INTRODUCTION TO NEURAL NETWORKS

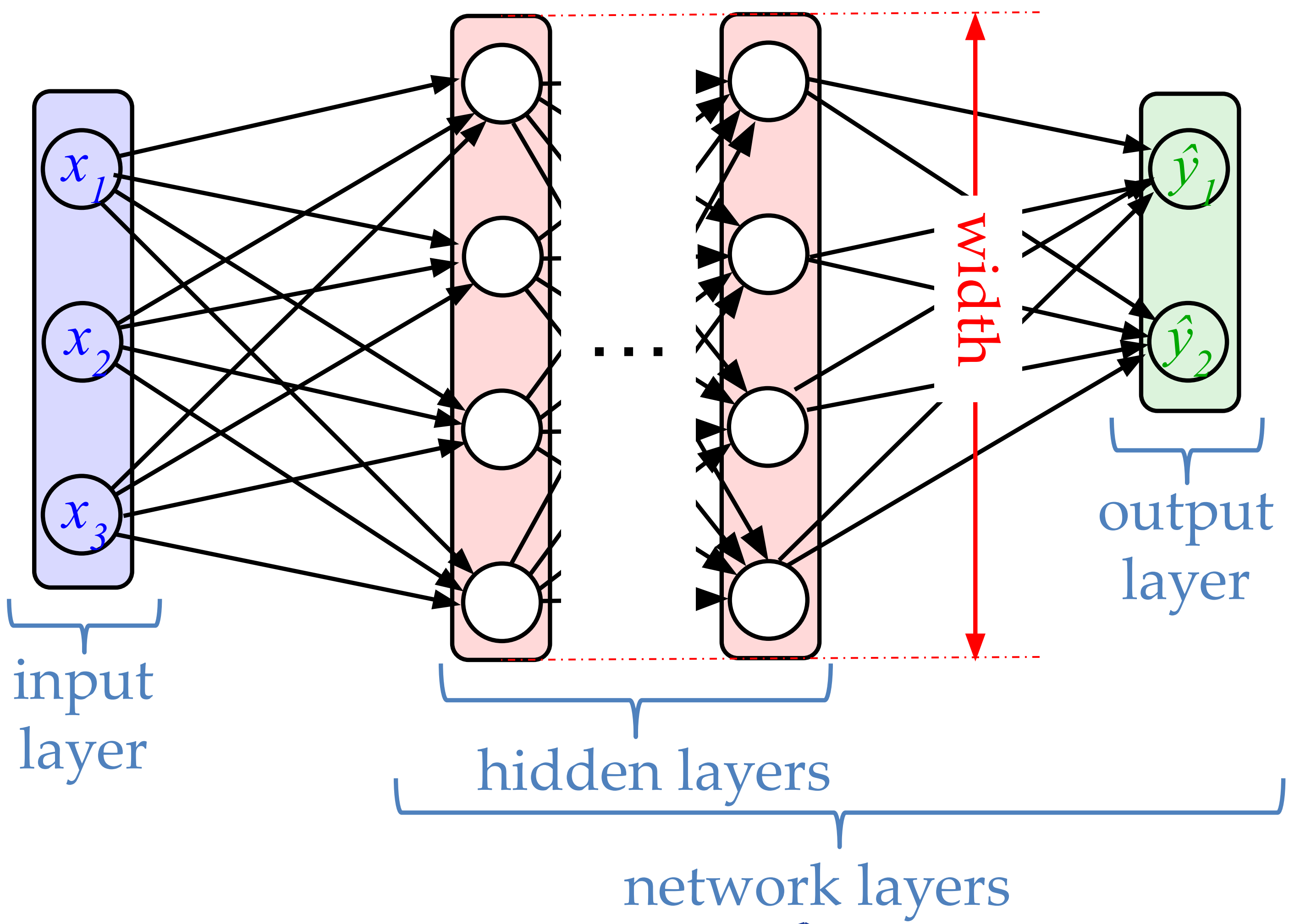
Backpropagation

Pascal Germain*, 2019

Translated to English by Vera Shalaeva, 2020

* Thanks to Philippe Giguère for his permit to reuse some of his slides.

Illustration and notions



$$\hat{y} = f^{(3)}\left(f^{(2)}\left(f^{(1)}\left(x\right)\right)\right)$$

Parameters to choose

- Architecture
  - # layers
  - # neurones (hidden) by layer
  - type of layer
- Output neuron function
- Loss function
- Optimizer
  - and other « *details* »

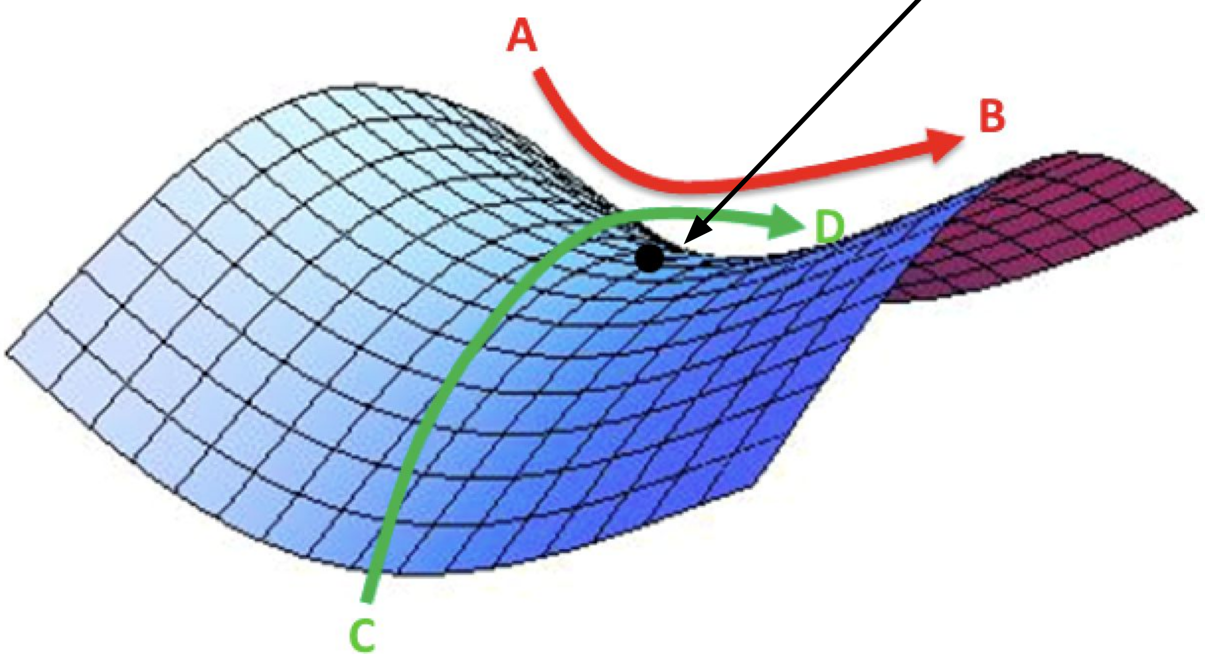
Comparison with classical methods

- Many learning methods are convex
  - Least squares
  - Logistic regression
  - SVM
- Neural Networks are not convex
  - Challenging to get theoretical guarantees.
  - Result varies according to initialization of the gradient descent.
  - We have to accept that local minimum might be a good solution.
  - Research shows that solutions are often saddle points
    - ratio (saddle points)/(local minimum) increase exponentially with the number of parameters to estimate

See Goodfellow et al. Section 8.2.3

Saddle point example

- Partial derivative is equal to zero at a saddle point.



https://www.safaribooksonline.com/library/view/fundamentals-of-deep/9781491925607/ch04.html

Profile of a loss function

Adapté de cs231n

Backpropagation algorithm
(«*backprop*»)

The chain rule.

$$\boxed{\begin{aligned} \frac{\partial f(h(x))}{\partial x} &= \frac{\partial f(h(x))}{\partial h(x)} \frac{\partial h(x)}{\partial x} \\ &= \left[\frac{\partial f(a)}{\partial a}\right]_{a=h(x)} \frac{\partial h(x)}{\partial x} \end{aligned}}$$

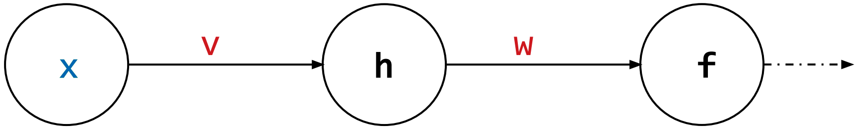
For example: $F(x) = (2x+3)^2$

$= f(2x+3)$ where: $f(x) = x^2$

$= f(h(x))$ where: $h(x) = 2x+3.$

Then :

$$\begin{aligned} \frac{\partial F(x)}{\partial x} &= \frac{\partial f(h(x))}{\partial h(x)} \frac{\partial h(x)}{\partial x} \\ &= 2\,h(x) \times 2 \\ &= 4(2x+3) \end{aligned}$$



$$\frac{\partial f(h(x))}{\partial x} = \left[\frac{\partial f(a)}{\partial a}\right]_{a=h(x)} \frac{\partial h(x)}{\partial x}$$

$$R_{v,w}(x) = f(w \cdot h(v \cdot x))$$

$$\frac{\partial L(R_{v,w}(x), y)}{\partial w} = \left[\frac{\partial L(r, y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \frac{\partial R_{v,w}(x)}{\partial w}$$

The chain rule.

$$\frac{\partial f(h(x))}{\partial x} = \frac{\partial f(h(x))}{\partial h(x)} \frac{\partial h(x)}{\partial x}$$
$$= \left[\frac{\partial f(a)}{\partial a}\right]_{a=h(x)} \frac{\partial h(x)}{\partial x}$$

We can also write:

$$(f \circ h)' = (f' \circ h) \times h'$$

$$(\, f(h(x))\,)' = f'(h(x)) \times h'(x)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial h} \frac{\partial h}{\partial x}$$

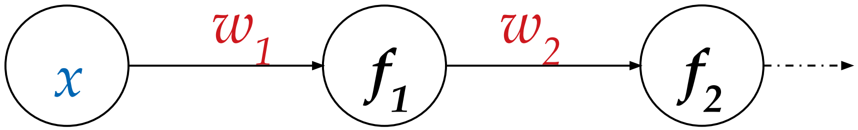
$$x \xrightarrow{v} h \xrightarrow{w} f \dashrightarrow$$

$$R_{v,w}(x) = f(w \cdot h(v \cdot x))$$

$$\boxed{\frac{\partial f(h(x))}{\partial x} = \left[\frac{\partial f(a)}{\partial a}\right]_{a=h(x)} \frac{\partial h(x)}{\partial x}}$$

$$\begin{aligned}
\frac{\partial L(R_{v,w}(x),y)}{\partial w} &= \left[\frac{\partial L(r,y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \frac{\partial R_{v,w}(x)}{\partial w} \\
&= \left[\frac{\partial L(r,y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \left[\frac{\partial f(a)}{\partial a}\right]_{a=w \cdot h(v \cdot x)} \cdot \frac{\partial w \cdot h(v \cdot x)}{\partial w} \\
&= \boxed{\left[\frac{\partial L(r,y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \left[\frac{\partial f(a)}{\partial a}\right]_{a=w \cdot h(v \cdot x)}} \cdot h(v \cdot x)
\end{aligned}$$

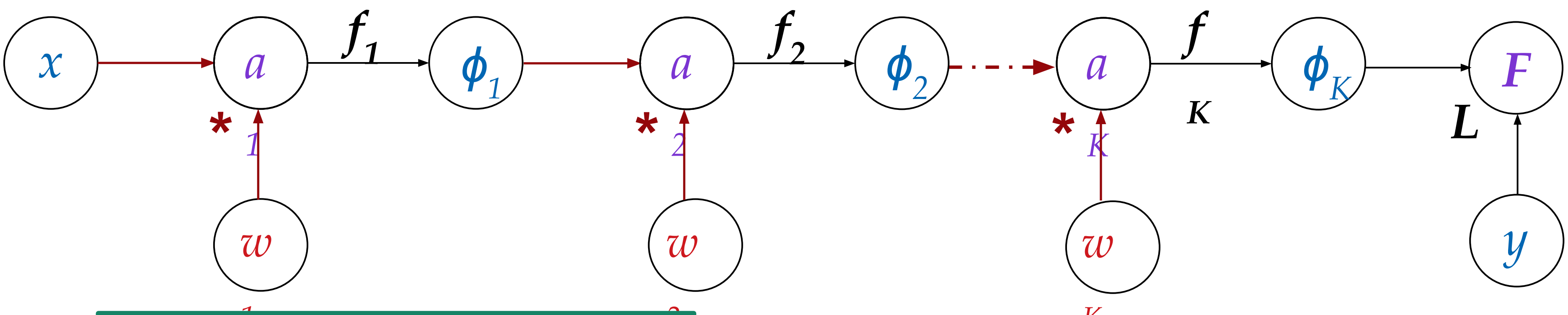
$$\begin{aligned}
\frac{\partial L(R_{v,w}(x),y)}{\partial v} &= \left[\frac{\partial L(r,y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \frac{\partial R_{v,w}(x)}{\partial v} \\
&= \left[\frac{\partial L(r,y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \left[\frac{\partial f(a)}{\partial a}\right]_{a=w \cdot h(v \cdot x)} \cdot \frac{\partial w \cdot h(v \cdot x)}{\partial v} \\
&= \left[\frac{\partial L(r,y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \left[\frac{\partial f(a)}{\partial a}\right]_{a=w \cdot h(v \cdot x)} \cdot w \cdot \frac{\partial h(v \cdot x)}{\partial v} \\
&= \left[\frac{\partial L(r,y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \left[\frac{\partial f(a)}{\partial a}\right]_{a=w \cdot h(v \cdot x)} \cdot w \cdot \left[\frac{\partial h(b)}{\partial b}\right]_{b=v \cdot x} \cdot \frac{\partial v \cdot x}{\partial v} \\
&= \boxed{\left[\frac{\partial L(r,y)}{\partial r}\right]_{r=R_{v,w}(x,y)} \cdot \left[\frac{\partial f(a)}{\partial a}\right]_{a=w \cdot h(v \cdot x)}} \cdot w \cdot \left[\frac{\partial h(b)}{\partial b}\right]_{b=v \cdot x} \cdot x
\end{aligned}$$



$$\boxed{\frac{\partial f(h(x))}{\partial x} = \left[\frac{\partial f(a)}{\partial a}\right]_{a=h(x)} \frac{\partial h(x)}{\partial x}}$$

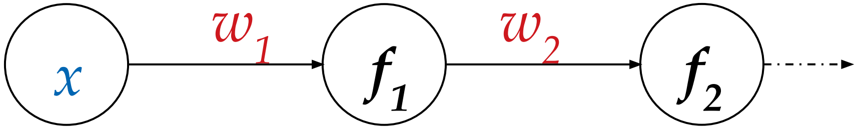
$$R(x) = f_2(w_2 \cdot f_1(w_1 \cdot x))$$

$$F = L\Big(R(x), y\Big)$$



Step 1: Forward propagation

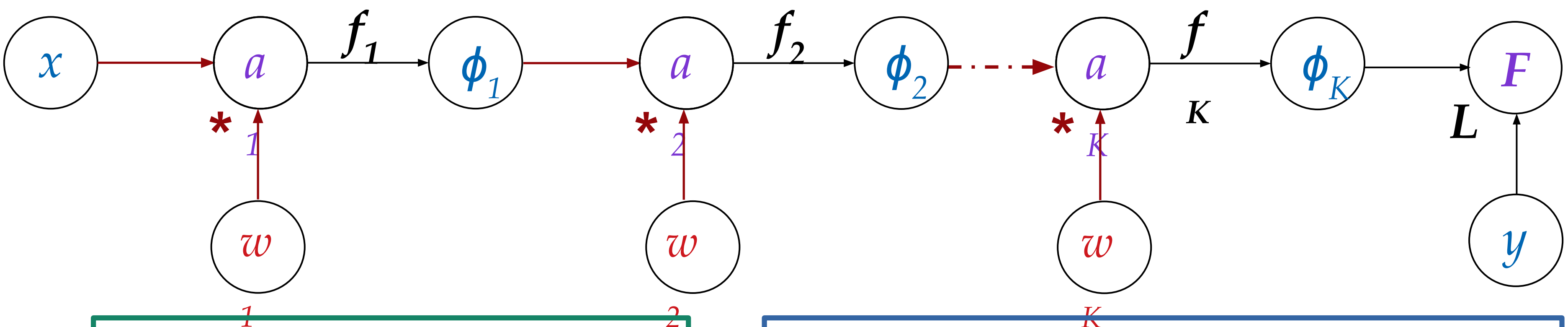
Step 2: Backpropagation of gradient



$$\frac{\partial f(h(x))}{\partial x} = \left[\frac{\partial f(a)}{\partial a}\right]_{a=h(x)} \frac{\partial h(x)}{\partial x}$$

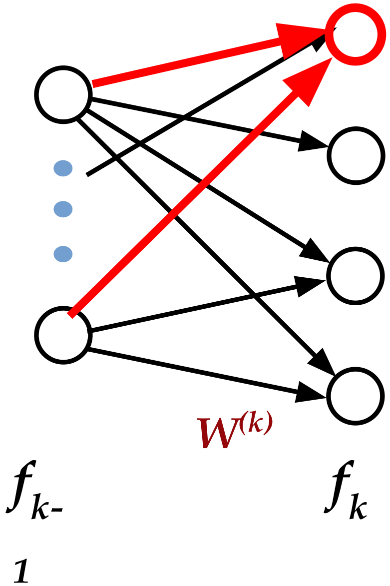
$$R(x) = f_2(w_2 \cdot f_1(w_1 \cdot x))$$

$$F = L\Big(R(x), y\Big)$$

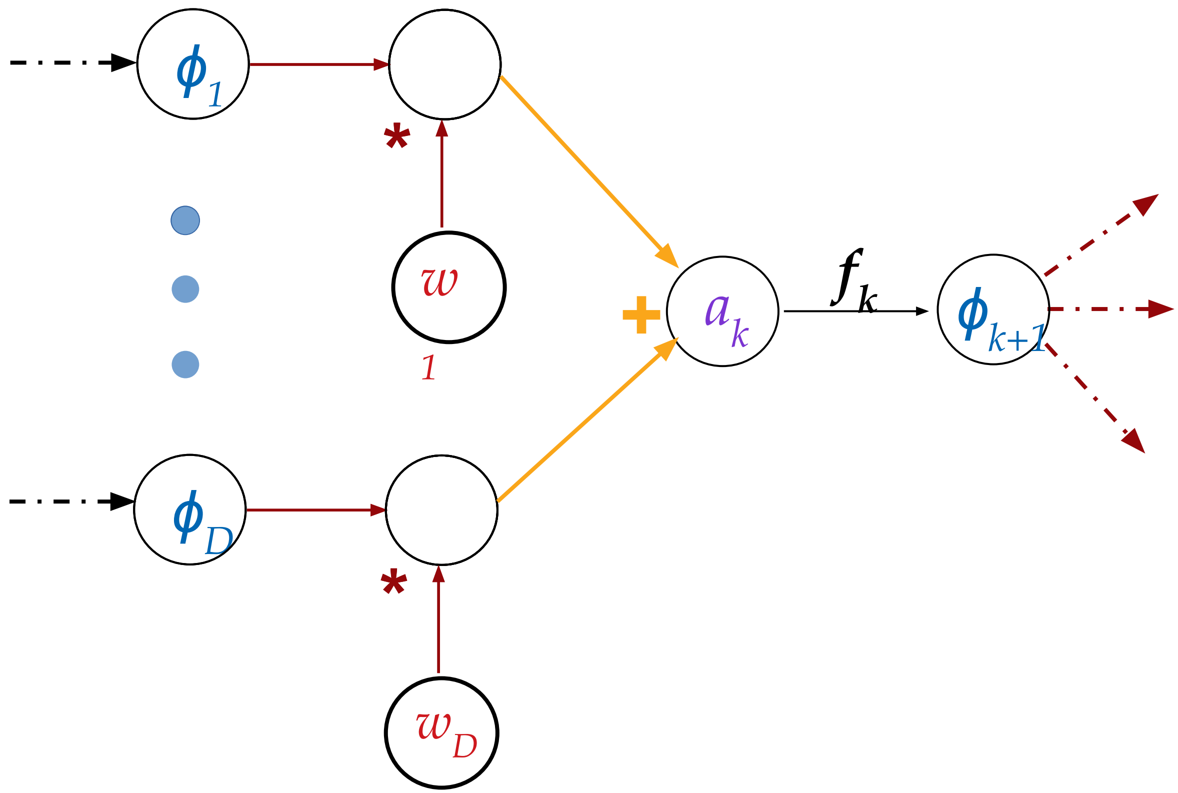


- $\phi_0 = x$
- For $k = 1, 2, \ldots K$:
    - $a_k = w_k \cdot \phi_{k-1}$
    - $\phi_k = f_k(a_k)$
- $F = L(\phi_K, y)$

- $\phi_K^\delta = \frac{\partial F}{\partial \phi_K} = L'(\phi_k, y)$
- For $k = K, K{-}1, \ldots, 1$:
    - $a_k^\delta = \frac{\partial F}{\partial \phi_k}\frac{\partial \phi_k}{\partial a_k} = \phi_k^\delta\, f_k'(a_k)$
    - $w_k^\delta = \frac{\partial F}{\partial a_k}\frac{\partial a_k}{\partial w_k} = a_k^\delta\, \phi_{k-1}$
    - $\phi_{k-1}^\delta = \frac{\partial F}{\partial a_k}\frac{\partial a_k}{\partial \phi_{k-1}} = a_k^\delta\, w_k$



$$\frac{\partial f(h(x))}{\partial x} = \left[\frac{\partial f(a)}{\partial a}\right]_{a=h(x)} \frac{\partial h(x)}{\partial x}$$



A network R of K hidden layers

- Activation functions: $f_1, \dots f_K$
- Weight matrices: $\mathbf{W}^{(1)}, \dots \mathbf{W}^{(K)}$
  - Each matrix $\mathbf{W}^{(k)}$ of size $d_k \times d_{k-1}$
  - d_k is the number of neurons at a layers k
  - $k = 0$ corresponds to the input layer $\mathbf{x} \in \mathbb{R}^{d_0}$

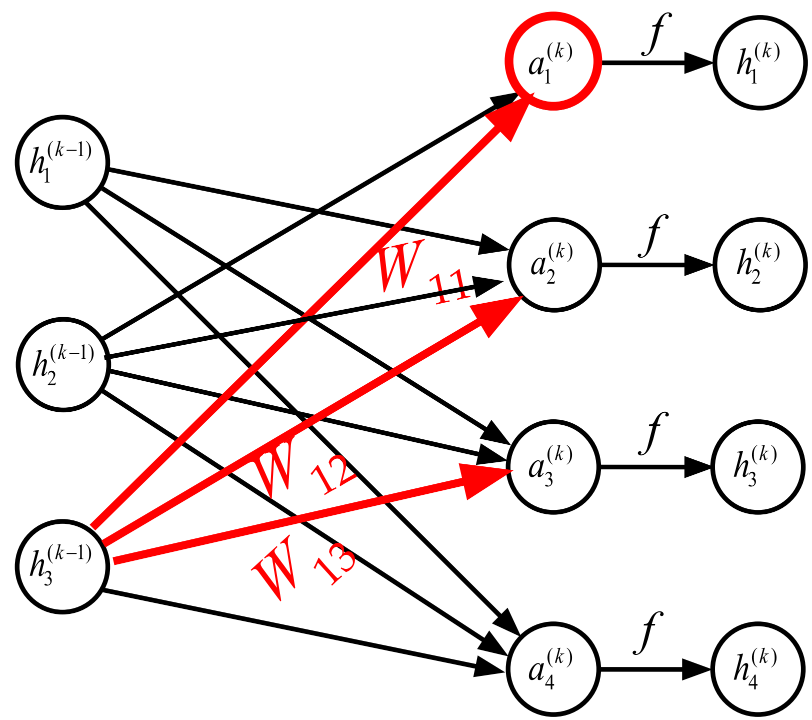
$$\mathbf{W}^{(k)} = \begin{bmatrix} W_{11} & \cdots & W_{1d_{K-1}} \\ \vdots & \ddots & \vdots \\ W_{d_K 1} & \cdots & W_{d_K d_{K-1}} \end{bmatrix}$$

Algorithm: Forward pass.

Input: A network *R*, Observation **x**

- $\mathbf{h}[0] \leftarrow \mathbf{x}$
- **For** k **from 1 to K:**
  - $\mathbf{a}[k] \leftarrow \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}$
  - $\mathbf{h}[k] \leftarrow f_k(\mathbf{a}[k])$

Output: $\mathbf{h}[K]$



Algorithm: Forward pass.

Input: A network R, Observation $\mathbf{x}$

- $\mathbf{h}[0] \leftarrow \mathbf{x}$
- For k de 1 à K:
  - $\mathbf{a}[k] \leftarrow \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}$
  - $\mathbf{h}[k] \leftarrow f_k\big(\mathbf{a}[k]\big)$

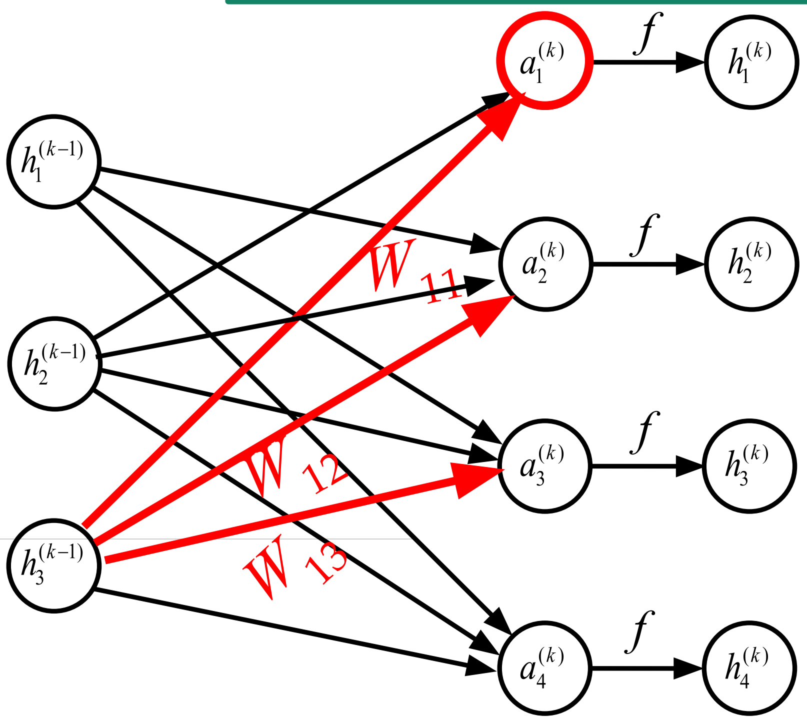
Output: $\mathbf{h}[K]$

Algorithm: Backpropagation.

Input: A network R, Observation $\mathbf{x}$, Loss L, output $\mathbf{y}$

- $\mathbf{g} \leftarrow L'\big(h[K], \mathbf{y}\big)$
- For k from K to 1:
  - $\mathbf{g} \leftarrow \mathbf{g} \odot f_k'(\mathbf{a}[k])$
  - $\nabla_{\mathbf{W}}[k] \leftarrow \mathbf{g}\,\mathbf{h}[k]^T$
  - $\mathbf{g} \leftarrow \mathbf{W}^{(k)T}\mathbf{g}$

Output: $\nabla_{\mathbf{W}}$



Automatic differentiation of the computational graph.

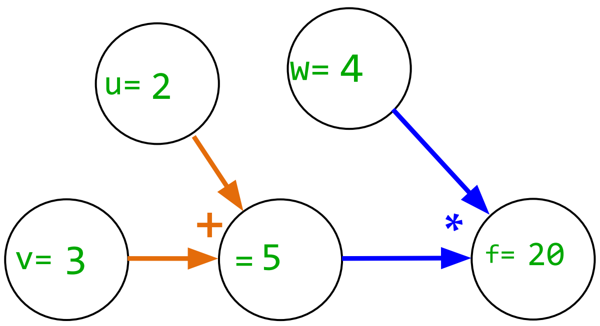
«Backprop» and automatic differentiation

- Algorithm that computes all gradients in a graph.
- **It is not optimization algorithm!**
- But all algorithm of neural network optimization use gradients computed by *backprop*.
- It is based on the chain rule.
- The modern libraries do the computations automatically (*pyTorch* et *TensorFlow*).

Example of computations on a simple graph

$$f = \boxed{\boxed{(u+v)}w}$$

node : variable
arrow : operation



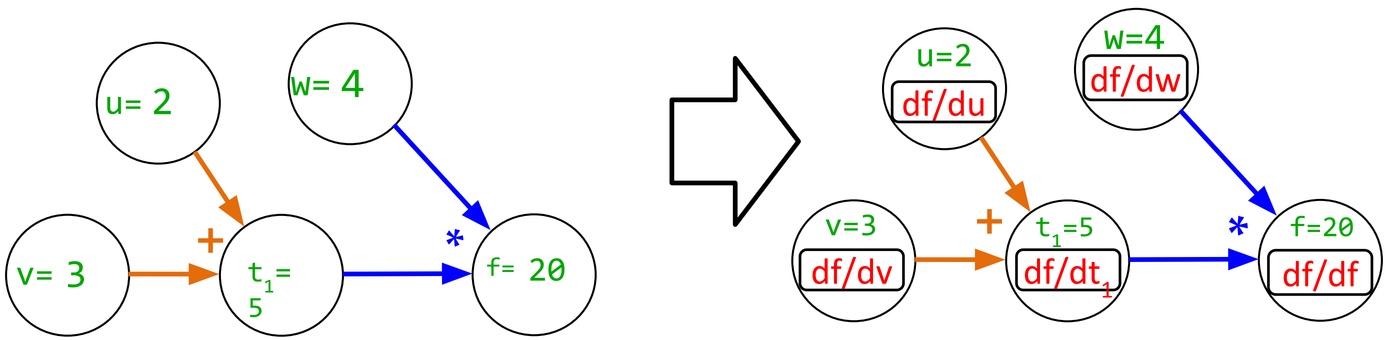
Initial values of variables:
u=2
v=3
w=4

Evaluation of graph to get f: **Forward pass**

Example of computations on a simple graph

From a forward pass graph computations:

We add a variable to save the gradients:

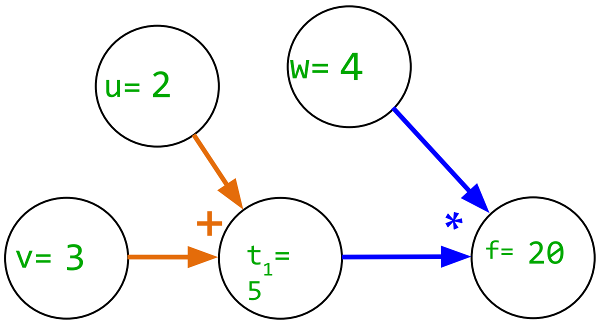


$$f=(u+v)w$$

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial h}\frac{\partial h}{\partial x}}$$

Example of computations on a simple graph

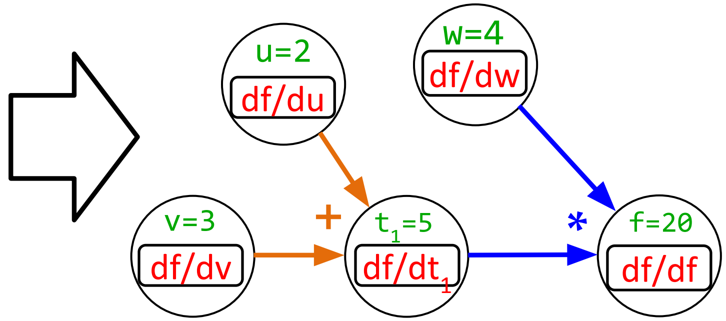
From a forward pass graph computations:



$$f=(u+v)w$$

$$f=t_1 w,\ t_1=u+v$$

We add a variable to save the gradients:



$$\boxed{\frac{\partial f}{\partial x}=\frac{\partial f}{\partial h}\frac{\partial h}{\partial x}}$$

$$\frac{\partial f}{\partial f}=1$$

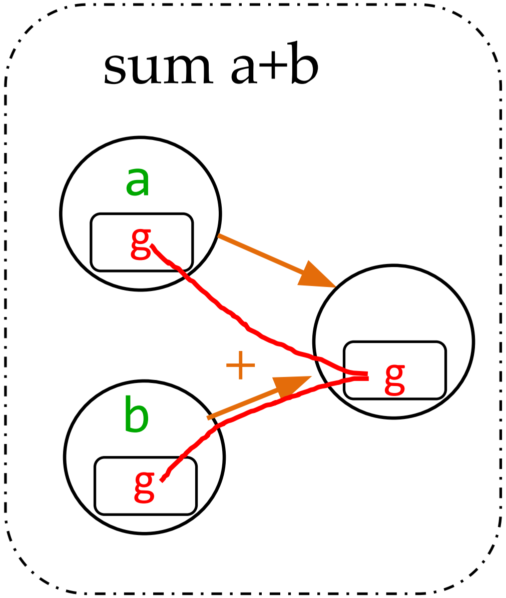
$$\frac{\partial f}{\partial w}=\frac{\partial}{\partial w}(t_1 w)\frac{\partial f}{\partial f}=t_1\cdot 1$$

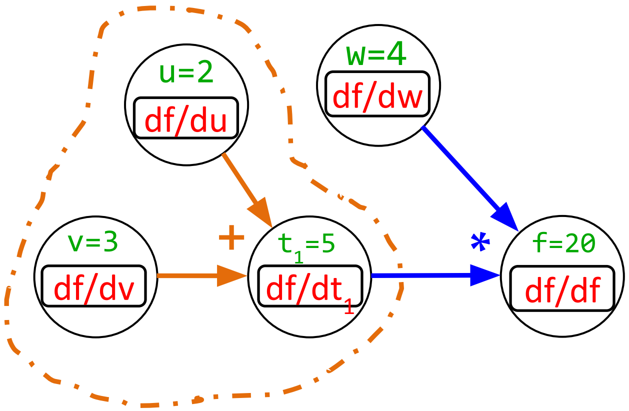
$$\frac{\partial f}{\partial t_1}=\frac{\partial}{\partial t_1}(t_1 w)\frac{\partial f}{\partial f}=w\cdot 1$$

$$\frac{\partial f}{\partial u}=\frac{\partial t_1}{\partial u}\frac{\partial f}{\partial t_1}=1\cdot\frac{\partial f}{\partial t_1}=4$$

$$\frac{\partial f}{\partial v}=\frac{\partial t_1}{\partial v}\frac{\partial f}{\partial t_1}=1\cdot\frac{\partial f}{\partial t_1}=4$$

Deriving the basic rules

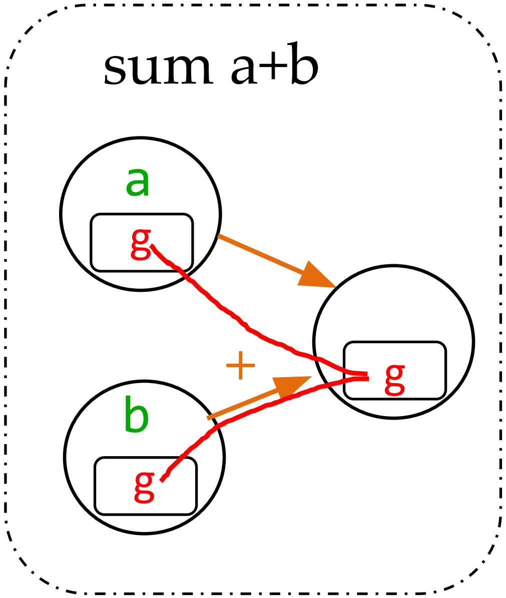


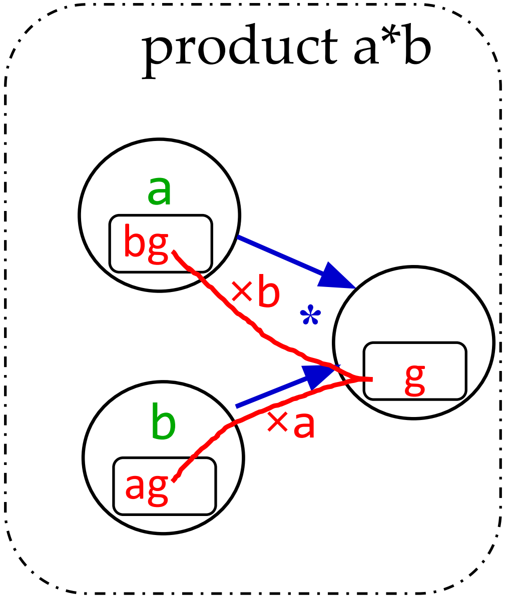


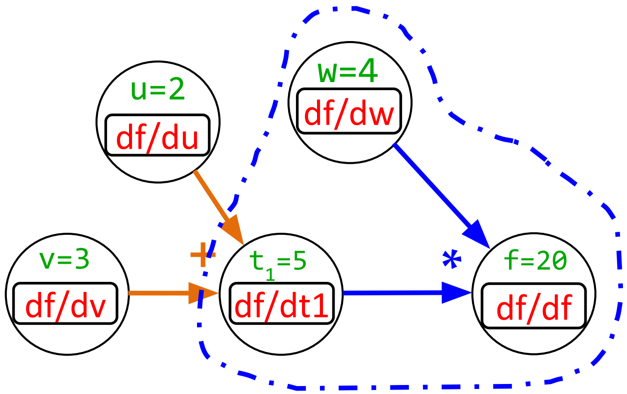
$$\frac{\partial f}{\partial w} = \frac{\partial}{\partial w}(t_1 w)\frac{\partial f}{\partial f} = t_1 \cdot 1 \qquad \frac{\partial f}{\partial t_1} = \frac{\partial}{\partial t_1}(t_1 w)\frac{\partial f}{\partial f} = w \cdot 1$$

$$\frac{\partial f}{\partial u} = \frac{\partial t_1}{\partial u}\frac{\partial f}{\partial t_1} = 1 \cdot \frac{\partial f}{\partial t_1} = 4 \qquad \frac{\partial f}{\partial v} = \frac{\partial t_1}{\partial v}\frac{\partial f}{\partial t_1} = 1 \cdot \frac{\partial f}{\partial t_1} = 4$$

Deriving the basic rules







$$\frac{\partial f}{\partial w} = \frac{\partial}{\partial w}(t_1 w)\frac{\partial f}{\partial f} = t_1 \frac{\partial f}{\partial f} \qquad \frac{\partial f}{\partial t_1} = \frac{\partial}{\partial t_1}(t_1 w)\frac{\partial f}{\partial f} = w\frac{\partial f}{\partial f}$$

$$\frac{\partial f}{\partial u} = \frac{\partial t_1}{\partial u}\frac{\partial f}{\partial t_1} = 1 \cdot \frac{\partial f}{\partial t_1} = 4 \qquad \frac{\partial f}{\partial v} = \frac{\partial t_1}{\partial v}\frac{\partial f}{\partial t_1} = 1 \cdot \frac{\partial f}{\partial t_1} = 4$$

Derivative of activation functions and classical loss functions.

Derivative of loss functions.

Quadratic loss

$$L_{\text{quad}}(\hat{y}, y) = (\hat{y} - y)^2$$

$$\begin{aligned} L'_{\text{quad}}(\hat{y}, y) &= \frac{\partial L_{\text{quad}}(\hat{y}, y)}{\partial \hat{y}} \\ &= 2(\hat{y} - y) \end{aligned}$$

Derivative of loss functions.

Quadratic loss.

$$L_{\text{quad}}(\hat{y}, y) = (\hat{y} - y)^2$$

$$\begin{aligned} L'_{\text{quad}}(\hat{y}, y) &= \frac{\partial L_{\text{quad}}(\hat{y}, y)}{\partial \hat{y}} \\ &= 2(\hat{y} - y) \end{aligned}$$

Negative log likelihood.

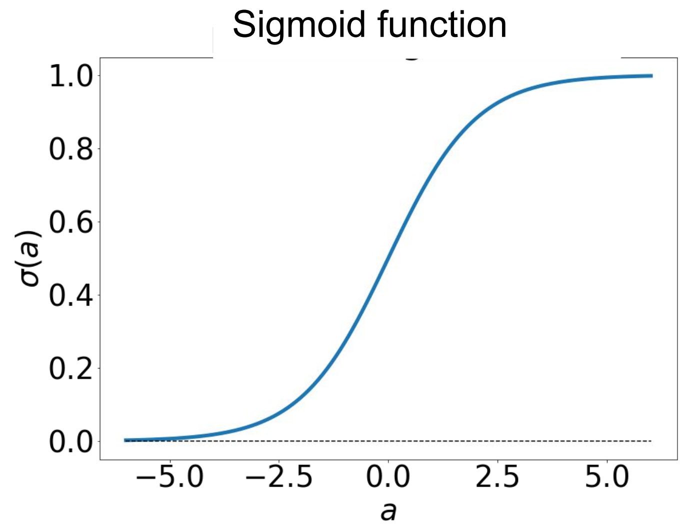
$$L_{\text{nlv}}(\hat{y}, y) = -y \log(\hat{y}) - (1 - y)\log(1 - \hat{y})$$

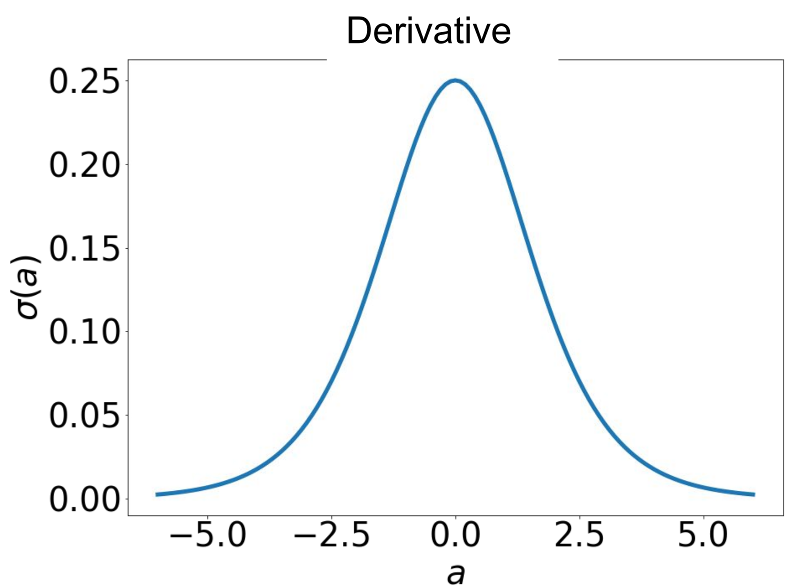
$$\begin{aligned} L'_{\text{nlv}}(\hat{y}, y) &= \frac{\partial L_{\text{nlv}}(\hat{y}, y)}{\partial \hat{y}} \\ &= -\frac{y}{\hat{y}} - \frac{1 - y}{1 - \hat{y}} \end{aligned}$$

Derivative of activation functions.

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

$$\begin{aligned}
\sigma'(a) &= \frac{\partial}{\partial a}(1+e^{-a})^{-1} \\
&= -(1+e^{-a})^{-2}\,\frac{\partial}{\partial a}(1+e^{-a}) \\
&= -\frac{1}{(1+e^{-a})^2}\left[-\frac{\partial}{\partial a}e^{a}\right] \\
&= \frac{e^{a}}{(1+e^{-a})^2}
\end{aligned}$$

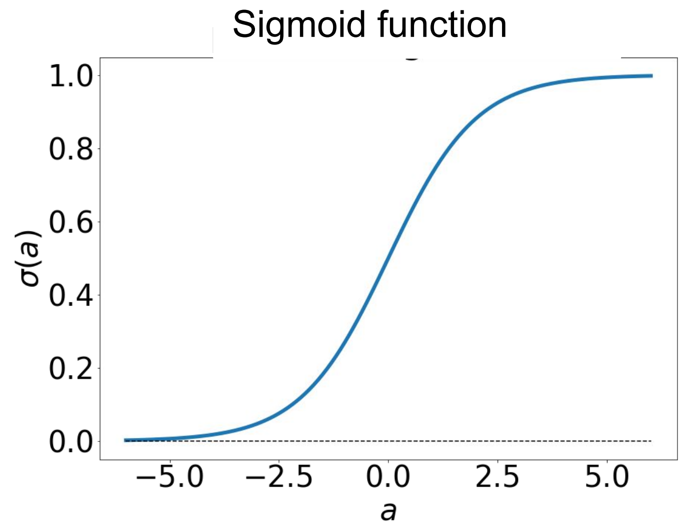


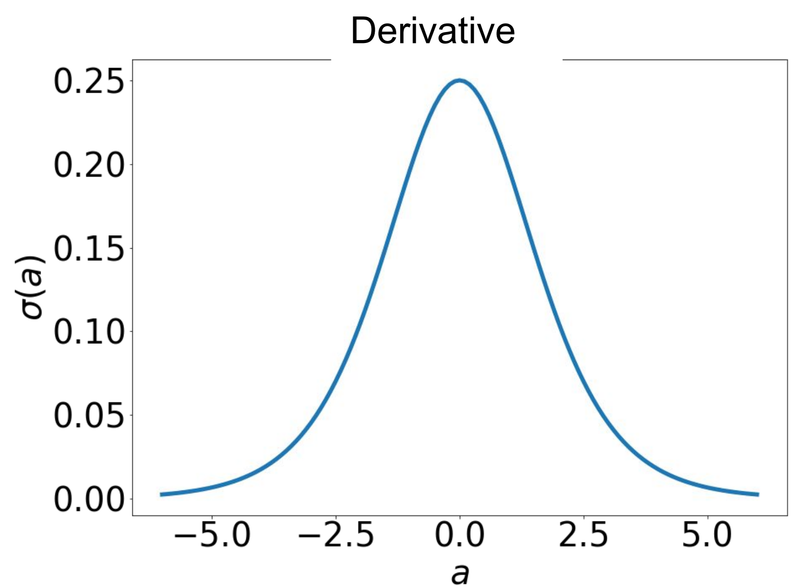


Derivative of activation functions.

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

$$\begin{aligned}
\sigma'(a) &= \frac{\partial}{\partial a}(1+e^{-a})^{-1} \\
&= -(1+e^{-a})^{-2}\,\frac{\partial}{\partial a}(1+e^{-a}) \\
&= -\frac{1}{(1+e^{-a})^2}\left[-\frac{\partial}{\partial a}e^{a}\right] \\
&= \frac{e^{a}}{(1+e^{-a})^2} \\
&= \frac{1+e^{a}}{(1+e^{-a})^2} - \frac{1}{(1+e^{-a})^2} \\
&= \frac{1}{1+e^{-a}} - \left(\frac{1}{1+e^{-a}}\right)^2 \\
&= \sigma(a) - \Big(\sigma(a)\Big)^2 \\
&\boxed{= \sigma(a)\Big(1-\sigma(a)\Big)}
\end{aligned}$$

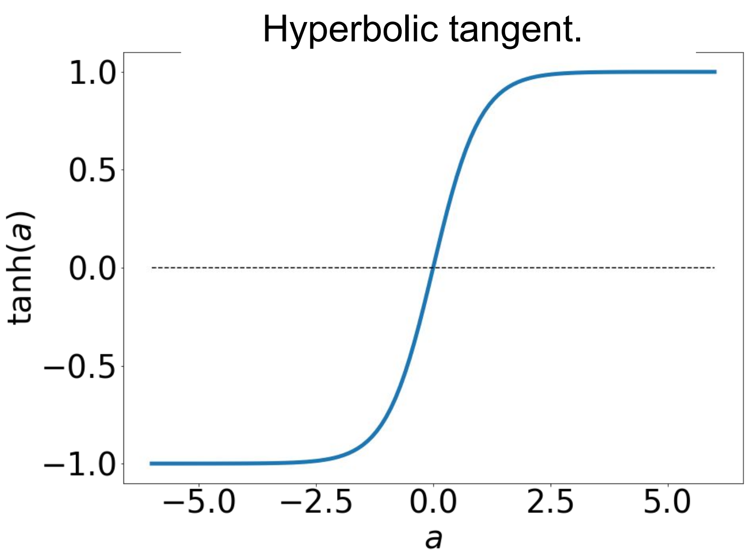


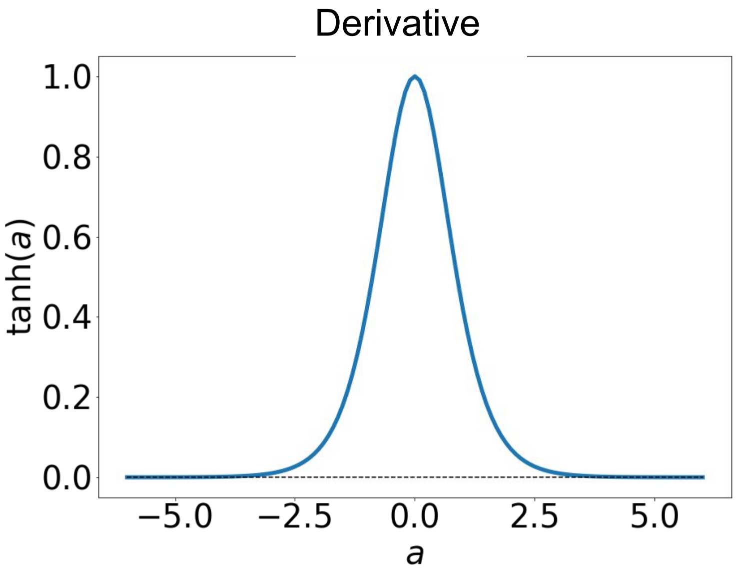


Derivative of activation functions.

$$\tanh(a) = \frac{e^{2a} - 1}{e^{2a} + 1}$$
$$= 2\sigma(2a) - 1$$

$$\tanh'(a) = \frac{\partial \tanh(a)}{\partial a}$$
$$= 4\,\sigma'(2a)$$
$$= 1 - \Big(\tanh(a)\Big)^2$$

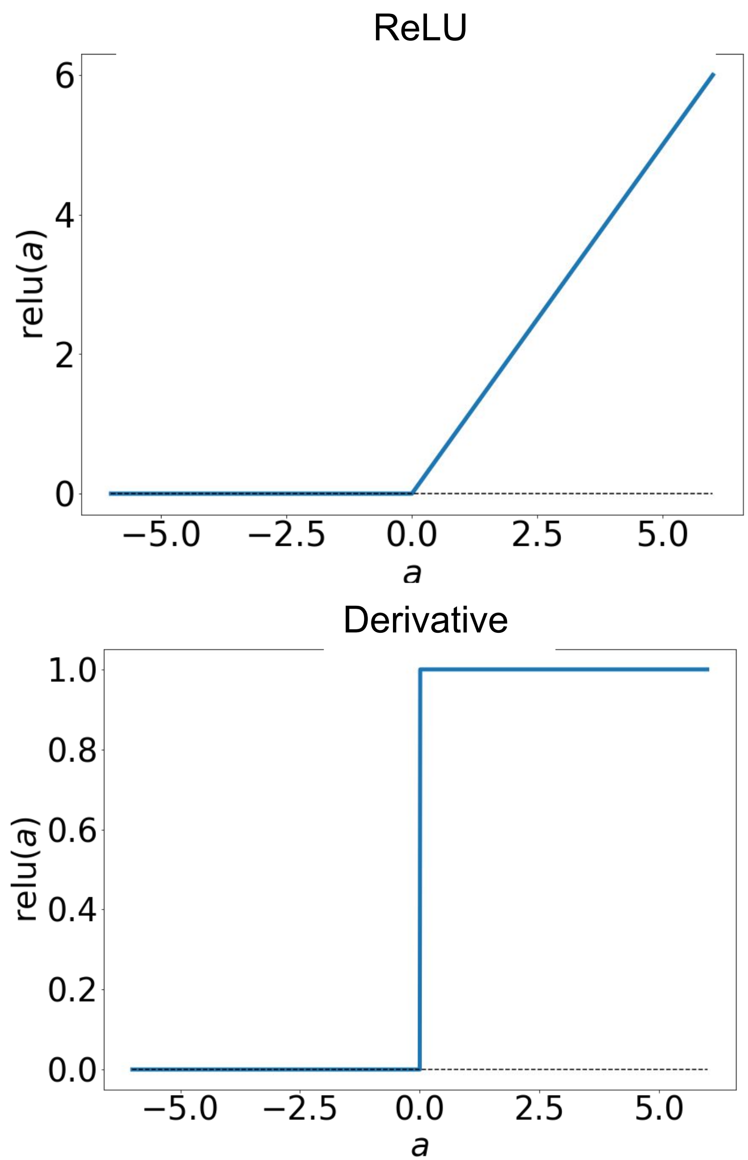




Derivative of activation functions.

$$\mathrm{relu}(a) = \max(0, a)$$

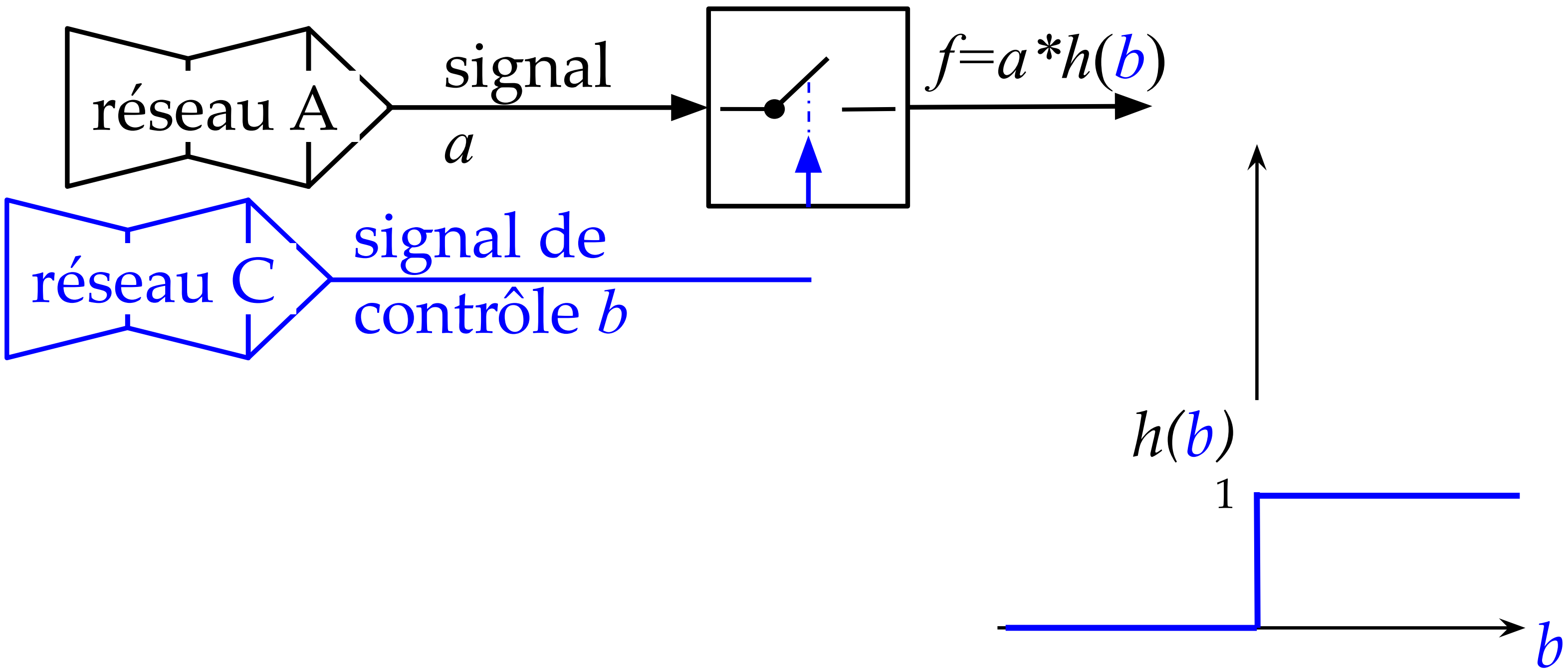
$$\mathrm{relu}'(a) = \frac{\partial\, \mathrm{relu}(a)}{\partial a} = \mathbb{1}_{a>0}$$



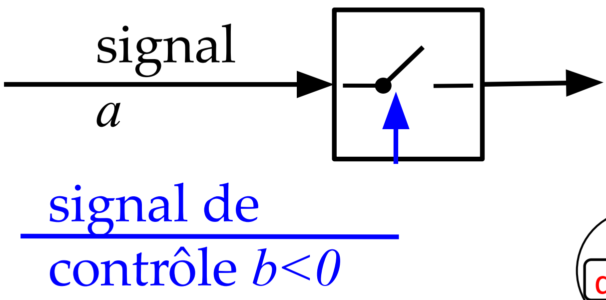
Avertissement!

L’exemple qui suit n’est pas une architecture répendue, mais illustre l’importance de toujours avoir en tête le gradient lorsque l’on désire concevoir des architectures de réseaux de neurones originales.

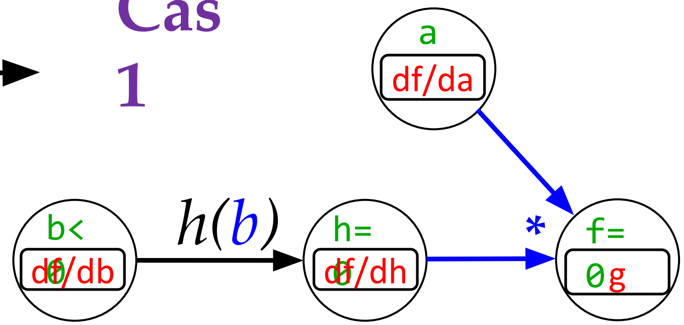
Gradient avec *gate* tout-ou-rien

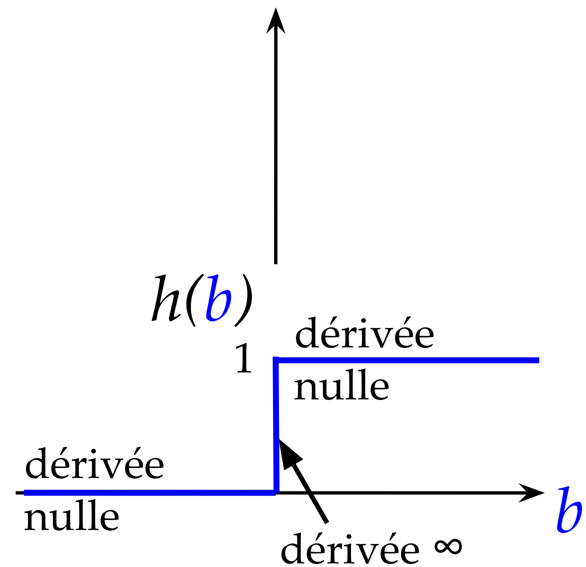


Gradient avec *gate* tout-ou-rien

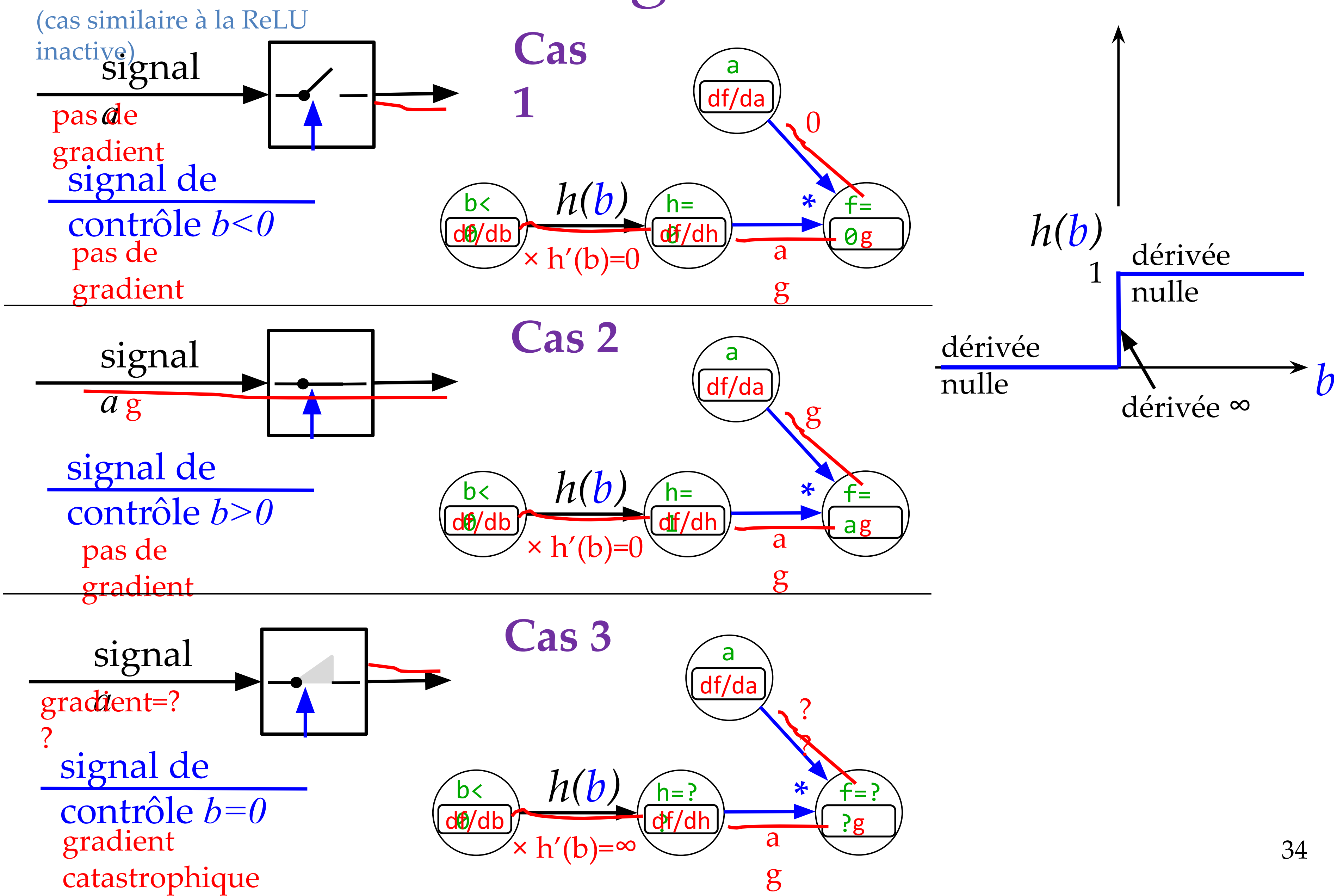


Cas 1

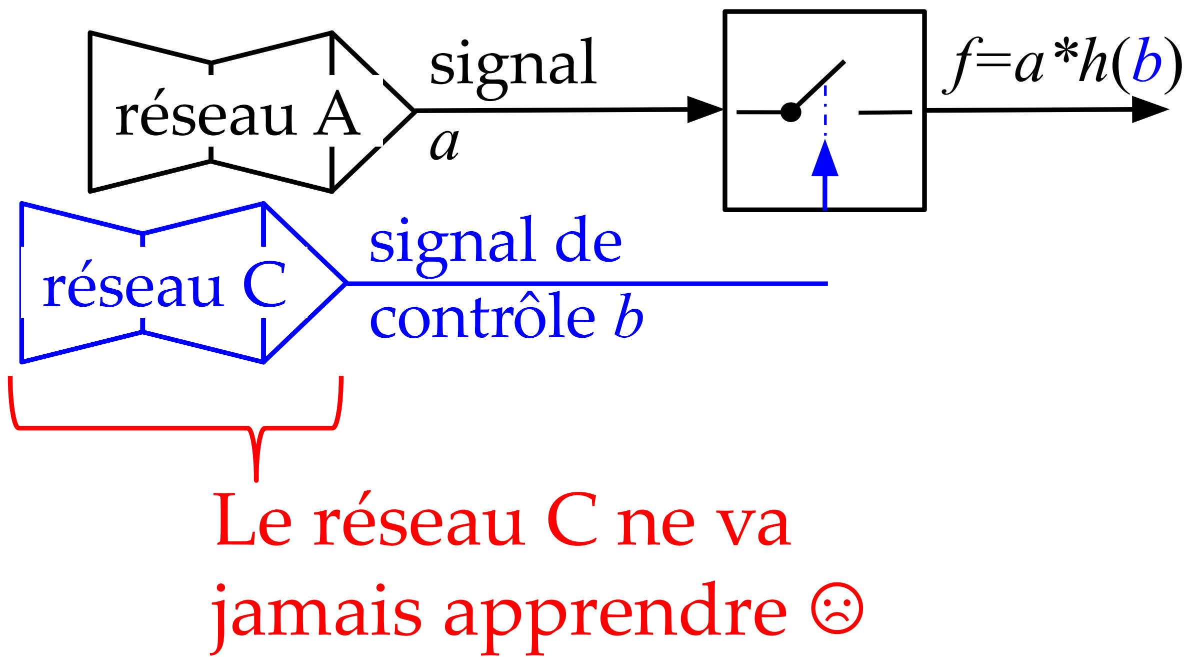




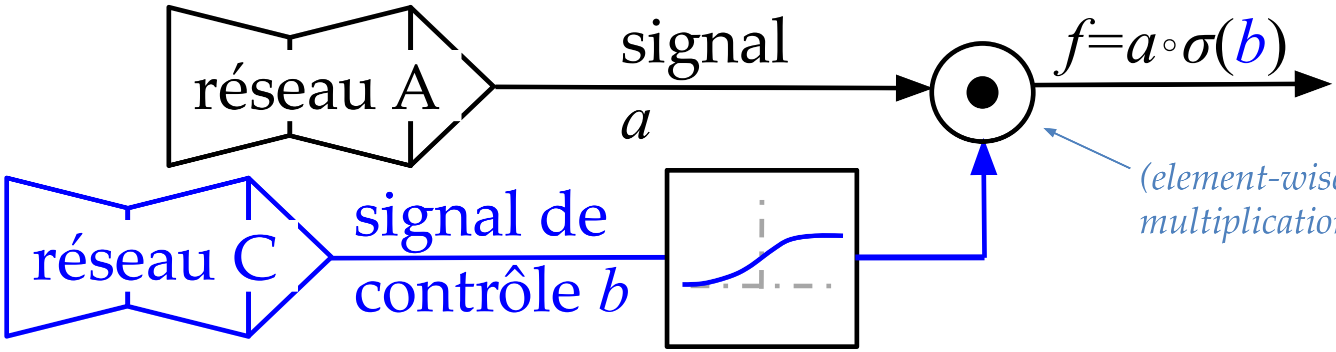
Gradient avec *gate* tout-ou-rien



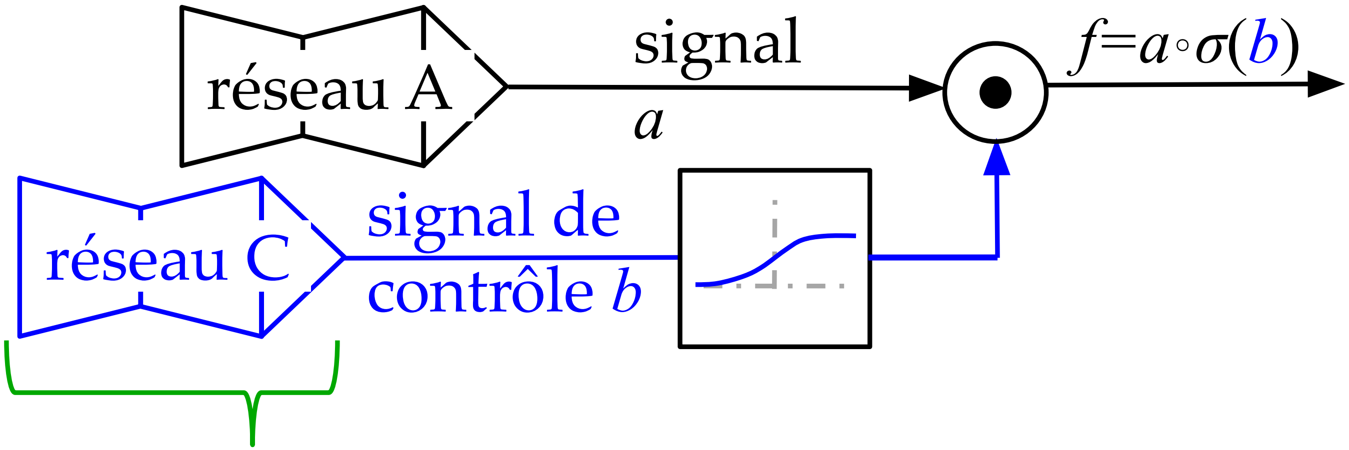
Gradient avec *gate* tout-ou-rien



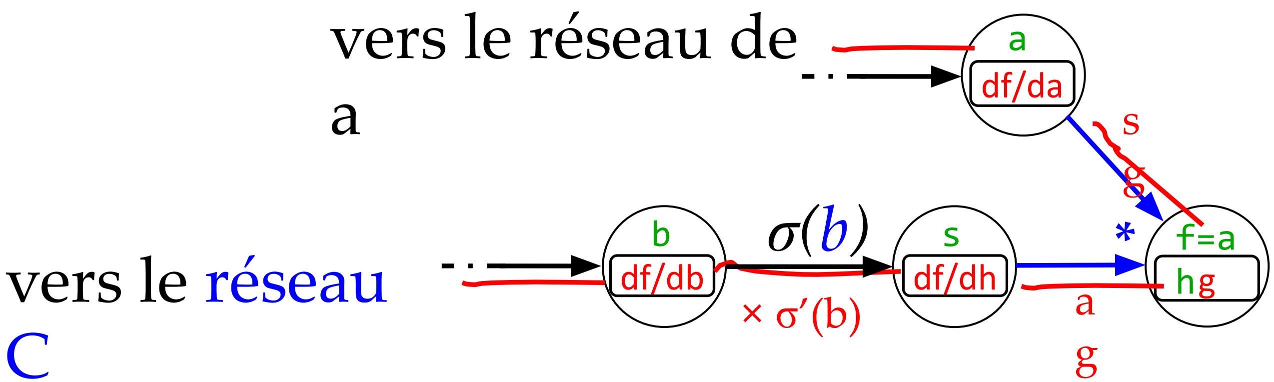
Gradient avec *gate* sigmoïde



Gradient avec *gate* sigmoïde



Ce réseau va apprendre ☺

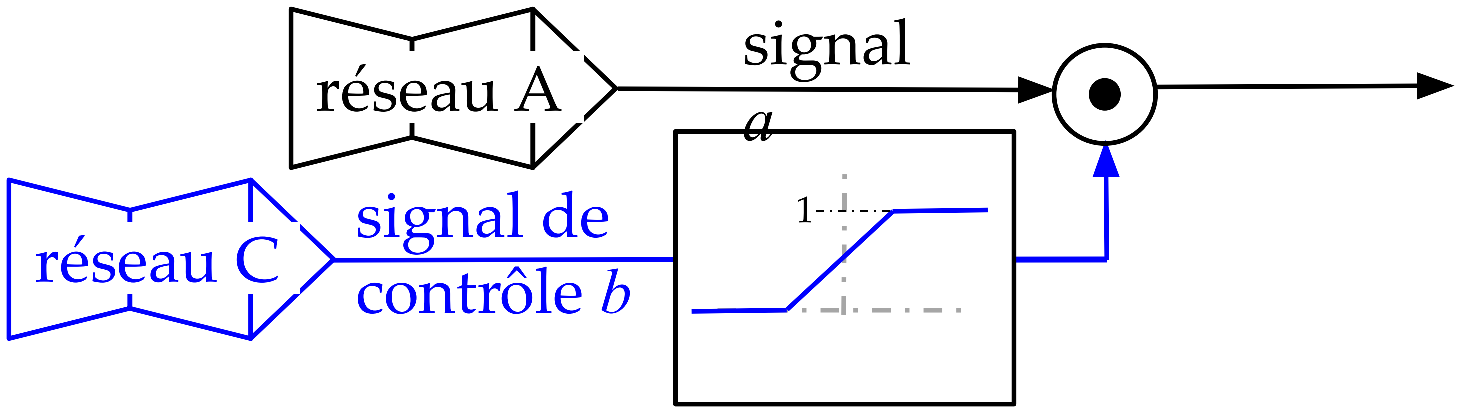


$$\sigma'(b) = \sigma(b)(1 - \sigma(b))$$

Importance d'être gentillement dérivable *end-to-end*

Question!

- Que pensez-vous de cette *gate*?



- On conserve encore le signal sortant entre 0 et *a* (c'est bien)
- Mais le réseau C n'apprendra que si le signal *b* est dans la zone linéaire!