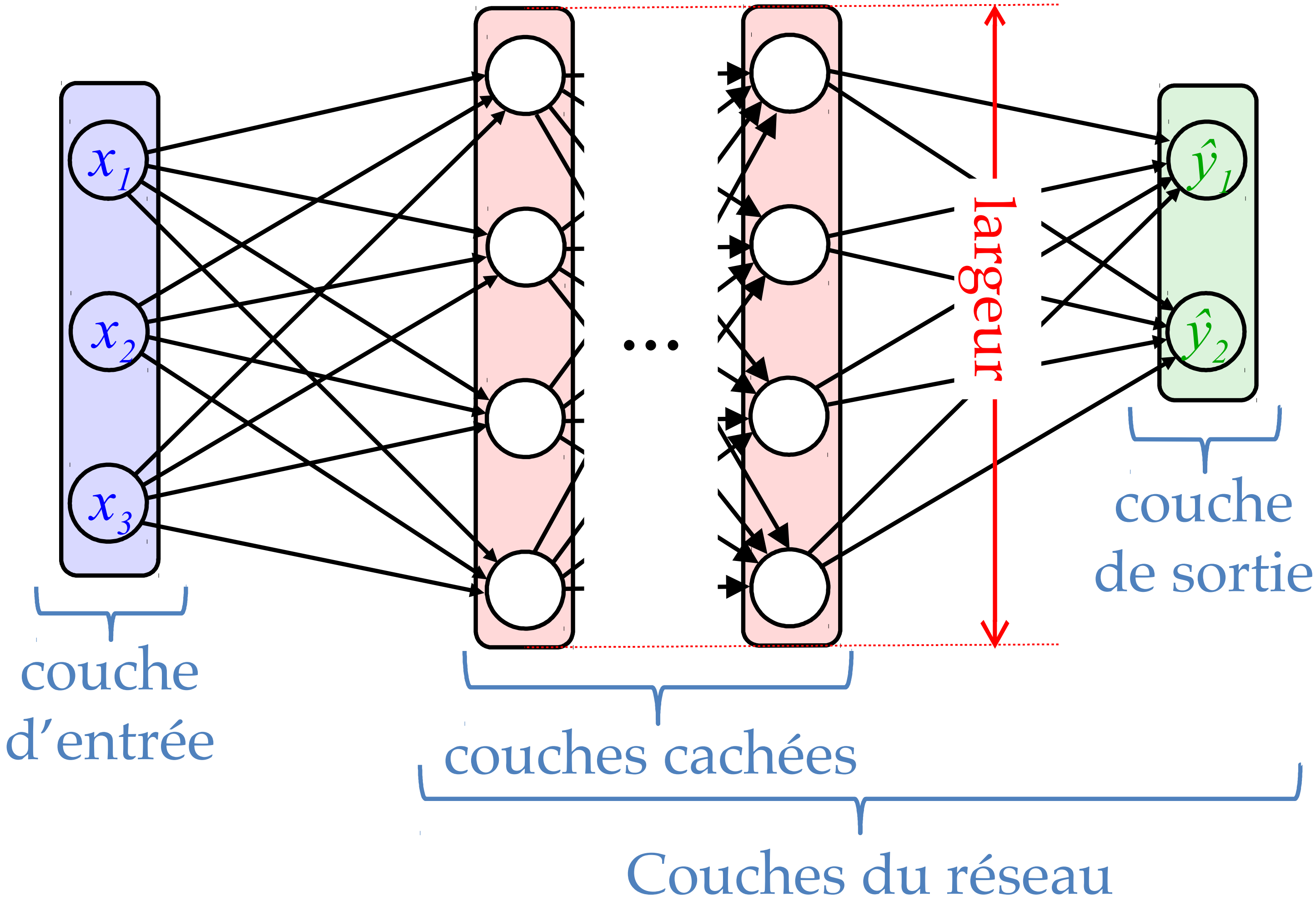
INTRODUCTION AUX RÉSEAUX DE NEURONES

Rétropropagation du gradient

Pascal Germain*, 2018

* Merci spécial à Philippe Giguère pour m'avoir permis de réutiliser une partie de ces transparents.

Illustration et nomenclature



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

Choix à faire

- Architecture
  - # couches
  - # neurones (cachés) par couche
  - type de couche
- Forme de la sortie et fonction de sortie
- Fonction de perte
- Optimiseur
  - et autres « *détails* »

Profil de la fonction de perte $L(\theta)$

Adapté de cs231n

Profil de la fonction de perte $L(\theta)$

Réalité : trouver le fond de la vallée embrumée, à tâtons…

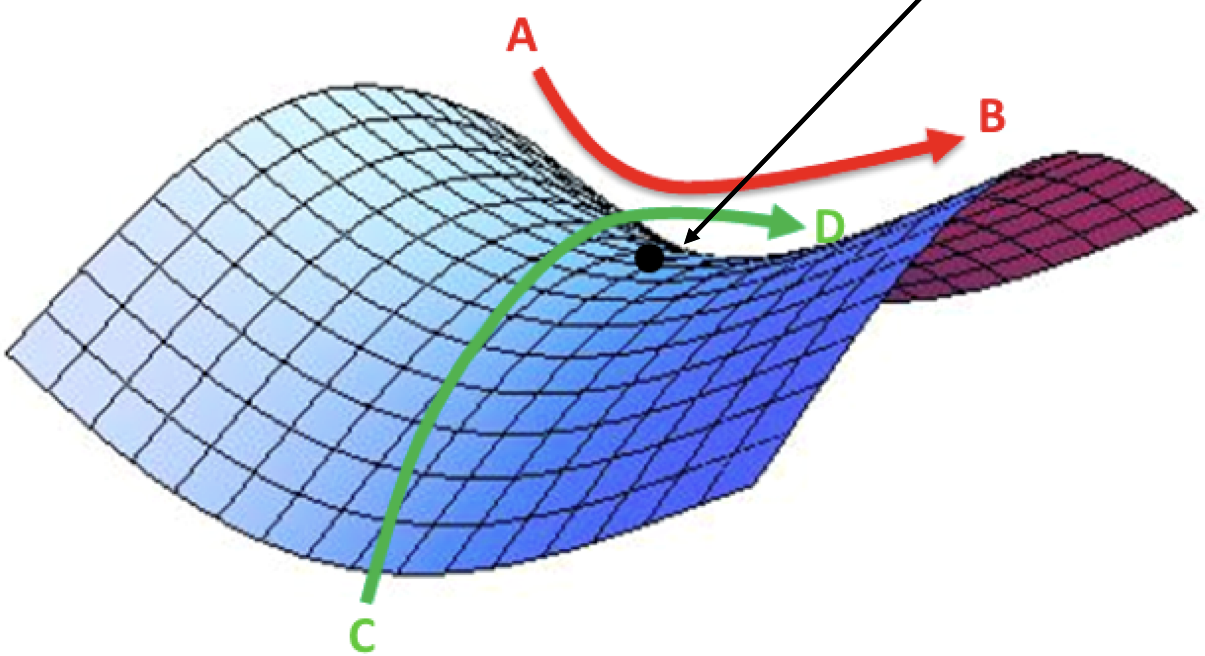
Comparaison avec autres méthodes

- Beaucoup de méthodes d'apprentissage sont convexes
  - Moindre carrés
  - Regression logistique
  - SVM
- Réseaux de neurones sont non-convexes
  - demande un abandon de garanties théoriques
  - va dépendre de l'initialisation
  - peur historique des minimums locaux
  - réalisation graduelle que les solutions sont plus des points de selle
    - ratio (points de selle)/(minimum locaux) augmente exponentiellement avec nombre $|\theta|$ de paramètres

Voir Goodfellow et al. Section 8.2.3

Exemple point de selle

- Dérivées partielles nulles au point de selle



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

Graphes de calculs et algorithme de rétropropagation (`backprop`)

Règle de dérivation en chaîne

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

Par exemple:

$$\begin{aligned}F(x) &= (2x+3)^2 \\ &= f(2x+3) \qquad \text{où } f(x) = x^2 \\ &= f(h(x)) \qquad \text{où } h(x) = 2x+3.\end{aligned}$$

Donc :

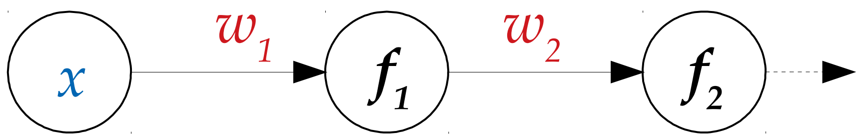
$$\begin{aligned}\frac{\partial F(x)}{\partial x} &= \frac{\partial f(h(x))}{\partial h(x)} \frac{\partial h(x)}{\partial x} \\ &= 2\,h(x) \times 2 \\ &= 4(2x+3)\end{aligned}$$

Règle de dérivation en chaîne

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

On écrit aussi: $$(f \circ h)' = (h' \circ f) \times f'$$

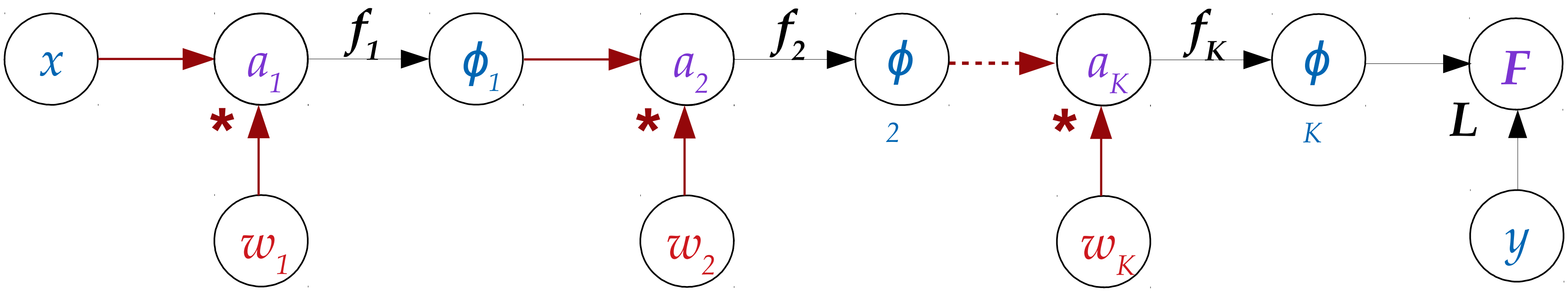
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial h} \frac{\partial h}{\partial x}$$



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

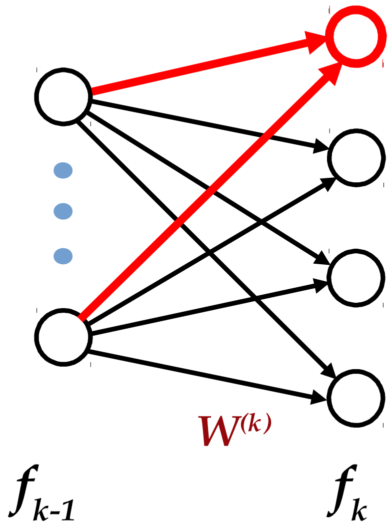
$$R(x) = f_2(w_2 \cdot f_1(w_1 \cdot x))$$

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

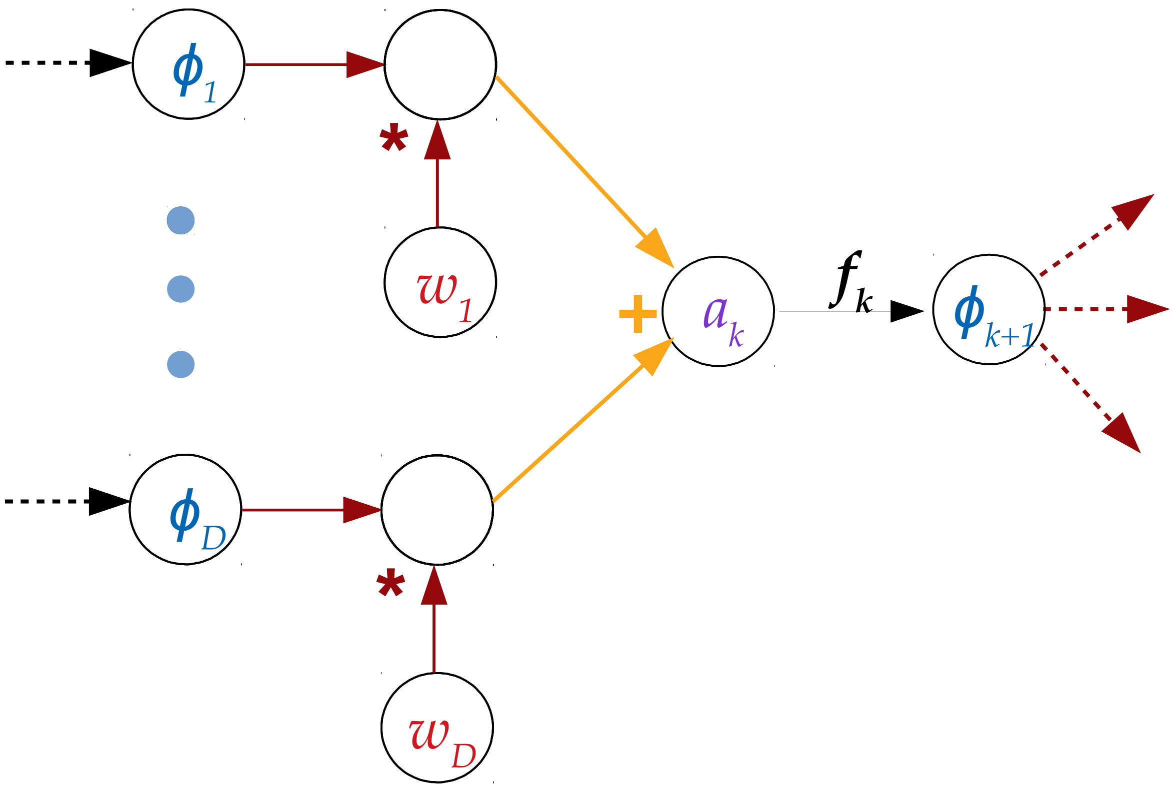


- $\phi_0 = x$
- Pour $k = 1, 2, \dots K$:
  - $a_k = w_k \cdot \phi_{k-1}$
  - $\phi_k = f_k(a_k)$
- $F \;= L(\phi_k, y)$

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



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



Soit un réseau R de K couches:

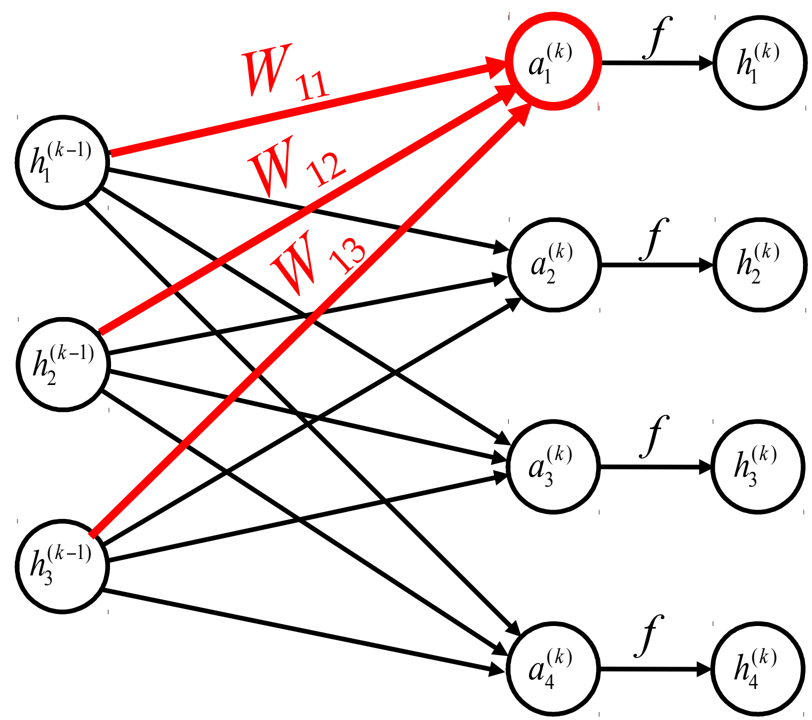
- Fonctions d'activations : $f_1, \ldots f_K$
- Matrices de poids: $\mathbf{W}^{(1)}, \ldots \mathbf{W}^{(K)}$
  - Chaque matrice $\mathbf{W}^{(k)}$ est de taille $d_k \times d_{k-1}$
  - d_k est le nombre de neurones sur la couche k
  - $k = 0$ correspond à la couche d'entrée: $\mathbf{x} \in \mathbb{R}^{d_0}$

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

Algorithme de propagation avant.
ENTRÉES: Réseau R, Observation $\mathbf{x}$

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

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



Algorithme de propagation avant.
ENTRÉES: Réseau R, Observation $\mathbf{x}$

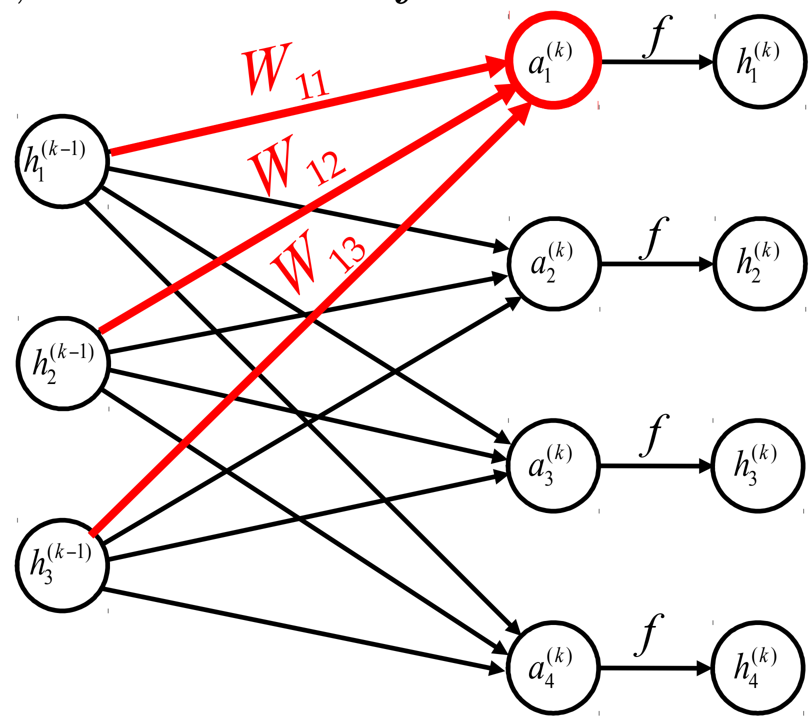
- $\mathbf{h}[0] \leftarrow \mathbf{x}$
- Pour k de 1 à K:
  - $\mathbf{a}[k] \leftarrow \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}$
  - $\mathbf{h}[k] \leftarrow f_k(\mathbf{a}[k])$

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

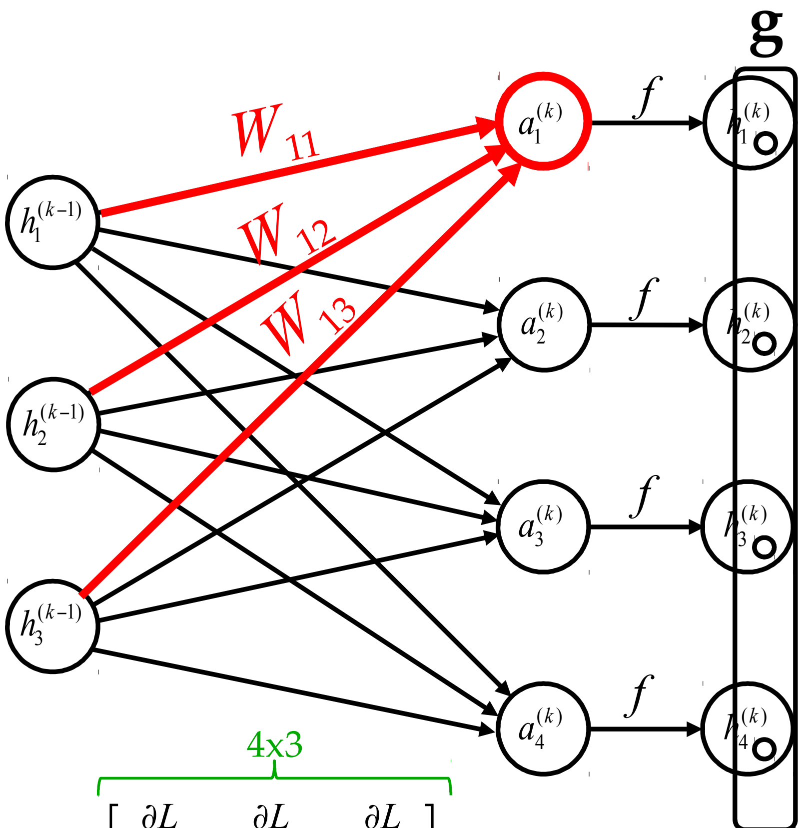
Algorithme de retropropagation.
ENTRÉES: Réseau R, Perte L, Observation $\mathbf{x}$, Sortie attendue $\mathbf{y}$

- $\mathbf{g} \leftarrow L'(h[K], \mathbf{y})$
- Pour k décroissant de K à 1:
  - $\mathbf{g} \leftarrow \mathbf{g} \odot f_k'(\mathbf{a}[k])$
  - $\nabla_{\mathbf{W}}[k] \leftarrow \mathbf{g}\,\mathbf{h}[k]^T$
  - $\mathbf{g} \leftarrow \mathbf{W}^{(k)T}\mathbf{g}$

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



Backprop couche quelconque



$$a^{(k)} = \overbrace{W^{(k)}}^{4\text{x}3} h + b^{(k)}$$

$$a_1^{(k)} = W_{11}h_1^{(k-1)} + W_{12}h_2^{(k-1)} + W_{13}h_3^{(k-1)}$$

Gradient $\nabla_W L$ des poids (comment les poids W affectent la perte L)

$$g \leftarrow g \odot f'(a^{(k)})$$

$$\frac{\partial L}{\partial W_{11}} = \frac{\partial a_1^{(k)}}{\partial W_{11}} \frac{\partial L}{\partial a_1^{(k)}} = g_1 h_1^{(k-1)}$$

$$\frac{\partial L}{\partial W_{12}} = \frac{\partial a_1^{(k)}}{\partial W_{12}} \frac{\partial L}{\partial a_1^{(k)}} = g_1 h_2^{(k-1)}$$

$$\frac{\partial L}{\partial W_{np}} = \frac{\partial a_n^{(k)}}{\partial W_{np}} \frac{\partial L}{\partial a_n^{(k)}} = g_n h_p^{(k-1)}$$

$$\nabla_W L = \overbrace{\begin{bmatrix} \frac{\partial L}{\partial W_{11}} & \frac{\partial L}{\partial W_{12}} & \frac{\partial L}{\partial W_{13}} \\ \frac{\partial L}{\partial W_{21}} & \frac{\partial L}{\partial W_{22}} & \frac{\partial L}{\partial W_{23}} \\ \frac{\partial L}{\partial W_{31}} & \frac{\partial L}{\partial W_{32}} & \frac{\partial L}{\partial W_{33}} \\ \frac{\partial L}{\partial W_{41}} & \frac{\partial L}{\partial W_{42}} & \frac{\partial L}{\partial W_{43}} \end{bmatrix}}^{4\text{x}3} = \begin{bmatrix} g_1 h_1^{(k-1)} & g_1 h_2^{(k-1)} & g_1 h_3^{(k-1)} \\ g_2 h_1^{(k-1)} & g_2 h_2^{(k-1)} & g_2 h_3^{(k-1)} \\ g_3 h_1^{(k-1)} & g_3 h_2^{(k-1)} & g_3 h_3^{(k-1)} \\ g_4 h_1^{(k-1)} & g_4 h_2^{(k-1)} & g_4 h_3^{(k-1)} \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \begin{bmatrix} h_1^{(k-1)} & h_2^{(k-1)} & h_3^{(k-1)} \end{bmatrix} = g h^{(k-1)T}$$

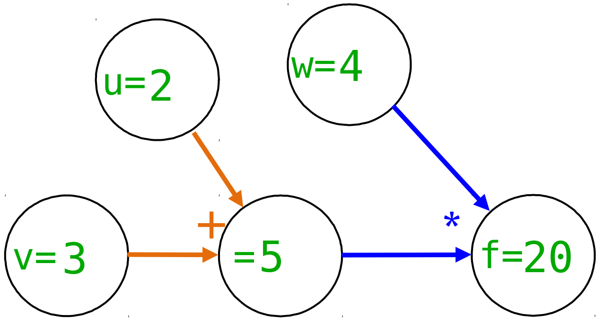
Backprop et dérivation automatique

- Algorithme qui calcule tous les gradients dans un graphe de calcul
- **N'est pas l'algo d'optimisation!**
- Mais tous les algos d'optimisation des réseaux de neurones utilisent les gradients calculés par *backprop*
- Basé sur la règle de dérivation en chaîne
- Les librairies modernes de réseau de neurones effectuent le calcul des dérivés automatiquement (comme pyTorch)

Exemple graphe calcul

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

nœud : variable
arête : opération

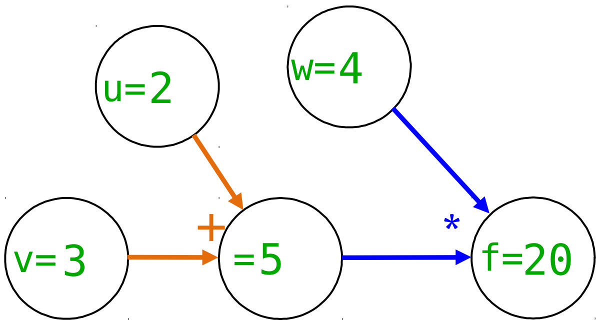


Instanciation des les variables:
u=2
v=3
w=4

Évalue le graphe pour avoir f : **Propagation avant** (forward pass)

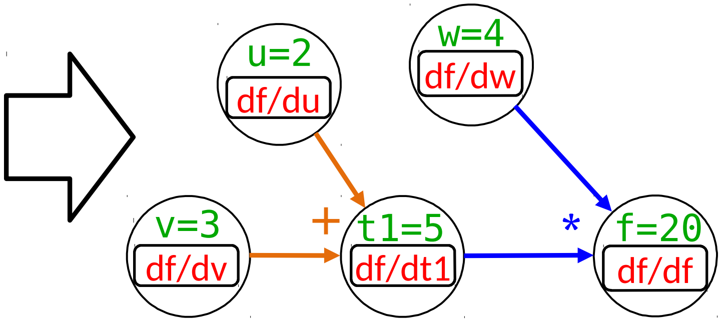
Exemple sur graphe calcul simple

Part d'un graphe
de calcul évalué :



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

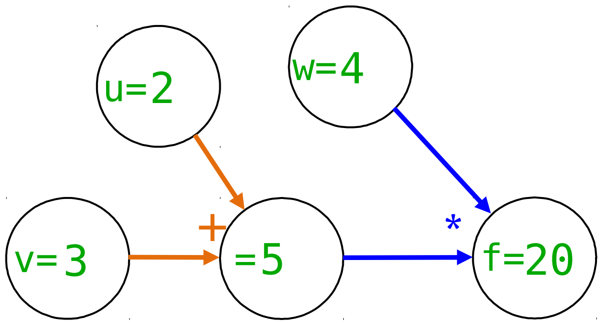
Ajouter une case pour
stocker les gradients :



(Rappel : on cherche la sensibilité de la sortie en fonction des variables du graphe)

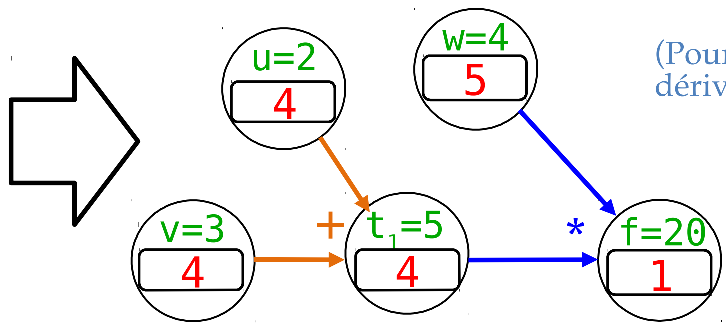
Exemple sur graphe calcul simple

Part d'un graphe de calcul évalué :



$$f=(u+v)w$$
$$f=t_1w,\ t_1=u+v$$

Ajouter une case pour stocker les gradients :



(Pourquoi /df? On cherche les dérivées partielle p.r. à la sortie, f)

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

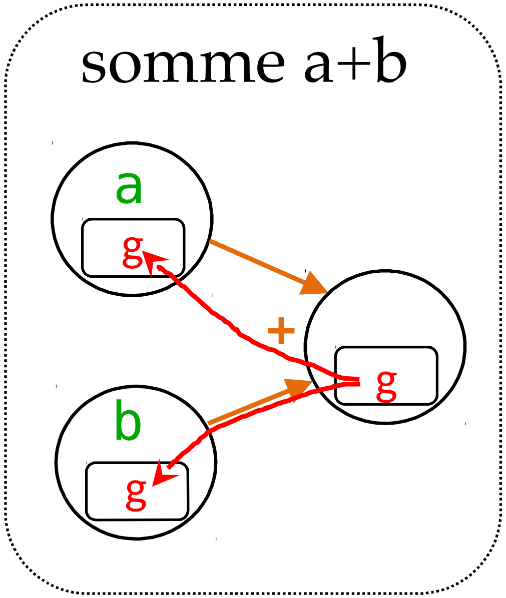
$$\frac{\partial f}{\partial w}=\frac{\partial}{\partial w}(t_1w)\frac{\partial f}{\partial f}=t_1\cdot 1$$

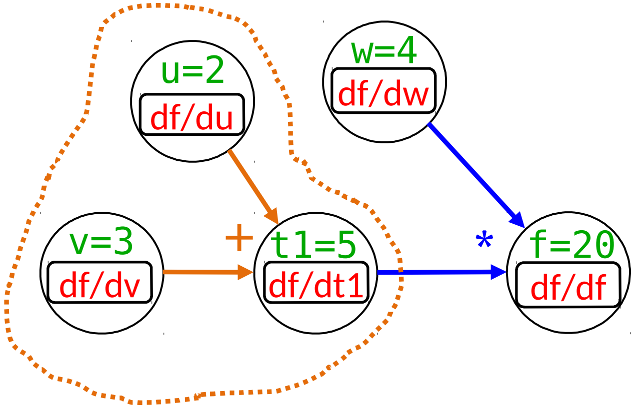
$$\frac{\partial f}{\partial t_1}=\frac{\partial}{\partial t_1}(t_1w)\frac{\partial f}{\partial f}=w\cdot 1$$

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

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

Tirer des règles de base

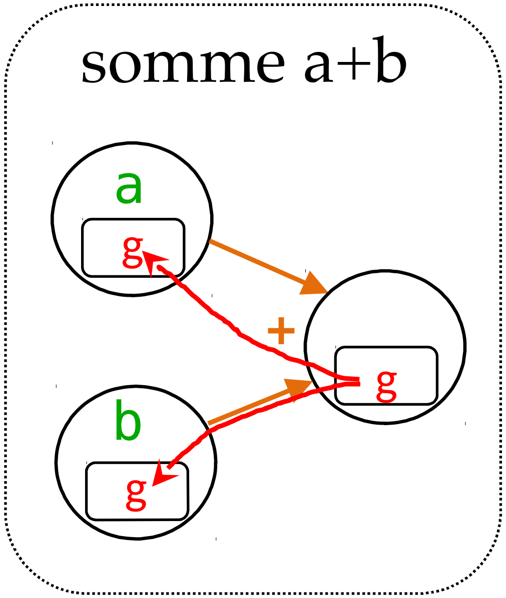


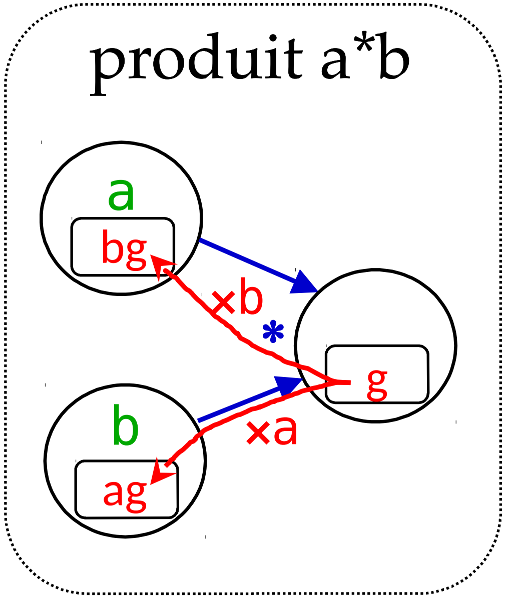


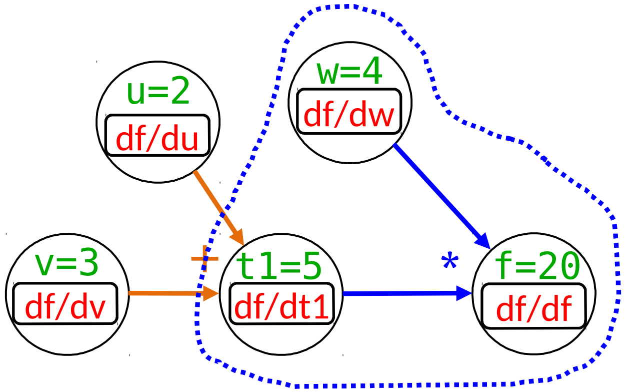
$$\frac{\partial f}{\partial w} = \frac{\partial}{\partial w}(t_1 w)\frac{\partial f}{\partial f} = t_1 \cdot 1 \qquad \frac{\partial f}{\partial t_1} = \frac{\partial}{\partial t_1}(t_1 w)\frac{\partial f}{\partial f} = w \cdot 1$$

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

Tirer des règles de base







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

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

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

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

Dérivées de fonctions de perte

Perte quadratique.

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

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

Perte négatif log vraisemblance.

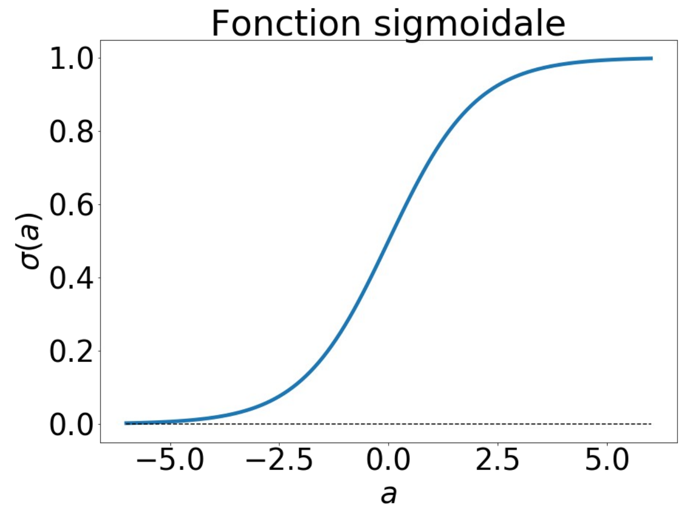
$$L_{\text{nlv}}\big(\hat{y}, y\big) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

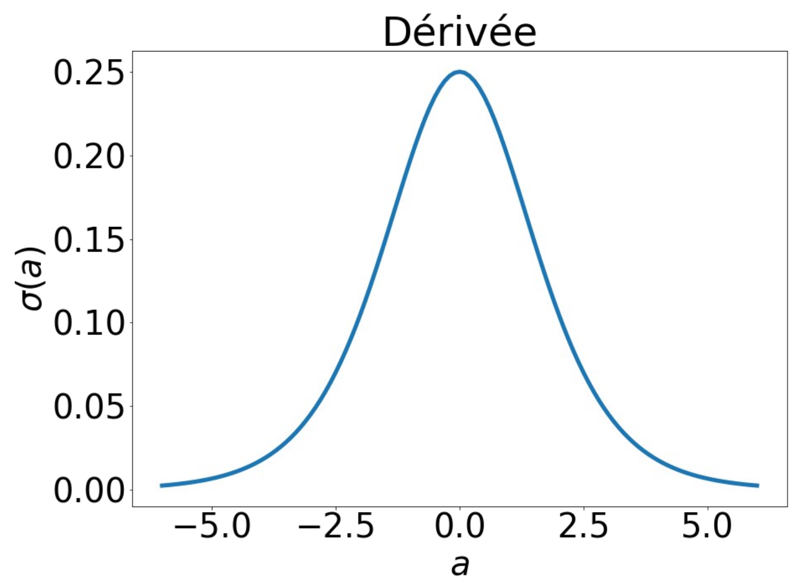
$$\begin{aligned} L'_{\text{nlv}}\big(\hat{y}, y\big) &= \frac{\partial L_{\text{nlv}}(\hat{y}, y)}{\partial \hat{y}} \\ &= -\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} \end{aligned}$$

Dérivées de fonctions d'activation

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

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

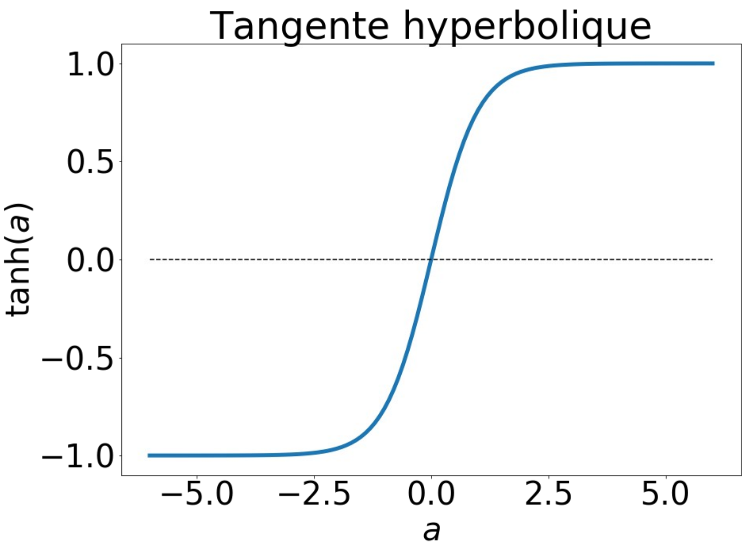


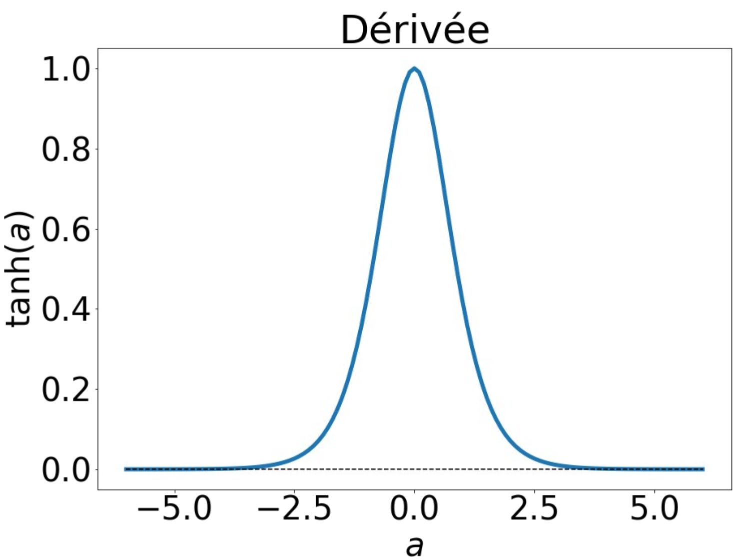


Dérivées de fonctions d'activation

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

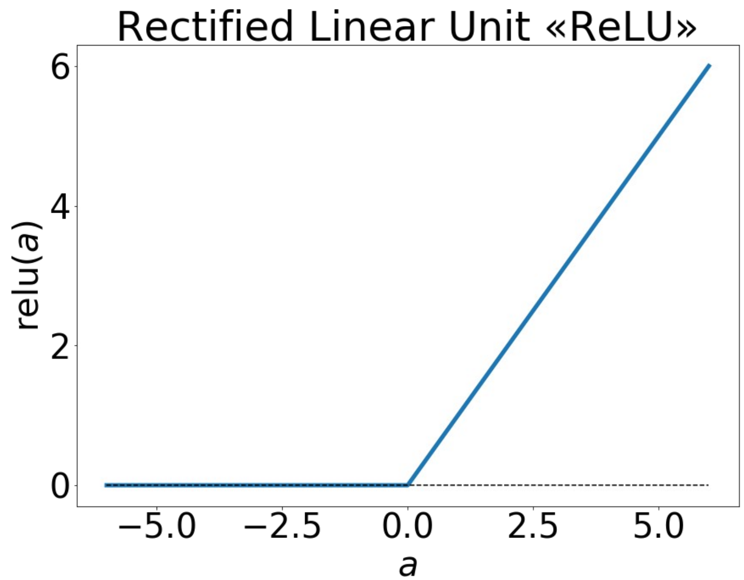
$$\tanh'(a) = \frac{\partial \tanh(a)}{\partial a}$$
$$= 4\,\sigma'(2a)$$
$$= 1 - \Big(\tanh(a)\Big)^2$$



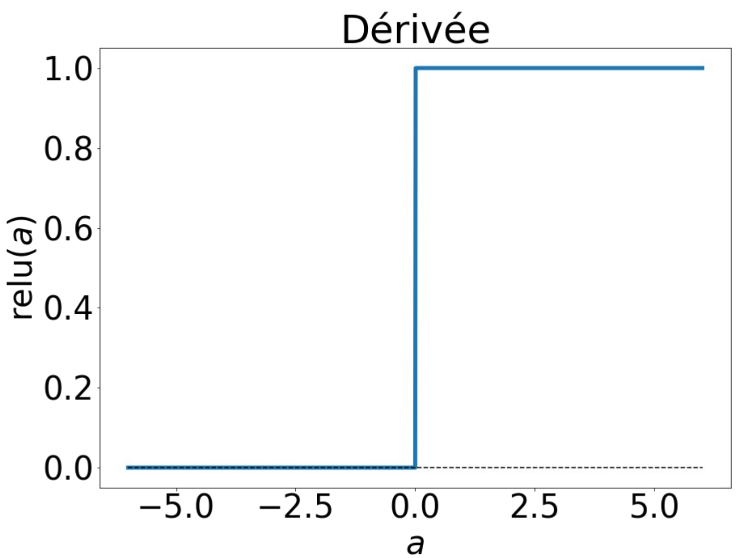


Dérivées de fonctions d’activation

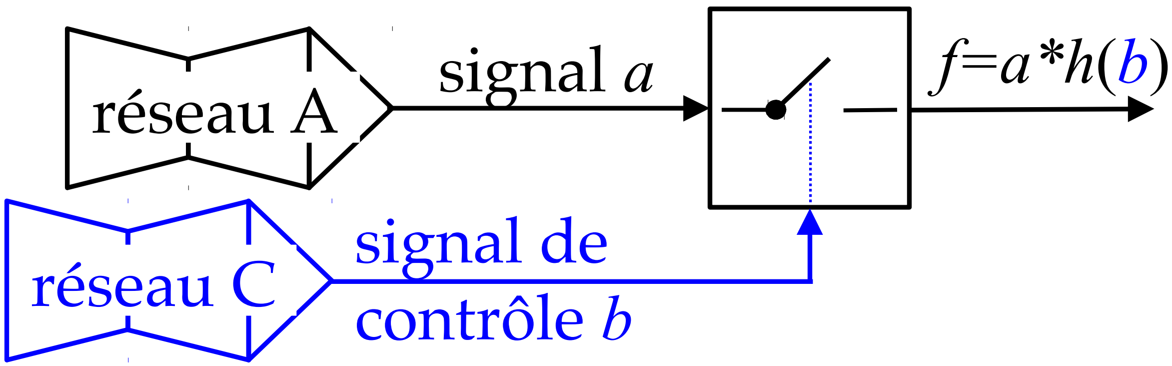
$$\mathrm{relu}(a) = \max(0, a)$$

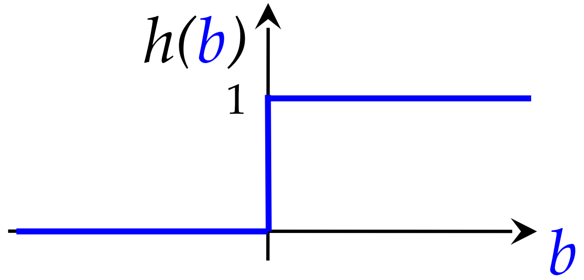


$$\mathrm{relu}'(a) = \frac{\partial\, \mathrm{relu}(a)}{\partial a}$$
$$= \mathbb{1}_{a>0}$$

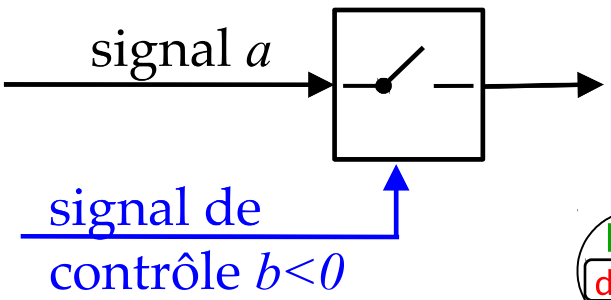


Gradient avec *gate* tout-ou-rien

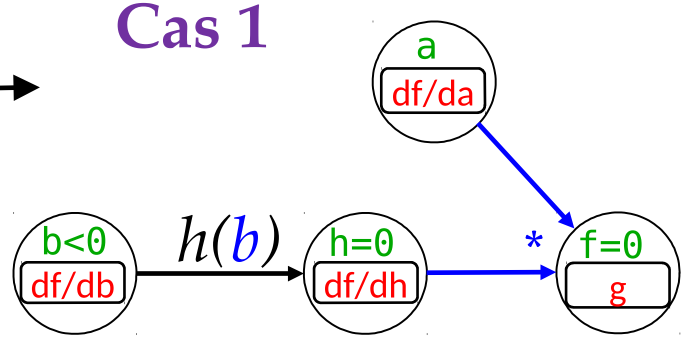


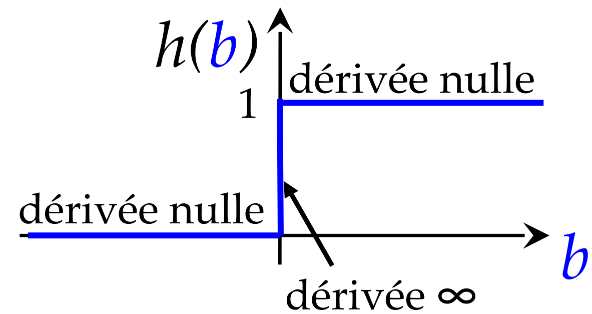


Gradient avec *gate* tout-ou-rien



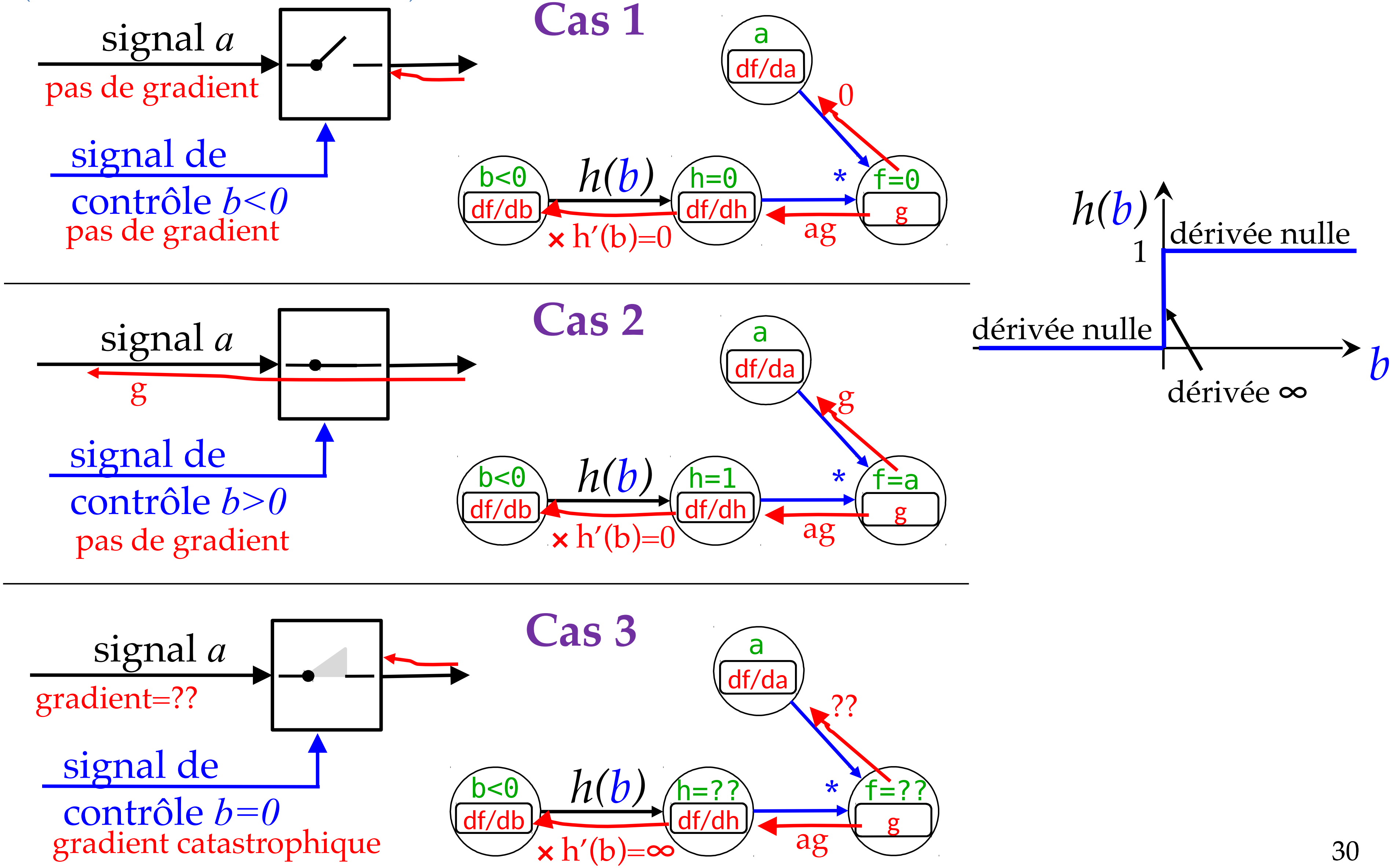
Cas 1



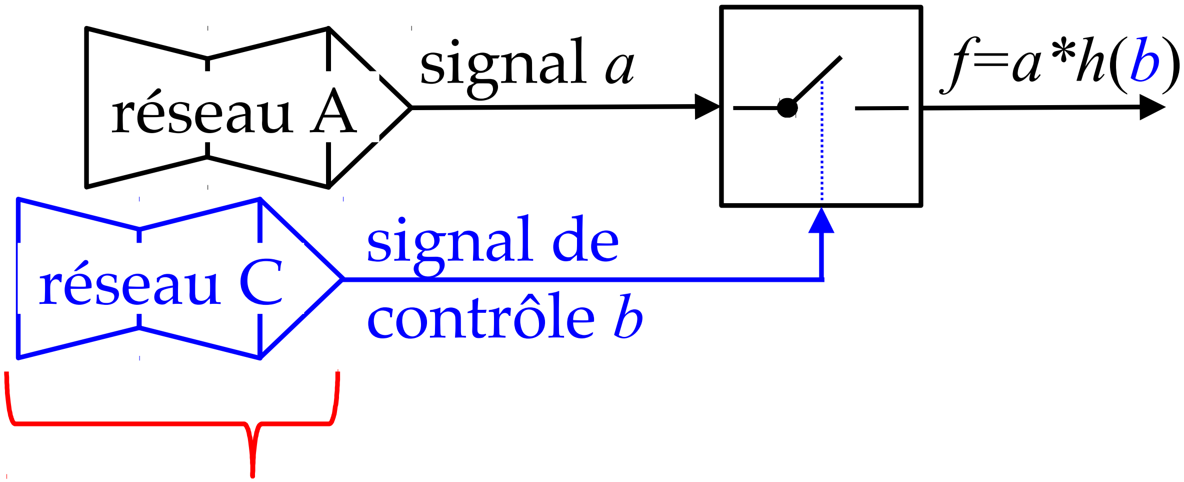


Gradient avec *gate* tout-ou-rien

(cas similaire à la ReLU inactive)

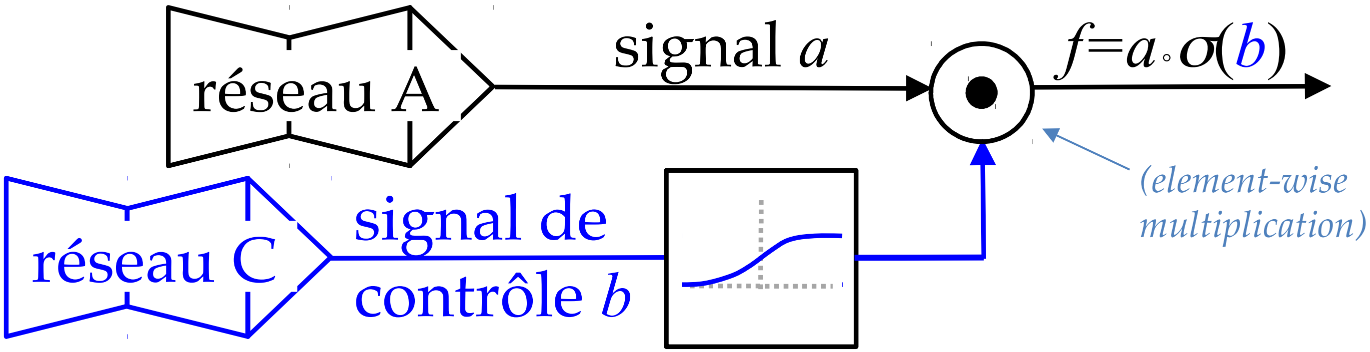


Gradient avec *gate* tout-ou-rien

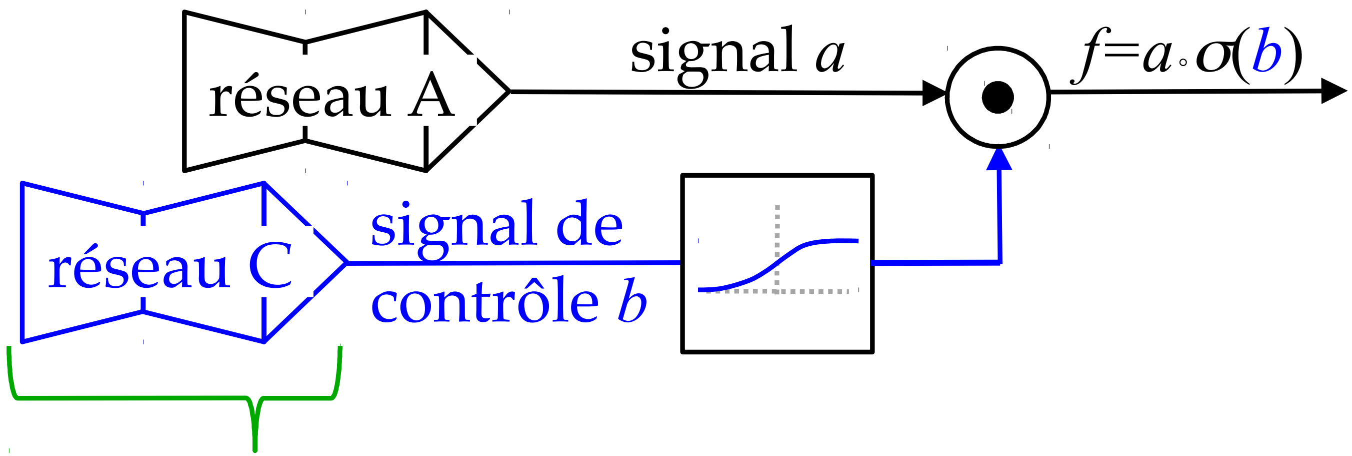


Le réseau C ne va
jamais apprendre ☹

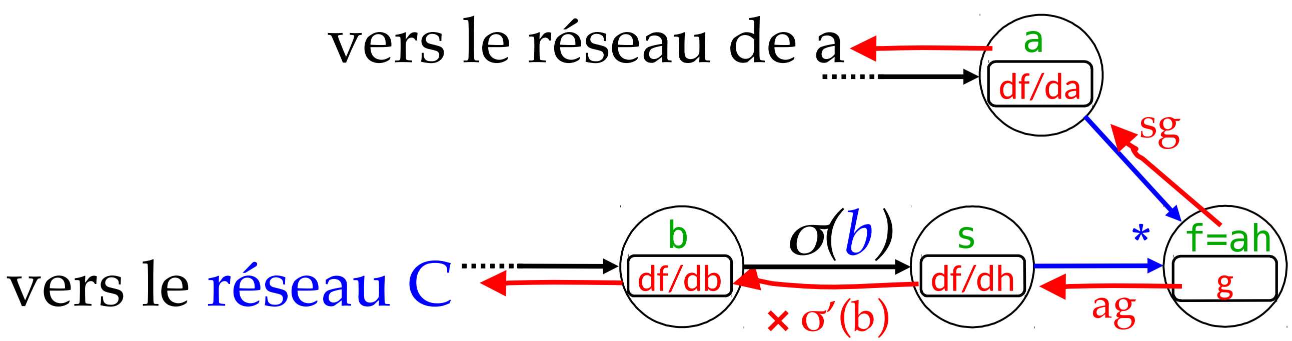
Gradient avec *gate* sigmoïde



Gradient avec *gate* sigmoïde



Ce réseau va apprendre ☺



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

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