Cours IA

Pr. Yassine ZARROUK

Introduction générale à l'Intelligence Artificielle

Définition formelle

L'**Intelligence Artificielle (IA)** peut être définie comme la **science et l'ingénierie de la fabrication de machines capables d'exécuter des tâches qui requièrent normalement l'intelligence humaine**.

Ces tâches incluent : la perception (vision, son, texte), le raisonnement, l'apprentissage, la planification, la prise de décision et l'interaction naturelle avec l'environnement.

> Définition (McCarthy, 1956) : « L'IA est la science et l'ingénierie de la fabrication de machines intelligentes, spécialement de programmes informatiques intelligents. »

IA faible vs IA forte

- **IA faible (ou étroite)** : conçue pour accomplir une tâche spécifique (ex. reconnaissance faciale, traduction automatique, conduite assistée).

  → Elle ne "comprend" pas réellement ce qu'elle fait, mais exécute des modèles appris.

- **IA forte (ou générale)** : viserait une intelligence comparable à celle de l'être humain, capable d'apprendre, raisonner et s'adapter dans n'importe quel domaine.

  → Aujourd'hui, **aucune IA forte n'existe** ; toutes les applications actuelles relèvent de l'IA faible.

Domaines d'application de l'IA

L'IA est désormais omniprésente dans tous les secteurs technologiques :

- **Santé** : diagnostic automatique, analyse d'imagerie médicale, médecine personnalisée.
- **Robotique** : navigation autonome, manipulation intelligente, interaction homme-robot.
- **Agriculture intelligente** : détection des maladies des plantes, estimation des rendements, robots agricoles.
- **Finance** : détection de fraude, trading algorithmique, scoring de crédit.
- **Industrie 4.0** : maintenance prédictive, optimisation de la production, contrôle qualité.
- **Véhicules autonomes** : perception de l'environnement, prise de décision, pilotage automatique.

II. Historique de l'IA

1950–1970 : L'ère de l'IA symbolique

- **1950 : Alan Turing** propose le célèbre **test de Turing**, qui définit un critère d'intelligence pour les machines.

- **1956 : Conférence de Dartmouth** (John McCarthy, Marvin Minsky, Allen Newell, Herbert Simon) : naissance officielle du terme *Artificial Intelligence*.
- **1957 : Perceptron de Rosenblatt**, premier modèle de neurone artificiel entraînable.
- Cette période est dominée par l'**IA symbolique**, basée sur la logique, les règles et la manipulation de symboles.

1980–2000 : L'essor des systèmes experts

- Apparition des **systèmes experts**, capables de raisonner à partir de bases de connaissances (ex. MYCIN pour le diagnostic médical).
- Développement de la **logique floue** (Zadeh) pour gérer l'incertitude.
- Introduction des **arbres de décision** et des **réseaux bayésiens** pour la modélisation probabiliste.
- Début de l'**apprentissage automatique** (*machine learning*) moderne.

2000–2010 : Renaissance du Machine Learning

- Apparition d'algorithmes plus puissants et généralisables :
  - **SVM (Support Vector Machines)**
  - **Boosting / AdaBoost**
  - **k-Nearest Neighbors (k-NN)**
  - **Random Forests**
- Amélioration de la disponibilité des données et des capacités de calcul.
- L'IA devient plus pragmatique et orientée données.

2012–2025 : L'ère du Deep Learning

- **2012 : AlexNet** (Krizhevsky, Hinton) remporte le concours ImageNet avec une précision révolutionnaire grâce au **GPU computing**.
- Expansion des **CNN (Convolutional Neural Networks)** pour la vision, des **RNN (Recurrent Neural Networks)** pour le langage et des **GAN (Generative Adversarial Networks)** pour la génération d'images.
- **2017 : Transformers** (*Attention Is All You Need*, Vaswani et al.) marquent une nouvelle révolution : parallélisation, attention multi-têtes et architecture universelle.
- **2020–2025 :** émergence de l'**IA générative** (GPT, DALL-E, Stable Diffusion), des **Vision Transformers (ViT)** et de l'intégration dans les systèmes **embarqués** et **autonomes**.

1. Principaux paradigmes de l'Intelligence Artificielle

L'IA repose sur plusieurs **paradigmes fondamentaux**, correspondant à différentes façons de représenter et de traiter la connaissance :

a) IA symbolique (ou logique)

Ce paradigme repose sur la manipulation explicite de **symboles et de règles logiques**.

Il vise à imiter le raisonnement humain à travers des **systèmes experts**, des **bases de connaissances** et des **moteurs d'inférence**.

Exemple : le système MYCIN (années 1970) pour le diagnostic médical.

Outils typiques : logique propositionnelle, arbres de décision, chaînes d'inférence.

b) IA connexionniste

Inspirée du fonctionnement du cerveau humain, elle repose sur les **réseaux de neurones artificiels**.

Les connaissances ne sont pas codées par des règles, mais **apprises à partir des données**.

L'apprentissage se fait par **propagation du gradient** et **ajustement des poids synaptiques**.

C'est le paradigme dominant du **deep learning** moderne.

c) IA évolutionniste

Ce paradigme s'inspire des **mécanismes de l'évolution naturelle** (sélection, mutation, recombinaison).

Les solutions sont vues comme des individus d'une population qui évolue au fil des générations.

Les **algorithmes génétiques**, **stratégies d'évolution**, ou encore **optimisations par essaims de particules** (PSO) en sont des exemples.

Ils sont souvent utilisés pour l'**optimisation de modèles** ou **des sstèmes**.

d) IA probabiliste

Fondée sur la **théorie des probabilités** et de l'**incertitude**, elle vise à modéliser des phénomènes aléatoires.

Les décisions sont prises en fonction de la **vraisemblance** et de la **mise à jour des croyances**.

Exemples : **réseaux bayésiens**, **modèles de Markov cachés (HMM)**, **algorithmes de filtrage de Kalman**.

Très utilisés en **reconnaissance vocale**, **robotique mobile**, et **vision incertaine**.

2. Grands types de tâches en Intelligence Artificielle

Les systèmes d'IA peuvent être classés selon la **nature de la tâche** qu'ils accomplissent. On distingue généralement les catégories suivantes :

- **Classification :** attribuer une étiquette à un élément parmi un ensemble fini de classes.

  Exemple : diagnostic médical — classer une image de feuille comme « saine » ou « infectée ».
- **Régression :** prédire une valeur numérique continue à partir de données d'entrée.

  Exemple : estimation du rendement agricole à partir de variables climatiques et d'indices de végétation.
- **Clustering (regroupement non supervisé) :** organiser des données similaires en groupes sans connaître les étiquettes à l'avance.

  Exemple : segmentation automatique de clients selon leurs habitudes d'achat.
- **Détection :** localiser et identifier des objets dans une image ou une scène.

  Exemple : reconnaissance d'objets agricoles sur une image de champ.
- **Segmentation :** attribuer une étiquette à chaque pixel d'une image pour délimiter précisément les régions d'intérêt.

  Exemple : délimitation des zones malades sur une feuille ou une plante.
- **Planification :** déterminer une séquence optimale d'actions pour atteindre un objectif.

  Exemple : plan de navigation autonome pour un robot agricole.
- **Raisonnement et prise de décision :** établir des déductions logiques ou choisir la meilleure action selon un ensemble de règles ou de politiques.

Exemple : système expert pour recommander un traitement phytosanitaire selon les symptômes détectés.

3. Qu'est-ce que le Machine Learning ?

Définition (Arthur Samuel, 1959) :

> « Le Machine Learning est la capacité d'un ordinateur à apprendre sans être explicitement programmé. »

Autre formulation (Tom Mitchell, 1997) :

> Un programme apprend à partir d'une expérience E, pour une tâche T et une mesure de performance P, si sa performance sur T, mesurée par P, s'améliore avec E.

Exemple illustratif :

Tâche (T) : reconnaître des images de tomates saines ou malades.

Expérience (E) : apprentissage sur 10 000 images annotées par des experts.

Performance (P) : précision de classification.

→ Plus le modèle voit d'images, plus sa précision augmente : il apprend !

Différence avec la programmation classique :

Programmation traditionnelle	Machine Learning
Les règles sont codées à la main par le programmeur	Les règles sont apprises automatiquement à partir des données
Entrée + Règles → Résultat	Entrée + Résultat → Règles
Exemple : tri de nombres, calcul d'impôts	Exemple : reconnaissance d'images, prédiction de ventes

Analogie simple :

Apprendre à un ordinateur à reconnaître un chat :

En programmation classique : tu écris des règles (« 4 pattes, 2 yeux, oreilles triangulaires… »).

En Machine Learning : tu montres des milliers d'images de chats, et le modèle **découvre** lui-même ce qui définit un chat.

---

4. Principe général du Machine Learning

Le Machine Learning consiste à **trouver une fonction** f qui approxime la relation entre les entrées x et les sorties y :

$y=f(x)+\varepsilon$

où :

x : vecteur de caractéristiques (features) ;

y : variable cible ;

ε : erreur de prédiction (bruit).

But : que f généralise bien aux nouvelles données jamais vues.

Exemple :

Entrée : image de feuille ;

Sortie : 0 = saine, 1 = infectée ;

f : modèle entraîné (réseau de neurones, SVM, etc.) ;

Performance : précision, rappel, F1-score…

5. Les trois grands modes d’apprentissage

Apprentissage supervisé

Principe :

On dispose d’un **ensemble de données étiquetées** (chaque exemple possède la « bonne » réponse).

Le but est d’apprendre une fonction f:X→Y à partir d’exemples (xi,yi).

Sous-catégories :

1. **Classification :** la sortie Y est **discrète** (catégorielle).

   Exemples :

   - Reconnaissance faciale : « personne A / personne B » ;
   - Détection de maladies sur des feuilles : « saine / infectée » ;
   - Diagnostic médical : « positif / négatif ».

   Méthodes courantes : k-NN, SVM, arbres de décision, réseaux de neurones.

2. **Régression :** la sortie Y est **continue**.

   Exemples :

   - Prédire le prix d’un bien immobilier ;
   - Estimer la température du sol ;
   - Calculer le rendement agricole attendu.

   Méthodes courantes : régression linéaire, régression polynomiale, réseaux neuronaux.

Schéma à projeter :

```
(Entrée : image de feuille) ——▶ [Modèle entraîné] ——▶ (Sortie : saine ou malade)
```

Apprentissage non supervisé

Principe :

Les données **ne sont pas étiquetées** ; le modèle doit découvrir **la structure cachée** ou des **groupes naturels**.

Tâches principales :

1. **Clustering (regroupement)** :

   Le modèle regroupe les points similaires. Exemples :

   - Regrouper les clients selon leurs comportements ;
   - Classer des plantes selon la texture de leurs feuilles.
   - Méthodes : K-means, DBSCAN, Mean Shift.

2. **Réduction de dimension** :

But : représenter des données complexes dans un espace plus petit tout en conservant l'essentiel de l'information. Exemples :

- PCA (Analyse en Composantes Principales) ;
- t-SNE, UMAP (visualisation de données en 2D).

Illustration :

Nuage de points non coloré (avant clustering) → K-means colorie les groupes selon leur similarité.

Formule :

$$\min \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Apprentissage par renforcement

Principe :

Un **agent intelligent** apprend en interagissant avec un **environnement**.

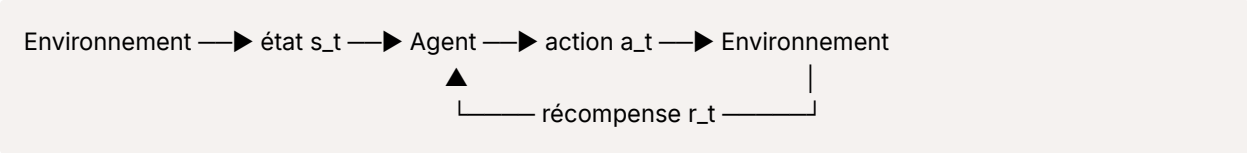
À chaque étape :

1. Il observe un **état** sts_tst.
2. Il choisit une **action** ata_tat.
3. L'environnement renvoie une **récompense** rtr_trt.
4. L'objectif est de maximiser les **récompenses cumulées**.

Exemples :

- Un robot agricole apprend à éviter les obstacles ;
- AlphaGo apprend à jouer au Go en jouant contre lui-même ;
- Un système d'arrosage intelligent apprend à économiser l'eau selon l'humidité.

Schéma de boucle d'interaction :



Objectif mathématique :

$$\max_{\pi} \ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, r_t\right]$$

où π est la politique (règle de décision) et γ un facteur d'actualisation (0 < γ < 1).

II .Comprendre la convolution dans un CNN

1. Principe général

Une image d'entrée, par exemple de taille 416×416×3 (largeur, hauteur, canaux de couleur), est traitée par un réseau de neurones convolutionnel (CNN) pour en extraire automatiquement des motifs visuels (bords, textures, formes).

Chaque couche du CNN apprend à détecter des motifs de plus en plus complexes :

- Les premières couches détectent des contours simples.
- Les couches intermédiaires identifient des parties d'objet.
- Les dernières couches reconnaissent des structures complètes.

L'objectif d'une convolution est de produire une **carte d'activation** (ou feature map) qui indique où un motif apparaît dans l'image.

---

2. La formule de la convolution

La valeur de sortie au point (x, y) d'une feature map est donnée par :

$$\text{sortie}(x, y) = \sum_{i,j} I(x+i, y+j) \times K(i, j) + b$$

Explication des termes :

- I(x+i,y+j) : valeur du pixel d'entrée à la position (x+i, y+j)
- K(i,j) : valeur du filtre (kernel) à la position (i,j)
- b : biais ajouté à la sortie
- x,y : coordonnées du pixel de sortie dans la feature map
- i,j : indices parcourant les cases du filtre

Chaque filtre (ou kernel) est une petite matrice (souvent 3×3 ou 5×5) qui glisse sur l'image pour détecter un motif local.

Pour chaque position (x, y), on multiplie les valeurs du patch de l'image par celles du filtre, on additionne le tout, puis on ajoute le biais.

---

3. Interprétation intuitive

- Si le motif local dans l'image **ressemble** à ce que le filtre cherche (par exemple un bord vertical), la somme sera **élevée**.
- Si le motif ne correspond pas, la somme sera **faible** ou **négative**.

Ces valeurs (appelées activations) forment les pixels de la nouvelle image appelée **feature map**.

---

4. La fonction d'activation (ReLU)

Après la convolution, chaque valeur passe par une fonction d'activation.

La plus utilisée est la fonction ReLU (Rectified Linear Unit) :

$$f(z) = \max(0,, z)$$

3. Formulation du problème de marge dure (hard margin)

Elle garde uniquement les valeurs positives et remplace les valeurs négatives par zéro.

Cela permet au réseau de ne conserver que les motifs pertinents.

5. Exemple de calcul

Si un filtre de taille 3×3 parcourt une image, on calcule pour chaque position :

$$sortie(x,y) = I(x,y)K(0,0) + I(x,y+1)K(0,1) + \cdots + I(x+2,y+2)K(2,2) + b$$

Le résultat est une seule valeur, qui devient un pixel dans la carte de sortie.

On répète cette opération sur toute l'image, ce qui produit une feature map complète.

6. Calcul de la taille de la feature map

La taille de sortie d'une couche de convolution se calcule par la formule :

$$\text{Sortie} = \left\lfloor \frac{W - F + 2P}{S} \right\rfloor + 1$$

où :

- W : taille de l'entrée (ex. 416)
- F : taille du filtre (ex. 3)
- P : padding (nombre de pixels ajoutés sur les bords)
- S : stride (pas de déplacement du filtre)

Exemple :

$$\text{Sortie} = \left\lfloor \frac{416 - 3 + 2 * (1)}{2} \right\rfloor + 1 = 208$$

Donc, une image 416×416 après convolution (filtre 3×3, stride 2, padding 1) devient une carte 208×208.

Si l'on utilise 32 filtres, on obtient 32 cartes → dimension finale : **208×208×32**.

7. Le pooling (MaxPooling)

Après la convolution et la ReLU, on applique souvent une opération de **pooling** pour réduire la taille tout en conservant les informations importantes.

Le plus courant est le **MaxPooling** 2×2 :

- On divise la feature map en blocs 2×2.
- On garde la **valeur la plus élevée** de chaque bloc.

Cela divise la largeur et la hauteur par 2.

Par exemple, 208×208 devient 104×104.

8. Le flatten (aplatir)

Après plusieurs couches convolutionnelles et de pooling, la dernière feature map peut être par exemple de taille **13×13×128**.

On la transforme en un vecteur 1D (appelé flatten) :

$$13 * 13 * 128 = 21632$$

Ce vecteur contient toutes les caractéristiques extraites de l'image et devient l'entrée de la **partie dense (fully connected)** du réseau.

---

9. Les couches denses

Chaque neurone dense reçoit toutes les valeurs du vecteur d'entrée et calcule :

$$z_j = \sum_i w_{ij} x_i + b_j$$

où :

- xi : entrée (valeur du vecteur flatten)
- wij : poids reliant l'entrée i au neurone j
- bj : biais du neurone j

Ces couches combinent toutes les caractéristiques extraites pour produire une décision globale (par exemple : feuille saine ou malade).

---

10. La couche de sortie et Softmax

La dernière couche dense produit un score par classe.

Ces scores sont transformés en probabilités grâce à la fonction Softmax :

$$P(y_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Le modèle choisit la classe ayant la probabilité la plus élevée.

---

11. Entraînement du réseau

Pendant l'apprentissage :

1. **Propagation avant (Forward pass)**

   L'image traverse toutes les couches et produit une prédiction.

2. **Calcul de la perte (Loss)**

   On compare la prédiction au label réel (ex. avec la fonction d'entropie croisée) :

$$\mathcal{L} = -\sum_i y_i \log(\hat{y}_i)$$

Rétropropagation (Backward pass)

On calcule le gradient de la perte par rapport aux poids, puis on les met à jour :

$$w \leftarrow w - \eta, \frac{\partial \mathcal{L}}{\partial w}$$

1. où η est le taux d'apprentissage.

Pendant cette phase, les **poids et biais** apprennent à reconnaître les bons motifs.

12. Inférence

Une fois le modèle entraîné :

- Les poids et biais sont figés.
- Les convolutions détectent les motifs appris.
- Les couches denses combinent ces motifs pour décider de la classe finale.

Le flux est donc :

Image → Convolutions + ReLU → Pooling → Flatten → Denses → Softmax → Décision

13. Interprétation synthétique

- Les **couches convolutionnelles** jouent le rôle d'un "système visuel" qui extrait les caractéristiques locales.
- Le **flatten** convertit ces informations en un vecteur utilisable par un réseau de neurones classique.
- Les **couches denses** fonctionnent comme un "cerveau décisionnel" qui interprète les caractéristiques et produit la prédiction finale.

$$\text{Image} \xrightarrow{\text{Convolution} + \text{ReLU}} \text{Feature maps} \xrightarrow{\text{Pooling}} \text{Vecteur flatten} \xrightarrow{\text{Couches denses}} \text{Softmax} \rightarrow \text{Classe prédite}$$