Paradigme connexionniste

Perceptron multi-couches

rétro propagation de gradient d’erreur

Université Bretagne Sud

G.Ménier : gildas.menier@univ-ubs.fr

Paradigme fonctionnel

- **Transformation de données en entrée**

$$\boldsymbol{Output = f(Input)}$$

- **Composition de fonctions explicite**
  - **Composition fonctionnelle**

- **Fonction adaptée / adaptable / sur mesure**
  - **Approximeur *universel***
  - **Paradigme connexionniste**

Principe

- **Transformation de données en entrée**

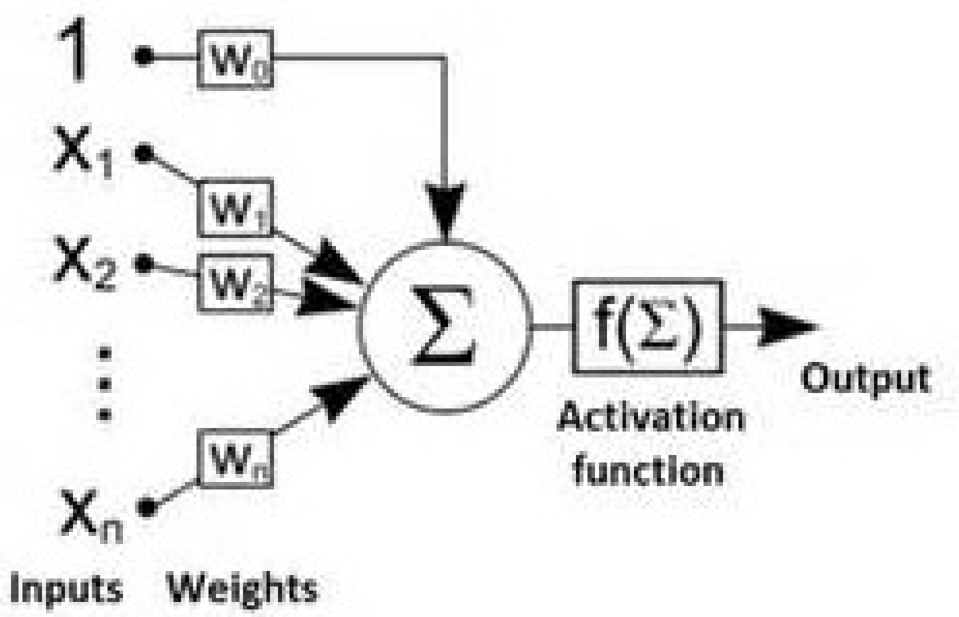
$$\boldsymbol{Output = f(Input)}$$

- **Plutôt que déterminer une série de fonctions à composer**
- **Création d'une fonction adaptée**
- **Apprentissage des entrées -> sorties**
- **HashMap ?**
- **Généralisation**

gildas.menier@univ-ubs.fr

Perceptron

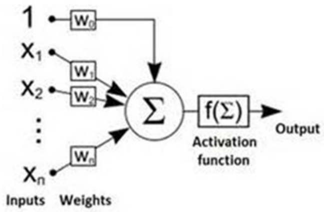
- McCulloch-Pitts **(1943 - 1957)**



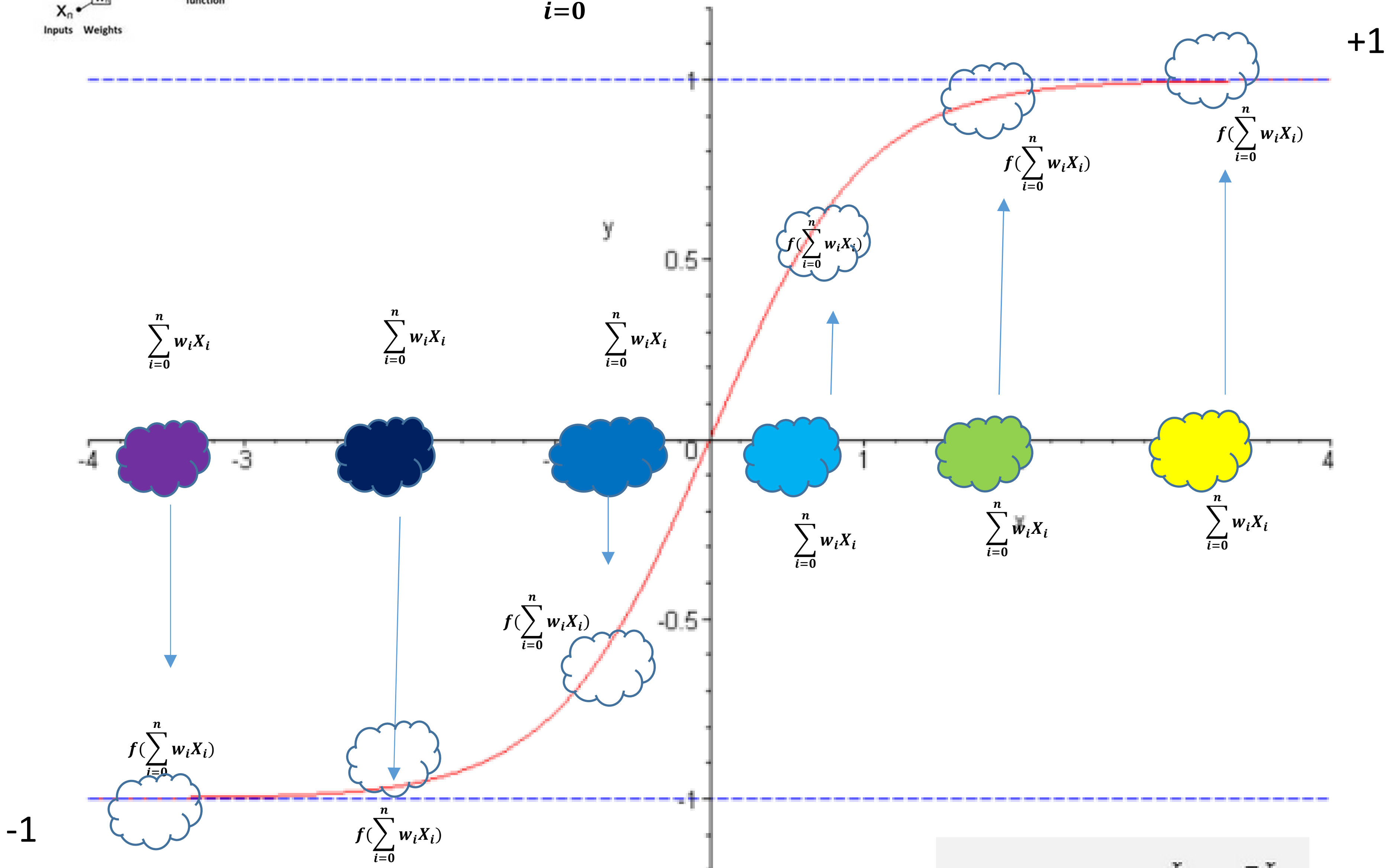
$$Output = f(\sum_{i=0}^{n} w_i X_i)$$

Appelé *abusivement* neurone

gildas.menier@univ-ubs.fr



$$Output = f(\sum_{i=0}^{n} w_i X_i)$$

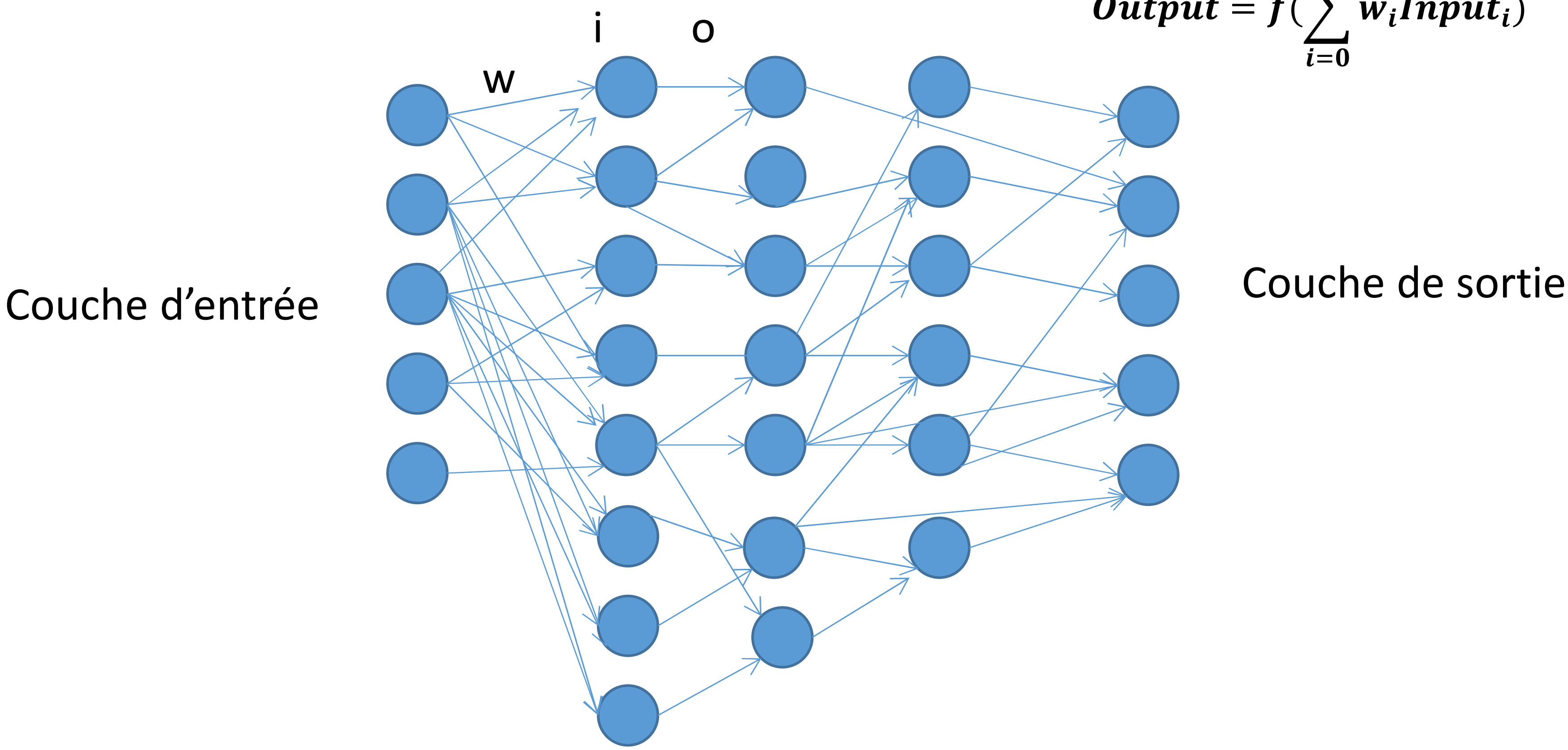


$$a = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Perceptron : Franck Rosenblatt

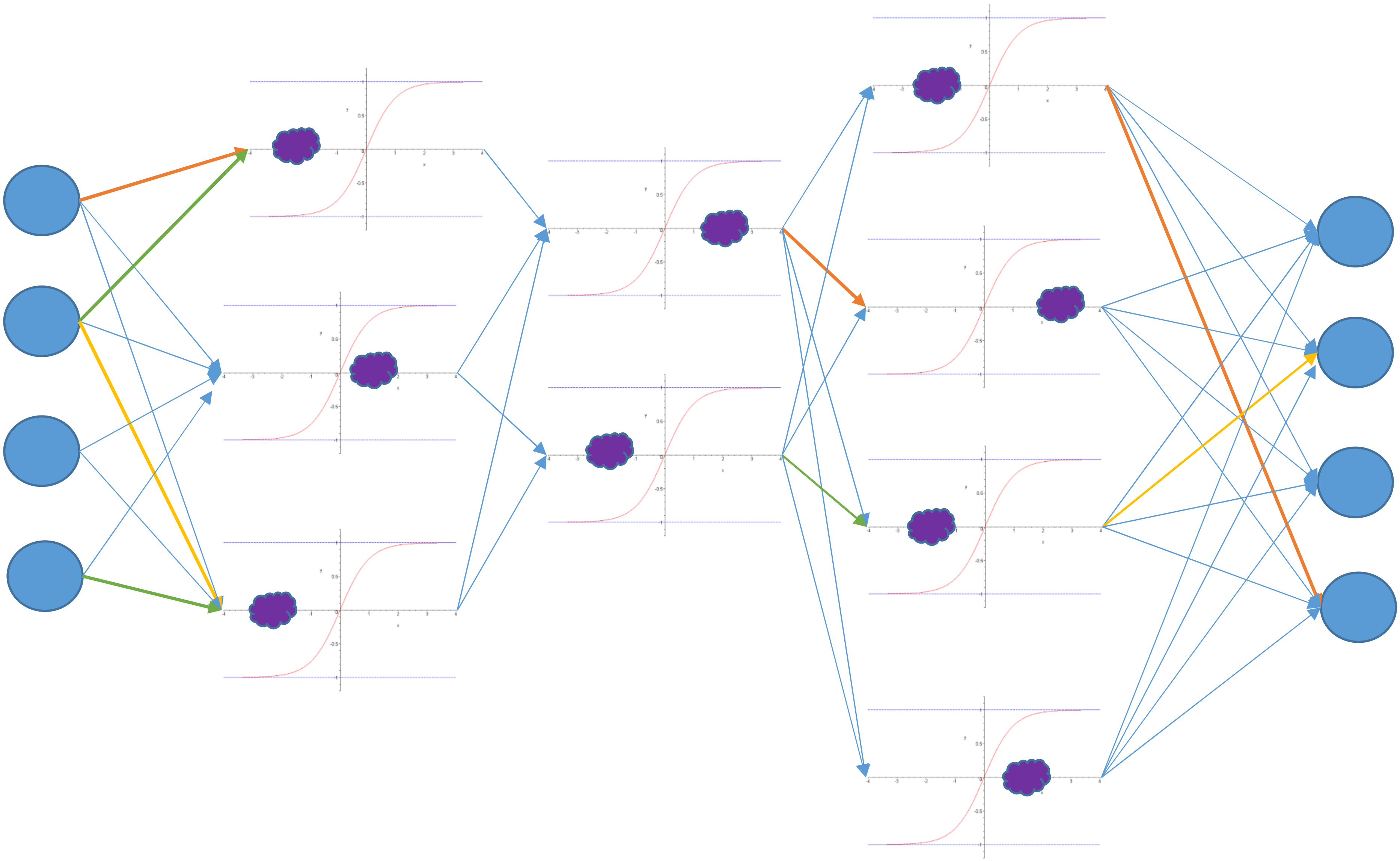
- **Perceptron multi-couches (& algo 84-86)**

$$Output = f\left(\sum_{i=0}^{n} w_i Input_i\right)$$



Appelé *abusivement* réseau de neurones

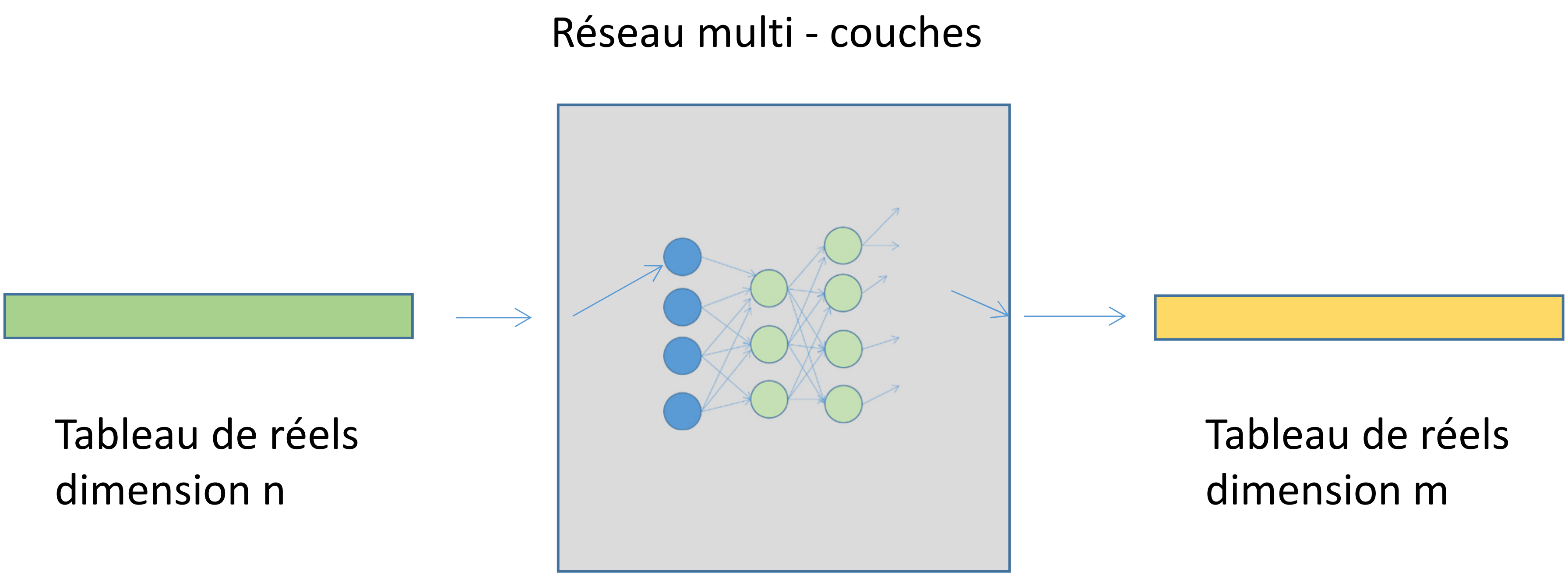
gildas.menier@univ-ubs.fr



gildas.menier@univ-ubs.fr

Perceptron

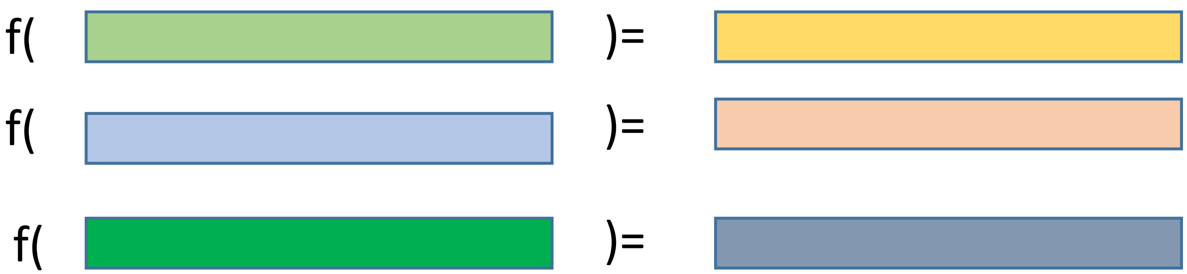
- **Perceptron multi-couches**



gildas.menier@univ-ubs.fr

Perceptron

- **Perceptron multi-couches : apprentissage**

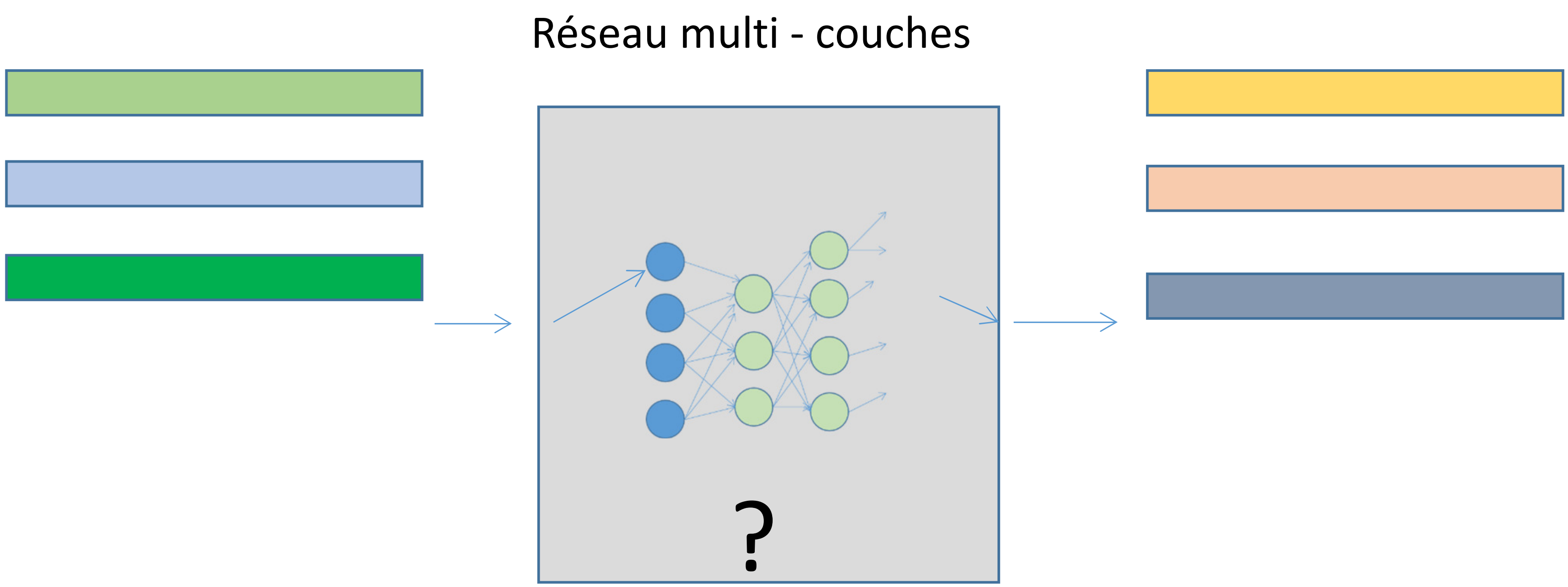


- **Trouver f tel que…**

gildas.menier@univ-ubs.fr

Perceptron

- **Perceptron multi-couches : apprentissage**

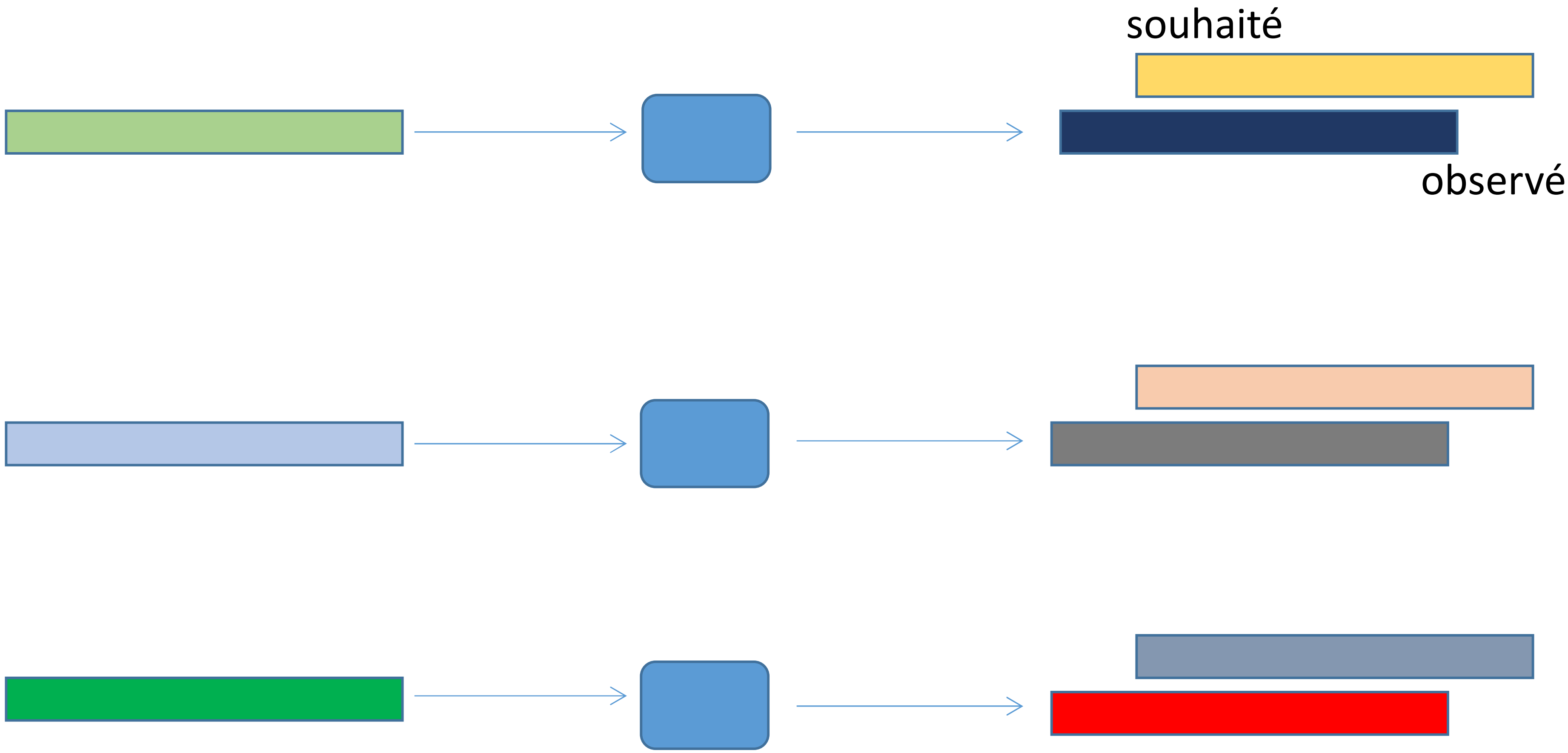


Trouver les poids wij tel que…

gildas.menier@univ-ubs.fr

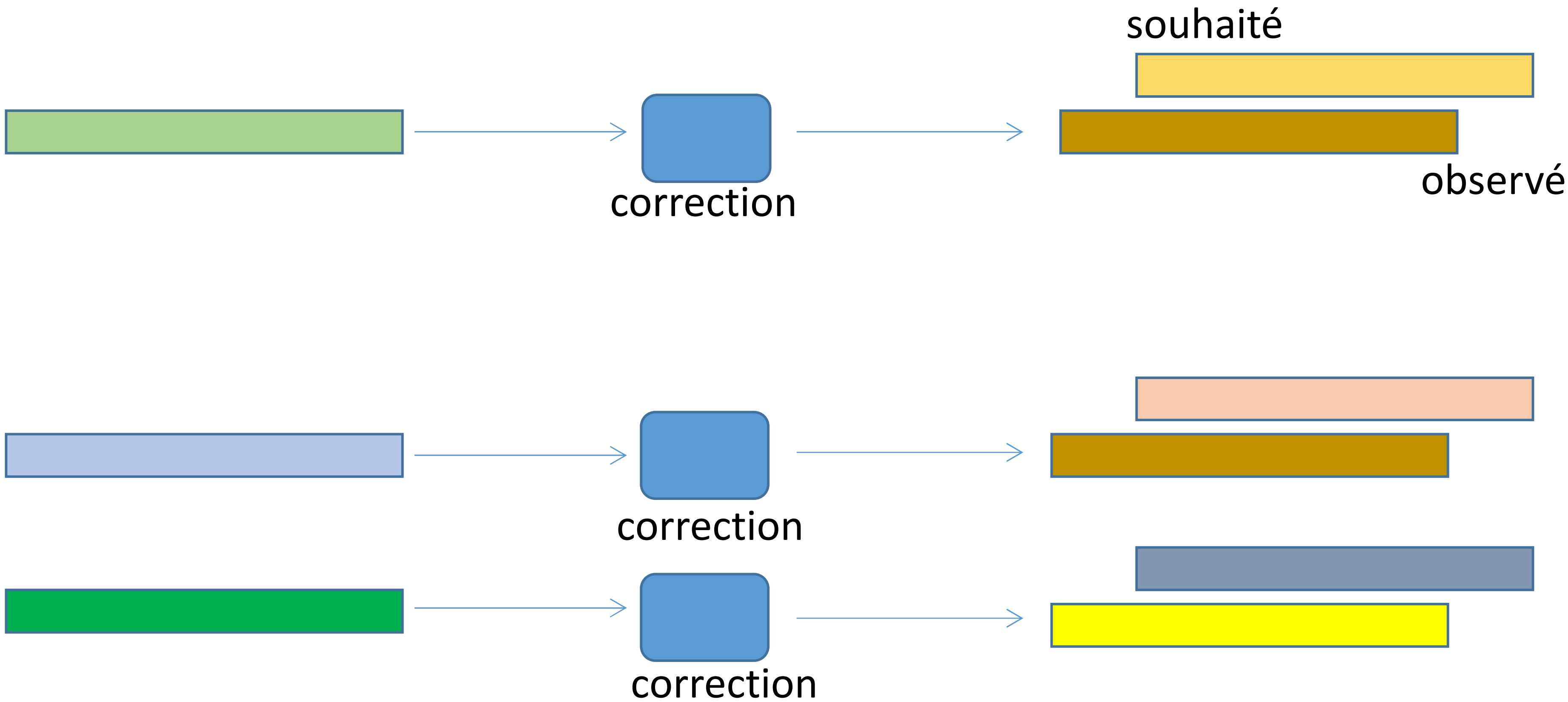
Perceptron

- **Perceptron multi-couches : apprentissage**



Perceptron

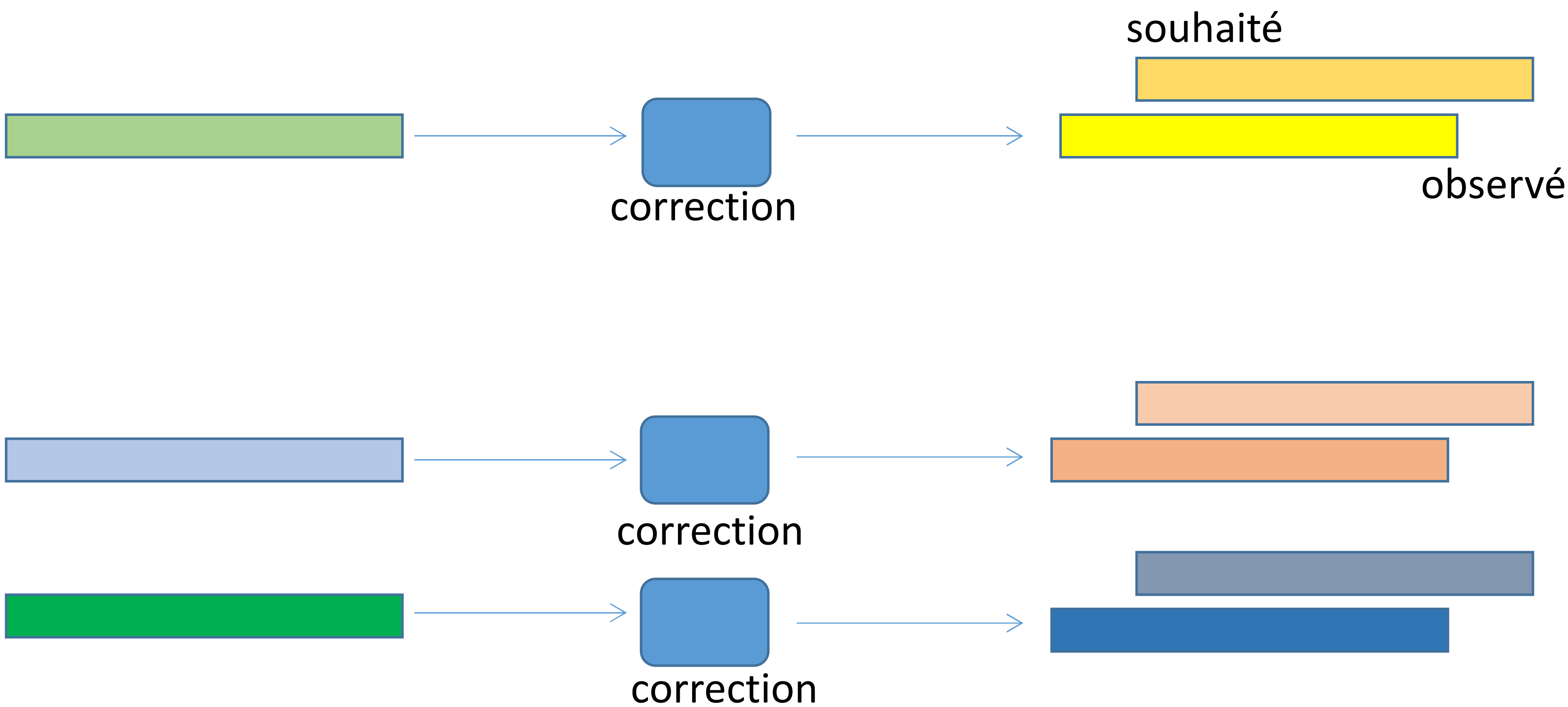
- **Perceptron multi-couches : apprentissage**



gildas.menier@univ-ubs.fr

Perceptron

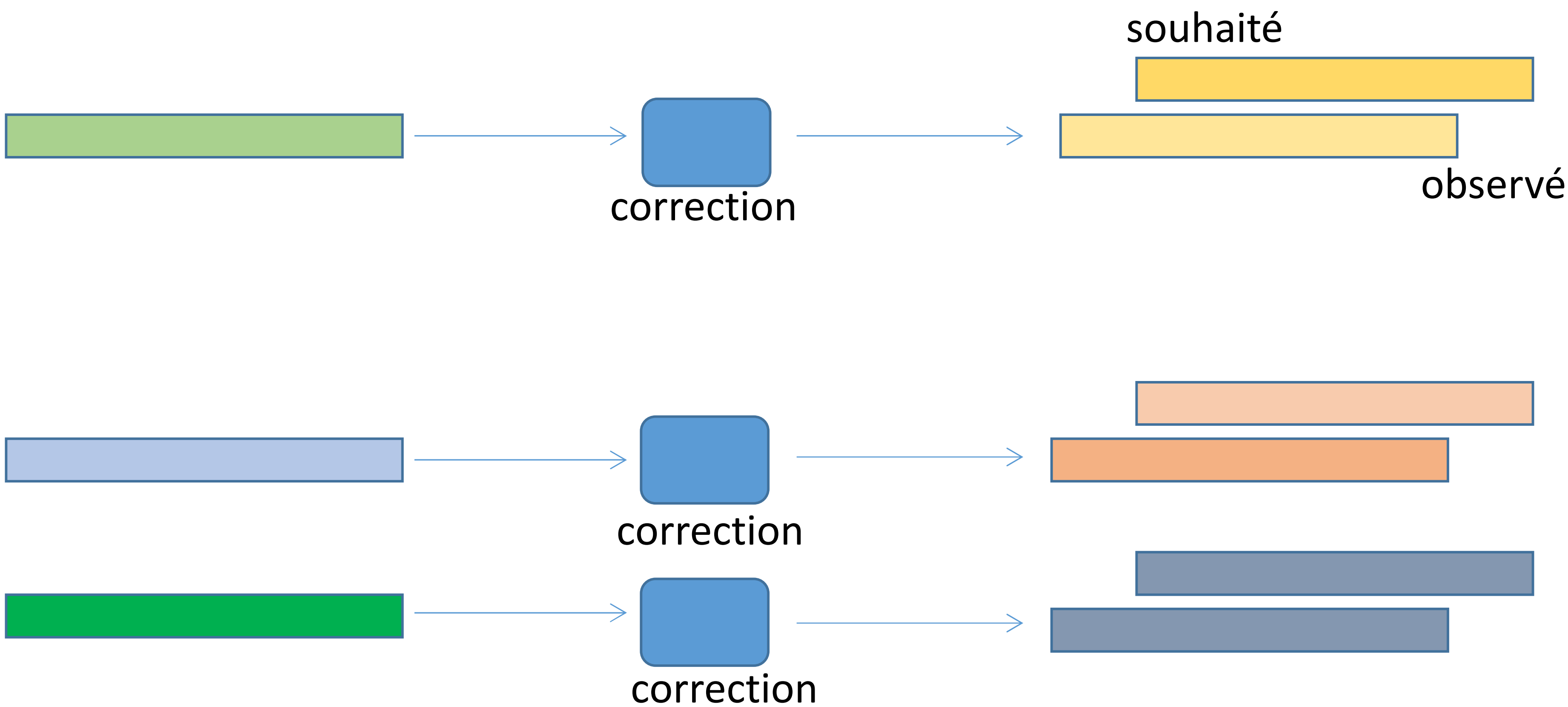
- **Perceptron multi-couches : apprentissage**



gildas.menier@univ-ubs.fr

Perceptron

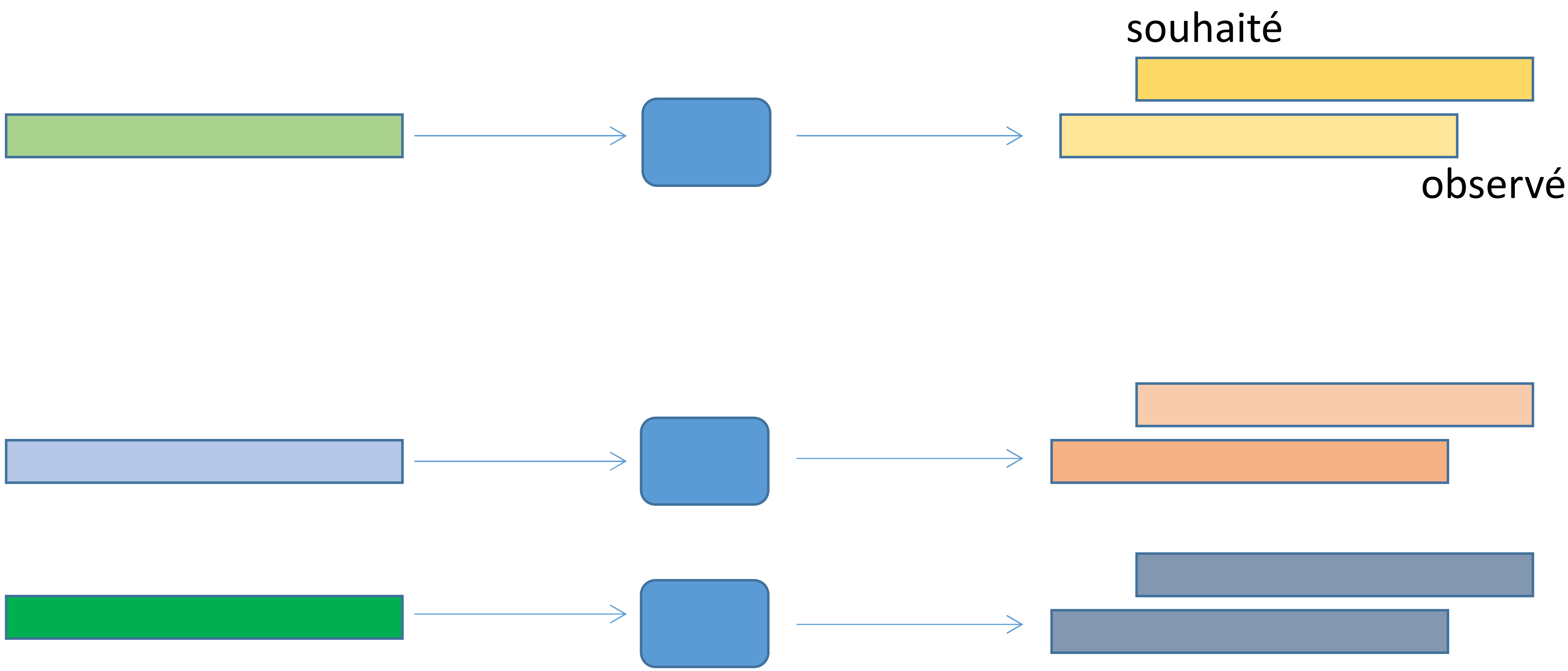
- **Perceptron multi-couches : apprentissage**



gildas.menier@univ-ubs.fr

Perceptron

- **Perceptron multi-couches : apprentissage**



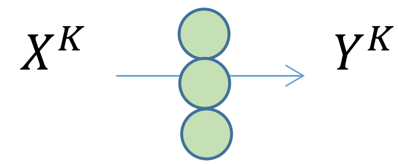
gildas.menier@univ-ubs.fr

Perceptron

- **Perceptron multi-couches : apprentissage**
- **Rétro propagation du gradient d'erreur**
- **Une activation provoque des erreurs en sortie**
- **On essaye de partir de la sortie et de remonter les erreurs et donc les corrections**
- **Dépend des paramètres**
- **De la fonction de transfert f**

gildas.menier@univ-ubs.fr

On souhaite que pour tout K



Perceptron

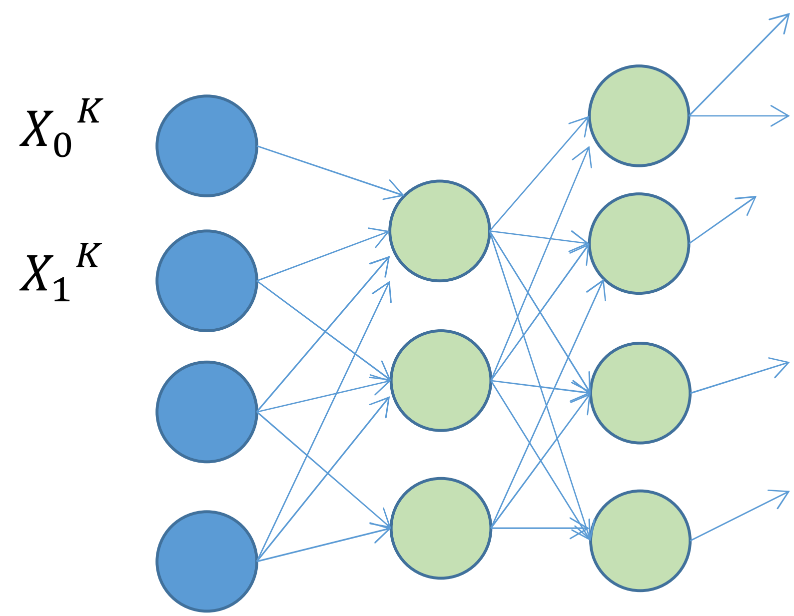
X : ensemble des exemples à apprendre

$$X = X^1, X^2, X^3, \ldots$$

$$X^K = \{X_0{}^K, X_1{}^K, X_2{}^K, \ldots\}$$

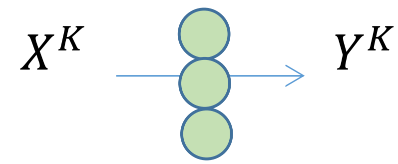
Un vecteur d’état d’activation de la couche d’entrée

X^K



gildas.menier@univ-ubs.fr

On souhaite que pour tout K



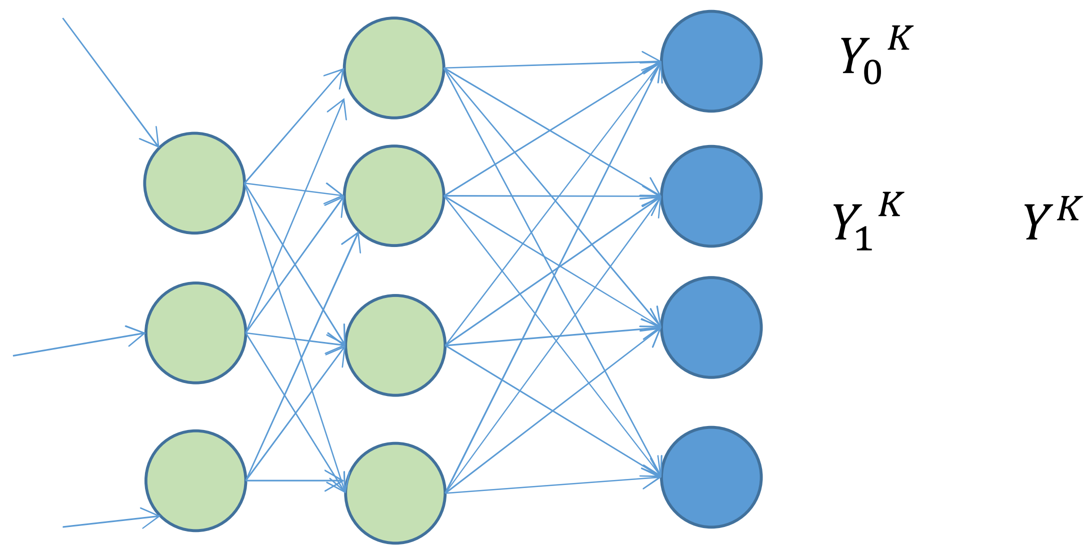
Perceptron

Y : ensemble des réponses à apprendre

Y= $Y^1, Y^2, Y^3, \ldots$

$Y^K = \{Y_0{}^K, Y_1{}^K, Y_2{}^K, \ldots\}$

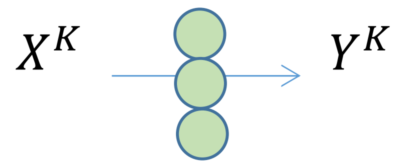
Un vecteur d'état souhaité de la couche de sortie



gildas.menier@univ-ubs.fr

Perceptron

On souhaite que pour tout K



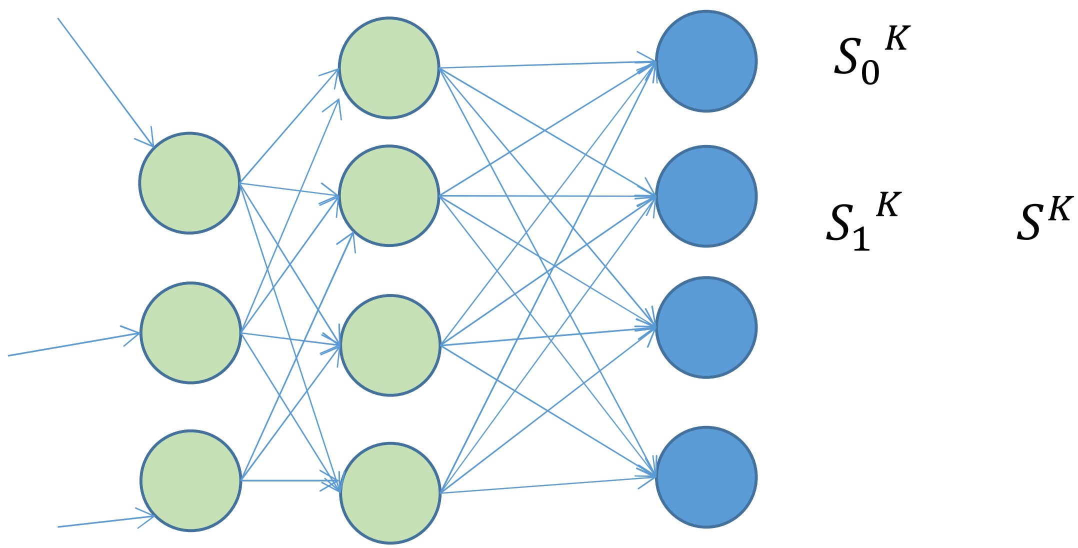
mais on obtient S^K

S : ensemble des réponses constatées

S= $S^1, S^2, S^3, \ldots$

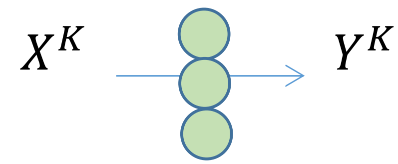
$$S^K = \{S_0{}^K, S_1{}^K, S_2{}^K, \ldots\}$$

Un vecteur d'état souhaité de la couche de sortie



gildas.menier@univ-ubs.fr

On souhaite que pour tout K



mais on obtient S^K

Perceptron

L'erreur mesurée est la différence entre Y et S

L'erreur dépend de l'ensemble des poids **w**

Pour un exemple K

Erreur quadratique :

$$E(w)^K = (S^K - Y^K)^2 = \sum_i (S_i^K - Y_i^K)^2$$

Erreur totale :

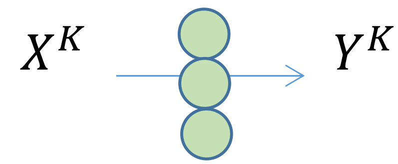
$$E(w) = \sum_K E(w)^K$$

La somme des erreurs quadratique sur tous les exemples

gildas.menier@univ-ubs.fr

Perceptron

On souhaite que pour tout K



mais on obtient S^K

Quand on fait varier un peu les poids w -> w',
on fait varier un peu l'erreur commise E -> E' :

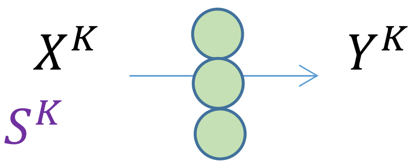
(E'-E)/(w'-w) est la variation :

$$\frac{\partial E(w)}{\partial w}$$

gildas.menier@univ-ubs.fr

Perceptron

On souhaite que pour tout K



mais on obtient S^K

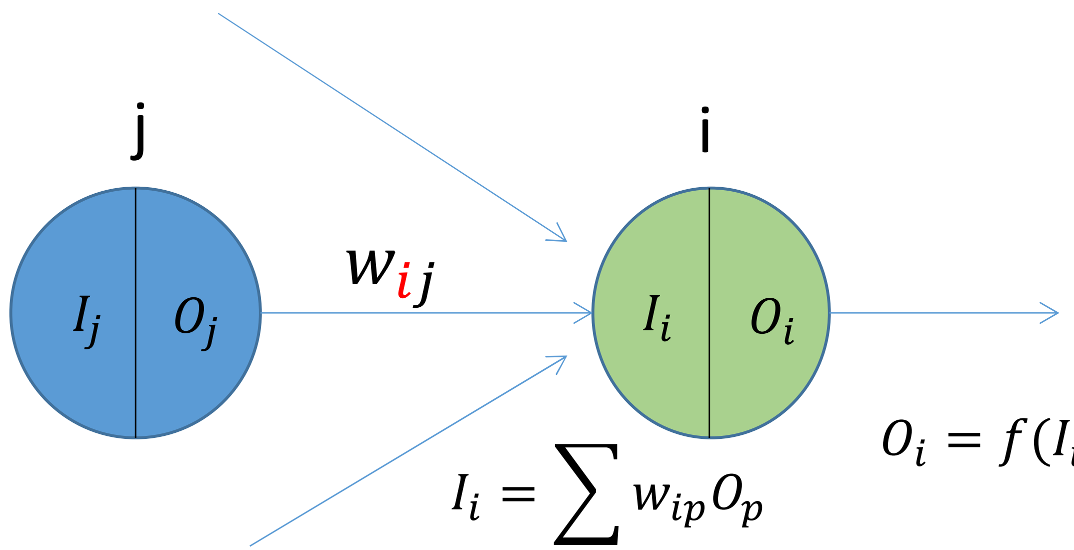
On veut corriger les poids w pour minimiser E

On veut modifier les poids w pour que S se rapproche de Y

Il faut rechercher comment faire varier w pour faire varier E

(dans le bon sens).

Gradient de E pour w : dérivée de E par rapport à w : $\frac{\partial E(w)}{\partial w}$



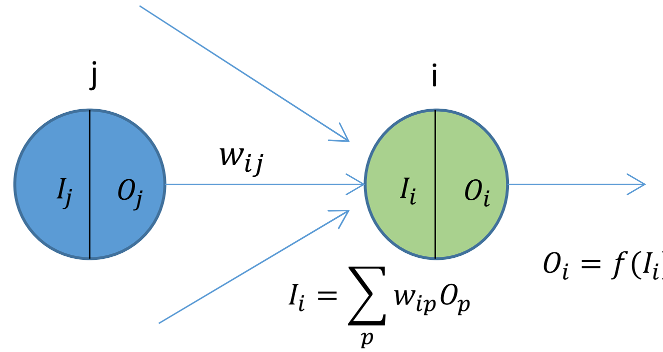
$$I_i = \sum_p w_{ip} O_p$$

$$O_i = f(I_i)$$

w_{ip} : qui arrive à i et vient de p

gildas.menier@univ-ubs.fr

Perceptron



Soit e(K) une valeur définie (positive).

On souhaite corriger les w_{ij} de manière incrémentale K par K :

$$w_{ij}(K) = w_{ij}(K-1) - e(K).\frac{\partial E^K}{\partial w_{ij}}$$

On souhaite enlever des w_{ij} ce qui provoque une erreur E^K

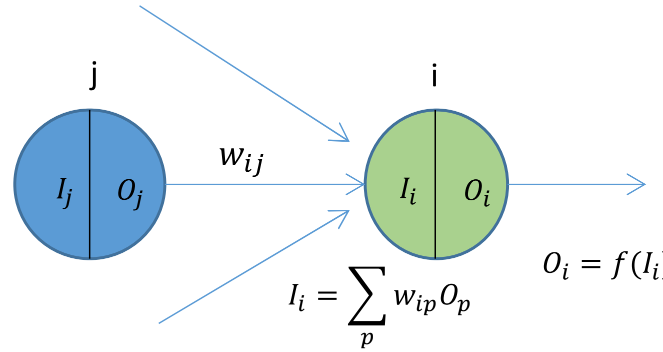
$$\frac{\partial E^K}{\partial w_{ij}} = \frac{\partial E^K}{\partial I_i}.\frac{\partial I_i}{\partial w_{ij}}$$

$$\frac{\partial I_i}{\partial w_{ij}} = \frac{\partial\left(\sum_p w_{ip}.O_p\right)}{\partial w_{ij}} = O_j$$

car ∂I_i ne dépend que de la sortie de j

gildas.menier@univ-ubs.fr

Perceptron



On souhaite corriger les w_{ij} de manière incrémentale K par K :

$$w_{ij}(\mathrm{K}) = w_{ij}(\mathrm{K-1}) - \mathrm{e(K)}.\frac{\partial E^K}{\partial w_{ij}}$$

$$\frac{\partial E^K}{\partial w_{ij}} = \frac{\partial E^K}{\partial I_i}.\frac{\partial I_i}{\partial w_{ij}} \quad \text{et} \quad \frac{\partial I_i}{\partial w_{ij}} = O_j$$

Donc

$$w_{ij}(\mathrm{K}) = w_{ij}(\mathrm{K-1}) - \mathrm{e(K)}.\frac{\partial E^K}{\partial I_i}.\,O_j = w_{ij}(\mathrm{K-1}) - \mathrm{e(K)}.d_i.\,O_j$$

avec

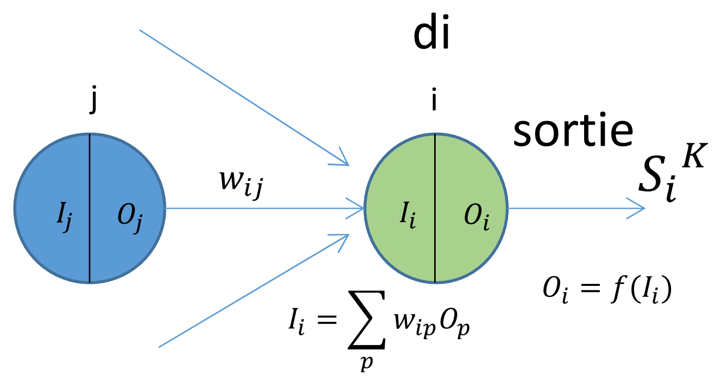
Pour n’importe quelle couche

$$d_i = \frac{\partial E^K}{\partial I_i}$$

Cas particuliers : couche de sortie et couche cachée

gildas.menier@univ-ubs.fr

Perceptron : sortie



On souhaite corriger les w_{ij} de manière incrémentale K par K :

$$w_{ij}(\mathrm{K}) = w_{ij}(\mathrm{K-1}) - \mathrm{e(K)}.\frac{\partial E^K}{\partial I_i}.\, O_j = w_{ij}(\mathrm{K-1}) - \mathrm{e(K)}.d_i.\, O_j$$

Pourquoi ? Où est passée Ys ?

Cas particulier de la couche de sortie

$$d_i = \frac{\partial E^K}{\partial I_i} = \frac{\partial(\sum_s ({S_s}^K - {Y_s}^K)^2)}{\partial I_i} = 2({S_i}^K - {Y_i}^K).\frac{\partial {S_i}^K}{\partial I_i}$$

Or ${S_i}^K = {O_i}^K = f(I_i)$

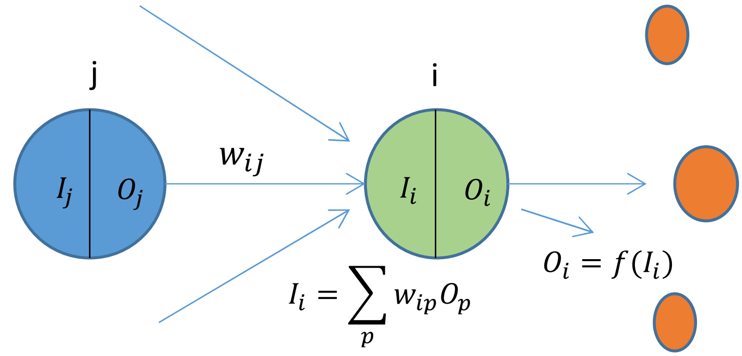
Donc $d_i = 2({S_i}^K - {Y_i}^K).\, f'(I_i)$

$$\boxed{w_{ij}(\mathrm{K}) = w_{ij}(\mathrm{K-1}) - \mathrm{e(K)}.\ 2({S_i}^K - {Y_i}^K).\, f'(I_i).\, O_j}$$

On sait calculer tous les termes (S est connu, Y est connu, Ii est connu, Oj est connu)

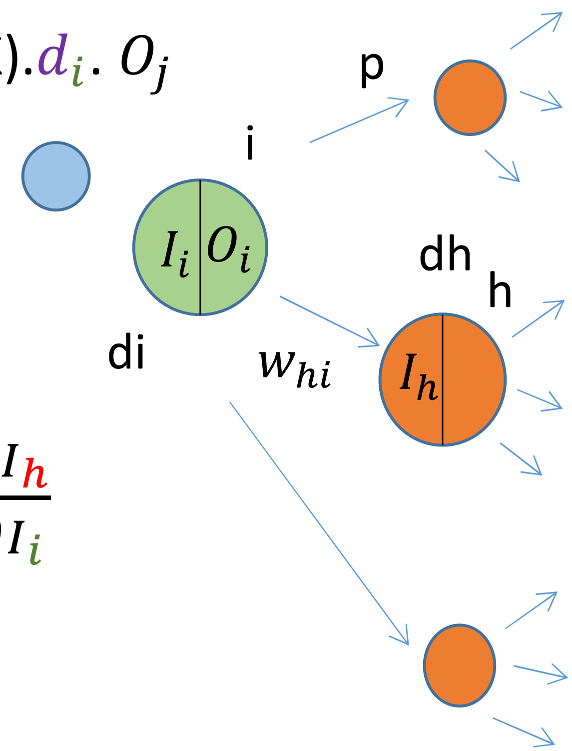
gildas.menier@univ-ubs.fr

Perceptron : cachée



On souhaite corriger les w_{ij} de manière incrémentale K par K :

$$w_{ij}(\mathrm{K}) = w_{ij}(\mathrm{K}-1) - e(\mathrm{K}).\frac{\partial E^K}{\partial I_i}.\, O_j = w_{ij}(\mathrm{K}-1) - e(\mathrm{K}).d_i.\, O_j$$



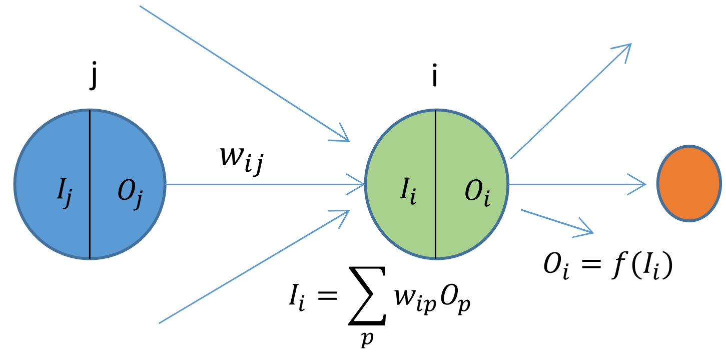
Dans le cas d'une couche cachée (*hidden*):

$$d_i = \frac{\partial E^K}{\partial I_i} = \sum_h \frac{\partial E^K}{\partial I_h} . \frac{\partial I_h}{\partial I_i} = \sum_h d_h . \frac{\partial I_h}{\partial I_i}$$

$$= \sum_h d_h . \frac{\partial I_h}{\partial O_i} . \frac{\partial O_i}{\partial I_i}$$

gildas.menier@univ-ubs.fr

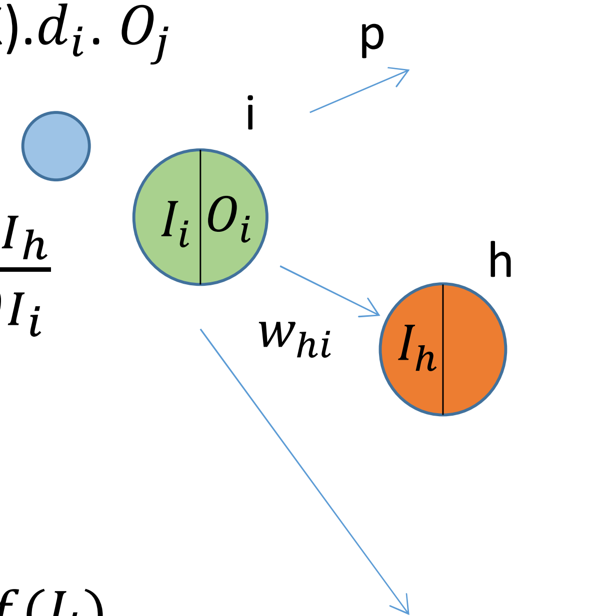
Perceptron : cachée



On souhaite corriger les w_{ij} de manière incrémentale K par K :

$$w_{ij}(K) = w_{ij}(K-1) - e(K).\frac{\partial E^K}{\partial I_i}.\, O_j = w_{ij}(K-1) - e(K).d_i.\, O_j$$

Dans le cas d'une couche précédente (cachée h*idden*):



$$d_i = \frac{\partial E^K}{\partial I_i} = \sum_h \frac{\partial E^K}{\partial I_h} . \frac{\partial I_h}{\partial I_i} = \sum_h d_h . \frac{\partial I_h}{\partial I_i}$$

$$= \sum_h d_h . \frac{\partial I_h}{\partial O_i} . \frac{\partial O_i}{\partial I_i}$$

$$\frac{\partial I_h}{\partial O_i} = \frac{\partial (\sum_p w_{hp}.O_p)}{\partial O_i} = w_{hi}$$

$$\frac{\partial f(I_i)}{\partial I_i} = f'(I_i)$$

$$= \sum_h d_h . w_{hi} . f'(I_i)$$

gildas.menier@univ-ubs.fr

Perceptron : retro propagation de gradient d’erreur

Pour les K tirés de manière aléatoire, e(K) pas d’apprentissage

On présente **un** exemple K et on active le réseau : les I et O sont calculés

activation

On corrige les poids w_{ij}

$$w_{ij}(\text{K}) = w_{ij}(\text{K-1}) - \text{e(K)}.d_i.\, O_j$$

Si le neurone i se trouve sur la couche de sortie, alors

$$d_i = 2({S_i}^K - {Y_i}^K).\, f'(I_i)$$

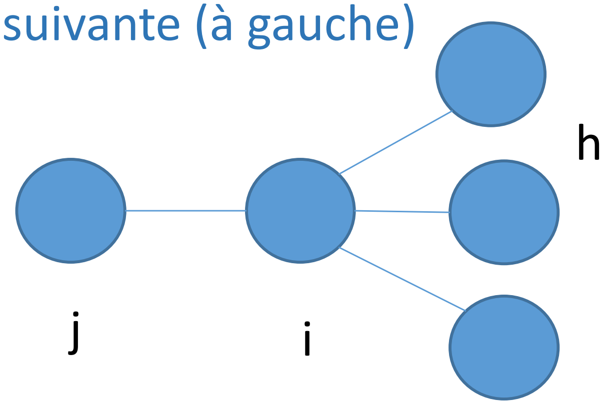
modification w

Sinon

$$d_i = \sum_h d_h . w_{hi} . f'(I_i)$$

h sont les neurones de la couche suivante (à gauche)

Tant que E n’est pas acceptable



gildas.menier@univ-ubs.fr

Pour les K tirés de manière aléatoire, e(K) pas d'apprentissage

On présente **un** exemple K et on active le réseau : les I et O sont calculés

On corrige les poids w_{ij}

$$w_{ij}(K) = w_{ij}(K-1) - e(K).d_i.\,O_j$$

Si le neurone i se trouve sur la couche de sortie, alors

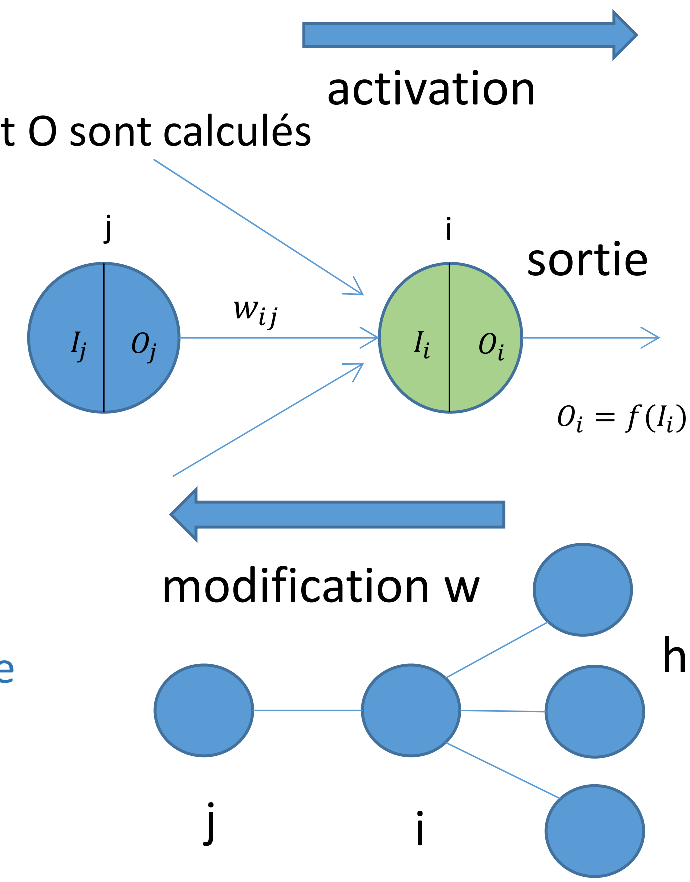
$$d_i = 2(S_i^K - Y_i^K).\,f'(I_i)$$

Sinon

$$d_i = \sum_h d_h.\,w_{hi}.f'(I_i)$$

h sont les neurones de la couche suivante

Tant que E n'est pas acceptable

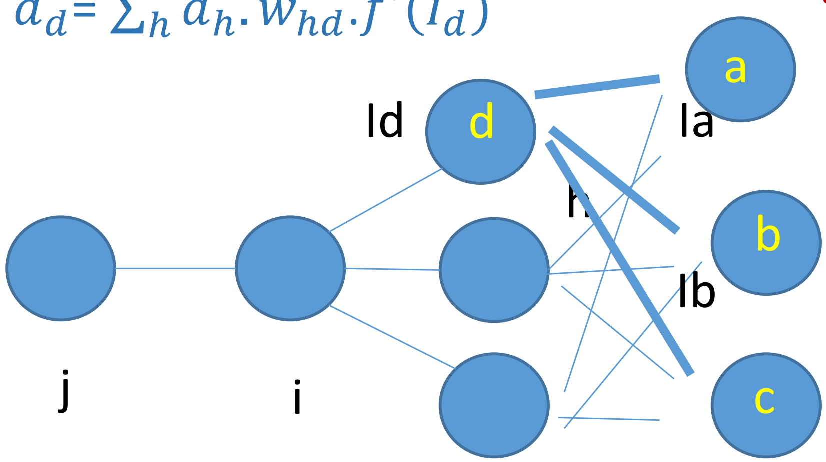


Perceptron : retro propagation de gradient d'erreur

w_{ij}(K) = w_{ij}(K-1) –e(K).d_i. O_j

$d_d = \sum_h d_h . w_{hd} . f'(I_d)$

d_a= 2($S_a{}^K - Y_a{}^K$). $f'(I_a)$ $\quad S_a{}^K$



d_b= 2($S_b{}^K - Y_b{}^K$). $f'(I_b)$ $\quad S_b{}^K$

d_c= 2($S_c{}^K - Y_c{}^K$). $f'(I_c)$ $\quad S_c{}^K$

gildas.menier@univ-ubs.fr

Perceptron multi-couches

Entrées : moyenne 0 et variance 1 $s^2 = \frac{\Sigma(X - \overline{X})^2}{n-1}$

Normalisation : centrées réduites (- moyenne / variance)

Nombre de couches ?

Pas  d’apprentissage e(K) ?

Nombre d’apprentissage ?

Présentation des exemples ?

Fonction de transfert f ?

Biais (neurone activé) ?

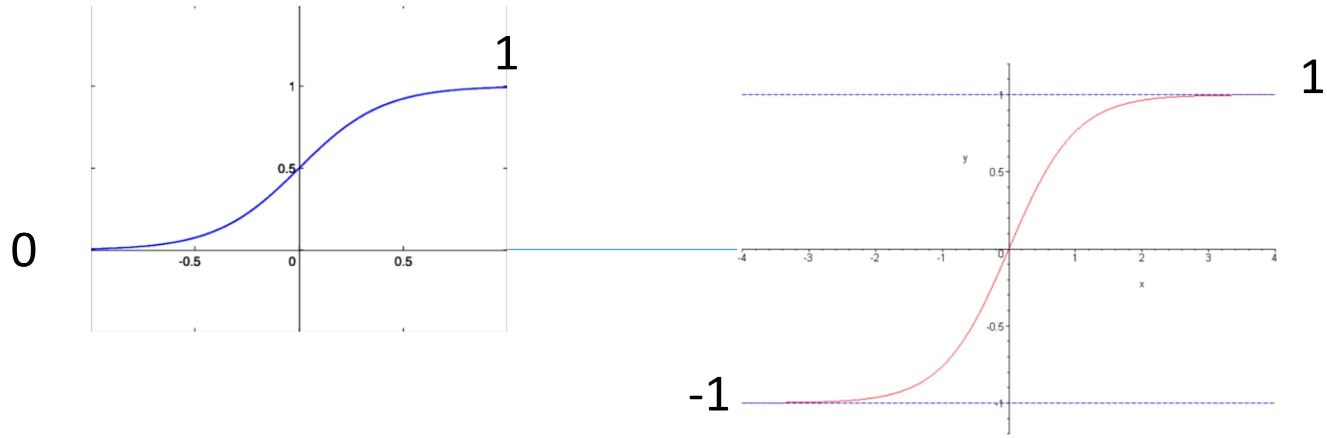
Calculabilité ?

Poids de départ ?

gildas.menier@univ-ubs.fr

Perceptron multi-couches

Fonctions de transfert



fonction sigmoïde

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

$$f'(x) = f(x).(1 - f(x))$$

fonction tangente hyperbolique

$$a = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$f'(x) = (1 + f(x)).(1 - f(x))$$

gildas.menier@univ-ubs.fr