Culture Sciences de l'Ingénieur

https://eduscol.education.fr/sti/si-ens-paris-saclay

Journées pédagogiques de la revue 3EI et du DER Nikola Tesla :

Réflexions sur les formations génie électrique et science de l'information

Jeudi 23 et vendredi 24 juin 2022

Présentation de la conférence :

Introduction à l'intelligence artificielle pour les sciences de l'ingénieur

par Anthony JUTON

Les présentations de ces journées se retrouvent sur :

https://eduscol.education.fr/sti/si-ens-paris-saclay/actualites/journees-3ei-der-nikola-tesla

école normale supérieure paris-saclay

Introduction à l'intelligence artificielle pour les sciences de l'ingénieur

université PARIS-SACLAY

Dossier IA et Génie électrique

école normale supérieure paris-saclay

https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/dossier-intelligence-artificielle

université PARIS-SACLAY

Qu’est-ce que l’intelligence artificielle ?

Applications grands publics de l’IA

- reconnaissance vocale,

Amazon echo dot, *Raimond Spekking, Wikimedia*

- traitement naturel du langage (bots, agents conversationnels, traduction),

Logo Google Translate

- classification / identification des images

Reconnaissance faciale,
Peyman Majidi Moein, Wikimedia

Création de Google Deep Dream, *Slate*

Et recommandations musicales, personnages non joueurs des jeux vidéos, suggestions de présentation des données par Powerpoint, génération de textes, de peintures...

Qu’est-ce que l’intelligence artificielle ?

Applications de l’IA aux sciences de l’ingénieur

- Voitures autonomes (reconnaissance de panneaux, classification des acteurs, conduite)
- Logistique (recherche des parcours optimaux)
- Imagerie médicale
- Algorithmes de segmentation ou de réduction de paramètres, réseaux de neurones de simulation de comportement physique non linéaire (mécanique des fluides), pour simuler des systèmes complexes

Google Self-Driving Car, R Boed, Wikimedia

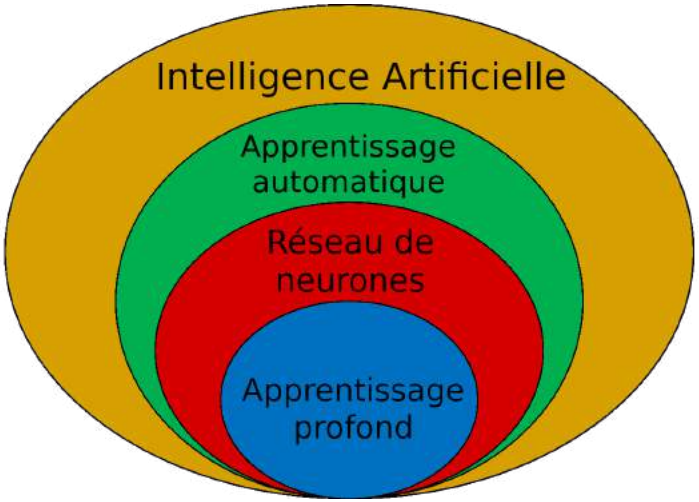
Et aussi : Recherche de signaux faibles dans le flux de données, aide au diagnostic médical, conception de vaccins, robots footballeurs

Amazon robots, JBLM PAO, Wikimedia

Qu’est-ce que l’intelligence artificielle ?

Une définition de l’IA : famille de méthodes visant à résoudre des problèmes.

IA forte (ou IA générale) : IA permettant de se comporter comme un humain (SF en 2022)
IA faible : IA permettant de résoudre un certain type de problème.



IA symbolique : utilise des règles haut-niveau écrites par un expert de la discipline (systèmes experts)
GOFAI ("Good Old-Fashioned Artificial Intelligence")

Apprentissage automatique / machine learning
L’ordinateur apprend à partir de données.

IA sub-symbolique dont réseaux de neurones

Apprentissage automatique profond / Deep learning
L’ordinateur optimise les poids de réseaux de neurones à partir de données.

Qu’est-ce que l’intelligence artificielle ?

Quelques repères historiques

1950
« les machines peuvent-elles penser ? »
Alan Turing

1956
Dartmouth,
Workshop Artificial Intelligence
John McCarthy

1957
Le perceptron
Frank Rosenblatt

1997
IBM Deep Blue
bat Kasparov aux échec

2012
Un réseau de neurones gagne ImageNet

2017
Google AlphaGo
bat Lee Sedol au Go

1947
Invention du transistor

1990
Naissance du web

1997
Premier GPU

Calculateur Jean Zay
4000 CPU, 3000 GPU,
28 PFlop/s

Source : Idris

15 articles du dossier

La Revue 3E.I

Introduction à l'apprentissage automatique

- Régression linéaire ou polyn. Algo K plus proches voisins Algo des k-moyennes
- Introduction à l'apprentissage profond
- Introduction à l'apprentissage par renforcement
- Séries temporelles et réseaux de neurones récurrents

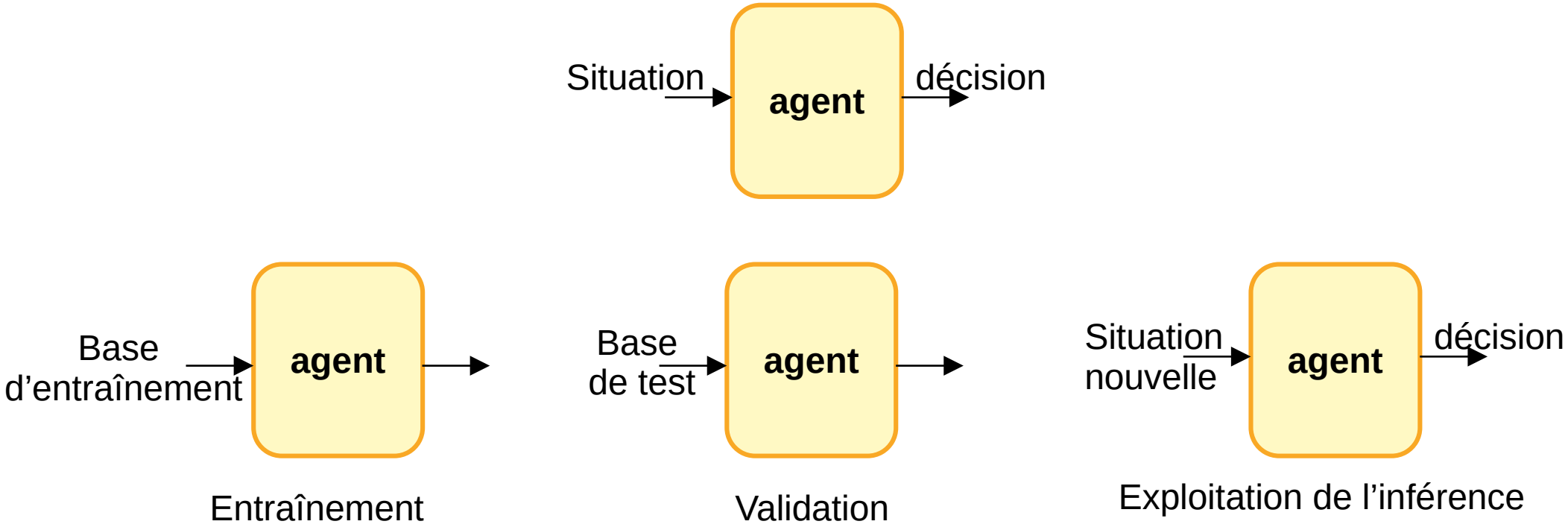
- Régression, K-means avec l'outil IA de Matlab
- Réseau de neurones l'outil IA de Matlab
- Stabilisation d'un pendule inversé Gym
- Conduite d'un véhicule sur AirSim + Gym
- Stabilisation d'un pendule inversé Matlab

- Suivi cellulaire par traitement d'images et apprentissage
- L'IA au service de la mobilité urbaine à Rennes Lacroix
- Amélioration du modèle des matériaux non linéaires
- Commande sans capteur de machines asynchrones
- Reconstruction IRM
- IA pour les réseaux électriques (auto-encodeur/ renforcement)

Introduction → Fondamentaux → Exercices pratiques → exemples d'applications

Qu’est-ce que l’apprentissage artificiel ?

A partir d’essais, l’agent apprend et généralise. Il peut ensuite appliquer ses « connaissances » à d’autres situations nouvelles mais similaires.

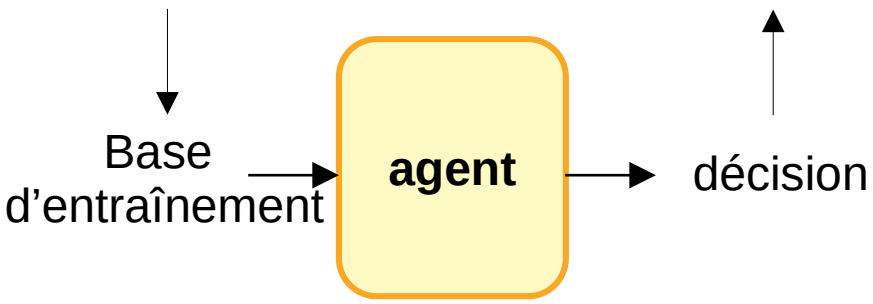


Importance de la représentativité de la base de test.
Apprentissage long.

Qu'est-ce que l'apprentissage artificiel ?

Apprentissage supervisé

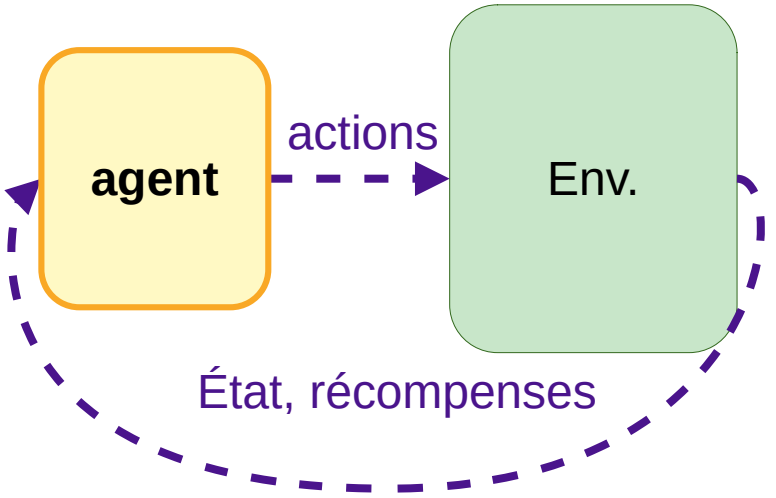
Base d'entraînement contient les **labels**



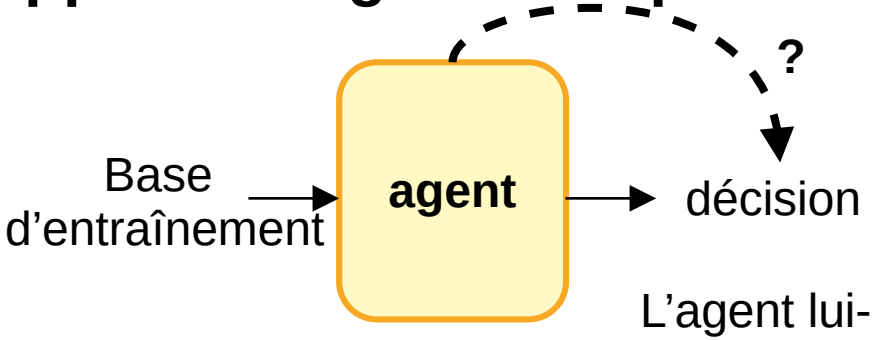
Entraînement à partir de la comparaison
décision de l'agent = label ?

Apprentissage par renforcement

L'agent teste ses décisions sur un environnement.
Des récompenses renforcent les bonnes décisions



Apprentissage non supervisé



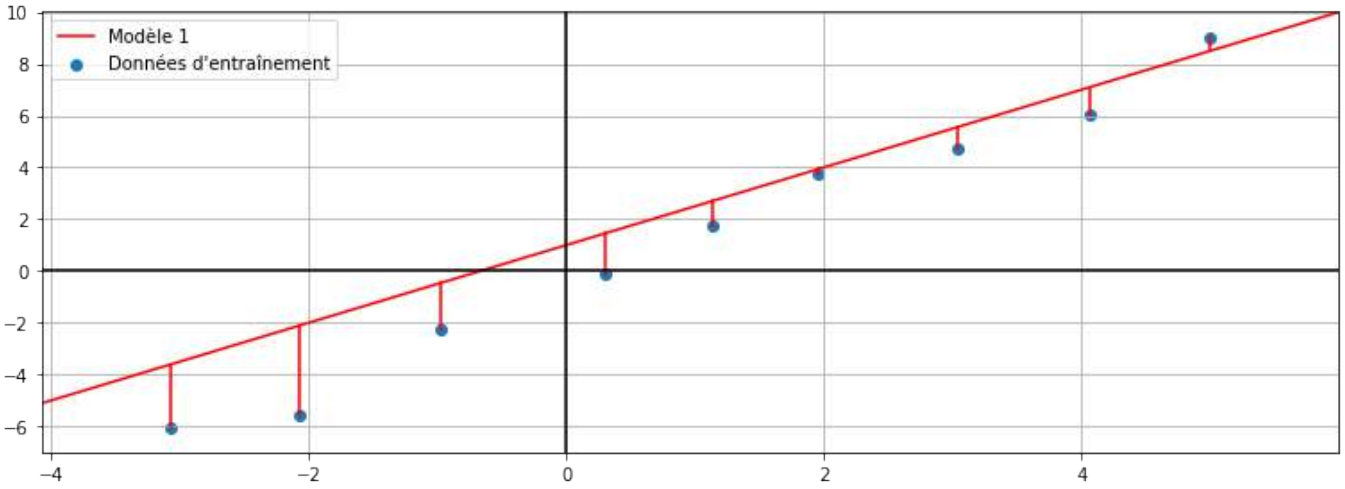
L'agent lui-même teste l'intérêt de ses décisions

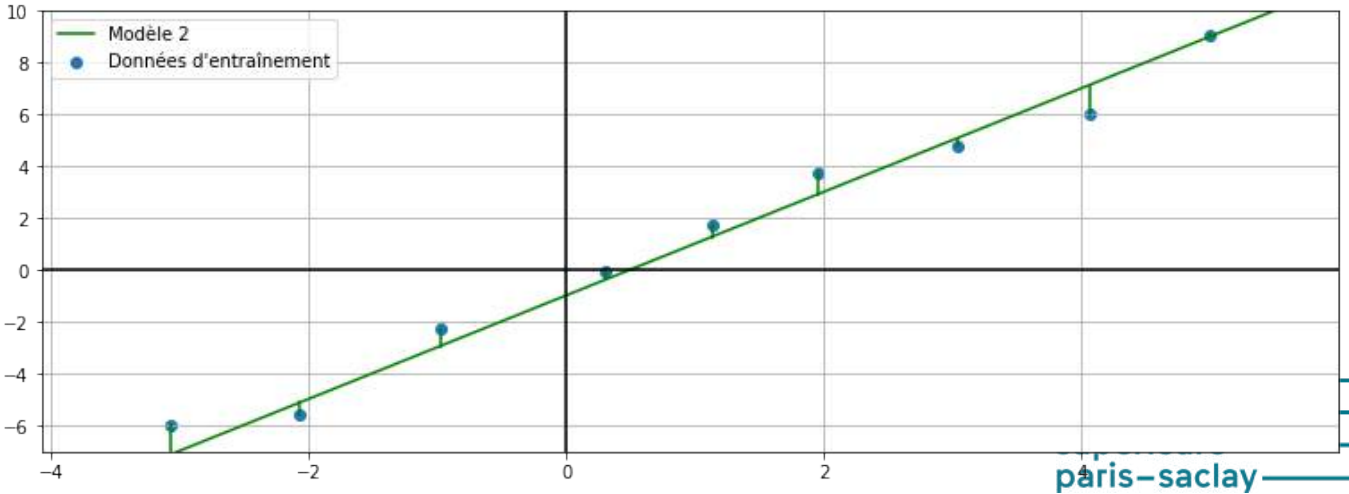
Apprentissage supervisé

Régression linéaire ou polynomiale

Deux modèles de régression linéaire :
Le premier modèle présente des écarts important entre les valeurs prédites et attendues tandis que le second minimise les carrés de ces écarts.

On peut faire des régressions avec des ordres plus importants



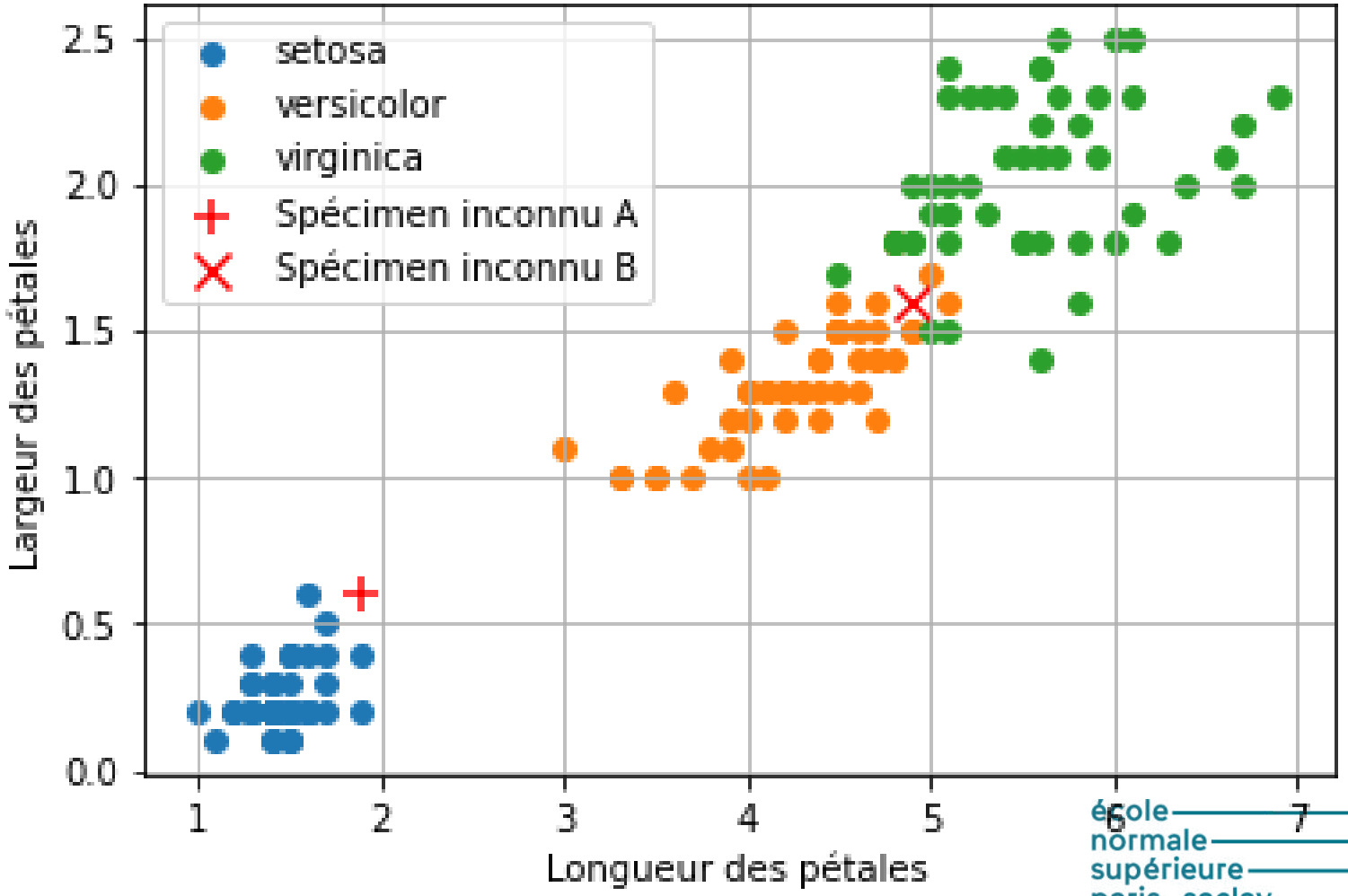


Apprentissage supervisé

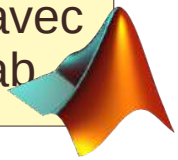
Algorithme des K plus proches voisins

Bases de donnée Iris de Fisher
Les 150 spécimens sont répartis en 3 espèces et positionnés en fonction de la longueur et largeur des pétales.

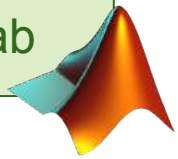
On cherche l'espèce des spécimens A et B.



Régression, KNN avec l’outil IA de Matlab



Réseau de neurones l’outil IA de Matlab



Apprentissage supervisé

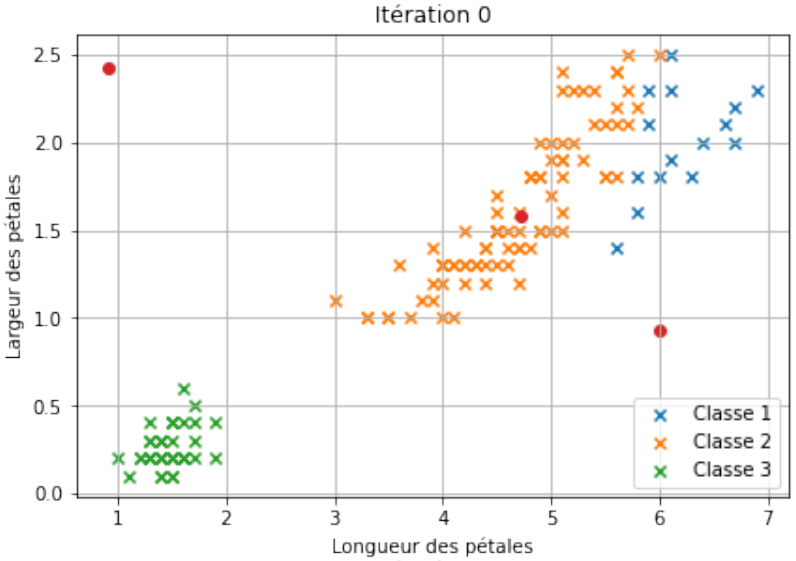
Autres algos supervisés

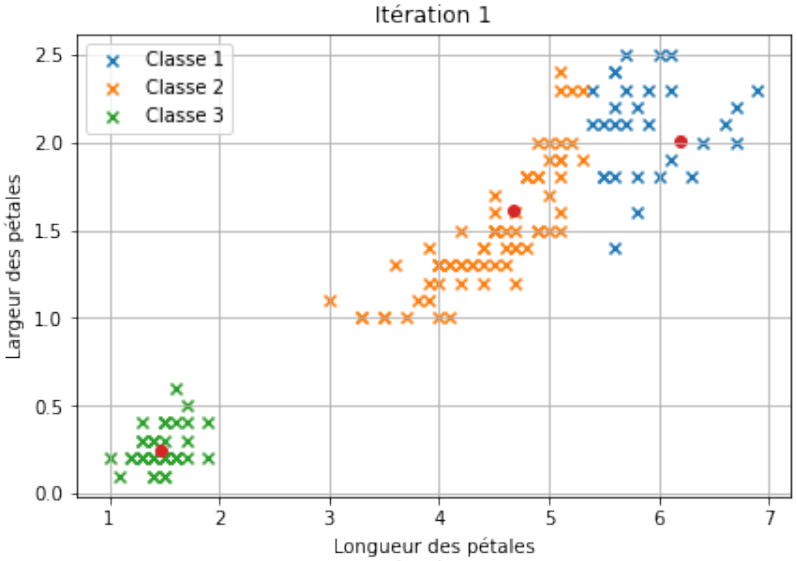
- Arbre de classification
- Réseaux de neurones artificiels (présentation suivante)
- Machine à support de vecteur (SVM)

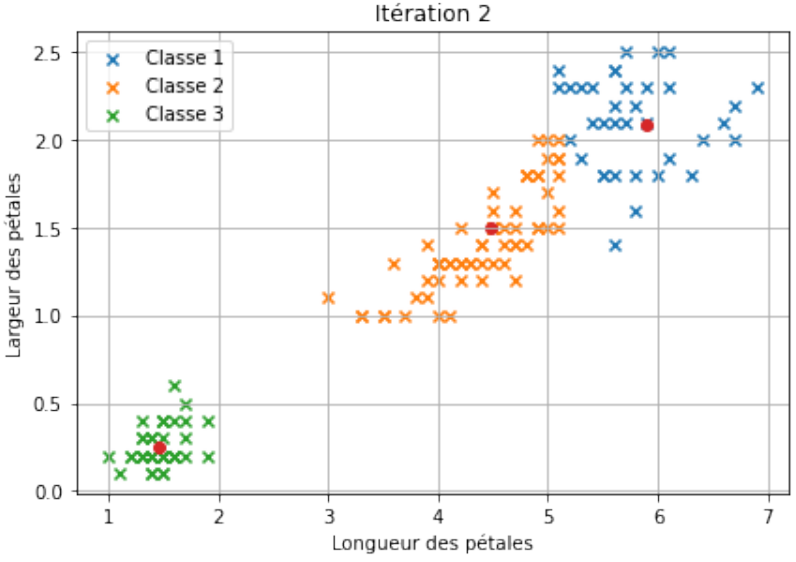
Apprentissage non-supervisé

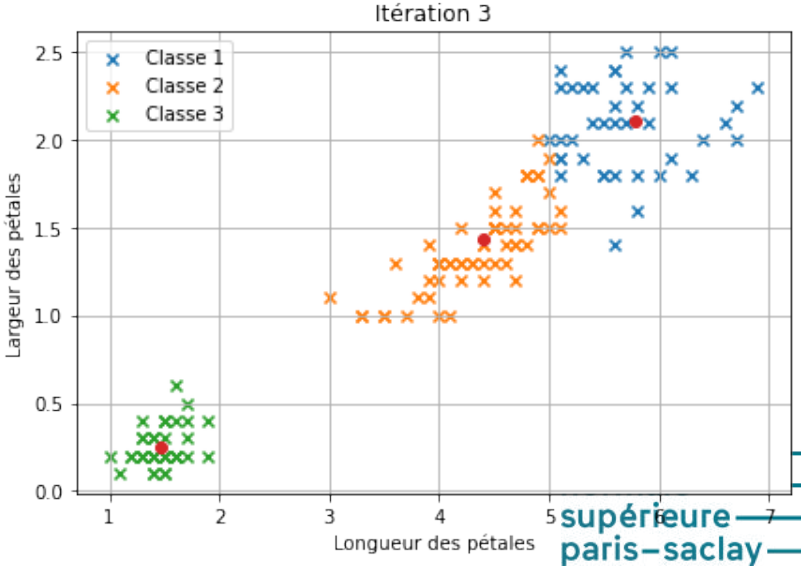
Algorithme des k-moyennes

K=3



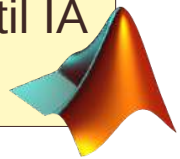






supérieure paris–saclay

K-means avec l’outil IA de Matlab



école
normale
supérieure
paris—saclay

Apprentissage profond
Les réseaux de neurones

université
PARIS-SACLAY

Apprentissage profond

Pas besoin de décrire ce qu'est une voiture ou ce qu'est un chat ou une voyelle. L'agent trouve lui-même ses règles implicites.

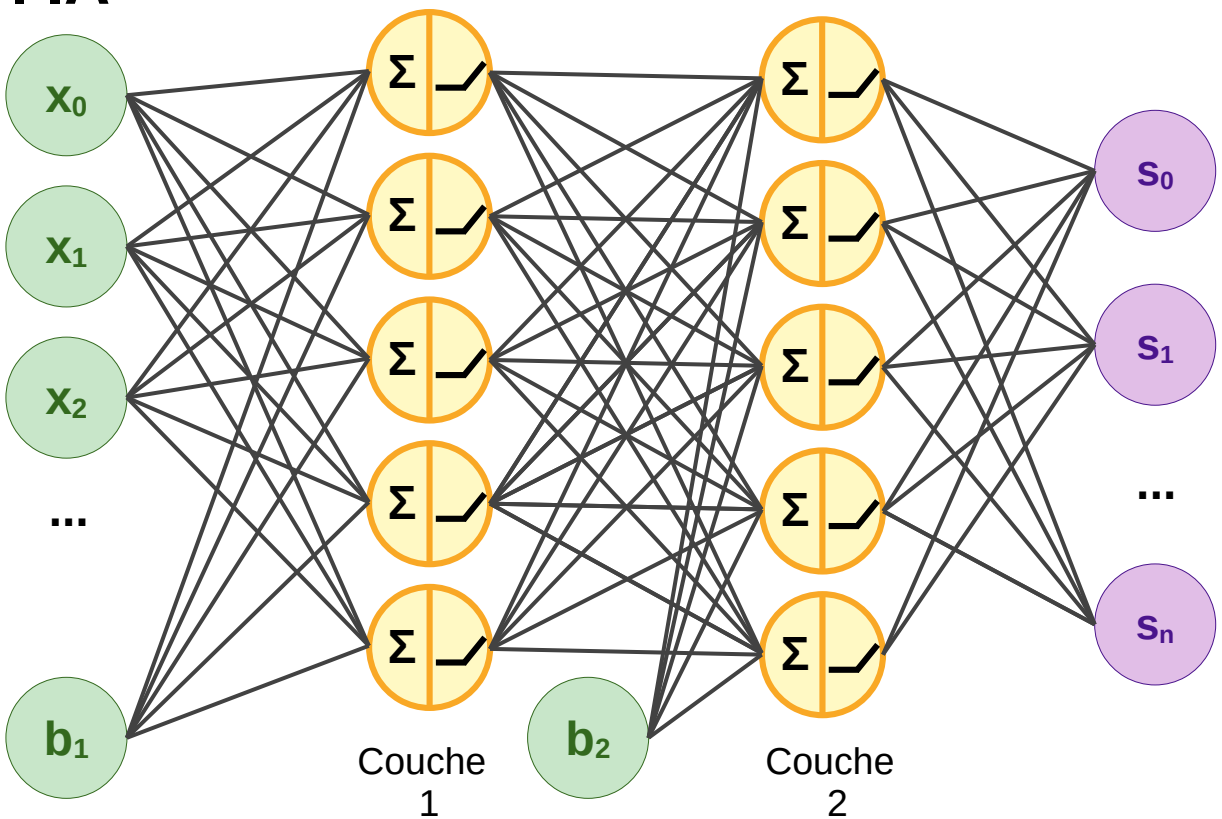
Les mécanismes de la décision sont opaques.
Des erreurs sont possibles.

Correction du système difficile

Besoin de données d'apprentissage :
Facile pour l'enseignement (mnist), moins pour la recherche
L'apprentissage est long parce qu'on explore toutes les pistes
L'apprentissage dépend des données d'entreaînement et donc de leurs biais ou insuffisances

Réseaux de neurones

Un outil de l'IA



Couche d'entrées x_k (nombres réels)

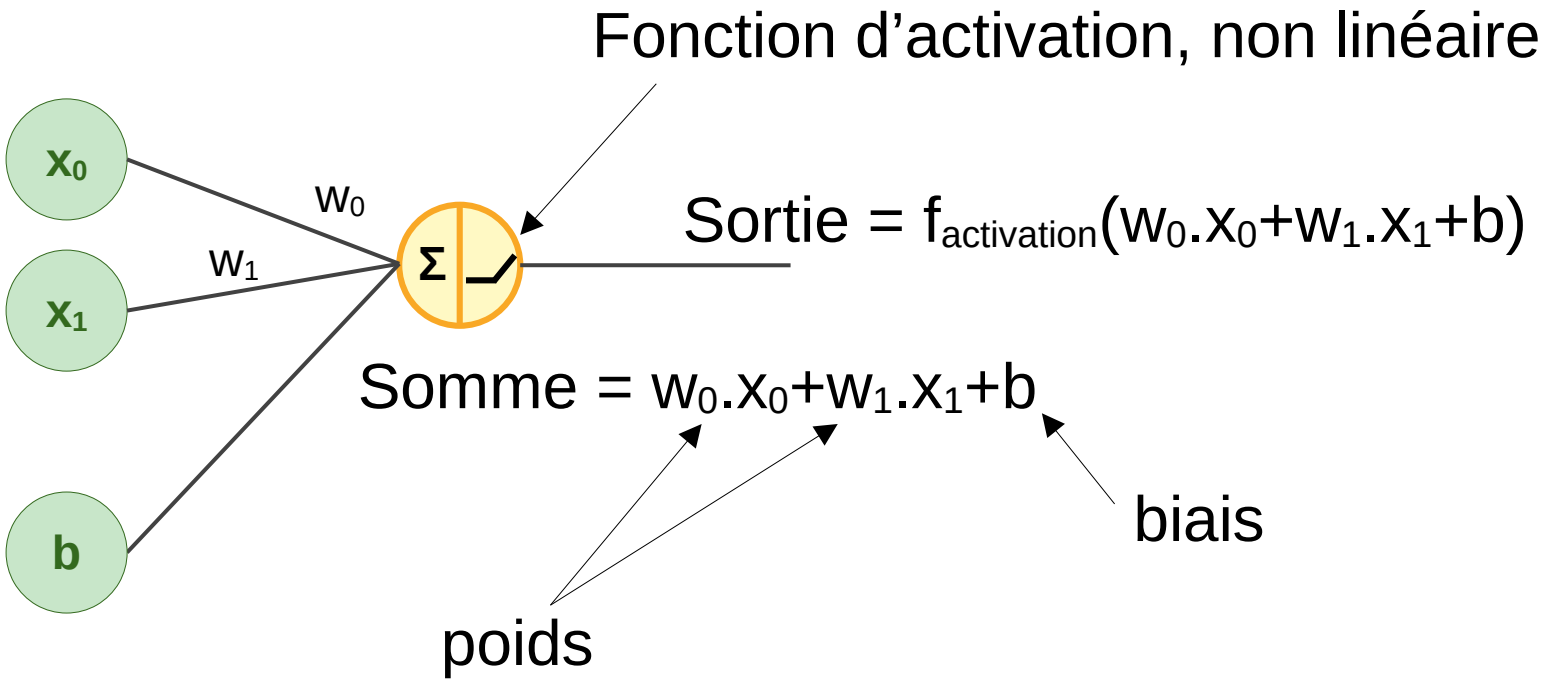
(pixels d'une image, valeurs d'un lidar, son)

Couche de sorties s_k (nombres réels)

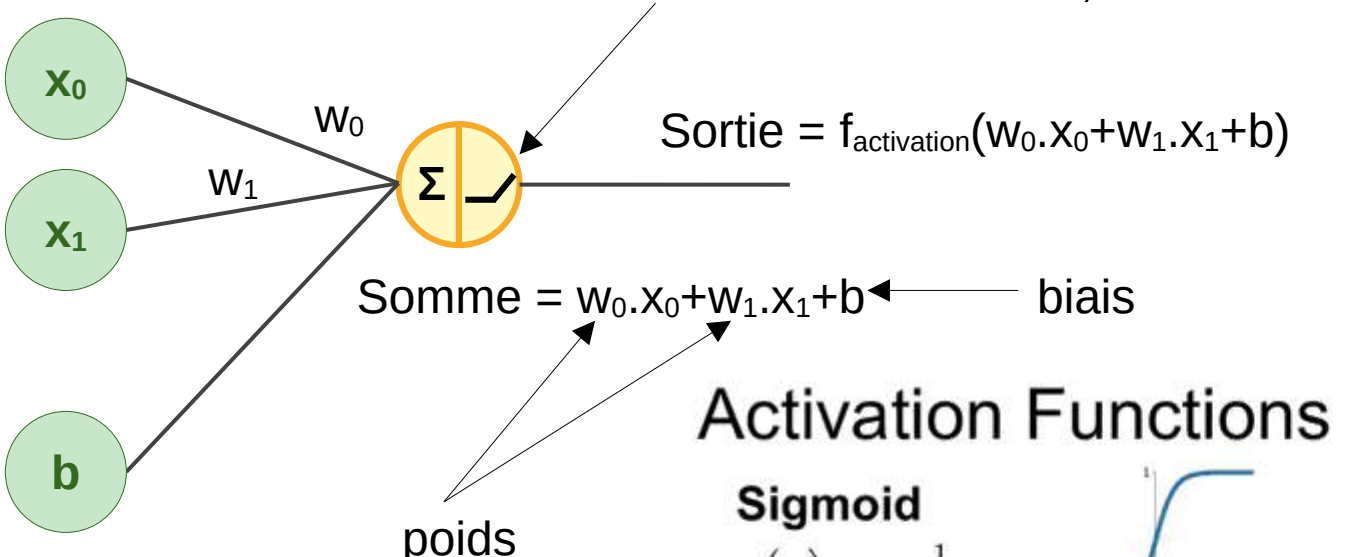
Classification, commandes...

Réseaux de neurones

Un neurone



Réseaux de neurones



Un neurone

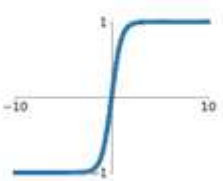
Activation Functions

Sigmoid

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

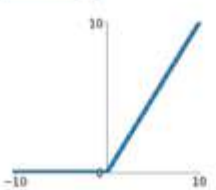
tanh

$\tanh(x)$



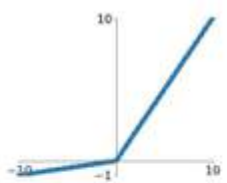
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

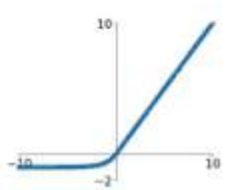


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

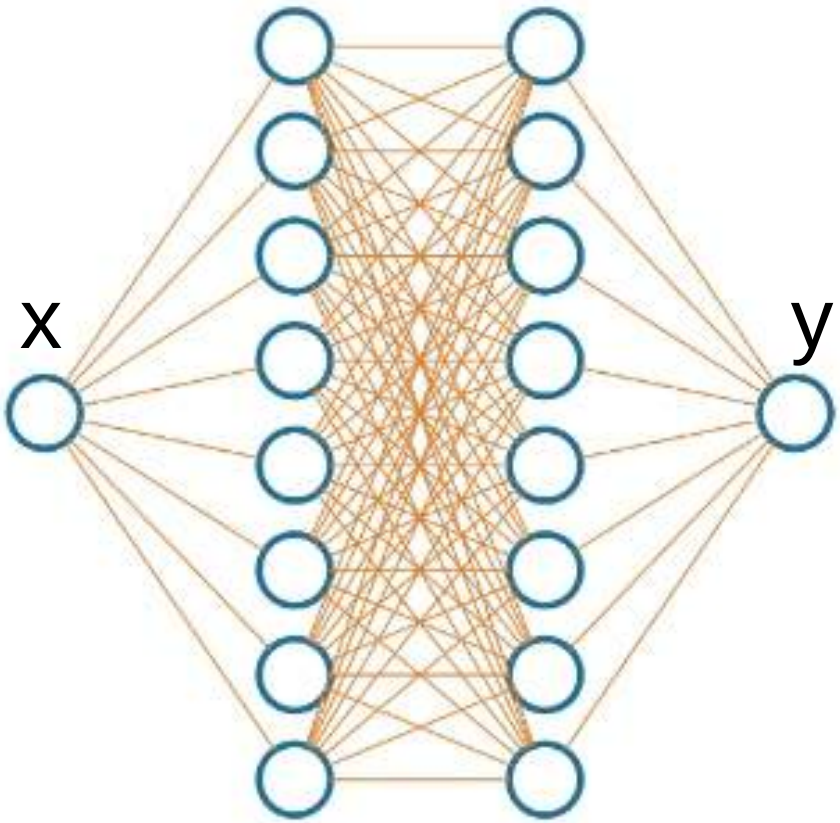


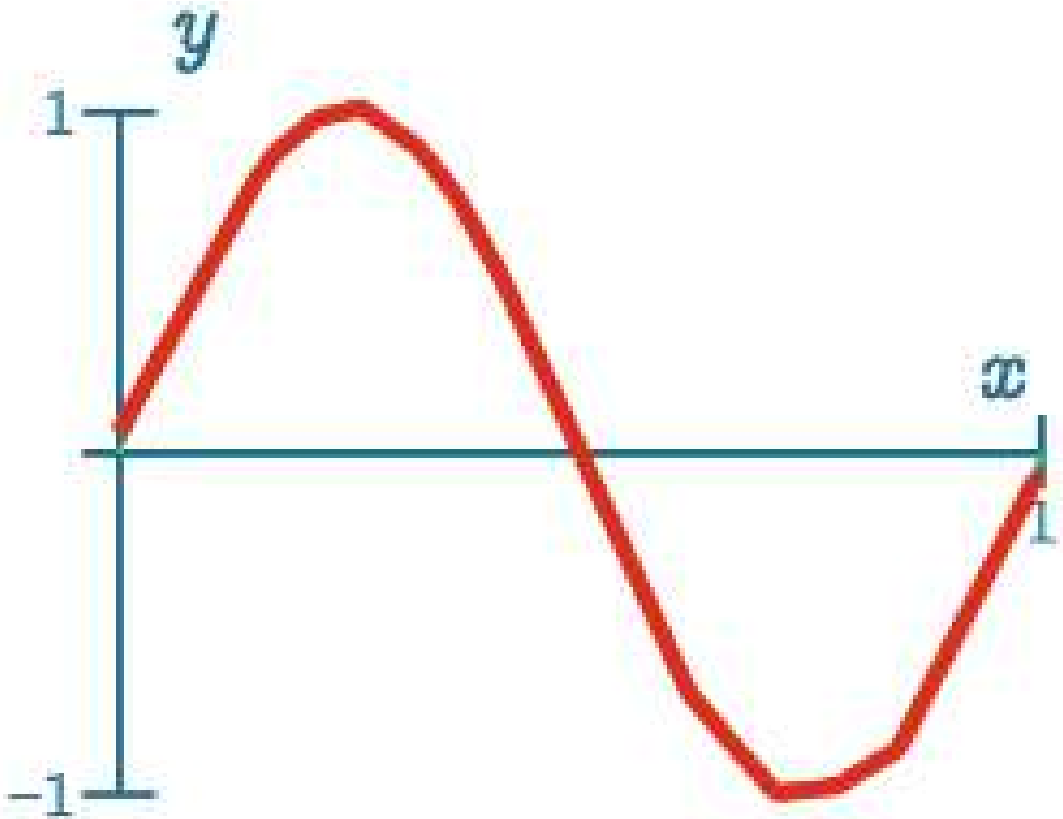
Réseaux de neurones

Les fonctions d'activation

Approximation y = sin(x) avec un réseau de neurones

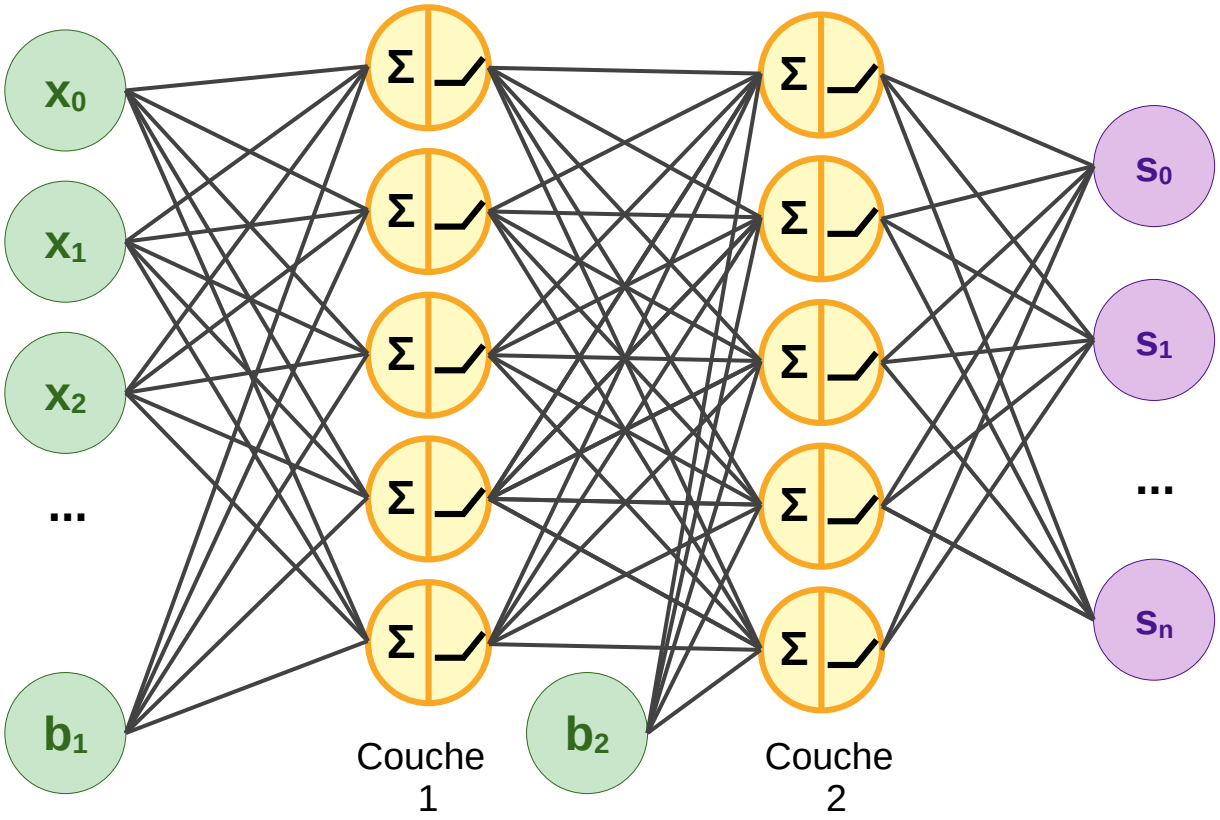
source : Neural Network from scratch





Réseaux de neurones

Couche cachée (Apprentissage profond → Deep Learning)

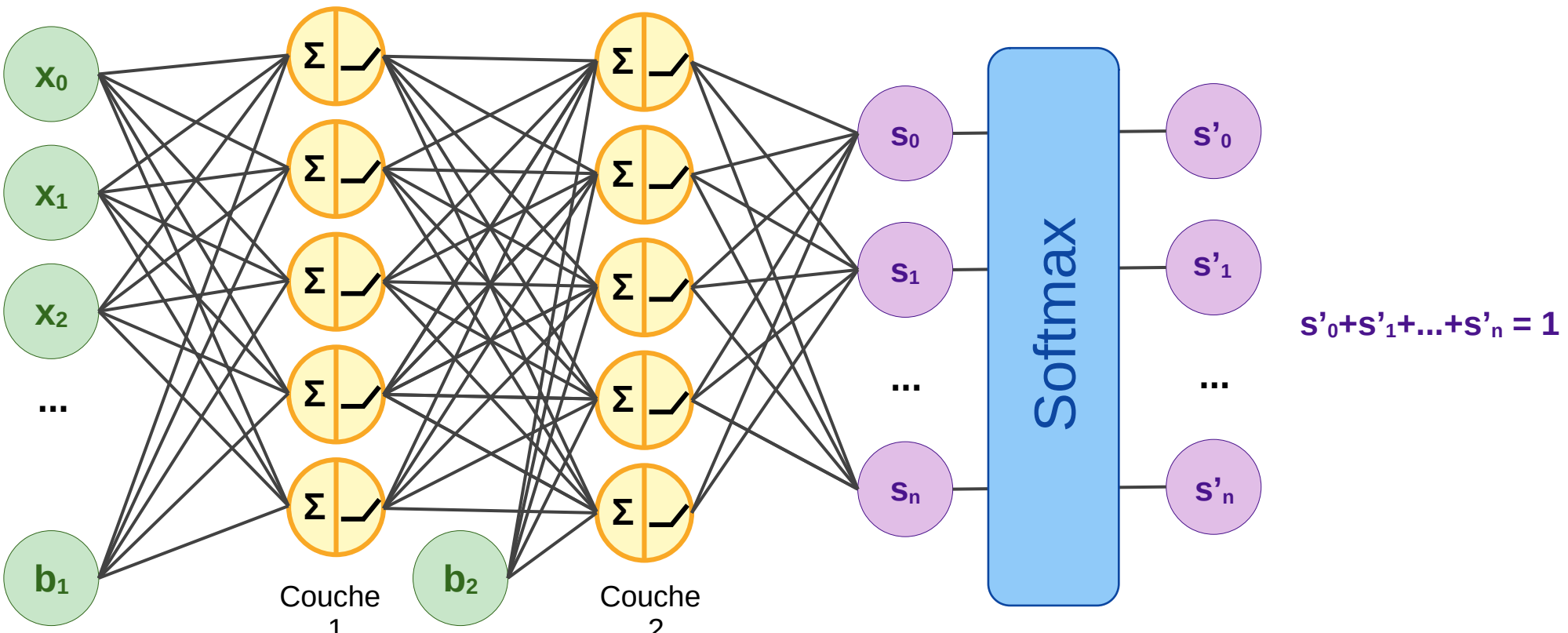


Entrées x_k

Sorties s_k

Réseaux de neurones

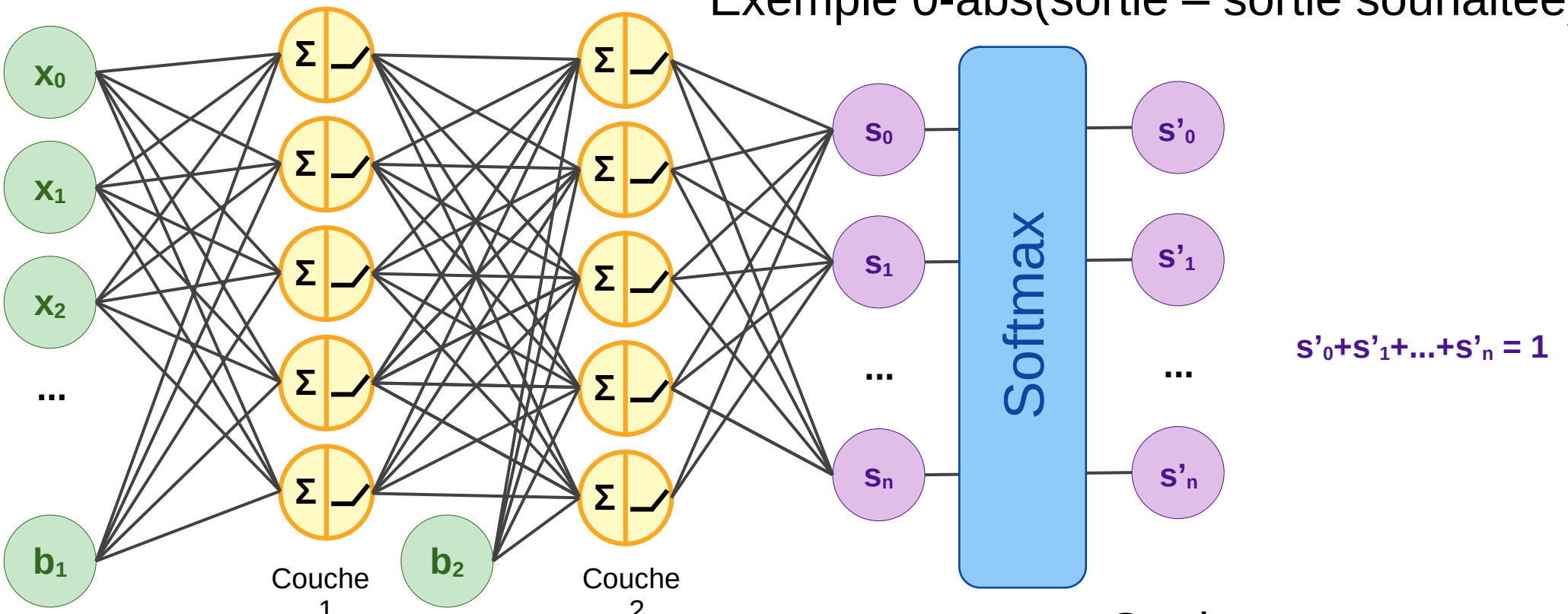
Normalisation des sorties : Softmax



Apprentissage d'un réseaux de neurones

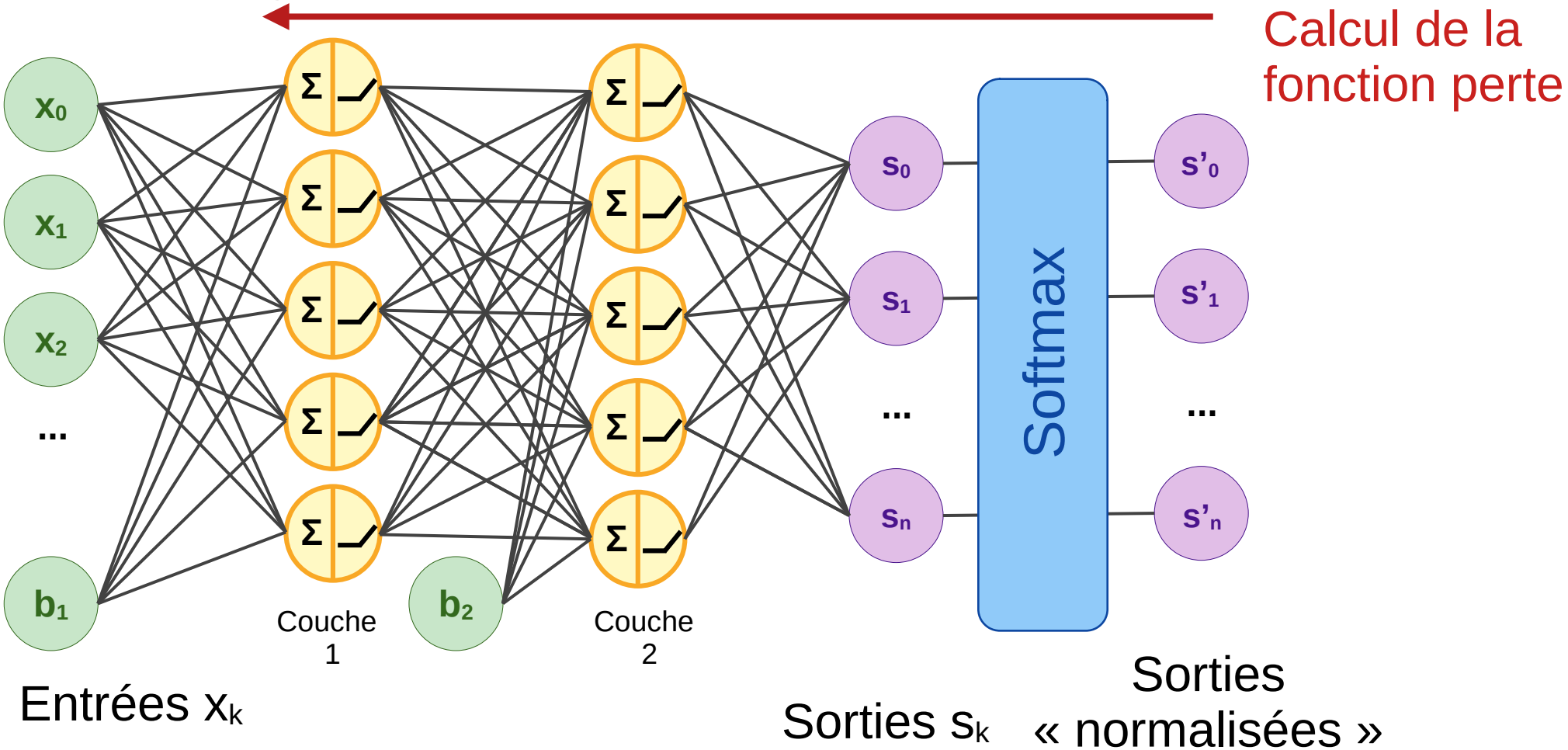
Trouver les valeurs des poids (+biais) minimisant une fonction de pertes

Fonction de pertes (loss function)
Exemple 0-abs(sortie – sortie souhaitée)



Apprentissage d'un réseaux de neurones

Méthodes d'optimisation : ex → descente de gradient
Rétropropagation des effets sur la sortie vers les couches initiales



Apprentissage d’un réseaux de neurones

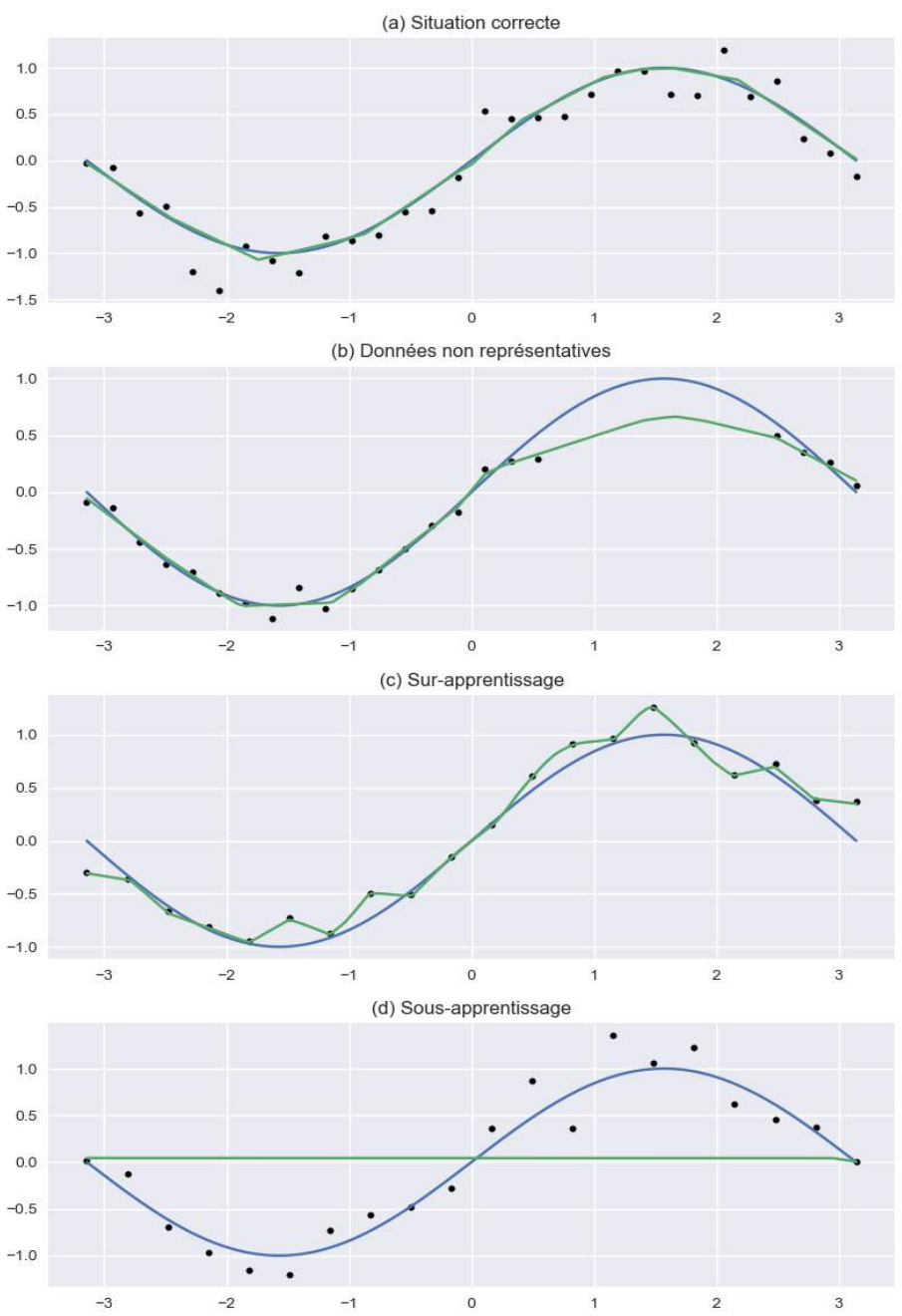
Choix des hyper-paramètres :

- Nombre de couches cachées, de neurones par couches
- Fonctions d’activations des neurones (Sigmoid, ReLU, softmax...)
- Fonction coût (Erreur quadratique, categorical cross-entropy...)
- Taux d’apprentissage (learning rate)
- Taille des lots d’entraînement (batches)
- Nombre d’époque pour l’entraînement
- Algorithme d’apprentissage (descente de gradient, algo génétique, Adam…)
- Condition d’arrêt de l’apprentissage (risque de sur-apprentissage)

+ Structure du réseau

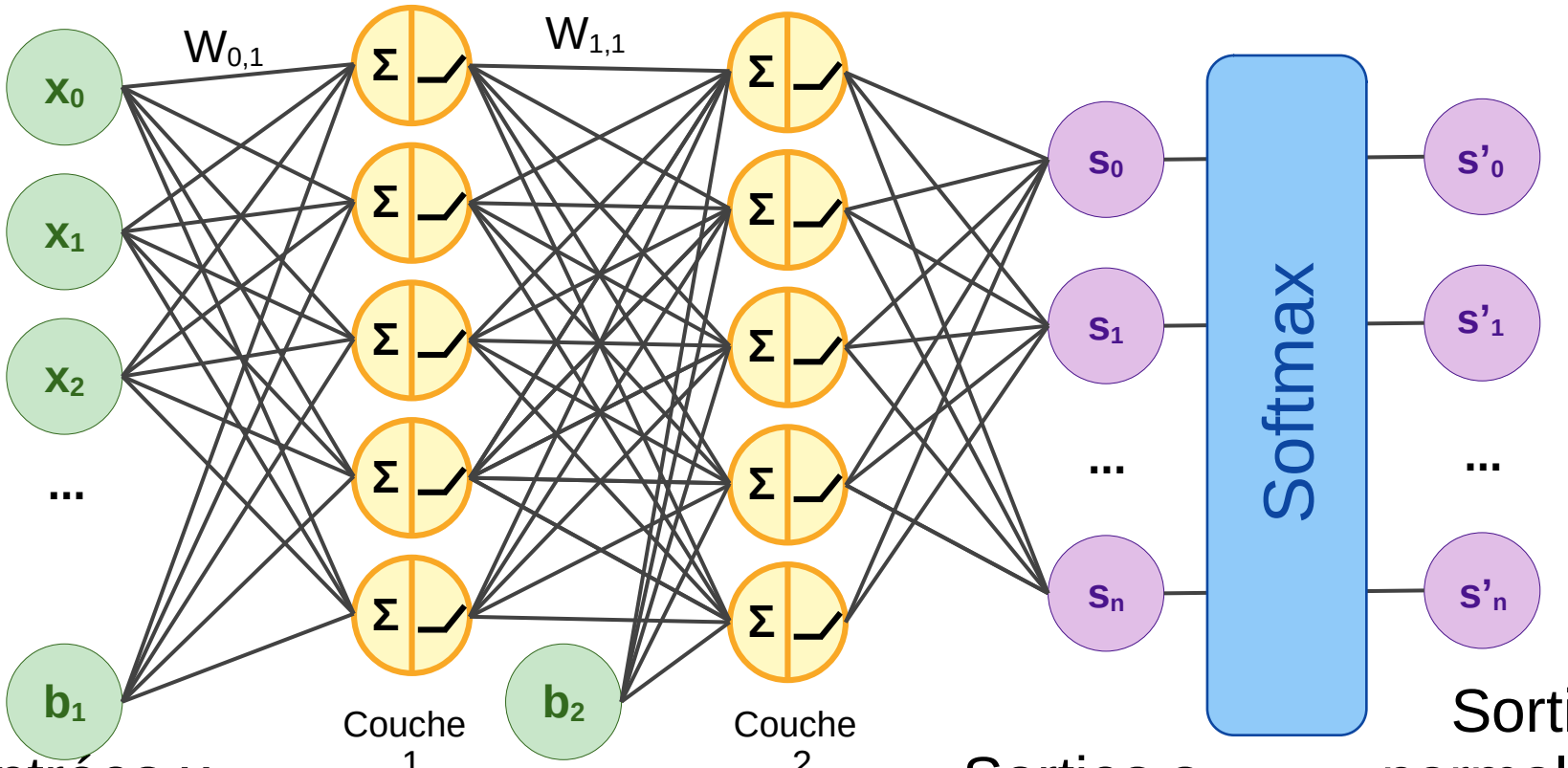
Apprentissage d’un réseaux de neurones

Résultat d'entraînement d’un MLP à une couche cachée à partir des mesures (points noirs) issues d’une fonction sinus.
(a) Les données représentent bien la fonction, et le MLP est bien choisi.
(b) Certaines données ont été supprimées, le réseau y est moins performant.
(c) Le MLP est sur-dimensionné, celui-ci ne généralise pas.
(d) Cette fois, le réseau est sous-dimensionné et ne parvient pas à apprendre quoi que ce soit.



Réseaux de neurones spécifiques

MLP : Mulitp-layer Perceptron fully-connected



Ex. classique → classification chiffres mnist

2 ensembles : training et test avec à chaque fois label et images (28 px*px).

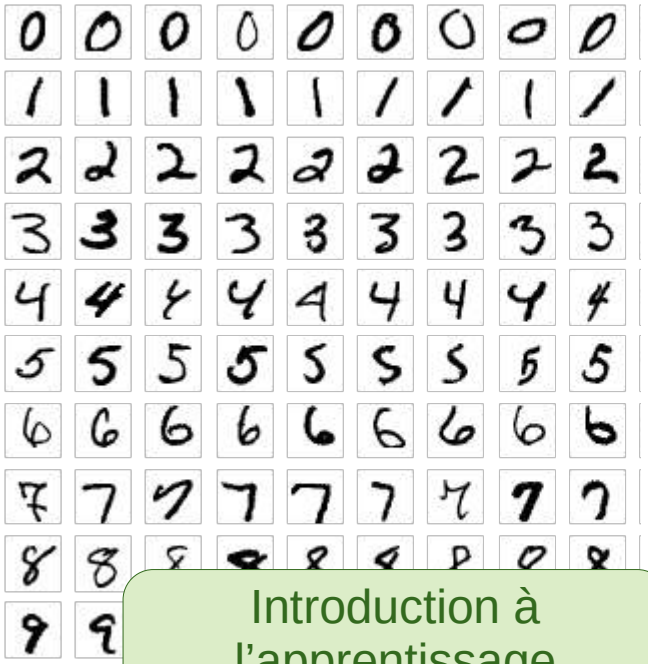
TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

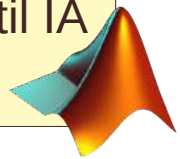


Introduction à l'apprentissage profond

http://yann.lecun.com/exdb/mnist/index.html

https://colab.research.google.com/github/trekhleb/machine-learning-experiments/blob/master/experiments/digits_recognition_mlp/digits_recognition_mlp.ipynb#scrollTo=RKGqJRrcLk5U

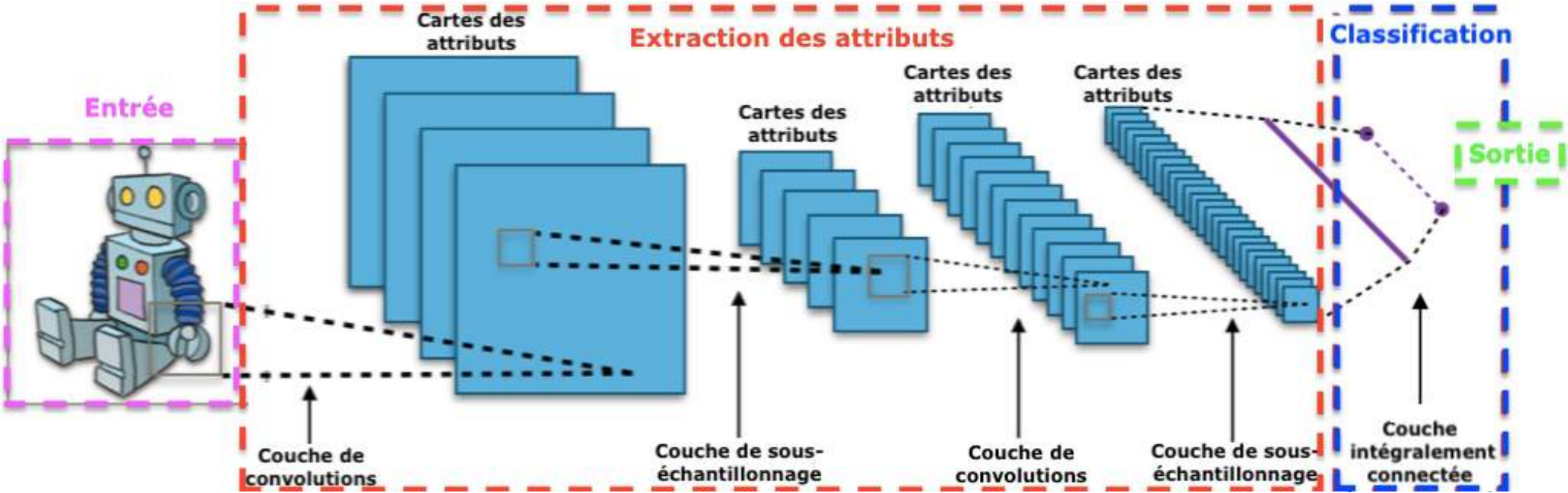
K-means avec l’outil IA de Matlab



Réseaux de neurones spécifiques

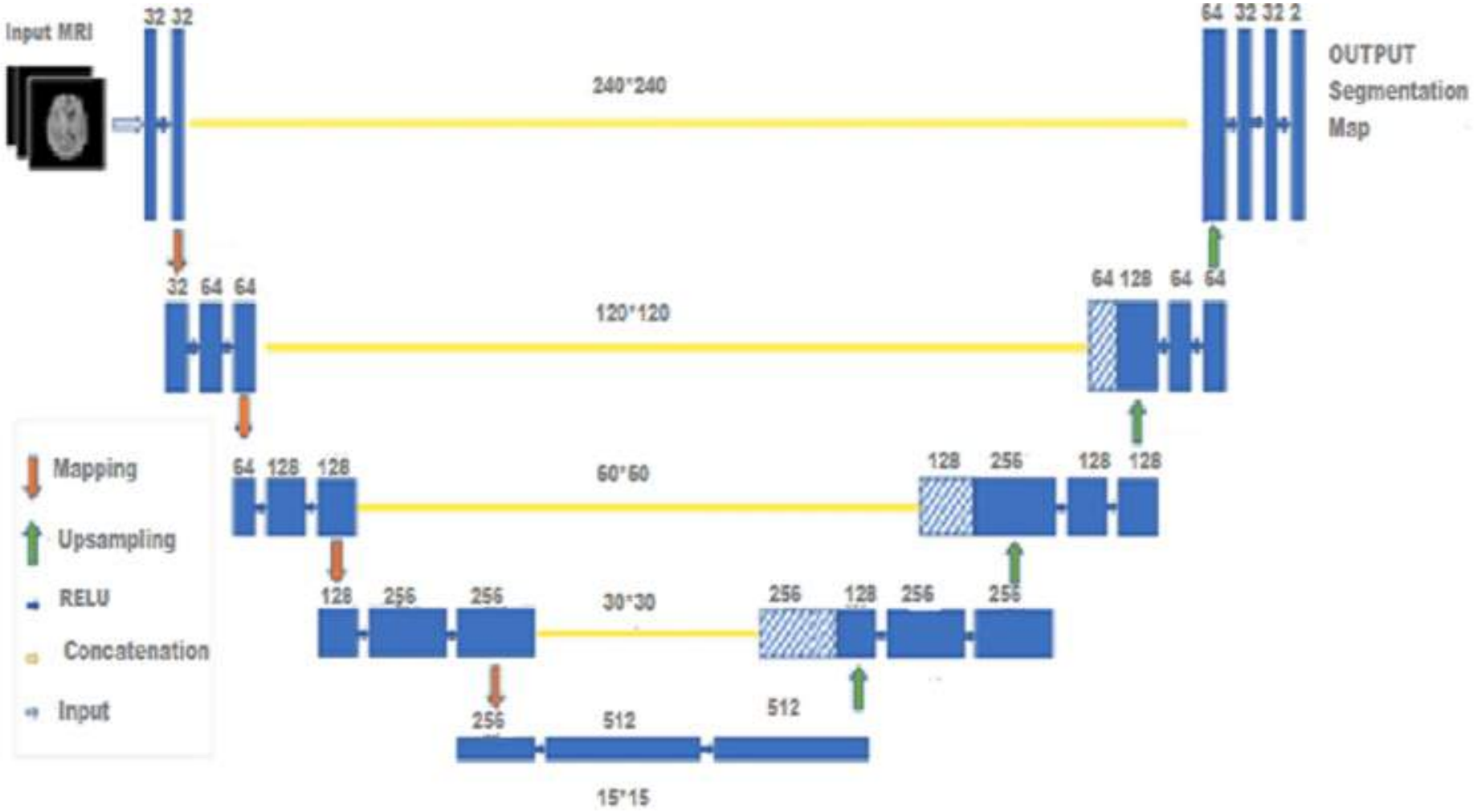
CNN : Convolution Neural Network

Utilisé en traitement d'images, avec des outils dédiés



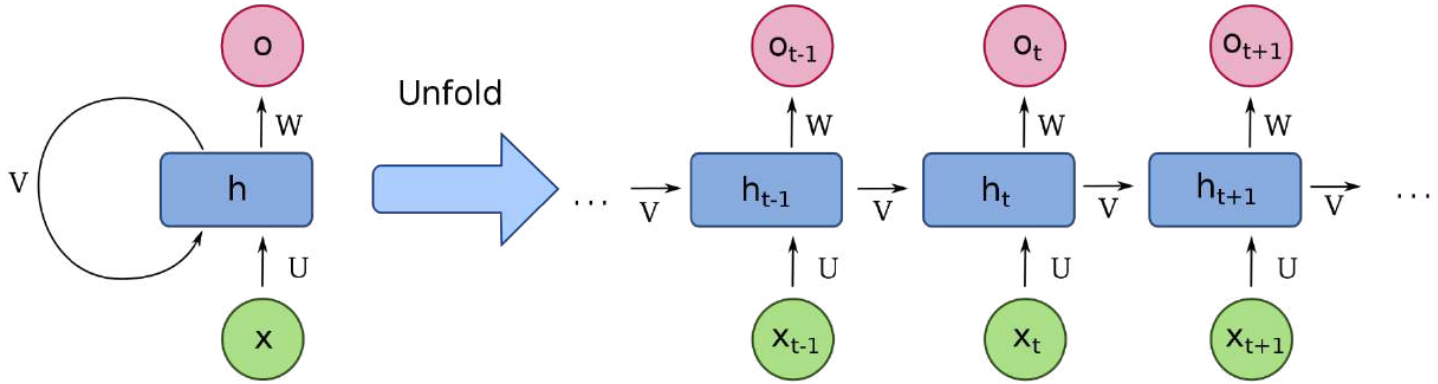
Réseaux de neurones spécifiques

U-net pour la détection sur des images



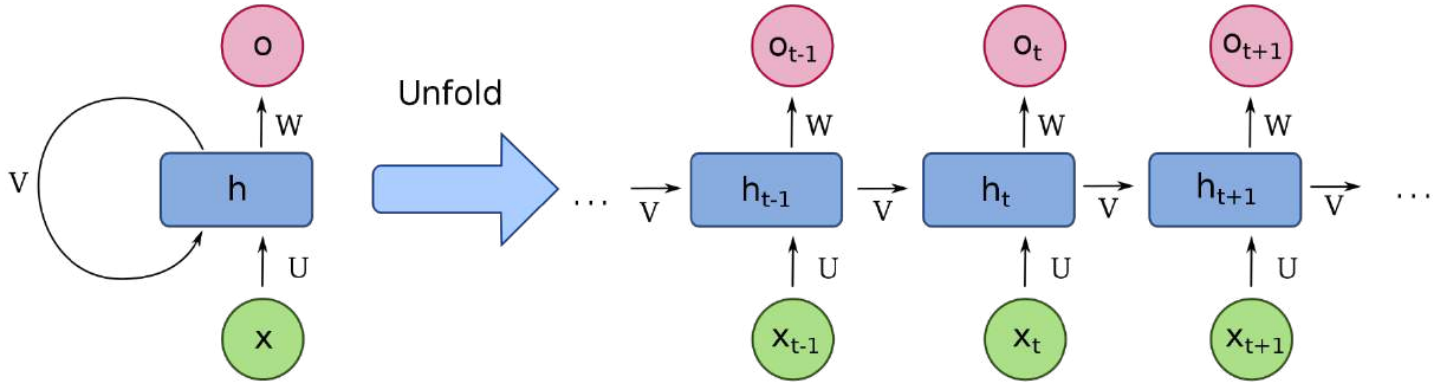
Réseaux de neurones spécifiques

Les réseaux récurrents (voir présentation suivante)



Réseaux de neurones spécifiques

Les réseaux auto-encodeurs



Les framework

Les framework proposent des réseaux et surtout des fonctions optimisées d'entraînement des réseaux, des outils pour définir la meilleure structure de réseaux et pour proposer des hyperparamètres corrects.

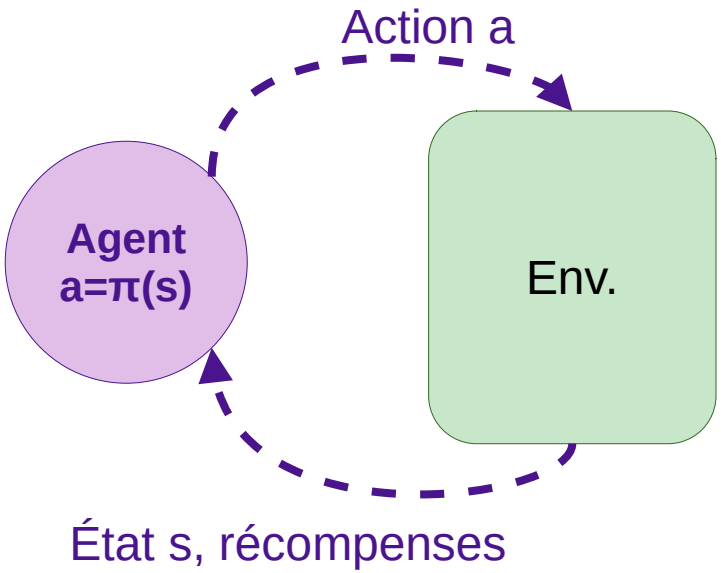
Tensorflow (Google)
PyTorch (Facebook – Meta)
Matlab

école
normale
supérieure
paris—saclay

Apprentissage par renforcement

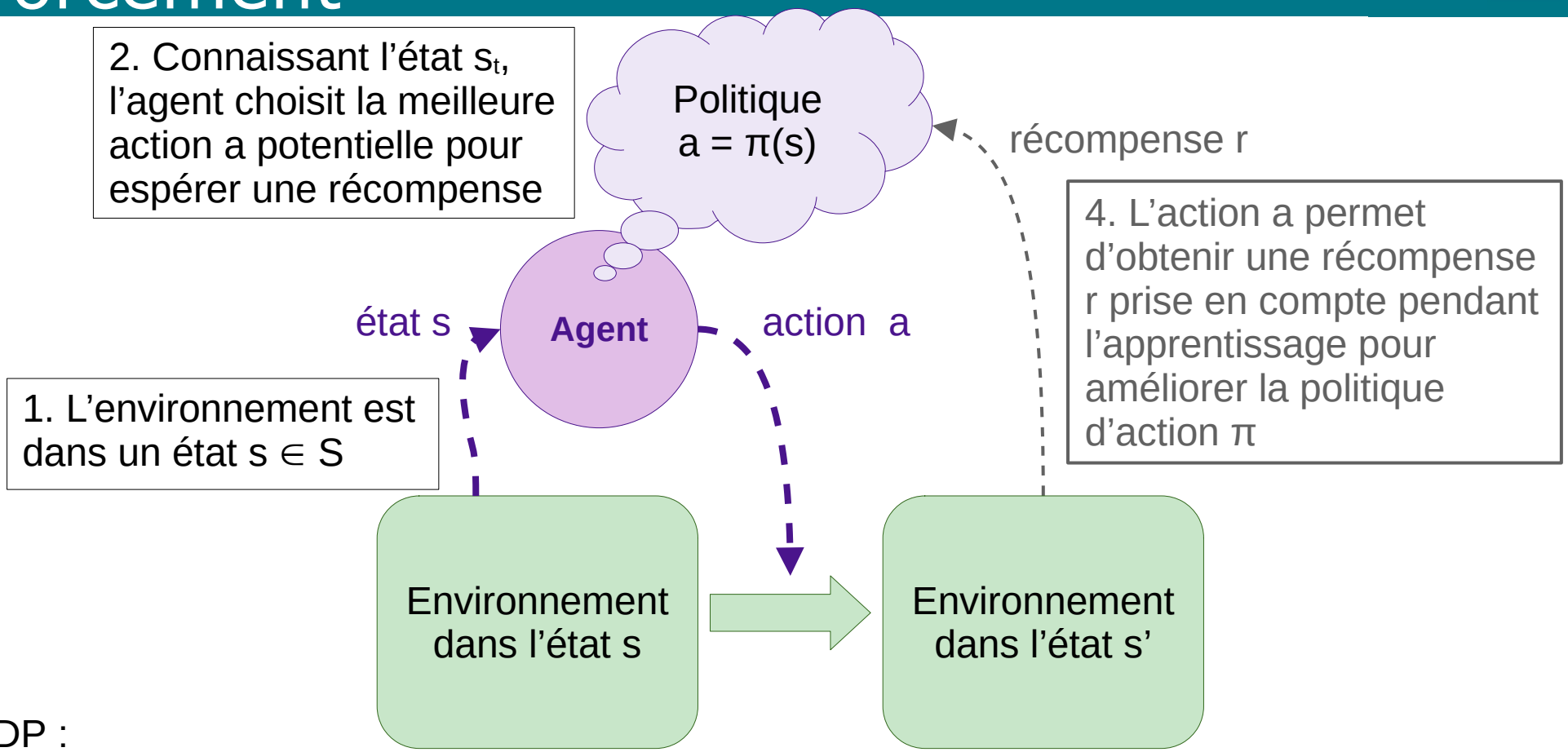
université
PARIS-SACLAY

Principes de l’apprentissage par renforcement



Durant l’apprentissage, l’agent améliore sa politique pour maximiser la récompense reçue.

Principes de l'apprentissage par renforcement



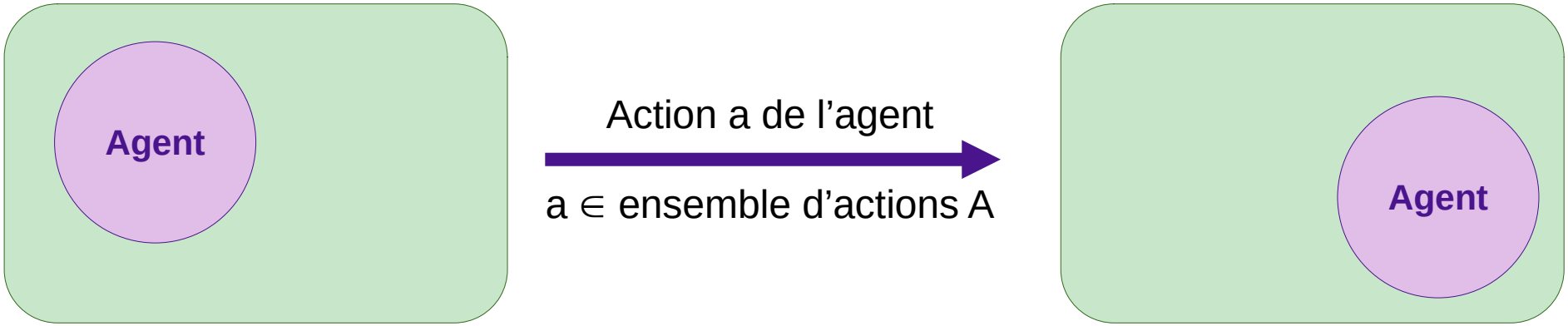
Un MDP :

- Ensemble d'état S
- Ensemble d'action A
- Fonction de transition T(s,a,s')
- Fonction de récompense R(s,a,s')

3. La fonction de transition T amène l'environnement d'un état s_t à un nouvel état s' ∈ S

Principes de l'apprentissage par renforcement – Le processus markovien

processus de décision markovien (MDP) :



Agent dans environnement
État s d'un ensemble d'état S

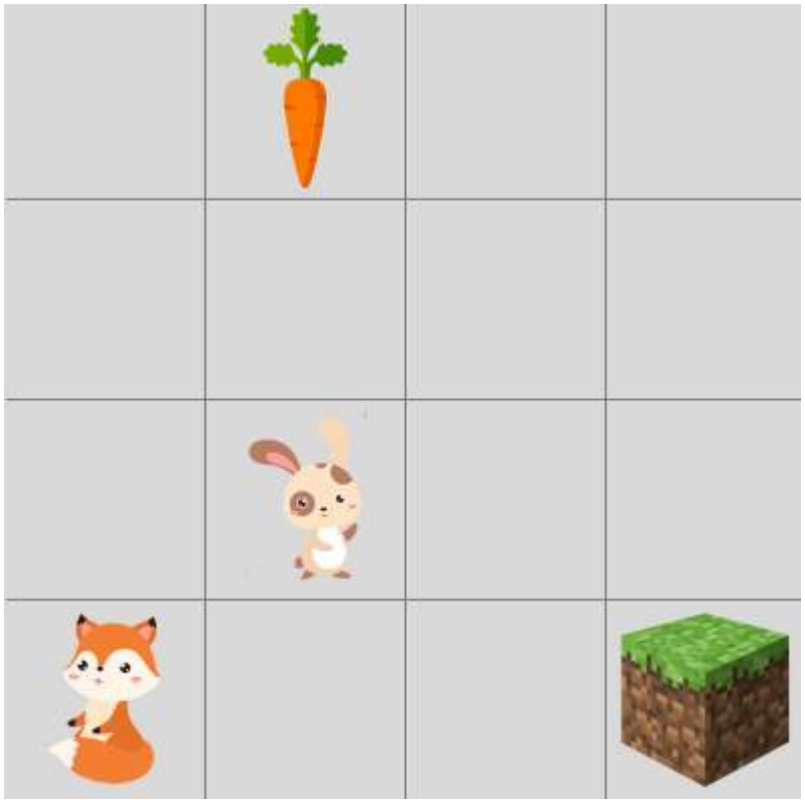
Agent dans environnement
État s' ∈ S

- Fonction de transition d'états : Distribution de probabilité de passer à s' si agent fait action a :

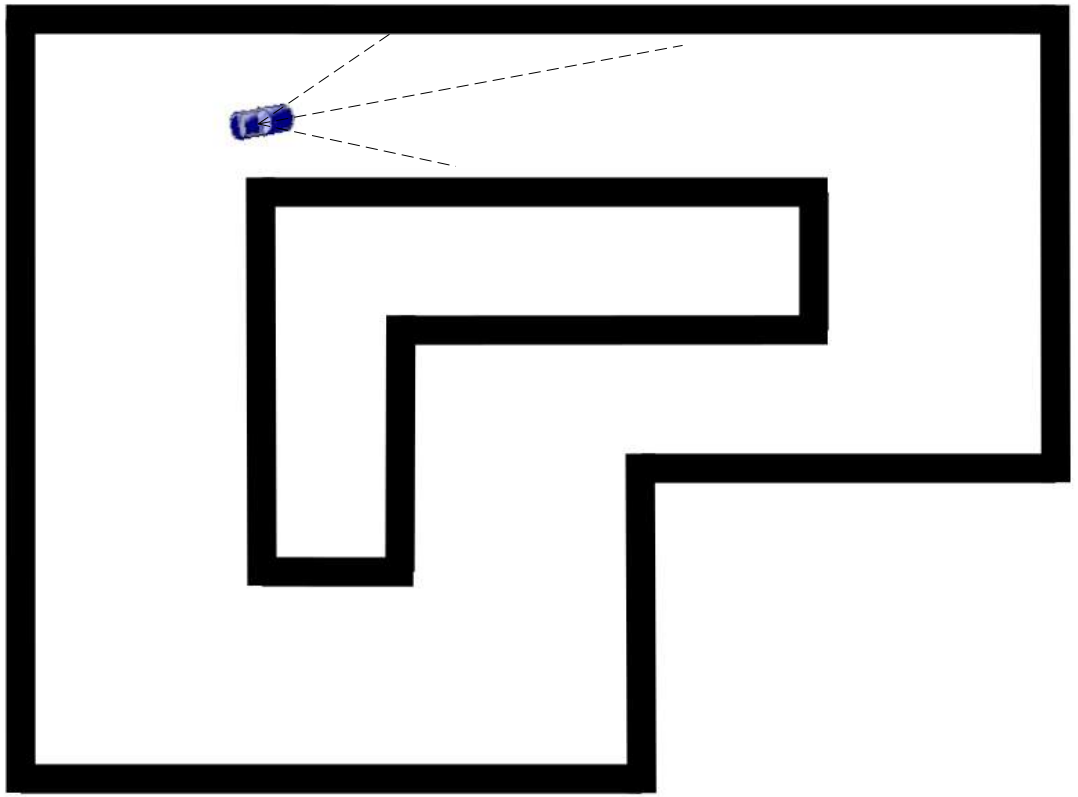
$T(s,a,s') = Pr(s_{t+1}=s' \mid s_t = s \text{ avec action } a)$

- Fonction de récompense R(s,a,s')
- Si on connaît s, l'historique n'a pas d'importance

Exemples



Frozen Lake



Conduite de voitures autonomes

jeu du morpion, pendule inversé, jeux Atari...

Principes de l’apprentissage par renforcement - méthodes

Objectif : Pour un problème donné (S, A, T, R) trouver une politique π permettant de maximiser la récompense.

Apprentissage par renforcement **en ligne** (simulation) ou hors ligne,

Apprentissage **sans modèle (model free)** ou basé sur un modèle

- Q-learning (apprentissage tabulaire)
- Deep Q-learning (approximation via des réseaux de neurones)

Q-learning

Une fonction Q (qualité) de valeur des états-actions :

$Q_\pi(s,a)$ représente le gain espéré par l'agent si

- il démarre à l'état s
- effectue l'action a,
- applique ensuite la politique π.

Qtable

	action a_0	action a_1	…	action a_n
état s_0	0,2	0,7		-1
état s_1	0,8	-0,2		-0,1
état s_2	-0,7	0,5		0,3
…				
état s_k	0,5	0,4		-0,4

Q-learning - Algorithme d'apprentissage

Initialiser Q[s,a]

Répéter un nombre N de fois

initialiser l'état s

répéter

choisir action a depuis s en utilisant Q et un peu d'aléatoire

exécuter l'action a

observer la récompense r et l'état s'

mettre à jour Q :

$Q[s, a] := Q[s, a] + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

s := s'

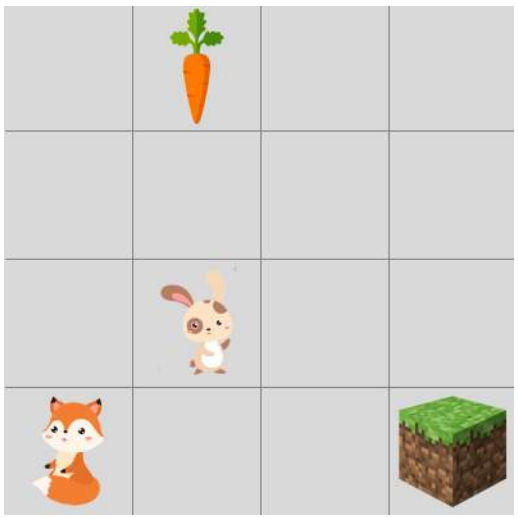
jusqu'à ce que s soit l'état terminal ou pour un nombre de boucles max

Exploitation – Exploration

Facteur d'actualisation

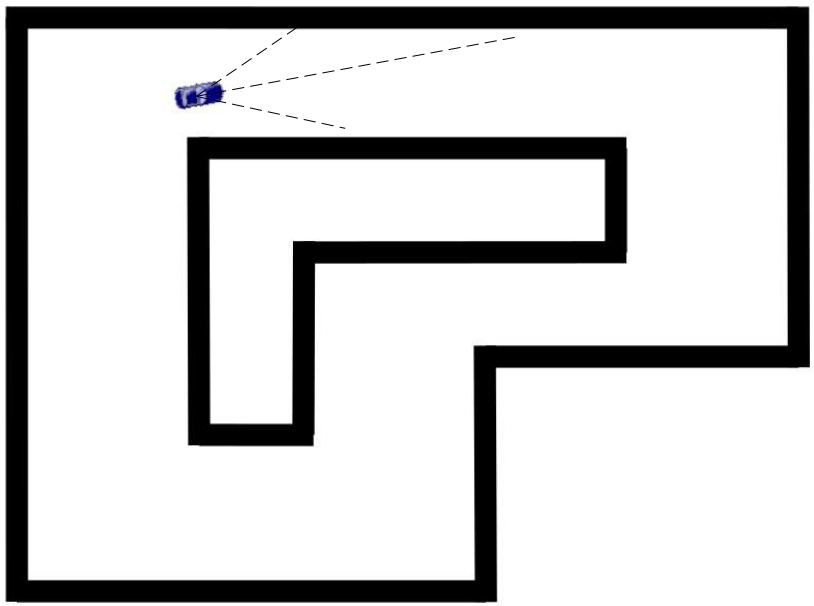
Facteur d'apprentissage (ou d'oubli)

Q-learning - l’atelier



D -1.71	D -1.71	D -0.98	D 0.00
H -1.82	H -1.69	H -0.98	H -0.85
G -1.69	G -1.69	G -0.85	G -0.85
B -1.96	B -0.85	B -0.85	B 8.50
D 4.20	D 8.30	D 9.97	D 0.00
H -1.69	H -0.85	H 0.00	H 0.00
G -1.69	G -1.00	G -0.85	G 0.00
B -1.71	B -0.85	B -0.85	B 0.00
D -1.69	D -0.85	D 0.00	D 0.00
H -1.67	H -1.69	H 0.00	H 0.00
G -0.85	G -0.85	G 0.00	G 0.00
B -8.50	B -0.85	B 0.00	B 0.00
D 0.00	D 0.00	D 0.00	D 0.00
H 0.00	H -0.85	H 0.00	H 0.00
G 0.00	G 0.00	G 0.00	G 0.00
B 0.00	B 0.00	B 0.00	B 0.00

Qtable de Frozen Lake



Frozen Lake

L’environnement, S, A sont donnés
Choisir R
Implanter l’algorithme de Q-learning
Regarder l’influence des hyper-paramètres

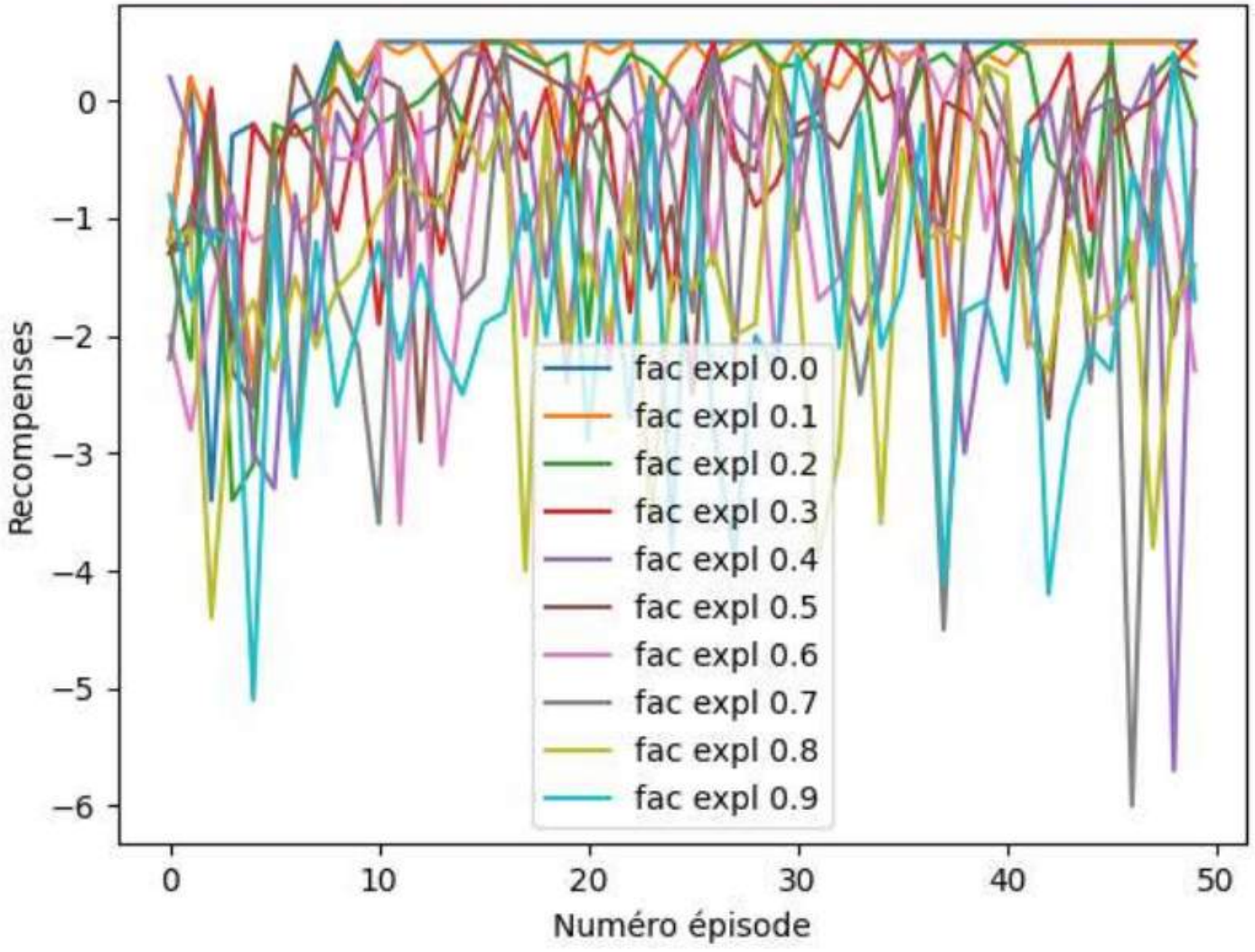
Conduite de voitures autonomes

Même travail
Résoudre le problème lié au sur-apprentissage

Les codes source sont disponibles sur Culture Sciences de l’ingénieur

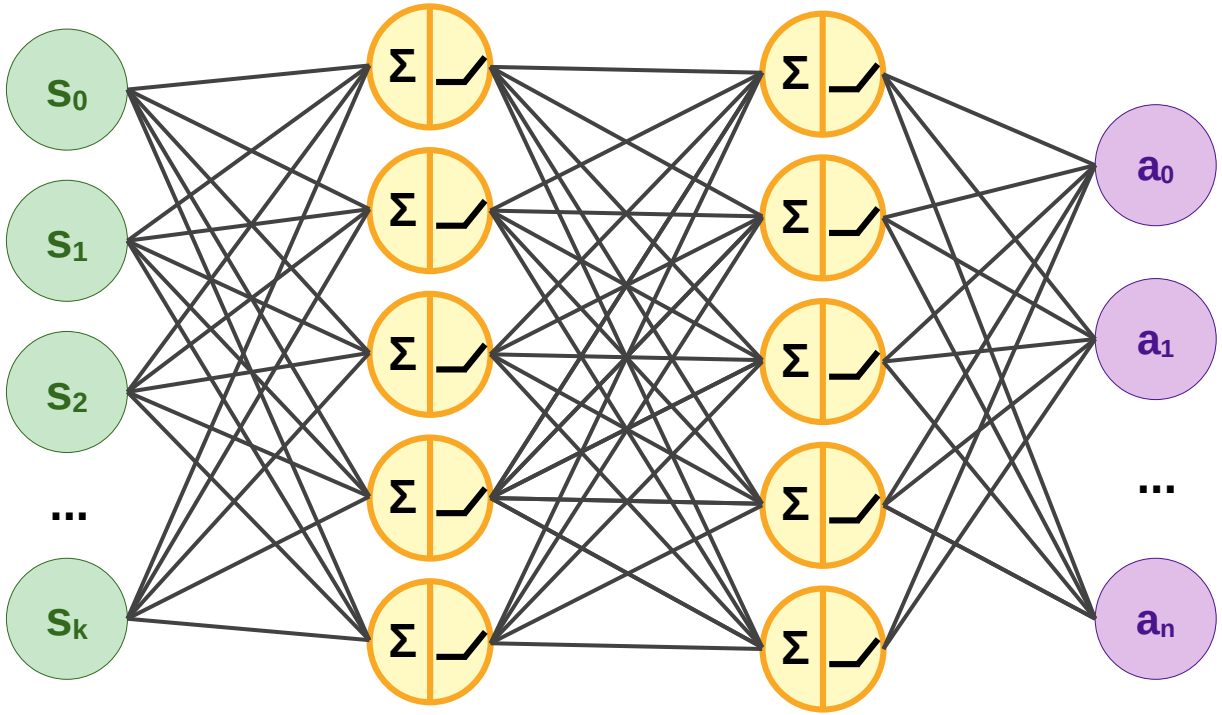
Influence des hyperparamètres

Exemple : Cumul des récompenses en fonction du facteur d'exploration



Deep Q-Learning

On remplace la Q-table par un réseau de neurones.
L'apprentissage vise alors à optimiser les paramètres du réseau de neurones (les poids rangé dans une matrice W). Q(s,a) devient Q(s,a;W).



États en entrées

Actions en sortie

Deep Q-Learning

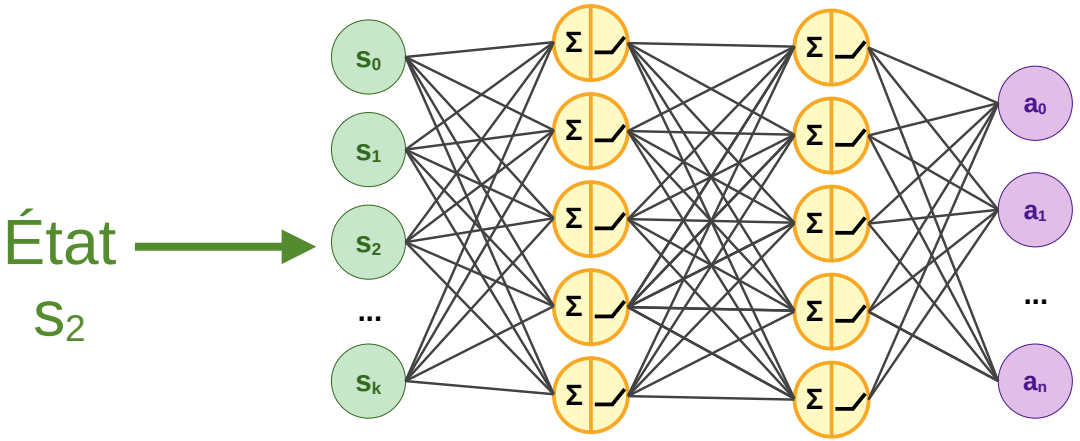
Q-table issu du Q-apprentissage

État s_2 →

	action a_0	action a_1	…	action a_n
état s_0	0,2	0,7		-1
état s_1	0,8	-0,2		-0,1
état s_2	-0,7	0,5		0,3
…				
état s_k	0,5	0,4		-0,4

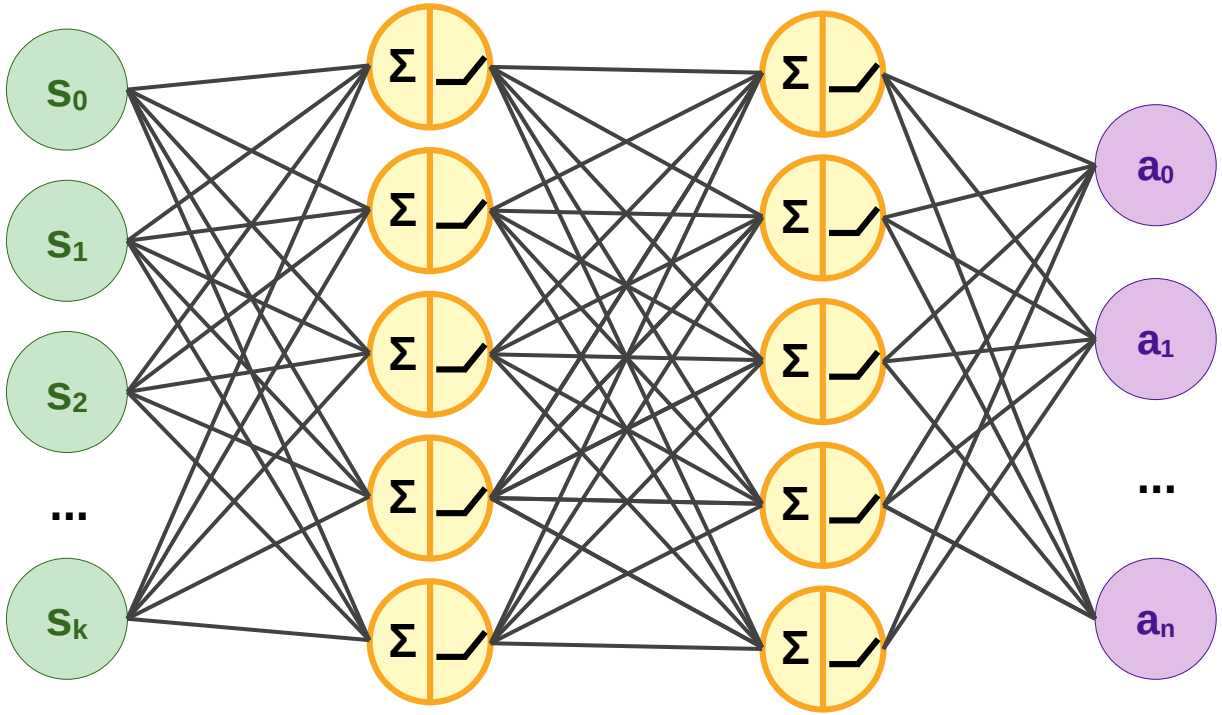
→ Action a_1

Réseau de neurones issu du Q-apprentissage profond



Deep Q-Learning

On remplace la Q-table par un réseau de neurones.
L'apprentissage vise alors à optimiser les paramètres du réseau de neurones (les poids rangé dans une matrice W). Q(s,a) devient Q(s,a;W).



États en entrées

Actions en sortie

Deep Q-Learning

Contrairement à l'apprentissage supervisé, il n'y a pas de données étiquetées pour entraîner le réseau.

Comme pour le Q-learning, l'objectif est de d'indiquer pour chaque état s la meilleure action a à effectuer.
Pour cela, Q donne pour chaque état s, la somme des récompenses que l'on peut espérer en exécutant a :

- r la récompense obtenue en exécutant a
- la somme maximale des récompenses que l'on obtiendra depuis l'état s' où nous mène l'action a (avec y pour assurer la convergence) :

$$r + \gamma \max_{a'} Q(s', a')$$

Deep Q-Learning

On rappelle l'équation du Q-learning pour optimiser les paramètres de la table Q :

$$Q[s, a] := Q[s, a] + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

S'inspirant du Q-learning, on optimise les paramètres du réseau (la matrice des poids W) avec une fonction coût L reprenant, pour chaque pas i :

$$L_i(W_i) = \left[\left(r + \gamma \max_{a'} Q(s', a'; W_i) - Q(s, a; W_i) \right)^2 \right]$$

L'algorithme d'optimisation du réseau (descente de gradient notamment) va ainsi rapprocher la valeur Q(s,a;W) de la récompense maximale que l'on peut espérer en exécutant a depuis l'état s : $r + \gamma \max_{a'} Q(s', a')$

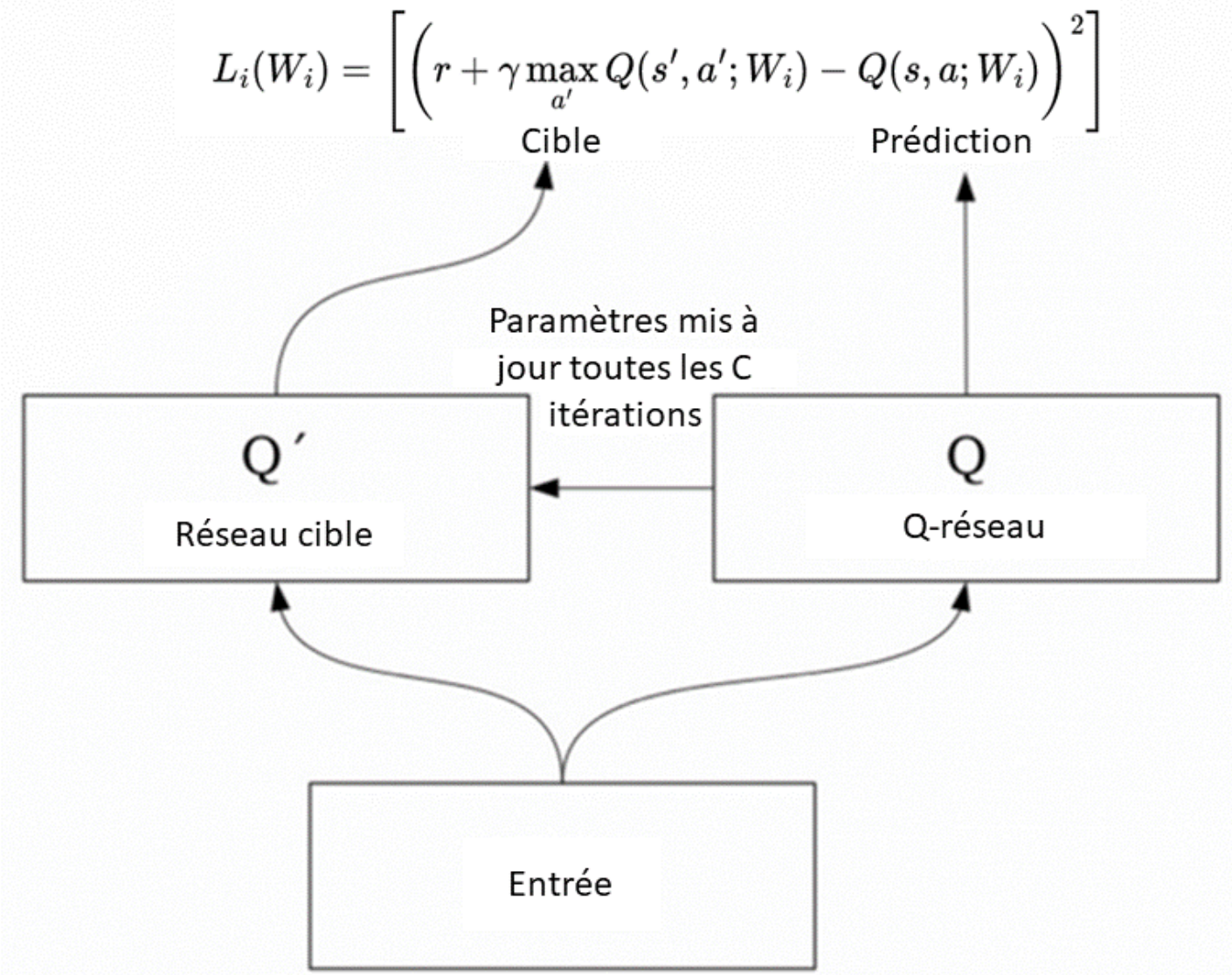
Le taux d'apprentissage α se retrouve dans la mise à jour des poids.

$w_{(i+1),j} = w_{i,j} - \alpha * \text{gradient}$

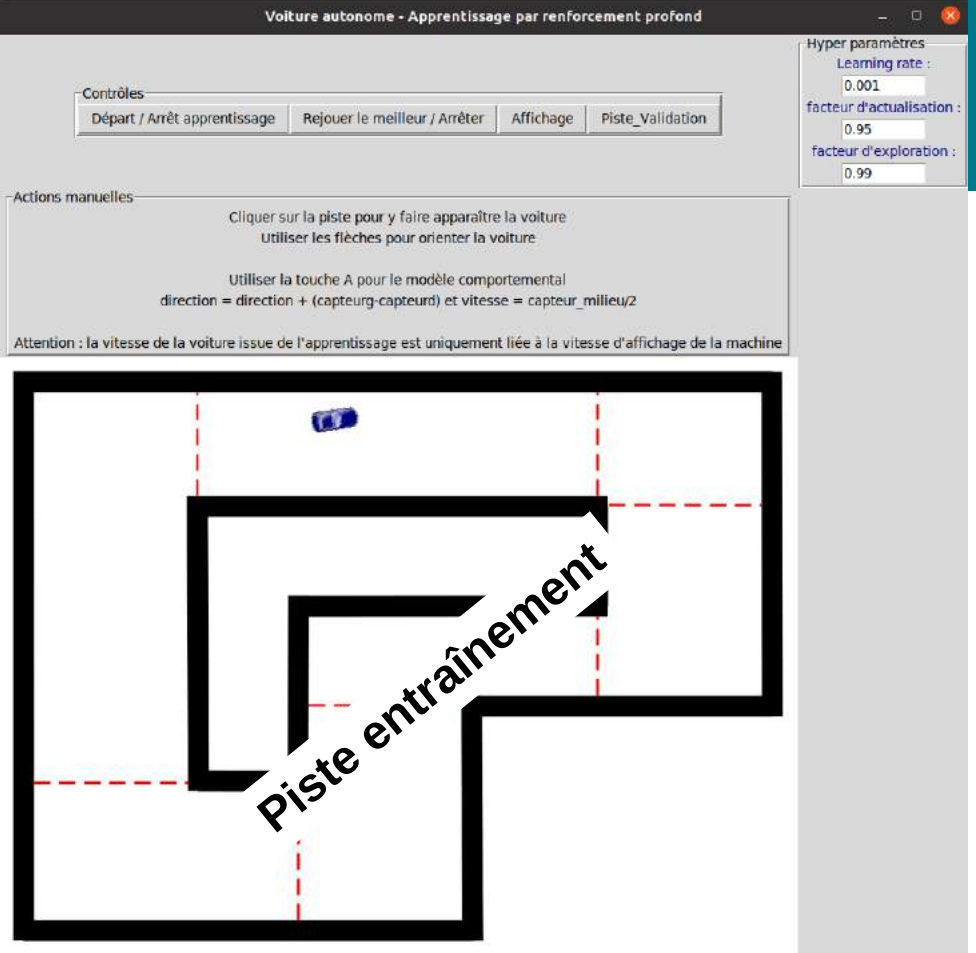
Deep Q-Learning

Pour stabiliser l'apprentissage, on évite de modifier trop rapidement la cible.

Pour cela, on utilise une copie du réseau Q, nommé Q' ou Qcible que l'on met à jour à un nombre de pas fixe (chaque 20 pas par exemple).

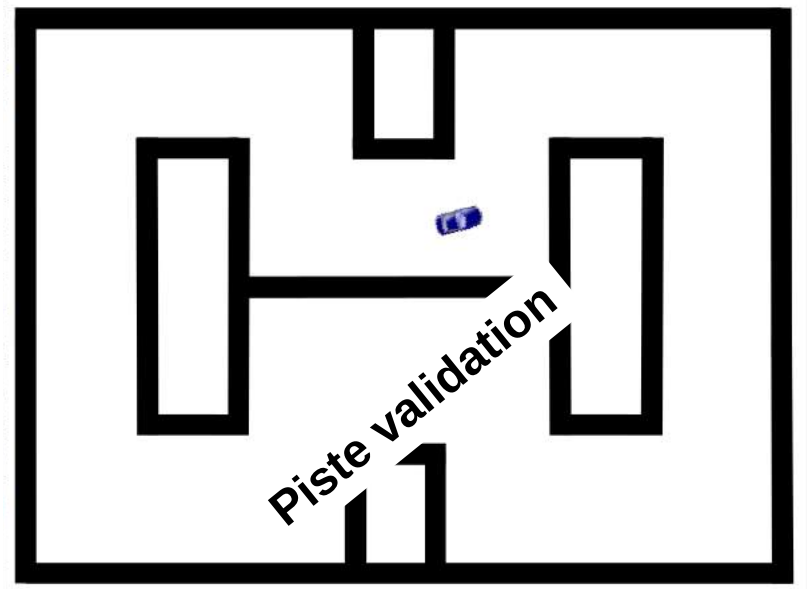


Deep Q-Learning l'atelier



Environnement : la voiture et la piste

Agent : le « conducteur »

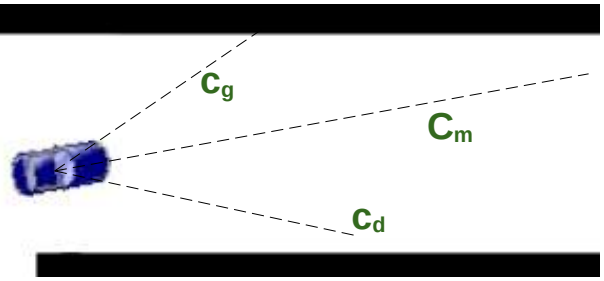


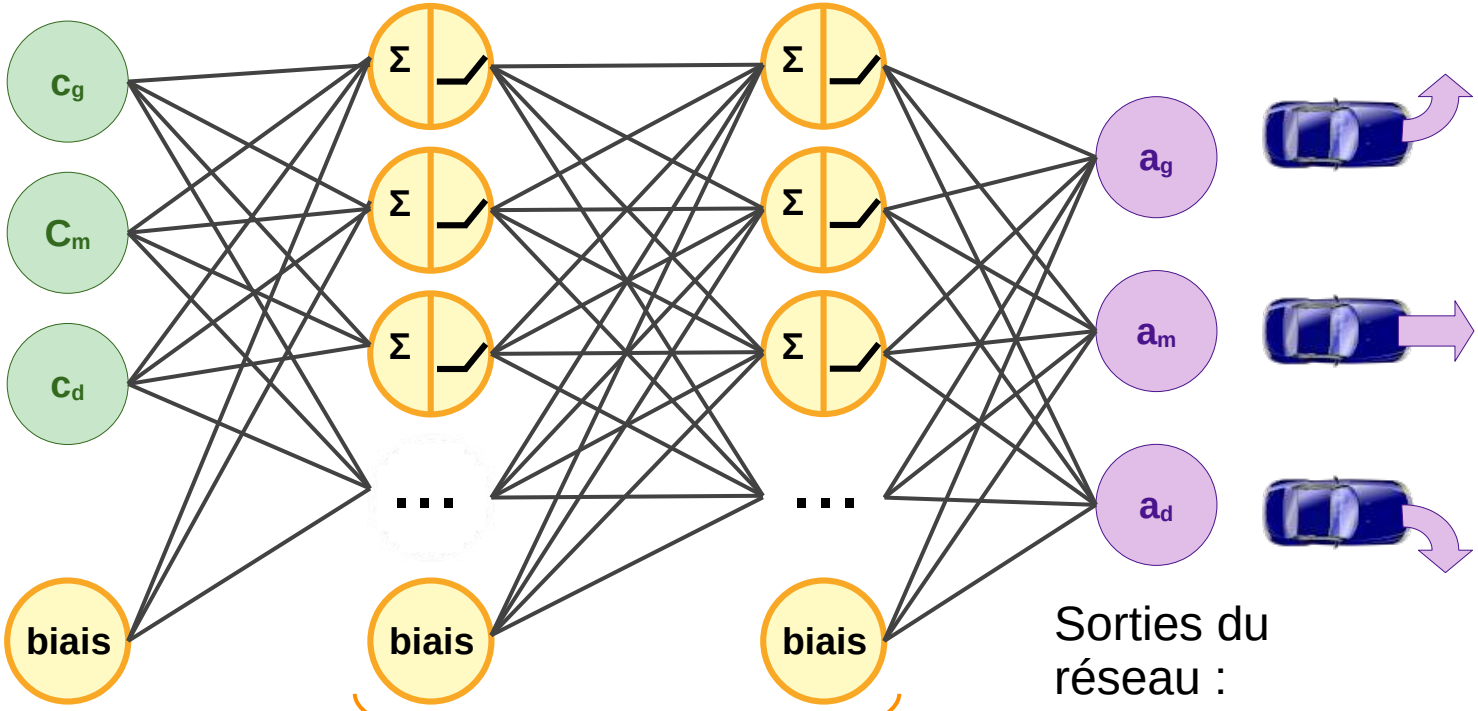
La touche 'A' donne un exemple d'IA symbolique, avec un algorithme comportemental.
On remplace la Q-table par un réseau de neurones à 2 couches cachées de 24 neurones
Démarrage aléatoire pour éviter le sur-apprentissage
Importance de la diversité des situations du circuit
La validation se fait sur une piste différente de la piste d'essai.

Les codes source sont disponibles sur Culture Sciences de l'ingénieur

Deep Q-Learning : l'atelier

Après chaque épisode, on optimise les paramètres du réseau.





Entrées du réseau : les 3 capteurs

2 couches cachées de 24 neurones
« Fully connected »
Avec fonction d'activation ReLu
4*24+25*24+25*3 = 771 poids

Sorties du réseau : les 3 actions (gauche, tout droit, droite)

Deep Q-Learning : l'atelier

La fonction d'activation et sa dérivée

```python
def relu(mat):
    return np.multiply(mat,(mat>0))

def relu_derivative(mat):
    return (mat>0)*1
```

La classe NNLayer pour une couche de neurones, avec les méthodes forward et backward

```python
# classe décrivant une couche de réseau de neurones
class NNLayer:
    # initialisation : nombres d'entrées, nombre de neurones (sorties), fonction d'activation
    def __init__(self, input_size, output_size, activation=None, lr = 0.001):
        self.input_size = input_size
        self.output_size = output_size
        #les poids sont initialisés avec une valeur aléatoire
        self.weights = np.random.uniform(low=-0.5, high=0.5, size=(input_size, output_size))
        self.activation_function = activation
        self.lr = lr
# Calcul des sorties d'une couche à partir des entrées (mode "forward")
def forward(self, inputs, remember_for_backprop=True):

#mise à jour des poids avec le gradient, pondéré par le taux d'apprentissage
def update_weights(self, gradient):
    self.weights = self.weights - self.lr*gradient
#mise à jour de la couche de neurones à partir du gradient venant de la couche suivante
def backward(self, gradient_from_above):
```

Deep Q-Learning : l'atelier

La classe RLAgent est celle dont l'instance, nommée model, sera l'agent de notre problème, chargé de prendre les décisions à partir d'un réseau de neurones et de l'optimiser.

```
#classe décrivant l'agent avant comme Q-fonction un réseau de neurones pour prendre ses décisions
class
    #calcul de la sortie du réseau cible. La mémorisation sera désactivée à l'usage
    def forward_target(self, state, remember_for_backprop=True):
    env = None

    #création de l'agent et de son réseau de neurones avec 3 entrées, 2 couches cachées de 24 neurones
    def __init__(self, env):
        self.env = ma_voiture
        self.hidden_size = 24
        self.input_size = 3
        self.output_size = 3
        self.num_hidden_layers = 2
        self.epsilon = 1.0
        self.gamma = 0.95 #taux d'actualisation par défaut

        #CREATION DU RESEAU DE NEURONES PRINCIPAL
        ...
        #CREATION DU RESEAU DE NEURONES CIBLE UTILISE POUR L'OPTIMISATION, à l'identique du principal
        ...
```

On retrouve dans l'initialisation les 2 réseaux identiques Q (main_NN) et Qcible (target_NN)

La méthode select_action est celle qui permet d'obtenir a depuis s. Le taux d'exploration est réglé par l'attribut epsilon, décroissant au fil des pas

```
#exploration ou application de la politique pour choisir une action
def select_action(self, state):
```

Deep Q-Learning : l'atelier

La méthode train de la classe RLAgent gère l'apprentissage. Elle utilise :

- les 2 méthodes forward (pour le réseau principal et le réseau cible) pour les calculs des sorties connaissant les entrées,
- la méthode bacward pour l'optimisation des paramètres par rétropopagation de gradient

```
#entrainement à partir de a, s',s, r : calcul des valeurs d'action données par Q et des
def train(self, done, action, new_state, state, reward):
    action_values = self.forward(state, remember_for_backprop=True)
    next_action_values = self.forward_target(new_state, remember_for_backprop=False)
    experimental_values = np.copy(action_values)
    #application de la formule du deep Q-learning avec -100 si crash ou reward sinon.
    if done:
            experimental_values[action] = -100
    else:
            experimental_values[action] = reward + self.gamma*np.max(next_action_values)

    #Mise à jour des poids par la propagation du gradient vers les couches amont
    self.backward(action_values, experimental_values)
#calcul de la sortie du réseau de neurones en calculant couche après couche,
def forward(self, state, remember_for_backprop=True):
#calcul de la sortie du réseau cible. La mémorisation sera désactivée à l'usage
def forward_target(self, state, remember_for_backprop=True):
#Propagagation du gradient vers les couches amont, mise à jour des poids
def backward(self, calculated_values, target_values):
```

L'algorithme est très similaire à celui du Q-learning

```
# The main program loop
for i_episode in range(NUM_EPISODES):
    #positionnement aléatoire de la voiture au départ
    ma_voiture.reset()
    state = ma_voiture.lecture_capteurs(fenetre.image_piste)
    fenetre.maj_affichage()
    cumul_reward = 0
    index_recopie_target_NN = 0

    # On commence un pas
    while True :
        index_recopie_target_NN +=1
        #on choisit une action en utilisant la politique ou l'exploration (c'est pris
        action = model.select_action(state)
        #on fait un pas et on cumule la récompense
        new_state, reward, done, info = ma_voiture.step(action,fenetre.image_piste)
        cumul_reward += reward

        # On entraine le réseau avec ce pas
        model.train(done, action, new_state, state, reward)
        state = new_state

        #A chaque nombre de pas fixé, on recopie le réseau principal dans le réseau cible
        if (index_recopie_target_NN %20 == 0) : #mise à jour de la NN_target
            for i in range(model.num_hidden_layers+1) :
                model.target_NN[i].weights = np.copy(model.main_NN[i].weights)

        if done:
            #on met à jour la meilleure récompense total et le meilleur réseau
            if cumul_reward > cumul_reward_best :
                for i in range(model.num_hidden_layers+1) :
                    model_best.main_NN[i].weights = np.copy(model.main_NN[i].weights)
                cumul_reward_best = cumul_reward
            break
```

initialisations

Choix d'une action, avec exploration (intégrée par *select_action*)

Exécution d'un pas

Optimisation du réseau Q avec Q_{cible}

Chaque 20 pas, on met à jour Q_{cible} (target_NN)

Si la récompense est très bonne, on garde une copie du réseau.

école normale supérieure paris-saclay

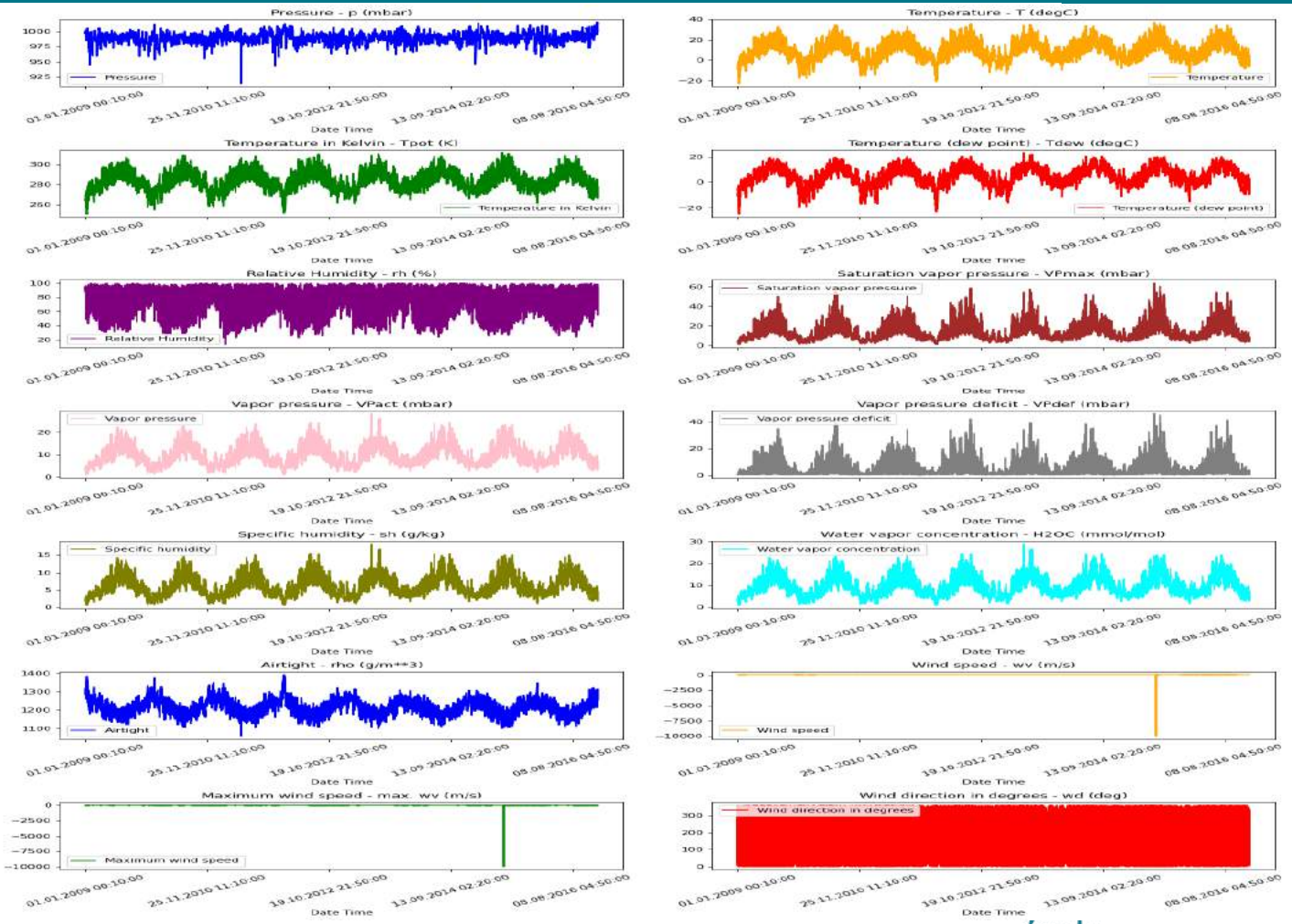
Séries temporelles et Réseaux de neurones récurrents

université PARIS-SACLAY

Séries temporelles et apprentissage par réseaux de neurones récurrents

- Des données statiques (i.e images) aux données dynamiques (i.e vidéos)

- Séries temporelles : Données échantillonnées en fonction d'une dimension temporelle avec un ordre temporel implicite.



Séries temporelles : caractéristiques météorologiques de 2009 à 2016

Séries temporelles et apprentissage par réseaux de neurones récurrents

Extraction des caractéristiques globales des sets de données :

Saisonnalité : affiche des modèles périodiques se répétant à une fréquence constante.
Tendance : une valeur croissante indique une tendance positive et une valeur décroissante, une tendance négative.
Reste : Après avoir extrait la tendance et la saisonnalité des données, ce qui reste est ce que nous appelons le reste (erreur) ou le résidu. Cela permet de détecter les anomalies dans les séries chronologiques.
Cycle : Les données de séries temporelles sont dites cycliques lorsqu'il existe des tendances sans répétitions fixes ou saisonnalité.
Stationnarité : Les données de séries temporelles sont stationnaires lorsque leurs caractéristiques statistiques ne changent pas dans le temps

Caractéristiques spécifiques des sets de données :

Dépend du set de données: par exemple, pour les prévisions météorologiques, les caractéristiques peuvent être la température, l'humidité relative, spécifique, etc.

Est-ce que le plus de caractéristiques nous avons, meilleure sera notre prédiction?

Approches algorithmiques pour la prévision de séries temporelles:

1) Préparation des données dynamiques
   a) Définir les informations disponibles
   b) Période pendant laquelle nous avons besoin de valeurs prévisionnelles.
2) Définir le modèle à utiliser:
   a) RNNs:
      i) Fully Recurrent Neural Network (FRNN)
      ii) Echo State Network (ESN)
      iii) Long-Short Term Memory (LSTM)
      iv) Gated Recurrent Unit (GRU)

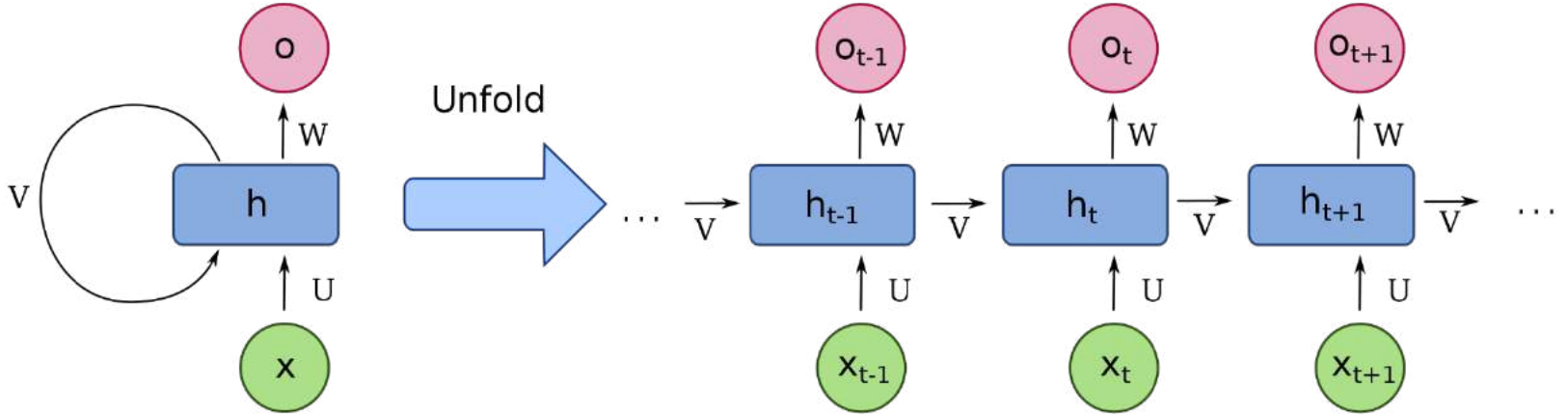


Schéma d'un réseau de neurones récurrent: FRNN, *Wikipédia*

Séries temporelles et apprentissage par réseaux de neurones récurrents

Problème de disparition de gradient:

- Empêche de modifier leurs poids en fonction d'événements antérieurs.
- Erreur ne se propage plus jusqu'aux pas les plus antérieurs.
- Solution: LSTM et GRU:
  - un état caché + une cellule "mémoire"
  - porte d'entrée: accepte ou non la modification de la cellule
  - porte de sortie: donne l'état de sortie de la cellule au LSTM
  - Que LSTM: porte d'oubli: réinitialisation de la cellule

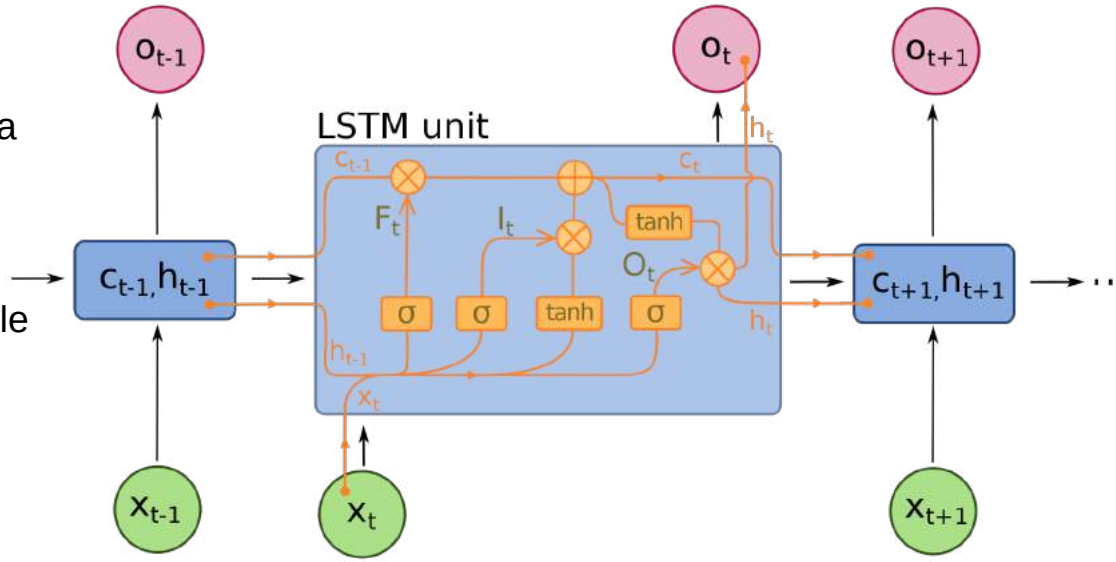


Schéma d'un LSTM, *Wikipédia*.

Application:

Google Collaboratory utilisant la bibliothèque Keras:
https://keras.io/examples/timeseries/timeseries_weather_forecasting/#prediction

Quelques références

Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press, 2016.

Artificial Intelligence: A modern approach, 2nd Ed., Dr. Afşar Saranli and Stuart Russel and Peter Norvig, 2010.

Intelligence artificielle : triomphes et déceptions, Melanie Mitchell, postface de Douglas Hofstadter, traduit de l'anglais (Etats-Unis) par Christian Jeanmougin, Dunod, 2021

Neural Network From Scratch : https://nnfs.io/

Chaine “3 Blue 1 Brown” : https://youtu.be/tIeHLnjs5U8

Quelques outils

- Google colab : pas besoin d'installer des paquets, connaît les liens vers les classiques, matplotlib fonctionne. Plus compliqué pour afficher une vidéo. Pratique quand on ne connaît pas les étudiants.
- Matlab : pas besoin de coder, interfaces matériels avec les systèmes (Dspace ou I/O plus simples). Coût du logiciel. Bibliothèques moins riches, communauté moins grande.
- Python : grande communauté - bibliothèques

Un exemple de reconnaissance des chiffres MNIST avec MLP :
https://colab.research.google.com/github/trekhleb/machine-learning-experiments/blob/master/experiments/digits_recognition_mlp/digits_recognition_mlp.ipynb