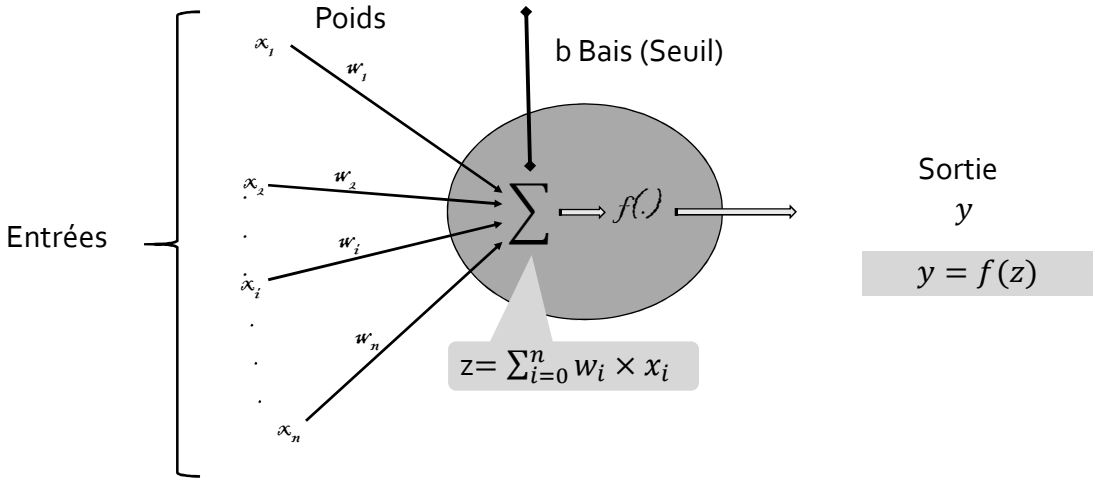
Réseaux de neurones

Réseaux multicouches

Sommaire

- **Rappel : Neurone artificiel**
- Problèmes linéaires
- Problèmes non linéaires
- Choix d'architecture
- Propagation en avant (Forward propagation)

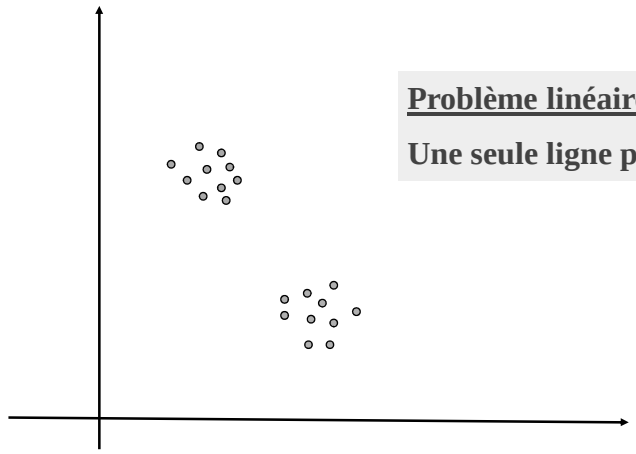
Rappel : Neurone artificiel



Sommaire

- Rappel : Neurone artificiel
- **Problèmes linéaires**
- Problèmes non linéaires
- Choix d'architecture
- Propagation en avant (Forward propagation)

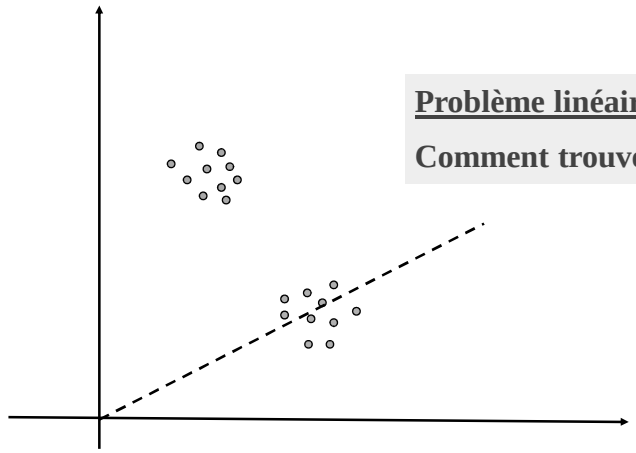
Problèmes linéaires



Problème linéaire

Une seule ligne permet de séparer les données

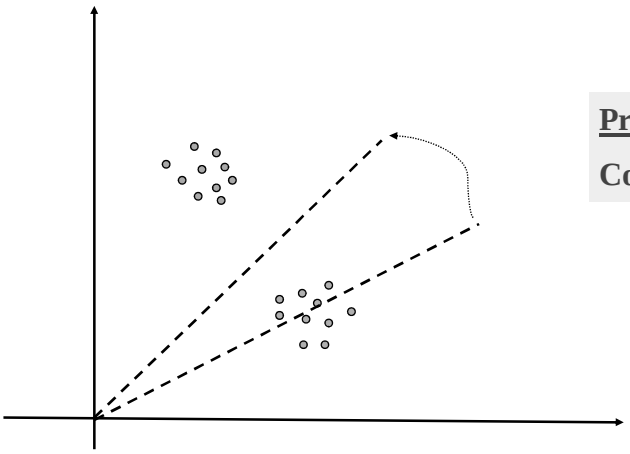
Problèmes linéaires



Problème linéaire

Comment trouver la ligne

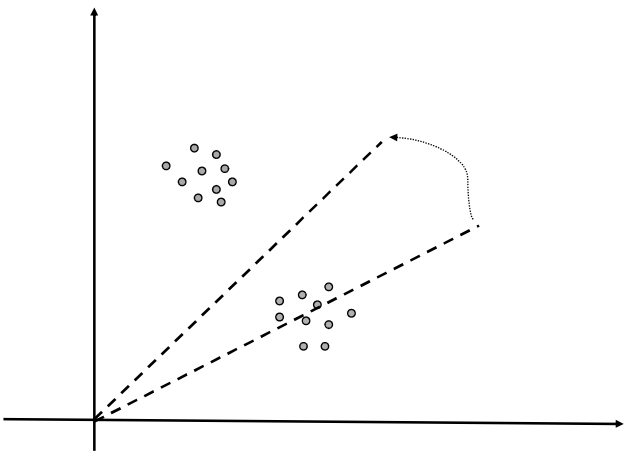
Problèmes linéaires

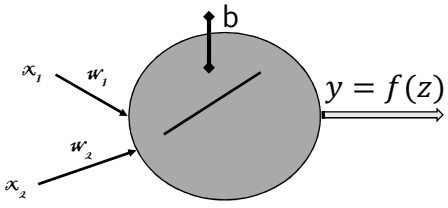


Problème linéaire

Corriger progressivement l'erreur

Problèmes linéaires





Problèmes linéaires: AND et OR

AND

And	Vrai	faux
Vrai	Vrai	faux
Faux	faux	faux

⟺

X_1	X_2	t
1	1	1
1	0	0
0	1	0
0	0	0

OR

Or	Vrai	faux
Vrai	Vrai	Vrai
Faux	Vrai	faux

⟺

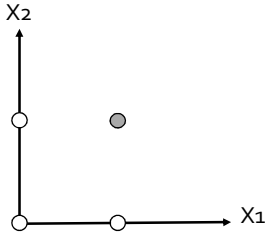
X_1	X_2	t
1	1	1
1	0	1
0	1	1
0	0	0

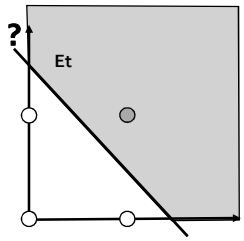
Problèmes linéaires: AND

And	Vrai	faux
Vrai	Vrai	faux
Faux	faux	faux

⟺

X_1	X_2	t	
1	1	1	●
1	0	0	○
0	1	0	○
0	0	0	○

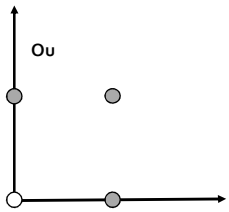


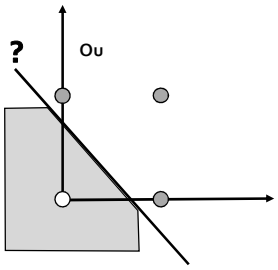


Problèmes linéaires: OR

Ou	Vrai	faux
Vrai	Vrai	Vrai
Faux	Vrai	faux

X_1	X_2	t	
1	1	1	●
1	0	1	●
0	1	1	●
0	0	0	○

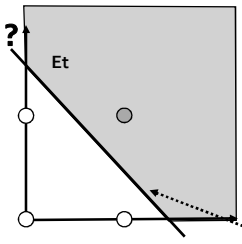




Problèmes linéaires: AND et OR

And	Vrai	faux
Vrai	Vrai	faux
Faux	faux	faux

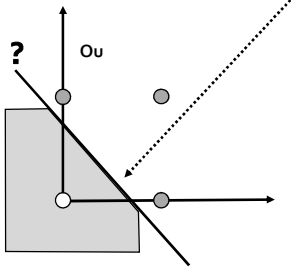
X_1	X_2	t
1	1	1
1	0	0
0	1	0
0	0	0



Limite de décision (decision boundary)

Ou	Vrai	faux
Vrai	Vrai	Vrai
Faux	Vrai	faux

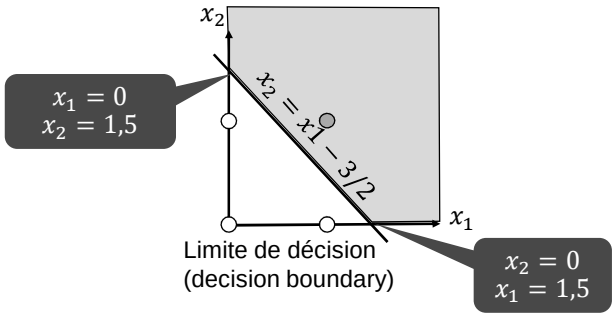
X_1	X_2	t
1	1	1
1	0	1
0	1	1
0	0	0



Problèmes linéaires: La résolution du AND (Et)

And	Vrai	faux
Vrai	Vrai	faux
Faux	faux	faux

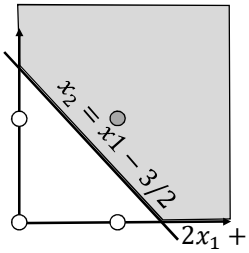
X1	X2	t
1	1	1
1	0	0
0	1	0
0	0	0



Problèmes linéaires: La résolution du AND (Et)

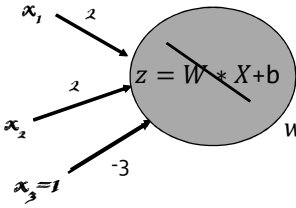
And	Vrai	faux
Vrai	Vrai	faux
Faux	faux	faux

X1	X2	t
1	1	1
1	0	0
0	1	0
0	0	0



$2x_1 + 2x_2 - 3 = 0$

Limite de décision
(decision boundary)



$w_1 * x_1 + w_{2,} * x_2 + b = 0$

$$W = \begin{bmatrix} 2 \\ 2 \\ -3 \end{bmatrix}$$

$2x_1 + 2x_2 - 3 = 0$

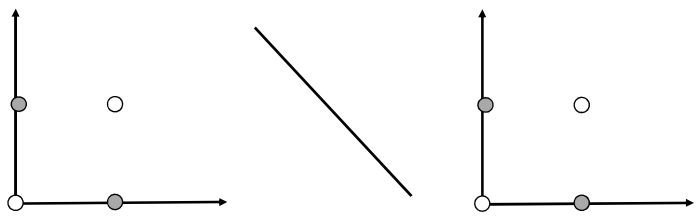
Sommaire

- Rappel : Neurone artificiel
- Problèmes linéaires
- **Problèmes non linéaires**
- Choix d'architecture
- Propagation en avant (Forward propagation)

Problèmes non linéaires: XOR logique

XOr	Vrai	faux
Vrai	Faux	Vrai
Faux	Vrai	faux

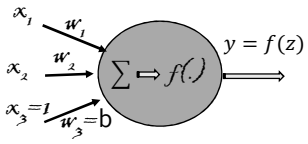
X_1	X_2	t	
1	1	0	○
1	0	1	○
0	1	1	○
0	0	0	○

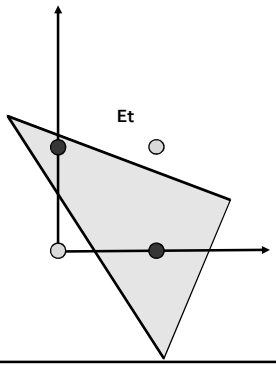


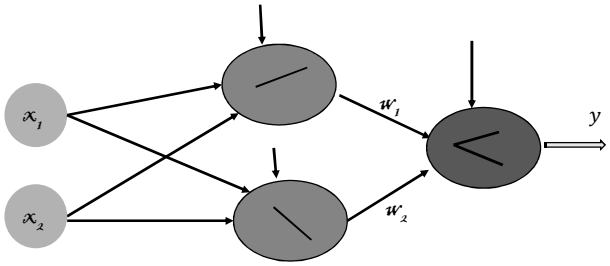
Problèmes non linéaires: La résolution du XOR

XOR	Vrai	faux
Vrai	faux	Vrai
Faux	Vrai	faux

X1	X2	t
1	1	0
1	0	1
0	1	1
0	0	0

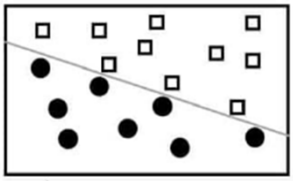


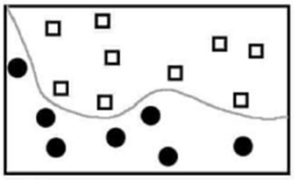




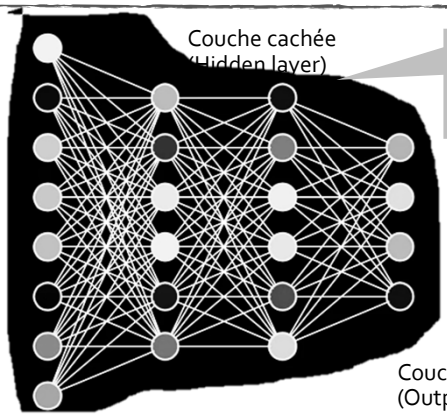
Problèmes non linéaires

Les réseaux neuronaux sont souvent utilisés pour résoudre des problèmes non linéaires, c'est-à-dire des problèmes qui ne peuvent pas être résolus en séparant les classes par une ligne droite.





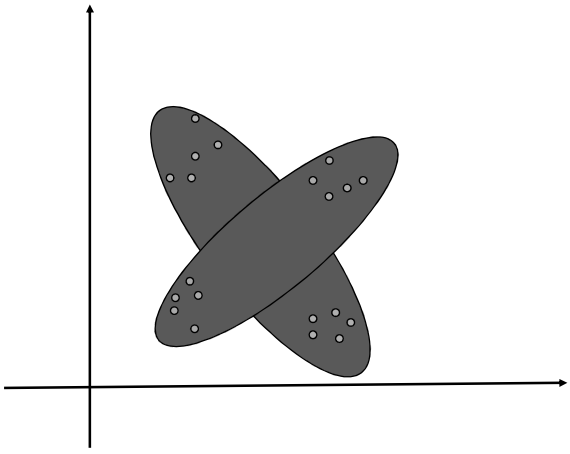
Problèmes non linéaires : Notion de couches



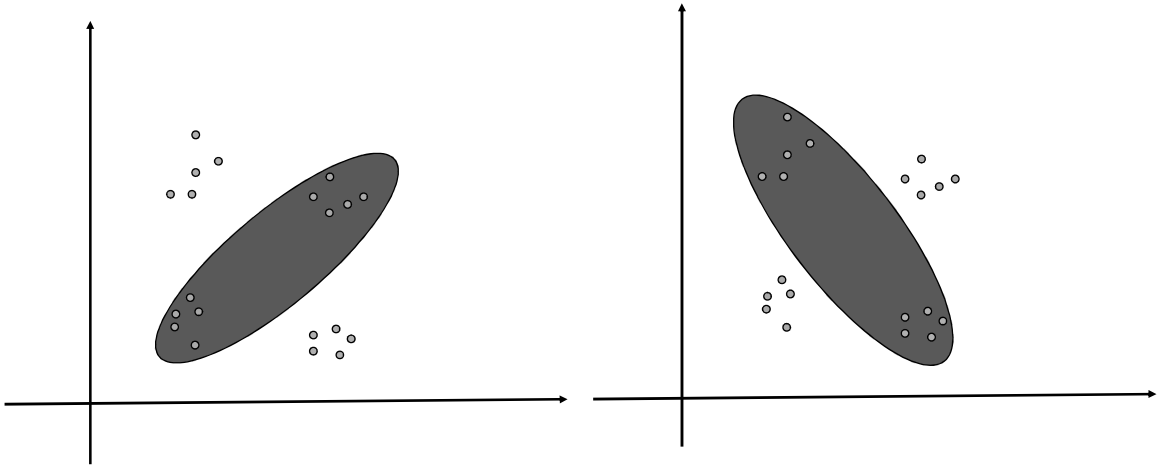
Les couches cachées sont utilisées uniquement dans le cas de données non linéairement séparables.

- **En général, un seul neurone, même avec de nombreuses entrées, peut ne pas être suffisant. Parfois, Il faut plusieurs neurones, fonctionnant en parallèle, on parle alors de "couche".**

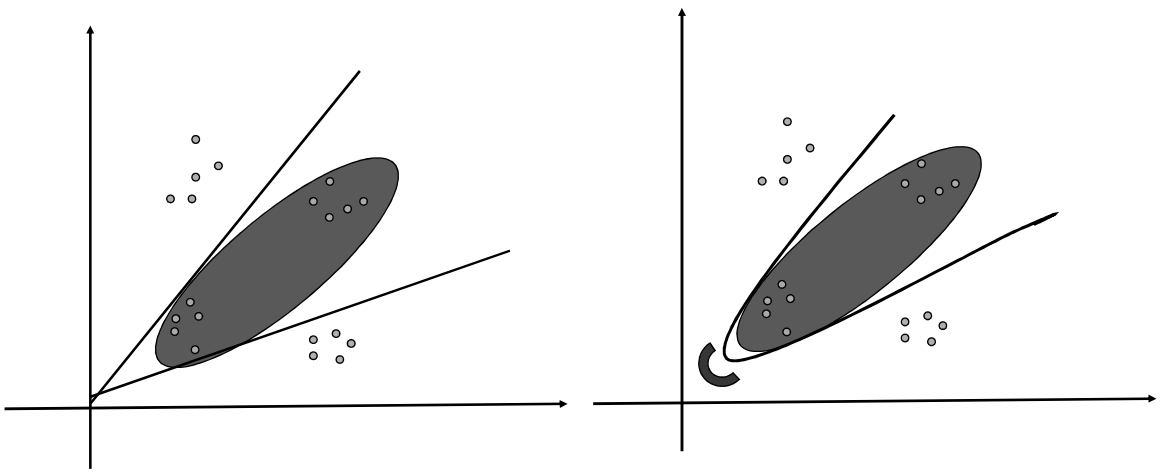
Problèmes non linéaires: Exemple 2



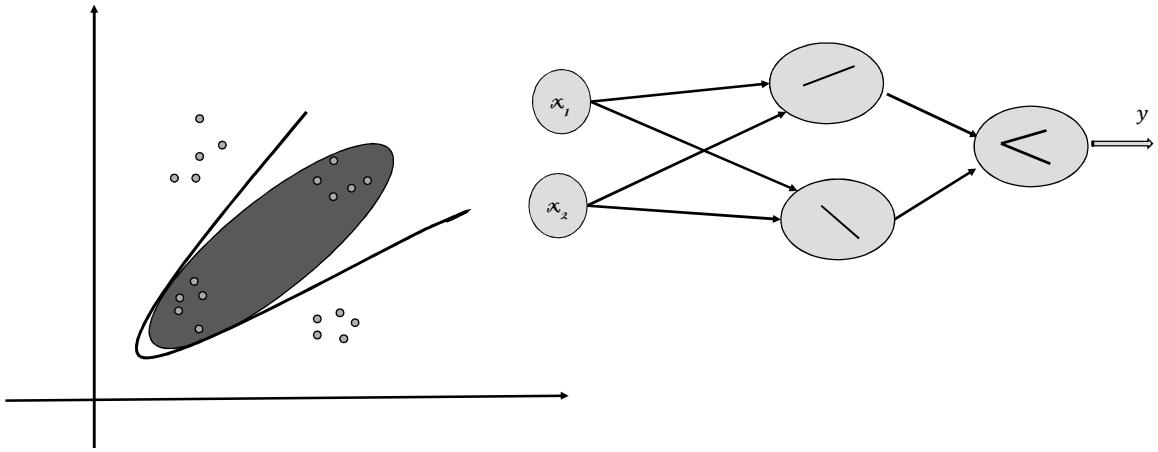
Problèmes non linéaires: Exemple 2



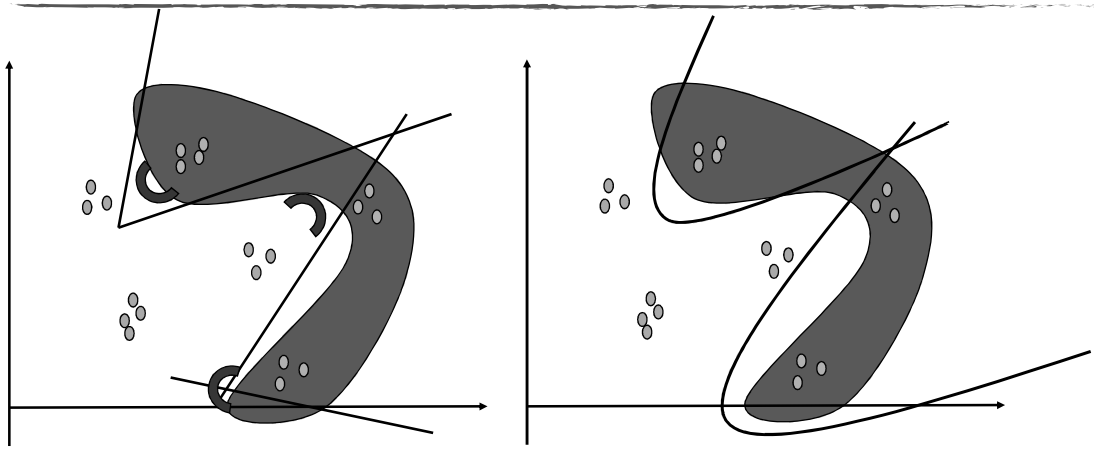
Problèmes non linéaires: Exemple 2



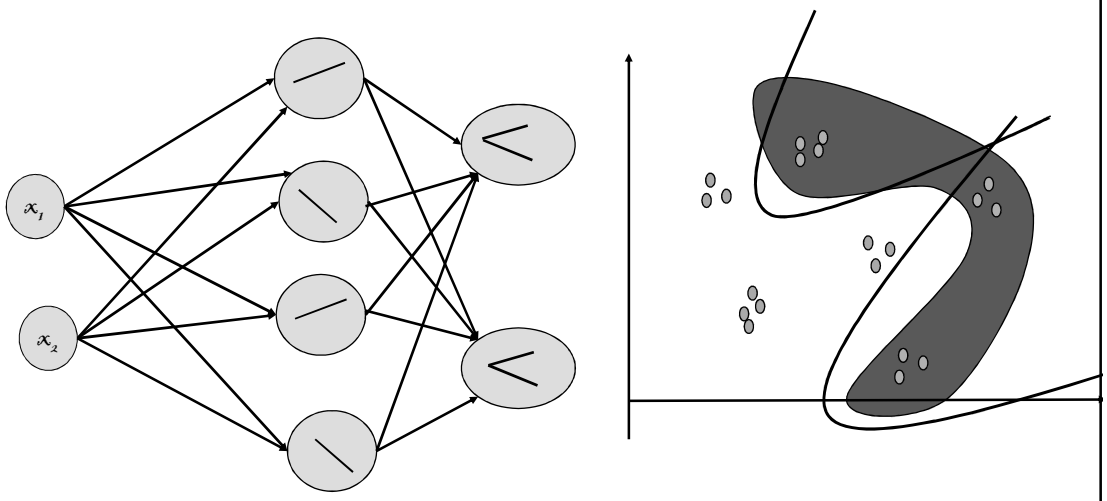
Problèmes non linéaires: Exemple 2



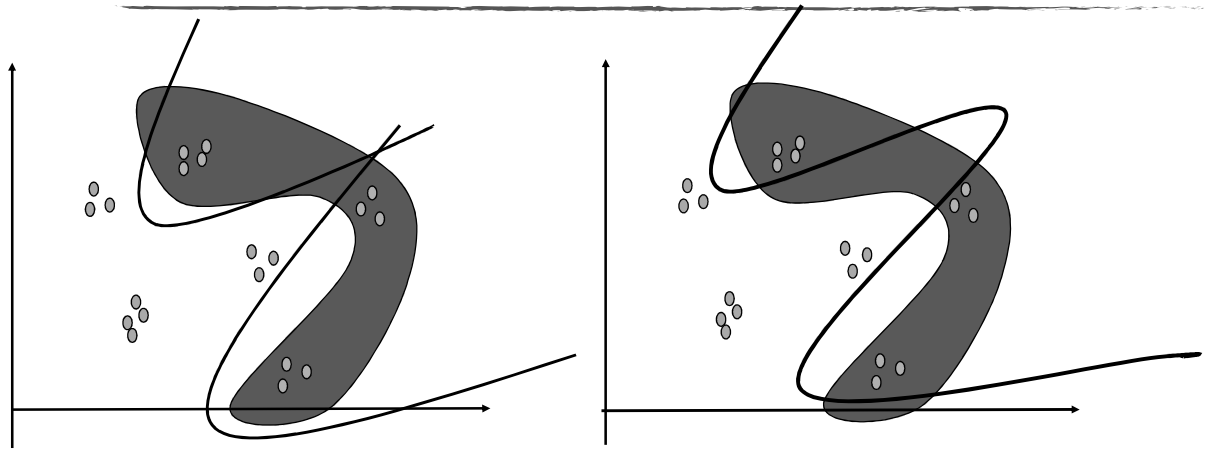
Problèmes non linéaires: Exemple 3



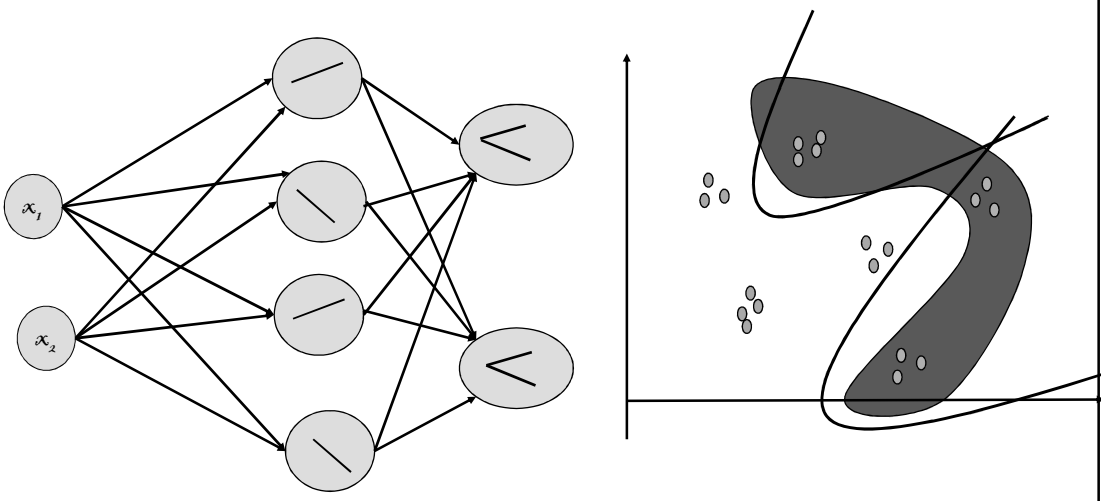
Problèmes non linéaires: Exemple 3



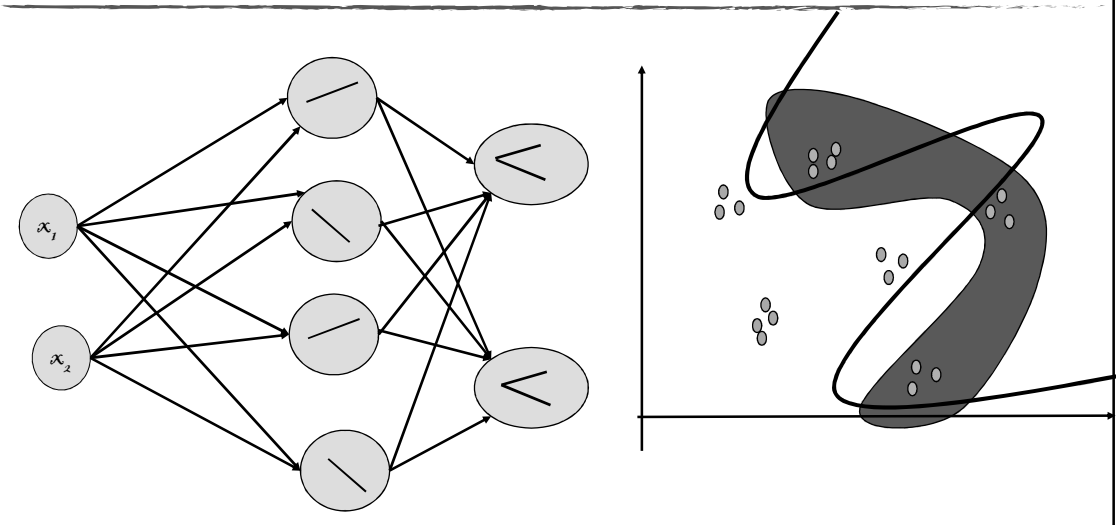
Problèmes non linéaires: Exemple 3



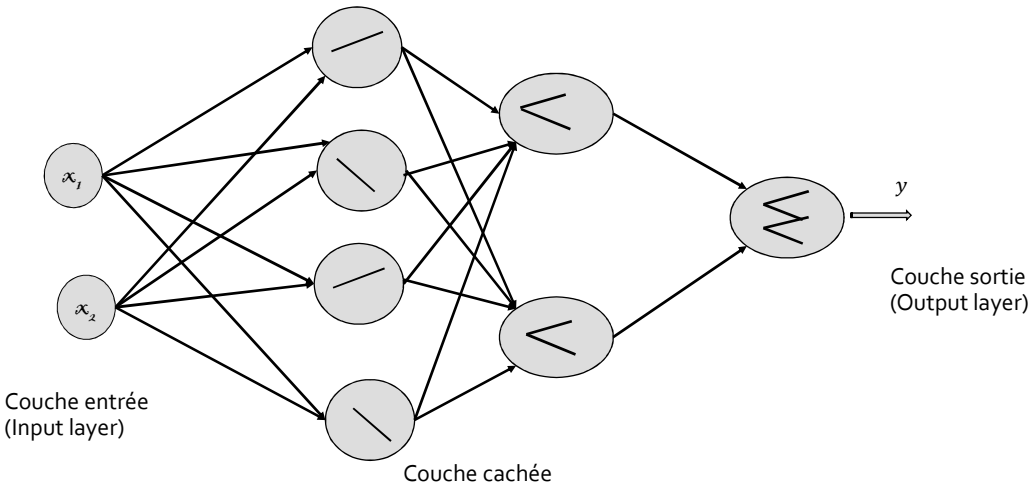
Problèmes non linéaires: Exemple 3



Problèmes non linéaires: Exemple 3



Problèmes non linéaires: Exemple 3



Sommaire

- Rappel : Neurone artificiel
- Problèmes linéaires
- Problèmes non linéaires
- **Choix d'architecture**
- Propagation en avant (Forward propagation)

Choix d'architecture

- Le nombre d'entrées du réseau et le nombre de sorties du réseau sont définis par les spécificités du problème à résoudre.
  - Par exemple, s'il y a quatre variables à utiliser comme entrées, il y a quatre entrées dans le réseau. De même, si le réseau doit produire sept sorties, il doit y avoir sept neurones dans la couche de sortie.
- Les caractéristiques souhaitées du signal de sortie aident également à sélectionner la fonction de transfert pour la couche de sortie.
  - Par exemple, si une sortie doit être soit 1 soit -1, alors une fonction de transfert symétrique doit être utilisée.
- Quant au nombre de couches cachées, la plupart des réseaux de neurones n'ont que deux ou trois couches et rarement quatre couches. Pour un nombre de couches plus important on parle de Deep learning.

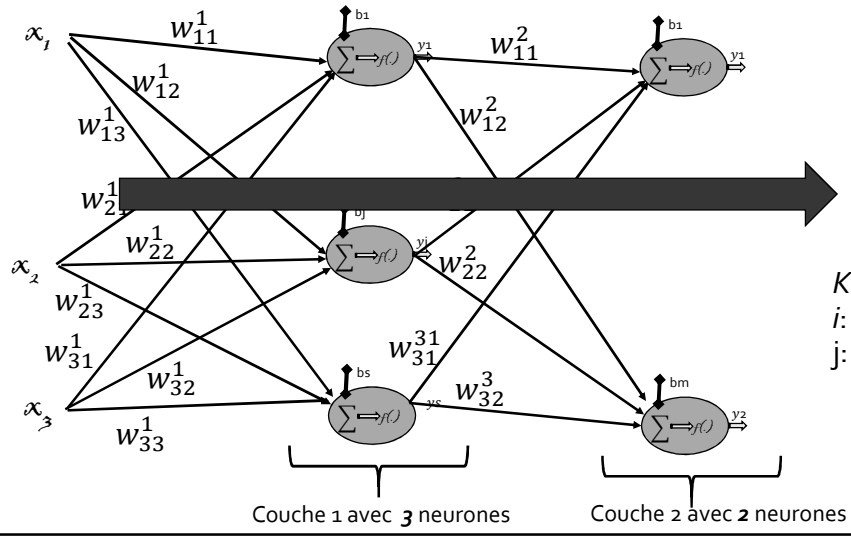
Choix d'architecture

- On peut choisir des neurones avec ou sans biais.
- Pratiquement, le biais donne au réseau une variable supplémentaire.
- Les réseaux avec biais sont plus puissants que ceux qui n'en ont pas.
- Vu qu'un neurone sans biais aura toujours une entrée égale à zéro lorsque les entrées du réseau sont nulles. Cela n'est pas forcément souhaitable et peut être évité par l'utilisation d'un biais.

Sommaire

- Rappel : Neurone artificiel
- Problèmes linéaires
- Problèmes non linéaires
- Choix d'architecture
- **Propagation en avant (Forward propagation)**

Propagation en avant (Forwad propagation)



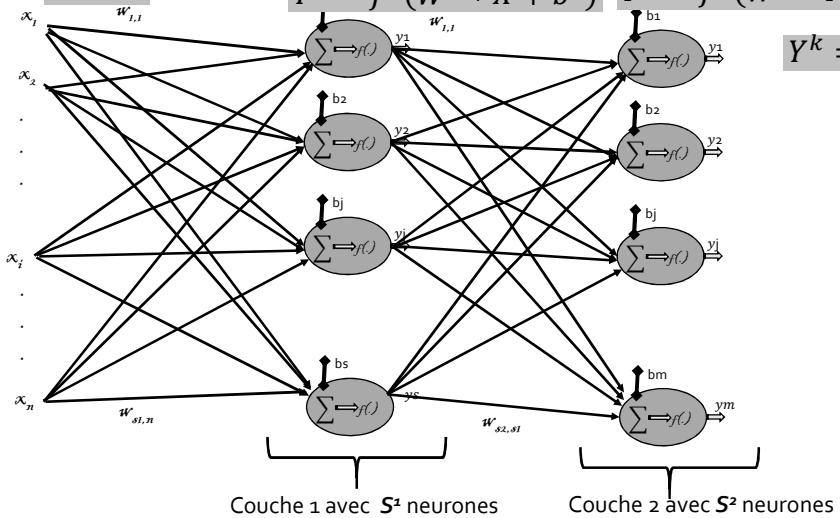
$$w_{ij}^k$$

K: Numéro de couche
i: Numéro d'entrée
j: Numéro du neurone

Propagation en avant (Forwad propagation)

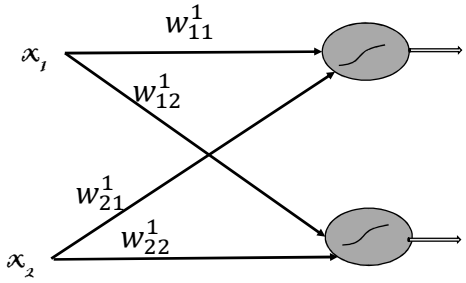
$Y^0 = X$ $\quad Y^1 = f^1(W^1 * X + b^1)$ $\quad Y^2 = f^2(W^2 * Y^1 + b^2)$

$Y^k = f^k(W^k * Y^{k-1} + b^k)$



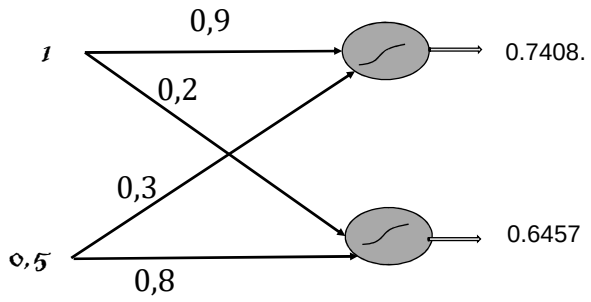
Couche 1 avec S^1 neurones Couche 2 avec S^2 neurones

Forwad propagation : Application 1



Variable	valeur
x1	1
x2	0,5
w_{11}^1	0,9
w_{12}^1	0,2
w_{21}^1	0,3
w_{22}^1	0,8

Forwad propagation : Application 1



z1 = (0.9*1) + (0.3*0,5)= 0.9 + 0.15 = 1.05

y1 = 1 / (1 + 0.3499) = 1 / 1.3499 = 0.7408

z2 = (0.2*1) + (0.8*0,5) = 0.2 + 0.4= 0,6

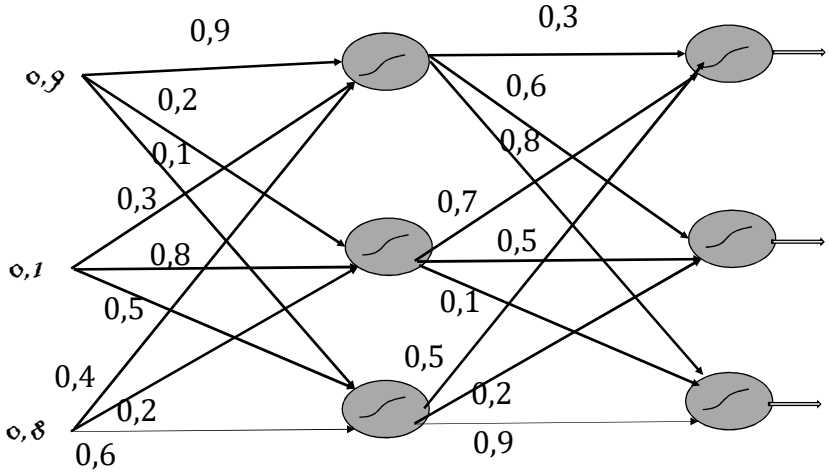
y2 = 1/(1 + 0.5488) = 1/(1.5488) = 0.6457.

$$\begin{matrix} 0,9 & 0,3 \\ 0,2 & 0,8 \end{matrix} \times \begin{matrix} 1 \\ 0,5 \end{matrix}$$

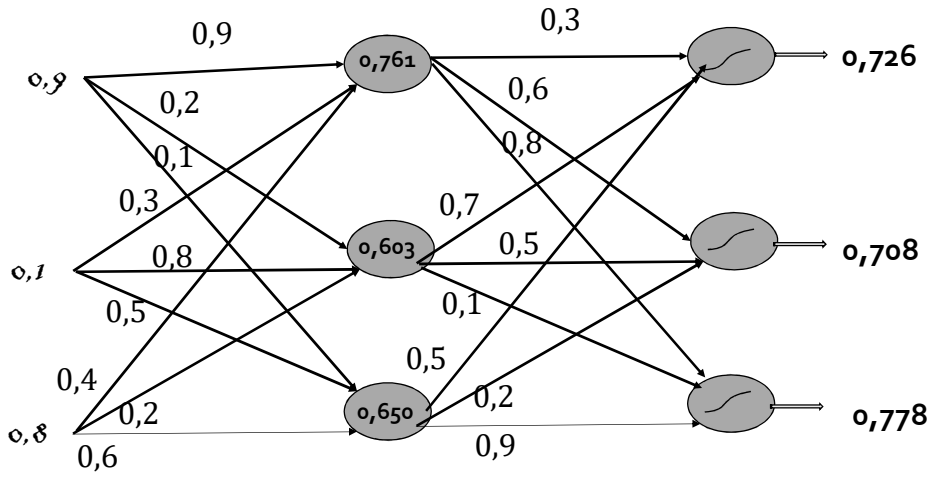
Z = W ◦ X

Y= sigmoid (Z)

Forwad propagation : Application 2

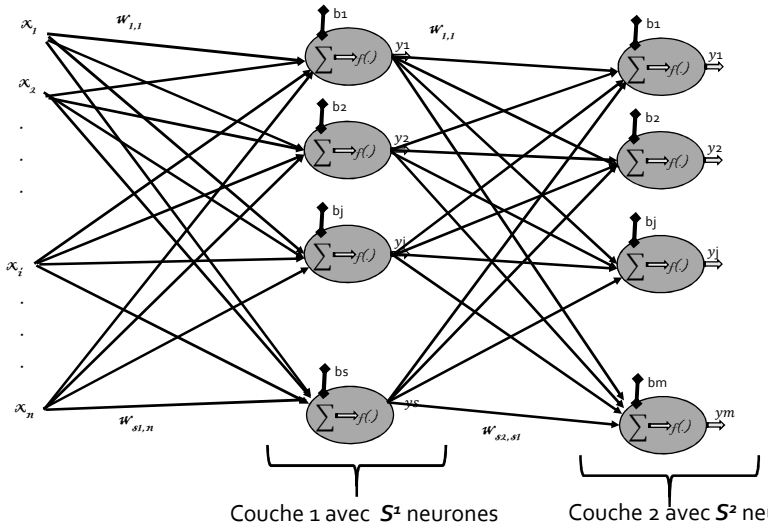


Forwad propagation : Application 2



Programmation

Propagation en avant (Forwad propagation)



$$Y^0 = X$$

$$Y^1 = f^1(W^1 * X + b^1)$$

$$Y^2 = f^2(W^2 * Y^1 + b^2)$$

$$\dots$$

$$Y^i = f^i(W^i * Y^{i-1} + b^i)$$

$$\dots$$

$$Y^k = f^k(W^k * Y^{k-1} + b^k)$$