T4: Backpropagation algorithm

Pablo Arias Martínez - ENS Paris-Saclay, UPF
pablo.arias@upf.edu

October 13, 2022

Optimization and inference techniques for Computer Vision

Neural networks

A deep neural network is a complicated function that results from stacking many simple ones (layers).

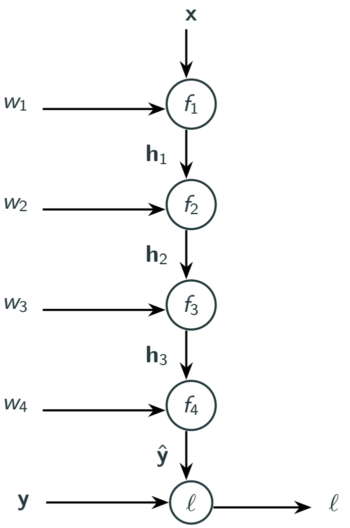
For example, a network with 4 "layers" $f_1, \ldots, f_4$. Each layer has parameters w_i. We denote by $\boldsymbol{\theta} = (w_1, \ldots, w_4)$ the: vector with all paramers:

$$\hat{\mathbf{y}} = \mathcal{F}(\mathbf{x}, \boldsymbol{\theta}) = f_4(w_4, f_3(w_3, f_2(w_2, f_1(w_1, \mathbf{x})))).$$

To train, we compute a loss $\ell = \ell(\hat{\mathbf{y}}, \mathbf{y})$ penalizing the error between the predicted $\hat{\mathbf{y}}$ and the desired $\mathbf{y}$.

We want to compute the gradient of the loss with respect to the parameters:

$$\nabla_\theta \ell(\mathcal{F}_\theta(\mathbf{x}), \mathbf{y}) = \left(\frac{\partial \ell}{\partial w_1}, \ldots, \frac{\partial \ell}{\partial w_4} \right)^T.$$



The backpropagation algorithm is an algorithm for computing derivatives of a function.

It is used in machine learning, for computing the gradient of the loss with respect to the parameters of a neural network,

$$\nabla_\theta \ell(\mathcal{F}_\theta(\mathbf{x}_i), \mathbf{y}_i)$$

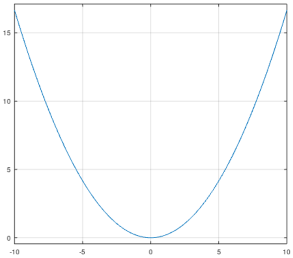
but in fact it can be used for computing derivatives of any function.

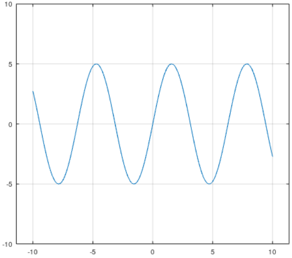
Chain rule

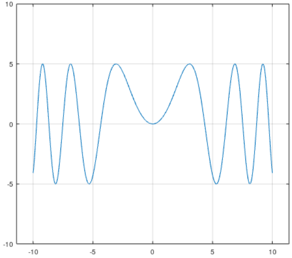
Derivative of a composition of functions. Let $f, g : \mathbb{R} \to \mathbb{R}$ two differentiable functions. We define

$$z = h(x) = f(g(x)).$$

$$x \longrightarrow g \xrightarrow{y = g(x)} f \longrightarrow z$$





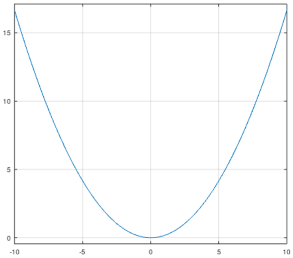


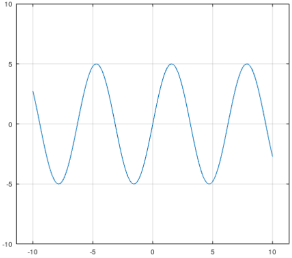
Chain rule

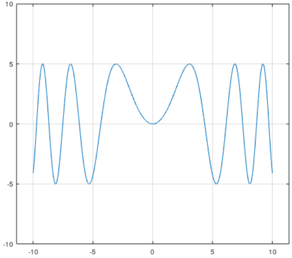
Derivative of a composition of functions. Let $f, g : \mathbb{R} \to \mathbb{R}$ two differentiable functions. We define

$$z = h(x) = f(g(x)).$$

$$x \longrightarrow g \xrightarrow{y = g(x)} f \longrightarrow z$$







The chain rule tells us how to compute the derivative of the composed function h:

$$h'(x) = f'(g(x))g'(x).$$

Chain rule - Leibnitz notation

Leibnitz notation for derivatives. For $y = g(x)$ we denote its derivative $g'(x)$ as $\frac{dy}{dx}(x)$.

This notation is inspired by the definition of derivative as the limit of a quotient:

$$g'(x) = \frac{dy}{dx}(x) = \lim_{\Delta x \to 0} \frac{g(x + \Delta x) - g(x)}{\Delta x}.$$

Chain rule - Leibnitz notation

Leibnitz notation for derivatives. For $y = g(x)$ we denote its derivative $g'(x)$ as $\dfrac{dy}{dx}(x)$.

This notation is inspired by the definition of derivative as the limit of a quotient:

$$g'(x) = \frac{dy}{dx}(x) = \lim_{\Delta x \to 0} \frac{g(x + \Delta x) - g(x)}{\Delta x}.$$

Remember our function composition:

$$x \longrightarrow g \xrightarrow{\;y\;} f \longrightarrow z$$

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

We can express the chain rule using Leibnitz notation:

$$h'(x) = f'(g(x))g'(x) \quad \Longrightarrow \quad \frac{dz}{dx}(x) = \frac{dz}{dy}(y(x))\frac{dy}{dx}(x) \quad \text{or} \quad \left.\frac{dz}{dx}\right|_{x} = \left.\frac{dz}{dy}\right|_{y(x)} \left.\frac{dy}{dx}\right|_{x}$$

We usualy simplify notation by removing the arguments of the derivatives: $\dfrac{dz}{dx} = \dfrac{dz}{dy}\dfrac{dy}{dx}$.

But keep in mind that each one needs to be evaluated in correct values!

Chain rule - more compositions!

If we compose several functions we use the chain rule several times. For example:

$$z = h(x) = f_3(f_2(f_1(x))).$$

$$x \longrightarrow \boxed{f_1} \xrightarrow{y_1} \boxed{f_2} \xrightarrow{y_2} \boxed{f_3} \longrightarrow z$$

$$y_1 = f_1(x), \quad y_2 = f_2(y_1), \quad z = f_3(y_2) = h(x)$$

By applying the chain rule two times we obtain:

$$h'(x) = f_3'(f_2(f_1(x)))f_2'(f_1(x))f_1'(x) \quad \text{or} \quad \frac{dz}{dx}(x) = \frac{dz}{dy_2}(y_2(y_1(x)))\frac{dy_2}{dy_1}(y_1(x))\frac{dy_1}{dx}(x)$$

If we omit the arguments with Leibnitz notation: $\dfrac{dz}{dx} = \dfrac{dz}{dy_2}\dfrac{dy_2}{dy_1}\dfrac{dy_1}{dx}$.

Using Leibnitz notation we can work with derivatives as if derivatives were quotients. (But remember they are not quotients!)

x

We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

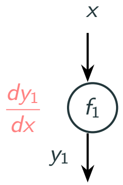
$$z = h(x) = f_3(f_2(f_1(x))).$$

We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.

We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

$$z = h(x) = f_3(f_2(f_1(x))).$$

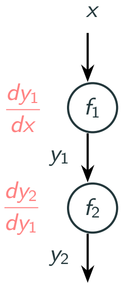
We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.



We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

$$z = h(x) = f_3(f_2(f_1(x))).$$

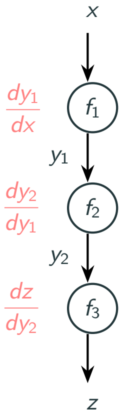
We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.



We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

$$z = h(x) = f_3(f_2(f_1(x))).$$

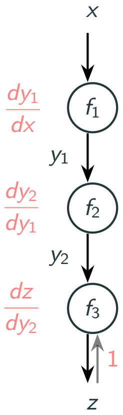
We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.



We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

$$z = h(x) = f_3(f_2(f_1(x))).$$

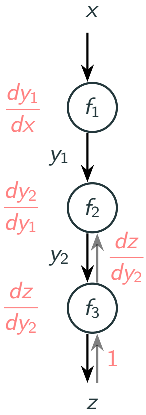
We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.



We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

$$z = h(x) = f_3(f_2(f_1(x))).$$

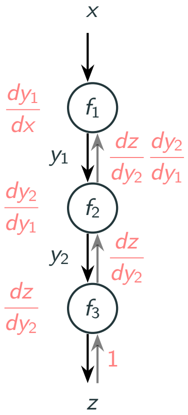
We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.



We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

$$z = h(x) = f_3(f_2(f_1(x))).$$

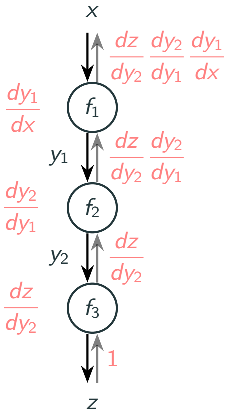
We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.



We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

$$z = h(x) = f_3(f_2(f_1(x))).$$

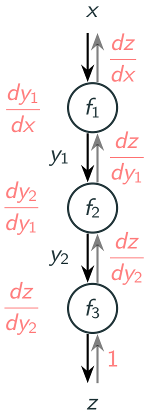
We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.



We consider our function as a directed graph (the **computational graph**). Nodes in this graph are functions. Two functions are connected if the outputs of one of the functions are inputs to the other.

$$z = h(x) = f_3(f_2(f_1(x))).$$

We pass two times through the graph: the **forward pass** evaluates the function, and the **backward pass** computes the derivative using the chain rule.



Towards the backpropogation algorithm

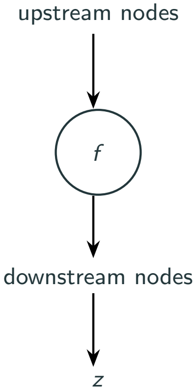
Let us zoom now on a single node. The action of each node is local: it only depends of its input and its output. Let us denote by i its input and by o its output.

Forward pass:

- wait for input i from upstream node,
- compute ouput $o = f(i)$,
- compute derivative $\frac{do}{di}$ and store them,
- pass outputs o to downstream nodes.

Backward pass:

- wait for derivative $\frac{dz}{do}$ from downstream node
- using the stored derivative, compute derivative $\frac{dz}{di}$ with respect to the node input,
- pass derivatives to upstream nodes.



Towards the backpropogation algorithm

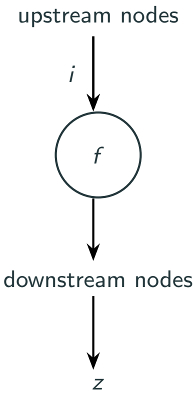
Let us zoom now on a single node. The action of each node is local: it only depends of its input and its output. Let us denote by i its input and by o its output.

Forward pass:

- wait for input i from upstream node,
- compute ouput $o = f(i)$,
- compute derivative $\frac{do}{di}$ and store them,
- pass outputs o to downstream nodes.

Backward pass:

- wait for derivative $\frac{dz}{do}$ from downstream node
- using the stored derivative, compute derivative $\frac{dz}{di}$ with respect to the node input,
- pass derivatives to upstream nodes.



Towards the backpropogation algorithm

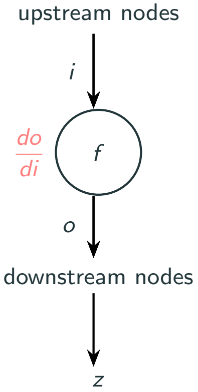
Let us zoom now on a single node. The action of each node is local: it only depends of its input and its output. Let us denote by i its input and by o its output.

Forward pass:

- wait for input i from upstream node,
- compute ouput $o = f(i)$,
- compute derivative $\frac{do}{di}$ and store them,
- pass outputs o to downstream nodes.

Backward pass:

- wait for derivative $\frac{dz}{do}$ from downstream node
- using the stored derivative, compute derivative $\frac{dz}{di}$ with respect to the node input,
- pass derivatives to upstream nodes.



Towards the backpropogation algorithm

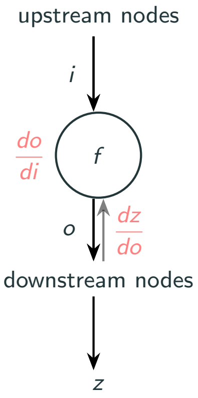
Let us zoom now on a single node. The action of each node is local: it only depends of its input and its output. Let us denote by i its input and by o its output.

Forward pass:

- wait for input i from upstream node,
- compute ouput $o = f(i)$,
- compute derivative $\frac{do}{di}$ and store them,
- pass outputs o to downstream nodes.

Backward pass:

- wait for derivative $\frac{dz}{do}$ from downstream node
- using the stored derivative, compute derivative $\frac{dz}{di}$ with respect to the node input,
- pass derivatives to upstream nodes.



Towards the backpropogation algorithm

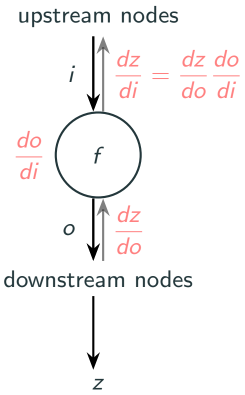
Let us zoom now on a single node. The action of each node is local: it only depends of its input and its output. Let us denote by i its input and by o its output.

Forward pass:

- wait for input i from upstream node,
- compute ouput $o = f(i)$,
- compute derivative $\frac{do}{di}$ and store them,
- pass outputs o to downstream nodes.

Backward pass:

- wait for derivative $\frac{dz}{do}$ from downstream node
- using the stored derivative, compute derivative $\frac{dz}{di}$ with respect to the node input,
- pass derivatives to upstream nodes.

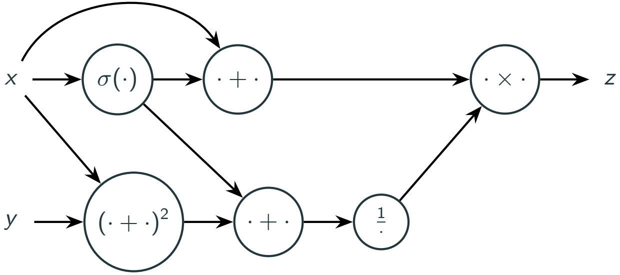


Computational graphs can become complicated!

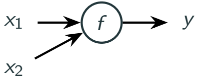
Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x + y)^2}$

Computational graphs can become complicated!

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x + y)^2}$



Computational graphs - functions with multiple inputs/outputs

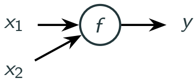


Multiple inputs: $z = h(f(x_1, x_2)), \quad y = f(x_1, x_2).$

We need to compute the partial derivatives with respect to the inputs:

$$\frac{dz}{dx_1} = \frac{dz}{dy}\frac{dy}{dx_1}, \quad \frac{dz}{dx_2} = \frac{dz}{dy}\frac{dy}{dx_2}$$

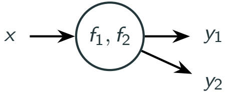
Computational graphs - functions with multiple inputs/outputs



Multiple inputs: $z = h(f(x_1, x_2)), \quad y = f(x_1, x_2).$

We need to compute the partial derivatives with respect to the inputs:

$$\frac{dz}{dx_1} = \frac{dz}{dy}\frac{dy}{dx_1}, \quad \frac{dz}{dx_2} = \frac{dz}{dy}\frac{dy}{dx_2}$$

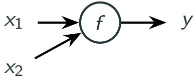


Multiple outputs: $z = h(f_1(x), f_2(x)), \quad y_1 = f_1(x), y_2 = f_2(x).$

Add the derivatives of each output with respect to the input:

$$\frac{dz}{dx} = \frac{dz}{dy_1}\frac{dy_1}{dx} + \frac{dz}{dy_2}\frac{dy_2}{dx}$$

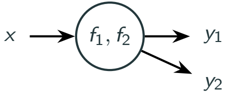
Computational graphs - functions with multiple inputs/outputs



Multiple inputs: $z = h(f(x_1, x_2)), \quad y = f(x_1, x_2)$.

We need to compute the partial derivatives with respect to the inputs:

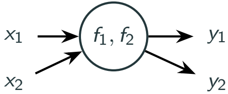
$$\frac{dz}{dx_1} = \frac{dz}{dy}\frac{dy}{dx_1}, \qquad \frac{dz}{dx_2} = \frac{dz}{dy}\frac{dy}{dx_2}$$



Multiple outputs: $z = h(f_1(x), f_2(x)), \quad y_1 = f_1(x), y_2 = f_2(x)$.

Add the derivatives of each output with respect to the input:

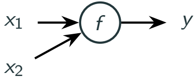
$$\frac{dz}{dx} = \frac{dz}{dy_1}\frac{dy_1}{dx} + \frac{dz}{dy_2}\frac{dy_2}{dx}$$



Multiple inputs & outputs: $z = h(\overbrace{f_1(x_1, x_2)}^{y_1}, \overbrace{f_2(x_1, x_2)}^{y_2})$.

$$\frac{dz}{dx_i} = \frac{dz}{dy_1}\frac{dy_1}{dx_i} + \frac{dz}{dy_2}\frac{dy_2}{dx_i}$$

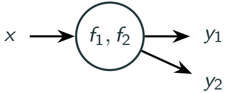
Computational graphs - functions with multiple inputs/outputs



Multiple inputs: $z = h(f(x_1, x_2)), \quad y = f(x_1, x_2).$

We need to compute the partial derivatives with respect to the inputs:

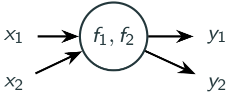
$$\frac{dz}{dx_1}(x_1, x_2) = \frac{dz}{dy}(y(x_1, x_2))\frac{dy}{dx_1}(x_1, x_2), \quad \frac{dz}{dx_2}(x_1, x_2) = \frac{dz}{dy}(y(x_1, x_2))\frac{dy}{dx_2}(x_1, x_2)$$



Multiple outputs: $z = h(f_1(x), f_2(x)), \quad y_1 = f_1(x), y_2 = f_2(x).$

Add the derivatives of each output with respect to the input:

$$\frac{dz}{dx}(x) = \frac{dz}{dy_1}(y_1(x), y_2(x))\frac{dy_1}{dx}(x) + \frac{dz}{dy_2}(y_1(x), y_2(x))\frac{dy_2}{dx}(y_2(x))$$

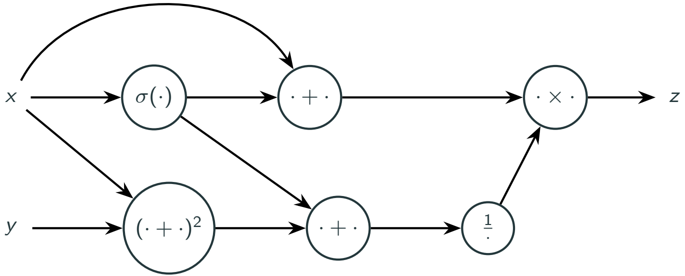


Multiple inputs & outputs: $z = h(\overbrace{f_1(x_1, x_2)}^{y_1}, \overbrace{f_2(x_1, x_2)}^{y_2}).$

$$\frac{dz}{dx_i}(x_1, x_2) = \frac{dz}{dy_1}(y_1(x_1, x_2), y_2(x_1, x_2))\frac{dy_1}{dx_i}(x_1, x_2) + \frac{dz}{dy_2}(y_1(x_1, x_2), y_2(x_1, x_2))\frac{dy_2}{dx_i}(y_2(x_1, x_2))$$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

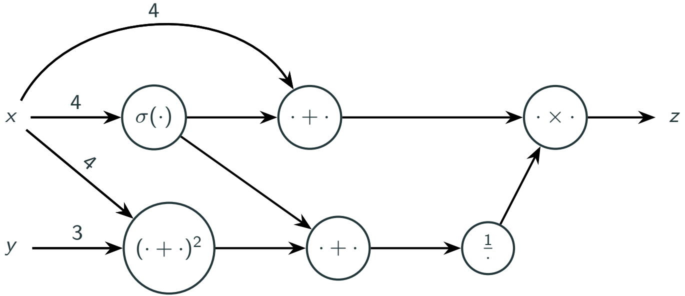


Forward pass: $x = 4, y = 3$

node	inputs i	outputs o	derivatives

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

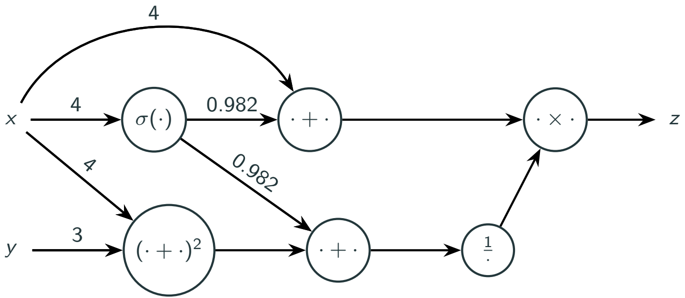


Forward pass: $x = 4, y = 3$

node	inputs i	outputs o	derivatives

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

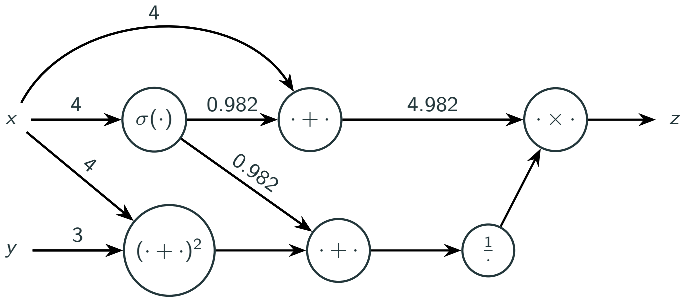


Forward pass: $x = 4, y = 3$

node	inputs i	outputs o	derivatives
$\sigma(\cdot)$	$i = 4$	$o_1, o_2 = \sigma(i) = 0.982$	$\frac{do_1}{di} = \frac{do_2}{di} = 0.0177$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

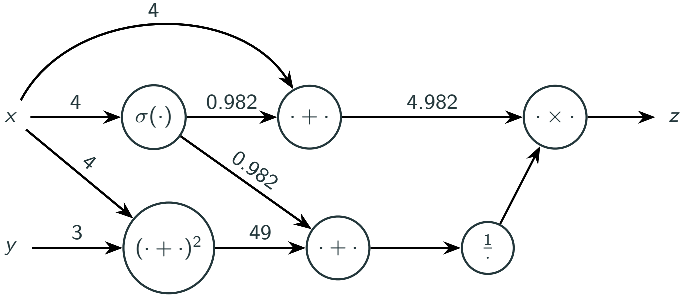


Forward pass: $x = 4, y = 3$

node	inputs i	outputs o	derivatives
$\sigma(\cdot)$	$i = 4$	$o_1, o_2 = \sigma(i) = 0.982$	$\frac{do_1}{di} = \frac{do_2}{di} = 0.0177$
$(\cdot + \cdot) \uparrow$	$i_1 = 4, i_2 = 0.982$	$o = (i_1 + i_2) = 4.982$	$\frac{do}{di_1} = 1, \frac{do}{di_2} = 1$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

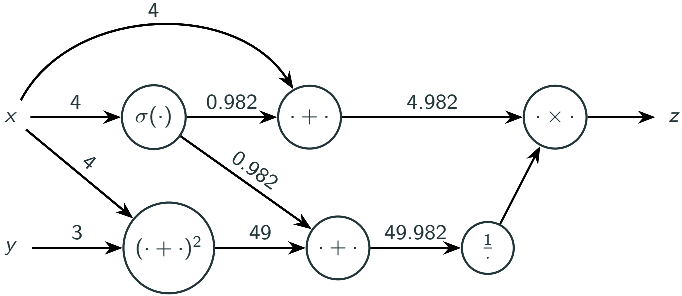


Forward pass: $x = 4, y = 3$

node	inputs i	outputs o	derivatives
$\sigma(\cdot)$	$i = 4$	$o_1, o_2 = \sigma(i) = 0.982$	$\frac{do_1}{di} = \frac{do_2}{di} = 0.0177$
$(\cdot + \cdot) \uparrow$	$i_1 = 4, i_2 = 0.982$	$o = (i_1 + i_2) = 4.982$	$\frac{do}{di_1} = 1, \frac{do}{di_2} = 1$
$(\cdot + \cdot)^2$	$i_1 = 4, i_2 = 3$	$o = (i_1 + i_2)^2 = 49$	$\frac{do}{di_1} = \frac{do}{di_2} = 2(i_1 + i_2) = 14$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x + y)^2}$

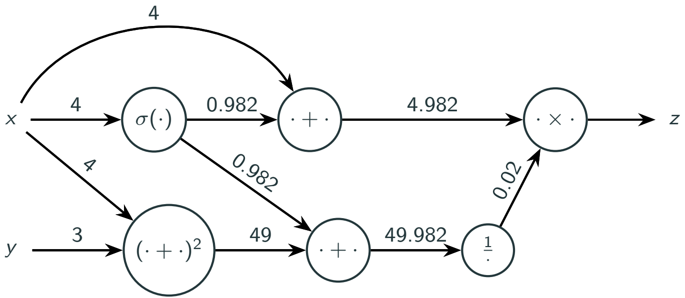


Forward pass: $x = 4, y = 3$

node	inputs i	outputs o	derivatives
$\sigma(\cdot)$	$i = 4$	$o_1, o_2 = \sigma(i) = 0.982$	$\frac{do_1}{di} = \frac{do_2}{di} = 0.0177$
$(\cdot + \cdot)\uparrow$	$i_1 = 4, i_2 = 0.982$	$o = (i_1 + i_2) = 4.982$	$\frac{do}{di_1} = 1, \frac{do}{di_2} = 1$
$(\cdot + \cdot)^2$	$i_1 = 4, i_2 = 3$	$o = (i_1 + i_2)^2 = 49$	$\frac{do}{di_1} = \frac{do}{di_2} = 2(i_1 + i_2) = 14$
$(\cdot + \cdot)\downarrow$	$i_1 = 0.982, i_2 = 49$	$o = (i_1 + i_2) = 49.982$	$\frac{do}{di_1} = 1, \frac{do}{di_2} = 1$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

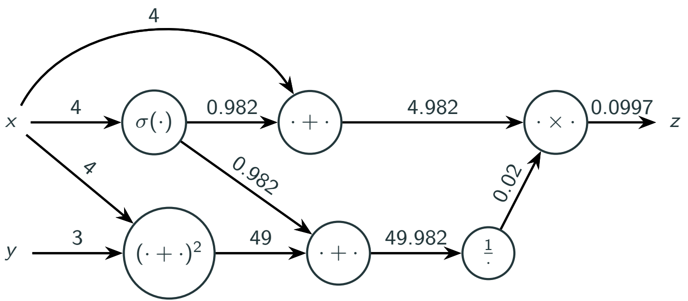


Forward pass: $x = 4, y = 3$

node	inputs i	outputs o	derivatives
$\sigma(\cdot)$	$i = 4$	$o_1, o_2 = \sigma(i) = 0.982$	$\frac{do_1}{di} = \frac{do_2}{di} = 0.0177$
$(\cdot + \cdot)\uparrow$	$i_1 = 4, i_2 = 0.982$	$o = (i_1 + i_2) = 4.982$	$\frac{do}{di_1} = 1, \frac{do}{di_2} = 1$
$(\cdot + \cdot)^2$	$i_1 = 4, i_2 = 3$	$o = (i_1 + i_2)^2 = 49$	$\frac{do}{di_1} = \frac{do}{di_2} = 2(i_1 + i_2) = 14$
$(\cdot + \cdot)\downarrow$	$i_1 = 0.982, i_2 = 49$	$o = (i_1 + i_2) = 49.982$	$\frac{do}{di_1} = 1, \frac{do}{di_2} = 1$
$1/\cdot$	$i = 49.982$	$o = 1/i = 0.02$	$\frac{do}{di} = -(1/49.982)^2 = -0.0004$

Let us go back to our complicated graph

Example: $z = \dfrac{x+\sigma(x)}{\sigma(x)+(x+y)^2}$

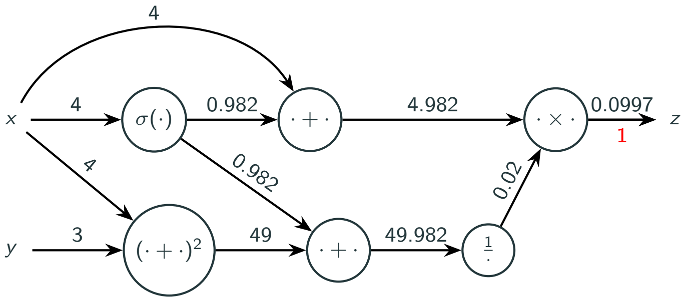


Forward pass: $x = 4, y = 3$

node	inputs i	outputs o	derivatives
$\sigma(\cdot)$	$i = 4$	$o_1, o_2 = \sigma(i) = 0.982$	$\frac{do_1}{di} = \frac{do_2}{di} = 0.0177$
$(\cdot+\cdot)\uparrow$	$i_1 = 4, i_2 = 0.982$	$o = (i_1 + i_2) = 4.982$	$\frac{do}{di_1} = 1, \frac{do}{di_2} = 1$
$(\cdot+\cdot)^2$	$i_1 = 4, i_2 = 3$	$o = (i_1 + i_2)^2 = 49$	$\frac{do}{di_1} = \frac{do}{di_2} = 2(i_1 + i_2) = 14$
$(\cdot+\cdot)\downarrow$	$i_1 = 0.982, i_2 = 49$	$o = (i_1 + i_2) = 49.982$	$\frac{do}{di_1} = 1, \frac{do}{di_2} = 1$
$1/\cdot$	$i = 49.982$	$o = 1/i = 0.02$	$\frac{do}{di} = -(1/49.982)^2 = -0.0004$
$\cdot\times\cdot$	$i_1 = 4.982, i_2 = 0.02$	$o = (i_1 i_2) = 0.0997$	$\frac{do}{di_1} = i_2 = 0.02, \frac{do}{di_2} = i_1 = 4.982$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x + y)^2}$

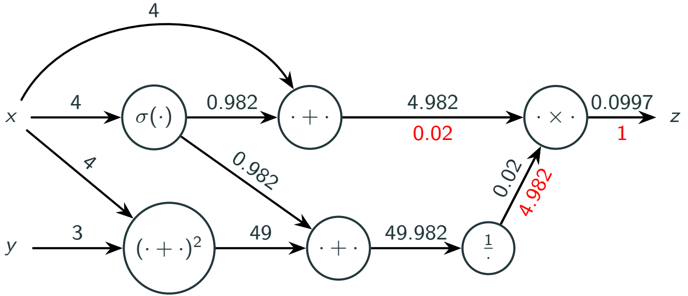


Backward pass: $x = 4, y = 3$

node	$\frac{dz}{do_i}$	$\frac{do_i}{di_j}$	$\frac{dz}{di_j}$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

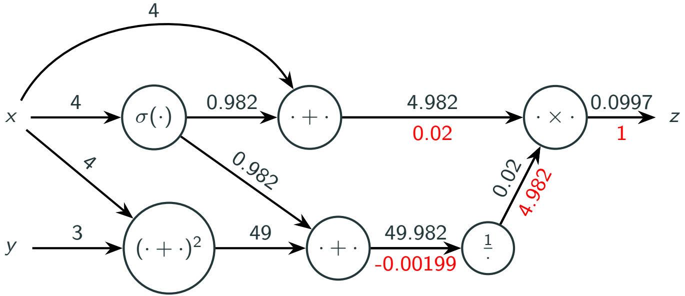


Backward pass: $x = 4, y = 3$

node	$\frac{dz}{do_i}$	$\frac{do_i}{di_j}$	$\frac{dz}{di_j}$
$\cdot \times \cdot$	1	$\frac{do}{di_1} = 0.02$, $\frac{do}{di_2} = 4.982$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = 0.02$, $\frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = 4.982$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

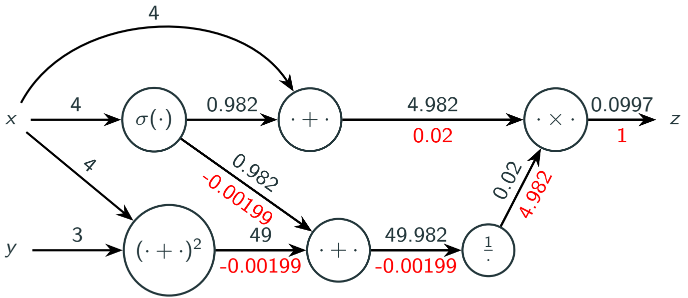


Backward pass: $x = 4, y = 3$

node	$\frac{dz}{do_i}$	$\frac{do_i}{di_j}$	$\frac{dz}{di_j}$
$\cdot \times \cdot$	1	$\frac{do}{di_1} = 0.02$, $\frac{do}{di_2} = 4.982$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = 0.02$, $\frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = 4.982$
$1/\cdot$	4.982	$\frac{do}{di} = -0.0004$	$\frac{do}{di} = \frac{dz}{do}\frac{do}{di} = -0.00199$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

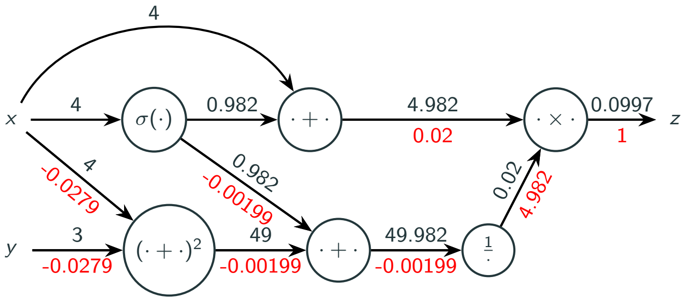


Backward pass: $x = 4, y = 3$

node	$\frac{dz}{do_i}$	$\frac{do_i}{di_j}$	$\frac{dz}{di_j}$
$\cdot \times \cdot$	1	$\frac{do}{di_1} = 0.02$, $\frac{do}{di_2} = 4.982$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = 0.02$, $\frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = 4.982$
$1/\cdot$	4.982	$\frac{do}{di} = -0.0004$	$\frac{do}{di} = \frac{dz}{do}\frac{do}{di} = -0.00199$
$(\cdot + \cdot)\downarrow$	-0.00199	$\frac{do}{di_1} = 1$, $\frac{do}{di_2} = 1$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = -0.00199$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

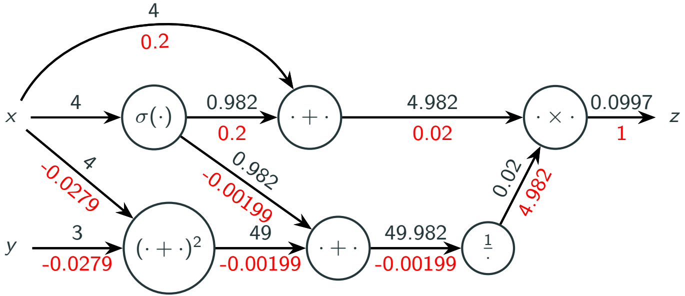


Backward pass: $x = 4, y = 3$

node	$\frac{dz}{do_i}$	$\frac{do_i}{di_j}$	$\frac{dz}{di_j}$
$\cdot \times \cdot$	1	$\frac{do}{di_1} = 0.02,\ \frac{do}{di_2} = 4.982$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = 0.02,\ \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = 4.982$
$1/\cdot$	4.982	$\frac{do}{di} = -0.0004$	$\frac{do}{di} = \frac{dz}{do}\frac{do}{di} = -0.00199$
$(\cdot + \cdot)\downarrow$	-0.00199	$\frac{do}{di_1} = 1,\ \frac{do}{di_2} = 1$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = -0.00199$
$(\cdot + \cdot)^2$	-0.00199	$\frac{do}{di_1} = \frac{do}{di_2} = 14$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = -0.0279,\ \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = -0.0279$

Let us go back to our complicated graph

Example: $z = \dfrac{x + \sigma(x)}{\sigma(x) + (x+y)^2}$

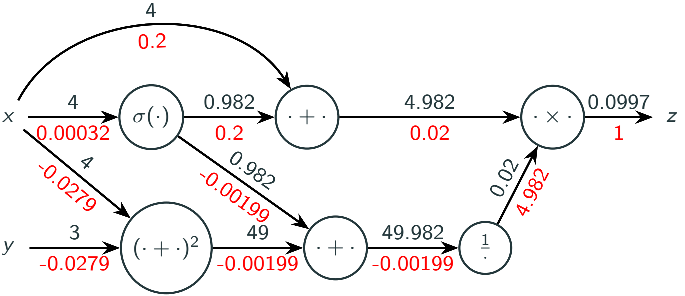


Backward pass: $x = 4, y = 3$

node	$\frac{dz}{do_i}$	$\frac{do_i}{di_j}$	$\frac{dz}{di_j}$
$\cdot \times \cdot$	1	$\frac{do}{di_1} = 0.02,\ \frac{do}{di_2} = 4.982$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = 0.02,\ \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = 4.982$
$1/\cdot$	4.982	$\frac{do}{di} = -0.0004$	$\frac{do}{di} = \frac{dz}{do}\frac{do}{di} = -0.00199$
$(\cdot + \cdot)\downarrow$	-0.00199	$\frac{do}{di_1} = 1,\ \frac{do}{di_2} = 1$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = -0.00199$
$(\cdot + \cdot)^2$	-0.00199	$\frac{do}{di_1} = \frac{do}{di_2} = 14$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = -0.0279,\ \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = -0.0279$
$(\cdot + \cdot)\uparrow$	0.02	$\frac{do}{di_1} = 1,\ \frac{do}{di_2} = 1$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = 0.02$

Let us go back to our complicated graph

Example: $z = \dfrac{x+\sigma(x)}{\sigma(x)+(x+y)^2}$



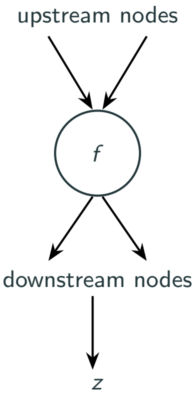
Backward pass: $x = 4, y = 3$

node	$\frac{dz}{do_i}$	$\frac{do_i}{di_j}$	$\frac{dz}{di_j}$
$\cdot \times \cdot$	1	$\frac{do}{di_1} = 0.02,\ \frac{do}{di_2} = 4.982$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = 0.02,\ \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = 4.982$
$1/\cdot$	4.982	$\frac{do}{di} = -0.0004$	$\frac{do}{di} = \frac{dz}{do}\frac{do}{di} = -0.00199$
$(\cdot + \cdot)\downarrow$	-0.00199	$\frac{do}{di_1} = 1,\ \frac{do}{di_2} = 1$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = -0.00199$
$(\cdot + \cdot)^2$	-0.00199	$\frac{do}{di_1} = \frac{do}{di_2} = 14$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = -0.0279,\ \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = -0.0279$
$(\cdot + \cdot)\uparrow$	0.02	$\frac{do}{di_1} = 1,\ \frac{do}{di_2} = 1$	$\frac{dz}{di_1} = \frac{dz}{do}\frac{do}{di_1} = \frac{dz}{di_2} = \frac{dz}{do}\frac{do}{di_2} = 0.02$
$\sigma(\cdot)$	$\frac{dz}{do_1} = 0.02,\ \frac{dz}{do_2} = -0.00199$	$\frac{do_1}{di} = \frac{do_2}{di} = 0.0177$	$\frac{dz}{di} = \frac{dz}{do_1}\frac{do_1}{di} + \frac{dz}{do_2}\frac{do_2}{di} = 0.00032$

Backpropagation

Forward pass:

- wait for inputs $x_1, \ldots, x_n$ from upstream nodes
- compute ouputs $o_1 = f_1(x_1, \ldots, x_n), \ldots, o_m = f_m(x_1, \ldots, x_n)$,
- compute all derivatives between each input/output pair: $\frac{do_i}{dx_j}$, for $i = 1, \ldots, m; j = 1, \ldots, n$ and store them,
- pass outputs $o_1, \ldots, o_m$ to downstream nodes.



Backward pass:

- wait for derivatives $\frac{dz}{do_1}, \ldots, \frac{dz}{o_m}$ from downstream nodes
- using the cached derivatives, compute derivatives of z with respect to all inputs $\frac{dz}{dx_1}, \ldots, \frac{dz}{x_n}$, with $\frac{dz}{dx_i} = \frac{dz}{do_1}\frac{do_1}{dx_i} + \cdots + \frac{dz}{do_m}\frac{do_m}{dx_i}$
- pass derivatives to upstream nodes.

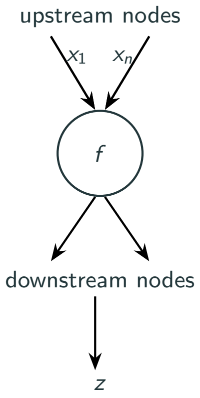
Backpropagation

Forward pass:

- wait for inputs $x_1, \ldots, x_n$ from upstream nodes
- compute ouputs $o_1 = f_1(x_1, \ldots, x_n), \ldots, o_m = f_m(x_1, \ldots, x_n)$,
- compute all derivatives between each input/output pair: $\frac{do_i}{dx_j}$, for $i = 1, \ldots, m; j = 1, \ldots, n$ and store them,
- pass outputs $o_1, \ldots, o_m$ to downstream nodes.

Backward pass:

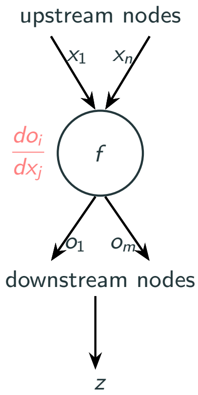
- wait for derivatives $\frac{dz}{do_1}, \ldots, \frac{dz}{o_m}$ from downstream nodes
- using the cached derivatives, compute derivatives of z with respect to all inputs $\frac{dz}{dx_1}, \ldots, \frac{dz}{x_n}$, with $\frac{dz}{dx_i} = \frac{dz}{do_1}\frac{do_1}{dx_i} + \cdots + \frac{dz}{do_m}\frac{do_m}{dx_i}$
- pass derivatives to upstream nodes.



Backpropagation

Forward pass:

- wait for inputs $x_1, \ldots, x_n$ from upstream nodes
- compute ouputs $o_1 = f_1(x_1, \ldots, x_n), \ldots, o_m = f_m(x_1, \ldots, x_n)$,
- compute all derivatives between each input/output pair: $\frac{do_i}{dx_j}$, for $i = 1, \ldots, m; j = 1, \ldots, n$ and store them,
- pass outputs $o_1, \ldots, o_m$ to downstream nodes.



Backward pass:

- wait for derivatives $\frac{dz}{do_1}, \ldots, \frac{dz}{o_m}$ from downstream nodes
- using the cached derivatives, compute derivatives of z with respect to all inputs $\frac{dz}{dx_1}, \ldots, \frac{dz}{x_n}$, with $\frac{dz}{dx_i} = \frac{dz}{do_1}\frac{do_1}{dx_i} + \cdots + \frac{dz}{do_m}\frac{do_m}{dx_i}$
- pass derivatives to upstream nodes.

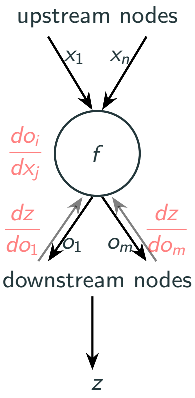
Backpropagation

Forward pass:

- wait for inputs $x_1, ..., x_n$ from upstream nodes
- compute ouputs $o_1 = f_1(x_1, ..., x_n), ..., o_m = f_m(x_1, ..., x_n)$,
- compute all derivatives between each input/output pair: $\frac{do_i}{dx_j}$, for $i = 1, ..., m; j = 1, ..., n$ and store them,
- pass outputs $o_1, ..., o_m$ to downstream nodes.

Backward pass:

- wait for derivatives $\frac{dz}{do_1}, ..., \frac{dz}{o_m}$ from downstream nodes
- using the cached derivatives, compute derivatives of z with respect to all inputs $\frac{dz}{dx_1}, ..., \frac{dz}{x_n}$, with $\frac{dz}{dx_i} = \frac{dz}{do_1}\frac{do_1}{dx_i} + \cdots + \frac{dz}{do_m}\frac{do_m}{dx_i}$
- pass derivatives to upstream nodes.

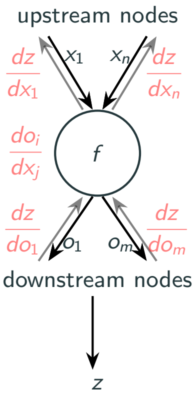


Forward pass:

- wait for inputs $x_1, \ldots, x_n$ from upstream nodes
- compute ouputs $o_1 = f_1(x_1, \ldots, x_n), \ldots, o_m = f_m(x_1, \ldots, x_n)$,
- compute all derivatives between each input/output pair: $\frac{do_i}{dx_j}$, for $i = 1, \ldots, m; j = 1, \ldots, n$ and store them,
- pass outputs $o_1, \ldots, o_m$ to downstream nodes.

Backward pass:

- wait for derivatives $\frac{dz}{do_1}, \ldots, \frac{dz}{o_m}$ from downstream nodes
- using the cached derivatives, compute derivatives of z with respect to all inputs $\frac{dz}{dx_1}, \ldots, \frac{dz}{x_n}$, with $\frac{dz}{dx_i} = \frac{dz}{do_1}\frac{do_1}{dx_i} + \cdots + \frac{dz}{do_m}\frac{do_m}{dx_i}$
- pass derivatives to upstream nodes.



Chain rule - vector functions

In practice, we work with functions process multi-dimensional inputs and produce multidimensional outputs.

Derivative of a composition of functions. Let $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$, $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^p$ two differentiable functions. We define $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^p$ as

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) = \mathbf{f}(\mathbf{g}(\mathbf{x})), \quad \mathbf{y} = \mathbf{g}(\mathbf{x}).$$

$$\mathbf{z} = \begin{pmatrix} z_1 \\ \vdots \\ z_p \end{pmatrix} = \begin{pmatrix} f_1(y_1, \ldots, y_m) \\ \vdots \\ f_p(y_1, \ldots, y_m) \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} g_1(x_1, \ldots, x_n) \\ \vdots \\ g_m(x_1, \ldots, x_n) \end{pmatrix}.$$

Chain rule - vector functions

We will compute the **Jacobian matrix**, which contains the derivatives of all output functions with respect to all inputs.

$$D_{\mathbf{x}}\mathbf{g}(x) = \frac{d\mathbf{y}}{d\mathbf{x}} = \begin{pmatrix} \frac{dy_1}{dx_1} & \cdots & \frac{dy_1}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{dy_m}{dx_1} & \cdots & \frac{dy_m}{dx_n} \end{pmatrix}.$$

The chain rule is the same as before, except that this time we multiply Jacobian matrices!

$$D\mathbf{h}(\mathbf{x}) = D\mathbf{f}(\mathbf{g}(\mathbf{x}))D\mathbf{g}(\mathbf{x}) \quad \Longrightarrow \quad \underbrace{\frac{d\mathbf{z}}{d\mathbf{x}}(\mathbf{x})}_{p\times n} = \underbrace{\frac{d\mathbf{z}}{d\mathbf{y}}(\mathbf{y}(\mathbf{x}))}_{p\times m} \underbrace{\frac{d\mathbf{y}}{d\mathbf{x}}(\mathbf{x})}_{m\times n} \quad \text{or} \quad \left.\frac{d\mathbf{z}}{d\mathbf{x}}\right|_{\mathbf{x}} = \left.\frac{d\mathbf{z}}{d\mathbf{y}}\right|_{\mathbf{y}(\mathbf{x})} \left.\frac{d\mathbf{y}}{d\mathbf{x}}\right|_{\mathbf{x}}$$

Propagation of gradients

Let's go back to our initial neural network. This time we assume that

$$\begin{aligned} \mathbf{x}, \mathbf{y}, \hat{\mathbf{y}} &\in \mathbb{R}^n \\ \mathbf{h}_i &\in \mathbb{R}^{n_i} \\ \mathbf{w}_i &\in \mathbb{R}^{p_i}. \end{aligned}$$

The loss value continues to be a scalar, i.e. $\ell(\mathbf{y}, \hat{\mathbf{y}}) \in \mathbb{R}$.

In this case, in the forward pass we need to store jacobian matrices, and in the backward pass we backpropagate gradients.

Let's go back to our initial neural network. This time we assume that

$$\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^n$$
$$\mathbf{h}_i \in \mathbb{R}^{n_i}$$
$$\mathbf{w}_i \in \mathbb{R}^{p_i}.$$

The loss value continues to be a scalar, i.e. $\ell(\mathbf{y}, \hat{\mathbf{y}}) \in \mathbb{R}$.

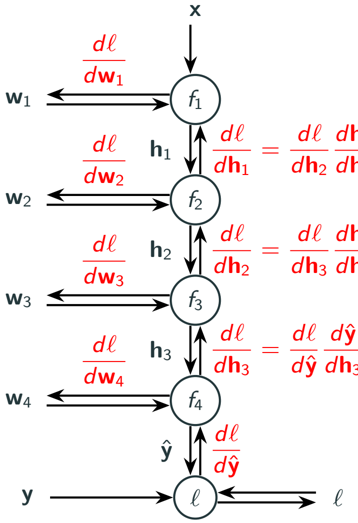
In this case, in the forward pass we need to store jacobian matrices, and in the backward pass we backpropagate gradients.

For example for node f_2, we need to store

$$\frac{d\mathbf{h}_2}{d\mathbf{h}_1},\ n_2 \times n_1 \quad \text{and} \quad \frac{d\mathbf{h}_2}{d\mathbf{w}_2},\ n_2 \times p_2.$$

and in the backprop pass we compute:

$$\underbrace{\frac{d\ell}{d\mathbf{h}_1}}_{1\times n_1} = \underbrace{\frac{d\ell}{d\mathbf{h}_2}}_{1\times n_2} \underbrace{\frac{d\mathbf{h}_2}{d\mathbf{h}_1}}_{n_2\times n_1}, \quad \text{and} \quad \underbrace{\frac{d\ell}{d\mathbf{w}_2}}_{1\times p_2} = \underbrace{\frac{d\ell}{d\mathbf{h}_2}}_{1\times n_2} \underbrace{\frac{d\mathbf{h}_2}{d\mathbf{w}_2}}_{n_2\times p_2}.$$



Any questions?