from $\mathcal{D}_{train}$
2. Apply f on sampled data i.e $o = f_\Theta(x)$
3. Update weights $\Theta_{t+1} = \Theta_t - \lambda \nabla_\Theta \mathcal{L}(o, y)$

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

Let f_Θ be the neural network

1. Sample (x, y) from $\mathcal{D}_{train}$
2. Apply f on sampled data i.e $o = f_\Theta(x)$
3. Update weights $\Theta_{t+1} = \Theta_t - \lambda \nabla_\Theta \mathcal{L}(o, y)$

Algorithm 6 SGD

Require: $f, \Theta, \mathcal{D}, \lambda$
  for `i=1 to E` **do**
    $(x, y) \sim \mathcal{D}$
    $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$
    $\Theta \leftarrow \Theta - \lambda \nabla_\Theta \mathcal{L}(f_\Theta(x), y)$
  end for

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

Let f_Θ be the neural network

1. Sample (x, y) from $\mathcal{D}_{train}$
2. Apply f on sampled data i.e $o = f_\Theta(x)$
3. Update weights $\Theta_{t+1} = \Theta_t - \lambda \nabla_\Theta \mathcal{L}(o, y)$

In practice

- See all the dataset (many times)

Algorithm 7 SGD

Require: $f, \Theta, \mathcal{D}, \lambda$
  for `i=1 to E` **do**
    $(x, y) \sim \mathcal{D}$
    $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$
    $\Theta \leftarrow \Theta - \lambda \nabla_\Theta \mathcal{L}(f_\Theta(x), y)$
  end for

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

Let f_Θ be the neural network

1. Sample (x, y) from $\mathcal{D}_{train}$
2. Apply f on sampled data i.e $o = f_\Theta(x)$
3. Update weights $\Theta_{t+1} = \Theta_t - \lambda \nabla_\Theta \mathcal{L}(o, y)$

In practice

- See all the dataset (many times)
- One turn (all dataset) is called an epoch

Algorithm 8 SGD

Require: $f, \Theta, \mathcal{D}, \lambda$
  for `i=1 to E` **do**
    $(x, y) \sim \mathcal{D}$
    $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$
    $\Theta \leftarrow \Theta - \lambda \nabla_\Theta \mathcal{L}(f_\Theta(x), y)$
  end for

The gradient descent algorithm: SGD

Stochastic Gradient Descent (SGD):

Update the weight according to the gradient of one randomly sampled example

Let f_Θ be the neural network

1. Sample (x, y) from $\mathcal{D}_{train}$
2. Apply f on sampled data i.e $o = f_\Theta(x)$
3. Update weights $\Theta_{t+1} = \Theta_t - \lambda \nabla_\Theta \mathcal{L}(o, y)$

In practice

- See all the dataset (many times)
- One turn (all dataset) is called an epoch
- We stop training when validation accuracy do not increase

Algorithm 9 SGD

Require: $f, \Theta, \mathcal{D}, \lambda$
  for `i=1 to E` **do**
    $(x, y) \sim \mathcal{D}$
    $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$
    $\Theta \leftarrow \Theta - \lambda \nabla_\Theta \mathcal{L}(f_\Theta(x), y)$
  end for

The gradient descent algorithm: Classical Training with SGD

Algorithm 10 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

`best_val` $\leftarrow -\inf$

The gradient descent algorithm: Classical Training with SGD

Algorithm 12 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

`best_val` $\leftarrow -\inf$

for `epoch=1 to E` **do**

The gradient descent algorithm: Classical Training with SGD

Algorithm 14 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

  `best_val` $\leftarrow -\inf$

  for `epoch=1 to E` **do**

    $\mathcal{T} \leftarrow shuffle(\mathcal{D}_{train})$

The gradient descent algorithm: Classical Training with SGD

Algorithm 16 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

  `best_val` $\leftarrow -\inf$

  for `epoch=1 to E` **do**

    $\mathcal{T} \leftarrow shuffle(\mathcal{D}_{train})$

    for $(x, y) \in \mathcal{T}$ **do**

The gradient descent algorithm: Classical Training with SGD

Algorithm 18 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

  `best_val` $\leftarrow -\inf$

  for `epoch=1 to E` **do**

    $\mathcal{T} \leftarrow shuffle(\mathcal{D}_{train})$

    for $(x, y) \in \mathcal{T}$ **do**

      $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$

The gradient descent algorithm: Classical Training with SGD

Algorithm 20 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

  `best_val` $\leftarrow -\inf$

  for `epoch=1 to E` **do**

    $\mathcal{T} \leftarrow shuffle(\mathcal{D}_{train})$

    for $(x, y) \in \mathcal{T}$ **do**

      $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$

      $\Theta \leftarrow \Theta - \lambda \nabla_\Theta \mathcal{L}(f_\Theta(x), y)$

The gradient descent algorithm: Classical Training with SGD

Algorithm 22 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

```
best_val ← − inf
for epoch=1 to E do
    T ← shuffle(D_train)
    for (x, y) ∈ T do
        loss ← L(f_Θ(x), y)
        Θ ← Θ − λ∇_Θ L(f_Θ(x), y)
    end for
    dev_acc ← Evaluate(f, D_val)
```

The gradient descent algorithm: Classical Training with SGD

Algorithm 24 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

```
best_val ← − inf
for epoch=1 to E do
    T ← shuffle(D_train)
    for (x, y) ∈ T do
        loss ← L(f_Θ(x), y)
        Θ ← Θ − λ∇_Θ L(f_Θ(x), y)
    end for
    dev_acc ← Evaluate(f, D_val)
    if dev_acx > best_val then
```

The gradient descent algorithm: Classical Training with SGD

Algorithm 26 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$

```
best_val ← − inf
for epoch=1 to E do
    T ← shuffle(D_train)
    for (x, y) ∈ T do
        loss ← L(f_Θ(x), y)
        Θ ← Θ − λ∇_Θ L(f_Θ(x), y)
    end for
    dev_acc ← Evaluate(f, D_val)
    if dev_acx > best_val then
        Θ̂ ← Θ
        dev_acx ← best_val
    end if
end for
```

The gradient descent algorithm: Classical Training with SGD

Algorithm 28 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$
  best_val $\leftarrow -\inf$
  for epoch=1 to E **do**
    $\mathcal{T} \leftarrow shuffle(\mathcal{D}_{train})$
    for $(x, y) \in \mathcal{T}$ **do**
      $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$
      $\Theta \leftarrow \Theta - \lambda \nabla_\Theta \mathcal{L}(f_\Theta(x), y)$
    end for
    dev_acc $\leftarrow$ Evaluate($f, \mathcal{D}_val$)
    if dev_acx $>$ best_val **then**
      $\hat{\Theta} \leftarrow \Theta$
      dev_acx $\leftarrow$ best_val
    end if
  end for
  Return $\hat{\Theta}$

Algorithm 29 Evaluate

Require: $f, \mathcal{D}_{val}$
  accuracy $\leftarrow 0$
  for $(x, y) \in \mathcal{D}_{val}$ **do**
    $\hat{y} \leftarrow \arg\max_y f_\Theta(x)_y$
    if $> \hat{y} = y$ **then**
      accuracy $\leftarrow$ accuracy+1
    end if
  end for
  Return $\frac{\text{accuracy}}{|\mathcal{D}_{val}|}$

The gradient descent algorithm: Classical Training with SGD

Algorithm 30 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$
  `best_val` $\leftarrow -\inf$
  for `epoch=1 to E` **do**
    $\mathcal{T} \leftarrow shuffle(\mathcal{D}_{train})$
    for $(x, y) \in \mathcal{T}$ **do**
      $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$
      $\Theta \leftarrow \Theta - \lambda \nabla_\Theta \mathcal{L}(f_\Theta(x), y)$
    end for
    `dev_acc` $\leftarrow$ `Evaluate`$(f, \mathcal{D}_val)$
    if `dev_acx` $>$ `best_val` **then**
      $\hat{\Theta} \leftarrow \Theta$
      `dev_acx` $\leftarrow$ `best_val`
    end if
  end for
  Return $\hat{\Theta}$

Algorithm 31 Evaluate

Require: $f, \mathcal{D}_{val}$
  `accuracy` $\leftarrow 0$
  for $(x, y) \in \mathcal{D}_{val}$ **do**
    $\hat{y} \leftarrow \arg\max_y f_\Theta(x)_y$
    if $> \hat{y} = y$ **then**
      `accuracy` $\leftarrow$ `accuracy+1`
    end if
  end for
  Return $\frac{\text{accuracy}}{|\mathcal{D}_{val}|}$

$\rightarrow$Not only accuracy

The gradient descent algorithm: Classical Training with SGD

Algorithm 32 Train

Require: $f, \Theta, \mathcal{D}_{train}, \mathcal{D}_{val}, \lambda$
  `best_val` $\leftarrow -\inf$
  for `epoch=1 to E` **do**
    $\mathcal{T} \leftarrow shuffle(\mathcal{D}_{train})$
    for $(x, y) \in \mathcal{T}$ **do**
      $loss \leftarrow \mathcal{L}(f_\Theta(x), y)$
      $\Theta \leftarrow \Theta - \lambda \nabla_\Theta \mathcal{L}(f_\Theta(x), y)$
    end for
    `dev_acc` $\leftarrow$ `Evaluate`$(f, \mathcal{D}_val)$
    if `dev_acx` $>$ `best_val` **then**
      $\hat{\Theta} \leftarrow \Theta$
      `dev_acx` $\leftarrow$ `best_val`
    end if
  end for
  Return $\hat{\Theta}$

Algorithm 33 Evaluate

Require: $f, \mathcal{D}_{val}$
  `accuracy` $\leftarrow 0$
  for $(x, y) \in \mathcal{D}_{val}$ **do**
    $\hat{y} \leftarrow \arg\max_y f_\Theta(x)_y$
    if $> \hat{y} = y$ **then**
      `accuracy` $\leftarrow$ `accuracy+1`
    end if
  end for
  Return $\frac{\text{accuracy}}{|\mathcal{D}_{val}|}$

$\rightarrow$Not only accuracy

$\rightarrow$Stop training when no improvment

Choosing step-size λ

Step-size (or learning rate *lr*)

$\Theta_{t+1} = \Theta_t - \lambda^{(t)} \nabla_{\Theta^t} \mathcal{L}(o, y) \rightarrow$ How to choose $\lambda^{(t)}$?

Choosing step-size λ

Step-size (or learning rate *lr*)

$\Theta_{t+1} = \Theta_t - \lambda^{(t)} \nabla_{\Theta^t} \mathcal{L}(o, y) \rightarrow$ How to choose $\lambda^{(t)}$?

Vanilla approach:

Choose a small fixed learning rate (default value in pytorch is $\lambda = 1 \times 10^{-2}$)

Choosing step-size λ

Step-size (or learning rate *lr*)

$\Theta_{t+1} = \Theta_t - \lambda^{(t)} \nabla_{\Theta^t} \mathcal{L}(o, y) \rightarrow$ How to choose $\lambda^{(t)}$?

Vanilla approach:

Choose a small fixed learning rate (default value in pytorch is $\lambda = 1 \times 10^{-2}$)

Simple heuristic:

Choosing step-size λ

Step-size (or learning rate *lr*)

$\Theta_{t+1} = \Theta_t - \lambda^{(t)} \nabla_{\Theta^t} \mathcal{L}(o, y) \rightarrow$ How to choose $\lambda^{(t)}$?

Vanilla approach:

Choose a small fixed learning rate (default value in pytorch is $\lambda = 1 \times 10^{-2}$)

Simple heuristic:

1. Start with a smal value, e.g. $\lambda = 1 \times 10^{-2}$

Choosing step-size λ

Step-size (or learning rate *lr*)

$\Theta_{t+1} = \Theta_t - \lambda^{(t)} \nabla_{\Theta^t} \mathcal{L}(o, y) \rightarrow$ How to choose $\lambda^{(t)}$?

Vanilla approach:

Choose a small fixed learning rate (default value in pytorch is $\lambda = 1 \times 10^{-2}$)

Simple heuristic:

1. Start with a smal value, e.g. $\lambda = 1 \times 10^{-2}$
2. If validation accuracy did not improve during last *n* epochs:
   $\rightarrow$ decrease the learning rate, e.g. $\lambda = \alpha.\lambda$ with $\alpha = 1 \times 10^{-1}$

Choosing step-size λ

Step-size (or learning rate *lr*)

$\Theta_{t+1} = \Theta_t - \lambda^{(t)} \nabla_{\Theta^t} \mathcal{L}(o, y) \rightarrow$ How to choose $\lambda^{(t)}$?

Vanilla approach:

Choose a small fixed learning rate (default value in pytorch is $\lambda = 1 \times 10^{-2}$)

Simple heuristic:

1. Start with a smal value, e.g. $\lambda = 1 \times 10^{-2}$
2. If validation accuracy did not improve during last *n* epochs:
   $\rightarrow$ decrease the learning rate, e.g. $\lambda = \alpha.\lambda$ with $\alpha = 1 \times 10^{-1}$

Other heuristic:

- Step decay: multiply λ by $\alpha \in [0, 1]$ every *n* epochs
- Exponential decay: $\lambda^{(t)} = \lambda^{(0)}.e^{-\alpha.t}$
- ...

The gradient descent algorithm:

The gradient descent algorithm:

- Use the direction of the gradient

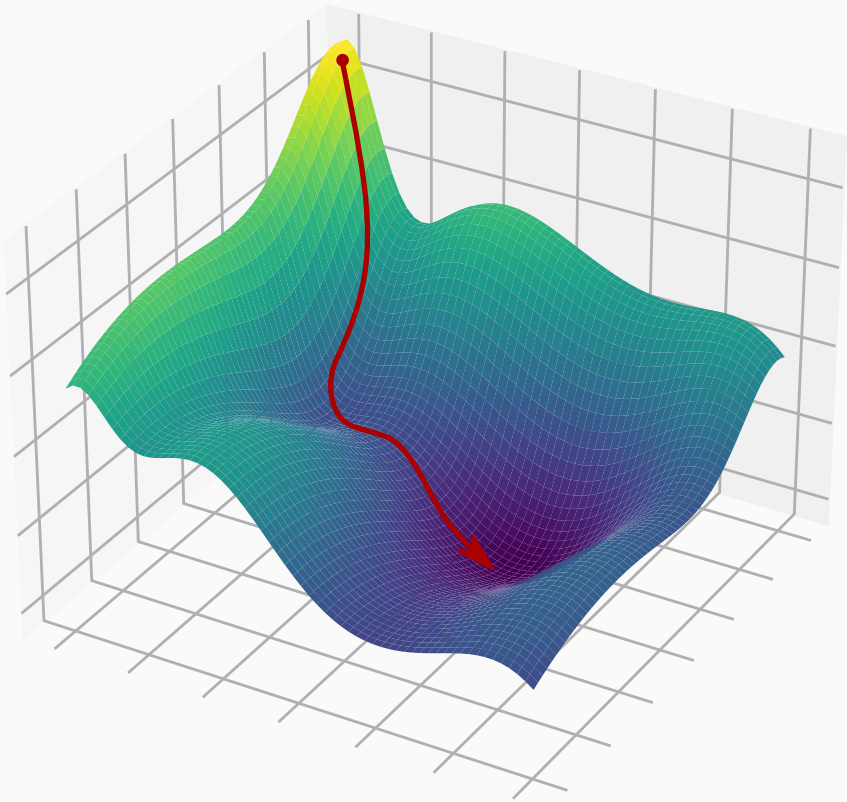
The gradient descent algorithm:

- Use the direction of the gradient
- Do small update

The gradient descent algorithm:

- Use the direction of the gradient
- Do small update

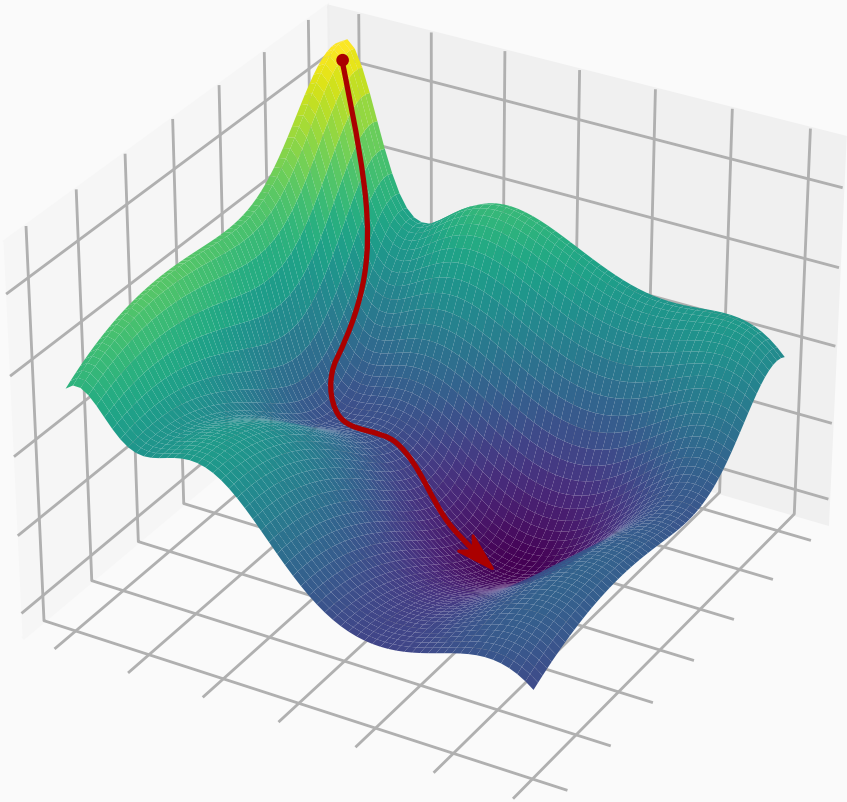
$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta^t} \mathcal{L}(o, y)$$



The gradient descent algorithm:

- Use the direction of the gradient
- Do small update

$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta^t} \mathcal{L}(o, y)$$

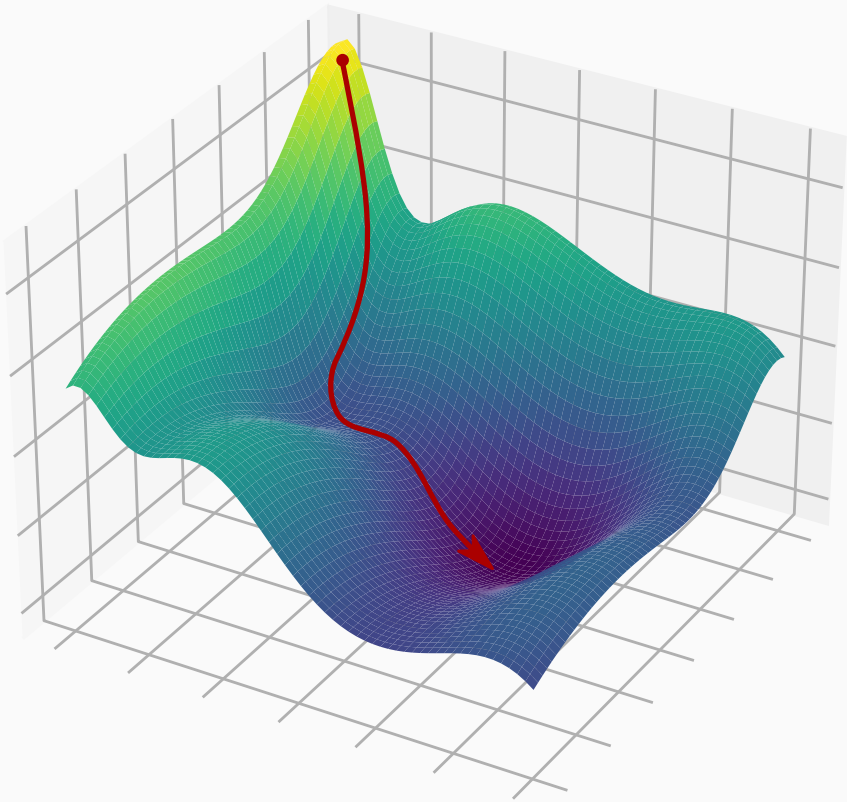


How to find the gradient for each parameters?

The gradient descent algorithm:

- Use the direction of the gradient
- Do small update

$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta^t} \mathcal{L}(o, y)$$



How to find the gradient for each parameters?

Use the back-propagation algorithm!

The backpropagation algorithm

Derivative of function:

Let be f a function:

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto f(x)$$

[0]NB: In this chapter we will consider all the function differentiable at least by part

Derivative of function:

Let be f a function:

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto f(x)$$

For a given x, how does an infinitesimal change of x impact the output?

[0] NB: In this chapter we will consider all the function differentiable at least by part

Derivative of function:

Let be f a function:

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto f(x)$$

For a given x, how does an infinitesimal change of x impact the output?

$$\frac{df}{dx} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

[0]NB: In this chapter we will consider all the function differentiable at least by part

The backpropagation algorithm (basics): chain rule

Derivative of composition (chain rule):

Let be f defined by (composition of function):

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto h(g(x))$$

Then we can write $f'(x)$ as:

Derivative of composition (chain rule):

Let be f defined by (composition of function):

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto h(g(x))$$

Then we can write $f'(x)$ as:

$$f'(x) = \lim_{\epsilon \to 0} \frac{h(g(x+\epsilon)) - h(g(x))}{\epsilon} \tag{1}$$
$$= \lim_{\epsilon \to 0} \frac{h(g(x+\epsilon)) - h(g(x))}{g(x+\epsilon) - g(x)} \frac{g(x+\epsilon) - g(x)}{\epsilon} \tag{2}$$
$$= \lim_{\epsilon \to 0} \frac{h(g(x+\epsilon)) - h(g(x))}{g(x+\epsilon) - g(x)} g'(x) \tag{3}$$
$$= \lim_{\epsilon \to 0} \frac{h(g(x)+k) - h(g(x))}{k} g'(x) \text{ with } k = g(x+\epsilon) - g(x) \tag{4}$$
$$= h'(g(x)).g'(x) \text{ cause } \lim_{\epsilon \to 0} k = \lim_{\epsilon \to 0} g(x+\epsilon) - g(x) = 0 \tag{5}$$

The backpropagation algorithm (basics): example (chain rule)

A naive example:

Let be f such that $f(x) = (2x + 1)^2$

$\rightarrow$ **What is its derivative $f'(x)$?**

The backpropagation algorithm (basics): example (chain rule)

A naive example:

Let be f such that $f(x) = (2x + 1)^2$

$\rightarrow$ **What is its derivative $f'(x)$?**

Example without the chain rule:

$$\begin{aligned} f'(x) &= \Big((2x+1)^2\Big)' \\ &= (4x^2 + 4x + 1)' \\ &= 8x + 4 \end{aligned}$$

The backpropagation algorithm (basics): example (chain rule)

A naive example:

Let be f such that $f(x) = (2x+1)^2$

$\rightarrow$ **What is its derivative $f'(x)$?**

Example without the chain rule:

$$\begin{aligned} f'(x) &= \Big((2x+1)^2\Big)' \\ &= (4x^2 + 4x + 1)' \\ &= 8x + 4 \end{aligned}$$

Example with the chain rule:

Let $g(x) = 2x + 1$ and $h(x) = x^2$

$$\begin{aligned} f'(x) &= (h \circ g)'(x) = \frac{dh(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx} \\ &= 2g(x) \times 2 \\ &= 2(2x+1) \times 2 \\ &= 8x + 4 \end{aligned}$$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$
- $o = a^{(3)}z^{(2)}$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$
- $o = a^{(3)}z^{(2)}$

and the loss function $\mathcal{L}(o, y) = (o - y)^2$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$
- $o = a^{(3)}z^{(2)}$

and the loss function $\mathcal{L}(o, y) = (o - y)^2$

Gradient Descent:

For one example in the dataset

$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta_t} \mathcal{L}(o, y)$$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$
- $o = a^{(3)}z^{(2)}$

and the loss function $\mathcal{L}(o, y) = (o - y)^2$

Gradient Descent:

For one example in the dataset

$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta_t} \mathcal{L}(o, y)$$

More precisely,

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$
- $o = a^{(3)}z^{(2)}$

and the loss function $\mathcal{L}(o, y) = (o - y)^2$

Gradient Descent:

For one example in the dataset

$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta_t} \mathcal{L}(o, y)$$

More precisely,

- $a^{(1,t+1)} = a^{(1,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(1,t)}}$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$
- $o = a^{(3)}z^{(2)}$

and the loss function $\mathcal{L}(o, y) = (o - y)^2$

Gradient Descent:

For one example in the dataset

$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta_t} \mathcal{L}(o, y)$$

More precisely,

- $a^{(1,t+1)} = a^{(1,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(1,t)}}$
- $a^{(2,t+1)} = a^{(2,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(2,t)}}$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$
- $o = a^{(3)}z^{(2)}$

and the loss function $\mathcal{L}(o, y) = (o - y)^2$

Gradient Descent:

For one example in the dataset

$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta_t} \mathcal{L}(o, y)$$

More precisely,

- $a^{(1,t+1)} = a^{(1,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(1,t)}}$
- $a^{(2,t+1)} = a^{(2,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(2,t)}}$
- $a^{(3,t+1)} = a^{(3,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(3,t)}}$

The backpropagation algorithm: Scalar NN example

Scalar parametrized neural network:

Let $a^{(i)} \in \mathbb{R}$ (scalar) and $\sigma(x) = x^2$

We consider the following neural network:

- $z^{(1)} = \sigma(a^{(1)}x)$
- $z^{(2)} = \sigma(a^{(2)}z^{(1)})$
- $o = a^{(3)}z^{(2)}$

and the loss function $\mathcal{L}(o, y) = (o - y)^2$

Gradient Descent:

For one example in the dataset

$$\Theta_{t+1} = \Theta_t - \lambda \nabla_{\Theta_t} \mathcal{L}(o, y)$$

More precisely,

- $a^{(1,t+1)} = a^{(1,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(1,t)}}$
- $a^{(2,t+1)} = a^{(2,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(2,t)}}$
- $a^{(3,t+1)} = a^{(3,t)} - \lambda \frac{d\mathcal{L}(o,y)}{da^{(3,t)}}$

What is the gradient for all parameters i.e. $\frac{d\mathcal{L}(o,y)}{da^{(3,t)}}, \frac{d\mathcal{L}(o,y)}{da^{(2,t)}}, \frac{d\mathcal{L}(o,y)}{da^{(1,t)}}$?

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{da^{(3)}}$$

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{da^{(3)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d(o-y)^2}{do} \times \frac{da^{(3)}z^{(2)}}{da^{(3)}}$$

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{da^{(3)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d(o-y)^2}{do} \times \frac{da^{(3)}z^{(2)}}{da^{(3)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{do}$

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{da^{(3)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d(o-y)^2}{do} \times \frac{da^{(3)}z^{(2)}}{da^{(3)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{do}$

$$\frac{d\mathcal{L}(o,y)}{do} = \frac{d(o-y)^2}{do}$$

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{da^{(3)}}$$
$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d(o-y)^2}{do} \times \frac{da^{(3)}z^{(2)}}{da^{(3)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{do}$

$$\frac{d\mathcal{L}(o,y)}{do} = \frac{d(o-y)^2}{do}$$
$$= \frac{du^2}{du} \times \frac{d(o-y)}{do}$$

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{da^{(3)}}$$
$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d(o-y)^2}{do} \times \frac{da^{(3)}z^{(2)}}{da^{(3)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{do}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d(o-y)^2}{do} \\ &= \frac{du^2}{du} \times \frac{d(o-y)}{do} \\ &= 2u \times 1\end{aligned}$$

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{da^{(3)}}$$
$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d(o-y)^2}{do} \times \frac{da^{(3)}z^{(2)}}{da^{(3)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{do}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d(o-y)^2}{do} \\ &= \frac{du^2}{du} \times \frac{d(o-y)}{do} \\ &= 2u \times 1 \\ &= 2(o-y) \times 1\end{aligned}$$

Let us first derivate $\frac{do}{da^{(3)}}$

The backpropagation algorithm: Scalar NN example

Let start with $\frac{d\mathcal{L}(o,y)}{da^{(3)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{da^{(3)}}$$
$$\frac{d\mathcal{L}(o,y)}{da^{(3)}} = \frac{d(o-y)^2}{do} \times \frac{da^{(3)}z^{(2)}}{da^{(3)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{do}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d(o-y)^2}{do} \\ &= \frac{du^2}{du} \times \frac{d(o-y)}{do} \\ &= 2u \times 1 \\ &= 2(o-y) \times 1\end{aligned}$$

Let us first derivate $\frac{do}{da^{(3)}}$

$$\frac{da^{(3)}z^{(2)}}{da^{(3)}} = z^{(2)}$$

We finally obtain: $\frac{d\mathcal{L}(o,y)}{da^{(3)}} = 2(o-y) \times z^{(2)}$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(2)}}{da^{(2)}}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(2)}}{da^{(2)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(2)}}$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(2)}}{da^{(2)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{do} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{dz^{(2)}}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(2)}}{da^{(2)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{do} = \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{dz^{(2)}}$$

$$= \frac{d\mathcal{L}(o,y)}{do} \times \frac{da^{(3)}z^{(2)}}{dz^{(2)}}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(2)}}{da^{(2)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(2)}}$

$$\begin{aligned}
\frac{d\mathcal{L}(o,y)}{do} &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{dz^{(2)}} \\
&= \frac{d\mathcal{L}(o,y)}{do} \times \frac{da^{(3)}z^{(2)}}{dz^{(2)}} \\
&= \frac{d\mathcal{L}(o,y)}{do} \times a^3
\end{aligned}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(2)}}{da^{(2)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(2)}}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{dz^{(2)}} \\ &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{da^{(3)}z^{(2)}}{dz^{(2)}} \\ &= \frac{d\mathcal{L}(o,y)}{do} \times a^3\end{aligned}$$

Let us first derivate the $\frac{dz^{(2)}}{da^{(2)}}$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(2)}}{da^{(2)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(2)}}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{dz^{(2)}} \\ &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{da^{(3)}z^{(2)}}{dz^{(2)}} \\ &= \frac{d\mathcal{L}(o,y)}{do} \times a^3\end{aligned}$$

Let us first derivate the $\frac{dz^{(2)}}{da^{(2)}}$

$$\begin{aligned}\frac{d\sigma(a^{(2)}z^{(1)})}{da^{(2)}} &= \frac{d\sigma u}{du} \times \frac{da^{(2)}z^{(1)}}{da^{(2)}} \\ &= 2u \times z^{(1)} \\ &= 2a^2z^{(2)} \times z^{(1)}\end{aligned}$$

We finally obtain:

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(2)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}}$$
$$\frac{d\mathcal{L}(o,y)}{da^{(2)}} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(2)}}{da^{(2)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(2)}}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{dz^{(2)}} \\ &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{da^{(3)}z^{(2)}}{dz^{(2)}} \\ &= \frac{d\mathcal{L}(o,y)}{do} \times a^3\end{aligned}$$

Let us first derivate the $\frac{dz^{(2)}}{da^{(2)}}$

$$\begin{aligned}\frac{d\sigma(a^{(2)}z^{(1)})}{da^{(2)}} &= \frac{d\sigma u}{du} \times \frac{da^{(2)}z^{(1)}}{da^{(2)}} \\ &= 2u \times z^{(1)} \\ &= 2a^2z^{(2)} \times z^{(1)}\end{aligned}$$

We finally obtain:

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{da^{(2)}} &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{dz^{(2)}} \times \frac{dz^{(2)}}{da^{(2)}} \\ &= 2(o-y) \times a^{(3)} \times 2a^{(2)}z^{(2)} \times z^{(1)}\end{aligned}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{da^{(2)}z^{(1)}}{da^{(1)}}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{da^{(2)}z^{(1)}}{da^{(1)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(1)}}$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{da^{(2)}z^{(1)}}{da^{(1)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{do} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{do}{dz^{(1)}}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{da^{(2)}z^{(1)}}{da^{(1)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{do} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{do}{dz^{(1)}}$$

$$= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(1)}}{dz^{(1)}}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{da^{(2)}z^{(1)}}{da^{(1)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(1)}}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{do}{dz^{(1)}} \\ &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(1)}}{dz^{(1)}} \\ &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times a^{(2)}\end{aligned}$$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{da^{(2)}z^{(1)}}{da^{(1)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(1)}}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{do}{dz^{(1)}} \\ &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(1)}}{dz^{(1)}} \\ &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times a^{(2)}\end{aligned}$$

Let us first derivate the $\frac{dz^{(1)}}{da^{(1)}}$

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{da^{(2)}z^{(1)}}{da^{(1)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{do} = \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{do}{dz^{(1)}}$$

$$= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(1)}}{dz^{(1)}}$$

$$= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times a^{(2)}$$

Let us first derivate the $\frac{dz^{(1)}}{da^{(1)}}$

$$\frac{d\sigma(a^{(1)}x)}{da^{(2)}} = \frac{d\sigma u}{du} \times \frac{da^{(1)}x}{da^{(1)}}$$

$$= 2u \times x$$

$$= 2a^{(1)}z^{(1)} \times x$$

We finally obtain:

The backpropagation algorithm: Scalar NN example

Let us find $\rightarrow \frac{d\mathcal{L}(o,y)}{da^{(1)}}$

$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{dz^{(2)}}{da^{(2)}}$$
$$\frac{d\mathcal{L}(o,y)}{da^{(1)}} = \frac{d\mathcal{L}(o,y)}{dz^{(1)}} \times \frac{da^{(2)}z^{(1)}}{da^{(1)}}$$

Let us derivate $\frac{d\mathcal{L}(o,y)}{dz^{(1)}}$

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{do} &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{do}{dz^{(1)}} \\ &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{da^{(2)}z^{(1)}}{dz^{(1)}} \\ &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times a^{(2)}\end{aligned}$$

Let us first derivate the $\frac{dz^{(1)}}{da^{(1)}}$

$$\begin{aligned}\frac{d\sigma(a^{(1)}x)}{da^{(2)}} &= \frac{d\sigma u}{du} \times \frac{da^{(1)}x}{da^{(1)}} \\ &= 2u \times x \\ &= 2a^{(1)}z^{(1)} \times x\end{aligned}$$

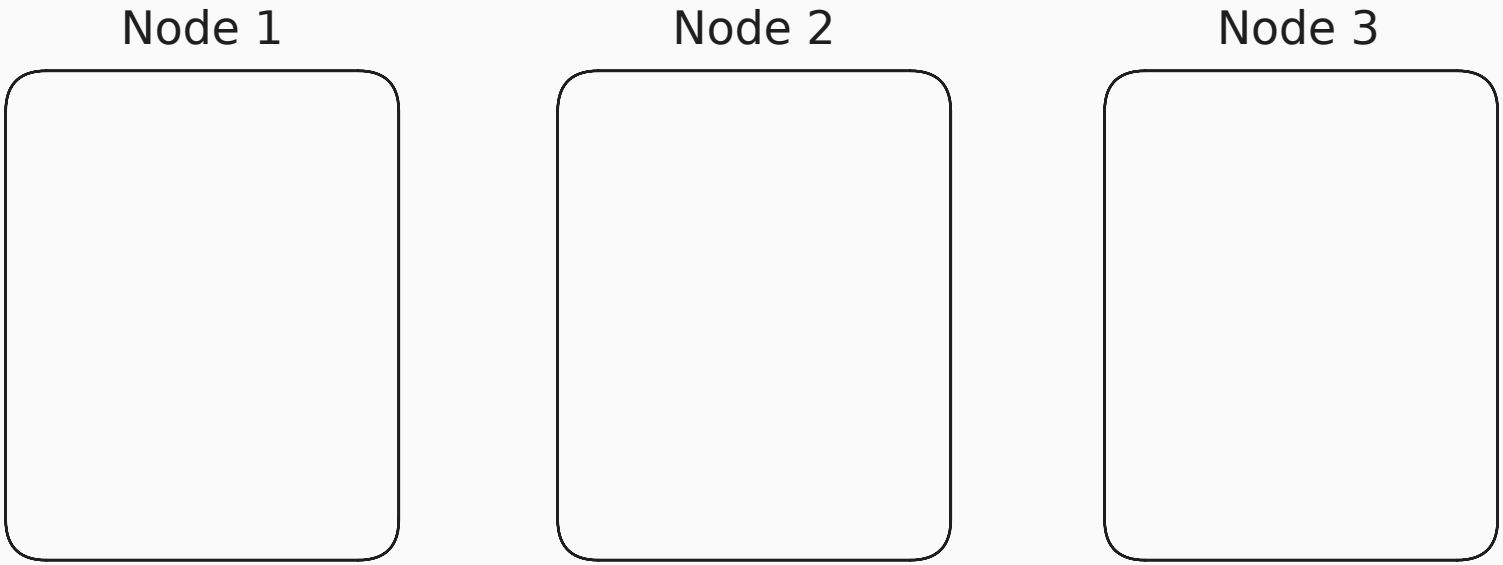
We finally obtain:

$$\begin{aligned}\frac{d\mathcal{L}(o,y)}{da^{(2)}} &= \frac{d\mathcal{L}(o,y)}{dz^{(2)}} \times \frac{dz^{(2)}}{dz^{(1)}} \times \frac{dz^{(1)}}{da^{(1)}} \\ &= \frac{d\mathcal{L}(o,y)}{do} \times \frac{do}{dz^{(2)}} \times \frac{dz^{(2)}}{dz^{(1)}} \times \frac{dz^{(1)}}{da^{(1)}}\end{aligned}$$

Computational graph

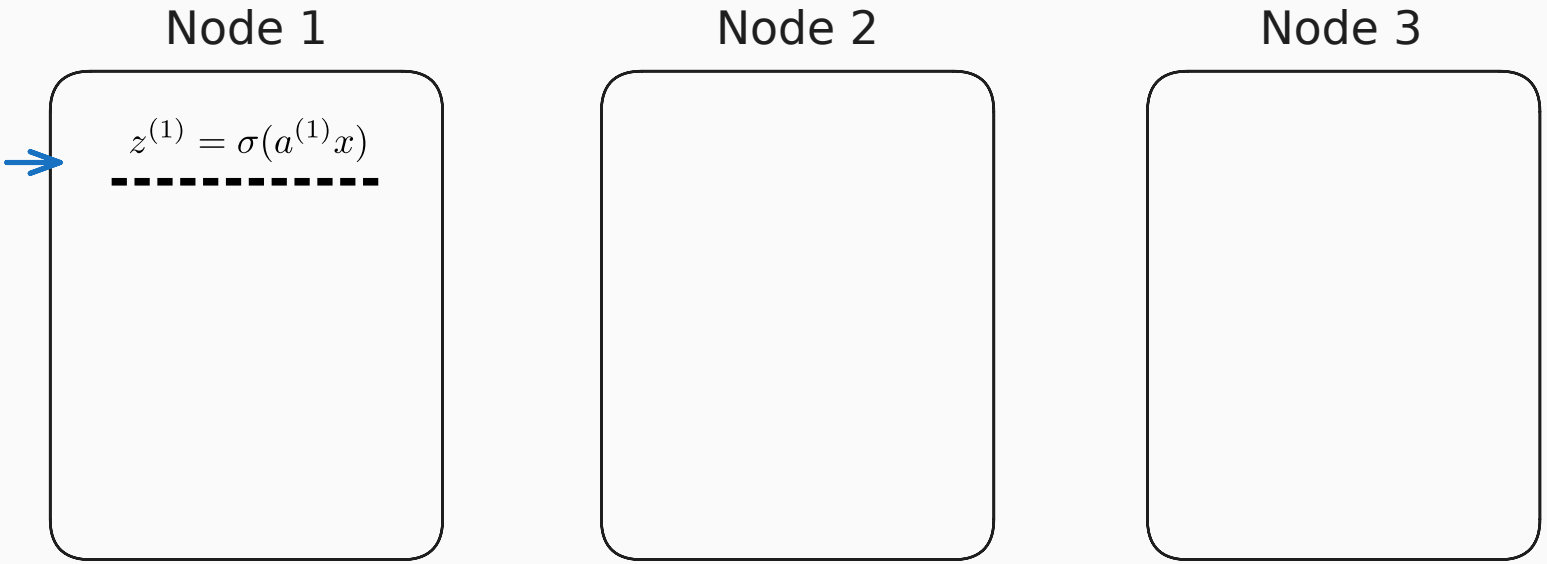
How did we proceed?

FORWARD



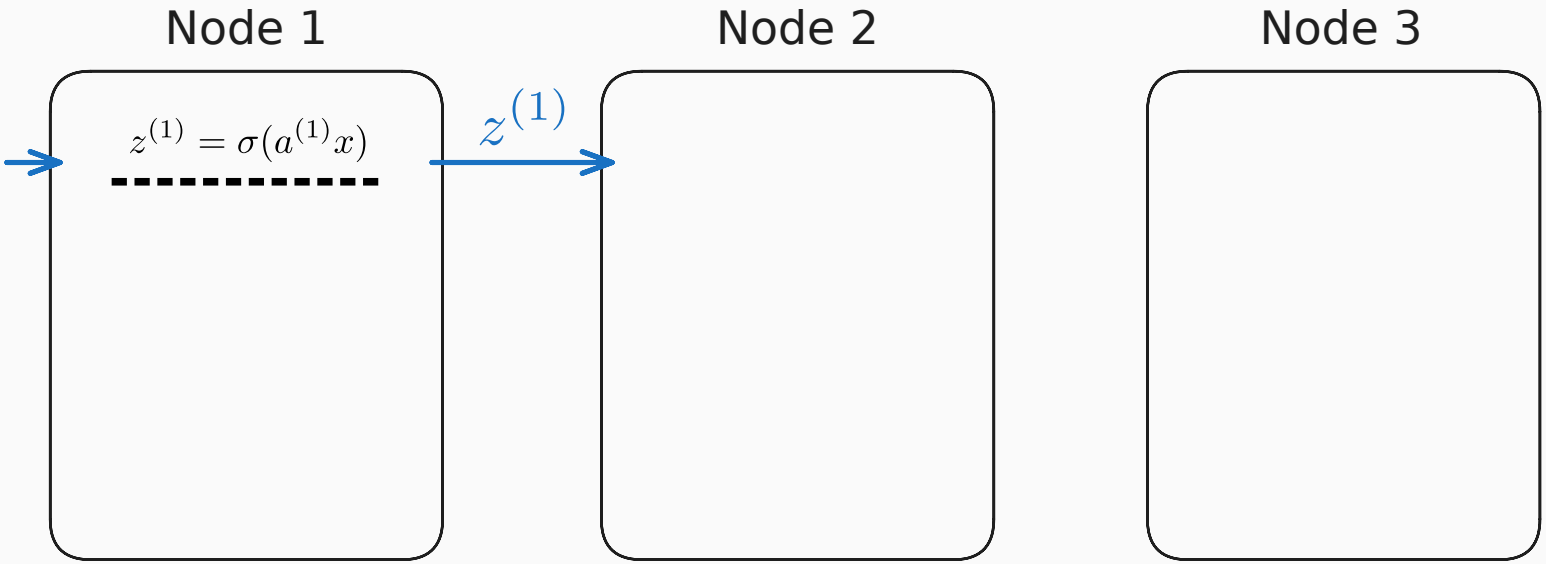
How did we proceed?

FORWARD



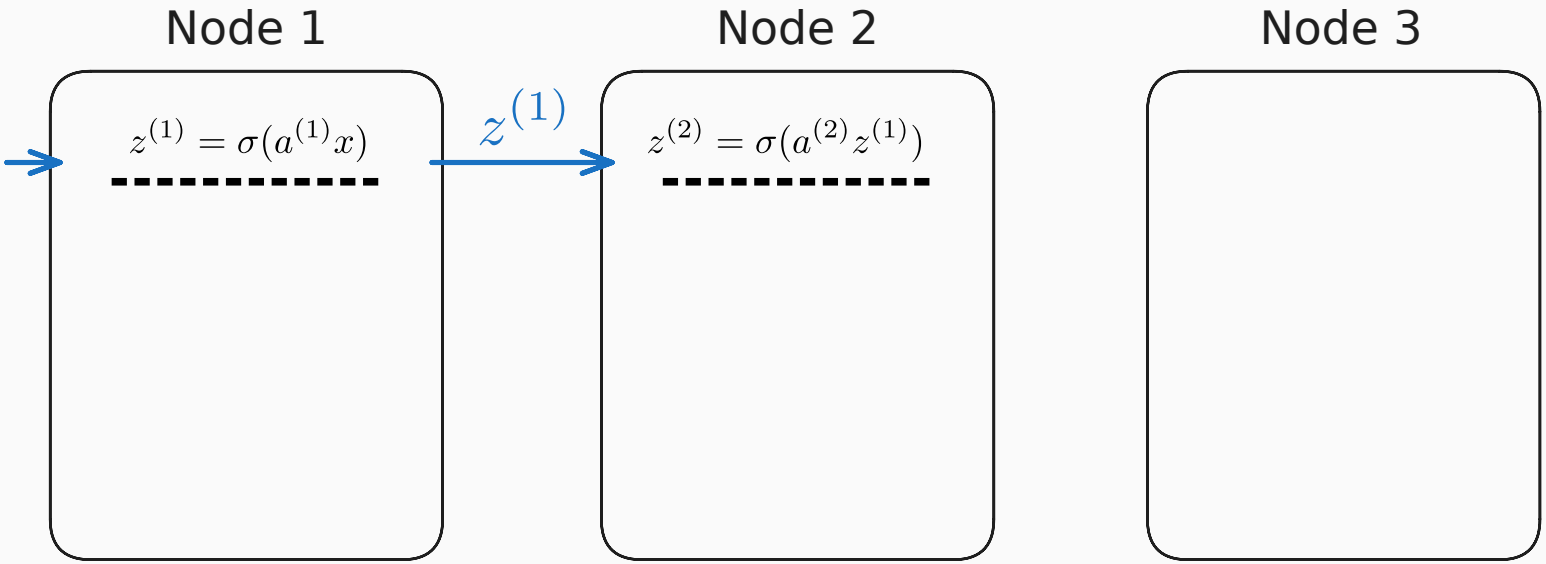
How did we proceed?

FORWARD



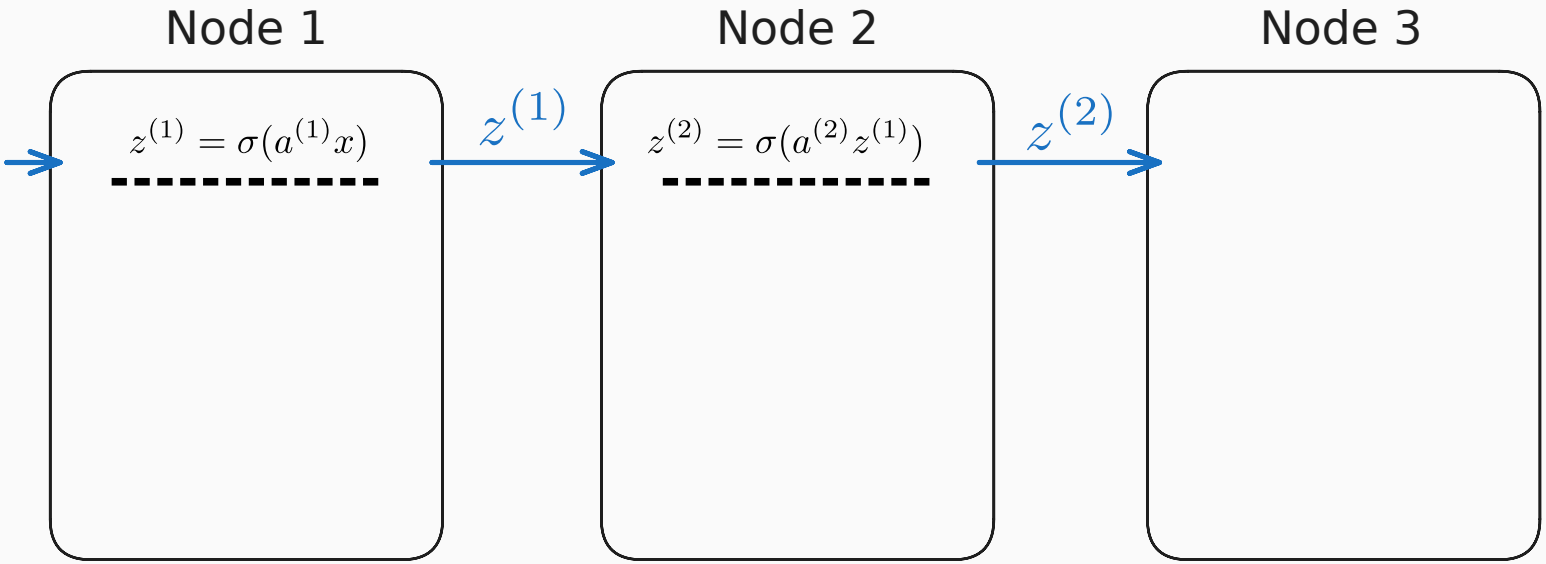
How did we proceed?

FORWARD



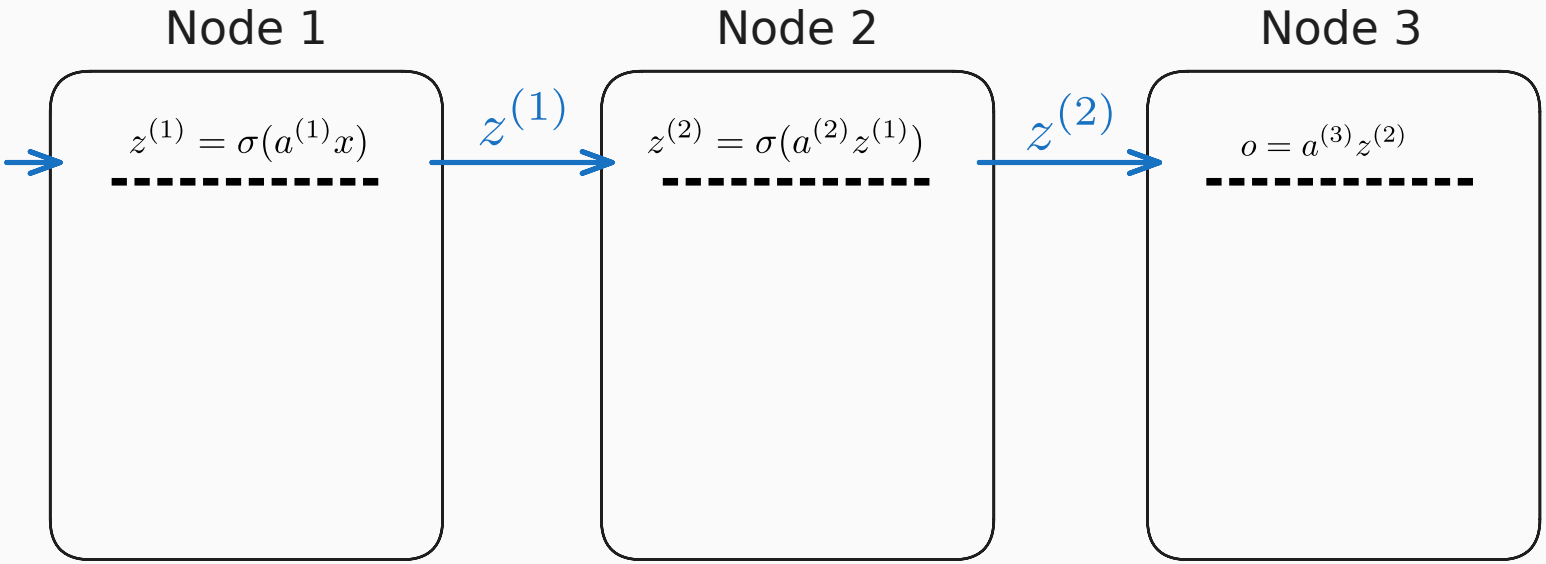
How did we proceed?

FORWARD



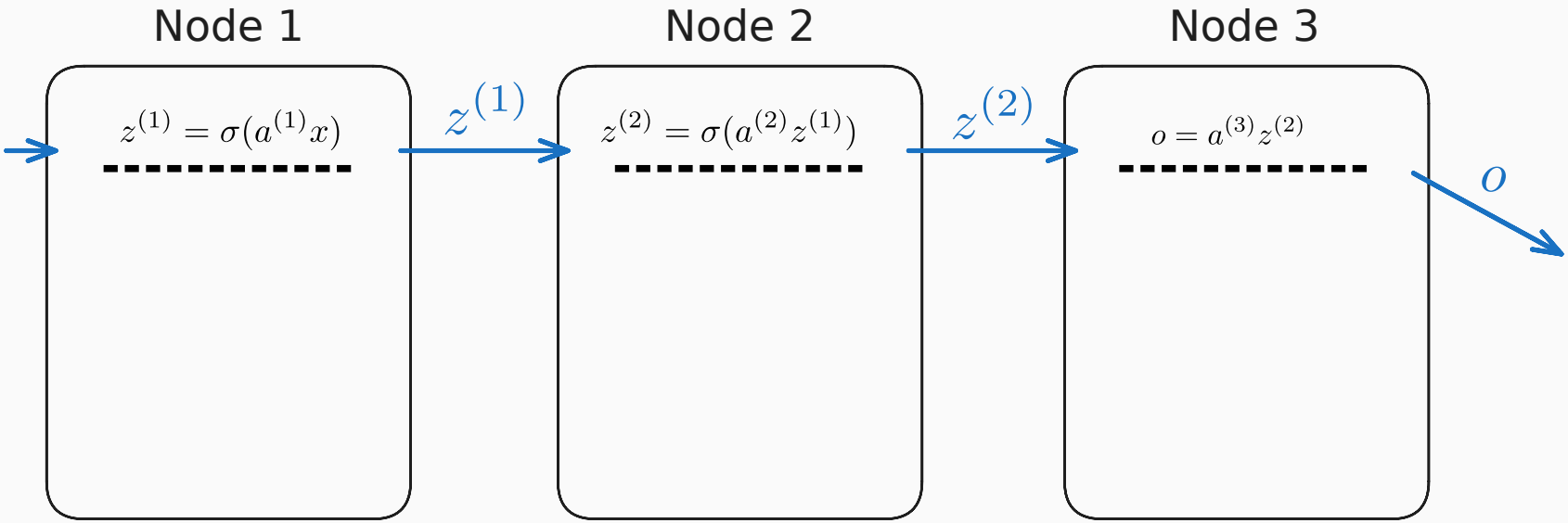
How did we proceed?

FORWARD

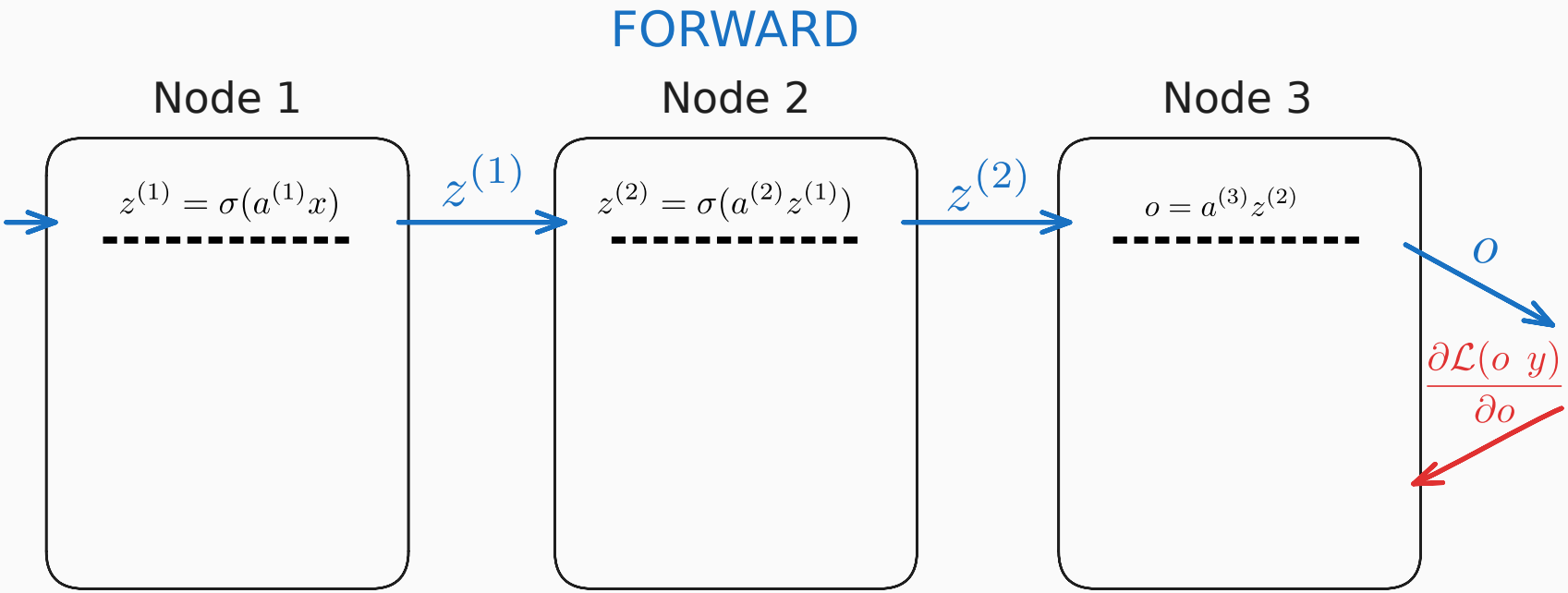


How did we proceed?

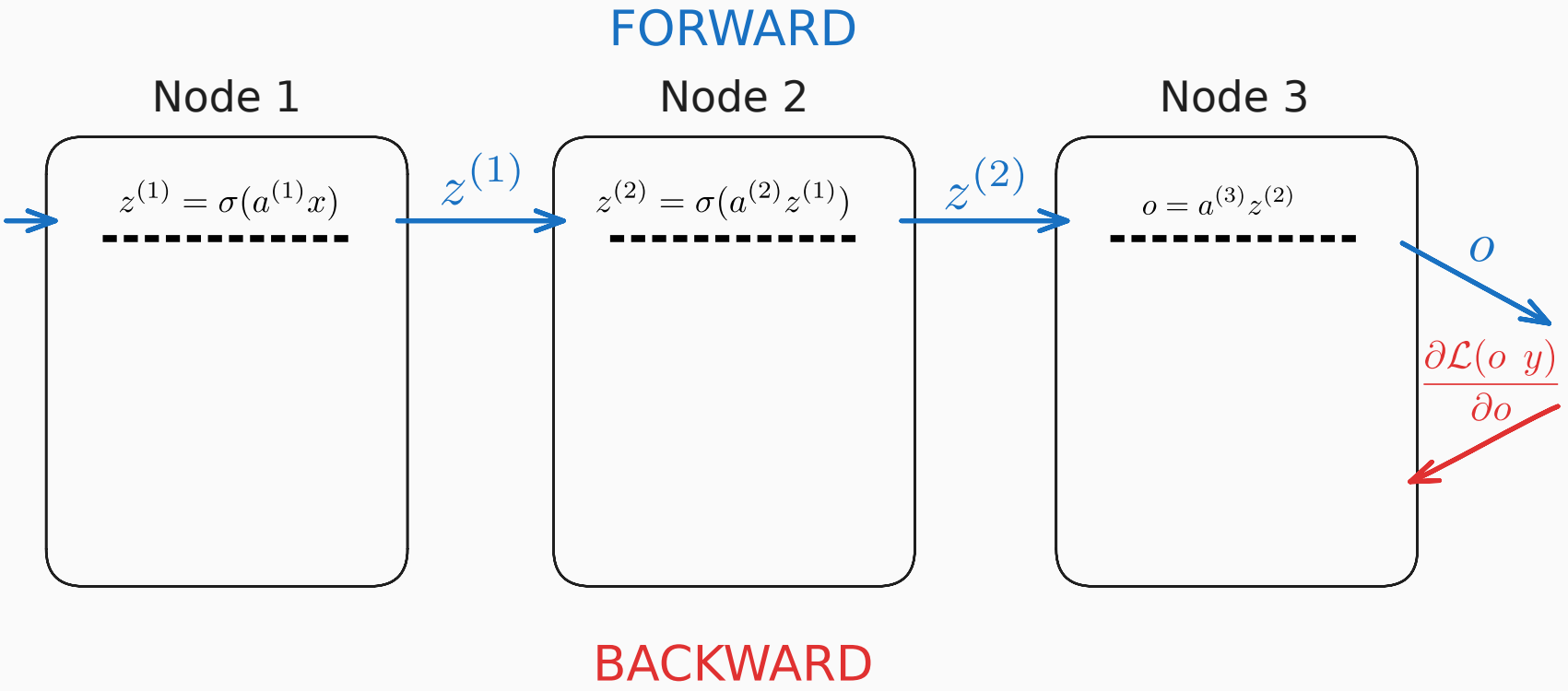
FORWARD



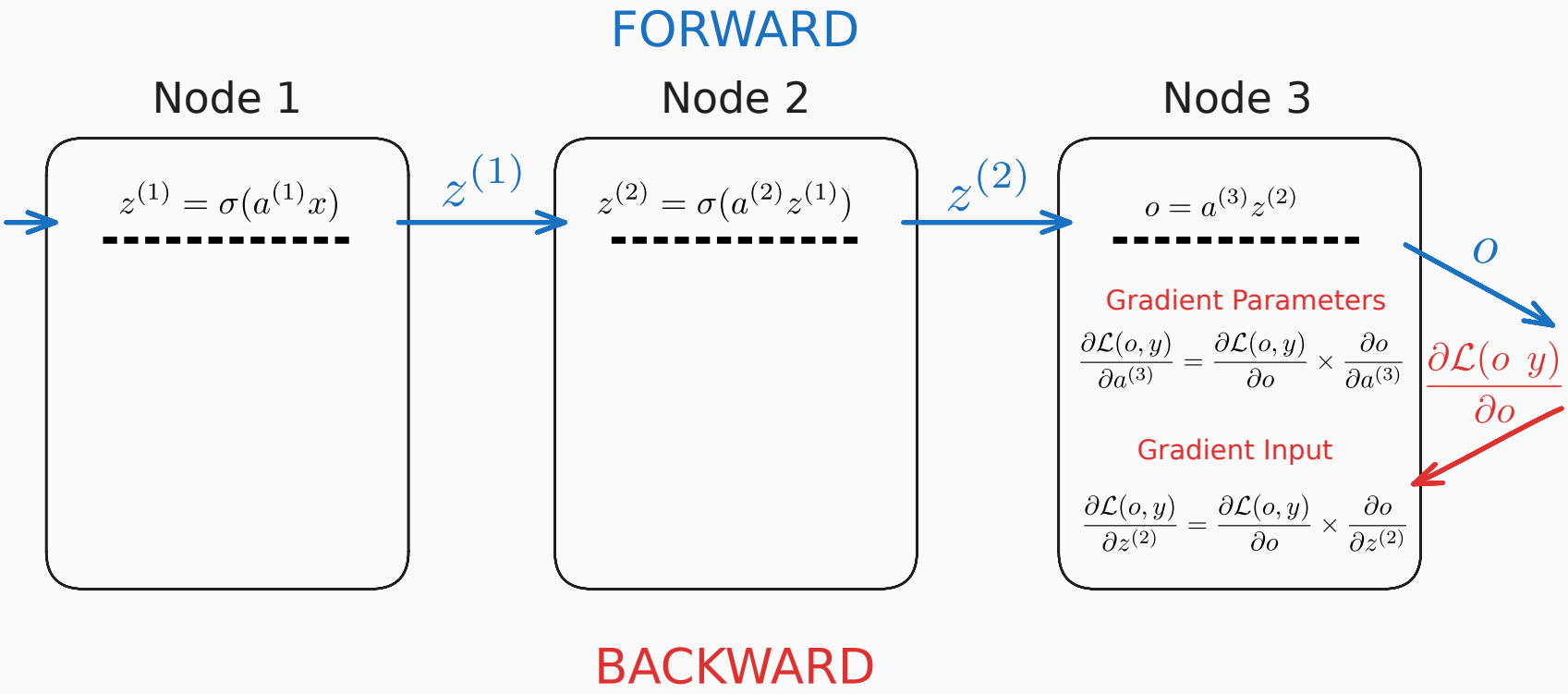
How did we proceed?



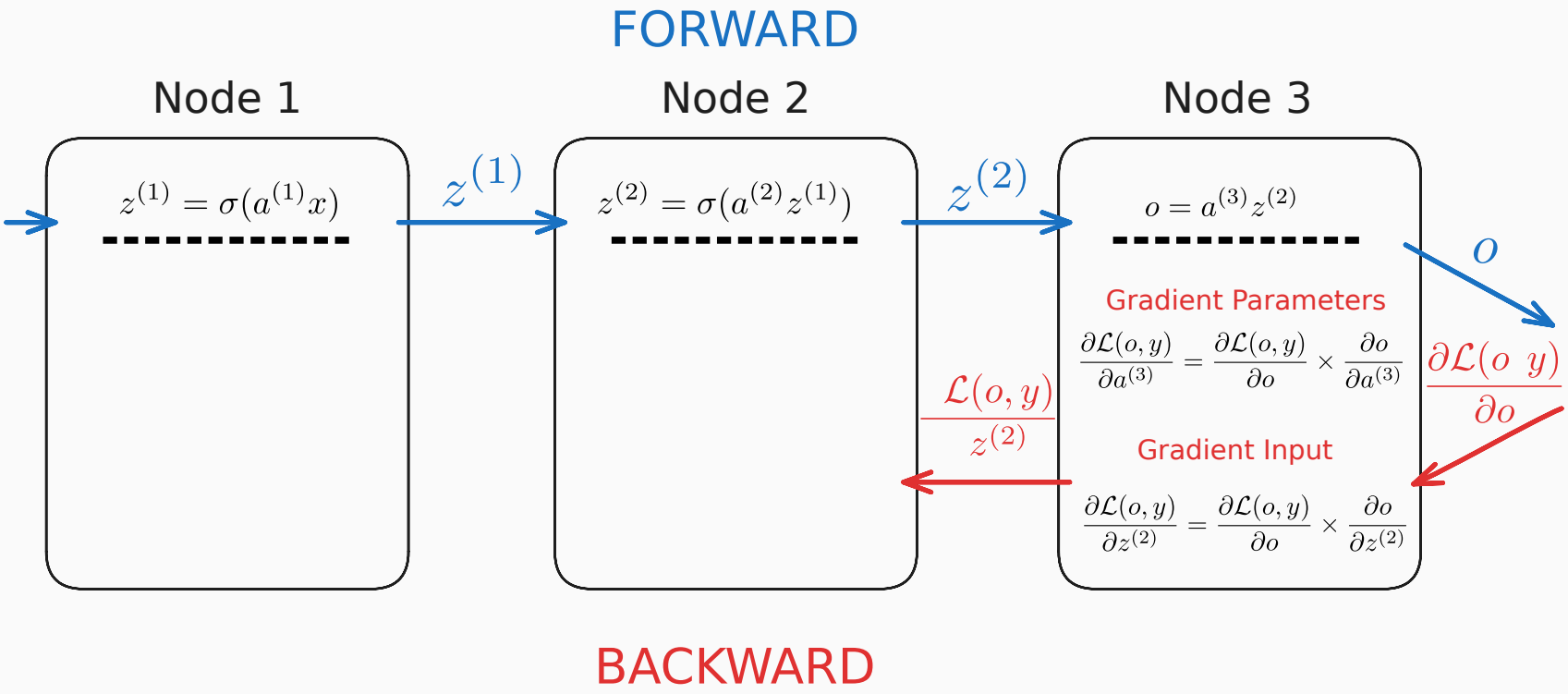
How did we proceed?



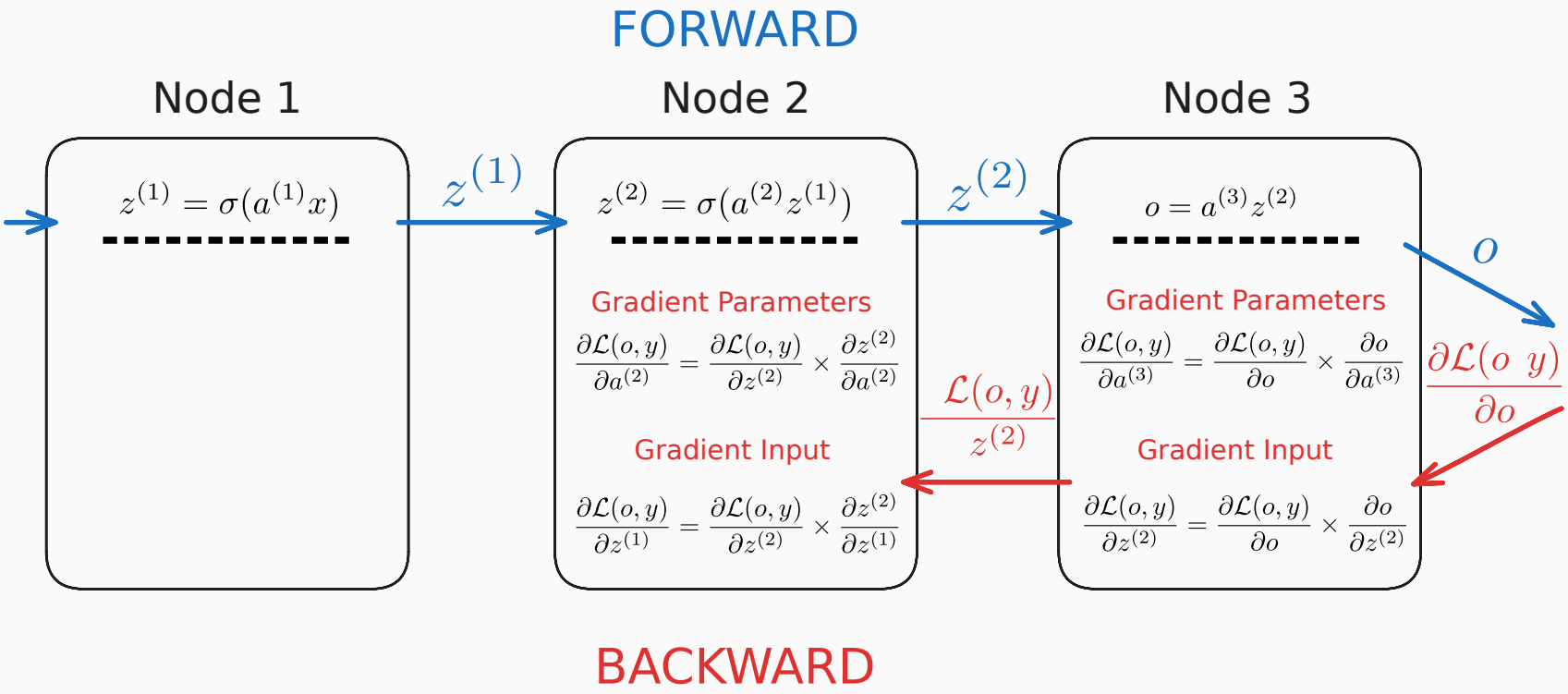
How did we proceed?



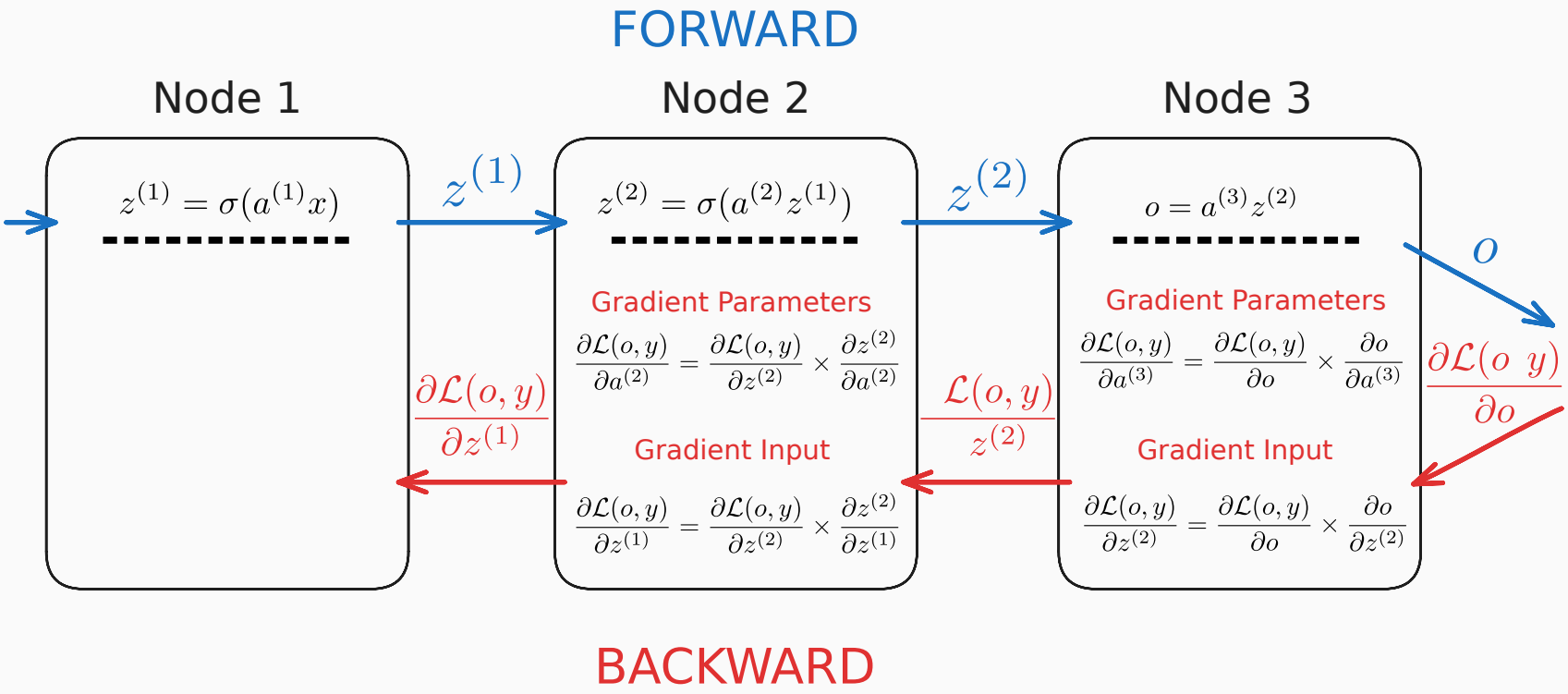
How did we proceed?



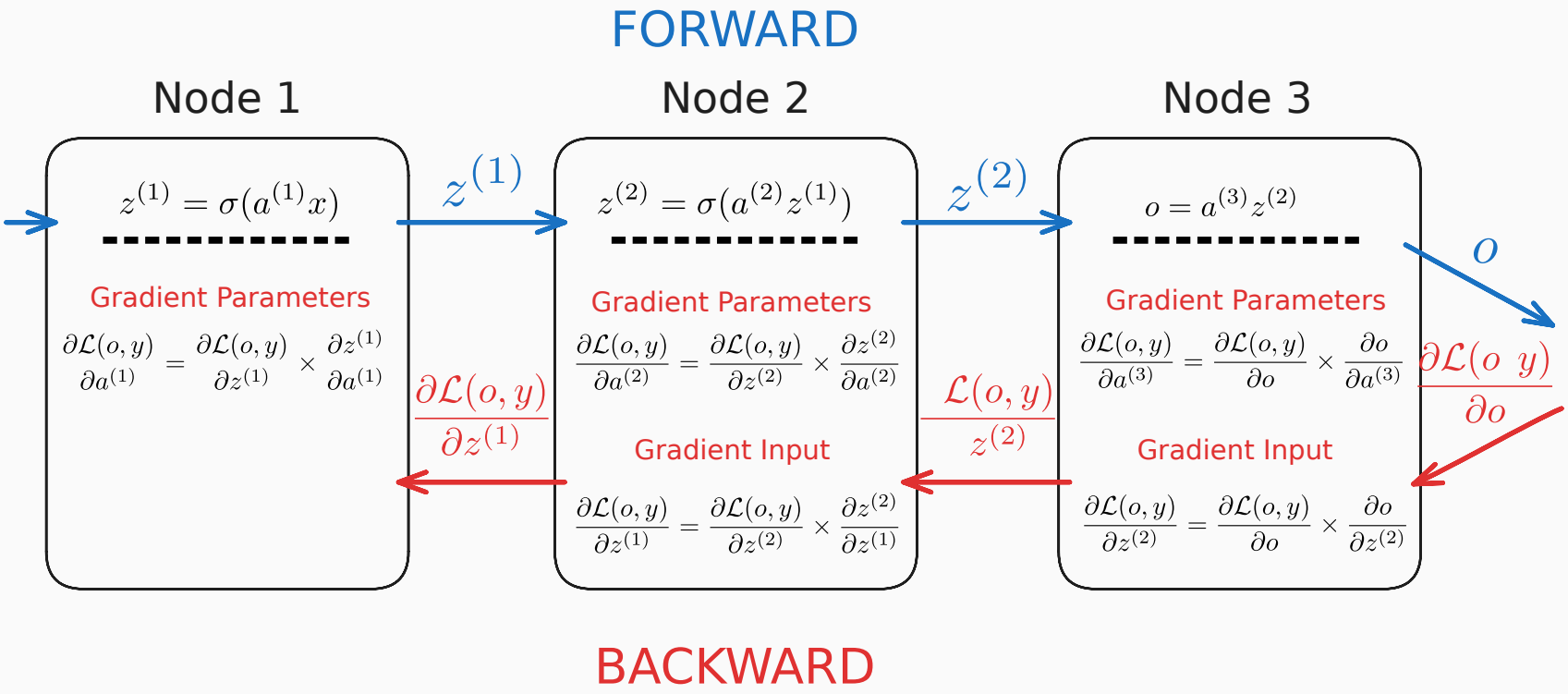
How did we proceed?



How did we proceed?



How did we proceed?



Computational graph: Automatically compute the gradient

Computational graph:

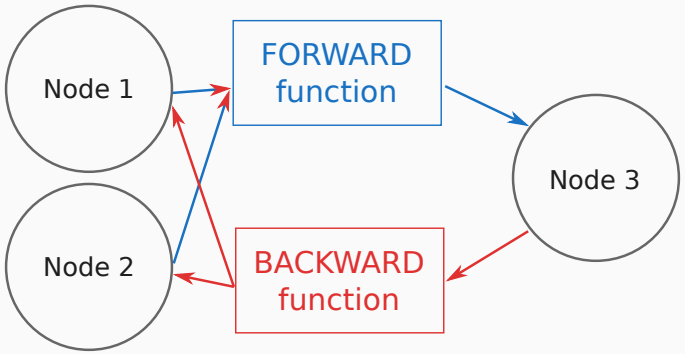
- At forward create the graph
- Each operation consist of a node:
  - For each operation (in forward) retain input
  - For each operation (in forward), retain the gradient function
  - In backward return to the previous operation gradient of error/loss and the input stored

What unit operation?

- Larger $\Longrightarrow$ optimization possible for backward pass (factorization of the function)
- Smaller $\Longrightarrow$ less operations to implement

$\rightarrow$ **Ideally both**

Computational graph: How to implement?



The Nodes:

- Each node must contains information on the graph
  - A pointer on the gradient function (backward function)
  - The previous nodes (In our case node 1 and 2)
  - Its gradient
- These information are added in the forward function
- In the backward step the node call the backward function with previous nodes and gradient

Differentiation with vector inputs

With vector input
Let $f : \mathbb{R}^m \to \mathbb{R}$ be a function and $x \in \mathbb{R}^m, y \in \mathbb{R}$ be variables such that :

$$y = f(x)$$

Partial derivative

For a given x, how an infinitesimal change of x_i impact y?

$$\frac{\partial y}{\partial x_i}$$

Gradient

For a given x, how does an infinitesimal change of x impact y?

$$\nabla_x y = \begin{pmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_m} \end{pmatrix}$$

Differentiation with vector inputs (example)

Example with scalar product

Let $x \in \mathbb{R}^3$ and $a \in \mathbb{R}^3$ be vectors and $f : \mathbb{R}^3 \to \mathbb{R}$ such that:

$$f(x) = \sum_{i=1}^{3} x_i a_i = x_1 a_1 + x_2 b_2 + x_3 b_3$$

Partial derivative

What is $\frac{\partial f(x)}{\partial x_i}$?

Differentiation with vector inputs (example)

Example with scalar product

Let $x \in \mathbb{R}^3$ and $a \in \mathbb{R}^3$ be vectors and $f : \mathbb{R}^3 \to \mathbb{R}$ such that:

$$f(x) = \sum_{i=1}^{3} x_i a_i = x_1 a_1 + x_2 b_2 + x_3 b_3$$

Partial derivative

What is $\frac{\partial f(x)}{\partial x_i}$?

$$\frac{\partial f(x)}{\partial x_i} = a_i$$

Differentiation with vector inputs (example)

Example with scalar product

Let $x \in \mathbb{R}^3$ and $a \in \mathbb{R}^3$ be vectors and $f : \mathbb{R}^3 \to \mathbb{R}$ such that:

$$f(x) = \sum_{i=1}^{3} x_i a_i = x_1 a_1 + x_2 b_2 + x_3 b_3$$

Partial derivative

What is $\frac{\partial f(x)}{\partial x_i}$?

$$\frac{\partial f(x)}{\partial x_i} = a_i$$

Gradient

What is $\nabla_x f(x)$?

Differentiation with vector inputs (example)

Example with scalar product

Let $x \in \mathbb{R}^3$ and $a \in \mathbb{R}^3$ be vectors and $f : \mathbb{R}^3 \to \mathbb{R}$ such that:

$$f(x) = \sum_{i=1}^{3} x_i a_i = x_1 a_1 + x_2 b_2 + x_3 b_3$$

Partial derivative

What is $\frac{\partial f(x)}{\partial x_i}$?

$$\frac{\partial f(x)}{\partial x_i} = a_i$$

Gradient

What is $\nabla_x f(x)$?

$$\nabla_x f(x) = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Differentiation with vector inputs (example 2 linear function)

Let consider $A \in \mathbb{R}^{n \times m}$ and $x \in \mathbb{R}^m$:

Differentiation with vector inputs (example 2 linear function)

Let consider $A \in \mathbb{R}^{n \times m}$ and $x \in \mathbb{R}^m$:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix}, \; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{6}$$

Differentiation with vector inputs (example 2 linear function)

Let consider $A \in \mathbb{R}^{n \times m}$ and $x \in \mathbb{R}^m$:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix}, \; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{6}$$

The matrix A is weights of a linear function

Differentiation with vector inputs (example 2 linear function)

Let consider $A \in \mathbb{R}^{n \times m}$ and $x \in \mathbb{R}^m$:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix}, \; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{6}$$

The matrix A is weights of a linear function

Let z_j be the j^{th} output of the linear function:

$$z_j = \sum_{k=1}^{m} a_{j,k} x_k$$

Differentiation with vector inputs (example 2 linear function)

Let consider $A \in \mathbb{R}^{n \times m}$ and $x \in \mathbb{R}^m$:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix}, \; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{6}$$

The matrix A is weights of a linear function

Let z_j be the j^{th} output of the linear function:

$$z_j = \sum_{k=1}^{m} a_{j,k} x_k$$

$$z = Ax = \begin{pmatrix} a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,m}x_m \\ a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,m}x_m \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \dots + a_{n,m}x_m \end{pmatrix}$$

Differentiation with vector inputs (example 2 linear function)

Gradient $\frac{\partial (Ax)_j}{\partial x_i}$

The partial derivative $\partial(Ax)$ for j^{th} component according to the i^{th} input is given by:

Differentiation with vector inputs (example 2 linear function)

Gradient $\frac{\partial (Ax)_j}{\partial x_i}$

The partial derivative $\partial(Ax)$ for j^{th} component according to the i^{th} input is given by:

$$\frac{\partial (Ax)_j}{\partial x_i} = \frac{\partial \sum_{k=1}^{m} a_{j,k} x_k}{\partial x_i} = a_{j,i}$$

Differentiation with vector inputs (example 2 linear function)

Gradient $\frac{\partial (Ax)_j}{\partial x_i}$

The partial derivative $\partial(Ax)$ for j^{th} component according to the i^{th} input is given by:

$$\frac{\partial (Ax)_j}{\partial x_i} = \frac{\partial \sum_{k=1}^{m} a_{j,k} x_k}{\partial x_i} = a_{j,i}$$

Gradient $\frac{\partial (Ax)}{\partial x_j}$

The partial derivative Ax according to the j^{th} input is given by:

Differentiation with vector inputs (example 2 linear function)

Gradient $\frac{\partial (Ax)_j}{\partial x_i}$

The partial derivative $\partial(Ax)$ for j^{th} component according to the i^{th} input is given by:

$$\frac{\partial (Ax)_j}{\partial x_i} = \frac{\partial \sum_{k=1}^{m} a_{j,k} x_k}{\partial x_i} = a_{j,i}$$

Gradient $\frac{\partial (Ax)}{\partial x_j}$

The partial derivative Ax according to the j^{th} input is given by:

$$\frac{\partial Ax}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial (Ax)_j}{\partial x_i} = \sum_{j=1}^{n} a_{j,i}$$

Differentiation with vector inputs (example 2 linear function)

Gradient $\frac{\partial (Ax)_j}{\partial x_i}$

The partial derivative $\partial(Ax)$ for j^{th} component according to the i^{th} input is given by:

$$\frac{\partial (Ax)_j}{\partial x_i} = \frac{\partial \sum_{k=1}^{m} a_{j,k} x_k}{\partial x_i} = a_{j,i}$$

Gradient $\frac{\partial (Ax)}{\partial x_j}$

The partial derivative Ax according to the j^{th} input is given by:

$$\frac{\partial Ax}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial (Ax)_j}{\partial x_i} = \sum_{j=1}^{n} a_{j,i}$$

The gradient according to inputs features

Consequently the vector according to the inputs is given by:

Differentiation with vector inputs (example 2 linear function)

Gradient $\frac{\partial (Ax)_j}{\partial x_i}$

The partial derivative $\partial(Ax)$ for j^{th} component according to the i^{th} input is given by:

$$\frac{\partial (Ax)_j}{\partial x_i} = \frac{\partial \sum_{k=1}^{m} a_{j,k} x_k}{\partial x_i} = a_{j,i}$$

Gradient $\frac{\partial (Ax)}{\partial x_j}$

The partial derivative Ax according to the j^{th} input is given by:

$$\frac{\partial Ax}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial (Ax)_j}{\partial x_i} = \sum_{j=1}^{n} a_{j,i}$$

The gradient according to inputs features

Consequently the vector according to the inputs is given by:

$$\nabla_x Ax = \begin{pmatrix} \sum_{j=1}^{n} a_{j,1} \\ \sum_{j=1}^{n} a_{j,2} \\ \vdots \\ \sum_{j=1}^{n} a_{j,m} \end{pmatrix} \tag{7}$$

Differentiation with vector inputs : Composition (chain rule)

Chain rule

Let $h : \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$ be two functions and $x \in \mathbb{R}^m, z \in \mathbb{R}^n, y \in \mathbb{R}$ be variables such that :

$$y = hog(x)$$

$$z = g(x)$$
$$y = h(z)$$

Differentiation with vector inputs : Composition (chain rule)

Chain rule

Let $h : \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$ be two functions and $x \in \mathbb{R}^m, z \in \mathbb{R}^n, y \in \mathbb{R}$ be variables such that :

$$y = hog(x)$$

$$z = g(x)$$
$$y = h(z)$$

For a given x, how an infinitesimal change of x_i impact y?

Differentiation with vector inputs : Composition (chain rule)

Chain rule

Let $h : \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$ be two functions and $x \in \mathbb{R}^m, z \in \mathbb{R}^n, y \in \mathbb{R}$ be variables such that :

$$y = hog(x) \qquad \begin{aligned} z &= g(x) \\ y &= h(z) \end{aligned}$$

For a given x, how an infinitesimal change of x_i impact y?

$$\frac{\partial y}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial y}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_i}$$

Differentiation with vector inputs : Composition (example 3)

Let $h : \mathbb{R}^n \times \mathbb{R}^k$ and $g : \mathbb{R}^m \times \mathbb{R}^n$ be two functions such that $g(x) = Ax$ and $A \in \mathbb{R}^{n \times m}$.

The gradient $\partial h(g(x))$ according to the j^{th} input is given by:

$$\frac{\partial h(g(x))}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial h(z)}{\partial z_j} . \frac{\partial z_j}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial h(z)}{\partial z_j} . a_{j,i}$$

With $z = Ax$

The gradient according to inputs features

Consequently the vector according to the inputs is given by:

$$\nabla_x Ax = \begin{pmatrix} \sum_{j=1}^{n} \frac{\partial h(z)}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_1} \\ \sum_{j=1}^{n} \frac{\partial h(z)}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_2} \\ \vdots \\ \sum_{j=1}^{n} \frac{\partial h(z)}{\partial z_j} \cdot \frac{\partial z_j}{\partial x_m} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n} \frac{\partial h(z)}{\partial z_j} . a_{j,1} \\ \sum_{j=1}^{n} \frac{\partial h(z)}{\partial z_j} . a_{j,2} \\ \vdots \\ \sum_{j=1}^{n} \frac{\partial h(z)}{\partial z_j} . a_{j,m} \end{pmatrix} \tag{8}$$

The Jacobian Matrix

Let f be a function $f : \mathbb{R}^m \to \mathbb{R}^n$ where its first-order derivative exists on $\mathbb{R}^m$. Then the jacobian matrice denoted by $J_x y \in \mathbb{R}^{n \times m}$ whose entry $J_x y_{(j,i)}$ is defined by $\frac{\partial y_j}{\partial x_i}$:

The Jacobian Matrix

Let f be a function $f : \mathbb{R}^m \to \mathbb{R}^n$ where its first-order derivative exists on $\mathbb{R}^m$. Then the jacobian matrice denoted by $J_x y \in \mathbb{R}^{n \times m}$ whose entry $J_x y_{(j,i)}$ is defined by $\frac{\partial y_j}{\partial x_i}$:

$$J_x y = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_m} \end{pmatrix}$$

Jacobian notation and chain rule

Let $h : \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$ be two functions and $x \in \mathbb{R}^m, z \in \mathbb{R}^n, y \in \mathbb{R}$ be variables such that :

$$y = hog(x)$$

$$z = g(x)$$
$$y = h(z)$$

Jacobian notation and chain rule

Let $h : \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$ be two functions and $x \in \mathbb{R}^m, z \in \mathbb{R}^n, y \in \mathbb{R}$ be variables such that :

$$y = hog(x)$$

$$z = g(x)$$
$$y = h(z)$$

Partial notation:

$$\frac{\partial y}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial y}{\partial z_j} . \frac{\partial z_j}{\partial x_i}$$

Jacobian notation:

$$\nabla_x y = J_x z^\intercal . \nabla_z y$$

$$\nabla_x y = J_x z^\intercal . \nabla_z y = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_2}{\partial x_1} & \cdots & \frac{\partial z_n}{\partial x_1} \\ \frac{\partial z_1}{\partial x_2} & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_1}{\partial x_m} & \frac{\partial z_2}{\partial x_m} & \cdots & \frac{\partial z_n}{\partial x_m} \end{pmatrix} . \begin{pmatrix} \frac{\partial y}{\partial z_1} \\ \frac{\partial y}{\partial z_2} \\ \vdots \\ \frac{\partial y}{\partial z_n} \end{pmatrix}$$

Jacobian notation and chain rule: Exemple 4

Let $h : \mathbb{R}^n \times \mathbb{R}^k$ and $g : \mathbb{R}^m \times \mathbb{R}^n$ be two functions such that $z = g(x) = Ax$ and $A \in \mathbb{R}^{n \times m}$.

Jacobian notation and chain rule: Exemple 4

Let $h : \mathbb{R}^n \times \mathbb{R}^k$ and $g : \mathbb{R}^m \times \mathbb{R}^n$ be two functions such that $z = g(x) = Ax$ and $A \in \mathbb{R}^{n \times m}$.

$$\nabla_x hog(x) = J_x z^\intercal . \nabla_z h(z) = \begin{pmatrix} a_{1,1} & a_{2,1} & \dots & a_{n,1} \\ a_{1,2} & a_{2,2} & \dots & a_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,m} & a_{2,m} & \dots & a_{n,m} \end{pmatrix} . \begin{pmatrix} \frac{\partial h(z)}{\partial z_1} \\ \frac{\partial h(z)}{\partial z_2} \\ \vdots \\ \frac{\partial h(z)}{\partial z_2} \end{pmatrix}$$

The Linear Function: gradient ∇_A

Let $\mathcal{L} : (\mathbb{R}^k, \mathbb{R}^k) \to \mathbb{R}$ and $g : \mathbb{R}^m \times \mathbb{R}^n$ be two functions such that $z = g(x) = Ax$ and $A \in \mathbb{R}^{n \times m}$

$$\frac{\partial \mathcal{L}(f(z), y)}{\partial a_{i,k}} = \sum_{j=1}^{n} \frac{\partial \mathcal{L}(f(z), y)}{\partial z_j} \cdot \frac{\partial z_j}{\partial a_{i,k}}$$

The Linear Function: gradient ∇_A

Let $\mathcal{L} : (\mathbb{R}^k, \mathbb{R}^k) \to \mathbb{R}$ and $g : \mathbb{R}^m \times \mathbb{R}^n$ be two functions such that $z = g(x) = Ax$ and $A \in \mathbb{R}^{n \times m}$

$$\frac{\partial \mathcal{L}(f(z), y)}{\partial a_{i,k}} = \sum_{j=1}^{n} \frac{\partial \mathcal{L}(f(z), y)}{\partial z_j} \cdot \frac{\partial z_j}{\partial a_{i,k}}$$

$$\frac{\partial z_j}{\partial a_{i,k}} = \frac{\partial \sum_{l=1}^{m} a_{i,l} x_l}{a_{i,k}}$$

$$\frac{\partial z_j}{\partial A_{i,k}} = \begin{cases} x_k \text{ if } i = j \\ 0 \text{ else} \end{cases}$$

The Linear Function: gradient ∇_A

Let $\mathcal{L} : (\mathbb{R}^k, \mathbb{R}^k) \to \mathbb{R}$ and $g : \mathbb{R}^m \times \mathbb{R}^n$ be two functions such that $z = g(x) = Ax$ and $A \in \mathbb{R}^{n \times m}$

$$\frac{\partial \mathcal{L}(f(z), y)}{\partial a_{i,k}} = \sum_{j=1}^{n} \frac{\partial \mathcal{L}(f(z), y)}{\partial z_j} \cdot \frac{\partial z_j}{\partial a_{i,k}}$$

$$\frac{\partial z_j}{\partial a_{i,k}} = \frac{\partial \sum_{l=1}^{m} a_{i,l} x_l}{a_{i,k}}$$

$$\frac{\partial z_j}{\partial A_{i,k}} = \begin{cases} x_k \text{ if } i = j \\ 0 \text{ else} \end{cases}$$

$$\nabla_A \mathcal{L} = \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial z_1}.x_1 & \frac{\partial \mathcal{L}}{\partial z_1}.x_2 & \dots & \frac{\partial \mathcal{L}}{\partial z_1}.x_m \\ \frac{\partial \mathcal{L}}{\partial z_2}.x_1 & \frac{\partial \mathcal{L}}{\partial z_2}.x_2 & \dots & \frac{\partial \mathcal{L}}{\partial z_2}.x_m \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial z_n}.x_1 & \frac{\partial \mathcal{L}}{\partial z_n}.x_2 & \dots & \frac{\partial \mathcal{L}}{\partial z_n}.x_m \end{pmatrix} = \nabla_z \mathcal{L}.x^\intercal$$

(Outer product)

Backward Pass With Jacobian for $f(Ax^\intercal)$?

Let g be

$$g\colon \mathbb{R}^{m\times n} \to \mathbb{R}^n$$
$$A \mapsto z$$

Notice that the x_i are the parameters in this configuration

$$\begin{pmatrix} a_{1,1} \\ a_{1,2} \\ \vdots \\ a_{1,m} \\ a_{2,1} \\ \vdots \\ a_{n,m} \end{pmatrix} \mapsto \begin{pmatrix} a_{1,1}x_1 & + & a_{1,2}x_2 & \dots & + & a_{1,m}x_m = z_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{2,1}x_1 & + & a_{2,2}x_2 & \dots & + & a_{2,m}x_m = z_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n,1}x_1 & + & a_{n,2}x_2 & \dots & + & a_{n,m}x_m = z_n \end{pmatrix}$$

What form has the jacobian $J_A z^\intercal$?

Backward Pass With Jacobian for $f(Ax^\intercal)$?

What form has the jacobian $J_A z^\intercal$?

$$J_A z = \begin{pmatrix} \frac{\partial z_1}{\partial a_{a_{1,1}}} & \frac{\partial z_1}{\partial a_{a_{1,2}}} & \cdots & \frac{\partial z_1}{\partial a_{a_{1,m}}} & \frac{\partial z_1}{\partial a_{a_{2,1}}} & \frac{\partial z_1}{\partial a_{a_{2,2}}} & \cdots & \frac{\partial z_1}{\partial a_{a_{n,m}}} \\ \frac{\partial z_2}{\partial a_{a_{1,1}}} & \frac{\partial z_2}{\partial a_{a_{1,2}}} & \cdots & \frac{\partial z_2}{\partial a_{a_{1,m}}} & \frac{\partial z_2}{\partial a_{a_{2,1}}} & \frac{\partial z_2}{\partial a_{a_{2,2}}} & \cdots & \frac{\partial z_2}{\partial a_{a_{n,m}}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_m}{\partial a_{a_{1,1}}} & \frac{\partial z_m}{\partial a_{a_{1,2}}} & \cdots & \frac{\partial z_m}{\partial a_{a_{1,m}}} & \frac{\partial z_m}{\partial a_{a_{2,1}}} & \frac{\partial z_m}{\partial a_{a_{2,2}}} & \cdots & \frac{\partial z_m}{\partial a_{a_{n,m}}} \end{pmatrix} \tag{9}$$

Backward Pass With Jacobian for $f(Ax^\intercal)$?

What form has the jacobian $J_A z^\intercal$?

$$J_A z^\intercal = \begin{pmatrix} \frac{\partial z_1}{\partial a_{a_{1,1}}} & \frac{\partial z_2}{\partial a_{a_{1,1}}} & \cdots & \frac{\partial z_n}{\partial a_{a_{1,1}}} \\ \frac{\partial z_1}{\partial a_{a_{1,2}}} & \frac{\partial z_2}{\partial a_{a_{1,2}}} & \cdots & \frac{\partial z_n}{\partial a_{a_{1,2}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_1}{\partial a_{a_{1,m}}} & \frac{\partial z_2}{\partial a_{a_{1,m}}} & \cdots & \frac{\partial z_n}{\partial a_{a_{1,m}}} \\ \frac{\partial z_1}{\partial a_{a_{2,1}}} & \frac{\partial z_2}{\partial a_{a_{2,1}}} & \cdots & \frac{\partial z_n}{\partial a_{a_{2,1}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_1}{\partial a_{a_{2,m}}} & \frac{\partial z_2}{\partial a_{a_{2,m}}} & \cdots & \frac{\partial z_n}{\partial a_{a_{2,m}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_1}{\partial a_{a_{n,m}}} & \frac{\partial z_2}{\partial a_{a_{n,m}}} & \cdots & \frac{\partial z_n}{\partial a_{a_{n,m}}} \end{pmatrix} = \begin{pmatrix} x_1 & 0 & 0 & \cdots & 0 \\ x_2 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_m & 0 & 0 & \cdots & 0 \\ 0 & x_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & x_m & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & x_m \end{pmatrix}$$

Backward Pass With Jacobian for $f(Ax^\intercal)$?

What form has the jacobian $J_A z^\intercal$?

$$J_A z^\intercal . \nabla_z \mathcal{L} = \begin{pmatrix} x_1 & 0 & 0 & \dots & 0 \\ x_2 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_m & 0 & 0 & \dots & 0 \\ 0 & x_1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & x_m & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & x_m \end{pmatrix} . \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial z_1} \\ \frac{\partial \mathcal{L}}{\partial z_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial z_n} \end{pmatrix} = \begin{pmatrix} x_1 \frac{\partial \mathcal{L}}{\partial z_1} = \frac{\partial \mathcal{L}}{\partial a_{1,1}} \rightarrow \text{line 1} \\ x_2 \frac{\partial \mathcal{L}}{\partial z_1} = \frac{\partial \mathcal{L}}{\partial a_{1,2}} \rightarrow \text{line 1} \\ \vdots \\ x_m \frac{\partial \mathcal{L}}{\partial z_1} = \frac{\partial \mathcal{L}}{\partial a_{1,m}} \rightarrow \text{line 1} \\ x_1 \frac{\partial \mathcal{L}}{\partial z_2} = \frac{\partial \mathcal{L}}{\partial a_{2,1}} \rightarrow \text{line 2} \\ \vdots \\ x_m \frac{\partial \mathcal{L}}{\partial z_2} = \frac{\partial \mathcal{L}}{\partial a_{2,m}} \rightarrow \text{line 2} \\ \vdots \\ x_m \frac{\partial \mathcal{L}}{\partial z_n} = \frac{\partial \mathcal{L}}{\partial a_{n,m}} \rightarrow \text{line n} \end{pmatrix}$$

Forward and backward pass, general formula

Forward pass

$z^{(1)} = f_{\Theta_1}(x)$

Forward and backward pass, general formula

Forward pass

$$z^{(1)} = f_{\Theta_1}(x)$$
$$\downarrow$$
$$z^{(2)} = f_{\Theta_2}(z^{(1)})$$

Forward and backward pass, general formula

Forward pass

$$
\begin{aligned}
z^{(1)} &= f_{\Theta_1}(x) \\
&\downarrow \\
z^{(2)} &= f_{\Theta_2}(z^{(1)}) \\
&\downarrow \\
z^{(3)} &= f_{\Theta_3}(z^{(2)})
\end{aligned}
$$

Forward and backward pass, general formula

Forward pass

$$
\begin{aligned}
z^{(1)} &= f_{\Theta_1}(x) \\
&\downarrow \\
z^{(2)} &= f_{\Theta_2}(z^{(1)}) \\
&\downarrow \\
z^{(3)} &= f_{\Theta_3}(z^{(2)}) \\
&\downarrow \\
z^{(4)} &= f_{\Theta_4}(z^{(3)})
\end{aligned}
$$

Forward and backward pass, general formula

Forward pass

$$
\begin{aligned}
z^{(1)} &= f_{\Theta_1}(x) \\
&\downarrow \\
z^{(2)} &= f_{\Theta_2}(z^{(1)}) \\
&\downarrow \\
z^{(3)} &= f_{\Theta_3}(z^{(2)}) \\
&\downarrow \\
z^{(4)} &= f_{\Theta_4}(z^{(3)}) \\
&\downarrow \\
y^{(4)} &= f_{\Theta_5}(z^{(4)})
\end{aligned}
$$

Forward and backward pass, general formula

Forward pass

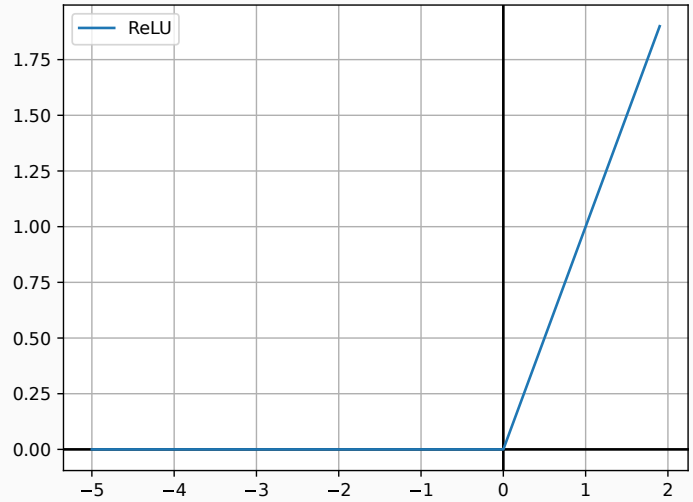
$$z^{(1)} = f_{\Theta_1}(x)$$
$$\downarrow$$
$$z^{(2)} = f_{\Theta_2}(z^{(1)})$$
$$\downarrow$$
$$z^{(3)} = f_{\Theta_3}(z^{(2)})$$
$$\downarrow$$
$$z^{(4)} = f_{\Theta_4}(z^{(3)})$$
$$\downarrow$$
$$y^{(4)} = f_{\Theta_5}(z^{(4)})$$

Backward pass

$$\nabla_{z^{(1)}} y = J_{z^{(1)}} z^{(2)\intercal} . \nabla_{z^{(2)}} y \quad \uparrow \quad \nabla_{\Theta^{(1)}} y = J_{\Theta^{(1)}} z^{(1)\intercal} . \nabla_{z^{(1)}} y$$
$$\nabla_{z^{(2)}} y = J_{z^{(2)}} z^{(3)\intercal} . \nabla_{z^{(3)}} y \quad \uparrow \quad \nabla_{\Theta^{(2)}} y = J_{\Theta^{(2)}} z^{(2)\intercal} . \nabla_{z^{(2)}} y$$
$$\nabla_{z^{(3)}} y = J_{z^{(3)}} z^{(4)\intercal} . \nabla_{z^{(4)}} y \quad \uparrow \quad \nabla_{\Theta^{(3)}} y = J_{\Theta^{(3)}} z^{(3)\intercal} . \nabla_{z^{(3)}} y$$
$$\uparrow \quad \nabla_{\Theta^{(4)}} y = J_{\Theta^{(4)}} z^{(4)\intercal} . \nabla_{z^{(4)}} y$$
$$\nabla_{z^{(4)}} y \qquad \nabla_{\Theta^{(5)}} y$$

Rectified Linear Unit (RELU):

$$\sigma(x) = \begin{cases} x & \text{if x } \geq 0 \\ 0 & \text{if x } < 0 \end{cases}$$

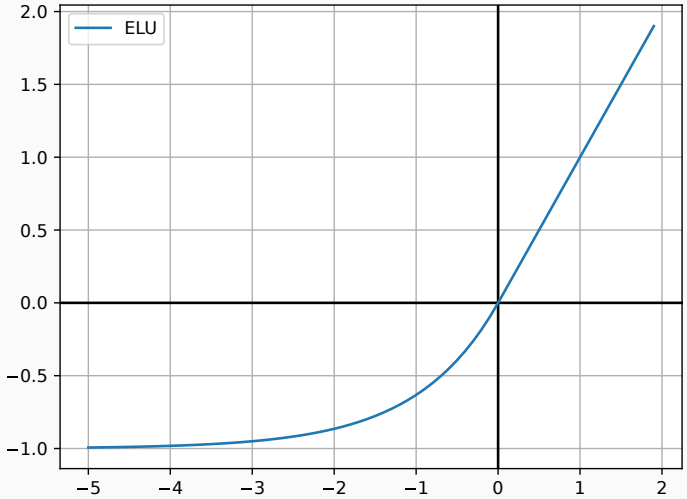


Derivative of the Rectified Linear Unit:

$$\sigma'(x) = \begin{cases} 1 & \text{if x } \geq 0 \\ 0 & \text{if x } < 0 \end{cases}$$

Exponential Linear Unit (ELU) :

$$\sigma(x) = \begin{cases} x & \text{if x } \geq 0 \\ e^x - 1 & \text{if x } \leq 0 \end{cases}$$

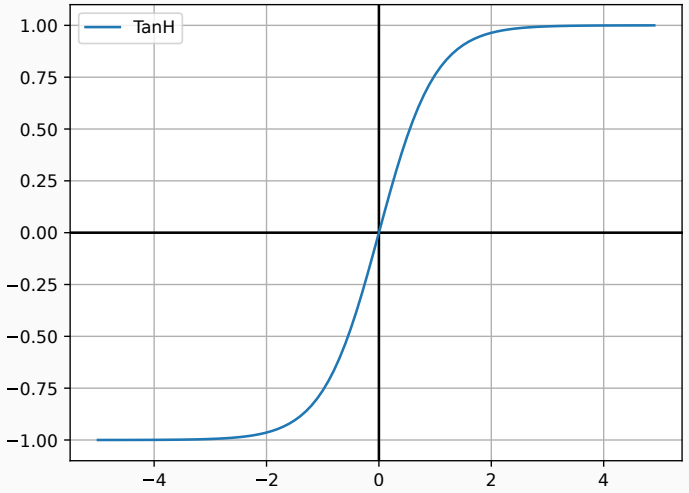


Derivative:

$$\sigma'(x) = \begin{cases} 1 & \text{if x } \geq 0 \\ e^x & \text{if x } < 0 \end{cases}$$

Hyperbolic tangent (TanH):

$$\sigma : \mathbb{R} \longrightarrow]-1, 1[$$
$$x \longmapsto \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Derivative of the Hyperbolic Tangent :

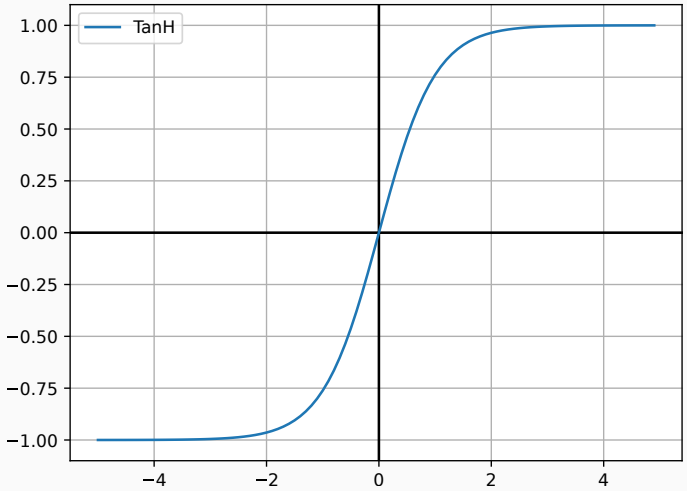
Let $u(x) = e^x - e^{-x}$ and $v(x) = e^x + e^{-x}$
Hence:

- $u'(x) = e^x + e^{-x} = v(x)$
- $v'(x) = e^x - e^{-x} = u(x)$

What is $\sigma'(x)$

Hyperbolic tangent (TanH):

$$\sigma : \mathbb{R} \longrightarrow]-1, 1[$$
$$x \longmapsto \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Derivative of the Hyperbolic Tangent :

Let $u(x) = e^x - e^{-x}$ and $v(x) = e^x + e^{-x}$
Hence:

- $u'(x) = e^x + e^{-x} = v(x)$
- $v'(x) = e^x - e^{-x} = u(x)$

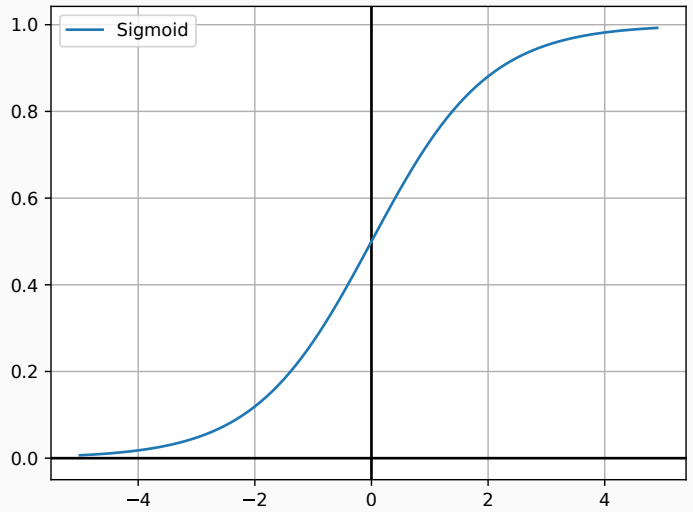
What is $\sigma'(x)$

$$\begin{aligned}\sigma'(x) &= \frac{u'(x)v(x) - u(x)v'(x)}{v(x)^2} \\ &= \frac{v(x)v(x) - u(x)u(x)}{v(x)^2} \\ &= \frac{v(x)^2}{v(x)^2} - \frac{u(x)^2}{v(x)^2} \\ &= 1 - \sigma(x)^2\end{aligned}$$

Neural Networks: Sigmoid activation function

Sigmoid activation (logistic function):

$$\sigma : \mathbb{R} \longrightarrow]-1, 1[$$
$$x \longmapsto \frac{1}{1+e^{-x}}$$



Derivative of the sigmoid function :

Let $u(x) = 1 + e^{-x}$

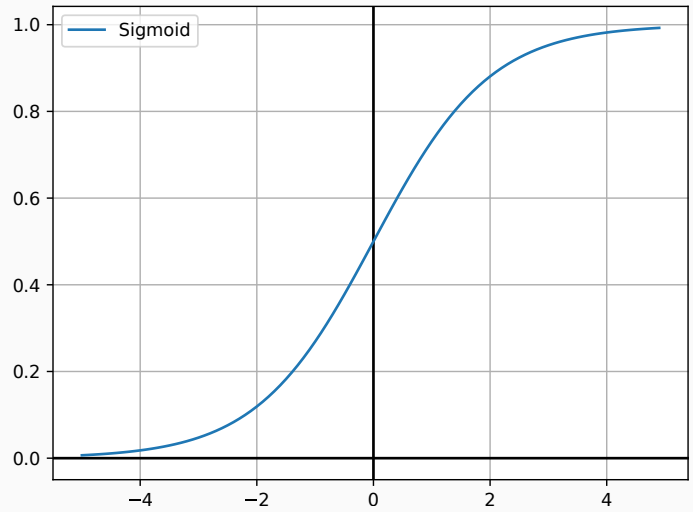
- $\sigma(x) = \frac{1}{u}$
- $u'(x) = -e^{-x}$

What is $\sigma'(x)$

Neural Networks: Sigmoid activation function

Sigmoid activation (logistic function):

$$\sigma : \mathbb{R} \longrightarrow]-1, 1[$$
$$x \longmapsto \frac{1}{1+e^{-x}}$$



Derivative of the sigmoid function :

Let $u(x) = 1 + e^{-x}$

- $\sigma(x) = \frac{1}{u}$
- $u'(x) = -e^{-x}$

What is $\sigma'(x)$

$$\begin{aligned}
\sigma'(x) &= \frac{-u'(x)}{u(x)^2} \\
&= \frac{e^{-x}}{u(x)^2} \\
&= \frac{1}{u(x)} \frac{e^{-x}+1-1}{u(x)} \\
&= \sigma(x) \frac{(1+e^{-x})-1}{u(x)} \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}$$

Back to Computational Graph

ComputationalGraphNode

- Value
- Graadient
- Backward computation operations
- Pointers to previous nodes

ParameterNode
A **ComputationalGraphNode** !!

- Persistent values (weights)

Back to Computational Graph

We know gow to decompose gradient using gradient of next Computation Graph Node Let consider the operation :

$$z^{i+1} = Az^{i}$$

with z^{j} is a **ComputationGraphNode** !!!

Function Linear Forward pass

inputs : node $z^{(i)}$ and node A

- Create the node $z^{(i+1)} = ComputationGraphNode()$
- Compute forward value $z^{(i+1)}.value = A(z^{(i)}.value)$
- Store previous nodes $z^{(i+1)}.input\_nodes = A, z^{(i)}$
- Store backward function $z^{(i+1)}.grad\_operation = linear\_gradient$

Return : node $z^{(i+1)}$

Back to Computational Graph

We know gow to decompose gradient using gradient of next Computation Graph Node Let consider the operation :

$$z^{i+1} = Az^{i}$$

with z^{j} is a **ComputationGraphNode** !!!

Function Linear Backward pass

inputs : node $z^{(i+1)}$ and $\nabla_{z^{(i+1)}}\mathcal{L}$

- Store and compute input gradient :

$$\nabla_{A}\mathcal{L}, \nabla_{z^{(i)}}\mathcal{L} \leftarrow z^{(i+1)}.grad\_operation(z^{(i+1)}.input\_nodes, \nabla_{z^{(i+1)}}\mathcal{L})$$

- Call $backward(z^{(i+1)}.A, \nabla_{A}\mathcal{L})$ and $backward(z^{(i+1)}.z^{(i)}, \nabla_{z^{(i)}}\mathcal{L})$

Next session

- Optimization and regularisation methods (short lecture)
- Getting started with numpy (lab exercise)
- Implement neural network with numpy (lab exercise)
- Implement graph differentation library (lab exercise)