



# Leçon 6 : Réseaux et apprentissage II

Jonas Ranft  
Neurosciences computationnelles

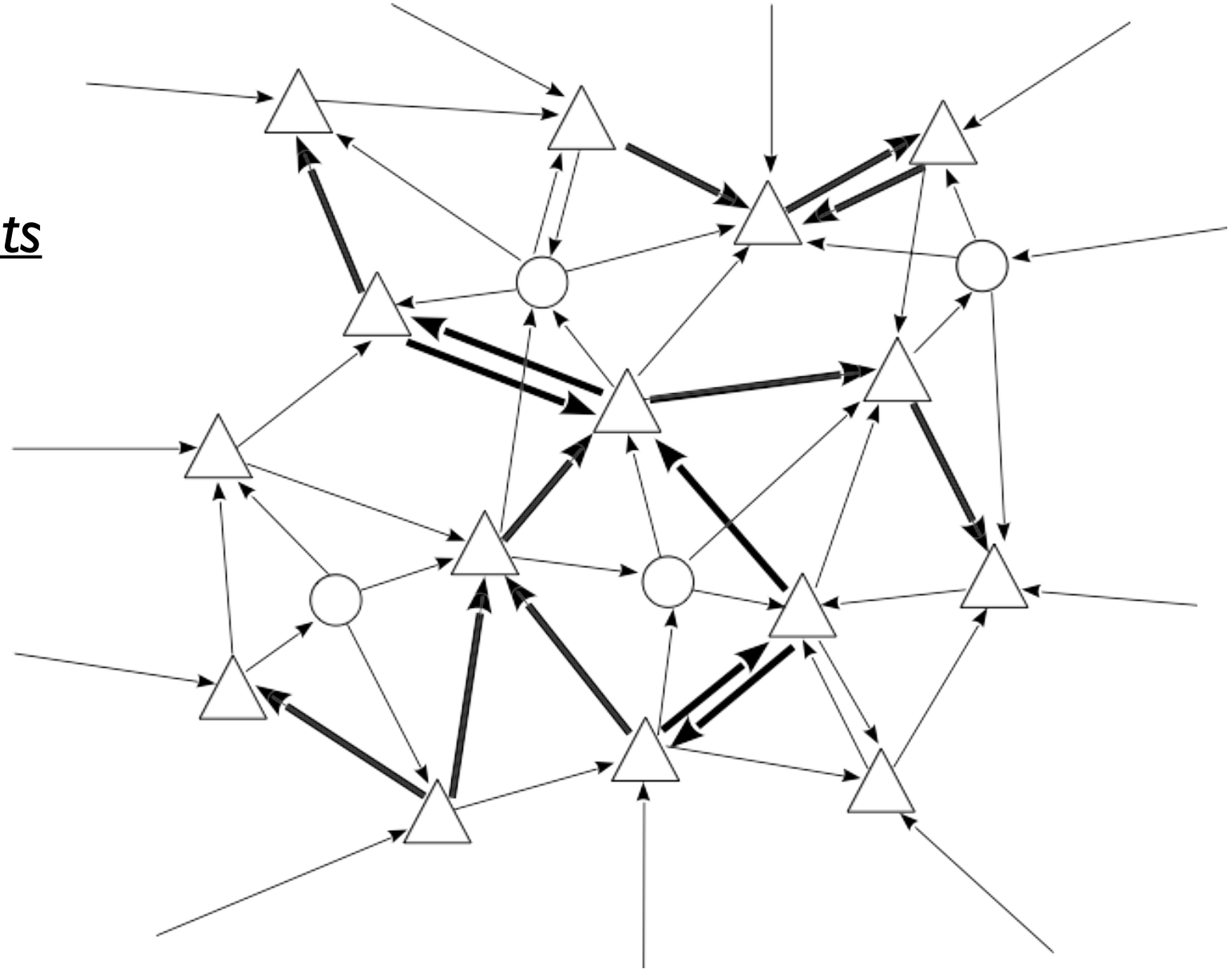
Nov–Dec 2020

# Mécanismes d'apprentissage

---

Les stimuli externes modifient l'activité neuronale dans le cerveau.

- Si un stimulus ne laisse pas de trace  
→ aucun souvenir !
- Les changements d'activité engendrent des changements de connectivité (plasticité structurelle/synaptique) !
- Un autre stimulus peut modifier l'activité dans un sous-ensemble différent.
- La connectivité synaptique résulte de la superposition des traces laissées par les entrées externes.



# Réseaux et apprentissage

---

Ce que nous savons déjà :

- Le plasticité structurel et/ou synaptique permet la modification des connexions synaptiques.
- Les connexions synaptiques peuvent servir comme *substrat de mémoire* :  
A partir d'un "indice", la dynamique du réseau (l'activité neuronale) évolue alors vers un *état attracteur* qui représente cette information.  
*Par ex. modélisé par le modèle de Hopfield.*

# Réseaux et apprentissage

---

## Ce que nous savons déjà :

- Le plasticité structurel et/ou synaptique permet la modification des connexions synaptiques.
- Les connexions synaptiques peuvent servir comme *substrat de mémoire* :  
A partir d'un "indice", la dynamique du réseau (l'activité neuronale) évolue alors vers un *état attracteur* qui représente cette information.  
*Par ex. modélisé par le modèle de Hopfield.*

## Ce que nous n'avons pas encore étudié :

- Comment apprendre les connexions synaptiques qui permettent à un réseau de remplir une fonction donnée ?

# Réseaux et apprentissage

---

## Ce que nous savons déjà :

- Le plasticité structurel et/ou synaptique permet la modification des connexions synaptiques.
- Les connexions synaptiques peuvent servir comme *substrat de mémoire* :  
A partir d'un "indice", la dynamique du réseau (l'activité neuronale) évolue alors vers un *état attracteur* qui représente cette information.  
*Par ex. modélisé par le modèle de Hopfield.*

## Ce que nous n'avons pas encore étudié :

- **Comment apprendre les connexions synaptiques qui permettent à un réseau de remplir une fonction donnée ?**

## Ce qu'il faut distinguer :

- L'apprentissage *supervisé* de l'apprentissage *non-supervisé*, ainsi que l'apprentissage *par renforcement*.
- Les *modèles d'apprentissage biologiques* d'une part, et les *algorithmes* qui en *principe* permettent le bon apprentissage d'autre part.

# Réseaux et apprentissage

---

Ce que nous savons déjà :

- Le plasticité structurel et/ou synaptique permet la modification des connexions synaptiques.
- Les connexions synaptiques peuvent servir comme *substrat de mémoire* :  
A partir d'un "indice", la dynamique du réseau (l'activité neuronale) évolue alors vers un *état attracteur* qui représente cette information.  
*Par ex. modélisé par le modèle de Hopfield.*

Ce que nous n'avons pas encore étudié :

- **Comment apprendre les connexions synaptiques qui permettent à un réseau de remplir une fonction donnée ?**

Ce qu'il faut distinguer :

aujourd'hui

- L'apprentissage **supervisé** de l'apprentissage *non-supervisé*, ainsi que l'apprentissage *par renforcement*.
- Les *modèles d'apprentissage biologiques* d'une part, et les **algorithmes** qui en *principe* permettent le bon apprentissage d'autre part.

# Les différents types d'apprentissage

---

- **Apprentissage *non-supervisé*** : Les stimuli externes présentés au réseau entraînent, ensemble avec l'activité du réseau, des changements synaptiques sans d'autre intervention ou signal guidant la plasticité.
  - Exemple : construction d'une représentation neuronale d'un stimulus par apprentissage Hebbien

# Les différents types d'apprentissage

---

- **Apprentissage *non-supervisé*** : Les stimuli externes présentés au réseau entraînent, ensemble avec l'activité du réseau, des changements synaptiques sans d'autre intervention ou signal guidant la plasticité.
  - Exemple : construction d'une représentation neuronale d'un stimulus par apprentissage Hebbien
- **Apprentissage *supervisé*** : Les connexions synaptiques sont modifiées en fonction du résultat d'une classification, régression etc. d'un, en connaissance du résultat souhaité.
  - Exemples : "deep learning" (réseau multi-couche pour classifier par ex. des images, où l'image et l'objet à reconnaître sont connus lors de l'entraînement), *perceptron*



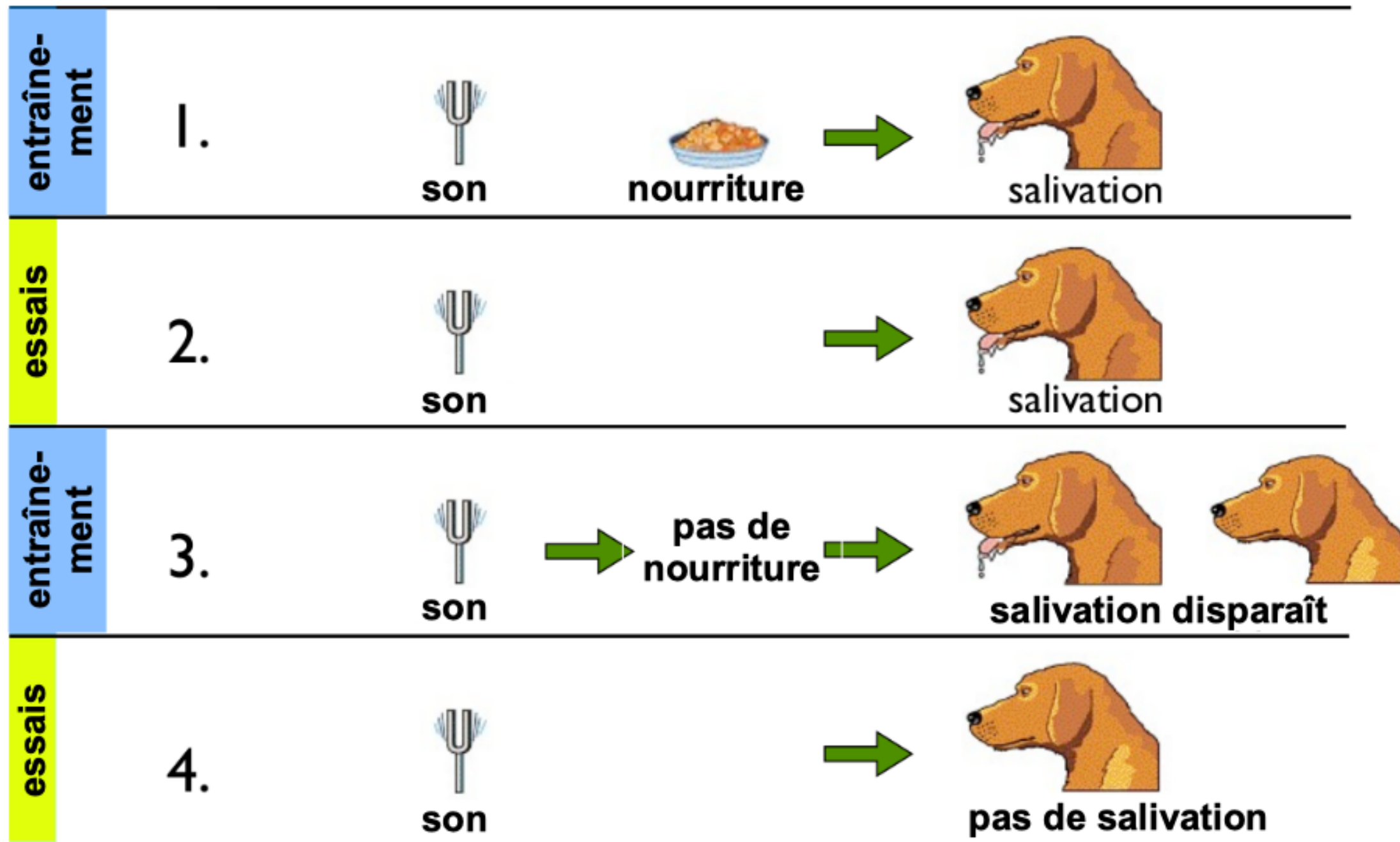
# Les différents types d'apprentissage

---

- **Apprentissage *non-supervisé*** : Les stimuli externes présentés au réseau entraînent, ensemble avec l'activité du réseau, des changements synaptiques sans d'autre intervention ou signal guidant la plasticité.
  - Exemple : construction d'une représentation neuronale d'un stimulus par apprentissage Hebbien
- **Apprentissage *supervisé*** : Les connexions synaptiques sont modifiées en fonction du résultat d'une classification, régression etc. d'un, en connaissance du résultat souhaité.
  - Exemples : "deep learning" (réseau multi-couche pour classifier par ex. des images, où l'image et l'objet à reconnaître sont connus lors de l'entraînement), *perceptron*
- **Apprentissage *par renforcement*** : Apprendre la bonne réponse à un stimulus sur la base des récompenses ou punitions y associées.
  - Exemple : conditionnement classique

# Apprentissage par renforcement

Exemple : l'apprentissage (conditionnement) du “réflexe Pavlovien”



# Apprentissage supervisé

---

- **But** : Apprendre une fonction  $F$  sur les entrées à partir d'un ensemble de données, par ex. pour classifier des images.
- **Intérêt** : Utiliser la fonction apprise sur de nouvelles entrées (jeux de données) qui n'ont pas été utilisées lors de l'entraînement.



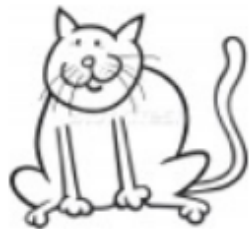
→ chat



→ chien



→ chien



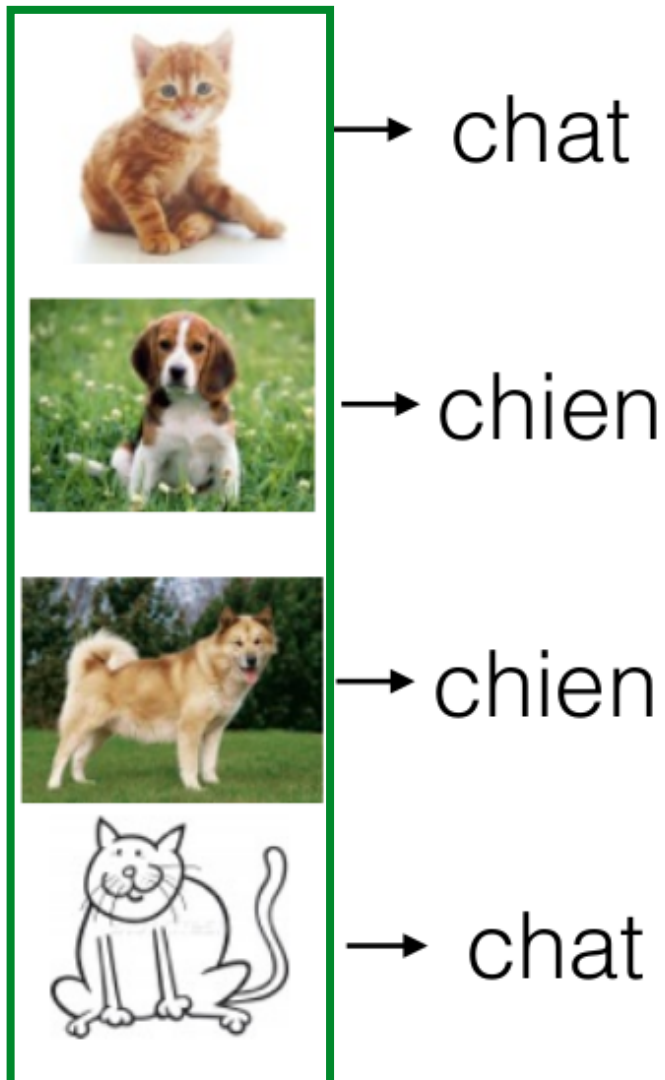
→ chat

# Apprentissage supervisé

---

- **But** : Apprendre une fonction  $F$  sur les entrées à partir d'un ensemble de données, par ex. pour classifier des images.
- **Intérêt** : Utiliser la fonction apprise sur de nouvelles entrées (jeux de données) qui n'ont pas été utilisées lors de l'entraînement.

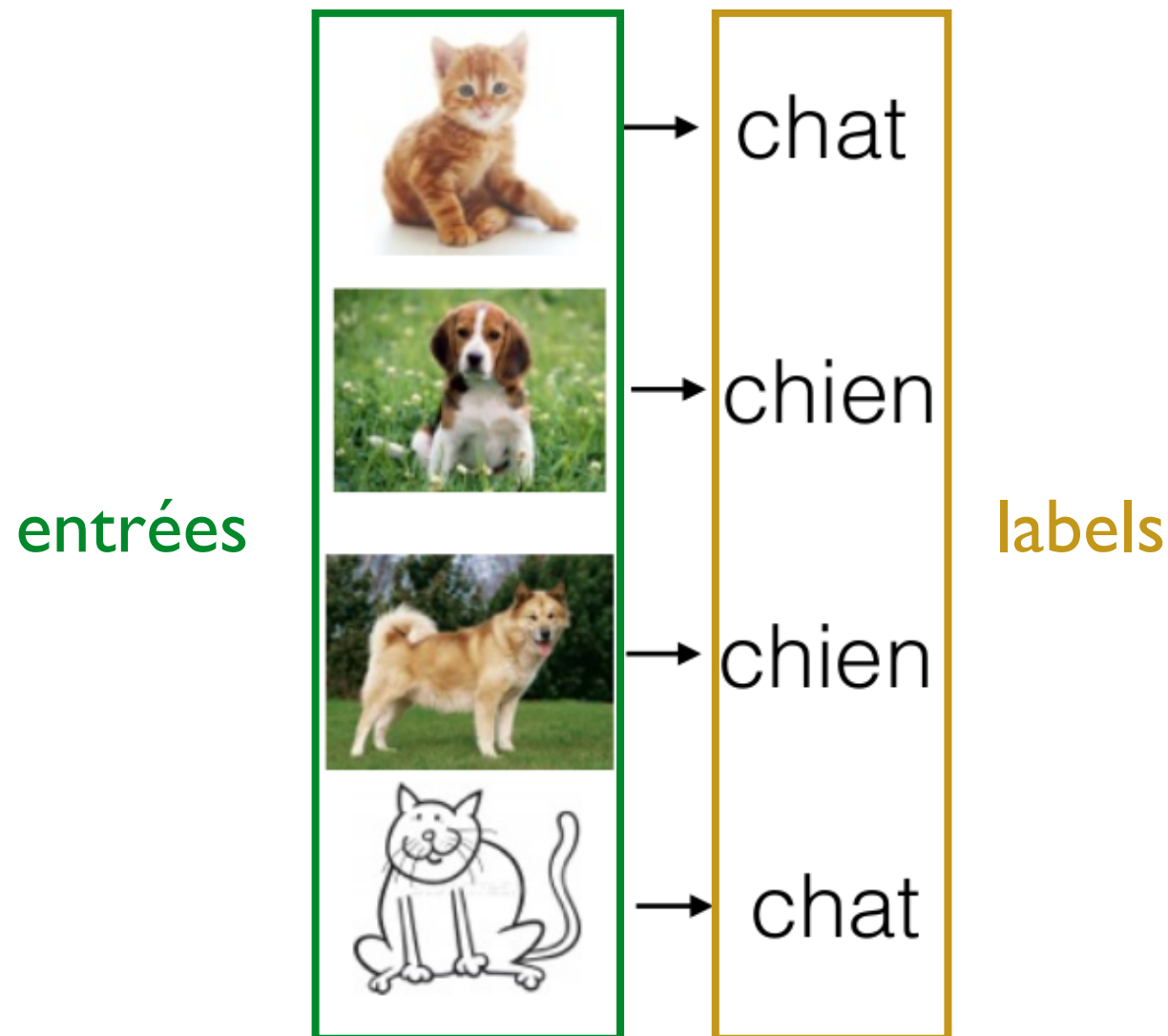
entrées



# Apprentissage supervisé

---

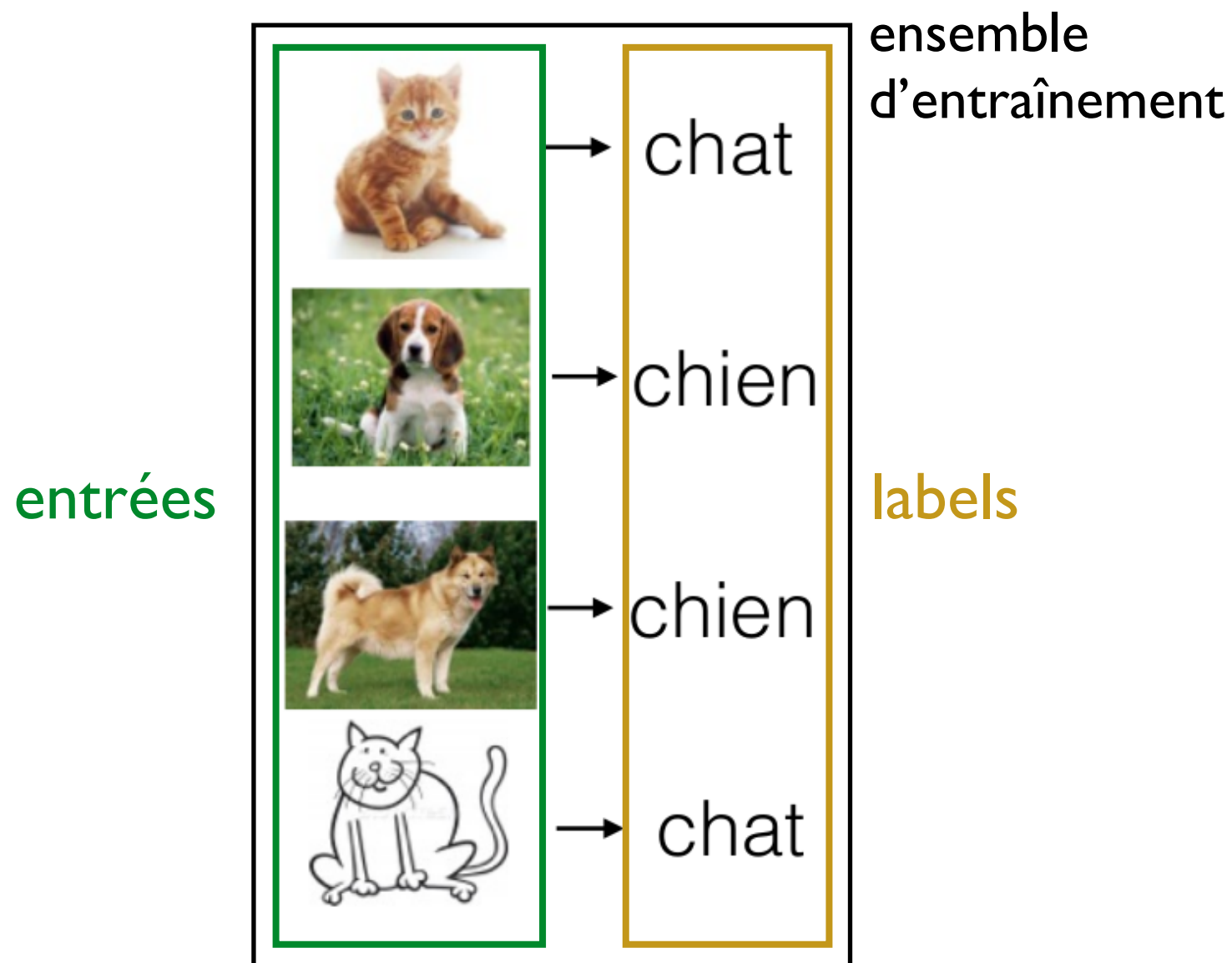
- **But** : Apprendre une fonction  $F$  sur les entrées à partir d'un ensemble de données, par ex. pour classifier des images.
- **Intérêt** : Utiliser la fonction apprise sur de nouvelles entrées (jeux de données) qui n'ont pas été utilisées lors de l'entraînement.



# Apprentissage supervisé

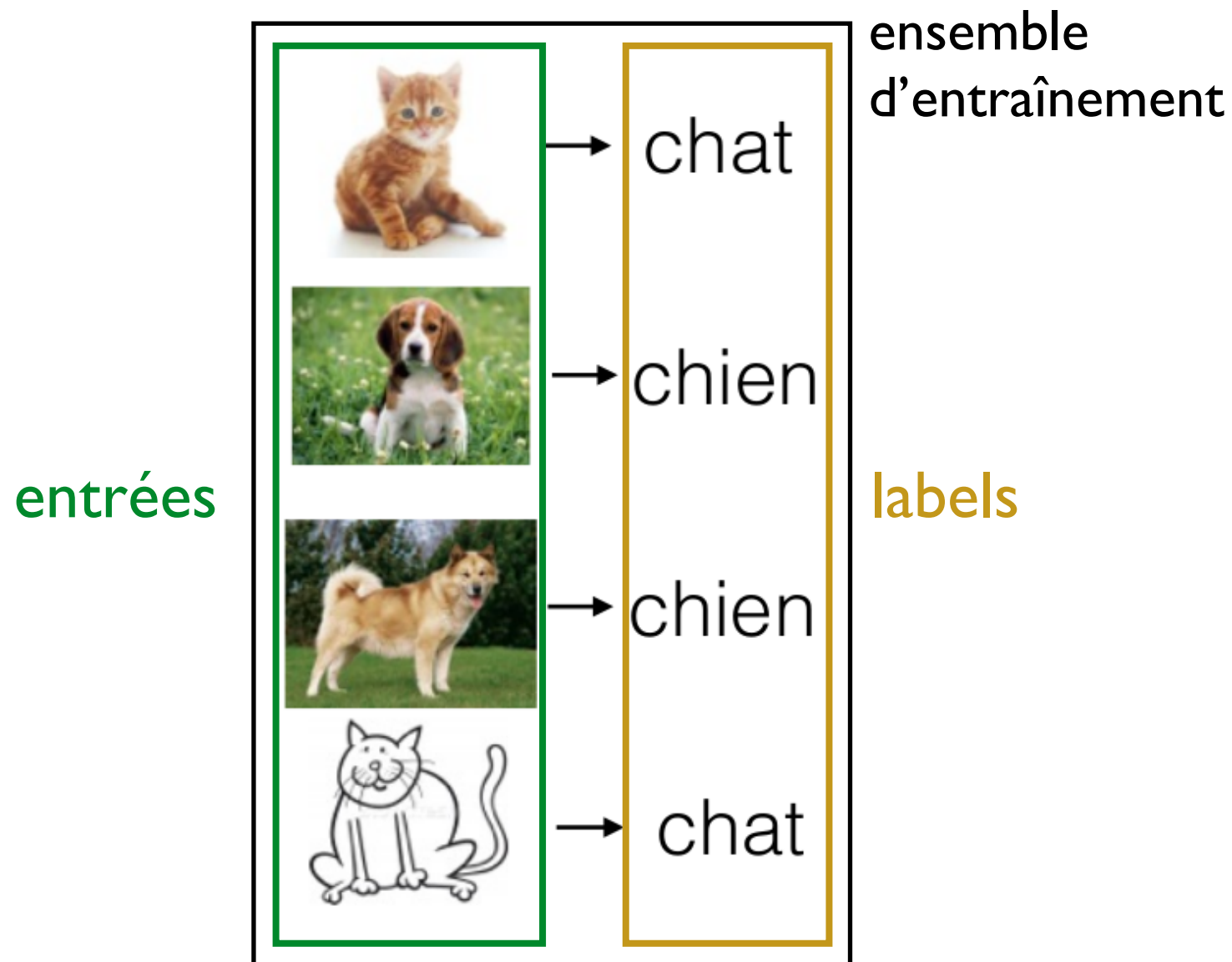
---

- **But** : Apprendre une fonction  $F$  sur les entrées à partir d'un ensemble de données, par ex. pour classer des images.
- **Intérêt** : Utiliser la fonction apprise sur de nouvelles entrées (jeux de données) qui n'ont pas été utilisées lors de l'entraînement.



# Apprentissage supervisé

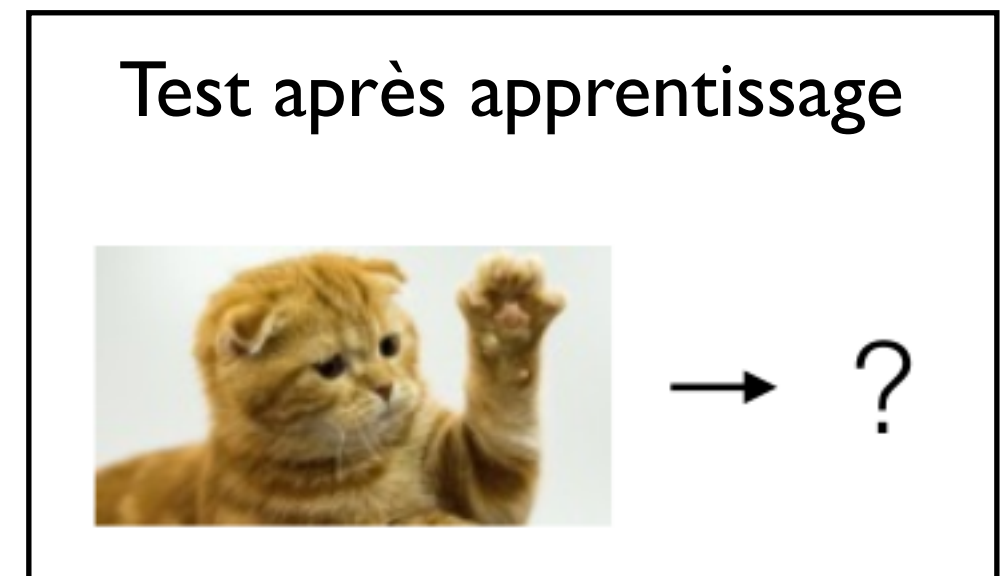
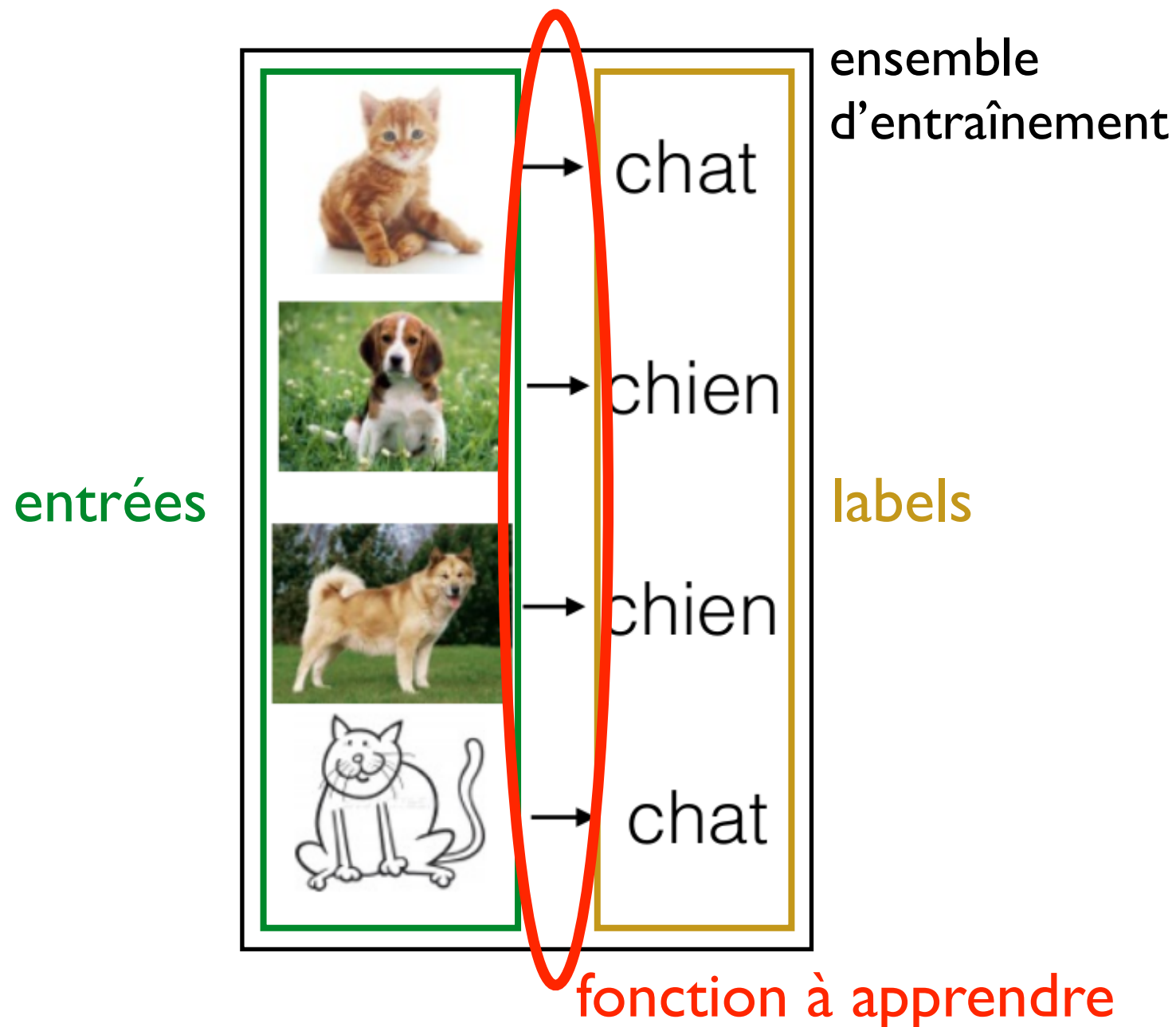
- **But** : Apprendre une fonction  $F$  sur les entrées à partir d'un ensemble de données, par ex. pour classer des images.
- **Intérêt** : Utiliser la fonction apprise sur de nouvelles entrées (jeux de données) qui n'ont pas été utilisées lors de l'entraînement.





# Apprentissage supervisé

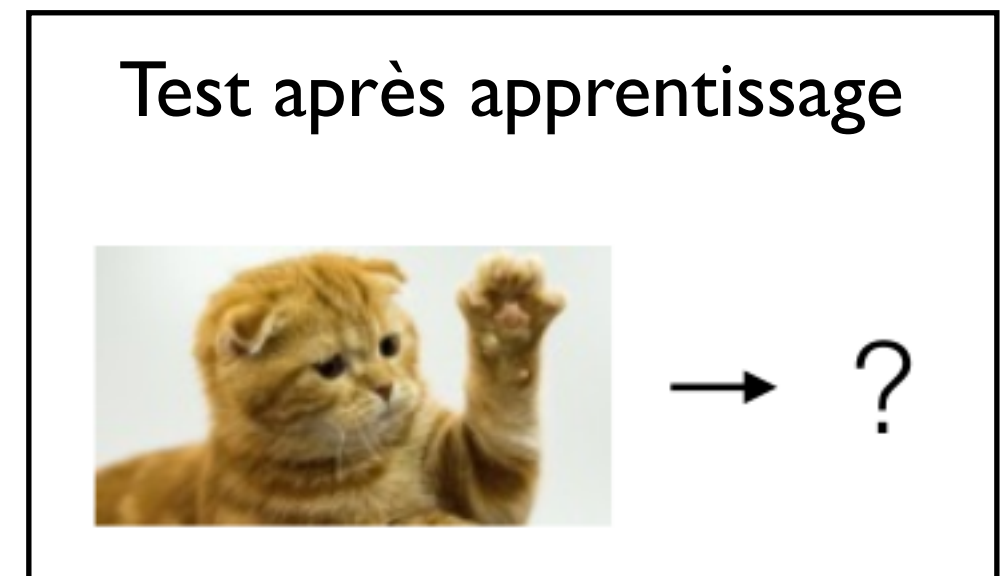
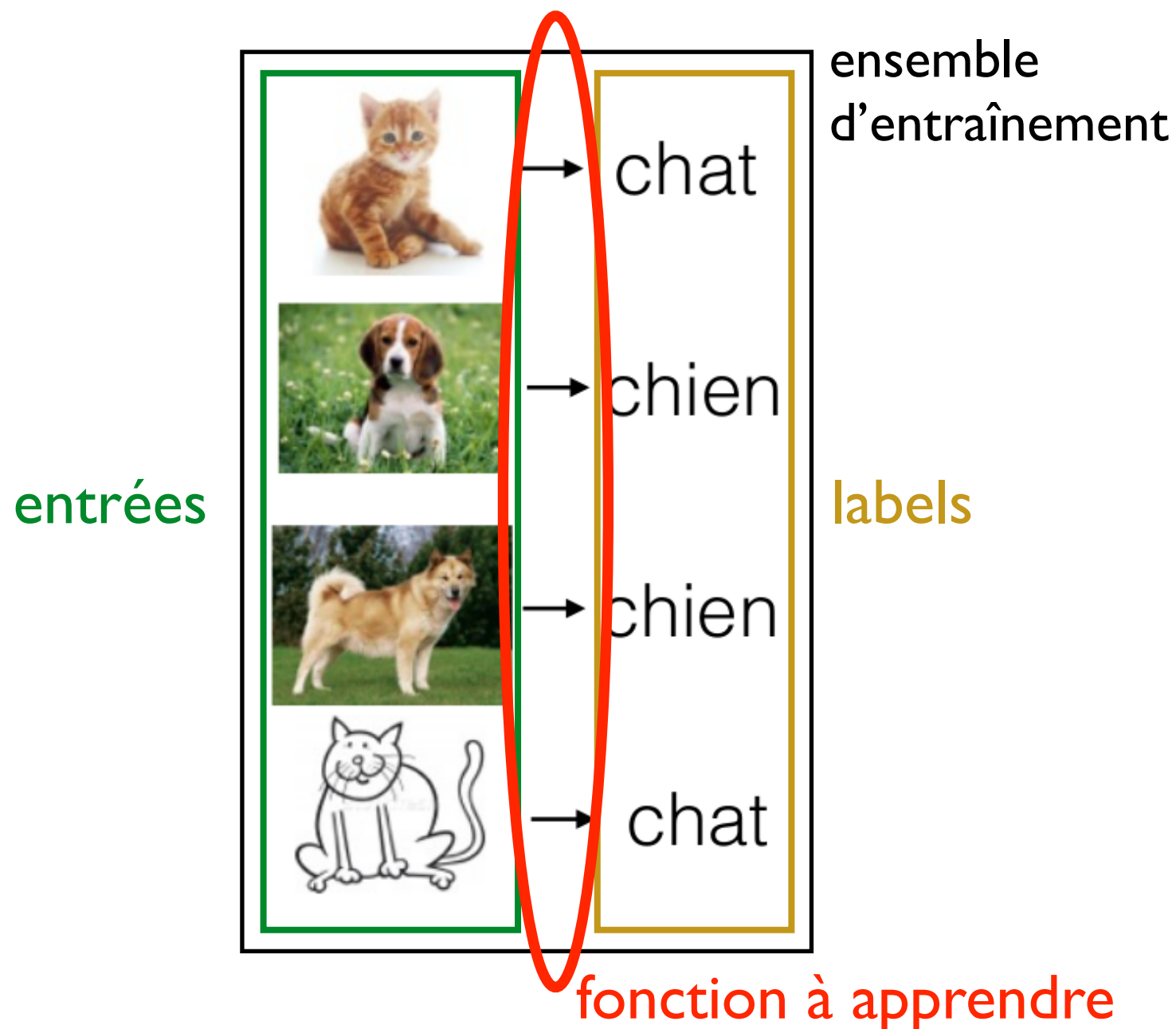
- **But** : Apprendre une fonction  $F$  sur les entrées à partir d'un ensemble de données, par ex. pour classer des images.
- **Intérêt** : Utiliser la fonction apprise sur de nouvelles entrées (jeux de données) qui n'ont pas été utilisées lors de l'entraînement.





# Apprentissage supervisé

- **But** : Apprendre une fonction  $F$  sur les entrées à partir d'un ensemble de données, par ex. pour classifier des images.
- **Intérêt** : Utiliser la fonction apprise sur de nouvelles entrées (jeux de données) qui n'ont pas été utilisées lors de l'entraînement.



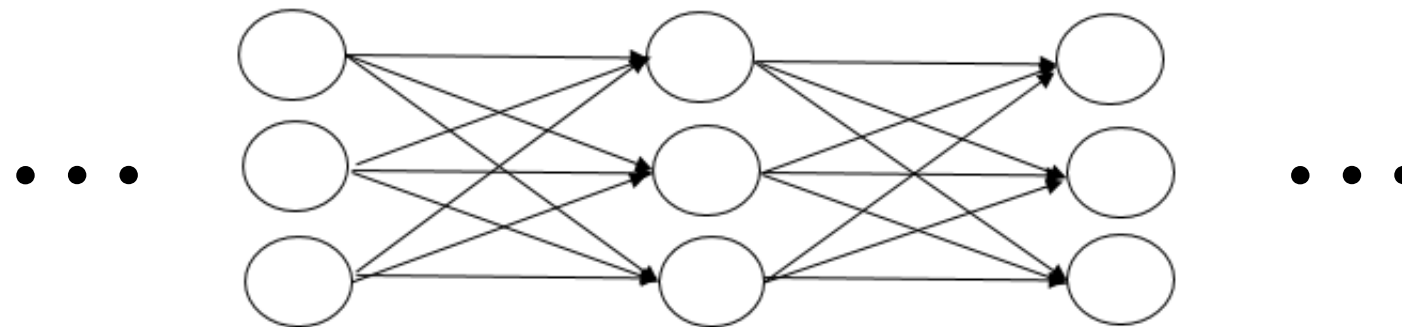
**Difficulté** : L'apprentissage peut nécessiter un ensemble de données d'entraînement très conséquent.

# Les réseaux “feed-forward”

---

## Structure des réseaux sans connexions récurrentes

- Lorsque les neurones sont organisés en couches sans connexions récurrentes, on parle d'un réseau “feed-forward”, où les sorties des neurones d'une couche sont les entrées de la couche suivante :



- Un réseau “feed-forward” a une couche comprend une *couche d'entrée* ( $N_{in}$  neurones) et une *couche de sortie* ( $N_{in}$  neurones).
- L'apprentissage dit “profond” consiste à apprendre les poids synaptiques des réseaux “feed-forward” à plusieurs couches.
- Par ex. dans le traitement initial des entrées visuelles par la rétine et le cortex visuelle on peut également parler de réseaux “feed-forward” tant les connexions récurrentes ne rentrent pas en jeu.

# Apprentissage supervisé : Le perceptron

---

## The New York Times

### ***NEW NAVY DEVICE LEARNS BY DOING***

*July 8, 1958*

“The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence... Dr. Frank Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers”



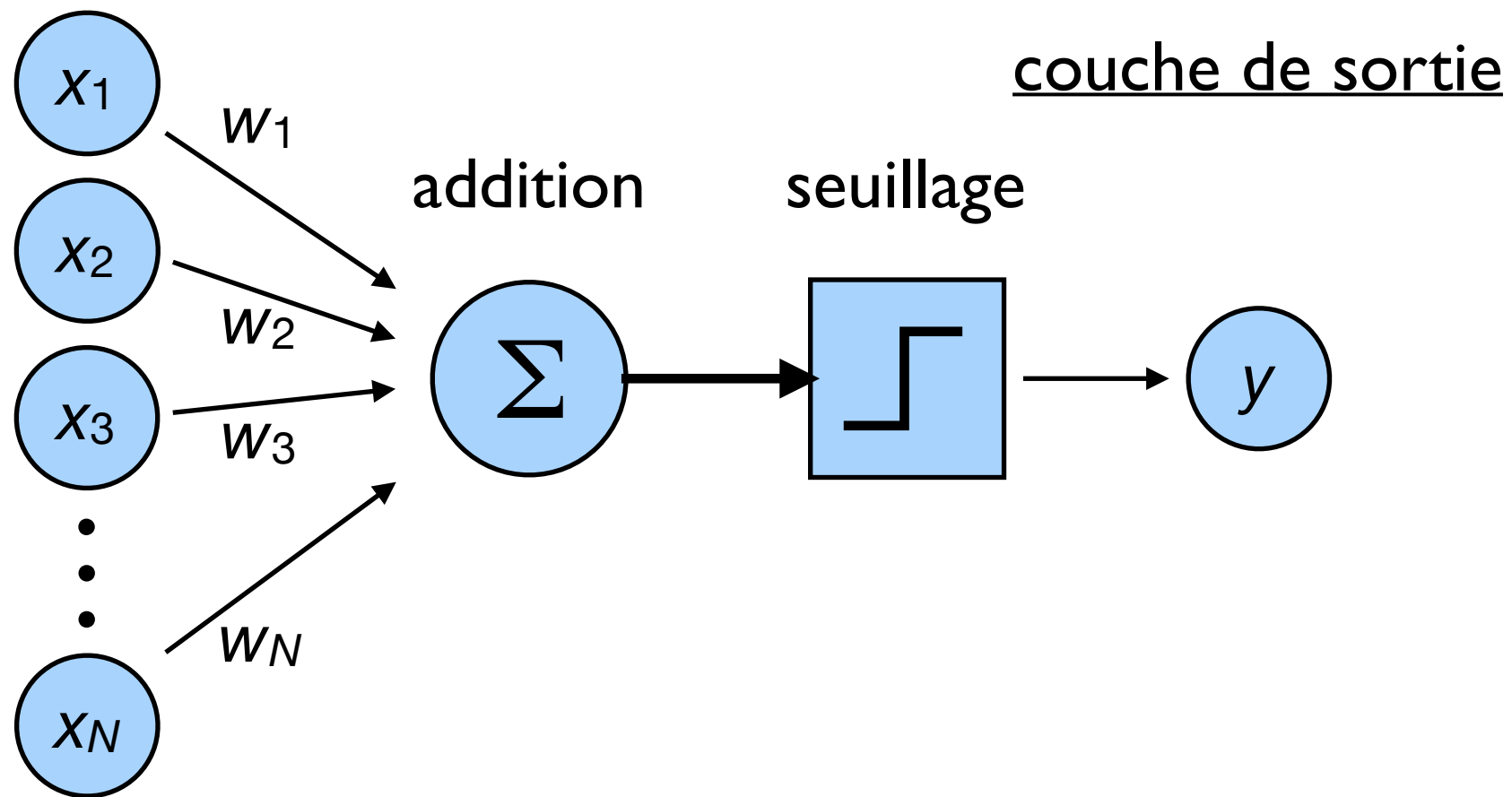
Frank Rosenblatt, 1958

# Apprentissage supervisé : Le perceptron

---

Une machine qui permet de faire des classifications binaires

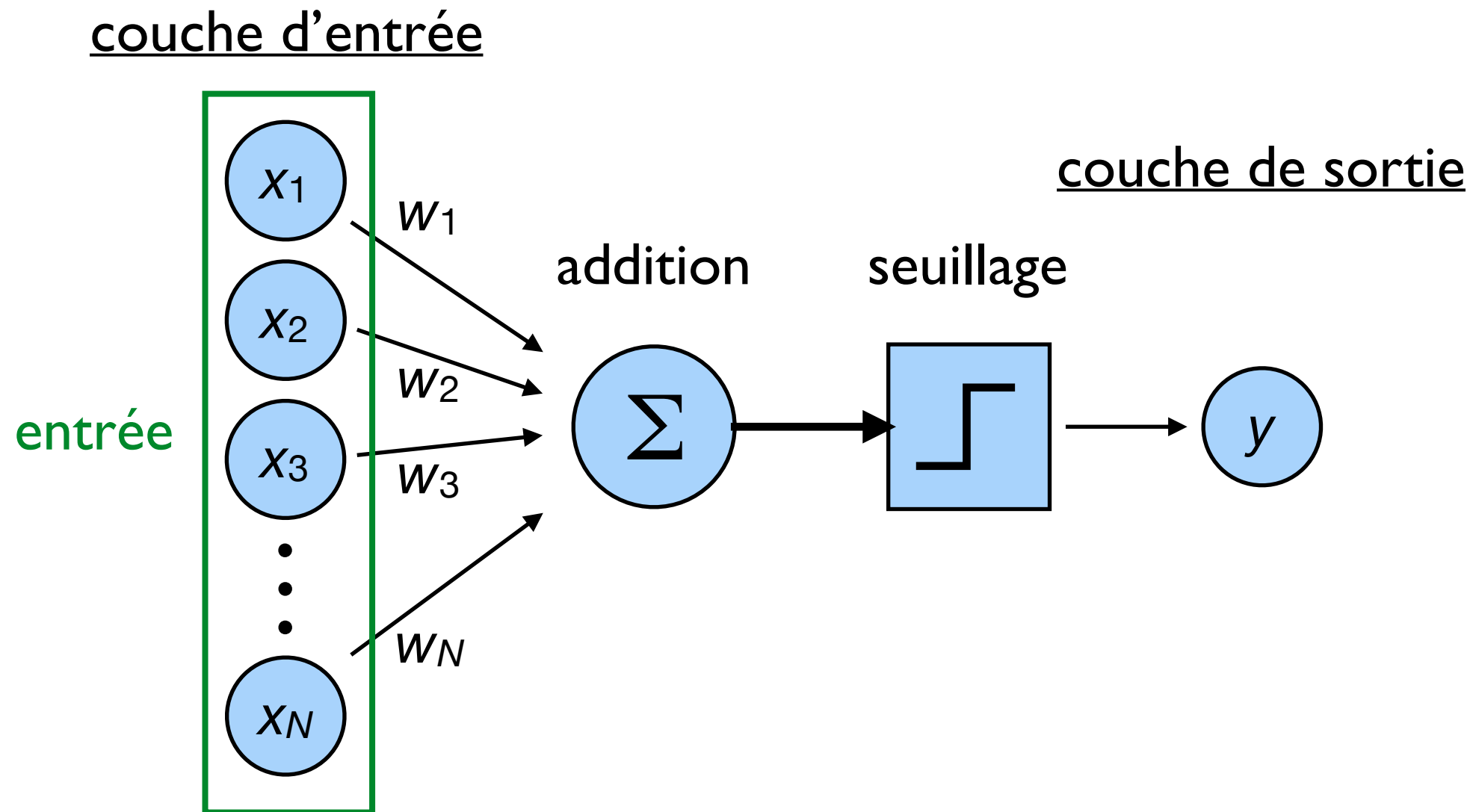
couche d'entrée



# Apprentissage supervisé : Le perceptron

---

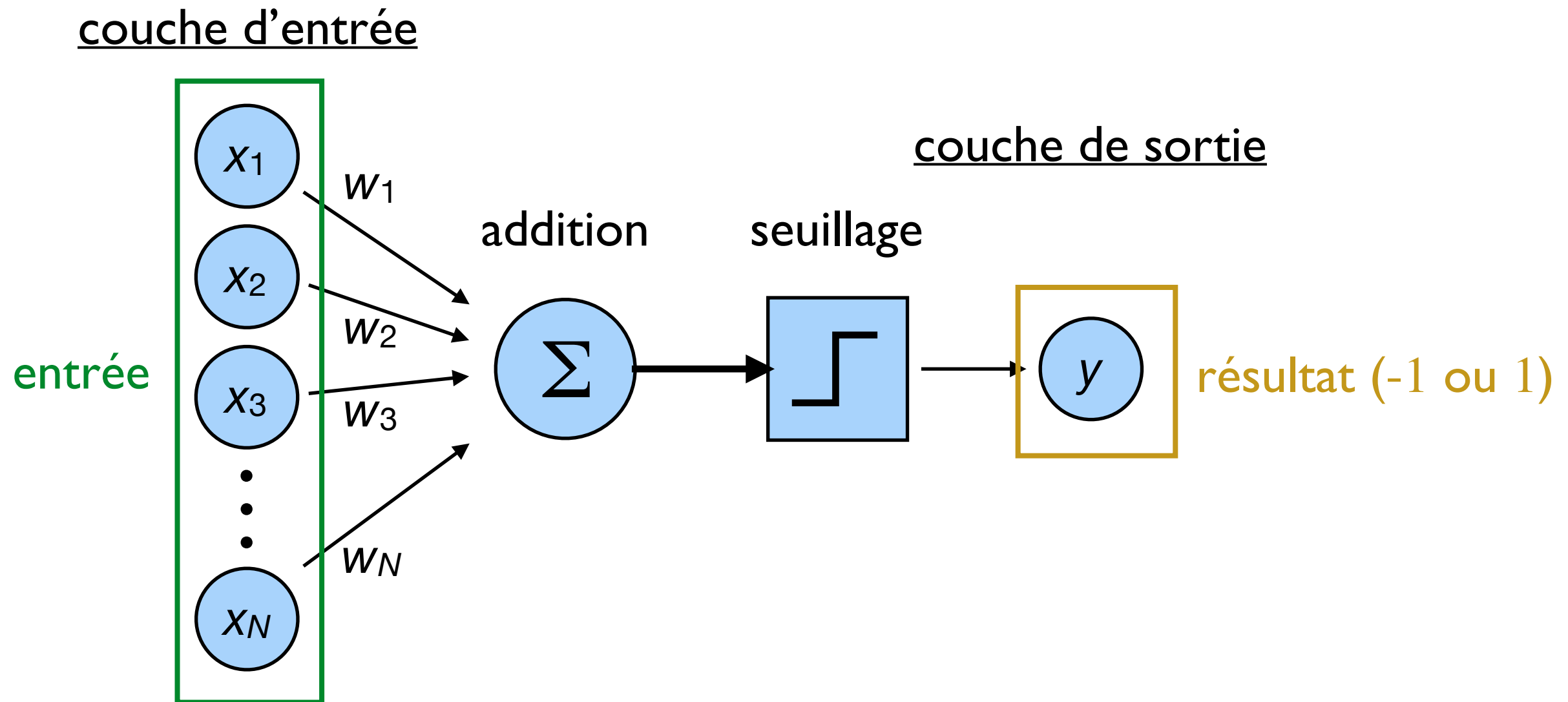
Une machine qui permet de faire des classifications binaires



# Apprentissage supervisé : Le perceptron

---

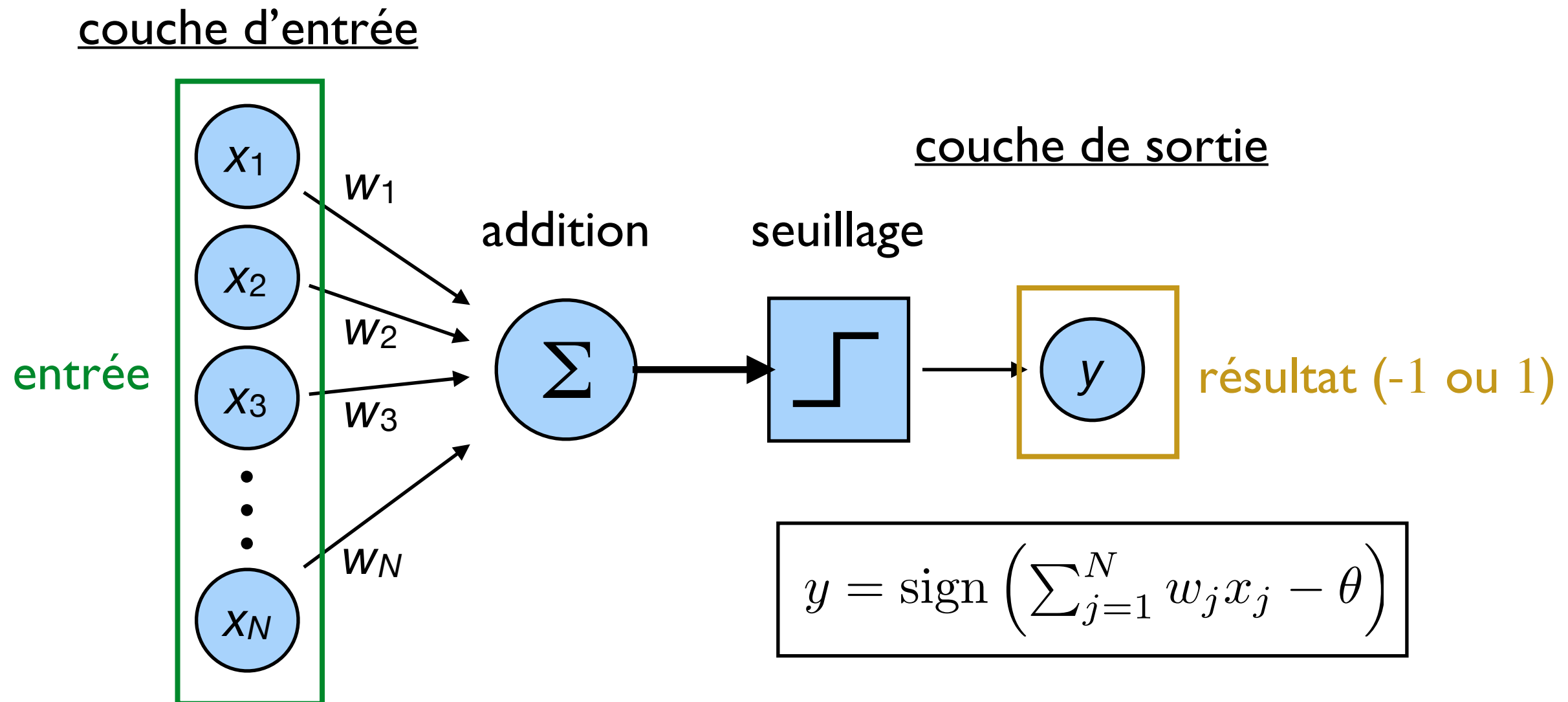
Une machine qui permet de faire des classifications binaires



# Apprentissage supervisé : Le perceptron

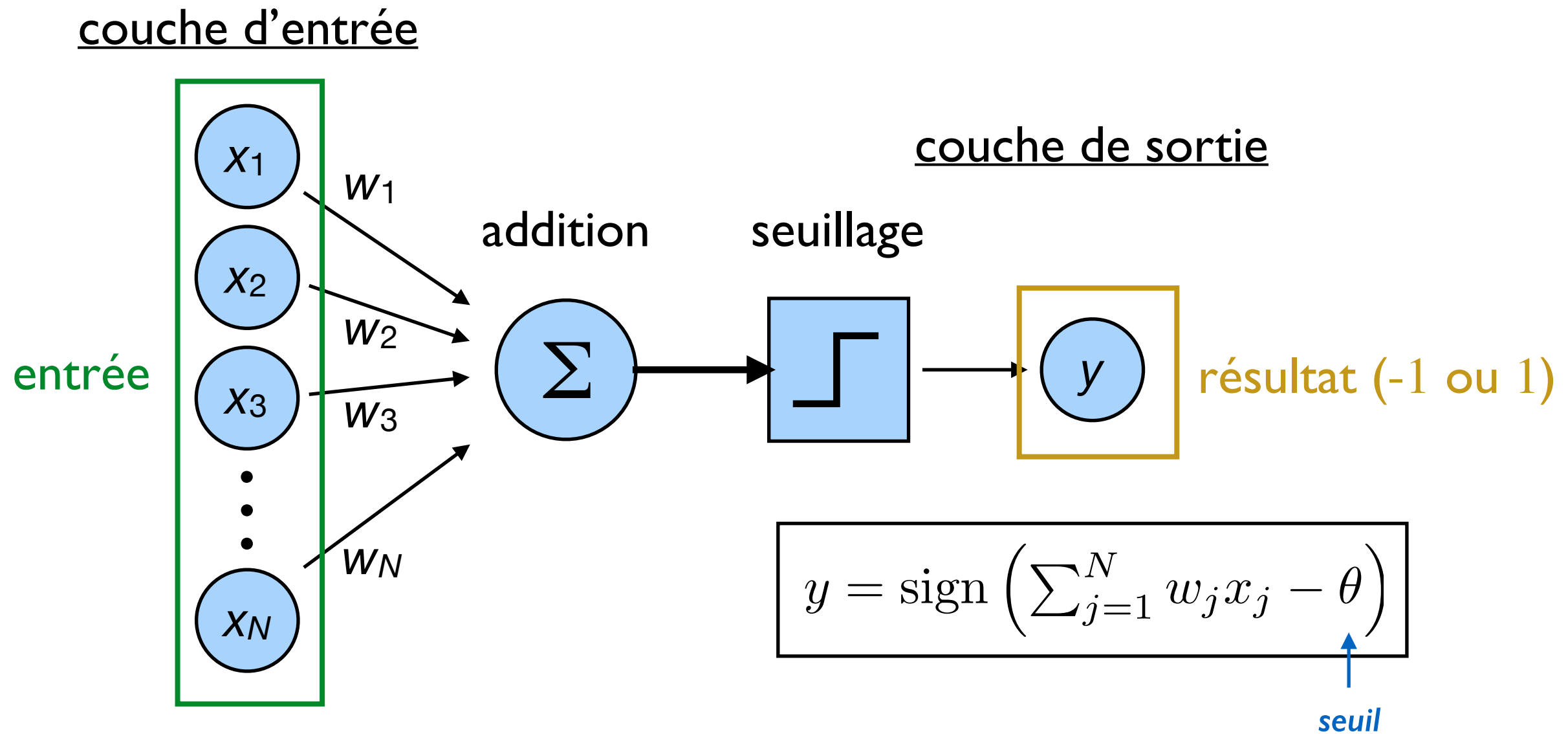
---

Une machine qui permet de faire des classifications binaires



# Apprentissage supervisé : Le perceptron

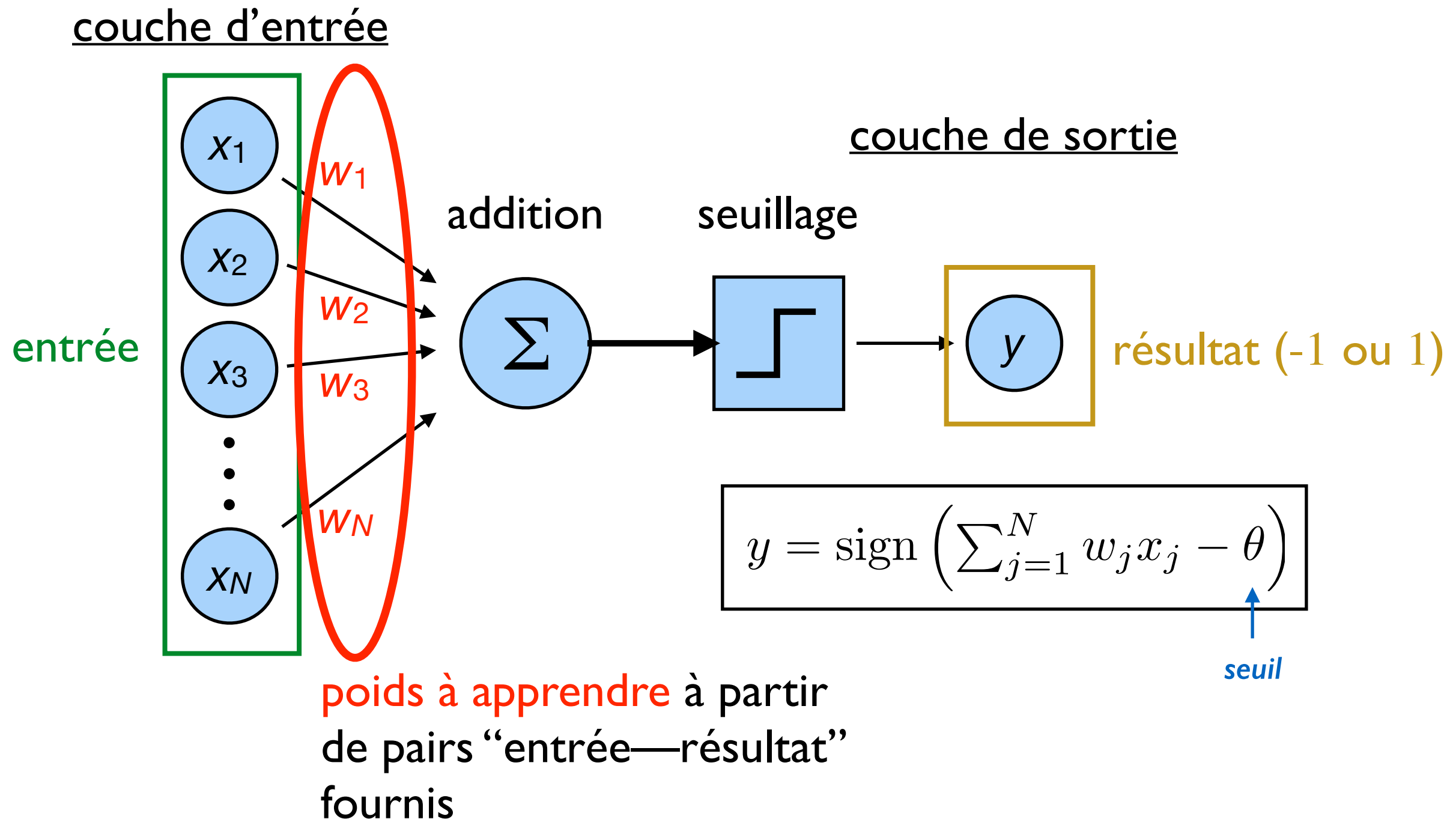
Une machine qui permet de faire des classifications binaires





# Apprentissage supervisé : Le perceptron

Une machine qui permet de faire des classifications binaires



# Apprentissage supervisé : Le perceptron

---

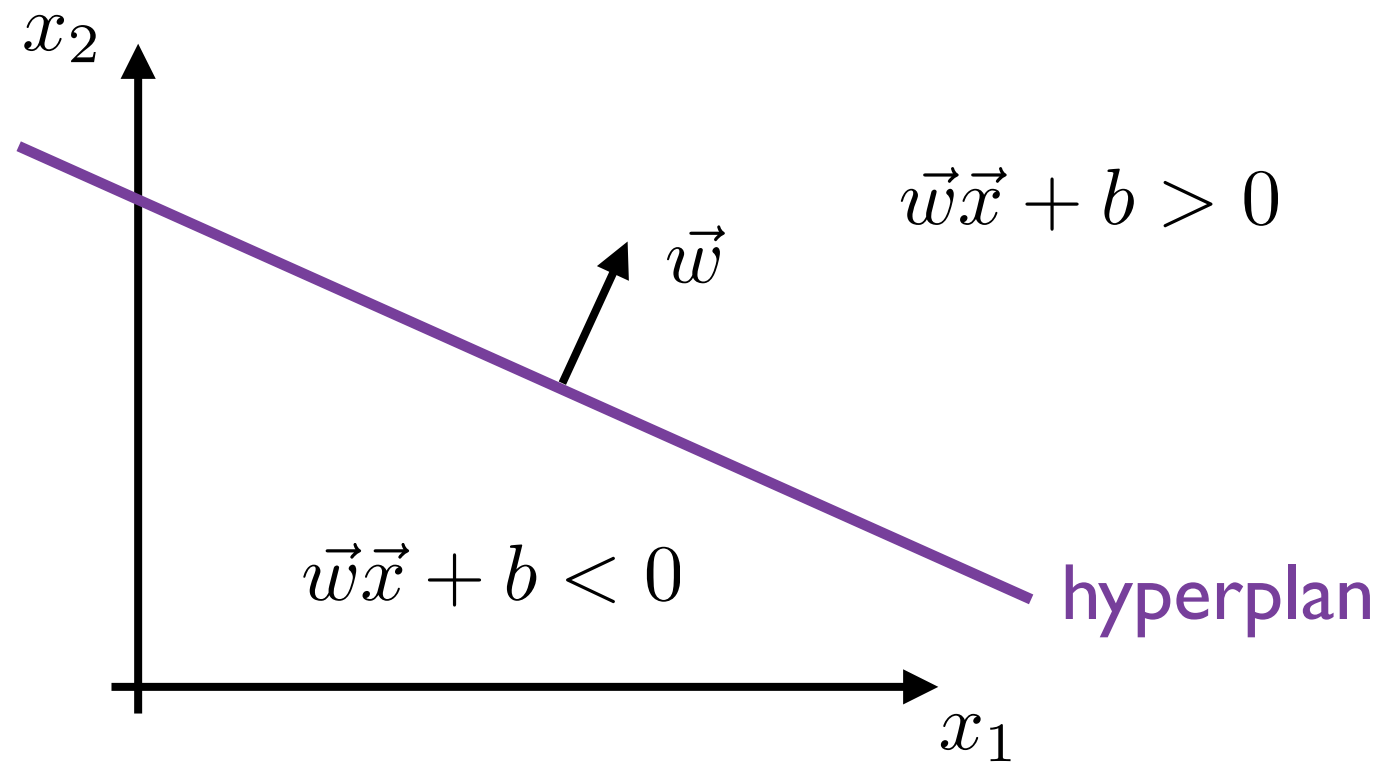
Une machine qui permet de faire des classifications binaires

- Les perceptrons à une seule couche sont uniquement capables d'apprendre à distinguer des motifs *linéairement séparables*.
- Les réseaux “feed-forward” à deux ou plusieurs couches (aussi appelés *perceptrons multi-couches* si les neurones de chaque couche sont décrits par l'équation des perceptrons) sont beaucoup plus performants → cf. apprentissage profond...
- Outil pour étudier les questions de capacité de stockage.

# Le perceptron, interprétation géométrique

---

- Equation du perceptron :  $y = \text{sign} \left( \sum_{j=1}^N w_j x_j - \theta \right)$
- Equation d'un hyperplan :  $\vec{w}\vec{x} + b = 0$  ( $b \leftrightarrow -\theta$ )

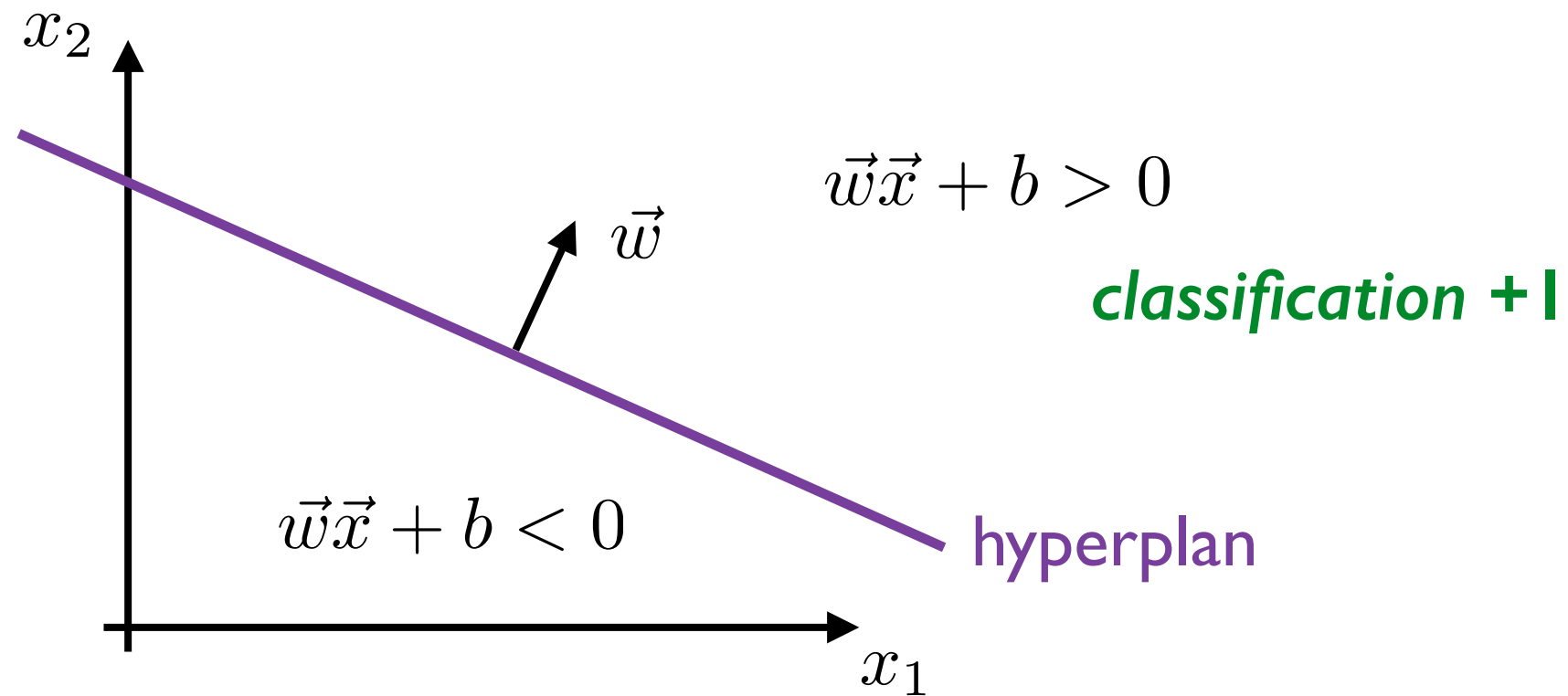


- Chaque entrée correspond à un point dans un espace de  $N$  dimensions.
- Deux ensembles de points sont *linéairement séparables* s'il existe un hyperplan (de dimension  $N-1$ ) qui sépare les deux ensembles.

# Le perceptron, interprétation géométrique

---

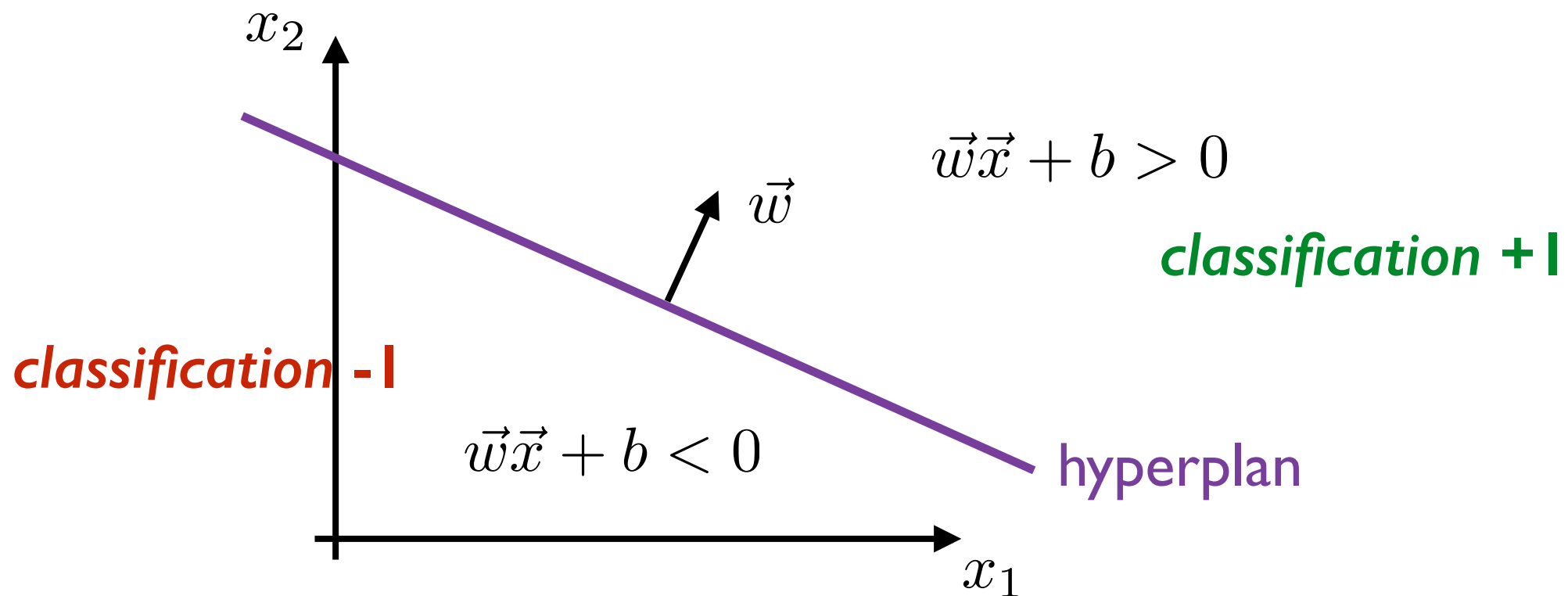
- Equation du perceptron :  $y = \text{sign} \left( \sum_{j=1}^N w_j x_j - \theta \right)$
- Equation d'un hyperplan :  $\vec{w}\vec{x} + b = 0 \quad (b \leftrightarrow -\theta)$



- Chaque entrée correspond à un point dans un espace de  $N$  dimensions.
- Deux ensembles de points sont *linéairement séparables* s'il existe un hyperplan (de dimension  $N-1$ ) qui sépare les deux ensembles.

# Le perceptron, interprétation géométrique

- Equation du perceptron :  $y = \text{sign} \left( \sum_{j=1}^N w_j x_j - \theta \right)$
- Equation d'un hyperplan :  $\vec{w}\vec{x} + b = 0$  ( $b \leftrightarrow -\theta$ )

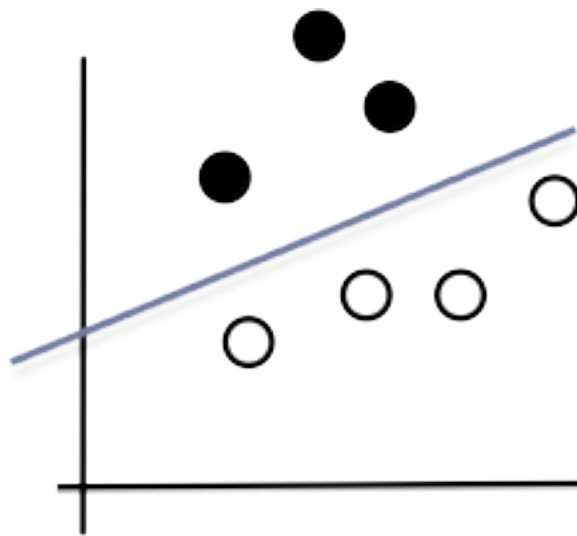


- Chaque entrée correspond à un point dans un espace de  $N$  dimensions.
- Deux ensembles de points sont *linéairement séparables* s'il existe un hyperplan (de dimension  $N-1$ ) qui sépare les deux ensembles.

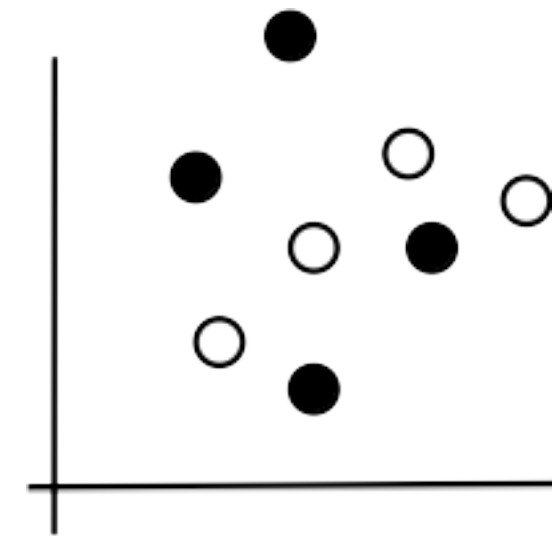
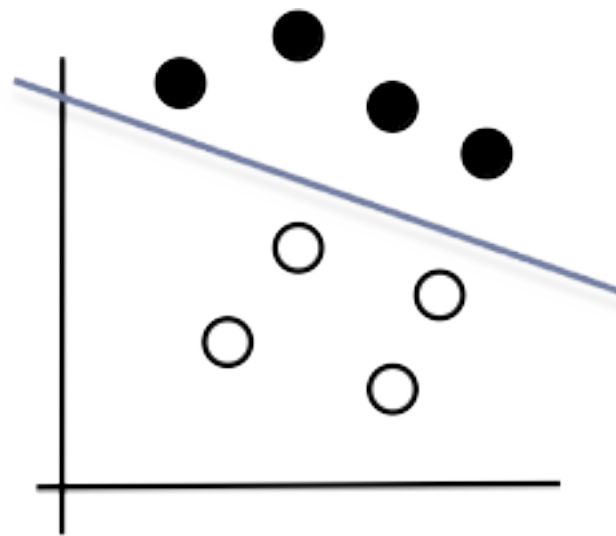
# Séparabilité linéaire

---

Le perceptron peut distinguer des points linéairement séparables.



linéairement séparable



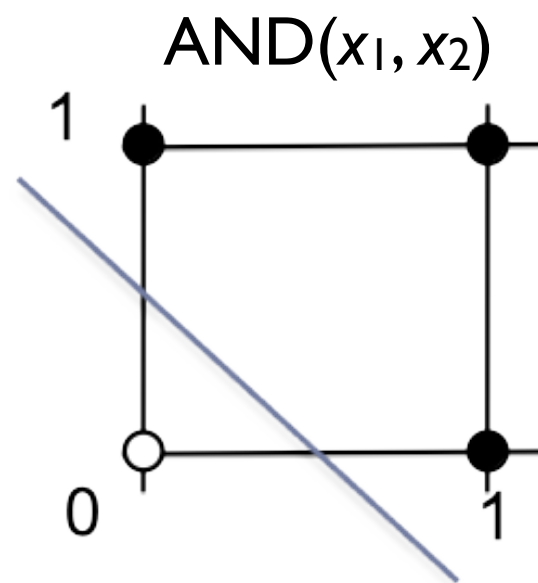
pas linéairement  
séparable

À remarquer qu'à très haute dimension ( $N \gg 1$ ), des entrées sont généralement plus facilement séparables.

# Exemples : AND, OR, et XOR

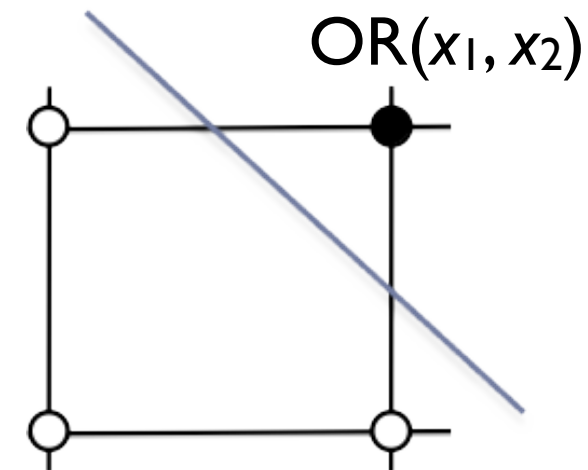
Le cas binaire :  $x_1, x_2 = 0$  ou  $1$

- Pour des entrées binaires  $x_1, x_2$ , est-ce que le perceptron peut apprendre les fonctions logiques AND (et), OR (ou) et XOR (ou exclusif) ?



○ = Faux  
● = Vrai

$w_1=1$   
 $w_2=1$   
 $b=-1/2$



$w_1=1$   
 $w_2=1$   
 $b=-3/2$

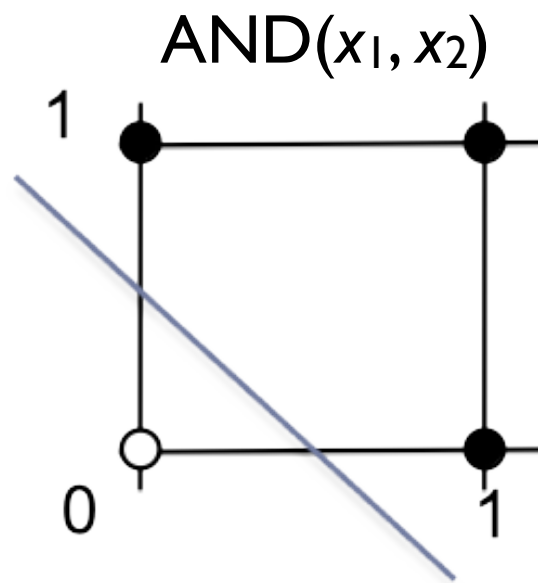


Un perceptron peut exécuter  
un *et* et un *ou*.

# Exemples : AND, OR, et XOR

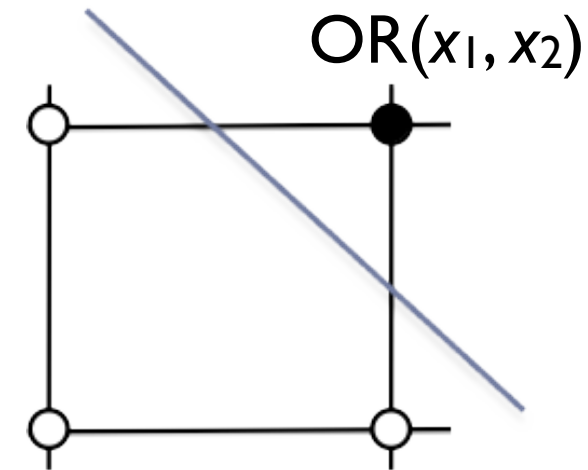
Le cas binaire :  $x_1, x_2 = 0$  ou  $1$

- Pour des entrées binaires  $x_1, x_2$ , est-ce que le perceptron peut apprendre les fonctions logiques AND (et), OR (ou) et XOR (ou exclusif) ?

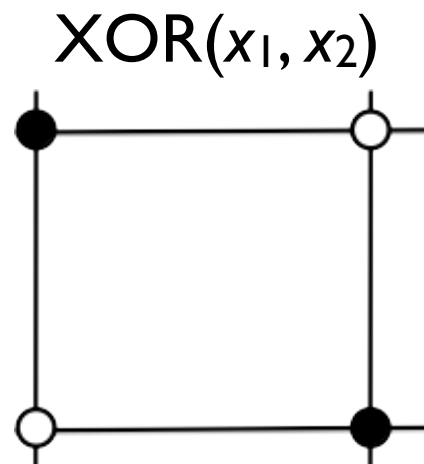


○ = Faux  
● = Vrai

$w_1=1$   
 $w_2=1$   
 $b=-1/2$



$w_1=1$   
 $w_2=1$   
 $b=-3/2$



Un perceptron peut exécuter  
un *et* et un *ou*.

Un perceptron ne peut pas exécuter  
un *ou exclusif*.



# Algorithme d'apprentissage du perceptron

---

1. Initialiser les poids et le seuil : choisir  $w_i, b$  aléatoirement dans l'intervalle  $[-1, 1]$

# Algorithme d'apprentissage du perceptron

---

1. **Initialiser les poids et le seuil** : choisir  $w_i, b$  aléatoirement dans l'intervalle  $[-1, 1]$
2. **Sélectionner aléatoirement une entrée  $k$  ( $x_i^k$ ) aléatoirement parmi l'ensemble des  $K$  paires “entrée–sortie” d'entraînement (connaissant le résultat souhaité  $y^k$ ). Utiliser les poids et le seuil actuels pour faire la classification :**

$$\hat{y} = \text{sign} \left( \sum_{j=1}^N w_j x_j^k + b \right)$$

# Algorithme d'apprentissage du perceptron

---

1. **Initialiser les poids et le seuil** : choisir  $w_i, b$  aléatoirement dans l'intervalle  $[-1, 1]$
2. **Sélectionner aléatoirement une entrée  $k$  ( $x_i^k$ ) aléatoirement parmi l'ensemble des  $K$  paires “entrée–sortie” d'entraînement (connaissant le résultat souhaité  $y^k$ ). Utiliser les poids et le seuil actuels pour faire la classification :**

$$\hat{y} = \text{sign} \left( \sum_{j=1}^N w_j x_j^k + b \right)$$

3. **Si la classification est correcte ( $\hat{y} = y^k$ ), continuer avec 2.**

# Algorithme d'apprentissage du perceptron

---

1. **Initialiser les poids et le seuil** : choisir  $w_i, b$  aléatoirement dans l'intervalle  $[-1, 1]$
2. **Sélectionner aléatoirement une entrée  $k$  ( $x_i^k$ )** aléatoirement parmi l'ensemble des  $K$  paires “entrée–sortie” d'entraînement (connaissant le résultat souhaité  $y^k$ ). **Utiliser les poids et le seuil actuels pour faire la classification** :

$$\hat{y} = \text{sign} \left( \sum_{j=1}^N w_j x_j^k + b \right)$$

3. **Si la classification est correcte** ( $\hat{y} = y^k$ ), continuer avec 2.
4. **Si la classification est fausse, mettre à jour les poids et le seuil** selon la règle suivante :

$$b \rightarrow b + \eta y^k$$
$$w_i \rightarrow w_i + \eta y^k x_i^k$$

$\eta > 0$  ... taux d'apprentissage

# Théorème de convergence

Est-ce qu'on peut être sûr que le perceptron va bien apprendre la classification ?

Théorème (prouvé) :

*Pour tout ensemble de données linéairement séparable, la règle d'apprentissage du perceptron est garantie de trouver une solution dans un nombre fini d'itérations.*

Exemple de mise-à-jours consécutifs d'un perceptron

