

Introduction à l'apprentissage profond

Y. Pradat (*auteur*), L. Verlingue & D. Gautheret (*responsables cours*)

CentraleSupélec (labo MICS) & Institut Gustave Roussy (IGR)

24 janvier 2020



CentraleSupélec



1 Notations

2 Le perceptron

- Modèle à un neurone
- Adaline

3 Modèles multi-couches

- De l'entrée à la sortie
- La rétropropagation

4 Réseaux convolutionnels

Notations

Notations

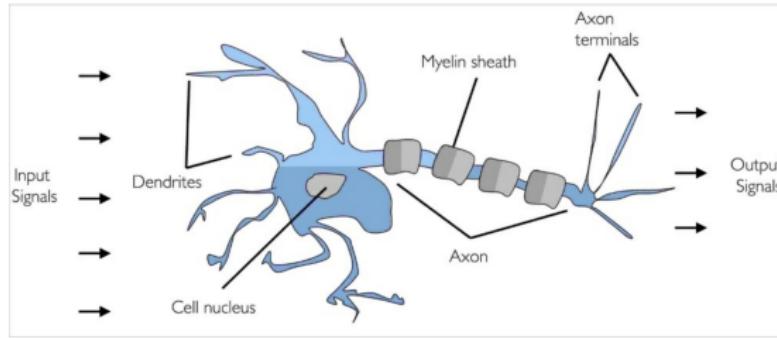
- $n, p \in \mathbb{N}^*$ nombre d'observations (unités, individus) et nombre de variables (facteurs, cofacteurs) ;
- \mathbf{X} variable aléatoire vectorielle ;
- $\mathbf{X}_{1:n}$ matrice des variables aléatoires $\mathbf{X}_1, \dots, \mathbf{X}_n$;
- x observation de \mathbf{X} ;
- $x_{1:n}$ observation de $\mathbf{X}_{1:n}$.

Objectifs :

- ① Proposer un modèle mathématique ;
- ② Estimer les paramètres du modèle.

Le neurone artificiel

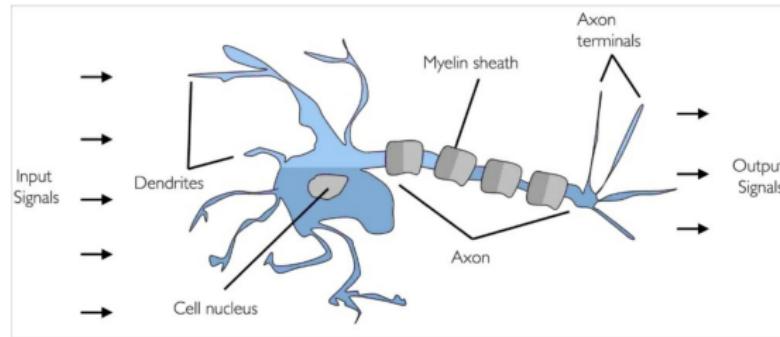
Un neurone biologique : si somme des signaux entrées > seuil : potential d'action généré ; sinon inactif.



source : S. Raschka,
Python for Machine Learning, 2017.

Le neurone artificiel

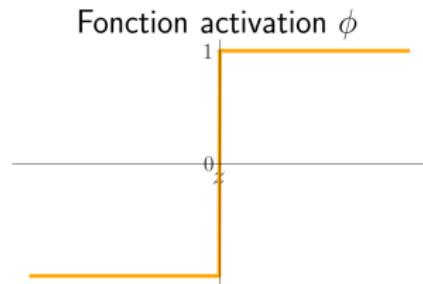
Un neurone biologique : si somme des signaux entrées > seuil : potentiel d'action généré ; sinon inactif.



source : S. Raschka,
Python for Machine Learning, 2017.

Un neurone artificiel. Rosenblatt, 1957.

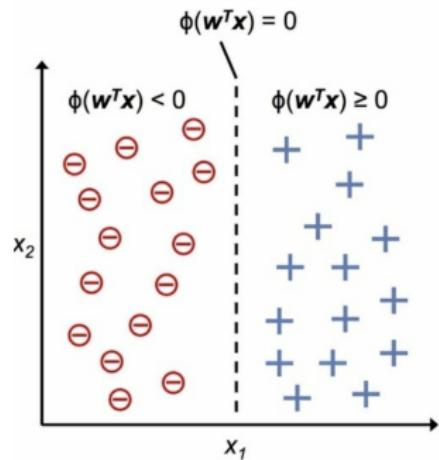
- Entrées $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$, poids $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$
- Entrées agrégées $z = \mathbf{x}^\top w$
- Activation $\phi(z)$.



La règle perceptron

Modèle de classification covariables $\mathbf{x}_{1:n}$, classes $y_{1:n}$.

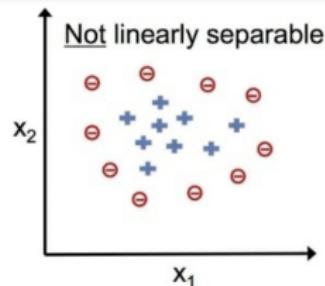
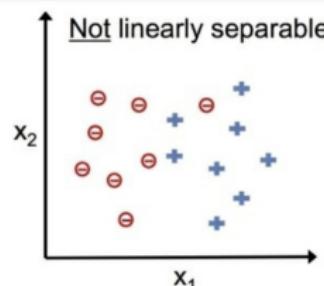
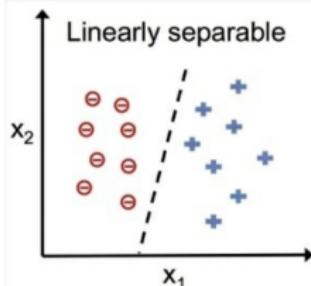
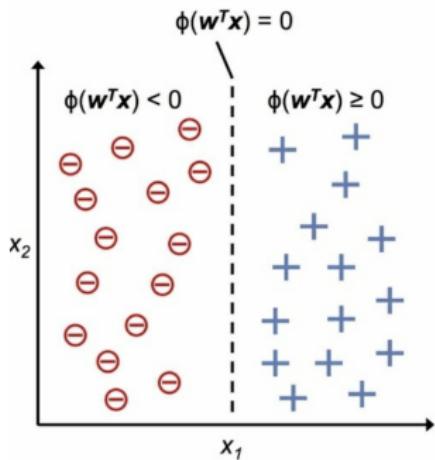
- ➊ initialisation w aléatoire ;
- ➋ pour $k = 1, \dots, k$,
 - ➌ pour chaque $i = 1, \dots, n$,
 - ➍ $\hat{y}_i = \phi(\mathbf{x}_i^\top w)$
 - ➎ $\delta = \eta(y_i - \hat{y}_i)$.
 - ➏ $w = w + \delta \mathbf{x}_i$



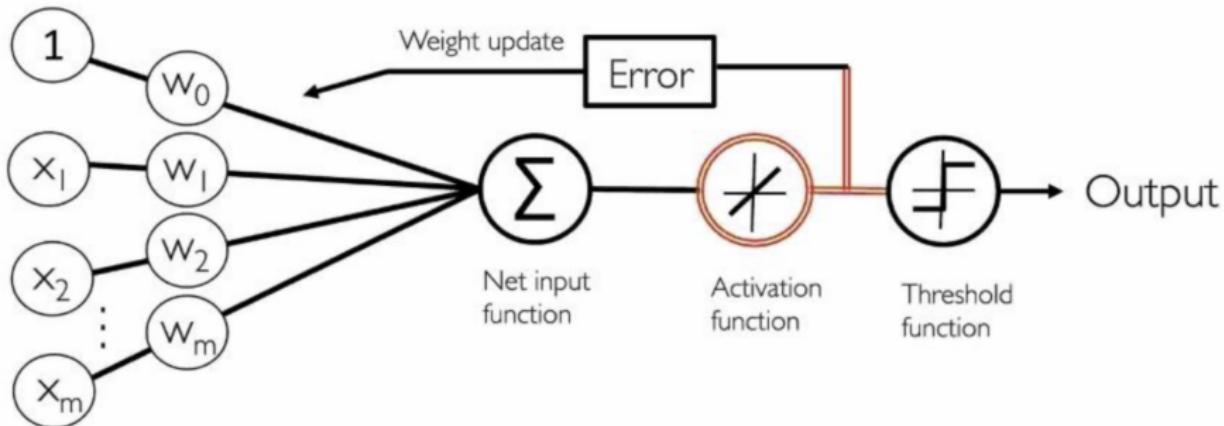
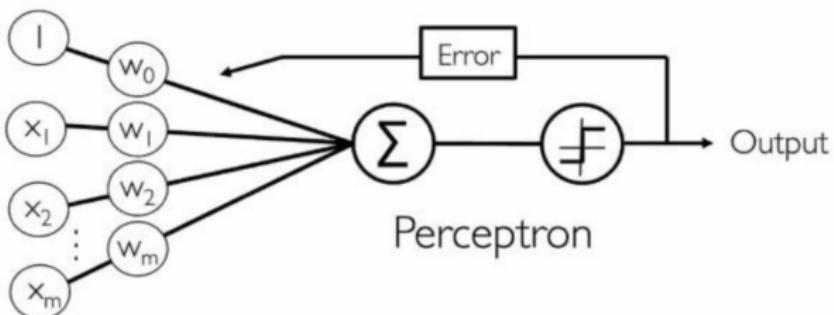
La règle perceptron

Modèle de classification covariables $x_{1:n}$, classes $y_{1:n}$.

- ➊ initialisation w aléatoire ;
- ➋ pour $k = 1, \dots, k$,
 - ➌ pour chaque $i = 1, \dots, n$,
 - ➍ $\hat{y}_i = \phi(x_i^T w)$
 - ➎ $\delta = \eta(y_i - \hat{y}_i)$.
 - ➏ $w = w + \delta x_i$



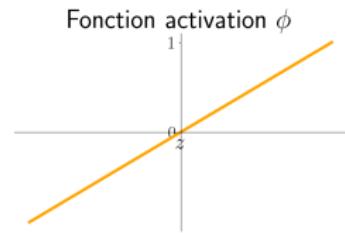
Adaline



Estimation w et Adaline

Modèle de régression covariables $\mathbf{x}_{1:n}$, classes $y_{1:n}$.

- ① initialisation w aléatoire ;
- ② pour $k = 1, \dots, k$,
 - ① pour chaque $i = 1, \dots, n$,
 - ① $\hat{y}_i = \phi(\mathbf{x}_i^\top w)$
 - ② $\delta = \eta(y_i - \hat{y}_i)$.
 - ③ $w = w + \delta \mathbf{x}_i$

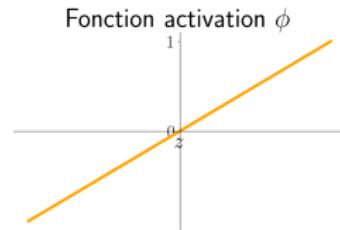


Estimation w et Adaline

Modèle de régression covariables $\mathbf{x}_{1:n}$, classes $y_{1:n}$.

- ➊ initialisation w aléatoire ;
- ➋ pour $k = 1, \dots, k$,
 - ➌ pour chaque $i = 1, \dots, n$,
 - ➍ $\hat{y}_i = \phi(\mathbf{x}_i^\top w)$
 - ➎ $\delta = \eta(y_i - \hat{y}_i)$.
 - ➏ $w = w + \delta \mathbf{x}_i$

Pourquoi $y_i - \hat{y}_i$



$$\hat{w}(\mathbf{x}_{1:n}, y_{1:n}) \in \underset{w \in \mathbb{R}^p}{\operatorname{argmin}} J(w) = \frac{1}{2} \sum_{i=1}^n (y_i - \mathbf{x}_i^\top w)^2 \quad (1)$$

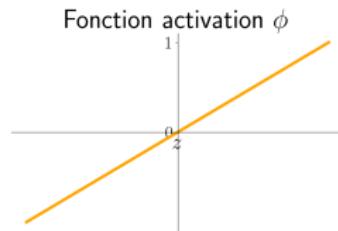
$$\text{descente de gradient } \nabla J(w) = - \sum_{i=1}^n \mathbf{x}_i (y_i - \mathbf{x}_i^\top w) \quad (2)$$

Estimation w et Adaline

Modèle de régression covariables $\mathbf{x}_{1:n}$, classes $y_{1:n}$.

- ➊ initialisation w aléatoire ;
- ➋ pour $k = 1, \dots, k$,
 - ➌ pour chaque $i = 1, \dots, n$,
 - ➍ $\hat{y}_i = \phi(\mathbf{x}_i^\top w)$
 - ➎ $\delta = \eta(y_i - \hat{y}_i)$.
 - ➏ $w = w + \delta \mathbf{x}_i$

Pourquoi $y_i - \mathbf{x}_i^\top w$



$$\hat{w}(\mathbf{x}_{1:n}, y_{1:n}) \in \underset{w \in \mathbb{R}^p}{\operatorname{argmin}} J(w) = \frac{1}{2} \sum_{i=1}^n (y_i - \mathbf{x}_i^\top w)^2 \quad (1)$$

$$\text{descente de gradient } \nabla J(w) = - \sum_{i=1}^n \mathbf{x}_i (y_i - \mathbf{x}_i^\top w) \quad (2)$$

Simple régression linéaire.

1 Notations

2 Le perceptron

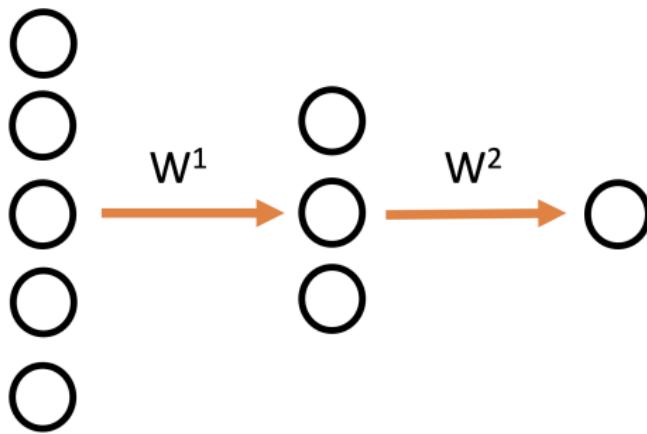
- Modèle à un neurone
- Adaline

3 Modèles multi-couches

- De l'entrée à la sortie
- La rétropropagation

4 Réseaux convolutionnels

Modèle multi-couches



$$a^1 = x$$

$$a^2 = W^1 z^1 + b^1$$

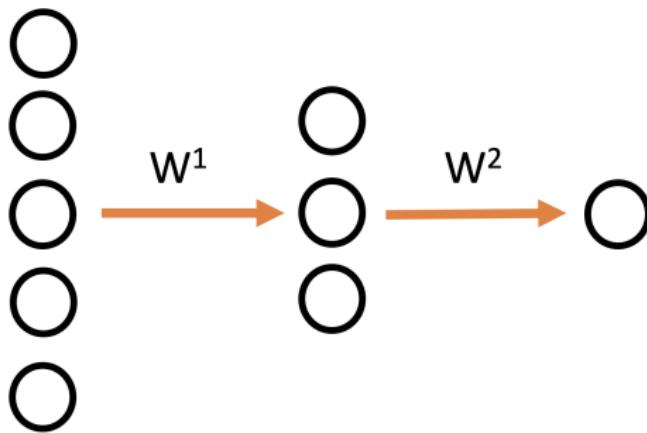
$$z^3 = W^2 z^2 + b^2$$

$$a^2 = \phi(z^2)$$

$$a^3 = \phi(z^3)$$

Exercice : que se passe-t-il si la fonction d'activation ϕ est l'identité ?

Modèle multi-couches



$$a^1 = x$$

$$a^2 = W^1 z^1 + b^1$$

$$z^3 = W^2 z^2 + b^2$$

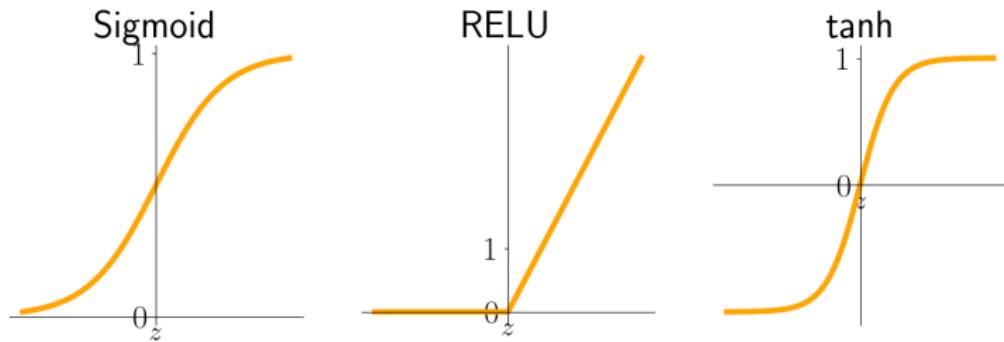
$$a^2 = \phi(z^2)$$

$$a^3 = \phi(z^3)$$

Exercice : que se passe-t-il si la fonction d'activation ϕ est l'identité ?
Simple régression linéaire !.

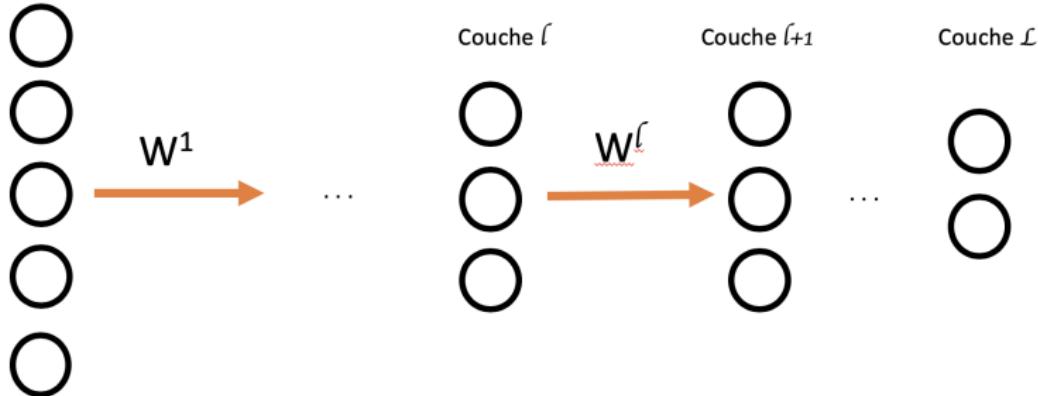
Modèles non linéaires

Autres fonctions d'activation



Permet de modéliser des relations **non linéaires**

modèle



- initialisation $a^1 = x$
- propagation

$$\begin{cases} z_j^{l+1} = \sum_k w_{jk}^l a_k^l + b_j^l & \text{neurone } j \text{ couche } l+1 \\ z^{l+1} = \mathbf{w}^l a^l + b^l \\ a^{l+1} = \phi(z^{l+1}) \end{cases} \quad (3)$$

- prediction $a^L = a_W^L(x)$.

La rétropropagation

La fonction de coût

① Exemple : $J(\mathbf{W}) = \frac{1}{2n} \sum_{i=1}^n \|y_i - a_{\mathbf{W}}^L(x_i)\|^2$

② Hypothèses :

- ① Somme sur les échantillons d'entraînement.
- ② Fonction de a^L .

La rétropropagation

La fonction de coût

① Exemple : $J(\mathbf{W}) = \frac{1}{2n} \sum_{i=1}^n \|y_i - a_{\mathbf{W}}^L(x_i)\|^2$

② Hypothèses :

- ① Somme sur les échantillons d'entraînement.
- ② Fonction de a^L .

Objectif : $\hat{\mathbf{W}} \in \underset{\mathbf{W}}{\operatorname{argmin}} J(\mathbf{W})$

La rétropropagation

La fonction de coût

① Exemple : $J(\mathbf{W}) = \frac{1}{2n} \sum_{i=1}^n \|y_i - a_{\mathbf{W}}^L(x_i)\|^2$

② Hypothèses :

- ① Somme sur les échantillons d'entraînement.
- ② Fonction de a^L .

Objectif : $\hat{\mathbf{W}} \in \underset{\mathbf{W}}{\operatorname{argmin}} J(\mathbf{W})$ Comment ? :

- Supposons $z_j^I \leftarrow z_j^I + \Delta z_j^I$. Alors

$$J \leftarrow J + \frac{\partial J}{\partial z_j^I} \Delta z_j^I \quad (4)$$

La rétropropagation

La fonction de coût

① Exemple : $J(\mathbf{W}) = \frac{1}{2n} \sum_{i=1}^n \|y_i - a_{\mathbf{W}}^L(x_i)\|^2$

② Hypothèses :

- ① Somme sur les échantillons d'entraînement.
- ② Fonction de a^L .

Objectif : $\hat{\mathbf{W}} \in \underset{\mathbf{W}}{\operatorname{argmin}} J(\mathbf{W})$ Comment ? :

- Supposons $z_j^l \leftarrow z_j^l + \Delta z_j^l$. Alors

$$J \leftarrow J + \frac{\partial J}{\partial z_j^l} \Delta z_j^l \tag{4}$$

- Choisir $\Delta z_j^l = -\frac{\partial J}{\partial z_j^l}$. On souhaite donc connaître $\frac{\partial J}{\partial w_{jk}^l}, \frac{\partial J}{\partial b_j^l}$ pour tout j, k, l .

Les équations

Objectif rétropropagation : calculer $\frac{\partial J}{\partial w_{jk}^l}, \frac{\partial J}{\partial b_j^l}$ pour tout j, k, l . On définit

Les équations

Objectif rétropropagation : calculer $\frac{\partial J}{\partial w_{jk}^l}$, $\frac{\partial J}{\partial b_j^l}$ pour tout j, k, l . On définit

$$\delta_j^l = \frac{\partial J}{\partial z_j^l} \tag{5}$$

❶ (BP1) $\delta^L = \nabla_{a^L} J \otimes \phi'(z^L)$

Les équations

Objectif rétropropagation : calculer $\frac{\partial J}{\partial w_{jk}^l}$, $\frac{\partial J}{\partial b_j^l}$ pour tout j, k, l . On définit

$$\delta_j^l = \frac{\partial J}{\partial z_j^l} \tag{5}$$

- ① (BP1) $\delta^L = \nabla_{a^L} J \otimes \phi'(z^L)$
- ② (BP2) Pour $l = L - 1, \dots, 1$, $\delta^l = ((W^{l+1})^\top \delta^{l+1}) \otimes \phi'(z^l)$

Les équations

Objectif rétropropagation : calculer $\frac{\partial J}{\partial w_{jk}^l}$, $\frac{\partial J}{\partial b_j^l}$ pour tout j, k, l . On définit

$$\delta_j^l = \frac{\partial J}{\partial z_j^l} \tag{5}$$

- ① (BP1) $\delta^L = \nabla_{a^L} J \otimes \phi'(z^L)$
- ② (BP2) Pour $l = L - 1, \dots, 1$, $\delta^l = ((W^{l+1})^\top \delta^{l+1}) \otimes \phi'(z^l)$
- ③ (BP3) $\frac{\partial J}{\partial b_j^l} = \delta_j^l$

Les équations

Objectif rétropropagation : calculer $\frac{\partial J}{\partial w_{jk}^l}$, $\frac{\partial J}{\partial b_j^l}$ pour tout j, k, l . On définit

$$\delta_j^l = \frac{\partial J}{\partial z_j^l} \tag{5}$$

- ① (BP1) $\delta^L = \nabla_{a^L} J \otimes \phi'(z^L)$
- ② (BP2) Pour $l = L-1, \dots, 1$, $\delta^l = ((W^{l+1})^\top \delta^{l+1}) \otimes \phi'(z^l)$
- ③ (BP3) $\frac{\partial J}{\partial b_j^l} = \delta_j^l$
- ④ (BP4) $\frac{\partial J}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$

1 Notations

2 Le perceptron

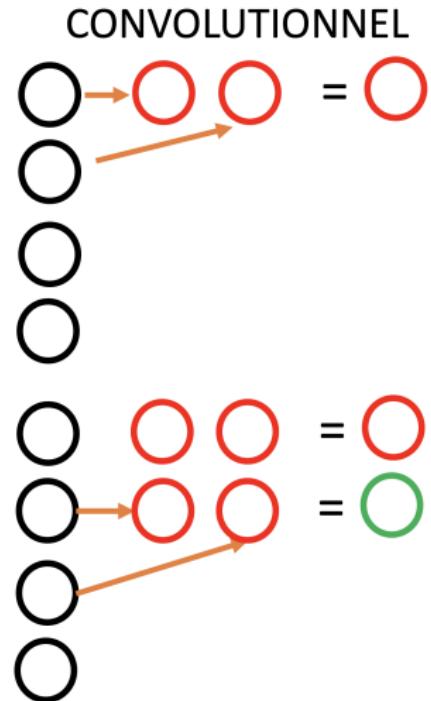
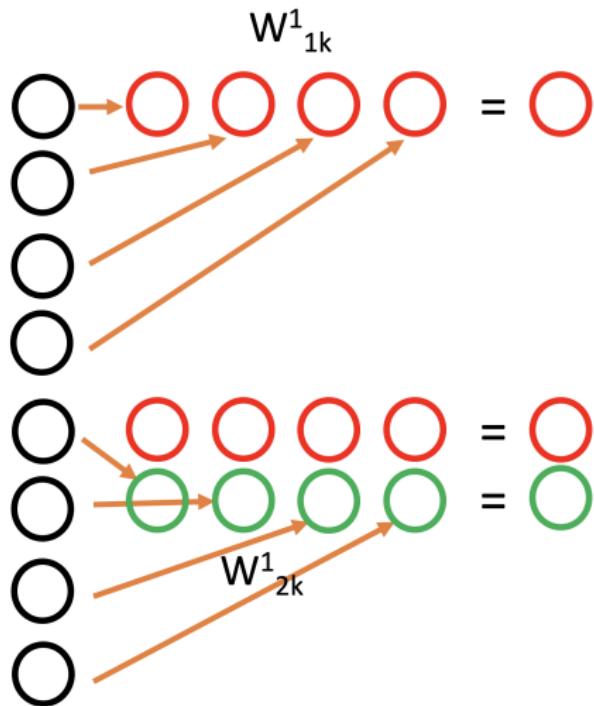
- Modèle à un neurone
- Adaline

3 Modèles multi-couches

- De l'entrée à la sortie
- La rétropropagation

4 Réseaux convolutionnels

Réseaux convolutionnels 1D



Réseaux convolutionnels 2D

0	0	0	0	0	0	0	...
0	156	155	156	158	158	...	
0	153	154	157	159	159	...	
0	149	151	155	158	159	...	
0	146	146	149	153	158	...	
0	145	143	143	148	158	...	
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



308

biais
+ =

0	0	0	0	0	0	0	...
0	156	155	156	158	158	...	
0	153	154	157	159	159	...	
0	149	151	155	158	159	...	
0	146	146	149	153	158	...	
0	145	143	143	148	158	...	
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



310

biais
+ =

Réseaux convolutionnels 2D - multi-canaux

0	0	0	0	0	0	0	...
0	156	155	156	158	158	...	
0	153	154	157	159	159	...	
0	149	151	155	158	159	...	
0	146	146	149	153	158	...	
0	145	143	143	148	158	...	
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



161

0	0	0	0	0	0	0	...
0	167	166	167	169	169	...	
0	164	165	168	170	170	...	
0	160	162	166	169	170	...	
0	156	156	159	163	168	...	
0	155	153	153	158	168	...	
...

Input Channel #2 (Green)

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-9

0	0	0	0	0	0	0	...
0	163	162	163	165	165	...	
0	160	161	164	166	166	...	
0	156	158	162	165	166	...	
0	155	155	158	162	167	...	
0	154	152	152	157	167	...	
...

Input Channel #3 (Blue)

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



659

+

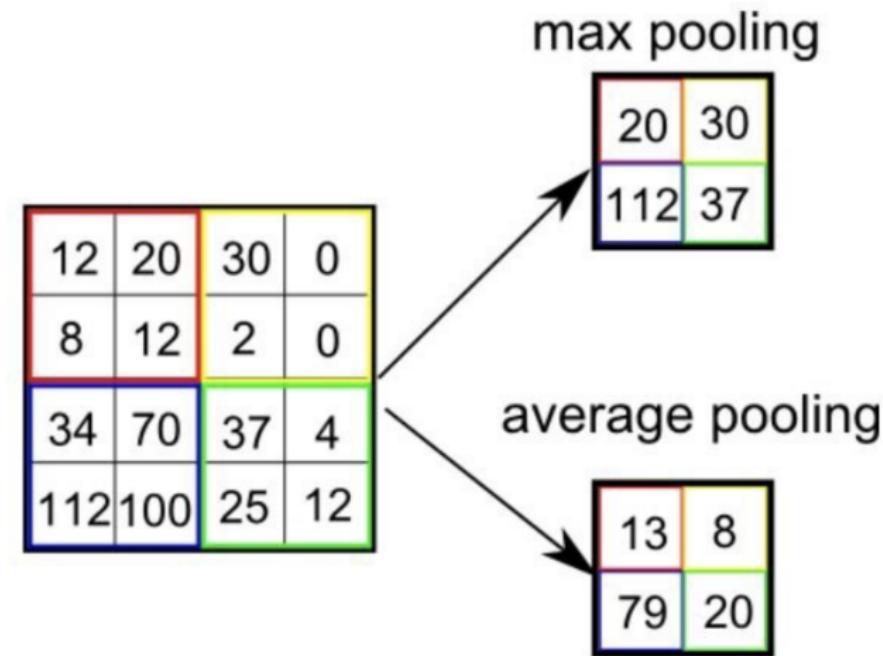
+ 1 = 812

Bias = 1

-25	466	466	475	...
295	787	798	812	...
...
...
...

Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

Réseaux convolutionnels 2D - pooling



Types of Pooling

Réseaux convolutionnels 2D - réseau complet

