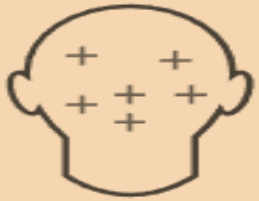
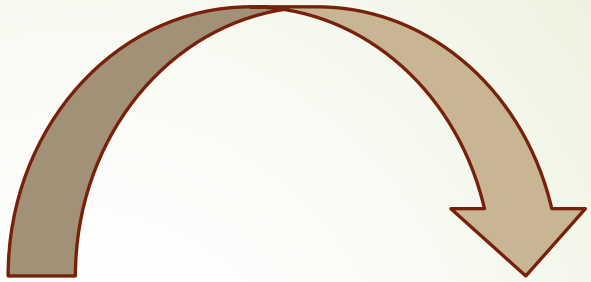
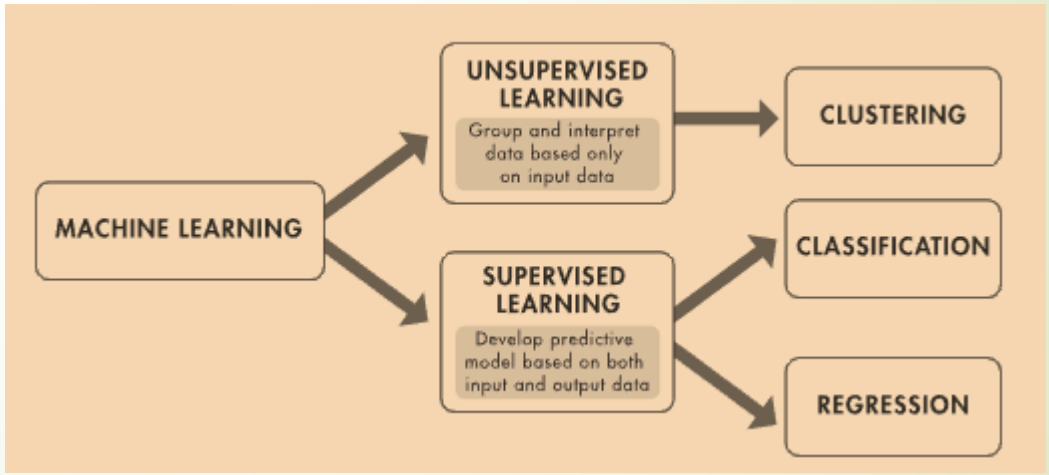
Intelligence artificielle



L'élaboration manuelle de règles ou d'équations est trop complexe, comme pour la reconnaissance faciale ou la reconnaissance vocale.





L'apprentissage supervisé entraîne le modèle sur des données entrée sortie connues. L'apprentissage non supervisé cherche des similitudes aux entrées présentées.

Intelligence artificielle

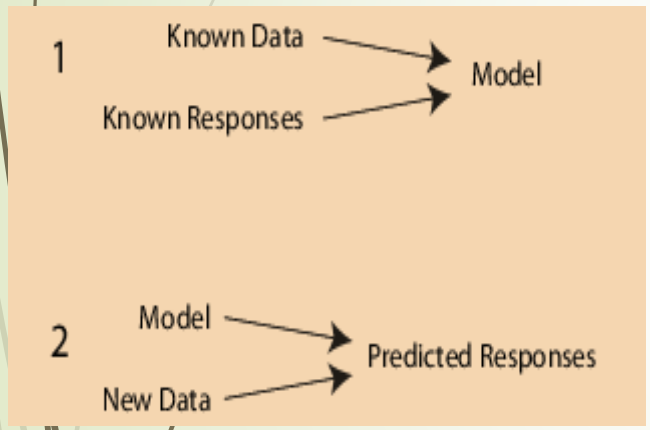
Exemple d'apprentissage supervisé

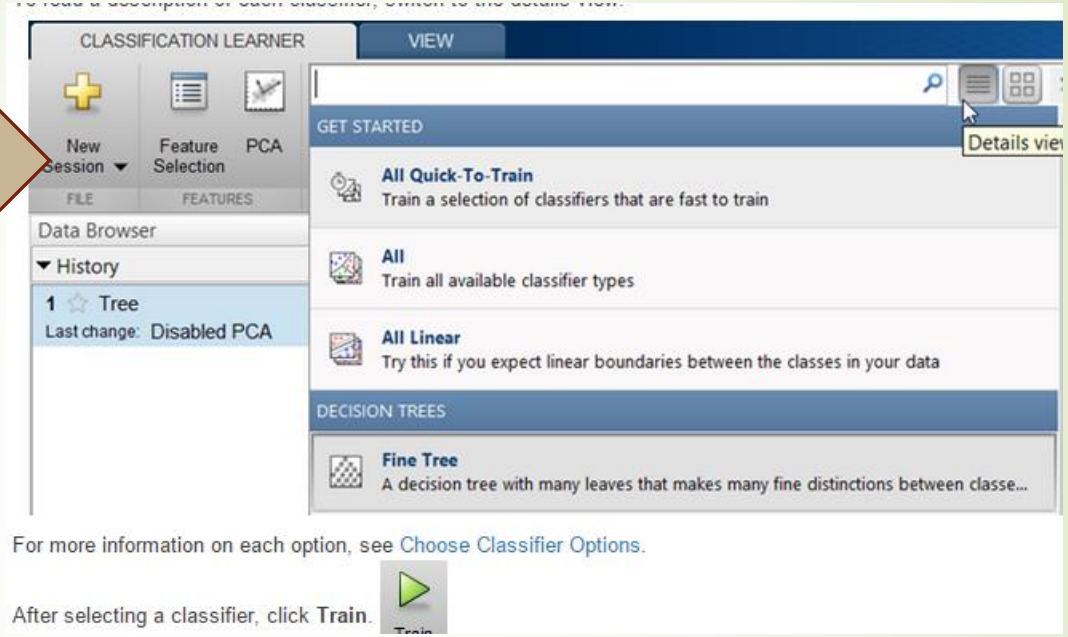
Prévoir les crises cardiaques à l'aide de l'apprentissage supervisé

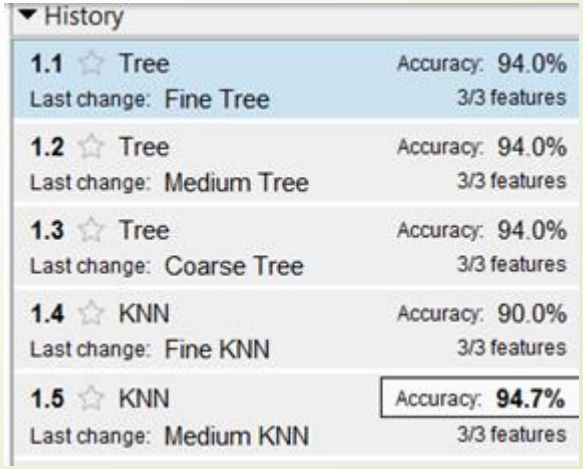
Supposons que les médecins veuillent prévoir la probabilité d'un individu d'être atteint d'un infarctus dans l'année. Ils disposent de données sur d'autres patients, comme l'âge, le poids, la taille et la tension artérielle. Ils savent si les autres patients ont été atteints d'un infarctus dans l'année. Le problème consiste donc à combiner les données existantes dans un modèle capable de prévoir si un nouvel individu fera un infarctus dans l'année.

Intelligence artificielle

Un outil d'apprentissage supervisé dans Matlab







Le logiciel va proposer plusieurs modèles pour classifier les données.

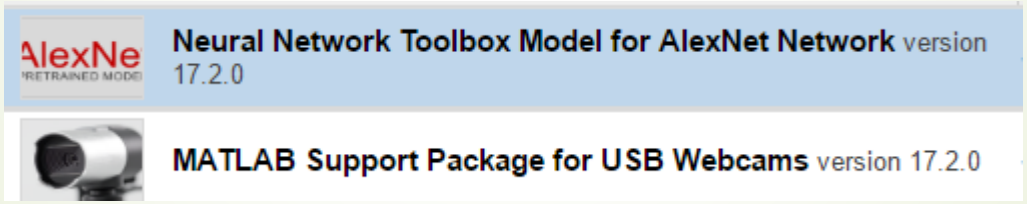
Intelligence artificielle

Modèle déjà élaboré

Identifier des objets à l'aide d'une webcam et d'une base de données. **AlexNet** *est un réseau neuronal de convolution pré-entraîné (CNN) formé sur plus d'un million d'images et capable de classer les images en 1 000 catégories d'objets (par exemple, clavier, souris, tasse à café, crayon et de nombreux animaux).*

Etapes sous Matlab

**Installer les supports package:*

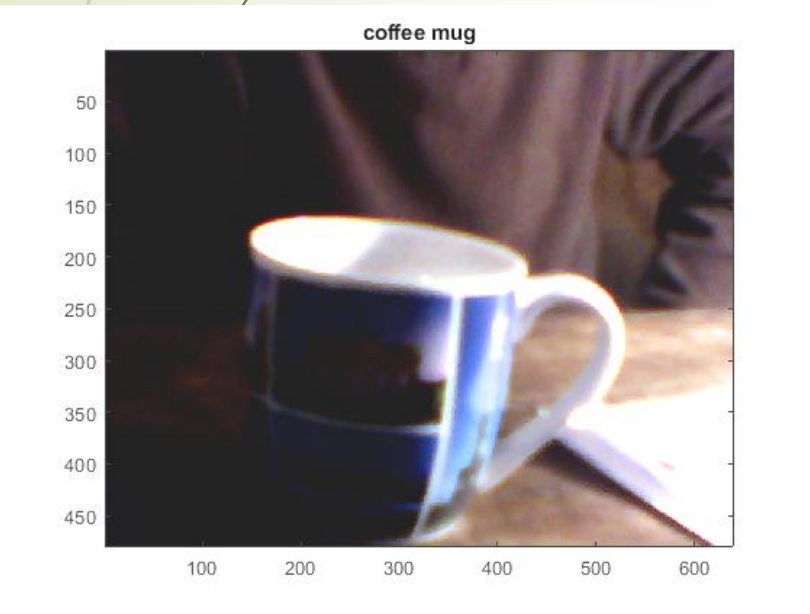


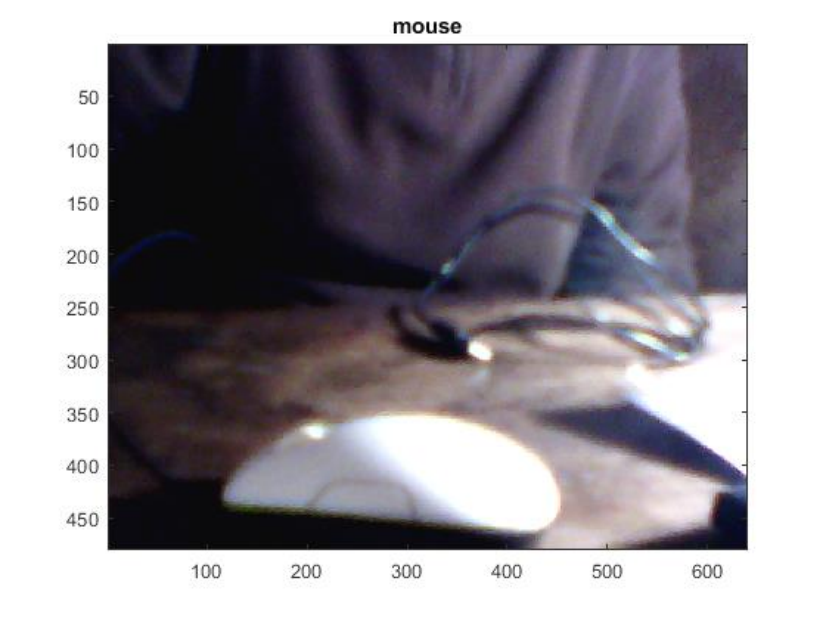
**Tester les supports package:*
Webcamlist ….
camera=webcam par exemple
puis éditer dans la fenêtre de commandes:

Intelligence artificielle

Etapes sous Matlab

```
net=alexnet
while true
    im = snapshot(camera);          % Acquérir une image
    image(im);                      % Afficher l'image
    im = imresize(im,[227 227]);    % Redimensioner l'image
    label = classify(net,im);       % Classer l'image
    title(char(label));             % Afficher la classe
    drawnow
end
```





Intelligence artificielle

Exemple 2 matlab

*Ouvrir classification leaner dans APPS
*load('ClassificationLearner_Example_Datasets.mat')
%espace commandes
*Placer le fichier ClassificationLearner_Example_Datasets.mat dans le répertoire courant, cliquer dessus pour faire apparaître les tables
*Cliquer sur new session puis workspace puis FisherIris.

Réponse (espèce)

Prédicteurs

Workspace Variable

FisherIris 150x5 table

Response

Species categorical 3 unique

Predictors

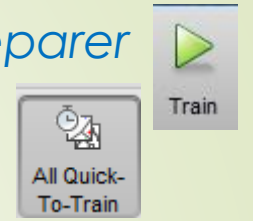
	Name	Type	Range
☑	SepalLength	double	4.3 .. 7.9
☑	SepalWidth	double	2 .. 4.4
☑	PetalLength	double	1 .. 6.9
☑	PetalWidth	double	0.1 .. 2.5
☐	Species	categorical	3 unique

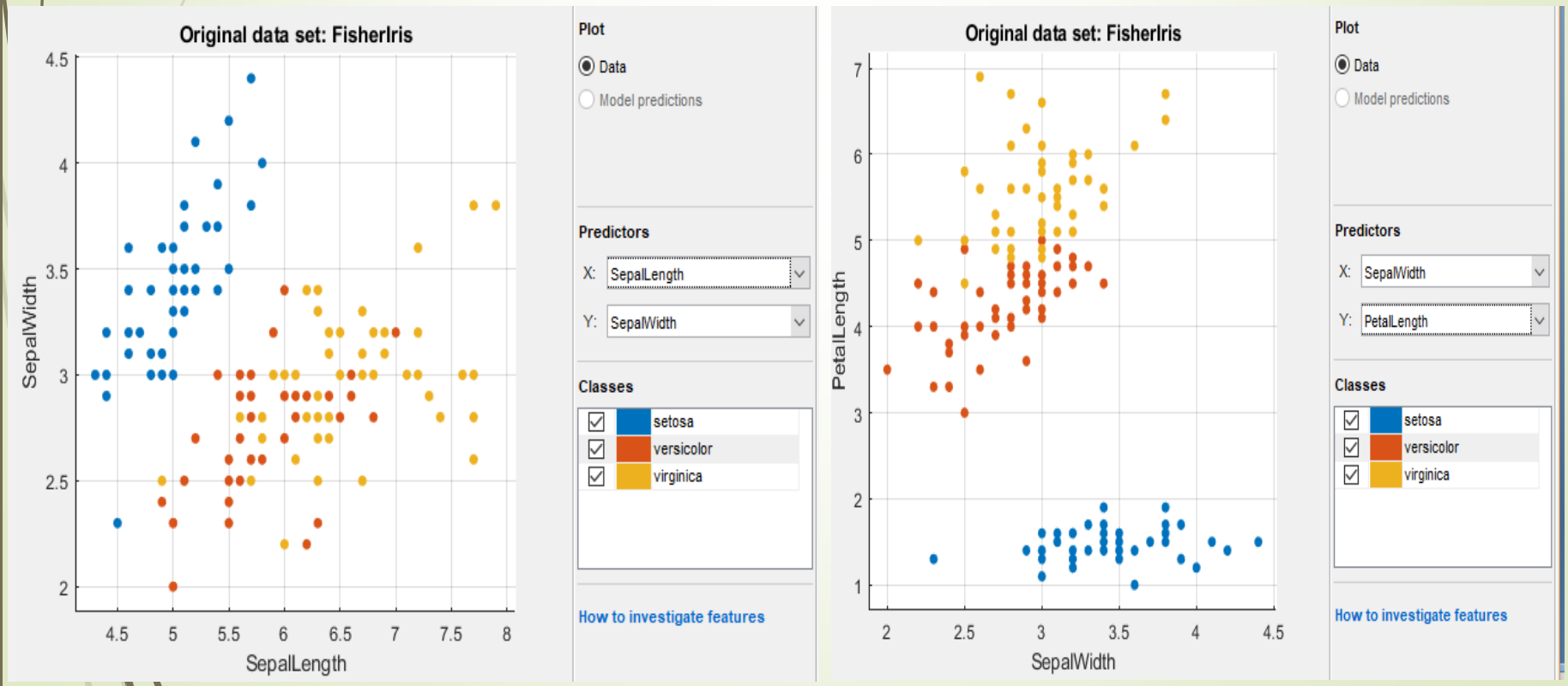
Intelligence artificielle

Classifier une base de données existante

*Lancer la session , changer de prédicteur pour séparer les espèces

*Faire tourner plusieurs modèles pour en choisir un

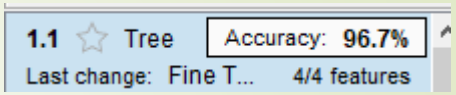


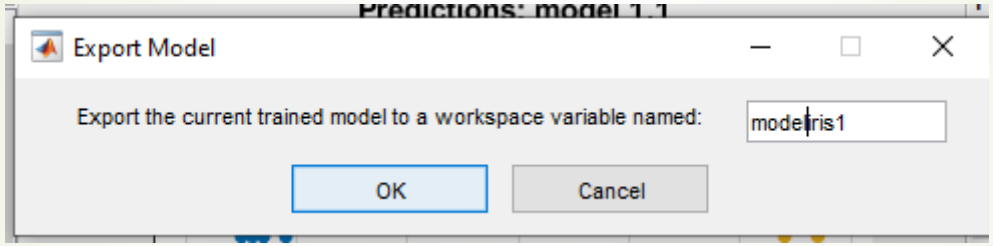


Intelligence artificielle

Tester le modèle choisi

*Exporter le modèle choisi





*Créer une nouvelle entrée:

V2=[5.3 3.6 1.4 0.2]

T2=array2table(V2,'variableName',{'SepalLength','SepalWidth','Petallength','PetalWidth'})

*Tester le modèle : yfit=modeliris1.predictFcn(T2)

*Réponse : categorical : setosa

Intelligence artificielle

Créer et classifier des données

*Créer une matrice contenant les données :

```
essai =

    10     0     0     0     1
    20     1     1     1     1
    25     0     0     0     0
    30     0     1     1     1
    40     1     0     1     1
    60     0     1     1     1
    80     1     1     0     1
    90     1     1     1     1
```

*Créer une table avec des prédicteurs à partir de cette matrice:

T = array2table (essai,'VariableNames' , { 'age' , 'fumeur' , 'alco','obese','malade' })

age	fumeur	alco	obese	malade
10	0	0	0	1
20	1	1	1	1
25	0	0	0	0
30	0	1	1	1
40	1	0	1	1
60	0	1	1	1
80	1	1	0	1
90	1	1	1	1

Intelligence artificielle

Créer et classifier des données

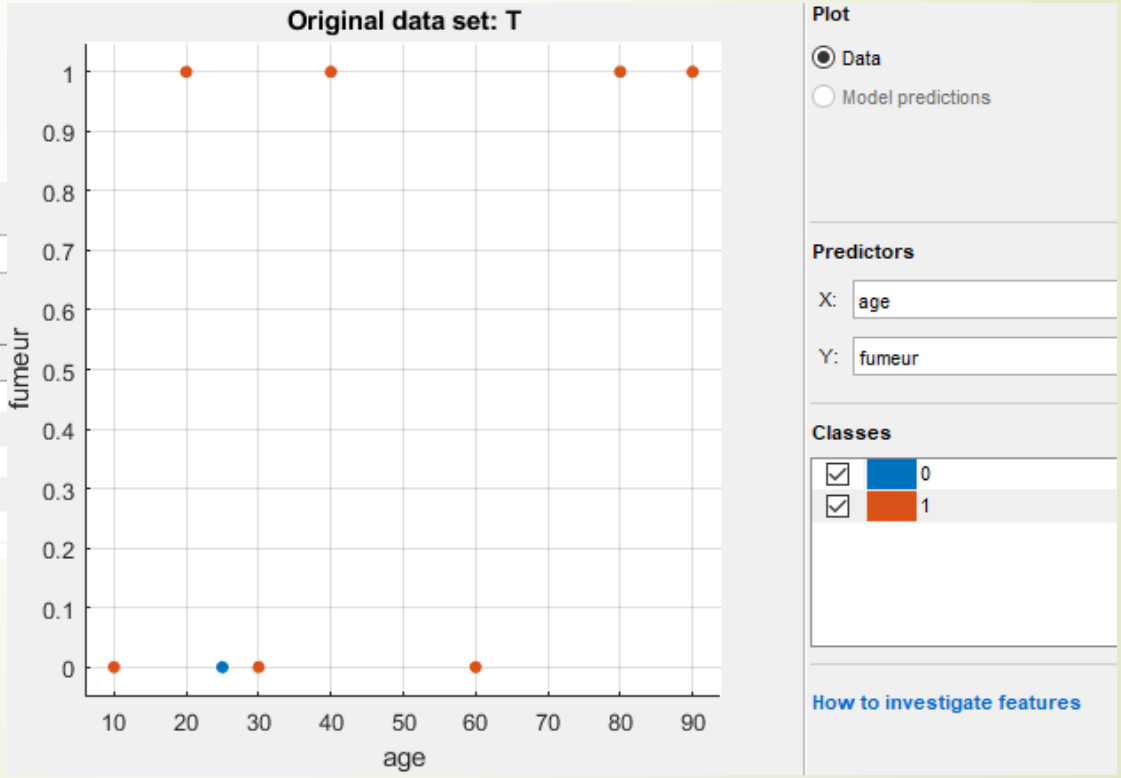
**Importer la table T puis classifier en choisissant les prédicteurs et la réponse .*

Response

malade	double	0 .. 1

Predictors

	Name	Type	Range
☑	age	double	10 .. 90
☑	fumeur	double	0 .. 1
☑	alco	double	0 .. 1
☑	obese	double	0 .. 1
☐	malade	double	0 .. 1



Intelligence artificielle

Utiliser le modèle avec de nouvelles données

*Exporter le modele :Exportcompactmodel puis le nommer.
*Créer une nouvelle table de données **sans la réponse**

```
v1 =

    15     1     0     1
    25     0     0     1
```

→

age	fumeur	alco	obese
15	1	0	1
25	0	0	1

*Lancer le modèle pour obtenir une réponse :

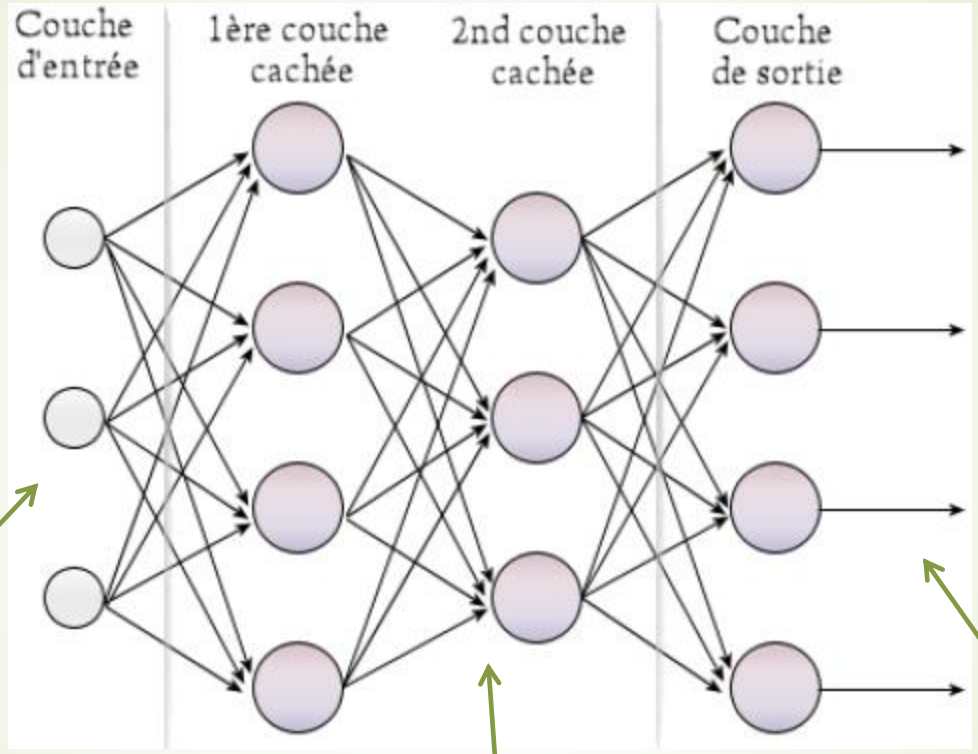
yfit = essaiModel.predictFcn (T1)

*Lire la réponse :

```
yfit =

     1
     0
```

Intelligence artificielle

Réseau de neurones



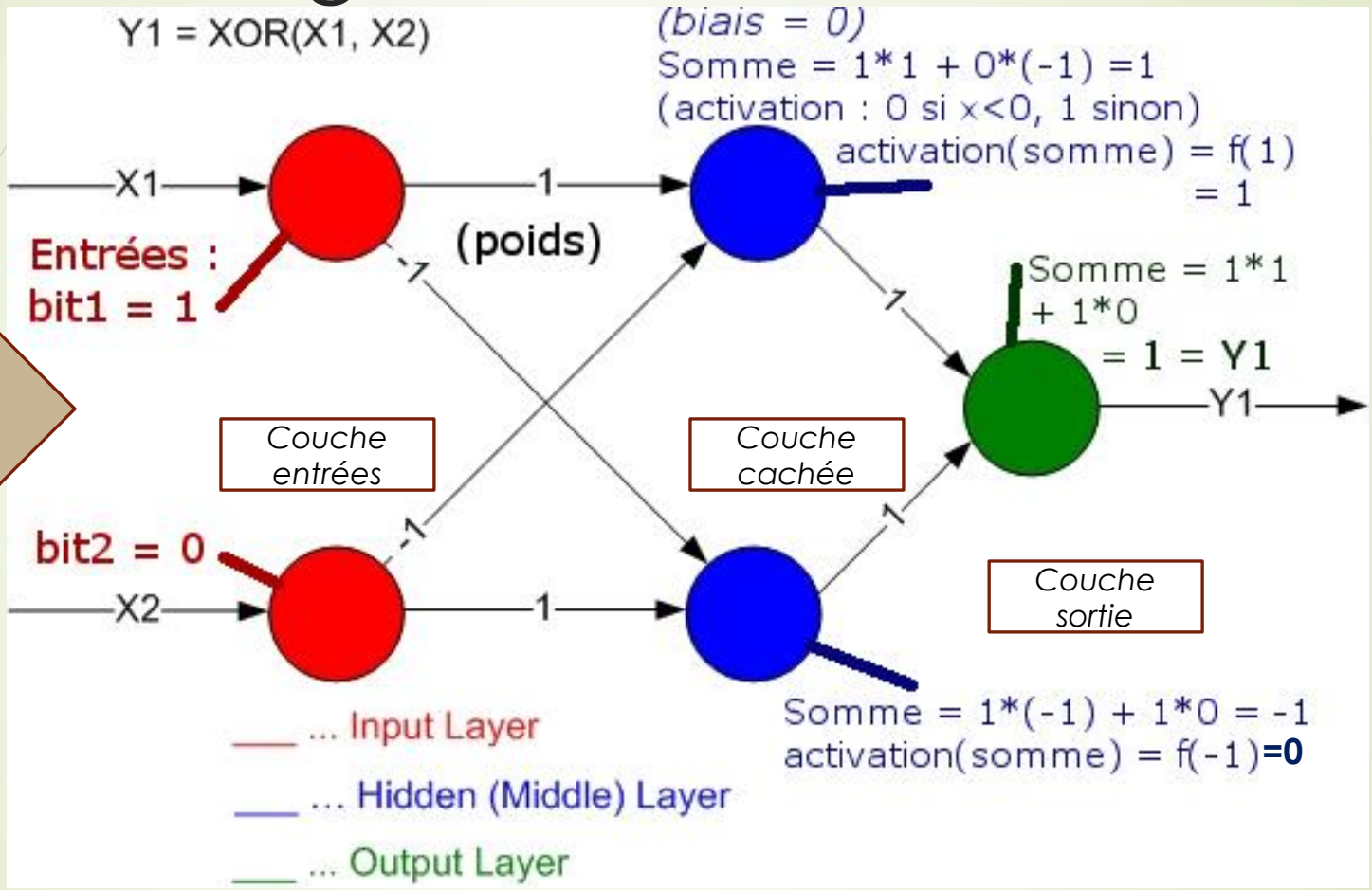
$$y = x$$

$$y = f_{\text{activation}}\left(b + \sum_i w_i \cdot x_i\right)$$

$$y = \sum_i w_i \cdot x_i$$

Intelligence artificielle

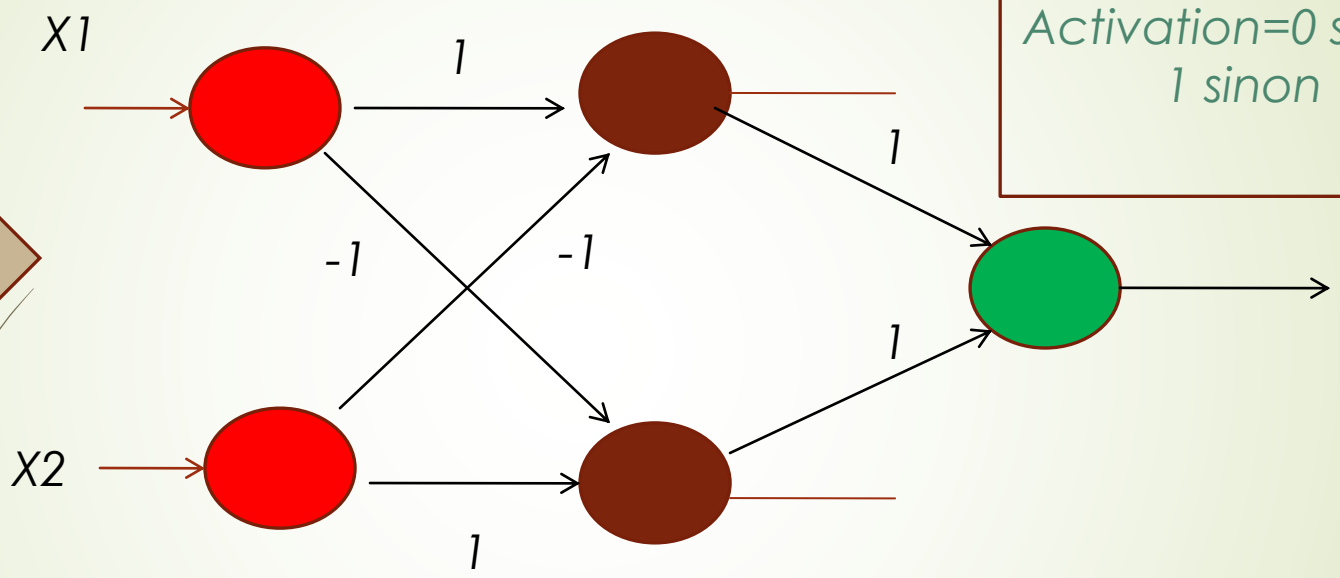
Réseau de neurones exemple XOR



Vérifier que pour que cela fonctionne avec toutes les combinaisons de X1 et X2, il faut que biais =-1.

Intelligence artificielle

Réseau de neurones
Exemple XOR



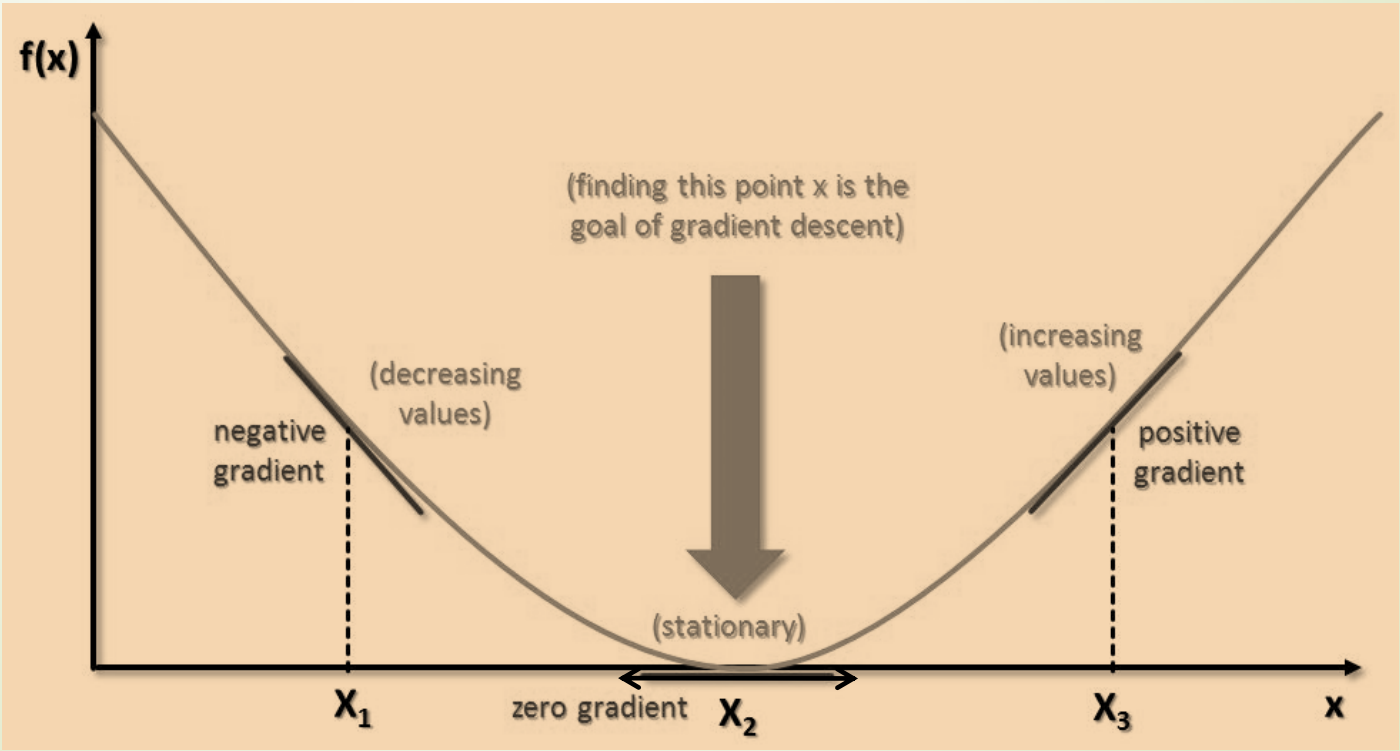
b=-1
Activation=0 si x< 0
1 sinon

X1=X2=1 donne y1caché = y2caché=0 donc ysortie = 0

Intelligence artificielle

Comment apprendre

Les poids et les biais sont les seules variables. On calcule l'erreur puis on la dérive partiellement par rapport au poids. On tente d'atteindre le minimum de la courbe d'erreur. La mise à jour d'un poids $w_i := w_i - \eta \cdot \alpha.$ *avec* η *vitesse d'apprentissage et* α *la pente de l'erreur.*



Fonctionnement du neurone artificiel

x_i : les vecteurs d'entrées, viennent des sorties d'autres neurones, ou de stimulis sensoriels (capteur visuel, sonore…) ;

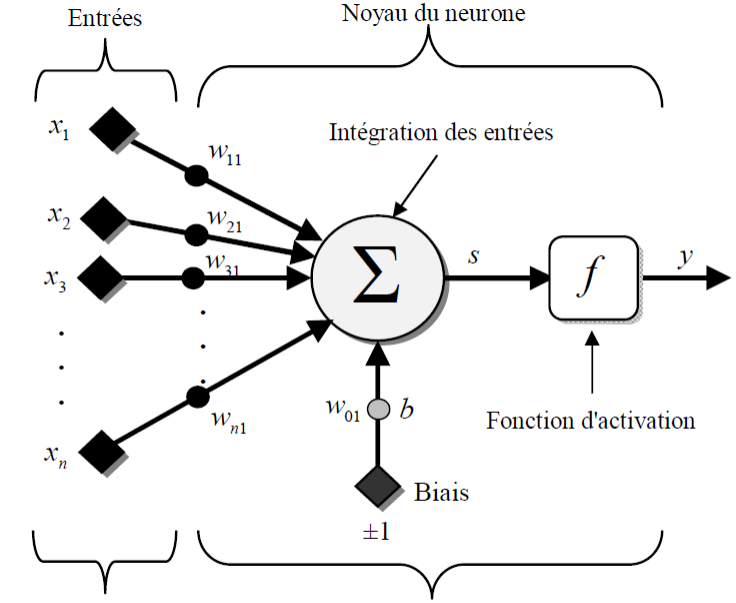


Fig.4 - Modèle d'un neurone artificiel.

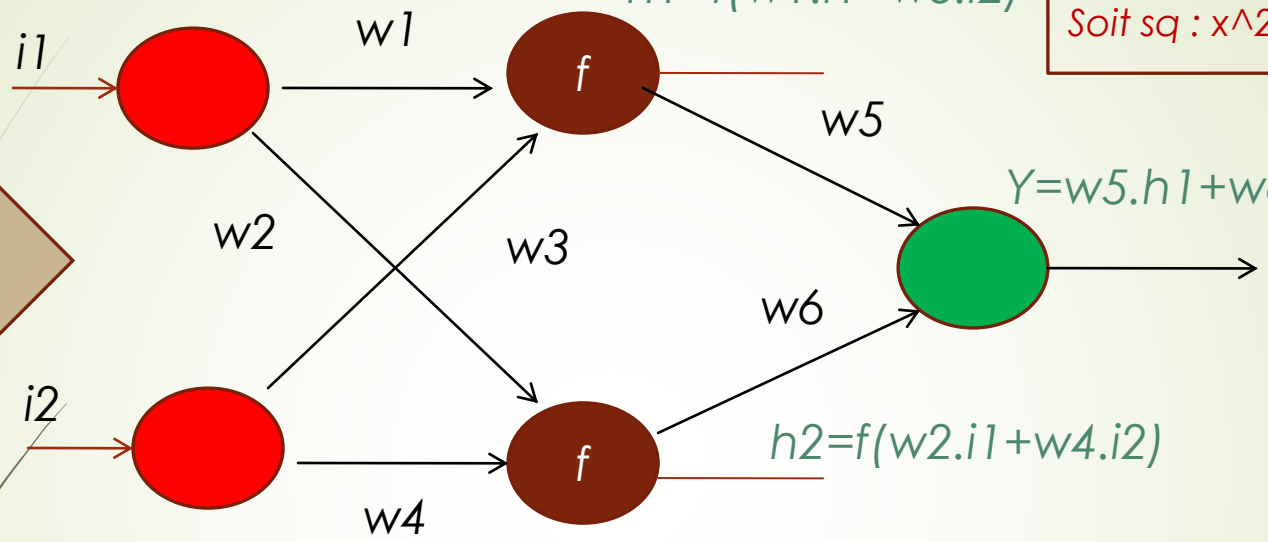
Noyau : intègre toutes les entrées et le biais et calcule la sortie du neurone selon une fonction d'activation qui est souvent non linéaire pour donner une plus grande flexibilité d'apprentissage

w_{ij} : poids synaptiques du neurone j . Ils correspondent à l'efficacité synaptique dans les neurones biologiques ($w_{ij} > 0$: synapse excitatrice; $w_{ij} < 0$: synapse inhibitrice). Ces poids pondèrent les entrées et peuvent être modifiés par apprentissage

Biais : valeurs -1 ou +1 qui permet d'ajouter de la flexibilité au réseau en permettant de varier le seuil de déclenchement du neurone par l'ajustement des poids et du biais lors de l'apprentissage;

Intelligence artificielle

Comment apprendre exemple



Erreur=E=0.5 * (y-ye)^2
Soit sq : x^2/2 donc sq'=x

$dE/dw5=d\ Sq(y-ye)\ /dw5 = Sq'\ (y-ye)\ .d(y-ye)/dw5 = (y-ye).h1$
De même $dE/dw6= (y-ye).h2$

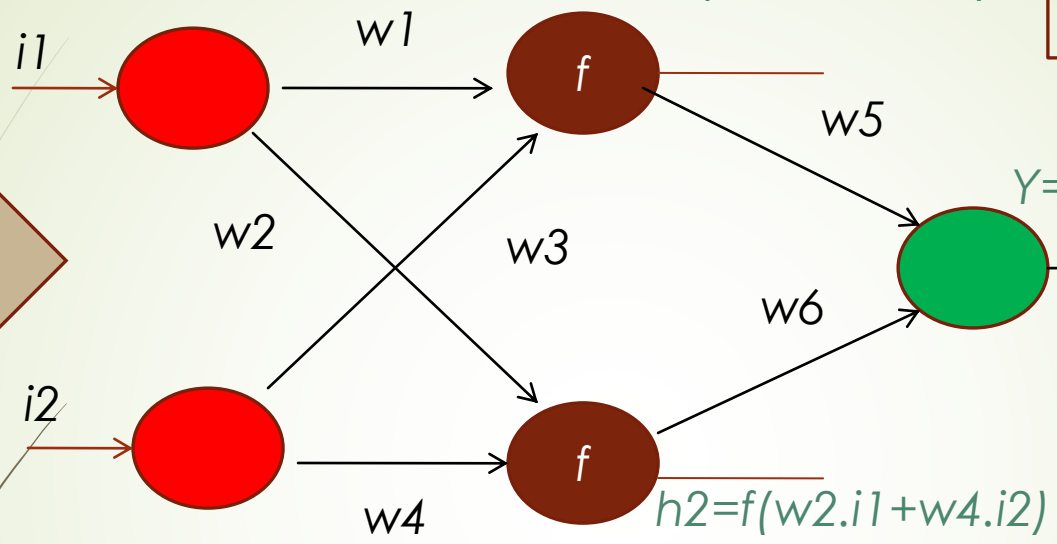
$w5=w5-\eta h1.(y-ye)$

$w6=w6-\eta h2.(y-ye)$

Mise à jour des poids de sortie w5 et w6.

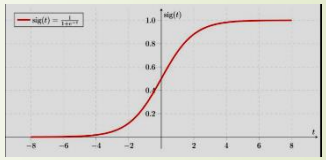
Intelligence artificielle

Comment apprendre exemple



Erreur=E=0.5 * (y-ye)^2
Soit sq : x^2/2 donc sq'=x

f=sig(x)=1/(1+exp(-x))
f'= sig(1-sig)=f(1-f)



$dE/dw1=d\ Sq(y-ye)\ /dw1 = Sq'\ (y-ye)\ .d(y-ye)/dw1 = (y-ye).d(y)/dw1$
$=w5.(y-ye)\ d\ f\ (w1.i1+w3.i2)/dw1=w5.i1.(y-ye).f'(w1.i1+w3.i2)$
$=i1.w5.(y-ye).h1.(1-h1)$

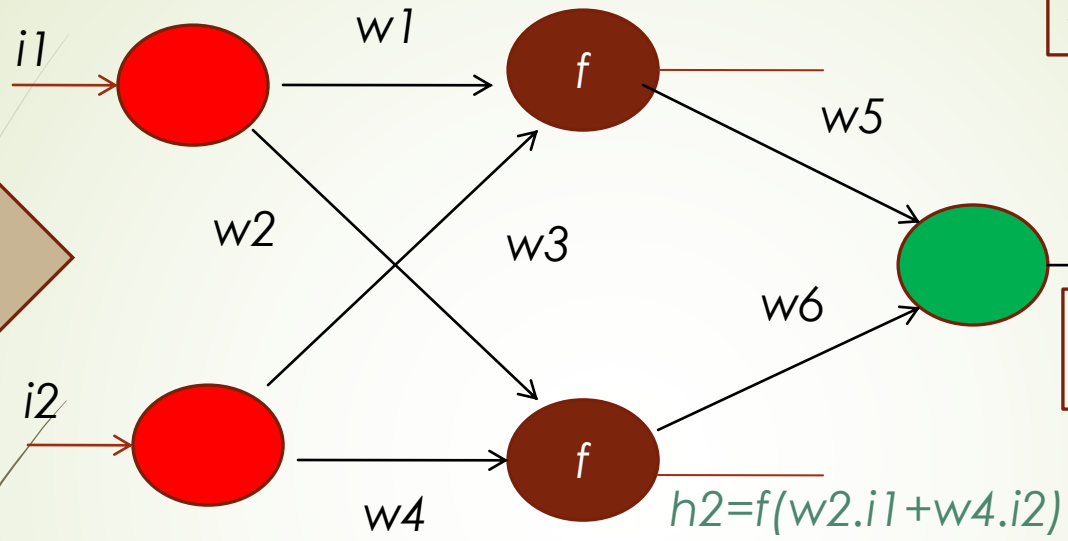
$w1=w1-\eta.i1.w5.(y-ye).h1.(1-h1)$

$w4=w4-\eta.i2.w6.(y-ye).h2.(1-h2)$

Mise à jour des poids d'entrée w1 et w4

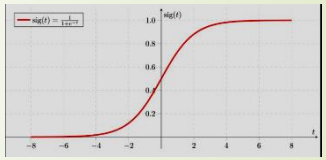
Intelligence artificielle

Comment apprendre exemple



Erreur=E=0.5 * (y-ye)^2
Soit sq : x^2/2 donc sq'=x

f=sig(x)=1/(1+exp(-x))
f'= sig(1-sig)



dE/dw2=d Sq(y-ye) /dw2 = Sq' (y-ye) .d(y-ye)/dw2 = (y-ye).d(y)/dw2
=w6.(y-ye) d f (w2.i1+w4.i2)/dw2=w6.i1.(y-ye).f'(w2.i1+w4.i2)
=i1.w6.(y-ye).h2.(1-h2)

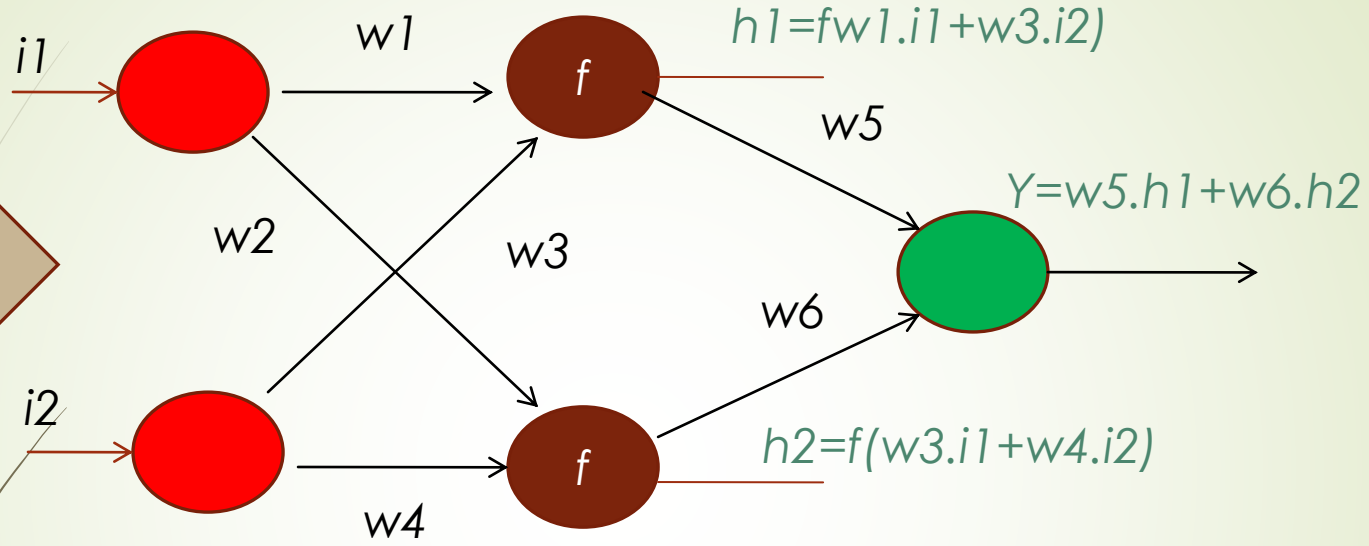
w2=w2-η.i1.w6.(y-ye).h2.(1-h2)

w3=w3-η.i2.w5.(y-ye).h1.(1-h1)

Mise à jour des poids d'entrée w2 et w3

Intelligence artificielle pour XOR

Programme

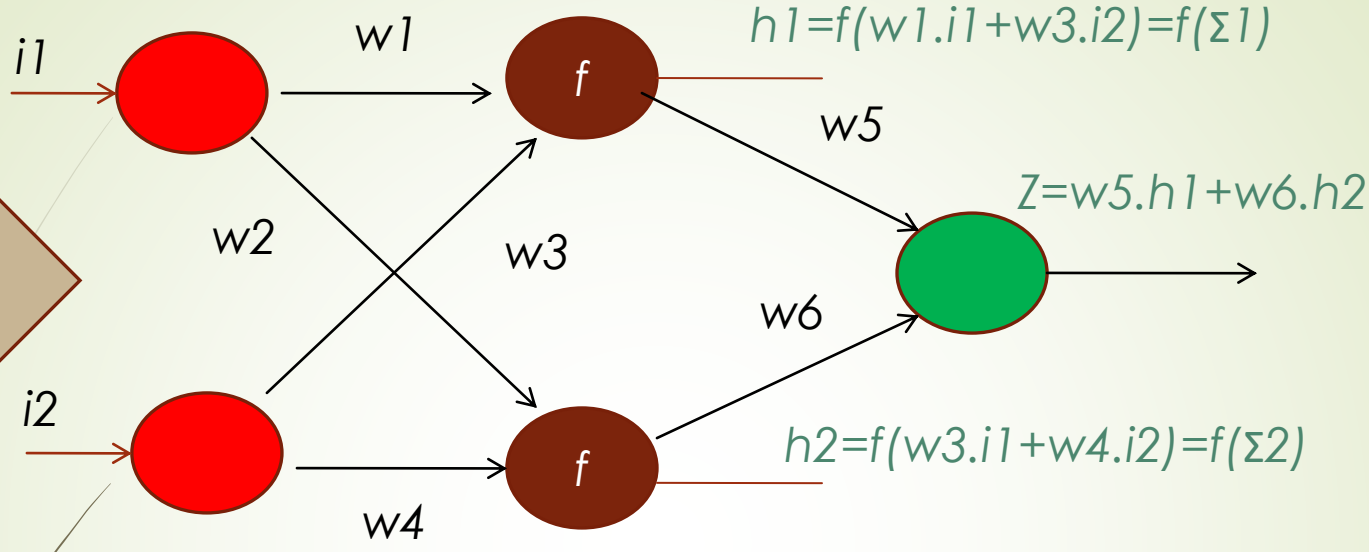


ENTREES		SORTIE
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

On va construire un réseau de neurones à une couche pour résoudre le problème du **OU EXCLUSIF (XOR)**. *Celui-ci prend 2 bits en entrée, A et B, puis renvoie 0 si A et B sont identiques et 1 s'ils sont inégaux.*

Intelligence artificielle

Programme




```
pip install numpy
import numpy as np
epochs = 20000                                  # Number of iterations
inputLayerSize, hiddenLayerSize, outputLayerSize = 2, 2, 1
L = .1                                          # learning rate
X = np.array([[0,0], [0,1], [1,0], [1,1]])
Y = np.array([ [0],   [1],   [1],   [0]])
def sigmoid(X): return 1/(1 + np.exp(-X))# activation function
def Dsigmoid_(X): return X * (1 - X)       # derivative of sigmoid
                                          # weights on layer inputs
Wh = np.random.uniform(size=(inputLayerSize, hiddenLayerSize))
Wz = np.random.uniform(size=(hiddenLayerSize,outputLayerSize))
```

Y=
```
[0]
[1]
[1]
[0]
```

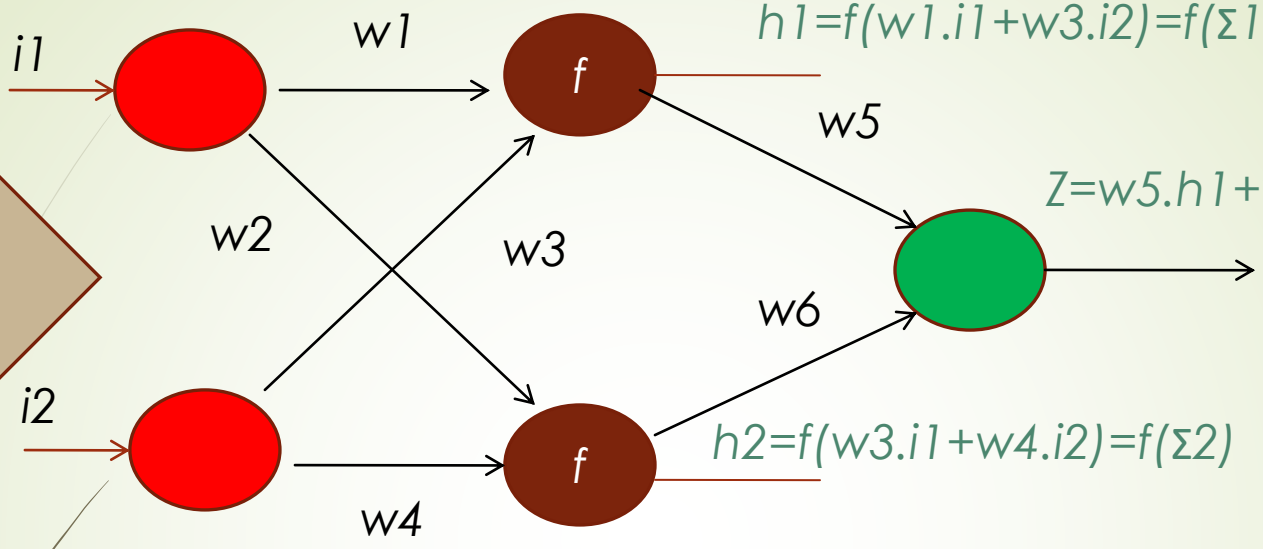
X=
```
   X1  X2
[0  0]
[0  1]
[1  0]
[1  1]
```

Wz=
```
[w5]
[w6]
```

Wh=
```
[w1 w2]
[w3  w4]
```

Intelligence artificielle

Programme



Y= [0] [1] [1] [0]

Wz= [w5] [w6]

X= X1 X2 [0 0] [0 1] [1 0] [1 1]

H= [h1(0) h2(0)] [h1(0) h2(1)] [h1(1) h2(0)] [h1(1) h2(1)]

Wh= [w1 w2] [w3 w4]

Z= [Z(0,0] [Z(0.1] [Z(1,0)] [Z(1.1]

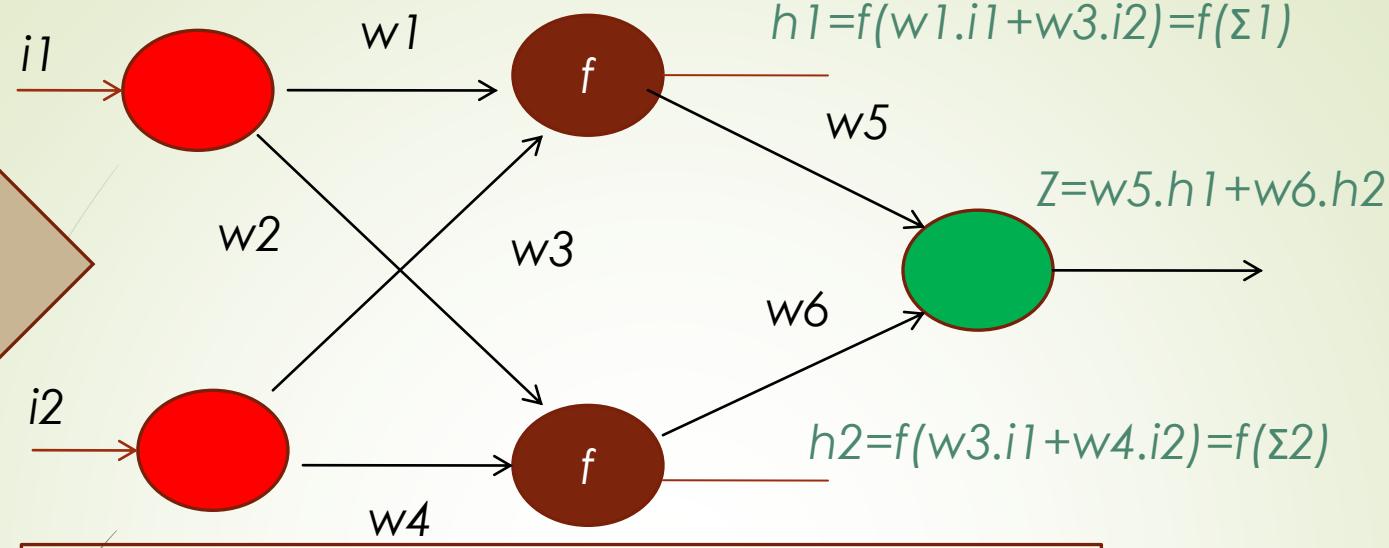
```
for i in range(epochs):

    H = sigmoid(np.dot(X, Wh))      # hidden layer results
    Z = np.dot(H,Wz)                # output layer, no activation
    delta = Y - Z                   # how much we missed (error)
    dZ = delta * L                  # delta Z
    Wz +=  H.T.dot(dZ)              # update output layer weights
    dH = dZ.dot(Wz.T) * Dsigmoid_(H)        # delta H
    Wh +=  X.T.dot(dH)              # update hidden layer weights

print(Z)
```

Wz+=mise à jour [w5] [w6]

Intelligence artificielle

23

Programme




```
import numpy as np
epochs = 20000                                          # Number of iterations
inputLayerSize, hiddenLayerSize, outputLayerSize = 2, 2, 1
L = .1                                                  # learning rate
X = np.array([[0,0], [0,1], [1,0], [1,1]])
Y = np.array([ [0],   [1],   [1],   [0]])
def sigmoid (x): return 1/(1 + np.exp(-x))              # activation function
def Dsigmoid_(x): return x * (1 - x)                    # derivative of sigmoid
                                                        # weights on layer inputs
Wh = np.random.uniform(size=(inputLayerSize, hiddenLayerSize))
Wz = np.random.uniform(size=(hiddenLayerSize,outputLayerSize))
for i in range(epochs):
    H = sigmoid(np.dot(X, Wh))                          # hidden layer results
   Z = np.dot(H,Wz)                                     # output layer, no activation
   delta = Y - Z                                        # how much we missed (error)
   dZ = delta * L                                       # delta Z
   Wz +=  H.T.dot(dZ)                                   # update output layer weights
   dH = dZ.dot(Wz.T) * Dsigmoid_(H)                     # delta H
   Wh +=  X.T.dot(dH)                                   # update hidden layer weights
print(Z)                                                # what have we learnt?
```

Résultats pour epochs = 20000

```
[[-0.07183499]
 [ 0.98890334]
 [ 0.9889025 ]
 [ 0.05725415]]
```

Résultats pour epochs = 50000

```
[[-0.05710839]
 [ 0.99390696]
 [ 0.99390848]
 [ 0.04035901]]
```