Data Mining et Machine Learning

John Samuel

CPE Lyon

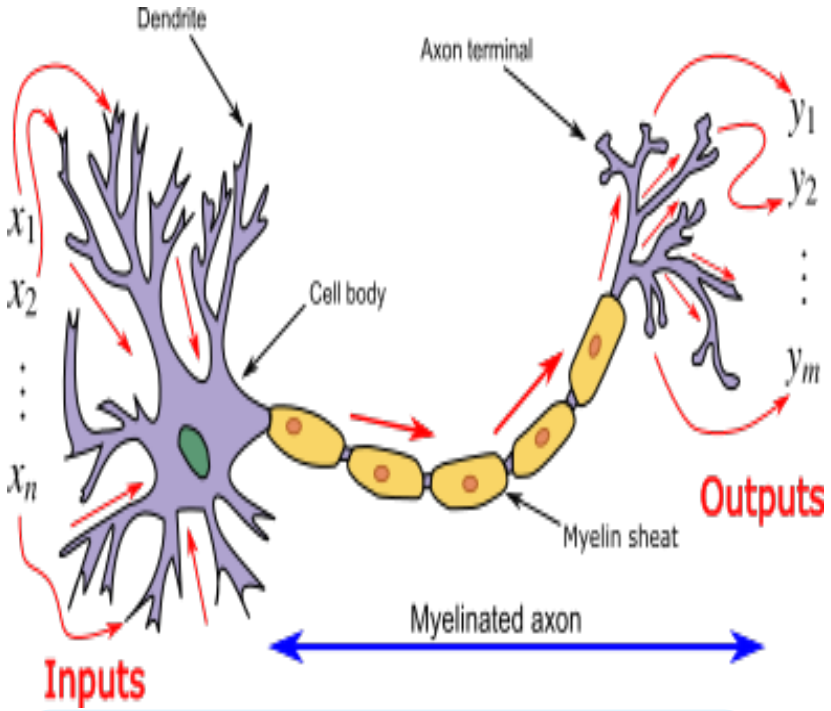
Année: 2024-2025

Courriel: john.samuel@cpe.fr

Objectifs

1. Apprentissage machine
2. Apprentissage profond
3. Apprentissage par renforcement
4. Licences de données, éthique et vie privée

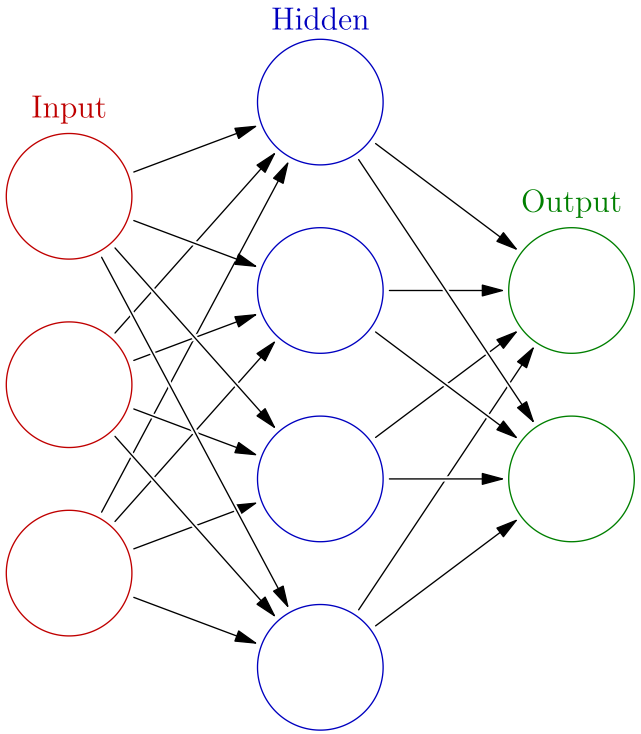
Neurones biologiques



Neurone biologique[1]

1. https://en.wikipedia.org/wiki/File:Neuron3.png

Introduction



Réseaux de neurones artificiels

3.1. Apprentissage machine

Réseau de neurones

Les réseaux de neurones sont couramment utilisés dans le domaine de l'apprentissage machine, en particulier dans des tâches telles que la classification, la régression, la reconnaissance d'images, le traitement du langage naturel, et bien d'autres. Un réseau de neurones artificiels est une collection d'unités interconnectées appelées neurones artificiels. Ces réseaux sont inspirés de la structure du cerveau biologique

- **Connexions** : Chaque connexion entre les neurones, similaire aux synapses dans le cerveau biologique, peut transmettre un signal aux autres neurones.
- **Transmission de signal** : Un neurone artificiel reçoit un signal, le traite à l'aide d'une fonction non linéaire, et peut ensuite transmettre un signal aux neurones qui lui sont connectés.
- **Fonction d'activation** : La sortie de chaque neurone est calculée par une fonction non linéaire appliquée à la somme pondérée de ses entrées. Cette fonction d'activation introduit une non-linéarité dans le réseau, permettant de modéliser des relations complexes.

Réseau de neurones

- **Poids ajustables** : Les neurones et les connexions ont généralement des poids qui sont ajustés au fur et à mesure de l'apprentissage. Ces poids déterminent l'importance relative des différentes entrées pour chaque neurone.
- **Ajustement des poids** : Les poids peuvent être ajustés pour augmenter ou diminuer la force du signal au niveau d'une connexion, influençant ainsi la contribution de cette connexion aux calculs du réseau.
- **Seuil** : Les neurones peuvent avoir un seuil, de sorte qu'un signal n'est envoyé que si la somme pondérée de ses entrées dépasse ce seuil. Cela permet au réseau de moduler sa sensibilité aux entrées.

Les couches

Les neurones sont organisés en couches. Il existe généralement trois types de couches dans un réseau de neurones :

- **Couche d'Entrée (Input Layer)** : Cette couche reçoit les signaux initiaux ou les données en entrée. Chaque neurone dans cette couche représente une caractéristique ou une variable d'entrée.
- **Couches Cachées (Hidden Layers)** : Ces couches effectuent des transformations non linéaires sur les entrées. Elles sont responsables de l'extraction et de la représentation des caractéristiques importantes des données. Un réseau de neurones peut avoir une ou plusieurs couches cachées.
- **Couche de Sortie (Output Layer)** : Cette couche génère la sortie du réseau. Le nombre de neurones dans cette couche dépend de la nature de la tâche, par exemple, une classification binaire aurait un neurone de sortie, tandis qu'une classification multi-classes en aurait plusieurs.

Les couches

- **Transformations** : Chaque couche, y compris la couche d'entrée, effectue des transformations sur les signaux qu'elle reçoit. Ces transformations sont déterminées par les poids des connexions entre les neurones.
- **Propagation des signaux** : Les signaux passent de la première couche (l'entrée) à la dernière couche (la sortie) à travers les connexions pondérées entre les neurones. Ce processus est souvent appelé la propagation avant (forward propagation). Pendant l'apprentissage, la rétropropagation (backpropagation) est utilisée pour ajuster les poids afin de minimiser l'erreur de prédiction.
- **Architecture** : La manière dont les couches sont organisées et connectées dans le réseau constitue son architecture. Les réseaux de neurones peuvent avoir des architectures diverses, y compris des réseaux profonds (avec de nombreuses couches cachées) ou des architectures plus simples.

L'entraînement

L'objectif global de l'entraînement est d'ajuster les poids du réseau de manière à ce qu'il puisse généraliser à de nouvelles données, produisant des résultats précis pour des exemples qu'il n'a pas vu pendant l'entraînement.

- **Données d'entraînement** : Les réseaux neuronaux apprennent à partir d'exemples. Chaque exemple se compose d'une "entrée" (les caractéristiques) et d'un "résultat" connu (l'étiquette ou la sortie attendue).
- **Calcul de l'erreur** : Lorsque le réseau produit une sortie pour une entrée donnée, l'erreur est calculée en comparant cette sortie à la sortie cible (le résultat connu). Il existe différentes mesures d'erreur, mais la somme des carrés des différences (Mean Squared Error, MSE) est couramment utilisée.

L'entraînement

- **Rétropropagation (Backpropagation)** : Le réseau ajuste ses poids en utilisant la rétropropagation. Cette technique minimise l'erreur en modifiant les poids à partir de la couche de sortie jusqu'à la couche d'entrée. La règle de la chaîne du calcul différentiel est appliquée pour propager l'erreur à travers le réseau.
- **Descente de gradient** : La règle d'apprentissage souvent utilisée pour ajuster les poids est la descente de gradient. Elle utilise le gradient de l'erreur par rapport aux poids pour mettre à jour les poids dans la direction qui minimise l'erreur.
- **Itérations** : Le processus d'ajustement des poids en fonction de l'erreur est répété pour de nombreux exemples du jeu de données d'entraînement. Chaque itération est appelée une "époque". Plusieurs époques peuvent être nécessaires pour que le réseau converge vers un état où l'erreur est suffisamment basse.
- **Optimisation** : Différentes techniques d'optimisation peuvent être utilisées pour améliorer la convergence du réseau, telles que l'ajustement adaptatif du taux d'apprentissage.

Composants des réseaux de neurones artificiels

- **Neurones** : Les neurones artificiels sont les unités de base d'un réseau de neurones. Chaque neurone reçoit des signaux d'entrée, effectue un calcul sur ces signaux à l'aide d'une fonction d'activation, et produit une sortie. Les neurones sont organisés en couches, à savoir la couche d'entrée, les couches cachées, et la couche de sortie.
- **Connexions et Poids** : Les connexions entre les neurones sont représentées par des poids. Chaque connexion a un poids associé, qui détermine l'importance relative de cette connexion dans le calcul du neurone de sortie. Pendant l'entraînement, ces poids sont ajustés pour minimiser l'erreur de prédiction du réseau.
- **Fonction de Propagation (Propagation avant)** : La fonction de propagation, également appelée propagation avant, décrit le processus par lequel les signaux se propagent à travers le réseau depuis la couche d'entrée jusqu'à la couche de sortie. Chaque neurone effectue une transformation sur les signaux qu'il reçoit, et ces signaux modifiés sont transmis aux neurones de la couche suivante.

Composants des réseaux de neurones artificiels

Neurones

Chaque neurone artificiel a des entrées, qui peuvent être les valeurs caractéristiques d'un échantillon de données externe, et produit une seule sortie. Cette sortie peut être envoyée à plusieurs autres neurones, formant ainsi la structure interconnectée du réseau neuronal. La **fonction d'activation** joue un rôle crucial dans le calcul de la sortie d'un neurone. Le processus comprend les étapes suivantes :

- **Somme pondérée** : Pour trouver la sortie du neurone, on prend la somme pondérée de tous les intrants (entrées). Chaque entrée est multipliée par le poids correspondant à la connexion.

Composants des réseaux de neurones artificiels

Neurones

- **Ajout d'un terme de biais** : Un terme de biais est ajouté à la somme pondérée. Le terme de biais est un paramètre supplémentaire qui permet au modèle d'apprendre un décalage ou une translation.
- **Activation** : La somme pondérée, parfois appelée activation, est ensuite passée par une fonction d'activation. Cette fonction est généralement non linéaire et introduit de la complexité dans le modèle, permettant au réseau de capturer des relations non linéaires dans les données

Composants des réseaux de neurones artificiels

Connexions et poids: Le réseau de neurones est constitué de connexions, où chaque connexion transmet la sortie d'un neurone comme entrée à un autre neurone. Chaque connexion possède un poids qui représente son importance relative dans la transmission du signal.

- Un neurone donné peut avoir **plusieurs connexions d'entrée**, recevant des signaux de différents neurones, et plusieurs connexions de sortie, transmettant des signaux à d'autres neurones. Les poids associés à ces connexions permettent au réseau de moduler l'influence de chaque neurone sur les autres, ajustant ainsi la force et la direction des signaux transmis à travers le réseau.
- Cette structure de connexion et de pondération est fondamentale dans le fonctionnement des réseaux de neurones, car elle permet au réseau d'apprendre des représentations complexes des données et d'ajuster ses paramètres pendant l'entraînement pour accomplir des tâches spécifiques.

Composants des réseaux de neurones artificiels

Fonction de propagation

Calcul de l'entrée d'un neurone : La fonction de propagation calcule l'entrée d'un neurone en prenant la somme pondérée des sorties de ses prédécesseurs, où chaque sortie est multipliée par le poids de la connexion correspondante. Cela peut être représenté mathématiquement comme suit :

$$\text{Entrée du Neurone} = \sum_{i=1}^{n} (\text{Sortie du Prédécesseur}_i \times \text{Poids}_i)$$

où n est le nombre de connexions d'entrée.

Composants des réseaux de neurones artificiels

Fonction de propagation

Ajout d'un terme de biais : Un terme de biais peut être ajouté au résultat de la propagation. Le terme de biais est un paramètre supplémentaire, souvent représenté par b dans les équations, qui permet au modèle d'apprendre un décalage ou une translation. Cela donne la forme finale de l'entrée du neurone :

$$\text{Entrée du Neurone} = \sum_{i=1}^{n} (\text{Sortie du Prédécesseur}_i \times \text{Poids}_i) + \text{Biais}$$

Composants des réseaux de neurones artificiels

Fonction de propagation

Fonction d'Activation : Après avoir calculé l'entrée du neurone, celle-ci est passée à travers une fonction d'activation. Cette fonction introduit une non-linéarité dans le modèle, permettant au réseau de neurones de capturer des relations complexes et d'apprendre des modèles non linéaires. Certaines des fonctions d'activation couramment utilisées comprennent :

- **Sigmoïde** : $\sigma(x) = \frac{1}{1+e^{-x}}$
- **Tangente hyperbolique (tanh)** : $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- **ReLU (Rectified Linear Unit)** : $\mathrm{ReLU}(x) = \max(0, x)$
- **Softmax** (pour la couche de sortie dans la classification) : $\mathrm{Softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$

Perceptron

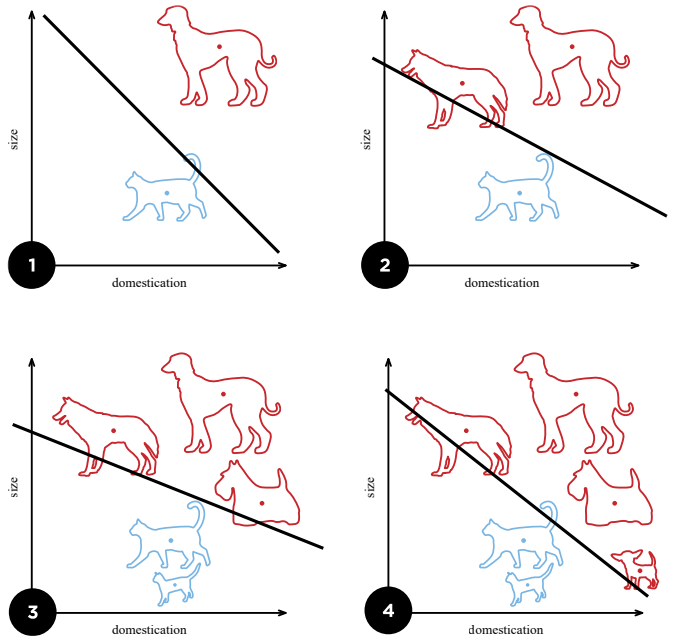
Le perceptron est un **algorithme d'apprentissage supervisé** utilisé pour la **classification binaire**. Il est conçu pour résoudre des problèmes où l'objectif est de déterminer si une entrée donnée appartient ou non à une classe particulière.

- Le perceptron a été inventé par **Frank Rosenblatt** en 1958. L'idée était de créer un modèle simple de neurone artificiel inspiré du fonctionnement des neurones biologiques. Rosenblatt a formulé un algorithme d'apprentissage qui permet au perceptron d'ajuster ses poids en fonction des erreurs de classification, améliorant ainsi ses performances au fil du temps.

Perceptron

- **Fonctionnement** : Le perceptron prend plusieurs entrées pondérées et les combine en une somme. Ensuite, cette somme est soumise à une fonction d'activation, généralement une fonction échelon (step function), qui produit la sortie binaire du perceptron.
- **Limitations** : Le perceptron a des limitations, notamment sa capacité à résoudre des problèmes non linéaires et son incapacité à apprendre des modèles complexes. Cependant, il a jeté les bases pour le développement de réseaux de neurones plus avancés, en particulier les réseaux multicouches qui peuvent apprendre des représentations hiérarchiques.

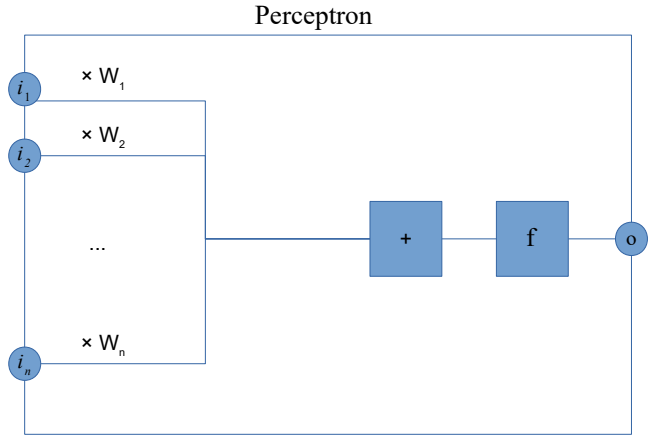
Perceptron



Perceptron en mettant à jour sa limite linéaire à mesure que d'autres exemples de formation sont ajoutés.[1]

1. Source: https://en.wikipedia.org/wiki/File:Perceptron_example.svg

Perceptron



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

Perceptron

Perceptron: Définition formelle

- Soit $y = f(z)$ la sortie du perceptron pour un vecteur d'entrée ***z***
- Soit N le nombre d'exemples d'entraînement
- Soit ***X*** l'espace de saisie des caractéristiques
- Soit $(x_1, d_1), \ldots, (x_N, d_N)$ be the ***N*** training examples, where
  - x_i est le vecteur caractéristique de *i*^{ème} exemple d'entraînement.
  - d_i est la valeur de sortie souhaitée
  - $x_{j,i}$ est la *i*^{ème} caractéristique de *j*^{ème} exemple d'entraînement.
  - $x_{j,0} = 1$

Perceptron: Définition formelle

- Les poids sont représentés de la manière suivante:
  - w_i est la *i*^{ème} valeur du vecteur de poids.
  - $w_i(t)$ est la *i*^{ème} valeur du vecteur de poids à un moment donné t.

Perceptron : Étapes

1. Initialiser les poids et les seuils
2. Pour chaque exemple, (x_j, d_j) dans l'ensemble d'entraînement
   - Calculer la sortie actuelle :

$$y_j(t) = f[w(t).x_j]$$

$$= f[w_0(t)x_{j,0} + w_1(t)x_{j,1} + w_2(t)x_{j,2} + \cdots + w_n(t)x_{j,n}]$$

   - Calculer le poids:

$$w_i(t+1) = w_i(t) + r.(d_j - y_j(t))x_{j,i}$$

r est le taux d'apprentissage.

Perceptron : Étapes

3. Répétez l'étape 2 jusqu'à l'erreur d'itération

$$\frac{1}{s}(\Sigma|d_j - y_j(t)|)$$

est inférieur au seuil spécifié par l'utilisateur γ, ou un nombre prédéterminé d'itérations ont été effectuées, où s est à nouveau la taille de l'ensemble de l'échantillon.

Fonction d'Échelon (Step Function)

Le perceptron utilise généralement une fonction d'activation simple, et la fonction d'échelon (step function) est fréquemment choisie pour cette tâche.

Définition

La fonction d'échelon attribue une sortie de 1 si la somme pondérée des entrées dépasse un certain seuil, et 0 sinon.

$$f(x) = \begin{cases} 1 & \text{si } x \geq \text{seuil} \\ 0 & \text{sinon} \end{cases}$$

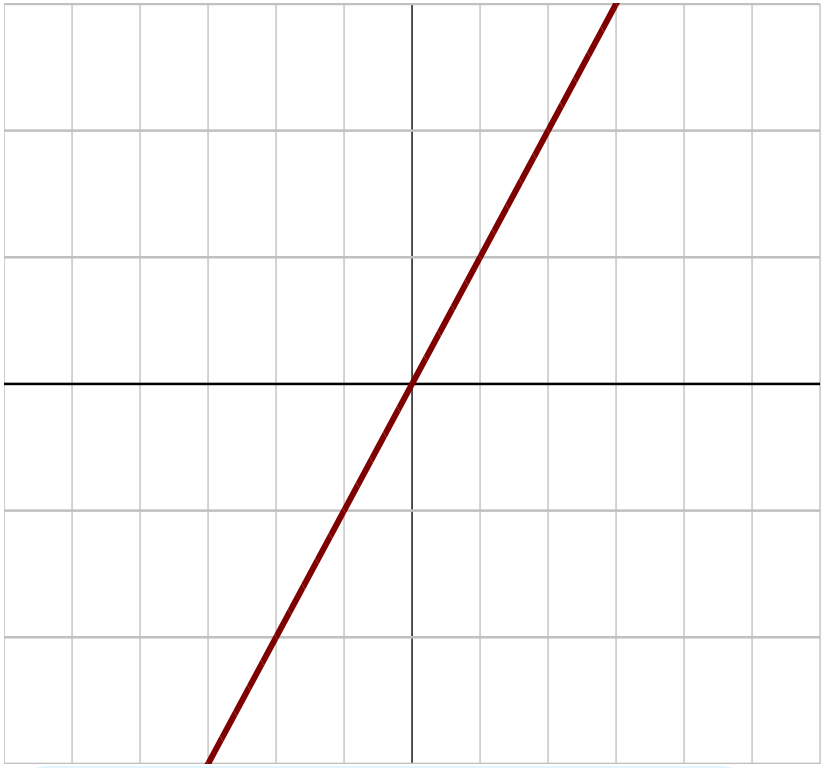
Fonction d'activation: fonction d'identité

Équation

$$f(x) = x$$

Dérivée

$$f'(x) = 1$$



Fonction d'identité

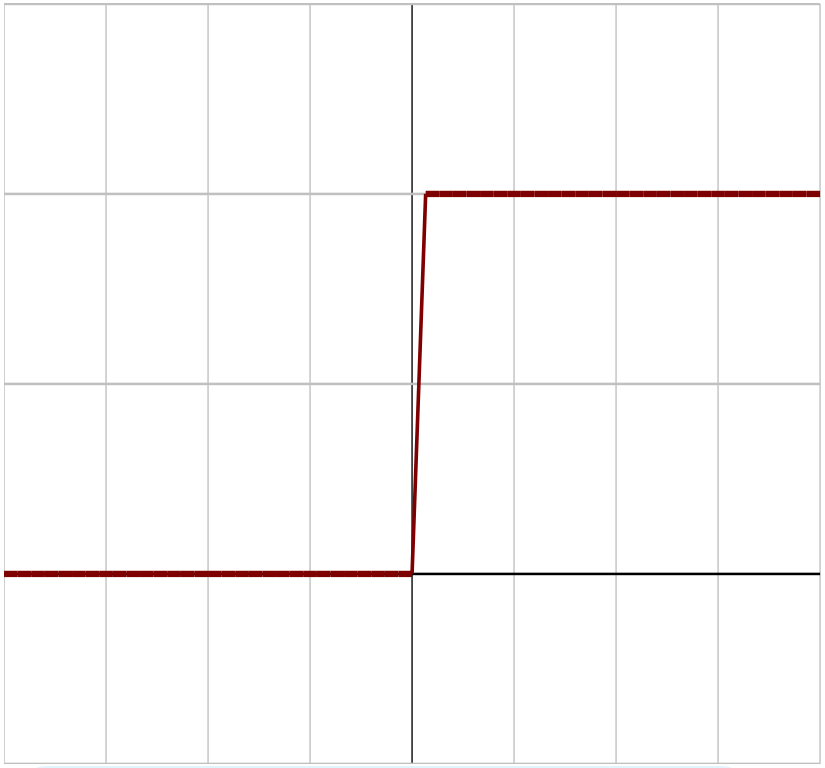
Fonction d'activation: pas binaire

Équation

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

Dérivée

$$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$$

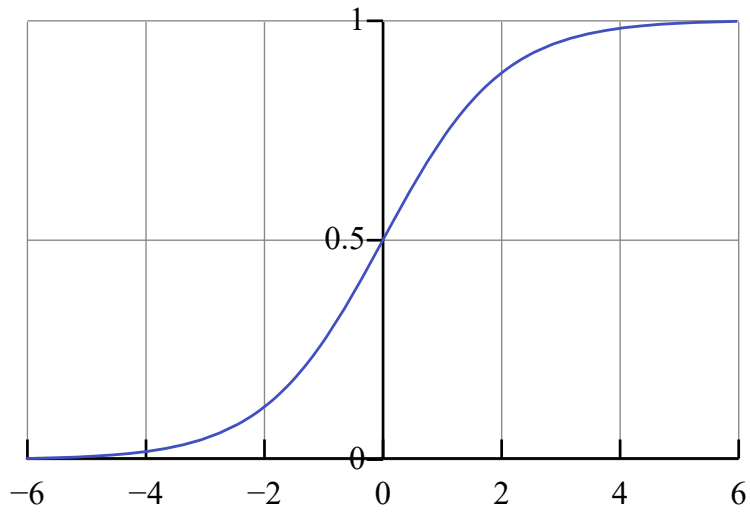


Pas binaire

Fonction d'activation: fonction sigmoïde

Équation

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



La fonction sigmoïde

Dérivée

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

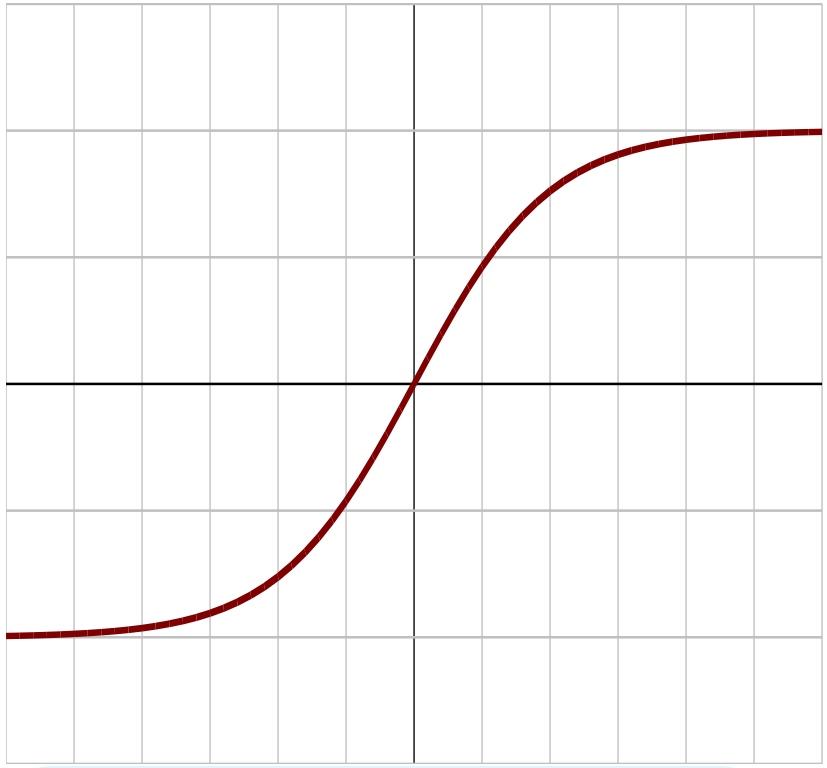
Fonction d'activation: TanH

Équation

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

Dérivée

$$f'(x) = 1 - f(x)^{2}$$



TanH

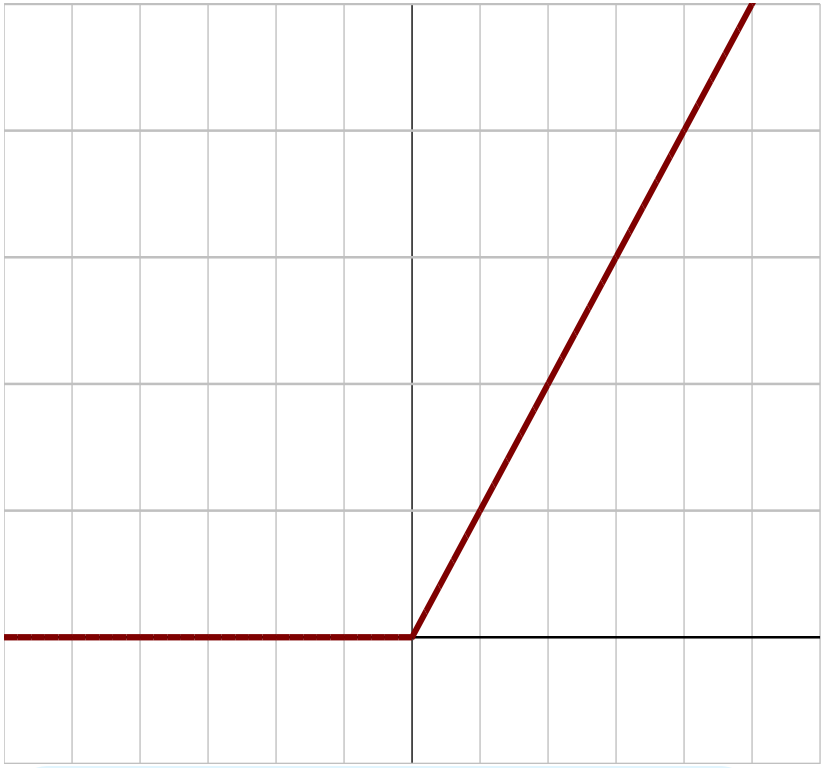
Fonction d'activation: Rectified linear unit: ReLU

Équation

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} = \max\{0, x\} = x 1_{x>0}$$

Dérivée

$$f'(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$$



Unité linéaire rectifiée (ReLU)

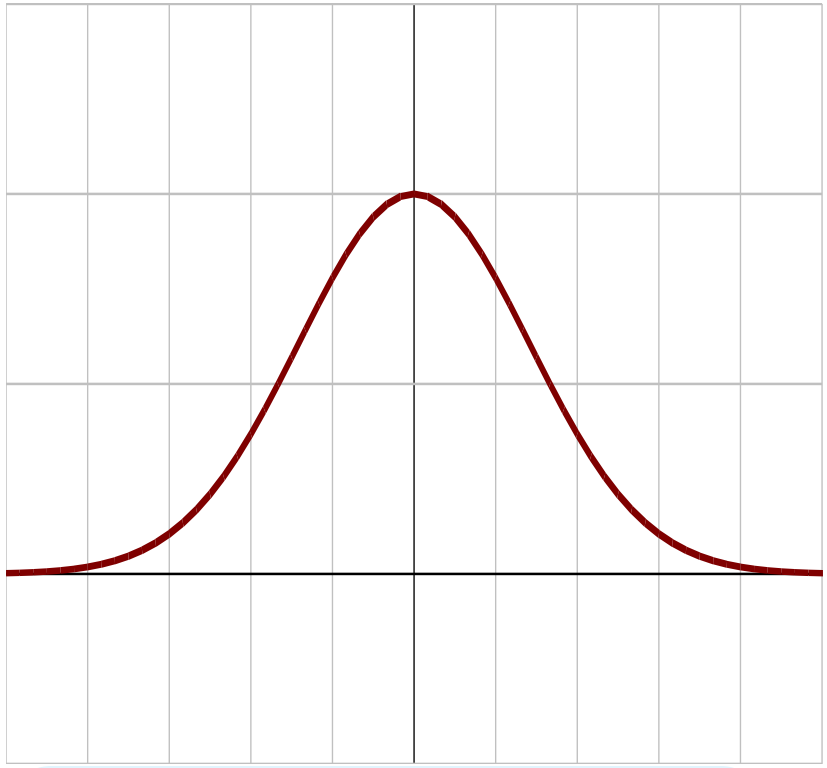
Fonction d'activation: Gaussien

Équation

$$f(x) = e^{-x^2}$$

Dérivée

$$f'(x) = -2xe^{-x^2}$$



Gaussien

Perceptron multiclasse

- Perceptron peut être généralisé à la classification multiclasse.
- Une fonction de représentation d'élément $f(x, y)$ fait correspondre chaque paire d'entrée/sortie possible à un vecteur d'élément à valeur réelle en dimension finie.
- le vecteur de caractéristique est multiplié par un vecteur de poids w, mais le score obtenu est maintenant utilisé pour choisir parmi de nombreux résultats possibles :

$$\hat{y} = \operatorname{argmax}_y f(x, y) \cdot w.$$

- La réapprentissage se fait par itération sur les exemples, en prédisant un résultat pour chacun, en laissant les poids inchangés lorsque le résultat prédit correspond à l'objectif, et en les modifiant lorsqu'il ne correspond pas. La mise à jour devient :

$$w_{t+1} = w_t + f(x, y) - f(x, \hat{y})$$

.

3.2. Apprentissage profond

Un **réseau de neurones profond**, également connu sous le nom de réseau de neurones profondément hiérarchisé ou réseau neuronal profond (DNN pour Deep Neural Network en anglais), est un type de réseau de neurones artificiels qui comprend plusieurs couches de traitement, généralement plus de deux. Ces réseaux sont appelés "profonds" en raison de leur architecture empilée de couches, permettant la création de représentations hiérarchiques complexes des données.

Architecture en couches : Les réseaux de neurones profonds sont composés de multiples couches, généralement divisées en trois types principaux :

- **Couche d'Entrée** : Reçoit les données brutes ou caractéristiques en entrée.
- **Couches Cachées** : Effectuent des transformations non linéaires et apprennent des représentations hiérarchiques des données.
- **Couche de Sortie** : Produit la sortie du réseau, adaptée à la tâche spécifique (classification, régression, etc.).

3.2. Apprentissage profond

- **Apprentissage Hiérarchique** : Les couches cachées d'un réseau de neurones profond apprennent des caractéristiques de plus en plus abstraites et complexes à mesure que l'on progresse en profondeur. Chaque couche représente une abstraction des caractéristiques extraites par les couches précédentes.
- **Fonctions d'Activation** : Des fonctions d'activation non linéaires, telles que ReLU (Rectified Linear Unit) ou ses variantes, sont couramment utilisées dans les couches cachées pour permettre au réseau d'apprendre des relations non linéaires.
- **Apprentissage Profond** : L'apprentissage profond implique l'ajustement simultané des poids de toutes les couches du réseau pour minimiser l'erreur de prédiction. Cela est généralement réalisé en utilisant des techniques de rétropropagation et de descente de gradient.
- **Utilisations** : Les réseaux de neurones profonds sont utilisés dans une variété de tâches, notamment la vision par ordinateur, la reconnaissance vocale, le traitement du langage naturel, la traduction automatique, la recommandation de contenu, et bien d'autres. Leur capacité à apprendre des représentations complexes a conduit à des avancées significatives dans de nombreux domaines de l'intelligence artificielle.

Apprentissage profond

- Le terme "profond" se réfère à un réseau qui a un grand nombre de couches, généralement plus de trois.
- Ces réseaux sont également appelés "réseaux de neurones profonds" ou "réseaux neuronaux profonds".
- Les réseaux de neurones profonds ont été rendus populaires par leurs capacités à apprendre des représentations hiérarchiques complexes.

Exemple: Tensorflow

```
# Importation des bibliothèques nécessaires de TensorFlow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD

# Étape 1: Création d'un modèle séquentiel
model = Sequential()

# Étape 2: Ajout d'une couche dense avec une fonction d'activation ReLU
# La couche a 4 neurones, une fonction d'activation 'relu', et prend une entrée
model.add(Dense(4, activation='relu', input_shape=(3,)))
```

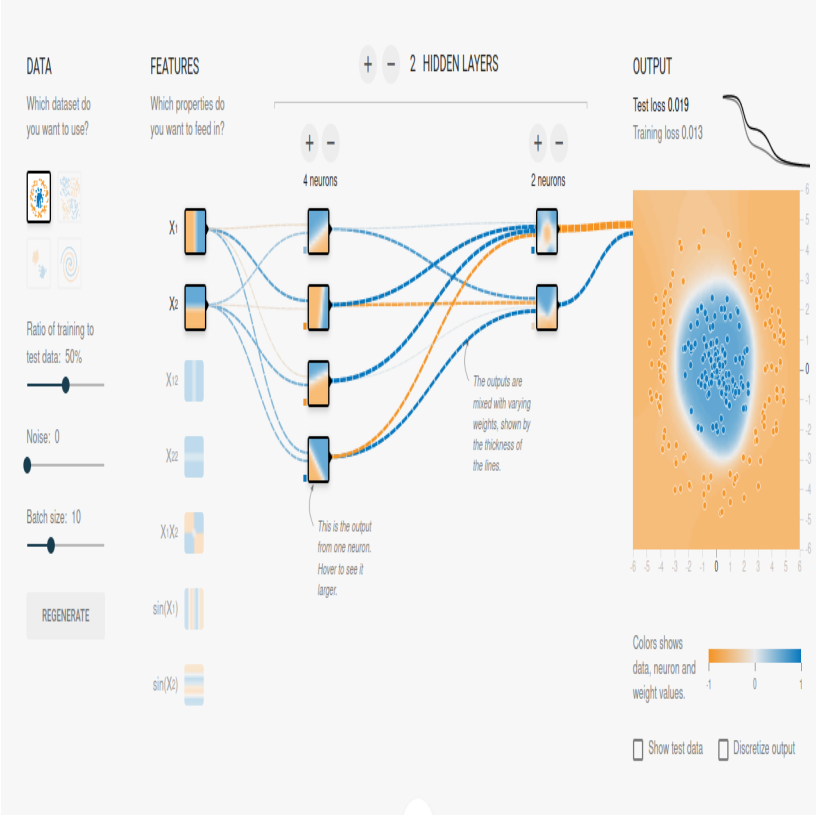
Exemple: Tensorflow

```
# Étape 3: Ajout d'une couche dense de sortie avec une fonction d'activation sof
# La couche a 2 neurones pour une tâche de classification binaire, et softmax es
# pour obtenir des probabilités
model.add(Dense(units=2, activation='softmax'))

# Étape 4: Compilation du modèle
# Utilisation de la descente de gradient stochastique (SGD) comme optimiseur ave
# La fonction de perte est 'mean_squared_error' pour un problème de régression
# Les performances du modèle seront mesurées en termes de 'accuracy' (précision)
sgd = SGD(lr=0.01)
model.compile(loss='mean_squared_error', optimizer=sgd, metrics=['accuracy'])
```

- **Étape 1**: On crée un modèle séquentiel, qui est une pile linéaire de couches.
- **Étape 2**: On ajoute une couche dense avec 4 neurones utilisant la fonction d'activation ReLU. La couche prend une entrée de forme (3,) - cela signifie que chaque exemple d'entraînement a trois caractéristiques.
- **Étape 3**: On ajoute une couche dense de sortie avec 2 neurones utilisant la fonction d'activation softmax. Cela est couramment utilisé pour les tâches de classification binaire, fournissant des probabilités pour chaque classe.
- **Étape 4**: On compile le modèle en spécifiant l'optimiseur (SGD avec un taux d'apprentissage de 0.01), la fonction de perte ('mean_squared_error' pour une tâche de régression), et les métriques de performance ('accuracy' pour mesurer la précision du modèle).

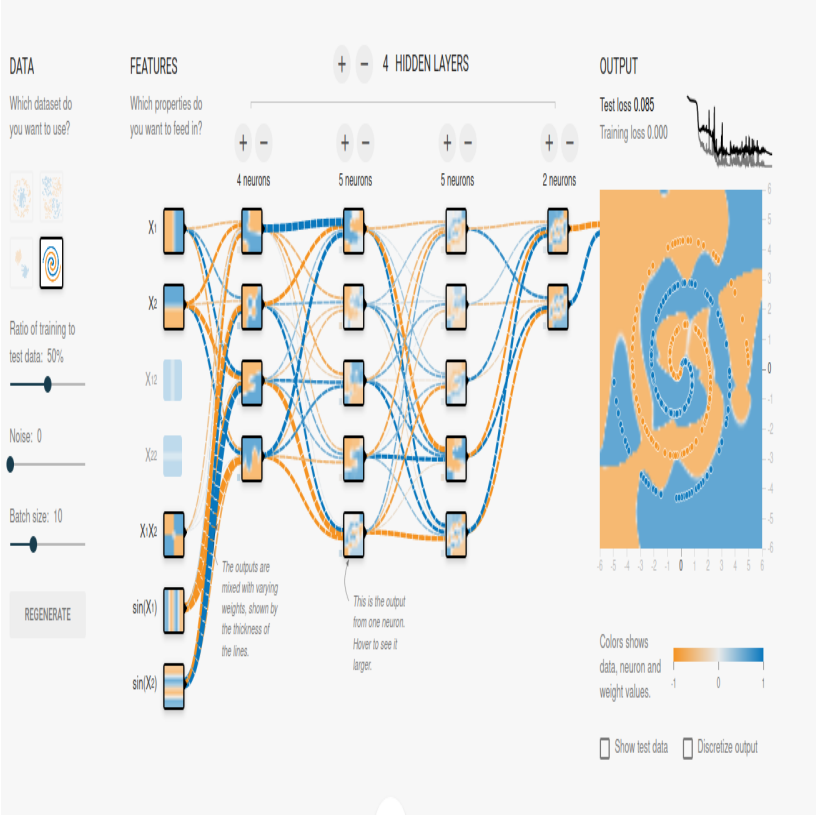
3.2. Apprentissage profond



Source: https://

playground.tensorflow.org/

3.2. Apprentissage profond



Source: https://

playground.tensorflow.org/

3.2. Apprentissage profond

Composants des réseaux de neurones artificiels

Organisation

Un réseau de neurones profond est une architecture complexe où l'information circule de la couche d'entrée à travers les couches cachées jusqu'à la couche de sortie. Chaque connexion entre les neurones est associée à un poids qui est ajusté pendant le processus d'apprentissage pour optimiser les performances du modèle sur la tâche spécifique. L'utilisation de plusieurs couches cachées permet au réseau d'apprendre des représentations de plus en plus abstraites et complexes des données.

Composants des réseaux de neurones artificiels

Organisation

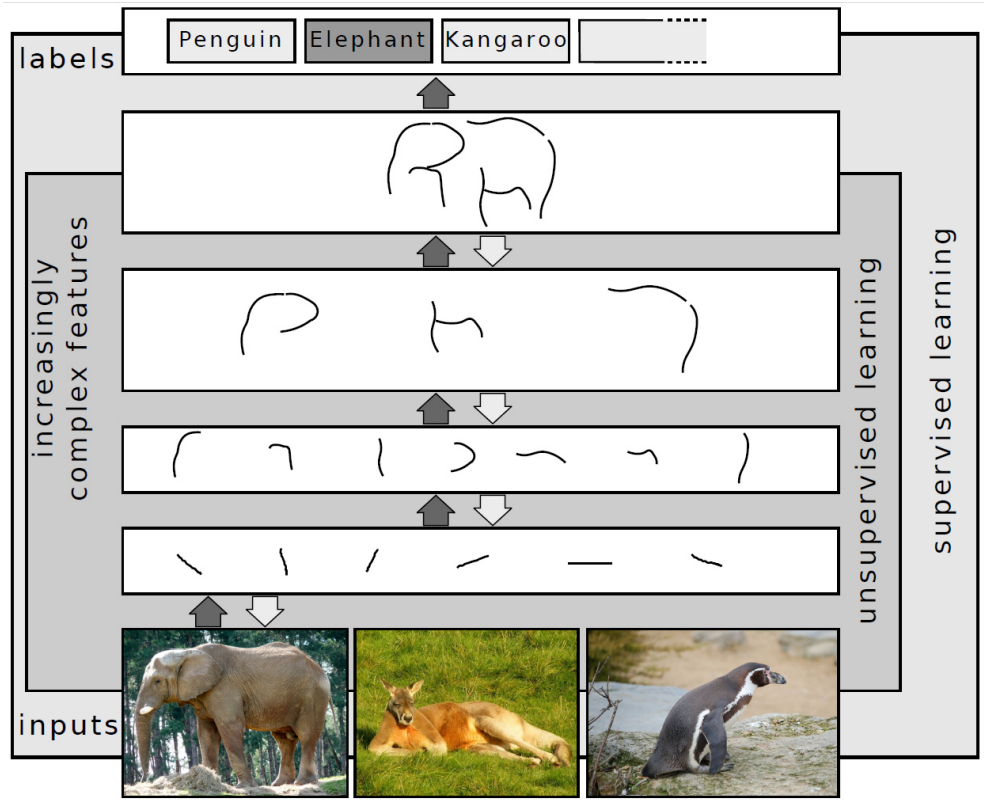
- **Couche d'entrée** : La couche d'entrée est la première couche du réseau. Elle reçoit les données externes, souvent représentées par des caractéristiques d'un ensemble de données. Chaque neurone dans la couche d'entrée correspond à une caractéristique spécifique.
- **Couche de sortie** : La couche de sortie est la dernière couche du réseau. Elle produit le résultat final du modèle en fonction de la tâche spécifique, telle que la classification d'une image, la prédiction d'une valeur, etc. Le nombre de neurones dans cette couche dépend du type de problème (par exemple, un neurone pour chaque classe dans une tâche de classification).
- **Couches cachées** : Entre la couche d'entrée et la couche de sortie, il peut y avoir zéro ou plusieurs couches cachées. Ces couches sont responsables de l'extraction de caractéristiques complexes à partir des données d'entrée. Chaque neurone dans une couche cachée combine les informations des neurones de la couche précédente pour apprendre des représentations hiérarchiques.

Composants des réseaux de neurones artificiels

Organisation et connectivité

Connectivité entièrement connectée: Dans une connectivité entièrement connectée, chaque neurone d'une couche est connecté à chaque neurone de la couche suivante. Cela signifie que toutes les informations de la couche précédente sont transmises à chaque neurone de la couche suivante. C'est la configuration la plus courante dans les couches totalement connectées, généralement présentes dans les parties du réseau proches de la sortie.

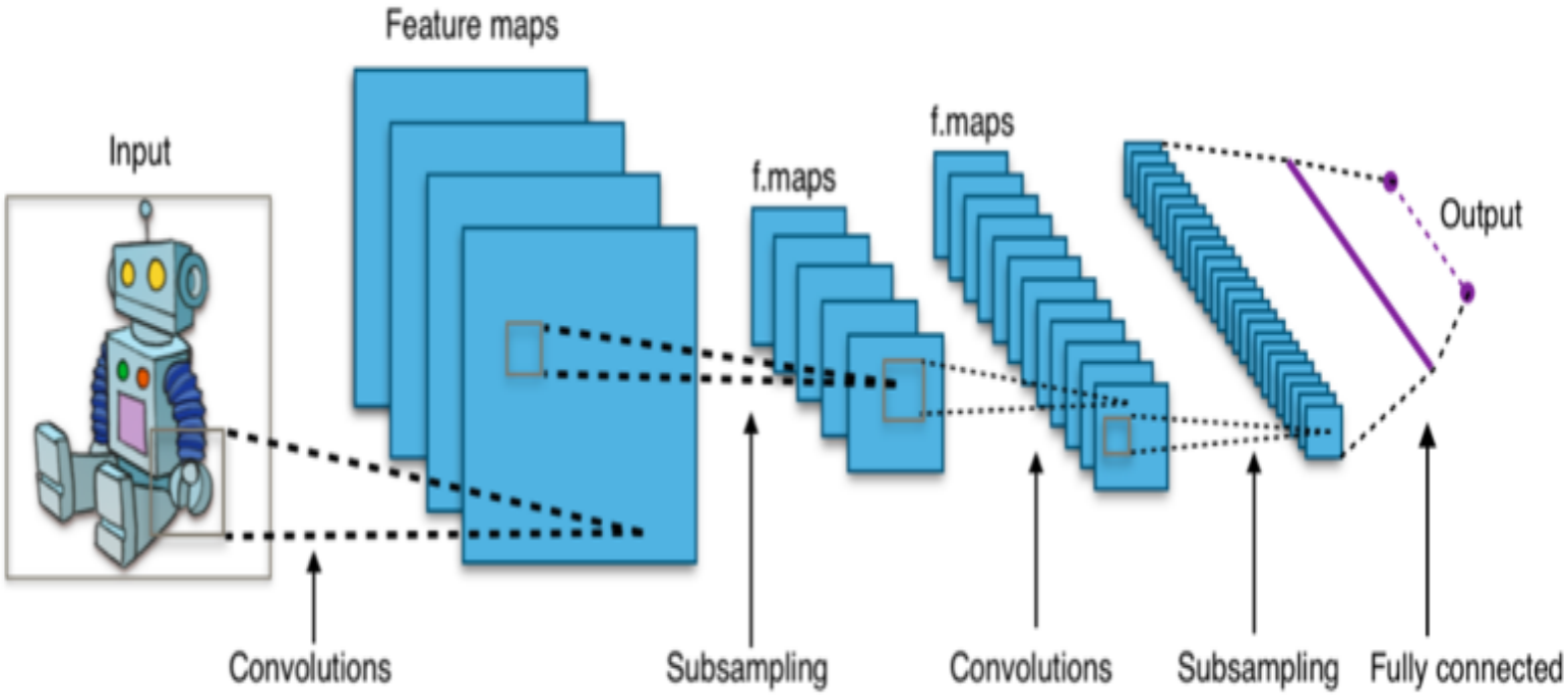
Réseaux de neurones convolutionnels



Source: https://en.wikipedia.org/wiki/

File:Deep_Learning.jpg

Réseaux de neurones convolutionnels



Réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels (CNN) sont une classe d'architectures de réseaux neuronaux conçues principalement pour l'analyse des images. Ils ont été particulièrement efficaces dans des tâches telles que la classification d'images, la détection d'objets, et la segmentation d'images.

- **Analyse des Images** : Les CNN sont spécifiquement conçus pour travailler avec des données structurées en grilles, comme les images. Ils sont capables de capturer des motifs et des caractéristiques spatiales importantes dans les images.

Réseaux de neurones convolutionnels

- **Utilise la convolution** La convolution est une opération mathématique linéaire utilisée pour extraire des caractéristiques locales à partir de l'image. Les filtres de convolution sont appliqués à l'image pour détecter des motifs tels que des bords, des textures, ou des formes.
- **Architecture en couches** : Les CNN suivent généralement une architecture en couches. Ils ont une couche d'entrée pour recevoir l'image, une ou plusieurs couches cachées composées principalement de couches convolutives, et une couche de sortie pour produire les résultats finaux.

Réseaux de neurones convolutionnels

- **Couches convolutives** : Les couches convolutives sont responsables de l'extraction des caractéristiques de l'image. Chaque couche peut avoir plusieurs filtres de convolution qui apprennent à détecter des motifs spécifiques. Ces couches sont souvent suivies de couches de pooling pour réduire la dimensionnalité de la représentation tout en préservant les caractéristiques importantes.
- **Applications** : Les CNN sont largement utilisés dans des applications telles que la classification d'images (par exemple, reconnaître des animaux dans des photos), la détection d'objets (localiser et identifier des objets spécifiques), et la segmentation d'images (diviser une image en régions sémantiquement significatives)..

3.2. Apprentissage profond

Réseaux de neurones convolutionnels:

- **Modèle hiérarchique des données** : Les réseaux neuronaux convolutifs (CNN) sont en effet conçus pour capturer des caractéristiques hiérarchiques dans les données, en particulier dans le contexte de l'analyse d'images. Cela signifie qu'ils peuvent apprendre des motifs simples dans les premières couches, puis combiner ces motifs pour former des caractéristiques plus complexes dans les couches suivantes.
- **Architecture d'un CNN** : Un réseau neuronal convolutif est généralement composé d'une couche d'entrée, de plusieurs couches cachées et d'une couche de sortie. Les couches cachées consistent principalement en couches convolutionnelles, mais peuvent également inclure d'autres types de couches telles que des couches de regroupement (pooling), des couches entièrement connectées, et des couches de normalisation.

Réseaux de neurones convolutionnels:

- **Couches convolutionnelles et fonction d'activation** : Les couches convolutionnelles appliquent des filtres pour extraire des caractéristiques de l'image. La multiplication est effectuée par la convolution. La fonction d'activation la plus couramment utilisée est ReLU (Rectified Linear Unit), qui introduit une non-linéarité dans le modèle. Cette non-linéarité est importante pour permettre au réseau d'apprendre des relations complexes dans les données.
- **Couches supplémentaires** : Après les couches de convolution, on peut avoir des couches de regroupement pour réduire la dimensionnalité, des couches entièrement connectées pour combiner des caractéristiques globales, et des couches de normalisation pour améliorer la stabilité de l'apprentissage.

Noyau (traitement d'image)

Un noyau dans le contexte du traitement d'images, également appelé filtre ou masque, est une petite matrice qui est appliquée sur une image à l'aide d'une opération de convolution. L'objectif de l'application de ces noyaux est de réaliser diverses opérations de filtrage sur l'image, telles que la détection de contours, l'amélioration des détails, la mise en évidence de certaines caractéristiques, etc.

- **Convolution dans les CNN** : Dans les CNN, la convolution est une opération clé qui consiste à appliquer un ensemble de filtres (noyaux) à une image d'entrée. Chaque filtre est conçu pour extraire des caractéristiques spécifiques de l'image, comme des bords, des textures, ou d'autres motifs.

Noyau (traitement d'image)

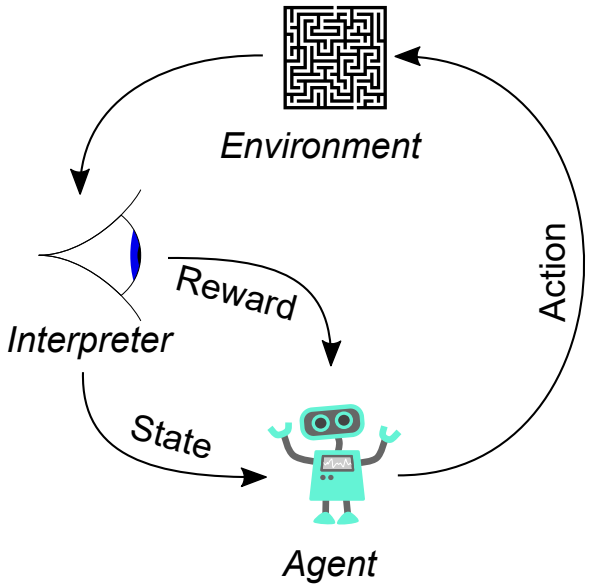
- **Apprentissage des noyaux** : L'une des caractéristiques importantes des CNN est la capacité d'apprendre les filtres (noyaux) de manière automatique pendant l'entraînement. Au lieu de définir manuellement les filtres comme dans le traitement d'images traditionnel, les CNN ajustent les poids des filtres pendant la phase d'apprentissage en fonction des caractéristiques qui sont importantes pour la tâche à accomplir.

Noyau (traitement d'image)

- **Rôle dans la hiérarchie des caractéristiques** : Les premières couches d'un CNN apprennent généralement des filtres simples qui détectent des contours ou des textures de base. À mesure que l'on progresse dans les couches du réseau, les filtres deviennent plus complexes, capturant des caractéristiques de niveau supérieur, jusqu'à ce que la sortie finale représente des caractéristiques abstraites de l'image d'entrée.
- **Réduction de dimension avec le pooling** : Après la convolution, les CNN utilisent souvent des couches de pooling pour réduire la dimension de la représentation, tout en préservant les caractéristiques importantes extraites par les filtres. Cela permet d'économiser des ressources computationnelles tout en maintenant les informations cruciales.

Apprentissage par renforcement

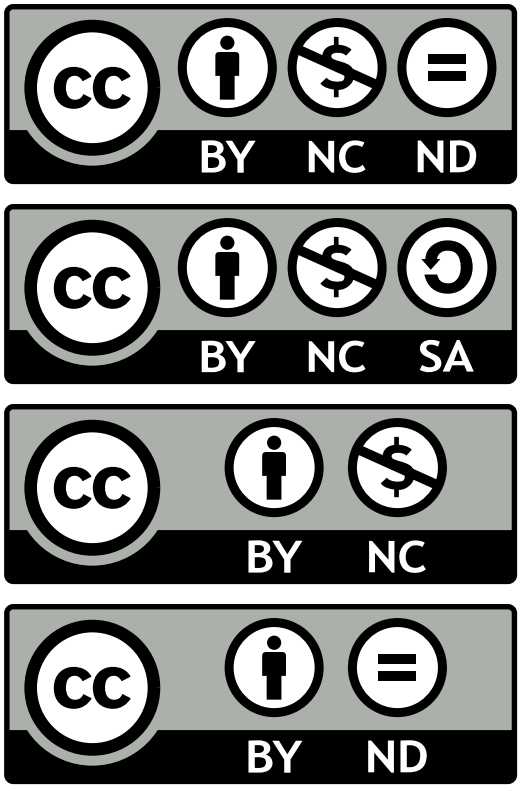
- L'apprentissage par renforcement (Reinforcement Learning - RL) est une branche de l'apprentissage automatique inspirée des théories de la psychologie animale.
- **Agent autonome** : RL implique un agent autonome interagissant avec un environnement.
- **Prise de décision** : L'agent prend des décisions en fonction de son état actuel.
- **Récompenses et pénalités** : L'environnement fournit à l'agent des récompenses, qui peuvent être positives ou négatives.
- **Objectif** : L'objectif est de maximiser la somme des récompenses cumulatives au fil du temps.

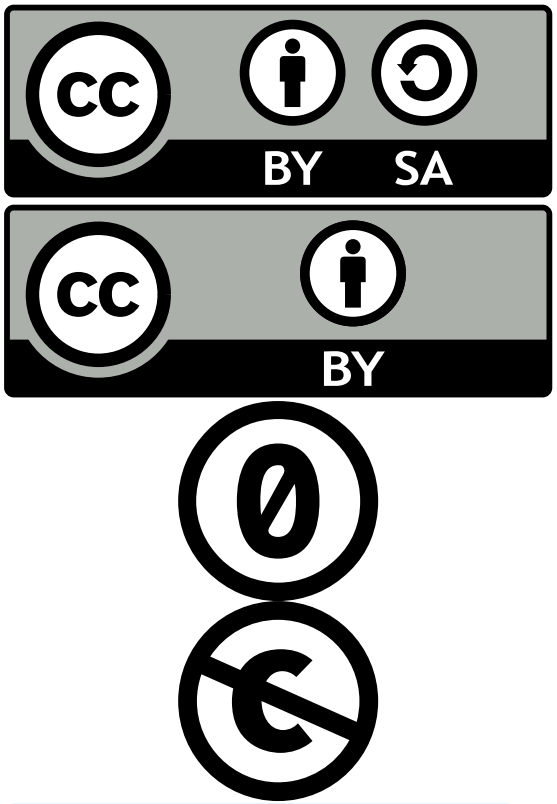


Licences, Éthique et la vie privé

- Droits d'utilisation des données
- Confidentialité et vie privée
- Éthique

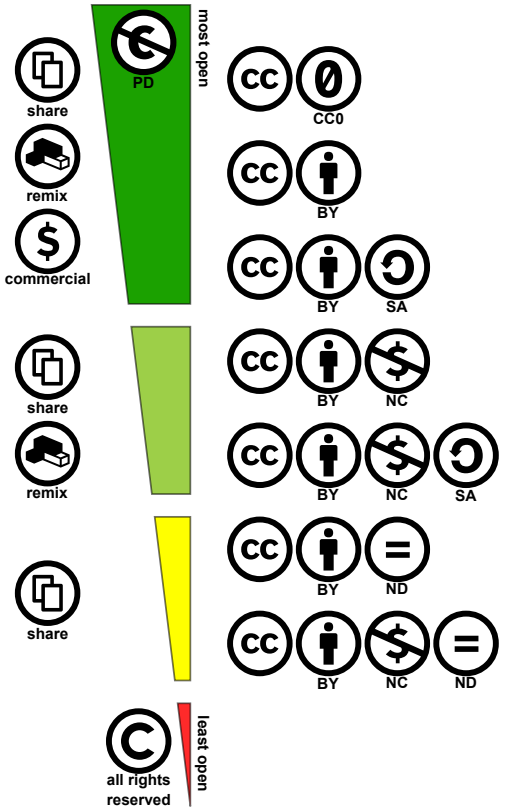
3.4. Licences, Ethiques et la vie privé





Exemples: Creative Commons

(CC)



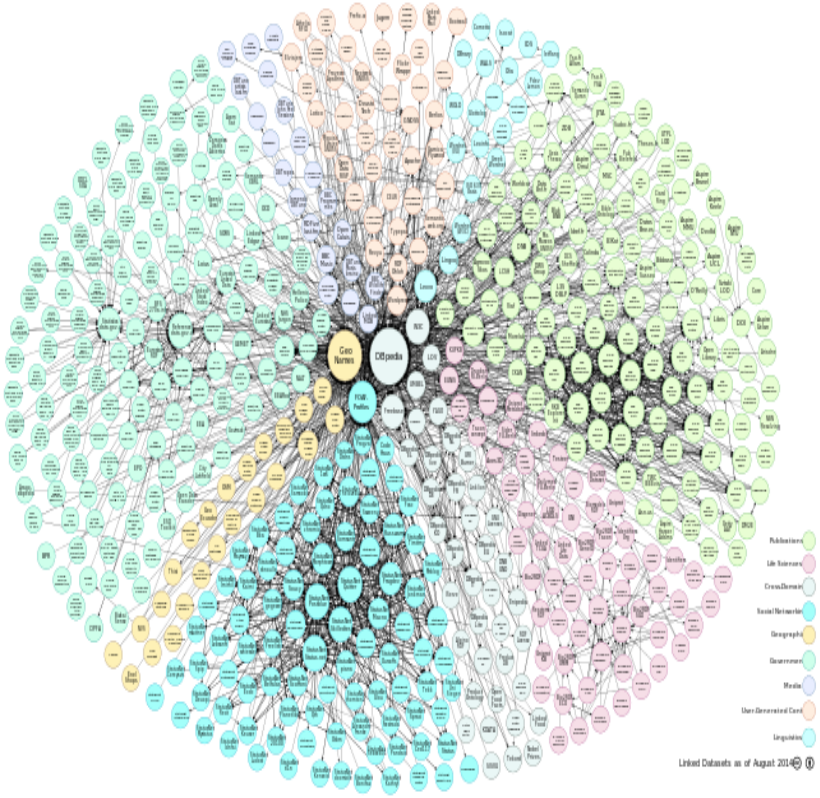
Exemples: Creative Commons

(CC)

3.4. Licences, Ethiques et la vie privé

Données ouvertes

3.4. Licences, Ethiques et la vie privé



Données ouvertes liées (Linked Open data: LOD)

Données archivées

Références

Ressources en ligne

- Artificial Neural Network
- Noyau (traitement d'image)
- Réseau neuronal convolutif
- Perceptron

Références

Couleurs

- Color Tool - Material Design

Images

- Wikimedia Commons