République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Aboubekr BELKAÏD - TLEMCEN
Faculté des Sciences
Département d'Informatique

TITRE

INTRODUCTION À L'INTELLIGENCE ARTIFICIELLE

Adressé aux étudiants niveau : Licences (L3)

Domaine : Mathématiques et Informatique
Filière : Informatique

Etabli Par :

Asma AMRAOUI

Année Universitaire : 2024 – 2025

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Polycopié de cours

Introduction à l'Intelligence Artificielle

Année universitaire : 2024 - 2025

Préambule

Ce document représente un recueil d'information concernant la gestion de projets informatique.

A qui s'adresse ce document ?

Ce cours s'adresse précisément pour les étudiants en 3ème année Licence du département Informatique de l'université de Tlemcen ; mais il peut en effet être utile à tous les étudiants voulant avoir des notions générales dans le domaine de l'Intelligence Artificielle (IA).

L'intelligence artificielle est aujourd'hui au cœur des évolutions technologiques majeures qui transforment nos sociétés, nos industries et nos modes de vie. Comprendre les concepts, les méthodes ainsi que les applications de l'IA est devenu indispensable pour tout informaticien ou ingénieur moderne.

Ce polycopié, intitulé **« Introduction à l'intelligence artificielle »**, a pour objectif de fournir aux étudiants une base solide et structurée sur les principaux fondements de l'IA, en alternant aspects théoriques et illustrations pratiques.

Structure du manuscrit

Le manuscrit est organisé en huit chapitres progressifs qui couvrent les notions essentielles :

Le premier chapitre est une introduction à l'intelligence artificielle où une présentation des origines, des objectifs et des domaines d'application de l'IA sont présentés.

Ensuite, dans le deuxième chapitre nous allons présenter la différence entre donnée, information et connaissance et par la suite voir les différents paradigmes de représentation des connaissances.

Dans le troisième chapitre, nous découvrons les systèmes experts : des systèmes capables de simuler le raisonnement d'un expert humain dans des domaines spécifiques, à travers des règles et des bases de connaissances.

Le quatrième chapitre va étudier les différentes méthodes de résolution de problèmes où nous commencerons par les méthodes informées et nous passerons par la suite aux méthodes heuristiques non informées.

Le cinquième chapitre décrit un cas particulier de problèmes à savoir : les problèmes de satisfaction de contraintes (CSP). Nous commencerons par présenter les CSP ensuite comment les modéliser et enfin quelques méthodes de résolution.

L’apprentissage automatique est abordé dans le 6ème chapitre où une introduction aux concepts de base du machine learning est décrite, les différents types existants et les critères de performance.

Après cela, le chapitre 7 va se concentrer sur les réseaux de neurones où nous allons définir la structure de ce genre de réseaux, la notion de perceptron et bien sur quelques définitions générales sur l’apprentissage profond.

Le dernier chapitre est plutôt une réflexion sur les impacts sociétaux, éthiques et philosophiques de l’intelligence artificielle, ainsi que sur les enjeux liés à la régulation et à la responsabilité.

Ce polycopié se veut accessible, pédagogique et illustré d’exemples concrets afin de faciliter l’assimilation des concepts. Il constitue une première immersion dans le vaste domaine de l’intelligence artificielle et vise à préparer les étudiants à des études plus avancées ou à des applications pratiques en entreprise ou en recherche.

Table des matières

Liste des figures

Liste des tableaux

Liste des acronymes

A*	Algorithme A étoile
AC -3	Arc Consistency Algorithm 3
Adam	Adaptive Moment Estimation (optimiseur avancé pour l'apprentissage profond)
BFS	Breadth First Search (recherche en largeur d'abord)
C	Langage de programmation C
CNN	*Convolutional Neural Network (*Réseau de Neurones Convolutif)
CSP	Constraint Satisfaction Problem (Problème de satisfaction de contraintes)
DFS	Depth First Search (recherche en profondeur d'abord)
DQN	*Deep Q-Networks (renforcement base sur apprentissage profond)*
FN	*Faux négatif*
FP	*Faux positif*
GPT	Generative Pre-trained Transformer (modèle de langage basé sur les Transformers)
GPU	Graphics processing unit (processeur graphique)
GPS	Global Positioning System (système de positionnement par satellites)
IA	Intelligence artificielle
IDDFS	*Iterative Deepening Depth-First Search* (recherche en profondeur itérative)
Lisp	List Programming
MLP	Multi Layer Perceptron (Perceptron multi couches)
NLP	Natural Language Processing (traitement de la langue naturelle)
PCA	*Principal Component Analysis* (*Analyse en Composantes Principales*
Prolog	Programmation logique
ReLu	Rectified Linear Unit (Fonction d'activation)
ROC	*Receiver Operating Characteristic* (courbe de performance d'un classifieur binaire)
RNN	*Recurrent Neural Network* (*Réseau de Neurones Récurrent*).
SE	Système expert
SGD	*Stochastic Gradient Descent* (*Descente de Gradient Stochastique*)
SVM	Support Vector Machine (*Machine à Vecteurs de Support)*
VN	Vrai négatif
VP	Vrai positif
XAI	*Explainable Artificial Intelligence* (*Intelligence Artificielle Explicable*)

Chapitre 1 :
Introduction à l'Intelligence Artificielle

1.1 Introduction

Depuis toujours, l'être humain s'interroge sur sa propre intelligence. Pourquoi sommes-nous capables de raisonner, de faire des choix, d'apprendre de nos erreurs ou de comprendre un langage ? Ces facultés, longtemps considérées comme exclusivement humaines, sont aujourd'hui au cœur d'un domaine scientifique fascinant : l'intelligence artificielle.

L'idée de créer des machines capables de penser, ou du moins d'imiter certains aspects de l'intelligence humaine, peut sembler audacieuse. Pourtant, elle ne date pas d'hier. Dès les débuts de l'informatique, certains chercheurs se sont demandé s'il serait possible de programmer des ordinateurs pour qu'ils prennent des décisions, apprennent de l'expérience ou dialoguent avec nous. C'est ainsi qu'est née l'intelligence artificielle, un domaine à la frontière de plusieurs disciplines : informatique, mathématiques, logique, psychologie, et même philosophie.

Aujourd'hui, l'IA est partout : dans nos téléphones, nos voitures, les hôpitaux, les entreprises, les réseaux sociaux… Elle influence nos choix, nous assiste dans nos tâches quotidiennes, et parfois même nous surprend par ses capacités. Derrière ces technologies impressionnantes se cachent des idées, des modèles, des algorithmes… mais aussi des questionnements éthiques, des limites techniques, et des débats de société.

Ce chapitre a pour objectif de te guider dans la découverte de l'intelligence artificielle. Il ne s'agit pas seulement d'apprendre des définitions ou des formules, mais de comprendre les grandes approches de l'IA, son histoire, ses réussites et ses limites.

1.2 Définitions de l'Intelligence Artificielle

Le concept d'intelligence artificielle recouvre une diversité de définitions issues de différentes approches disciplinaires. D'une manière générale, l'IA peut être vue comme la science qui cherche à automatiser les fonctions cognitives humaines : comprendre, apprendre, raisonner, résoudre des problèmes, prendre des décisions. Cette perspective implique de concevoir des machines capables non seulement d'exécuter des tâches, mais aussi de s'adapter à leur environnement de manière autonome.

Bellman, en 1978 [3], la définit comme : l'«automatisation des activités associées au raisonnement humain, telles que la décision, la résolution de problèmes, l'apprentissage... ». Cette définition met en évidence le cœur de l'IA : la capacité à reproduire des mécanismes de pensée.

De même, John McCarthy [2], l'un des fondateurs de l'IA, propose cette définition :

L'intelligence artificielle correspond à la construction de programmes informatiques capables d'accomplir des tâches qui requièrent une forme d'intelligence généralement attribuée à l'humain.

Dans la littérature, on distingue généralement quatre grandes manières de concevoir l'intelligence artificielle, selon qu'on s'attache à penser comme les humains**,** agir comme les humains**,** penser rationnellement**,** ou agir rationnellement.

Système qui pense comme des humains	**Système qui pense rationnellement**
Système qui agit comme des humains	**Système qui agit rationnellement**

1.2.1 Penser comme les humains

L'approche cognitive vise à penser comme les humains. Elle repose sur l'idée que les machines peuvent être construites pour imiter les mécanismes mentaux humains. Cette démarche s'appuie sur les sciences cognitives, un domaine interdisciplinaire qui mêle psychologie, neuroscience, intelligence artificielle et modélisation informatique. Les chercheurs construisent des modèles algorithmiques inspirés du fonctionnement du cerveau humain, puis les confrontent à des données issues d'expériences empiriques.

Un exemple notable de ce type d'approche est AlphaGo, un système développé par DeepMind, qui a battu le champion du monde de Go en 2016 grâce à une stratégie d'apprentissage inspirée du raisonnement humain.

Figure 1.1: Alpha Go

1.2.2 Penser rationnellement

À côté de cette approche, on trouve celle qui consiste à penser rationnellement. Il s'agit ici de formaliser les lois de la pensée logique, héritées de philosophes comme Aristote, Leibniz ou encore Boole. L'objectif est de reproduire la capacité de raisonnement logique à l'aide

de formules mathématiques. Si l'on sait que « tous les hommes sont mortels » et que « Socrate est un homme », alors la machine doit pouvoir conclure que « Socrate est mortel ». Cette approche repose fortement sur la logique formelle, comme on la retrouve dans les systèmes experts ou les langages logiques comme Prolog. Toutefois, traduire les connaissances du monde réel dans des systèmes purement logiques s'avère complexe en raison de l'incertitude inhérente à ces contextes.

1.2.3 Agir rationnellement

Une troisième approche propose d'agir rationnellement. Elle introduit la notion d'agent rationnel, c'est-à-dire une entité capable de percevoir son environnement et d'y réagir de manière à maximiser ses chances d'atteindre ses objectifs. Dans cette perspective, l'intelligence consiste à adopter les meilleures décisions possibles à partir des données disponibles. On retrouve cette idée dans les voitures autonomes, qui ajustent leur comportement à la circulation, ou dans les systèmes de trading algorithmique qui réagissent en temps réel à des fluctuations de marché.

1.2.4 Agir comme les humains

Enfin, l'approche qui consiste à agir comme les humains s'appuie sur le fameux « Test de Turing » proposé en 1950 [1]. Dans ce test, une machine réussit si, dans le cadre d'une conversation écrite, un interrogateur humain n'est pas capable de distinguer l'humain de l'IA. Cette approche met l'accent sur la simulation du comportement humain sans forcément chercher à reproduire les mécanismes internes du raisonnement. Aujourd'hui, de nombreux chatbots parviennent à tromper temporairement leurs utilisateurs, illustrant les progrès de l'IA dans ce domaine.

Une IA réussit le Test de Turing si un humain ne peut pas la distinguer d'un autre humain lors d'une conversation.

Ces différentes définitions et approches de l'intelligence artificielle ne s'excluent pas ; au contraire, elles se complètent. Chacune apporte un éclairage spécifique sur ce que signifie « être intelligent » et sur la manière dont une machine peut l'être. Dans la pratique, les systèmes d'IA modernes combinent souvent plusieurs de ces approches pour produire des résultats plus performants et plus réalistes.

1.3 Types d'intelligence artificielle

Quand on parle d'intelligence artificielle, on a parfois l'image d'une machine ultra-intelligente capable de tout faire, du diagnostic médical à la poésie. Mais en réalité, il existe plusieurs types d'IA, avec des niveaux de complexité et d'autonomie très différents. Pour

mieux comprendre ce domaine, il est essentiel de distinguer les grands types d'IA que les chercheurs et les ingénieurs conçoivent aujourd'hui.

1.3.1 Intelligence artificielle faible

C'est le type d'IA le plus courant et le plus utilisé actuellement. On l'appelle « faible » non pas parce qu'elle est inefficace, mais parce qu'elle est spécialisée dans une seule tâche. Une IA faible peut être très performante dans un domaine précis, mais elle est incapable de sortir de ce cadre. Un assistant vocal qui répond à des commandes, un système de recommandation sur Netflix ou un programme qui joue aux échecs sont tous des exemples d'IA faibles. Ils n'ont aucune conscience ni compréhension générale : ils appliquent des règles, des algorithmes, ou des modèles appris pour accomplir leur mission.

1.3.2 Intelligence artificielle forte

L'IA forte est plus ambitieuse. Elle vise à créer une machine dotée d'une intelligence comparable à celle d'un être humain, capable de raisonner, d'apprendre, de comprendre et d'interagir dans des situations variées, comme nous le faisons naturellement. Une IA forte pourrait, en théorie, comprendre le sens de ce qu'elle fait, s'adapter à de nouveaux contextes sans être reprogrammée, et même faire preuve de bon sens. Ce type d'IA n'existe pas encore, mais il fait l'objet de nombreuses recherches, et alimente souvent les récits de science-fiction.

1.3.3 Intelligence artificielle super intelligente

C'est une hypothèse théorique encore plus lointaine : celle d'une IA qui dépasserait largement les capacités humaines dans tous les domaines. Une telle IA serait capable d'inventer, de créer, de raisonner à une vitesse inimaginable. Si elle devait voir le jour, cette super intelligence poserait des défis éthiques et de sécurité majeure : comment la contrôler ? Quelles valeurs devrait-elle suivre ? Ces questions sont débattues dans le domaine de l'IA éthique et de la gouvernance technologique.

Le tableau 1.1 présente une comparaison entre les 3 types d'IA.

IA Faible	IA Forte	IA super intelligente
❑ Les programmes de ce type ne visent pas à évoluer. ❑ Un programme ne « pense » pas, il exécute. ❑ Très performants dans leur domaine. **Google Assistant, Siri, Alexa (uniquement reconnaissance vocale et recherche d'infos).**	❑ Approche la plus similaire au comportement humain. ❑ Capable d'éprouver une réelle conscience de soi, de ressentir de vrais sentiments et comprendre ce qui la pousse à faire telle ou telle action. ❑ Nécessite de savoir apprendre et modifier son comportement. ❑ Capable de raisonner, apprendre et comprendre comme un humain sur tous les sujets. **Encore hypothétique**	❑ Dépasserait largement l'intelligence humaine.

Tableau 1.1 : Types de l'IA

Pour mieux comprendre les différentes façons de concevoir l'intelligence artificielle, il est utile d'aborder une autre distinction majeure : celle qui oppose l'IA symbolique à l'IA connexionniste. Ces deux approches représentent des visions très différentes de ce que signifie « penser » ou « être intelligent ».

1.4 IA symbolique vs IA connexionniste

1.4.1 IA symbolique : raisonner avec des règles

L'IA symbolique est l'approche historique, dominante pendant les premières décennies de la recherche en intelligence artificielle. Elle repose sur une idée simple : l'intelligence peut être modélisée à l'aide de symboles et de règles logiques. Dans cette perspective, un système intelligent manipule des symboles (comme des mots ou des concepts) selon des règles bien définies (comme des lois logiques).

L'IA symbolique est la plus ancienne. Elle fonctionne à partir de règles logiques et explicites : on décrit des faits, des conditions, des lois (« si… alors… ») et la machine les applique pour déduire des conclusions. C'est le cas des systèmes experts ou des programmes écrits en Prolog. Cette approche est appréciée pour sa transparence : on peut

expliquer pourquoi la machine a pris telle ou telle décision. Mais elle a aussi des limites : il est souvent difficile, voire impossible, de tout formaliser sous forme de règles, surtout dans des situations floues ou ambiguës.

C'est ainsi que fonctionnent les systèmes experts, qui appliquent une base de connaissances et des règles conditionnelles du type « si… alors… » pour tirer des conclusions ou prendre des décisions. L'IA symbolique est transparente : on peut suivre le raisonnement du système étape par étape. Elle est donc utile dans des domaines où l'explicabilité est importante, comme la médecine ou le droit.

Cependant, cette approche a ses limites. Elle suppose qu'on peut formaliser toutes les connaissances et tous les raisonnements sous forme de règles explicites. Or, beaucoup de situations réelles sont floues, ambigües, ou dépendent de contextes trop complexes pour être codifiés à la main.

1.4.2 IA connexionniste : apprendre par l'expérience

Face à ces limites, une autre approche a émergé : l'IA connexionniste. Inspirée du fonctionnement du cerveau humain, elle repose sur des réseaux de neurones artificiels capables d'apprendre à partir de données. Ces réseaux ne manipulent pas des symboles explicites, mais ajustent des poids numériques entre des neurones artificiels pour extraire des régularités.

L'IA connexionniste est à la base de l'apprentissage automatique (machine learning) et du deep learning, qui ont connu un essor spectaculaire depuis les années 2010 [4]. Grâce à ces techniques, les machines peuvent apprendre à reconnaître des visages, traduire des langues, jouer à des jeux complexes, ou générer du texte et des images… sans qu'on leur fournisse explicitement de règles.

L'IA connexionniste, en revanche, s'inspire du cerveau humain. Elle repose sur des réseaux de neurones artificiels capables d'apprendre à partir de données. Elle ne suit pas des règles prédéfinies, mais détecte des modèles ou des corrélations en traitant de grandes quantités d'informations. Cette approche est à la base des progrès récents de l'IA, notamment en vision par ordinateur, reconnaissance vocale ou traduction automatique. Le revers de la médaille, c'est que ces modèles sont souvent opaques : on parle de « boîte noire », car il est difficile d'interpréter leur fonctionnement interne.

Cette approche est très puissante, mais elle présente aussi des défis. Les réseaux de neurones sont souvent opaques : on ne sait pas toujours expliquer pourquoi une IA a pris telle ou telle décision. On parle parfois de « boîte noire ». De plus, ces modèles nécessitent

de grandes quantités de données pour apprendre efficacement, ce qui soulève des questions de vie privée, de biais et de représentativité.

Le tableau 1.2 présente une comparaison entre IA symbolique et IA connexionniste.

Critère	**IA Symbolique**	**IA Connexionniste**
Basé sur	Règles et logique	Neurones artificiels
Approche	Déduction et raisonnement	Apprentissage automatique
Interprétabilité	Explicable	Boîte noire
Exemples d'usage	Systèmes experts, moteurs de règles	ChatGPT, reconnaissance d'image
Avantages	Compréhensible, bon pour problèmes structurés	Apprend, s'adapte à de nouveaux contextes
Inconvénients	Difficile pour les situations floues	Exige beaucoup de données et de puissance de calcul

Tableau 1.2 : IA symbolique vs IA connexionniste

Aujourd'hui, les chercheurs tendent à combiner les deux approches. On parle alors d'IA hybride : une IA qui utilise à la fois des mécanismes d'apprentissage automatique pour s'adapter, et des représentations symboliques pour structurer et expliquer ses raisonnements. Cette voie semble prometteuse, car elle cherche à tirer parti du meilleur des deux mondes.

Aujourd'hui, ces deux approches ne s'opposent plus autant qu'avant. De nombreux chercheurs travaillent sur des systèmes hybrides qui cherchent à combiner la rigueur des modèles symboliques avec la capacité d'apprentissage des modèles connexionnistes. Par exemple, on peut utiliser l'apprentissage automatique pour extraire des règles à partir de données, ou renforcer l'interprétabilité d'un réseau de neurones avec des représentations symboliques.

1.5 Historique de l'intelligence artificielle

L'intelligence artificielle n'est pas née d'un jour. Elle a évolué à travers des décennies d'expérimentations, de réussites, mais aussi de doutes. Comprendre son histoire, c'est mieux saisir comment les idées d'hier ont façonné les technologies d'aujourd'hui.

- **Les précurseurs de l'IA (1943–1956)**

L'histoire de l'IA commence bien avant qu'on ne parle d'ordinateurs intelligents. En 1943, Warren McCulloch et Walter Pitts publient un article fondateur dans lequel ils proposent un modèle mathématique du neurone [5]. Leur travail jette les bases de ce qu'on appellera plus tard les réseaux de neurones artificiels. En parallèle, le mathématicien Alan Turing, figure emblématique de l'informatique, propose en 1950 le Test de Turing, une expérience de pensée destinée à évaluer si une machine peut se faire passer pour un humain dans une conversation. Ces idées sont encore très théoriques, mais elles posent les premières pierres du domaine.

- **Naissance de l'IA**

En 1956, l'intelligence artificielle devient officiellement un champ de recherche à part entière. Lors de la conférence de Dartmouth, aux États-Unis, un groupe de scientifiques, dont John McCarthy, Marvin Minsky, Claude Shannon et Nathaniel Rochester, se réunit pour lancer ce qu'ils appellent le « projet d'été sur l'intelligence artificielle » [2]. C'est à ce moment-là que le terme *artificial intelligence* est utilisé pour la première fois.

- **Espoirs grandissants (1956–1970)**

Dans les années qui suivent, les chercheurs sont pleins d'optimisme. Ils conçoivent des programmes capables de démontrer des théorèmes mathématiques ou de résoudre des énigmes logiques. Des systèmes comme Logic Theorist ou General Problem Solver impressionnent par leur capacité à reproduire certaines formes de raisonnement. On pense alors que, dans quelques années, les machines seront capables de rivaliser avec l'intelligence humaine. Le premier robot intelligent, Shakey, est conçu en 1967 [1] : il peut percevoir son environnement, prendre des décisions simples, et se déplacer de façon autonome… en théorie du moins, car dans la pratique, il lui fallait parfois une heure de calcul pour faire un simple mouvement.

Figure 1.2 : Shakey premier robot

- **Le retour à la réalité : premières déceptions (1970–1980)**

Les années 1970 marquent une période de désillusion. Les ambitions initiales se heurtent à des limites techniques majeures : puissance de calcul insuffisante, mémoire limitée, difficulté à formaliser les connaissances du monde réel. Les traductions automatiques, par exemple, s'avèrent bien plus complexes qu'espéré : comprendre une phrase nécessite une compréhension fine du sens, du contexte, et des nuances culturelles. En 1966, le

gouvernement américain décide de couper les financements pour la recherche en traduction automatique, marquant le début de ce qu'on appellera plus tard un hiver de l'IA [4].

En 1969 [2], Marvin Minsky et Seymour Papert publient un livre qui critique les limites des réseaux de neurones de l'époque. Leur analyse, bien que techniquement juste, contribue à détourner la recherche de cette piste pourtant prometteuse.

- **Les systèmes experts : un nouvel espoir (1980–1987)**

L'IA rebondit dans les années 1980 grâce à une nouvelle approche : les systèmes experts [6]. Ceux-ci fonctionnent sur la base de règles logiques et de bases de connaissances structurées. Le système MYCIN, par exemple, aide à diagnostiquer des infections sanguines avec un niveau de performance proche de celui d'un médecin confirmé. Le succès de ces systèmes dans des domaines spécifiques (médecine, chimie, finance) redonne confiance à la communauté scientifique et attire l'attention des entreprises.

Mais là encore, les limites apparaissent rapidement : ces systèmes sont difficiles à maintenir, rigides, et nécessitent un travail fastidieux pour intégrer chaque nouvelle règle. Une fois de plus, l'enthousiasme retombe.

- **Le retour des réseaux de neurones et l'apprentissage (fin des années 1980 – 2000)**

C'est à la fin des années 1980 qu'un tournant s'opère avec la redécouverte d'une méthode d'apprentissage appelée rétropropagation du gradient (*backpropagation*) [4]. Cette technique permet enfin aux réseaux de neurones d'apprendre des fonctions complexes à partir de données. Les résultats sont encore modestes, mais les fondations du machine learning moderne sont posées.

- **L'explosion de l'IA moderne (depuis 2010)**

À partir des années 2010, l'IA entre dans une nouvelle ère grâce à trois facteurs clés : la puissance de calcul (GPU), la masse de données disponibles (big data), et l'amélioration des algorithmes. Le deep learning, ou apprentissage profond, permet à des modèles comme AlexNet (2012) de dépasser les performances humaines sur certaines tâches de reconnaissance d'image. L'IA devient soudainement performante, concrète, et omniprésente [10].

Les exploits s'enchaînent : en 2016, AlphaGo, développé par DeepMind, bat le champion du monde de Go, un jeu réputé bien plus complexe que les échecs. En 2020, GPT-3, puis ChatGPT en 2022, révolutionnent notre rapport au langage et à la créativité. L'IA

générative, capable de produire du texte, des images, du code ou de la musique, fascine autant qu'elle inquiète.

Aujourd'hui, l'intelligence artificielle n'est plus une promesse : elle transforme déjà nos sociétés, nos métiers, nos manières d'apprendre et de communiquer. Mais elle continue aussi de poser de nombreuses questions — éthiques, politiques, philosophiques — qui nous obligent à réfléchir collectivement à la place que nous voulons lui accorder.

Figure 1.3 : Historique de l'IA

1.6 Domaines d'application de l'intelligence artificielle

Si l'intelligence artificielle fascine autant, c'est parce qu'elle ne se limite pas aux laboratoires de recherche. Elle est déjà bien présente dans notre vie quotidienne, souvent sans qu'on s'en rende compte. De la médecine à la finance, en passant par les transports, le divertissement ou encore l'agriculture, l'IA s'impose comme un outil de transformation profonde de nos sociétés.

- **Santé et médecine**

C'est sans doute l'un des domaines où l'IA apporte le plus d'espoir. Grâce à des algorithmes d'apprentissage, les machines sont capables d'analyser des milliers d'images médicales pour détecter des anomalies avec une précision équivalente et parfois supérieure à celle des spécialistes humains. Des systèmes comme IBM Watson ont été utilisés pour proposer des diagnostics ou des plans de traitement personnalisés. L'IA intervient aussi en chirurgie assistée, dans la conception de médicaments ou dans la surveillance à distance des patients.

- **2. Transports et mobilité**

Les voitures autonomes ne relèvent plus de la science-fiction. Des entreprises comme Tesla, Waymo (filiale de Google) ou encore Baidu testent des véhicules capables de se déplacer en ville, de détecter les piétons, de lire les panneaux et de réagir en temps réel à leur environnement. L'IA optimise aussi les trajets, la gestion du trafic, ou encore la logistique dans les aéroports et les ports.

- **3. Finance et économie**

Dans le secteur financier, l'IA est utilisée pour détecter les fraudes, prédire les fluctuations des marchés, ou encore automatiser des décisions d'investissement. L'IA est aussi présente dans les banques, où elle aide à analyser les profils de risque ou à recommander des produits adaptés à chaque client.

- **4. Divertissement et réseaux sociaux**

Les plateformes de streaming audio ou vidéo ainsi que les réseaux sociaux utilisent des systèmes de recommandation sophistiqués qui analysent ton comportement pour te proposer du contenu qui te plaît. Dans les jeux vidéo, l'IA contrôle les personnages non-joueurs, adapte la difficulté en fonction de tes performances, ou crée des environnements réalistes.

- **5. Éducation**

L'intelligence artificielle s'introduit progressivement dans le domaine éducatif. Des tutoriels intelligents sont capables d'évaluer le niveau d'un élève, de repérer ses lacunes, et de lui proposer des exercices personnalisés. Certains outils adaptent le rythme d'apprentissage selon le profil de l'apprenant. D'autres assistent les enseignants dans la correction automatique, la génération de contenu ou l'analyse des résultats.

- **6. Industrie, agriculture, énergie**

Dans les usines, l'IA pilote des bras robotisés, surveille les chaînes de production, anticipe les pannes ou optimise les flux logistiques. En agriculture, elle aide à prévoir les rendements, détecter les maladies des plantes ou gérer l'irrigation de manière intelligente. Dans le secteur de l'énergie, l'IA participe à la gestion des réseaux électriques, à la prévision de la demande ou à l'optimisation des ressources renouvelables.

1.7 Conclusion

Dans ce chapitre, nous avons expliqué ce qu'est l'IA, ses grands types, son historiques, ses domaines d'application. L'IA n'est plus un simple concept théorique réservé aux chercheurs ou aux auteurs de science-fiction. Elle est devenue une réalité technologique qui transforme profondément nos sociétés, nos métiers et nos habitudes. Comprendre l'IA, ce n'est pas seulement maîtriser des algorithmes ou des outils informatiques, c'est aussi apprendre à penser ses usages, ses limites et ses implications.

L'IA est un champ en constante évolution. Ce que nous considérons aujourd'hui comme avancé pourra sembler basique demain. C'est pourquoi il est essentiel de garder un esprit critique, curieux et responsable. Que l'on devienne ingénieur, chercheur, utilisateur ou simple citoyen, chacun a un rôle à jouer dans le monde façonné par ces intelligences que nous construisons.

Dans le chapitre suivant, nous allons voir que l'IA doit avant tout permettre aux systèmes de traiter et d'interpréter l'information de manière structurée et intelligible et que les connaissances doivent être représentées de manière adéquate pour que ces systèmes puissent effectuer des raisonnements complexes.

La représentation des connaissances est une étape cruciale dans la conception de l'IA, car elle détermine la capacité de la machine à comprendre le monde qui l'entoure, à extraire des informations pertinentes et à en déduire des actions appropriées.

Chapitre 2 :
Représentation des connaissances

2.1 Introduction

Un système intelligent ne se contente pas de traiter des données ; il doit être capable de comprendre, d'interpréter et de manipuler des connaissances. La représentation des connaissances est un domaine central de l'IA, qui consiste à modéliser formellement tout ce que on sait : les objets, leurs propriétés, les relations, les règles, les exceptions afin que des programmes informatiques puissent utiliser ce savoir pour prendre des décisions ou faire des inférences.

Nous explorerons donc différentes approches de cette représentation permettant à l'IA de manipuler les données de façon logique et significative. Ce processus de représentation est essentiel pour que l'intelligence artificielle puisse s'adapter à des contextes variés et résoudre des problèmes complexes de manière autonome.

Dans ce chapitre, nous allons commencer par clarifier quelques notions fondamentales : la différence entre donnée, information et connaissance, ensuite nous explorerons les différents formalismes de représentation de connaissances : logiques et non logique.

2.2 Donnée, information, connaissance

Avant de pouvoir représenter la connaissance dans un système intelligent, il est essentiel de comprendre ce que recouvrent exactement les notions de donnée, information et connaissance [7]. Ces trois concepts sont liés, mais ils ne désignent pas la même chose.

2.2.1 Donnée (data)

Une donnée est une valeur brute, issue d'une mesure ou d'une observation. Elle est objective mais sans signification en elle-même. Par exemple :

- `37,2` est une donnée.
- `Rouge`, `15`, `true`… sont aussi des données.

Ces données seules ne permettent pas de prendre une décision. Elles doivent être interprétées pour devenir utiles.

2.2.2 Information (information)

L'information correspond à une donnée à laquelle on a attribué un sens dans un contexte donné. Elle naît lorsqu'on relie la donnée à une situation concrète.
Exemple :

- « 37,2°C » devient une information si on sait qu'il s'agit de la température corporelle d'un patient.

L'information répond généralement aux questions : *quoi*, *où*, *quand*. Elle est déjà plus riche que la donnée brute, mais elle ne permet pas encore de raisonner à un niveau complexe.

2.2.3 Connaissance (knowledge)

La connaissance est le niveau supérieur. C'est le résultat d'un traitement des informations, d'une expérience, ou d'un apprentissage. Elle permet d'agir, de décider, de résoudre un problème.

Exemple :

- Savoir que « une température corporelle normale chez un adulte se situe entre 36,5°C et 37,5°C » est une connaissance médicale.
- En analysant plusieurs cas similaires, on peut aussi en déduire des règles, des exceptions, des corrélations…

La connaissance répond aux questions : *comment* et *pourquoi*. Elle permet à un système d'IA non seulement de réagir, mais aussi de raisonner, planifier, adapter ses réponses.

2.3 Représentation des connaissances

Un système intelligent, qu'il s'agisse d'un assistant vocal, d'un robot autonome ou d'un programme médical, ne peut pas se contenter de stocker des informations. Il doit comprendre, déduire et agir.

Représenter les connaissances, c'est permettre à une machine de faire des inférences. Autrement dit, d'en déduire de nouvelles informations à partir de celles qu'elle possède déjà.

Exemple :

Si un système connaît les deux faits suivants :

- Tous les chats sont des mammifères
- Garfield est un chat

Alors il doit pouvoir en déduire :

- Garfield est un mammifère

Sans une représentation formelle de ces connaissances, cette déduction serait impossible.

Représenter les connaissances permet de donner du sens à ce monde pour la machine. Cela revient à construire un modèle mental du monde dans lequel elle évolue :

- Quels objets existent ?
- Quelles sont leurs propriétés ?
- Quelles relations les relient ?
- Quelles règles s'appliquent ?

Cette structuration est essentielle pour qu'un agent intelligent puisse naviguer dans un environnement, poser un diagnostic, jouer à un jeu, ou converser.

Une bonne représentation des connaissances permet à la machine d'être plus explicable, plus interprétable. C'est un enjeu majeur dans les domaines sensibles comme la santé, la justice, la finance ou l'éducation. Une IA bien conçue doit pouvoir expliquer ses décisions : pourquoi a-t-elle recommandé tel médicament ? Pourquoi a-t-elle détecté une fraude ? Pourquoi refuse-t-elle une commande vocale ?

2.4 Formalismes de représentation de la connaissance

Représenter la connaissance, c'est bien. Mais il faut aussi pouvoir la manipuler, la partager, en tirer des conclusions. Pour cela, on utilise des formalismes : des langages bien définis, avec des règles précises, capables de structurer l'information de manière exploitable pour une machine [7].

Un formalisme, c'est un peu comme une langue pour la machine : il lui permet de comprendre et de raisonner. Il existe plusieurs familles de formalismes, que l'on peut regrouper en deux grandes catégories : les approches logiques et les approches non logiques**.**

2.4.1 Approches logiques

a. Logique des propositions

Son vocabulaire comprend des éléments de base tels que les connecteurs logiques comme ∧ (et), ∨ (ou), ¬ (non), → (implique) ; ainsi que des variables propositionnelles qui représentent des affirmations pouvant être vraies ou fausses. Ce langage formel constitue un outil puissant pour modéliser et analyser des raisonnements complexes dans divers domaines de l'intelligence artificielle et de la logique.

Exemple 1:

- "S'il pleut, alors je prends un parapluie" peut se représenter par :
  `pluie → parapluie`

Exemple 2 :

- Si Ahmed est rentré chez lui, alors Omar est allé au cinéma.
- Meriem est à la bibliothèque ou Ahmed est rentré chez lui.
- Si Omar est allé au cinéma, alors Meriem est à la bibliothèque ou Ahmed est rentré chez lui.
- Meriem n'est pas à la bibliothèque et Omar est allé au cinéma.
- Ahmed est rentré chez lui.

Notons:

M la proposition Meriem est à la bibliothèque
O la proposition Omar est allé au cinéma
A la proposition Ahmed est rentré chez lui

- $A \Rightarrow O$
- $M \vee A$
- $O \Rightarrow (M \vee A)$
- $\neg M \wedge O$
- A

Cette logique est simple à mettre en œuvre, facile à automatiser et donne de bons résultats pour l'apprentissage des raisonnements formels ; mais assez limitée. En effet, elle permet de décrire uniquement des constructions simples du langage, se concentrant principalement sur des opérations booléennes effectuées sur des propositions. Grâce à la logique propositionnelle, il est possible d'étudier la valeur de vérité de formules relativement peu expressives. Toutefois, elle ne permet pas de décrire des relations complexes entre plusieurs variables et est donc insuffisante pour représenter des procédés de langage couramment utilisés en informatique, en linguistique ou en mathématiques.

Les limitations de cette logique la rendent inadaptée pour modéliser des systèmes plus sophistiqués qui nécessitent une représentation plus riche et détaillée des connaissances et des relations.

b. La logique des prédicats

Ici, on passe à un niveau supérieur. On ne manipule plus des phrases simples, mais des prédicats, c'est-à-dire des affirmations qui portent sur des objets, avec des variables.

Exemple :

- `aime(pierre, chocolat)` veut dire "Pierre aime le chocolat".
- `∀x(homme(x) → mortel(x))` se lit "Tous les hommes sont mortels".

On peut aussi introduire des quantificateurs :

- ∀ (pour tout),
- ∃ (il existe).

Cette logique est très expressive. Elle permet de modéliser une grande variété de situations complexes et elle est à la base de nombreux systèmes experts.

Exemple :

- Omar est plus grand que Meriem
  G(o, m)
- Ahmed a vu Meriem et elle ne l'a pas vu

V (a,m) ∧ ¬V (m, a)

- Si Omar est un homme, alors il est mortel
  H(o) → M(o)
- Un chat est entré
  ∃x(C(x) ∧ E(x))
- Certains enfants ne sont pas malades
  ∃x(E(x) ∧ ¬M(x))
- Tous les éléphants ont une trompe
  ∀x(E(x) → T(x))
- Tous les hommes n'aiment pas Meriem
  ∀x(H(x) → ¬A(x,m))

La logique des prédicats, contrairement à la logique propositionnelle, est beaucoup plus expressive. Elle permet de modéliser des règles générales ainsi que des relations complexes, offrant une flexibilité accrue pour décrire des systèmes plus sophistiqués. C'est d'ailleurs la base théorique de langages de programmation tels que Prolog, utilisés dans des domaines comme l'intelligence artificielle et la logique déclarative.

Cependant, la logique des prédicats est plus complexe à automatiser, ce qui peut rendre son implémentation plus difficile, notamment dans des systèmes de grande envergure. De plus, elle peut devenir rapidement difficile à manipuler lorsque les systèmes sont très grands, nécessitant des ressources et des techniques supplémentaires pour rester efficace et gérer la complexité croissante des relations modélisées.

2.4.2 Approches non logiques

Quand le monde devient trop flou, trop vaste ou trop variable pour être entièrement exprimé par des règles logiques, on a recours à d'autres outils.

a. Réseaux sémantiques

Un réseau sémantique est un graphe orienté et étiqueté. Une sémantique est associée par le biais des relations, et un réseau représente une conjonction de formules logiques associées à chacun des arcs [7].

Un réseau sémantique comprend des nœuds (objets), des arcs (relation entre deux objets) et des étiquettes d'arcs. Un nœud est représenté par un rectangle, un cercle ou une ellipse. Un arc relie un nœud source à un nœud cible.

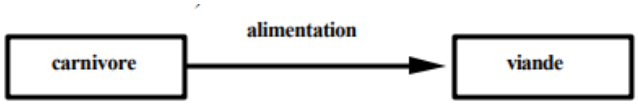
Les réseaux sémantiques utilisent généralement deux relations très particulières :

- **est_un :** relation entre un individu et une classe exprimant l'appartenance.
- **sorte_de :** relation entre deux classes exprimant l'inclusion.

Exemple 1:

- Garfield est un Chat.
- Les Chats sont des félins.

- Les carnivores mangent de la viande.



Exemple 2 :

Un félin est un carnivore. Un carnivore est un animal qui a les yeux dirigés vers l'avant et qui mange de la viande. Les pattes d'un félin ont des griffes à leurs extrémités. Un félin est un mammifère. Grisou et Garfield sont des chats. Grisou est un chat mâle. Léo est un lion. Les chats et les lions sont des félins. Un cheval est un équidé. Un équidé est un mammifère.

Nous allons commencer par chercher les relations entre les objets et donc chaque phrase va être transformée comme suit :

- félin **sorte_de** carnivore
- carnivore **direction des yeux** avant
- carnivore **alimentation** viande
- félin **extrémités pattes** griffes
- félin **sorte_de** mammifère
- Grisou **est_un** chat
- Garfield **est_un** chat
- Grisou **sexe** mâle
- Léo **est_un** lion
- chat **sorte_de** félin
- lion **sorte_de** félin
- cheval **sorte_de** équidé
- équidé **sorte_de** mammifère

Par la suite, nous pourrons dessiner notre réseau sémantique comme le montre la figure ci-dessous :

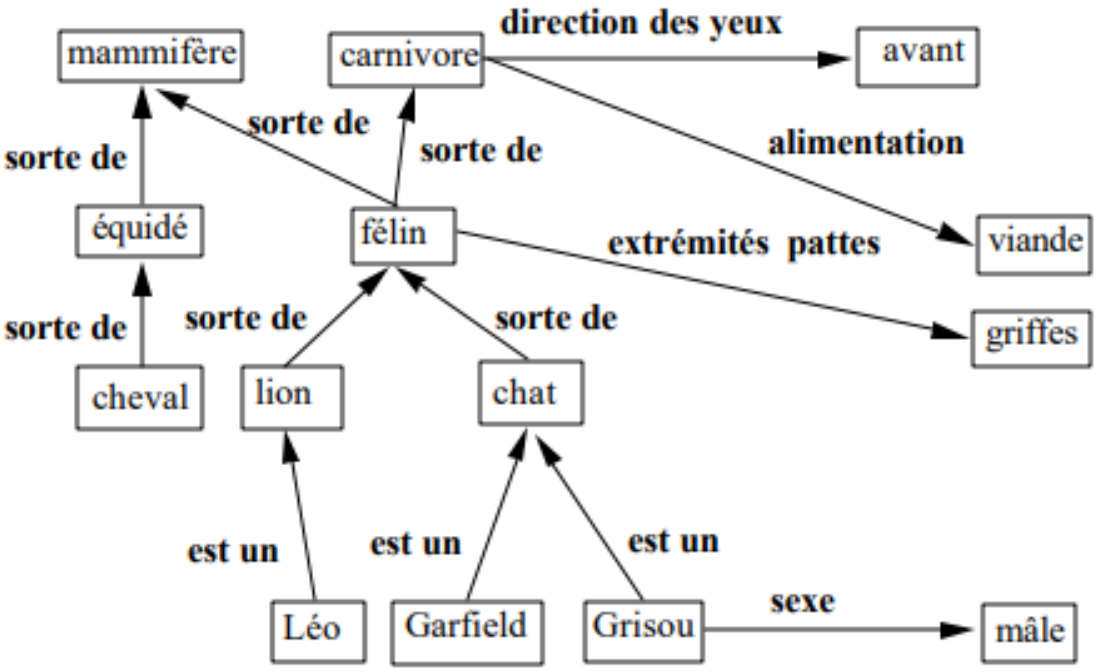


Figure 2.1 : Réseau sémantique

Ces réseaux permettent de visualiser le savoir comme un ensemble de relations hiérarchiques et associatives. Ils sont intuitifs, faciles à représenter et très utilisés dans les systèmes de type “ontologies”.

b. Les graphes conceptuels

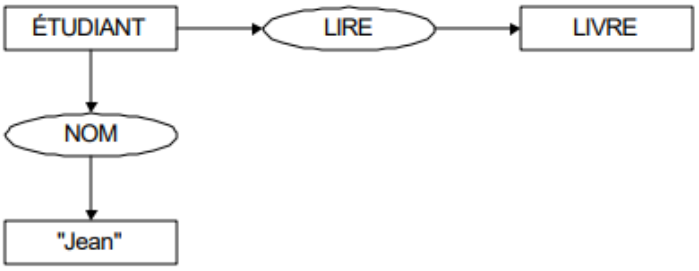
C’est une version un peu plus formelle et rigoureuse des réseaux sémantiques. Les graphes conceptuels sont des graphes finis, connexes et bipartites. Les deux sortes de nœuds d’un graphe bipartite sont les concepts et les relations conceptuelles.

On distingue clairement :

- Les concepts sont dans des rectangles
- Les relations sont dans des ovales
- Les flèches relient les deux types de nœuds

Exemple :

- Jean lit un livre



Ces graphes sont très utilisés en traitement automatique du langage naturel, car ils permettent de visualiser des phrases sous forme de structures compréhensibles qui peuvent être facilement exploitées pour des tâches telles que l'analyse sémantique ou la traduction automatique. De plus, les graphes conceptuels sont capables de représenter des situations complexes de manière structurée, en organisant les concepts et leurs relations dans un réseau cohérent.

Cependant, les graphes conceptuels sont moins répandus en pratique que d'autres formalismes, comme les logiques formelles ou les systèmes basés sur des règles, ce qui limite leur adoption dans certaines applications industrielles. Par ailleurs, le raisonnement dans les graphes conceptuels reste limité sans l'intégration d'un moteur d'inférence supplémentaire. En effet, pour effectuer des raisonnements complexes ou déduire de nouvelles connaissances à partir des relations existantes, il est souvent nécessaire de compléter ces graphes avec des outils d'inférence ou des algorithmes de raisonnement plus sophistiqués.

2.5 Programmation logique

La programmation logique est un type de paradigme de programmation qui est en grande partie basée sur la logique formelle. Tout programme écrit dans un langage de programmation logique est un ensemble de phrases sous forme logique, exprimant des faits et des règles concernant un domaine problématique. Ces faits et ces règles sont exploités par un moteur d'inférence, en réaction à une question ou requête.

Dans la plupart des langages de programmation classiques (comme C, Java ou Python), on donne à la machine une suite d'instructions à exécuter. C'est ce qu'on appelle la programmation impérative : on dit *comment* faire les choses, étape par étape.

La programmation logique fonctionne différemment. Au lieu de dire *comment* faire, on décrit *ce qu'on sait* (les faits) et *ce qu'on veut* (les objectifs). C'est ensuite un moteur d'inférence qui se charge de déduire les réponses.

La programmation logique est considérée comme une programmation déclarative plutôt qu'impérative, car elle s'attache davantage au *quoi* qu'au *comment*, le moteur assumant une large part des enchaînements. Elle est particulièrement adaptée aux besoins de l'intelligence artificielle, dont elle est un des principaux outils.

C'est un peu comme dans une enquête policière : tu poses les faits sur la table, et tu laisses le détective (ici, la machine) tirer les conclusions logiques.

Robert Kowalski, un des pionniers de la programmation logique, a résumé ce paradigme dans une formule célèbre :

algorithmes = logique + contrôle

qui signifie qu'un algorithme est créé en spécifiant des connaissances par des axiomes (logique) et que le problème est résolu par un mécanisme d'inférence qui agit sur lui (contrôle) [6]. Parmi les langages de programmation logique on peut mettre en évidence Prolog, Lisp ou Erlang.

a. Le langage Prolog

Le langage le plus emblématique de la programmation logique, c'est Prolog, acronyme de "PROgrammation en LOGique".

Dans Prolog :

- Les faits sont écrits simplement :
  `chat(garfield).` → Garfield est un chat.
  `mange(garfield, lasagne).` → Garfield mange des lasagnes.
- Les règles permettent de faire des inférences :
  `animal(X) :- chat(X).` → Si X est un chat, alors X est un animal.
- Les requêtes permettent d'interroger le système :
  `?- animal(garfield).` → Est-ce que Garfield est un animal ?

Si le système peut prouver que la requête est vraie à partir des faits et des règles, il répondra "Yes". Sinon, "No".

Prolog et la programmation logique sont particulièrement adaptés pour :

- Les systèmes experts (diagnostic, assistance à la décision),
- Les bases de connaissances**,**
- La représentation sémantique du langage naturel**,**
- La planification et la résolution de problèmes**.**

L'avantage de ce paradigme, c'est qu'il est déclaratif : on ne programme pas l'algorithme, on déclare ce qu'on sait, et le système fait le reste. Cela rend les programmes souvent plus courts, plus élégants, et surtout plus proches du raisonnement humain.

2.6 Conclusion

La représentation des connaissances est souvent moins spectaculaire que la vision artificielle ou les assistants vocaux. Et pourtant, sans elle, aucune IA ne pourrait vraiment *raisonner*, *comprendre* ou *expliquer* ses choix.

Dans ce chapitre, nous avons exploré les différences entre donnée, information et connaissance, ainsi que la nécessité de représenter cette connaissance en IA. Nous avons étudié les principaux formalismes, tels que la logique des propositions, la logique des prédicats, les réseaux sémantiques, les graphes conceptuels et Prolog.

Cela, nous a permis de voir qu'il ne suffit pas d'accumuler des données : il faut pouvoir leur donner du sens, les organiser, et les manipuler de manière intelligente. Dans le chapitre suivant, nous allons nous intéresser à l'un des domaines les plus applicatifs de ces techniques : les systèmes experts. Ces systèmes utilisent des bases de connaissances formelles et des moteurs d'inférence pour simuler les capacités de raisonnement d'un expert humain dans un domaine spécifique.

Chapitre 3 :
Les Systèmes Experts

3.1 Introduction

Imagine un programme capable de diagnostiquer une maladie, recommander une solution à un problème technique, ou même t'aider à faire une recette avec ce que tu as dans ton frigo. Ce programme ne sait pas tout… mais dans un domaine bien précis, il peut raisonner comme un expert humain. C'est exactement le rôle d'un système expert.

Les systèmes experts ont été parmi les premières grandes réussites de l'intelligence artificielle symbolique. Leur rôle est de capturer l'expertise humaine et la rendre accessible sous forme logicielle. Ils reposent sur des connaissances explicites (règles, faits, raisonnements) et peuvent expliquer leurs conclusions. Cependant, ils restent enfermés dans un domaine étroit et ont du mal à sortir du cadre qui leur a été défini.

Dans ce chapitre, nous allons explorer tout ce qui concerne les systèmes experts en partant de la définition, historique, ses composants, et son fonctionnement.

3.2 Définition d'un système expert

Un système expert (SE) est un programme informatique conçu pour imiter le raisonnement et le comportement d'un expert humain dans un domaine spécifique. Son rôle est de résoudre des problèmes complexes en se basant sur un ensemble de connaissances et de règles logiques [6].

Un SE est un programme qui permet l'exploitation des connaissances dans un domaine précis et rigoureusement limité. Tel un expert humain, un SE n'a aucune prétention en dehors de sa spécialité.

Un SE est un système informatique où les données (la base de connaissance) sont bien séparées du programme qui les manipule (le moteur d'inférences). La base de données des connaissances ainsi que le système d'inférence sont fondés sur la logique et l'inférence logique.

Les systèmes experts ont été utilisés dans une grande variété de domaines. Voici quelques exemples :

- **Diagnostic médical** : Des systèmes comme *MYCIN*, créés dans les années 1970, ont été utilisés pour diagnostiquer des infections sanguines et prescrire des traitements.
- **Diagnostic technique** : Des systèmes experts sont utilisés pour réparer des machines ou des équipements en analysant des symptômes de panne.
- **Aide à la décision** : Dans des domaines comme la finance, l'agriculture ou l'industrie, des systèmes experts aident les décideurs à analyser des données et à choisir les meilleures options.

3.3 Historique des systèmes experts

Les systèmes experts ont vu le jour dans les années 1960 et 1970, une époque où l'intelligence artificielle commençait à se définir comme un domaine à part entière. L'histoire des systèmes experts est marquée par des avancées majeures et des réalisations pratiques, qui ont transformé le domaine de l'IA [4].

a. Dendral : le premier système expert (1965)

Le premier système expert, nommé Dendral, a été développé en 1965 par Edward Feigenbaum, Bruce Buchanan, le médecin Joshua Lederberg et le chimiste Carl Djerassi. Ce système était conçu pour aider les chimistes à analyser les spectres de masse et la résonance magnétique nucléaire afin d'identifier les structures chimiques des molécules.

Dendral était un système expert dans le domaine de la chimie, capable de faire des déductions sur la base de règles chimiques. C'est ce système qui a réellement lancé l'idée de l'IA dans un cadre spécialisé. Plutôt que de faire des calculs abstraits, il cherchait à imiter le raisonnement d'un expert chimiste en utilisant des règles précises.

b. MYCIN : l'évolution du système expert (1972)

Le plus célèbre des premiers systèmes experts est MYCIN, créé en 1972. MYCIN était un système destiné à diagnostiquer des infections sanguines et à prescrire des médicaments. Ce système expert était capable de poser des questions à l'utilisateur pour récolter des informations sur les symptômes, puis d'utiliser des règles de déduction pour proposer un diagnostic.

MYCIN ne se contentait pas d'appliquer des règles simples. Il attribuait à chaque règle un poids de vraisemblance, ce qui permettait de tenir compte du degré de confiance dans chaque déduction. C'était un pas important vers des systèmes plus nuancés et adaptatifs.

3.4 Structure d'un système expert

Les systèmes experts sont généralement constitués de plusieurs composants essentiels qui travaillent en interaction pour résoudre des problèmes complexes de manière autonome. Ces composants permettent de manipuler les connaissances, de raisonner à partir de ces connaissances et d'interagir avec l'utilisateur. Voici les trois principaux éléments qui forment la structure d'un système expert.

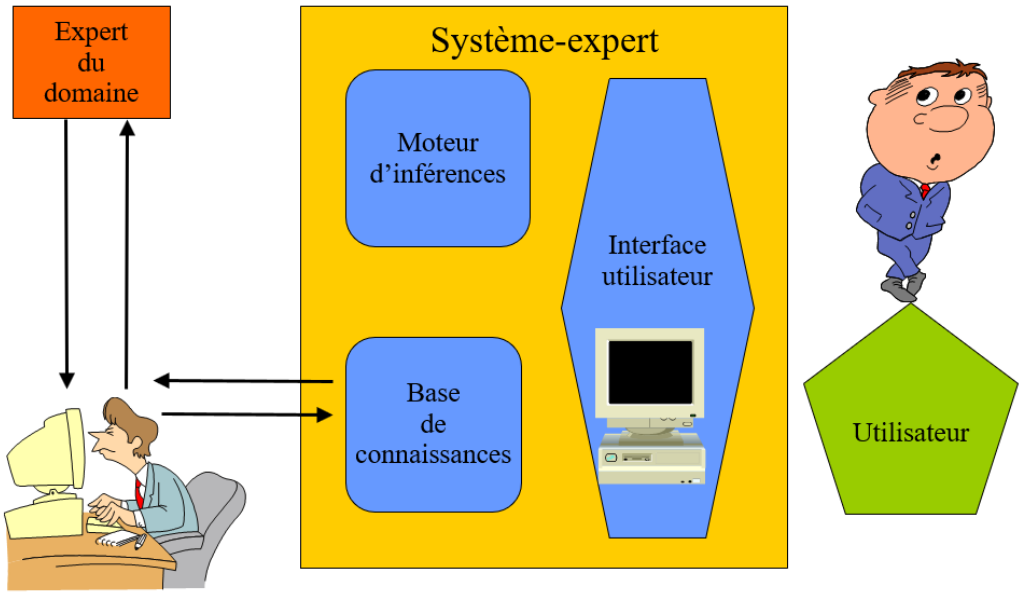


Figure 3.1 : Structure d'un système expert

a. Base de connaissances (Knowledge Base)

La base de connaissances est l'un des composants les plus importants d'un système expert. Elle contient toutes les informations nécessaires pour que le système puisse prendre des décisions éclairées. On peut la considérer comme la mémoire du système, où sont stockées les connaissances spécialisées dans un domaine particulier [6]:

- **Les standards d'engagement** (connaissances de l'expert) : l'informations de base et de configuration du système, mesures, lois, paramètres, données contractuelles.
- **Les règles d'inférence** (savoir-faire) : ensemble des règles logiques de déduction utilisées par le moteur d'inférence.
- **La base de faits** (expérience) Historisation et statistique des faits effectifs, des décisions et des buts. Le raisonnement va se baser sur ces faits pour déduire des conclusions.

1. **La base des faits (facts)** :
   Ce sont les **données de base**, les éléments d'information spécifiques au domaine, comme :
   - « Le patient a une température de 38°C. »

La base des faits contient l'ensemble des données de départ sur lequel le système va commencer à travailler. Cette base s'enrichit au fur et à mesure des déductions faites par le système. Cet espace de travail est un peu la mémoire à court terme, le système y stocke aussi les règles en attente, les sous-problèmes …

2. **Les règles d'inférence (inference rules)** :

Ce sont des règles logiques qui décrivent des relations et des actions, souvent sous la forme de **"si... alors..."**. Par exemple :

- « Si un patient a de la fièvre et des maux de tête, alors il peut avoir la grippe. »

Ces règles permettent au système d'appliquer un raisonnement déductif pour arriver à des conclusions à partir des faits connus.

La base de connaissances est donc l'endroit où toutes les connaissances d'un expert humain sont stockées, transformées et organisées pour être exploitées par le système.

b. Moteur d'inférence (Inference Engine)

Le moteur d'inférence est le cerveau du système expert. C'est lui qui prend les faits disponibles et les applique aux règles de la base de connaissances pour déduire de nouvelles informations ou prendre des décisions.

c. L'interface utilisateur (User Interface)

L'interface utilisateur est le pont entre le système et l'utilisateur (qu'il soit un expert, un technicien ou un simple utilisateur). Elle permet à l'utilisateur d'entrer des faits, de poser des questions et d'obtenir des recommandations ou des solutions basées sur les conclusions du système.

Cette interface peut être :

- **Graphique** : Une interface avec des fenêtres, des boutons et des menus.
- **Textuelle** : Un simple terminal ou une interface en ligne de commande.
- **Vocale** : Un système de reconnaissance vocale pour les applications mobiles ou les robots.

L'objectif de l'interface est de rendre le système aussi intuitif et facile d'accès que possible, tout en permettant une communication claire entre l'utilisateur et le moteur d'inférence.

3.5 Fonctionnement d'un système expert

Le moteur d'inférence fonctionne généralement soit en chainage avant ou chainage arrière comme expliqué précédemment, ceci dit, la capacité d'explication et la transparence est plus développée en chaînage arrière qu'en chaînage avant parce que toutes les questions sont posées pour répondre à des sous buts.

En chaînage avant les questions sont posées selon l'ordre de présentation des règles dans la base de connaissances et peuvent parfois refléter une incohérence. Par contre, le système peut poser des questions inutiles en chaînage avant, ce qui n'est pas le cas en chaînage arrière.

Le chainage avant est très utile quand on dispose de nombreux faits initiaux et que le but n'est pas bien défini à l'avance, ou bien lorsqu'on veut explorer toutes les conséquences possibles d'un état donné. Il fonctionne selon deux grandes méthodes de raisonnement :

3.5.1 Le chaînage avant (Forward Chaining) :

Dans le chaînage avant (mécanisme guidé par les données), le moteur d'inférence commence par les faits disponibles et applique les règles pour en déduire de nouvelles informations. Il fonctionne de manière progressive, ajoutant des faits à la base de faits à chaque étape [4].

- Détecter les règles dont les prémisses sont vérifiées
- Sélectionner la règle à appliquer
- Appliquer la règle
- Recommencer jusqu'à ce qu'il n'y ait plus de règle applicable
- ***Se base sur le Modus Ponens :*** *si f1 est vrai et f1 ⇒ f2 alors f2 est vrai.*

Exemple :

Si le système dispose des faits suivants :

- « Le patient a une température de 38°C. »
- « Le patient a mal à la tête. »

Et si la règle suivante existe dans la base de connaissances :

- « Si un patient a de la fièvre et des maux de tête, alors il peut avoir la grippe. »

Le moteur d'inférence appliquera cette règle et déduira que le patient pourrait avoir la grippe.

3.5.2 Le chaînage arrière (Backward Chaining) :

Dans le chaînage arrière, (mécanisme guidé par le but), le moteur d'inférence commence par un objectif ou une question et tente de trouver les faits et les règles nécessaires pour répondre à cette question. Il fonctionne de manière rétroactive, en cherchant les prémisses qui mènent à une conclusion.

- Détection des règles qui concluent à ce but
- Résolution de conflits

- Application de la règles (les éléments des prémisses deviennent de nouveau sous- buts à atteindre).
- Arrêt : pile vide ou aucune règle applicable
- ***Se base sur le Modus Tollens :*** *Si f2 est non vrai et si f1 ⇒ f2 alors f1 est non vrai [4].*

Exemple :

Si l'objectif est de savoir si un patient a la grippe, le moteur d'inférence cherchera des règles qui mènent à ce résultat. Il pourrait trouver une règle comme :

- « Si un patient a de la fièvre et des maux de tête, alors il peut avoir la grippe. »

Il cherchera ensuite à vérifier si les faits nécessaires (fièvre et maux de tête) sont présents dans la base de faits. Si c'est le cas, il conclura que le patient a la grippe.

Par contre, il peut générer beaucoup de faits inutiles si la base de règles est trop large ce qui entraîne un raisonnement coûteux en ressources. Il est moins pertinent quand on cherche à vérifier une hypothèse précise (dans ce cas, on privilégiera le chaînage arrière).

Il existe d'autre types de chainage comme par exemple [6] :

3.5.3 Chaînage mixte (ou bidirectionnel)

Le chaînage mixte combine les avantages des deux approches :

- Le **chaînage avant** pour explorer les données connues,
- Le **chaînage arrière** pour diriger le raisonnement vers un but précis.

Le moteur commence avec un but à atteindre (chaînage arrière), s'il manque des faits, il explore les règles activables (chaînage avant), puis il revient au but pour vérifier s'il peut maintenant être atteint.

3.5.4 Chaînage avec incertitude (ou raisonnement probabiliste)

Dans certains systèmes, les faits et les règles sont associés à des coefficients de certitude ou des probabilités. Le chaînage se fait alors en tenant compte du degré de confiance.

- **Règle 1** : SI fièvre ET toux → ALORS grippe [confiance = 0,8]
- **Règle 2** : SI fatigue ET courbatures → ALORS grippe [confiance = 0,6]

Le moteur d'inférence accumule les scores pour juger de la vraisemblance du diagnostic final.

3.5.5 Chaînage flou (logique floue)

Le chaînage flou s'appuie sur des règles imprécises, comme :

« Si la température est élevée, alors il est probable qu'il y ait infection »

Plutôt que de traiter les faits comme vrais/faux, il les traite comme des degrés de vérité (entre 0 et 1).

3.5.6 Chaînage basé sur les cas (Case-Based Reasoning)

Ici, on ne raisonne pas avec des règles formelles mais en comparant la situation actuelle avec des cas similaires passés. C'est un raisonnement par analogie.

3.6 Mise en place d'un système expert

Le fonctionnement d'un système expert repose sur l'interaction entre la base de connaissances**,** le moteur d'inférence et l**'**interface utilisateur. Le processus de raisonnement s'effectue généralement en plusieurs étapes qui permettent au système d'analyser les faits, de tirer des conclusions, et de fournir des recommandations ou des solutions.

Le raisonnement dans un système expert peut être divisé en deux grandes étapes : la collecte des faits et l'application des règles d'inférence**.**

1. **Etude de faisabilité**

   Evaluer le domaine et les risques d'échecs de la mise en place et de succès de l'outil auprès des professionnels et futurs utilisateurs.

2. **Extraction des données**

   Dialogue entre le cogniticien et l'expert afin d'extraire de ce dernier toutes ses connaissances et son savoir-faire. La collecte des faits est l'étape où l'utilisateur entre des informations de base dans le système.

3. **Formalisation**

   Formaliser les connaissances qu'il a récolté.

4. **Design et développement**

Faire appel aux développeurs informatique afin de commencer à définir le cahier des charges précis, la base de connaissance et les règles d'inférence.

5. **Tests et optimisations**

Une série de tests auprès des experts mais aussi auprès d'utilisateurs lambdas qui sont sensés à partir de cet outil fournir les résultats d'un expert débutant.

L'acquisition des données est une tâche longue et fastidieuse, il faut décrire le comportement d'un expert face à un problème particulier, et sa manière de le résoudre. Car ce que l'on souhaite obtenir n'est ni plus ni moins que l'expérience, la connaissance pratique de l'expert, et non la théorie que l'on peut trouver dans les livres.

3.7 Avantages et limites des systèmes experts

Avantages :

- Reproduisent l'expertise humaine dans des domaines complexes.
- Disponibles 24h/24, contrairement à un humain.
- Capables d'expliquer leur raisonnement (explicabilité).
- Réduisent les erreurs dans les processus critiques.
- Faciles à mettre à jour (il suffit de modifier ou ajouter des règles).

Limites :

- Très dépendants de la qualité et de l'exhaustivité de la base de connaissances.
- Ne peuvent pas apprendre par eux-mêmes (contrairement aux systèmes d'apprentissage automatique).
- Coût élevé de développement (extraction des connaissances longue et complexe).
- Peu efficaces en dehors de leur domaine de spécialisation.
- Sensibles aux incohérences dans les règles.

3.8 Autres types de systèmes experts

- **Systèmes experts incertains** : utilisent des degrés de confiance (ex. : logique floue, probabilités).
- **Systèmes experts répartis** : coopèrent à distance, chacun dans un sous-domaine.
- **Systèmes experts hybrides** : intègrent des techniques d'IA non symboliques (réseaux de neurones, algorithmes génétiques, etc.).
- **Systèmes à base de cas** : ne déduisent pas des règles, mais comparent à des Étapes du raisonnement :

3.9 Exemple illustré

Pour mieux comprendre comment un système expert fonctionne en pratique, prenons un exemple concret : un **système expert de pâtisserie**. Imaginons un programme conçu pour

aider un utilisateur à déterminer quel dessert il peut préparer en fonction des ingrédients disponibles dans sa cuisine.

Dans ce scénario, le système expert va utiliser des faits (les ingrédients disponibles) et des règles de recette pour guider l'utilisateur vers la création d'un dessert spécifique. Le système doit poser des questions à l'utilisateur pour savoir quels ingrédients il a, puis appliquer des règles pour déterminer quelles recettes peuvent être réalisées avec ces ingrédients.

On considère le système expert « patisserie » suivant :

Base de règles	Base de faits
R1 : Si farine et beurre et œufs et sel alors pâte **R2** : Si pommes et sucre alors pommes sucrées **R3** : Si pommes sucrées et pâte alors tarte aux pommes **R4** : Si abricots et pâte alors tarte aux abricots **R5** : Si poires et pâte alors tarte aux poires **R6** : Si cerises et pâte alors tarte aux cerises	Pommes Poires Abricots Farine Beurre œufs Sucre Sel

- **Chainage avant :** Avec les ingrédients (les faits) qu'est-ce que je peux faire ?
- **Chainage arrière :** Est-ce que je peux faire une tarte aux abricots ?

b. Chaînage avant : Raisonnement guidé par les données

Avec le **chaînage avant**, le système commence avec les faits disponibles (les ingrédients) et applique des règles pour déterminer les desserts possibles. Voici comment cela fonctionne :

- **R1 : est-elle applicable ?**

Oui (tous les faits sont dans la base des faits)

Le fait **pâte** est donc rajouté dans la base des faits et la R1 est désactivée.

Base de règles	Base de faits
~~**R1** : Si farine et beurre et œufs et sel alors pâte~~ **R2** : Si pommes et sucre alors pommes sucrées **R3** : Si pommes sucrées et pâte alors tarte aux pommes **R4** : Si abricots et pâte alors tarte aux abricots **R5** : Si poires et pâte alors tarte aux poires **R6** : Si cerises et pâte alors tarte aux cerises	Pommes Poires Abricots Farine œufs Beurre Sucre Sel Pâte

- **R2 : est-elle applicable ?**
- Oui : Le fait pommes sucrées est ajouté dans la base des faits et la R2 est désactivée.

Base de règles	Base de faits
~~**R1** : Si farine et beurre et œufs et sel alors pâte~~ ~~**R2** : Si pommes et sucre alors pommes sucrées~~ **R3** : Si pommes sucrées et pâte alors tarte aux pommes **R4** : Si abricots et pâte alors tarte aux abricots **R5** : Si poires et pâte alors tarte aux poires **R6** : Si cerises et pâte alors tarte aux cerises	Pommes Poires Abricots Farine œufs Beurre Sucre Sel Pâte Pommes sucrées

- De même, les règles R3, R4 et R5 sont sélectionnées et exécutées.

Base de règles	Base de faits
~~**R1** : Si farine et beurre et œufs et sel alors pâte~~ ~~**R2** : Si pommes et sucre alors pommes sucrées~~ ~~**R3** : Si pommes sucrées et pâte alors tarte aux pommes~~ ~~**R4** : Si abricots et pâte alors tarte aux abricots~~ ~~**R5** : Si poires et pâte alors tarte aux poires~~ **R6** : Si cerises et pâte alors tarte aux cerises	Pommes Poires Abricots Farine œufs Beurre Sucre Sel Pâte Pommes sucrées Tarte aux pommes Tarte aux abricots Tarte aux poires

- **R6 : elle ne peut pas être exécutée.**

Le fait "cerises" n'existe pas dans la base des faits.

Résultat : Le système propose à l'utilisateur : une tarte aux pommes ou une tarte aux abricots ou une tarte aux poires.

Arrêt : Le système continue d'appliquer des règles jusqu'à ce qu'il n'y ait plus de règles applicables ou qu'un objectif soit atteint. Et ici dans l'exemple, le mécanisme ne peut plus effectuer de déductions et le moteur s'arrête.

Chainage arrière : raisonnement guidé par le but

But = « faire une tarte aux abricots »
Le but « Tarte aux abricots » se trouve dans la règle R4.

Considérons R4 : les prémisses sont-elles vérifiées dans la base des faits ?

La première prémisse « abricot » est vérifiée.
La deuxième prémisse « pâte » doit être vérifiée ;

On vérifie s'il existe une règle qui a pour but « pâte » :
Oui, la règle R1.

On considère maintenant la règle R1 : les prémisses sont vérifiées

Donc, on peut conclure que le but tarte aux abricots est vérifié.

Résultat : Oui, on peut faire une tarte aux abricots.

Cet exemple de système expert de pâtisserie montre comment les deux types de raisonnement (chaînage avant et chaînage arrière) peuvent être utilisés pour résoudre des problèmes de manière autonome. Le chaînage avant est plus fluide et s'adapte bien aux situations où de nombreuses données sont disponibles, tandis que le chaînage arrière est plus adapté lorsque l'on cherche à atteindre un objectif précis.

3.10 Conclusion

Les systèmes experts représentent l'une des premières et des plus réussies applications de l'intelligence artificielle symbolique. Leur capacité à imiter le raisonnement d'un expert humain dans un domaine spécifique permet de résoudre des problèmes complexes de manière autonome, tout en fournissant des réponses justifiables et compréhensibles.

Les applications des systèmes experts sont multiples, allant du diagnostic médical aux systèmes de support à la décision dans de nombreux secteurs comme l'ingénierie, la finance, ou même la cuisine. Cependant, la création de ces systèmes reste un défi technique, en particulier en ce qui concerne l'extraction des connaissances d'un expert humain et leur formalisation en un ensemble de règles et de faits compréhensibles par un programme informatique.

Chapitre 4 :
Méthodes de résolution de problèmes

4.1 Introduction

L'intelligence artificielle n'a de sens que si elle est capable de résoudre des problèmes**.** Qu'il s'agisse de jouer aux échecs, de planifier un trajet, de diagnostiquer une panne ou de recommander un film, une IA doit pouvoir analyser une situation, évaluer ses options**,** et prendre des décisions intelligentes**.**

Mais comment fait-elle ? Par quels mécanismes une machine peut-elle passer d'un état de départ à une solution efficace ? Quels chemins suit-elle pour atteindre un objectif, surtout quand elle ne connaît pas l'environnement à l'avance ? C'est là qu'interviennent les méthodes de résolution de problèmes, un pilier fondamental de l'IA symbolique.

Dans ce chapitre, nous allons définir les notions de problème, espace de recherche en IA et ensuite nous expliquerons les principales stratégies de recherche pour résoudre les problèmes. En modélisant les situations sous forme de graphes d'états et en explorant ces graphes intelligemment, une IA peut prendre des décisions de manière totalement autonome.

4.2 Définition d'un problème

Un problème est une collection d'informations que l'agent utilise pour décider quelles actions accomplir. Il existe plusieurs types de problèmes, on peut citer les problèmes de jeux qui sont concis et bien définis tels que le jeu du taquin ou des N reines, leur modélisation est facile. Et il y a également les problèmes issus du monde réel qui sont généralement complexes et très ouverts et difficile à résoudre car il y a beaucoup de paramètres à prendre en considération, comme par exemple le problème du voyageur de commerce.

Dans le contexte de l'IA, résoudre un problème revient à guider un agent (ou un programme) pour qu'il passe d'un état initial à un état final souhaité, en choisissant judicieusement les actions qu'il doit exécuter [7].

Définir un problème, c'est [4] :

- **Identifier un état initial** : la situation de départ dans laquelle se trouve l'agent.
- **Lister les actions possibles** : c'est-à-dire les opérations que l'agent peut effectuer à chaque étape.
- **Définir les états atteignables** : chaque action transforme un état en un autre.
- **Spécifier un ou plusieurs états objectifs** : ce sont les situations que l'agent cherche à atteindre (états but).

Exemple :

Un GPS cherche un itinéraire entre un point A et un point B.

- État initial = position actuelle
- Actions = tourner, avancer, changer de rue
- État but = destination finale
- Critère = distance minimale ou temps le plus court

4.3 Principe de résolution de problèmes

La résolution d'un problème est une tentative de programmation des ordinateurs pour faire ce que l'humain fait naturellement : percevoir, raisonner et agir de manière autonome autrement pour faire des choses dont on dit qu'elle nécessite de l'intelligence quand elles sont faites par des humains. C'est aussi l'étude des calculs qui rendent possible la perception, le raisonnement et l'action.

4.4 Espace de recherche

L'espace de recherche est l'ensemble des états possibles atteignables depuis l'état initial par n'importe quelle séquence d'actions, c'est-à-dire les états que peut prendre le système pendant la résolution du problème.

Un espace de recherche peut être représenté par un graphe orienté. Les sommets sont les états, les arcs sont les actions. L'objectif est de trouver un chemin dans ce graphe qui va de l'état initial à un état but.

➢ Types d'espaces de recherche

Le tableau 4.1 représente les différents types d'espaces de recherche avec leur description ainsi que des exemples.

Type	Description	Exemples
Fini et déterministe	Nombre limité d'états, chaque action a un seul résultat	Jeu d'échecs, problème du taquin
Incomplet ou stochastique	États ou résultats incertains (probabilités, inconnues)	Poker, météo, comportements humains
Avec cycles	Possibilité de repasser par un même état	Labyrinthe
Sans cycles (arborescent)	Exploration unique, sans retour	Arbre de decision

Tableau 4.1 : Exemples types d'espaces de recherche

Exemples :

- **Etats**: Position des huit tuiles dans les cases.
- **Etat initial**: Les huit tuiles dans n'importe quelle case.
- **Actions**: Déplacement du trou (droite, gauche, haut, bas)
- **Test de but**: Un état qui correspond à l'état final.

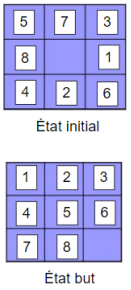


Figure 4.1 : Jeu du taquin

- **Etats:** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial**: Aucune reine sur l'échiquier
- **Actions**: Ajouter une reine sur une case vide
- **Test de but**: Les 8 reines sont placées sur l'échiquier sans attaque.

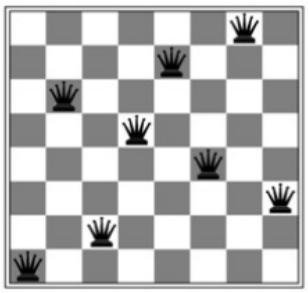


Figure 4.2 : Jeu des N reines

4.5 Résolution de problèmes

On part du principe que tous les problèmes étudiés sont composés d'un ensemble de situations que l'on peut décrire à l'aide d'un ensemble de variables appelés variables d'états.

L'état d'un problème est alors l'ensemble des valeurs prises par les variables à un instant donné. L'espace d'état d'un problème est l'ensemble de tous ses états possibles.

Chaque état possède un ensemble d'états qui lui sont accessibles dans le problème. Les opérateurs permettent de définir cet ensemble, et c'est la stratégie de résolution qui donnera la façon de choisir parmi ces états accessibles celle menant à une solution.

Un problème est défini pour un objectif particulier. On doit donc également définir une fonction de test de but atteint qui détermine si un état du problème correspond à un état but du problème [7].

Une solution est une séquence d'actions permettant de passer de l'état initial vers un état but. On peut considérer que la résolution d'un problème est la découverte d'un état du problème ayant des caractéristiques données (la solution).

Le processus de résolution de problème consiste donc à construire un arbre de recherche. Chaque nœud de l'arbre correspond soit à l'état initial du problème, soit à un développement du sommet parent par un des opérateurs du problème.

On représente un problème par un espace d'états (arbre/graphe). Chaque état est une configuration possible du problème. Résoudre le problème consiste à trouve un chemin dans le graphe.

- **Algorithme de recherche : idée générale**
  1. Démarrer la recherche avec la liste contenant l'état initial du problème.
  2. Si la liste n'est pas vide alors :
     a) Choisir à l'aide d'une stratégie un état e à traiter.
     b) Si e est un état final alors retourner recherche positive
     c) Sinon, rajouter tous les successeurs de e à la liste d'états à traiter et recommencer au point 2.
  3. Sinon retourner recherche négative.

4.6 Stratégies de recherche

4.6.1 Stratégies de recherche non informées (ou aveugles)

Les méthodes de recherche non informées sont appelées ainsi parce qu'elles ne disposent d'aucune information spécifique sur la proximité du but à atteindre. Elles se contentent d'explorer systématiquement l'espace de recherche en suivant une stratégie prédéfinie

Ces méthodes sont utiles lorsqu'on connaît les règles du problème (les actions, les états), mais on n'a aucun indice sur l'endroit où se trouve la solution.

- ***Recherche en largeur d'abord (Breadth-First Search – BFS)***

Cette stratégie donne une plus grande priorité aux sommets les moins profonds du graphe de recherche en explorant les sommets de même profondeur [4]. On explore d'abord tous les voisins immédiats de l'état initial, puis les voisins de ces voisins, et ainsi de suite (exploration niveau par niveau) comme le montre la figure 4.3.

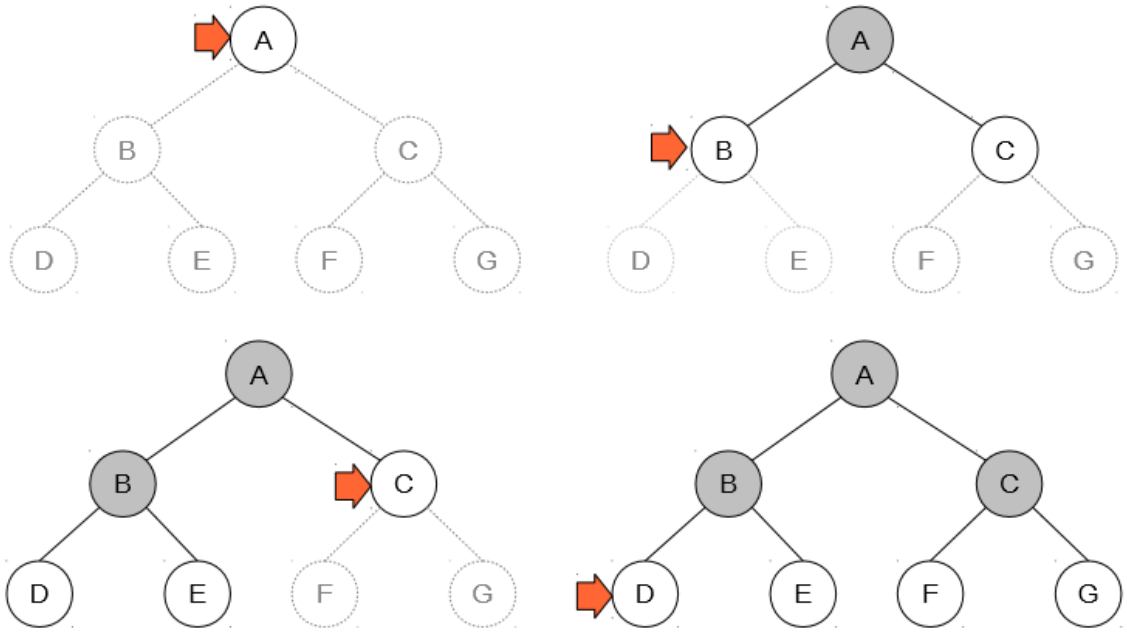


Figure 4.3 : Exemple BFS

La figure 4.4 ci-dessous l'algorithme de recherche BFS :

Algorithm 2 Recherche en largeur

FONCTION ExplorationLarg(E_0) : LISTE
VAR F : FILE
  $F \leftarrow fileVide$
  $L \leftarrow listeVide$
  $Ajouter(F, E_0)$
  TANTQUE NON vide(F) **FAIRE**
    inserer(L,premier(F))
    SI NON $premier(F) \in F$ **ALORS**
      POUR TOUT $(A_j, E_j) \in T(premier(F))$ **FAIRE**
        $ajouter(F, E_j)$
      FIN POUR
      supprimer(F)
    SINON
      $F \leftarrow fileVide$
    FIN SI
  FIN TANTQUE
RETOURNER L

Figure 4.4 : Algorithme BFS

- ***Recherche en profondeur d'abord (Depth-First Search – DFS)***

Dans cette stratégie, on explore chaque branche au maximum avant de revenir en arrière [4]. Chaque sommet choisi pour être exploré voit tous ses successeurs générés avant qu'un autre sommet ne soit exploré comme le montre la figure ci-dessous.

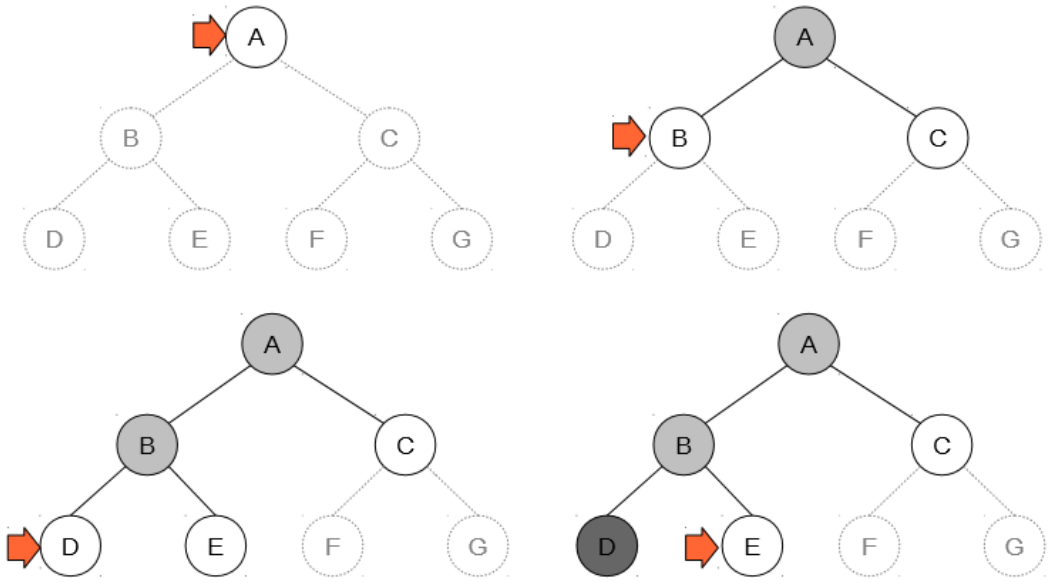


Figure 4.5 : Exemple DFS

Cette stratégie ne garantit ni la solution la plus courte, ni même une solution si l'arbre est infini ou contient des cycles, en effet l'algorithme risque de développer une branche infinie stérile sans que le mécanisme de retour en arrière puisse se déclencher. On ajoute dans ce cas une limite de profondeur

- ***Recherche avec profondeur limitée***

C'est une variante de DFS où on impose une limite maximale de profondeur pour éviter de se perdre dans des boucles infinies. Ceci dit, dans cette stratégie, il y a un risque de ne jamais trouver la solution si elle est plus profonde que la limite fixée.

- ***Recherche itérative en profondeur (Iterative Deepening DFS – IDDFS)***

On effectue plusieurs recherches en profondeur successives, en augmentant progressivement la limite de profondeur. Cette stratégie permet de trouver la solution optimale.

- ***Recherche à coût uniforme (Uniform Cost Search)***

On choisit toujours le chemin le moins coûteux à chaque étape, en tenant compte d'un coût cumulé (distance, temps, énergie, etc.) comme le montre la figure suivante.

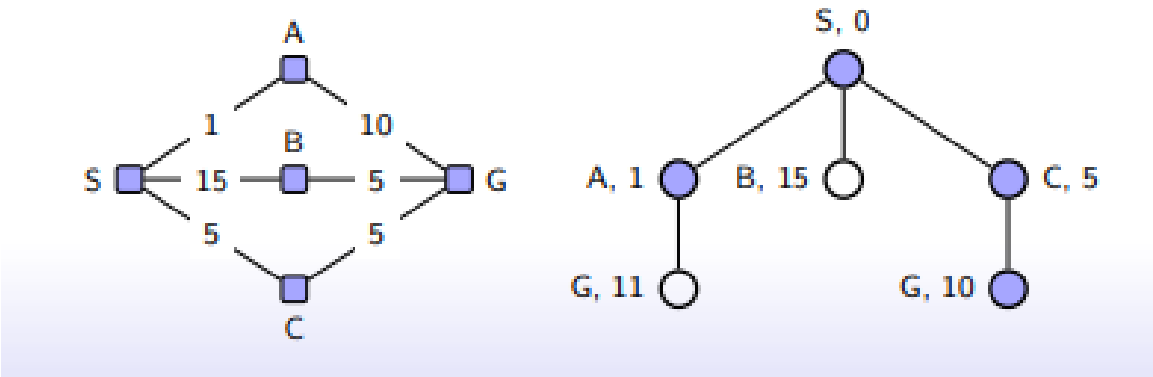


Figure 4.6 : Exemple Uniform cost search

4.6.2 Stratégies de recherche informées (ou heuristiques)

Les méthodes de recherche non informées que nous avons vues précédemment fonctionnent bien pour des petits espaces de recherche mais deviennent vite inefficaces dès que le problème grandit car elles explorent sans discernement, en suivant une stratégie rigide sans avoir une idée si elle se rapproche de la solution ou pas [7].

Les stratégies informées utilisent un ensemble de règles qui permettent d'évaluer la probabilité qu'un chemin allant du nœud courant au nœud solution soit meilleur que les autres. Cette information permet l'estimation des bénéfices des divers chemins avant de les parcourir améliore le processus de recherche.

Les algorithmes de recherche heuristique utilisent des fonctions appelées fonctions heuristiques. $h : E \rightarrow \boldsymbol{R}$

Heuristique

Une heuristique est une mesure associée à un état donné qu'on notera h(Ei). C'est un critère ou une méthode permettant de déterminer parmi plusieurs lignes de conduite celle qui promet d'être la plus efficace pour atteindre un but. Elle fournit rapidement une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation difficile.

Les heuristiques peuvent être classées en deux catégories :

- **Méthodes constructives** qui génèrent des solutions à partir d'une solution initiale en essayant d'en ajouter petit à petit des éléments jusqu'à ce qu'une solution complète soit obtenue.

- **Méthodes de fouilles locales** qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière répétitive essaie d'améliorer cette solution en explorant son voisinage.

En effet, dans certains cas, le prix à payer pour utiliser des méthodes de programmation mathématique peut être une très grande complexité mathématique qui se traduit par des temps de calculs souvent longs. Parce qu'en pratique, une solution à un problème décisionnel doit être proposée dans des temps raisonnables, on sacrifie en partie la qualité de la solution au sens purement mathématique pour accélérer le processus de résolution.

- **Recherche Greedy (gloutonne)**

Un algorithme glouton fait, étape par étape, le choix d'un optimum local. A chaque itération, on fixe la valeur d'une des variables décrivant le problème sans remettre en cause les choix antérieurs.

Le principe est donc de partir d'une solution incomplète que l'on complète de proche en proche en effectuant des choix définitifs : à chaque étape, on traite une partie des variables sur lesquelles on ne revient plus.

Ces algorithmes sont faciles à mettre en œuvre, rapides mais ne fournissent pas toujours la solution optimale au problème donné [7].

On explore en priorité l'état qui semble le plus proche du but, selon l'heuristique `h(n)`. Construit une solution pas à pas sans revenir sur ses décisions, Effectue à chaque étape le choix qui semble le meilleur.

L'algorithme glouton utilise une fonction d'évaluation *f(n)* pour chaque nœud

$$f(n) = h(n) \text{ (heuristique)}$$

où :

- **f(n)**= estimation du coût de ***n*** à ***but***
- **h(n)** = distance en ligne droite de ***n*** à **but**

Les algorithmes gloutons sont connus par leur rapidité car ils vont droit au but par contre ils ne permettent pas de trouver la solution optimale.

Exemple : Pièces de monnaies

On considère le problème où l'on doit rendre la monnaie pour une certaine somme **S** avec le moins possible de pièces de monnaies sachant qu'on a des pièces de 1, 2, 5, 10, 20.

- Quelle est la solution optimale si je dois rendre S=26 ?

L'algorithme glouton va commencer par trier les types de pièces par valeurs décroissantes. Ensuite, pour chaque valeur de pièce, maximiser le nombre de pièces choisies. Et enfin répéter le choix de la pièce de plus grande valeur qui ne dépasse pas la somme restante tant que celle-ci n'est pas nulle.

- Avec S=26 (3 pièces)
  - 1 pièce de 20
  - 1 pièce de 5
  - 1 pièce de 1

Supposons maintenant qu'on a S = 26 à payer en pièces de 7, 6, 1

- Algorithme glouton (**8** pièces)
  - 3 Pièces de 7
  - 5 pièces de 1
- Solution optimale (**4** pièces)
  - 2 pièces de 7
  - 2 pièces de 6

On remarque dans cet exemple que l'algorithme est passé à coté de la solution optimale.

- **Algorithme A* (prononcer « A étoile »)**

Cette stratégie permet de faire une estimation du chemin complet (du nœud initial jusqu'à un but) et utilise une fonction d'évaluation ***f(n) = g(n) + h(n)***

Où :

g(n) = coût jusqu'à présent pour atteindre *n*
h(n) = coût estimé de *n* à un but
f(n) = coût total du chemin passant par *n* à un but

Une bonne heuristique doit être admissible : ne jamais surestimer le coût réel, plus l'heuristique est proche de la réalité, plus la recherche sera rapide et efficace.

Exemple : Voyageur de Roumanie

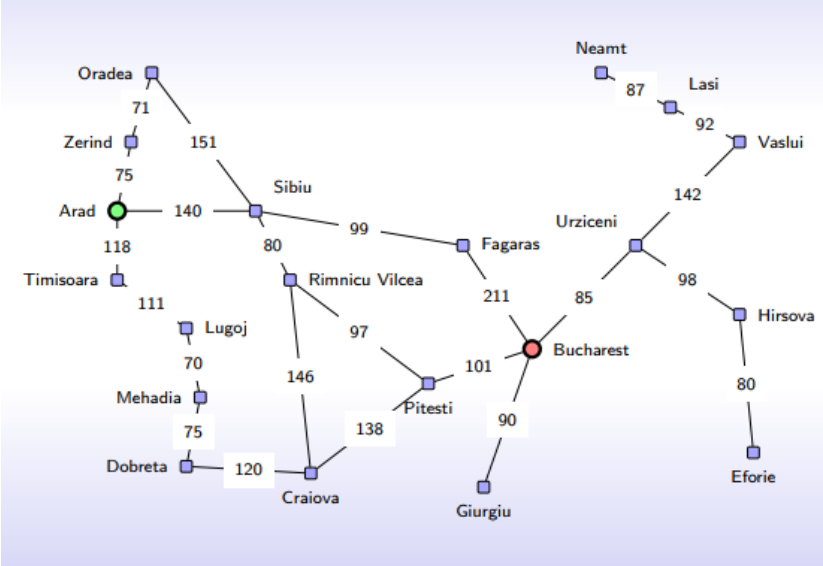


Figure 4.7 : Exemple voyageur de Roumanie

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244

Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Tableau 4.2 : Exemple voyageur de Roumanie

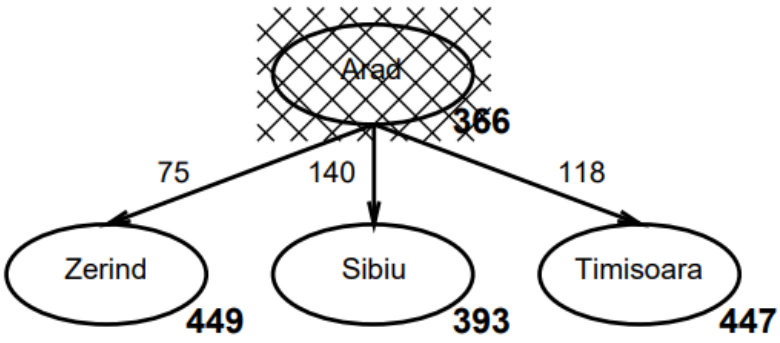


Figure 4.8 a : Déroulement exemple voyageur de Roumanie

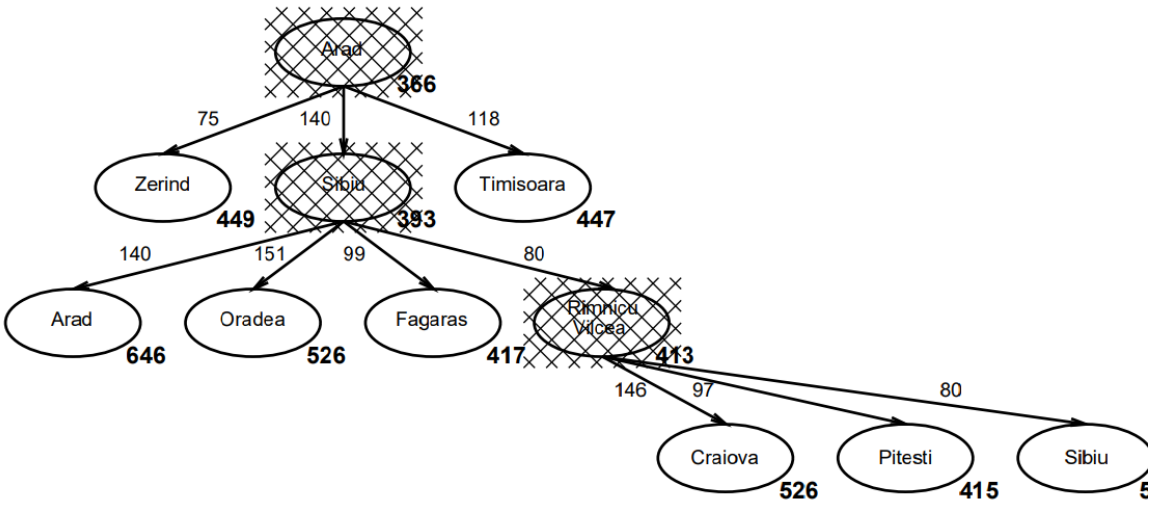


Figure 4.8 b : Déroulement exemple voyageur de Roumanie

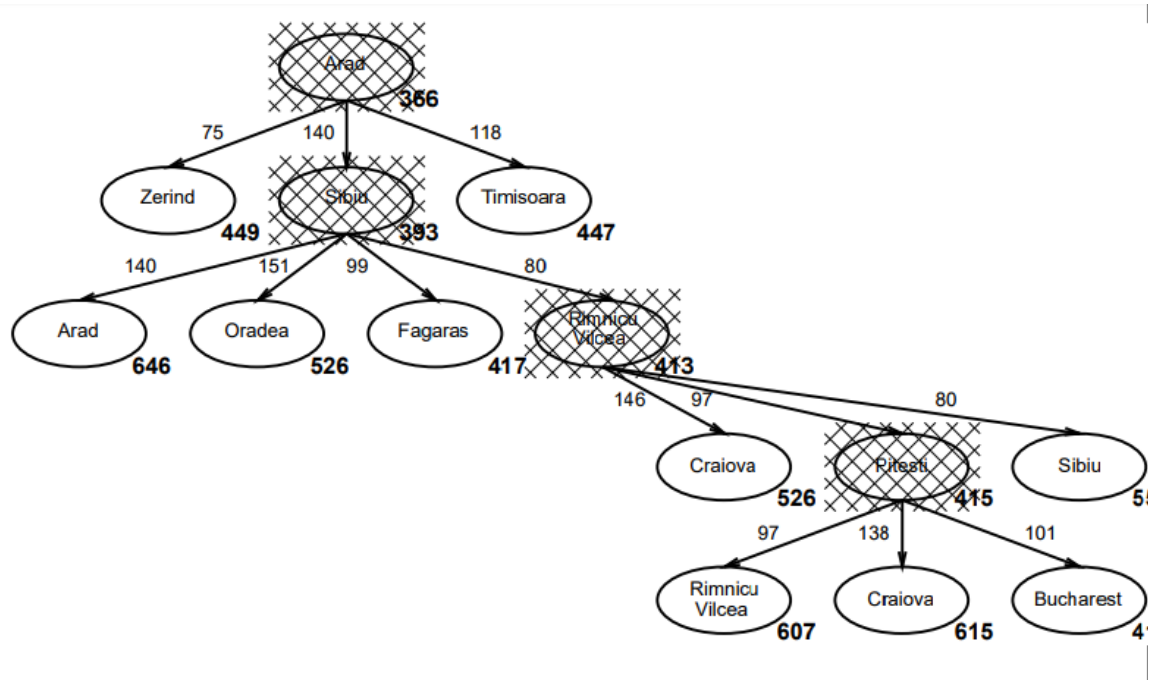


Figure 4.8 c : Déroulement exemple voyageur de Roumanie

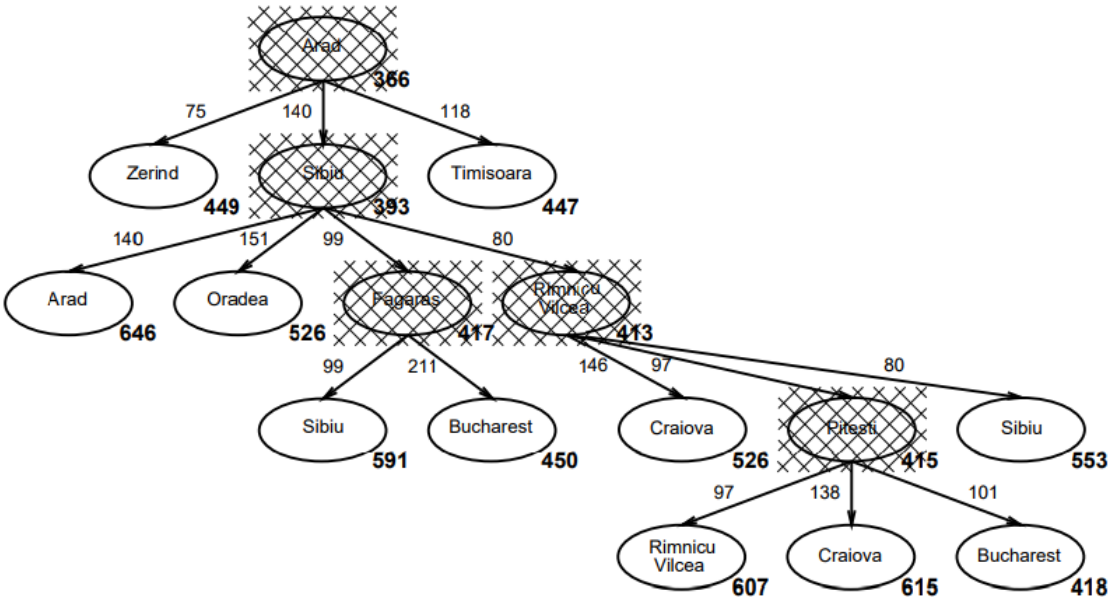


Figure 4.8 d : Déroulement exemple voyageur de Roumanie

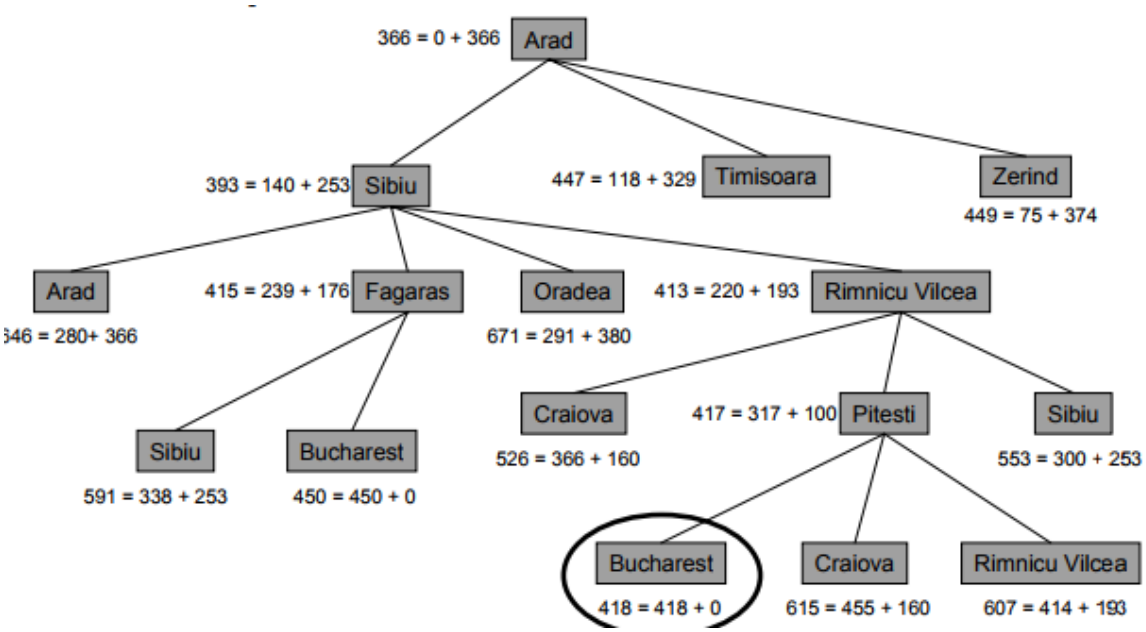


Figure 4.8 e : Déroulement exemple voyageur de Roumanie

4.7 Mesures de performance

- **Complétude** : est-ce que l'algorithme garantit d'obtenir une solution lorsqu'il en existe une ?
- **Optimalité** : est-ce que la stratégie trouve la solution optimale ?
- **Complexité en temps** : quelle est la durée pour l'obtention d'une solution ?
- **Complexité en espace** : de combien de mémoire faut-il disposer pour effectuer l'exploration ?

4.8 Conclusion

Résoudre un problème, en intelligence artificielle, c'est bien plus qu'appliquer des formules. C'est savoir modéliser une situation, explorer intelligemment les possibilités, et prendre des décisions malgré l'incertitude, les contraintes, ou l'ampleur des choix possibles.

On comprend alors que le cœur d'un système intelligent ne réside pas seulement dans sa capacité à traiter des données, mais aussi dans sa stratégie de raisonnement. Une IA efficace est une IA qui sait où chercher, comment chercher, et quand s'arrêter.

Ces techniques de recherche sont omniprésentes : dans les jeux, les assistants vocaux, les robots autonomes, les logiciels de planification, les systèmes de navigation GPS… Ce sont les briques fondamentales de l'intelligence artificielle symbolique.

Les recherches non informées sont simples et utiles quand on ne connaît rien du but, mais deviennent vite inefficaces. Par contre, les recherches informées (heuristique) sont plus intelligentes car elles exploitent des connaissances partielles pour guider la recherche.

A* est souvent considérée comme la méthode la plus équilibrée entre performance, précision et coût en mémoire, à condition d'avoir une bonne heuristique.

Chapitre 5 :

Problèmes de Satisfaction de Contraintes

(CSP)

5.1 Introduction

La satisfaction de contraintes est une discipline purement algorithmique issue de la programmation logique et de l'intelligence artificielle et apparue à la fin des années 1970 [8].

Beaucoup de problèmes algorithmiques de décision peuvent être reformulés en tant que problèmes de satisfaction de contraintes. Ces problèmes consistent à trouver des valeurs pour un ensemble de variables tout en respectant un ensemble de contraintes. Les CSP sont omniprésents dans des applications telles que la planification, l'ordonnancement, la configuration de systèmes, la vision par ordinateur et bien d'autres.

Ce chapitre explore les concepts fondamentaux des CSP, les méthodes de modélisation, les techniques de résolution, ainsi que des exemples illustratifs pour une meilleure compréhension.

5.2 Définition d'un CSP

Ensemble des problèmes, définis par des contraintes, et consistant à chercher une solution les respectant.

Un CSP est défini par un triplet (X,D,C) où :

X : un ensemble de variables {X1, X2,...,Xn}.
D : un ensemble de domaines {D1, D2,...,Dn}, chaque Di étant l'ensemble des valeurs possibles que peut prendre la variable Xi.
C : un ensemble de contraintes qui restreignent les combinaisons de valeurs que peuvent prendre les variables.

L'objectif est de trouver une affectation des variables telle que toutes les contraintes soient satisfaites.

Contrainte : Une contrainte est une relation logique établie entre différentes variables, chacune prenant sa valeur dans un ensemble qui lui est propre, appelé domaine.

Une contrainte est déclarative et relationnelle puisqu'elle définit une relation entre les variables sans spécifier de procédure opérationnelle pour assurer cette relation.

L'ordre dans lequel sont posées les contraintes n'est pas significatif : la seule chose importante à la fin est que toutes les contraintes soient satisfaites [7]

5.3 Exemples de CSP

Problème des N reines

Placer n reines sur un échiquier n×n de manière que aucune reine ne puisse en attaquer une autre.

- **Variables** : positions des reines sur l'échiquier.
- **Domaines** : lignes ou colonnes possibles.
- **Contraintes** : aucune reine ne partage la même ligne, colonne ou diagonale avec une autre.

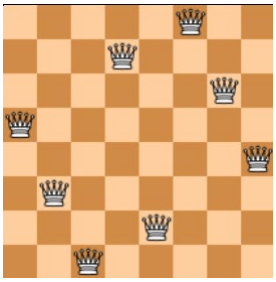


Figure 5.1 : N reines

Problème du Soduku

- **Variables** : Chaque case de la grille de Sudoku
- **Domaines** :
  L'ensemble des valeurs possibles pour chaque case, généralement :
  {1,2,3,4,5,6,7,8,9}
- **Contraintes** :

Chaque valeur de 1 à 9 ne doit apparaître **qu'une seule fois par ligne**.

Chaque valeur de 1 à 9 ne doit apparaître **qu'une seule fois par colonne**.

Chaque valeur de 1 à 9 ne doit apparaître **qu'une seule fois par sous-grille 3x3**.

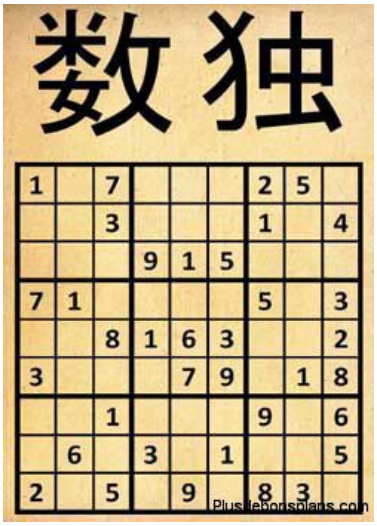


Figure 5.2 : Soduku

Coloration d'une carte

Attribuer des couleurs à des villes d'une carte géographique de manière que deux villes adjacentes n'aient pas la même couleur.

- **Variables** : chaque nœud du graphe.
- **Domaines** : ensemble des couleurs disponibles.
- **Contraintes** : deux nœuds adjacents doivent avoir des couleurs différentes.

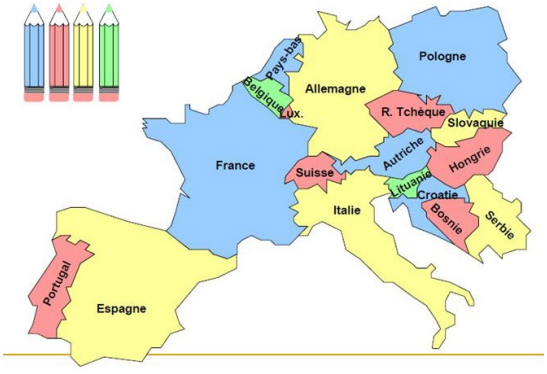


Figure 5.3 : Coloriage de carte

5.4 Modélisation d'un CSP

La modélisation d'un CSP implique :

1. **Identification des variables** : déterminer les éléments du problème qui peuvent varier.
2. **Définition des domaines** : spécifier les valeurs possibles pour chaque variable.
3. **Formulation des contraintes** : exprimer les restrictions ou relations entre les variables.

Une bonne modélisation est essentielle pour une résolution efficace du CSP.

5.5 Exemple illustratif de modélisation de CSP

Nous prenons l'exemple de coloration de graphe.
Quelle est le nombre minimal de couleur tel que deux nœuds adjacents reçoivent des couleurs différentes ?

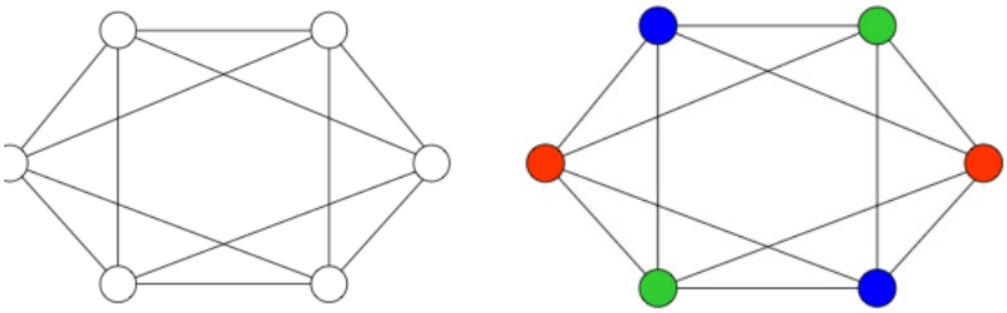


Figure 5.4 a : exemple coloriage de graphe

Il y a 6 nœuds, donc un maximum de 6 couleurs

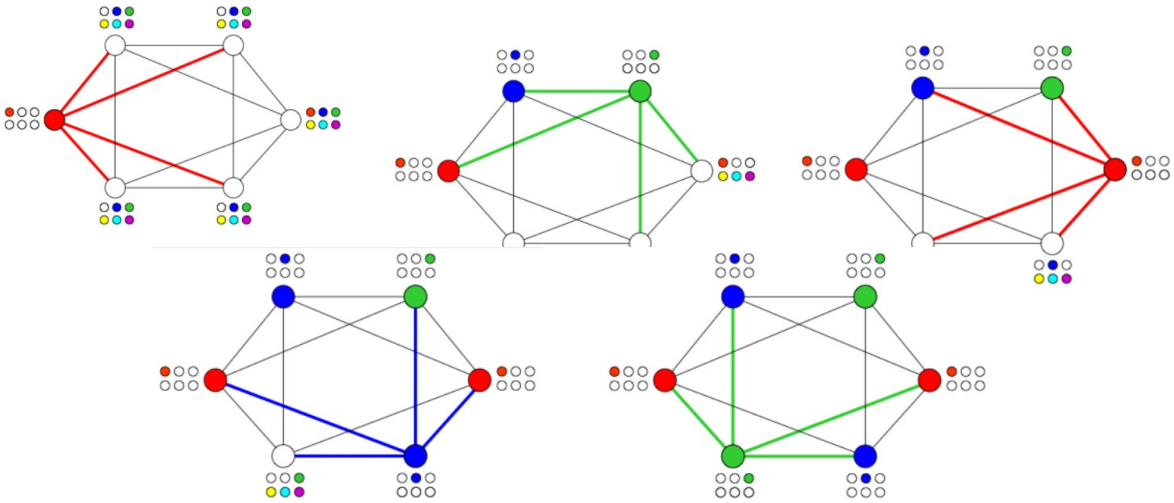


Figure 5.4 b-f : Exemple coloriage de graphe

- **<u>Etape 1 :</u> Identifier les variables**

Donner à chaque nœud un nom : A, B, C, D, E, F.

- **Etape 2 : Identifier les domaines**

  Initialement tous les nœuds peuvent être {rouge, bleu, vert, jaune, blue ciel, violet}

- **Etape 3 : Identifier les contraintes**

  Déclarer que deux nœuds adjacents ne peuvent prendre la même couleur

$$A \neq B,\ A \neq C,\ A \neq E,\ A \neq F,\ B \neq C,\ B \neq D,\ B \neq F,$$
$$C \neq D,\ C \neq E,\ E \neq E,\ D \neq F,\ E \neq F$$

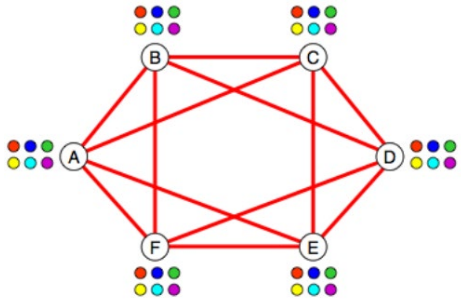


Figure 5.4 g : exemple coloriage de graphe

5.6 Résolution des CSP

La résolution de CSP est généralement combinatoire dans le sens où il faut envisager un très grand nombre de combinaisons avant d'en trouver une qui satisfasse toutes les contraintes.

Bien souvent, la puissance de calcul des ordinateurs ne suffit pas pour examiner toutes les combinaisons possibles en un temps acceptable, et il est nécessaire d'introduire des raisonnements et des heuristiques**.**

5.6.1 Approche Génère Et Teste

C'est une méthode naïve de résolution de problèmes qui consiste à générer et tester tous les ordres possibles. Chaque combinaison de variables et de valeurs est systématiquement générée et testée pour voir si elle satisfait toutes les contraintes du problème. La première combinaison satisfaisant toutes les contraintes est alors la solution du CSP. Résoudre un problème de cette manière devient rapidement impossible lorsque la taille des problèmes augmente.

```
fonction GET(A,(X,D,C)) : booléen
début
si toutes les variables de X sont affectées alors
   si A est consistante alors
      retourner VRAI
   sinon
      retourner FAUX
   finsi
sinon
   choisir une variable Xi de X qui n'est pas encore affectée
   pour toute valeur Vi appartenant à Di faire
      si GET(A ∪ {(Xi,Vi)}, (X,D,C)) = VRAI alors
         retourner VRAI
      finsi
   fin pour
   retourner FAUX
finsi
fin
```

Figure 5.5 : Algorithme générer et tester

5.6.2 Recherche avec retour arrière (Backtracking)

Le backtracking est une méthode récursive qui explore les affectations possibles des variables. À chaque étape, une valeur est assignée à une variable, et si une contrainte est violée, l'algorithme revient en arrière pour essayer une autre valeur.

Le retour en arrière est l'action de retourner vers une variable précédant la variable courante dans la sous-séquence d'instanciation, de modifier la valeur affectée à celle-ci et de poursuivre la résolution par la suite.

L'algorithme de backtracking est donc un algorithme à deux phases.

- Premièrement, une phase permettant d'avancer vers une prochaine variable à instancier et permettant d'étendre la sous-séquence et,
- Deuxièmement, une phase de retour en arrière consistant à choisir la variable vers laquelle sera fait le retour en arrière dans la sous-séquence.

L'inconvénient de cet algorithme est qu'il fait une détection tardive des conflits.

Algorithme de base :

1. Si toutes les variables sont assignées, retourner la solution.
2. Sélectionner une variable non assignée.
3. Pour chaque valeur possible du domaine de la variable :
   - Si la valeur est compatible avec les contraintes, assigner la valeur et continuer récursivement.
   - Sinon, essayer la valeur suivante.
4. Si aucune valeur ne mène à une solution, revenir en arrière.

```
fonction SRA(A,(X,D,C)) : booléen
début
si A n'est pas consistante alors retourner alors
   retourner FAUX
finsi
si toutes les variables de X sont affectées  alors
   retourner VRAI
sinon
   choisir une variable Xi de X qui n'est pas encore affectée
   pour toute valeur Vi appartenant à Di faire
      si SRA(A ∪ {(Xi,Vi)}, (X,D,C)) = VRAI alors
         retourner VRAI
      finsi
   fin pour
   retourner FAUX
finsi
fin
```

Figure 5.6 : Algorithme backtracking

5.6.3 Propagation de contraintes : Algorithme AC-3

L'algorithme AC-3 (Arc Consistency Algorithm 3) est utilisé pour réduire les domaines des variables en éliminant les valeurs qui ne peuvent pas faire partie d'une solution, en assurant la cohérence des arcs.

Principe :

1. Initialiser une file avec toutes les paires de variables liées par une contrainte.
2. Tant que la file n'est pas vide :
   - Retirer une paire (Xi,Xj) de la file.
   - Réviser le domaine de Xi pour assurer la cohérence avec Xj .
   - Si le domaine de Xi est modifié, ajouter toutes les paires (Xk,Xi) à la file, où Xk est voisin de Xi.

L'algorithme AC-3 est efficace pour réduire l'espace de recherche avant ou pendant la recherche avec backtracking.

5.7 Conclusion

Les Problèmes de Satisfaction de Contraintes offrent un cadre puissant pour modéliser et résoudre une grande variété de problèmes en intelligence artificielle et en informatique en général. La compréhension des concepts fondamentaux, des méthodes de modélisation et des techniques de résolution est essentielle pour aborder efficacement ces problèmes. Avec

l'évolution des outils et des algorithmes, les CSP continueront à jouer un rôle crucial dans le développement de solutions intelligentes et efficaces.

Les CSP reposent sur la définition explicite de règles que les solutions doivent obligatoirement respecter. On cherche une solution exacte en explorant un espace de possibilités tout en respectant ces contraintes. Cependant, dans de nombreuses situations réelles, les règles sont inconnues**,** incertaines ou trop complexes à modéliser manuellement (reconnaissance de visage, prédiction météo, …). Et donc, on passe à une approche basée sur les données : c'est là qu'intervient l'apprentissage automatique**.**

Chapitre 6 :

Apprentissage automatique

6.1 Introduction

L'apprentissage automatique, ou *machine learning*, est une branche de l'intelligence artificielle qui permet aux systèmes informatiques d'apprendre à partir de données, sans être explicitement programmés pour chaque tâche. Cette approche est devenue essentielle dans de nombreux domaines, tels que la reconnaissance vocale, la vision par ordinateur, la recommandation de contenus et bien d'autres.

C'est Alan Turing lui-même qui après avoir, en 1936, jeté les bases conceptuelles du calcul sur machine, donc de l'ordinateur. Il envisage en 1948 des « *learning machines* » susceptibles de construire elles-mêmes leurs propres codes.

6.2 Apprentissage

L'apprenrissage est le processus de construire un modèle général à partir de données (observations) particulières du monde réel. Ainsi, le but est double [7] :

- Prédire un comportement face à une nouvelle donnée.
- Approximer une fonction ou une densité de probabilité.

L'apprentissage est défini comme étant la capacité à améliorer les performances au fur et à mesure de l'exercice d'une activité.

6.3 Définition de l'apprentissage automatique

L'apprentissage automatique fait référence au développement, à l'analyse et à l'implémentation de méthodes qui permettent à une machine d'évoluer grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques.

L'apprentissage automatique consiste à développer des algorithmes capables d'améliorer leur performance sur une tâche donnée en analysant des données. Plutôt que de suivre des instructions précises, ces algorithmes détectent des motifs et des relations dans les données pour prendre des décisions ou faire des prédictions.

L'apprentissage automatique est également une technique de science des données qui permet aux ordinateurs d'utiliser des données existantes afin de prévoir les tendances, les résultats et les comportements futurs.

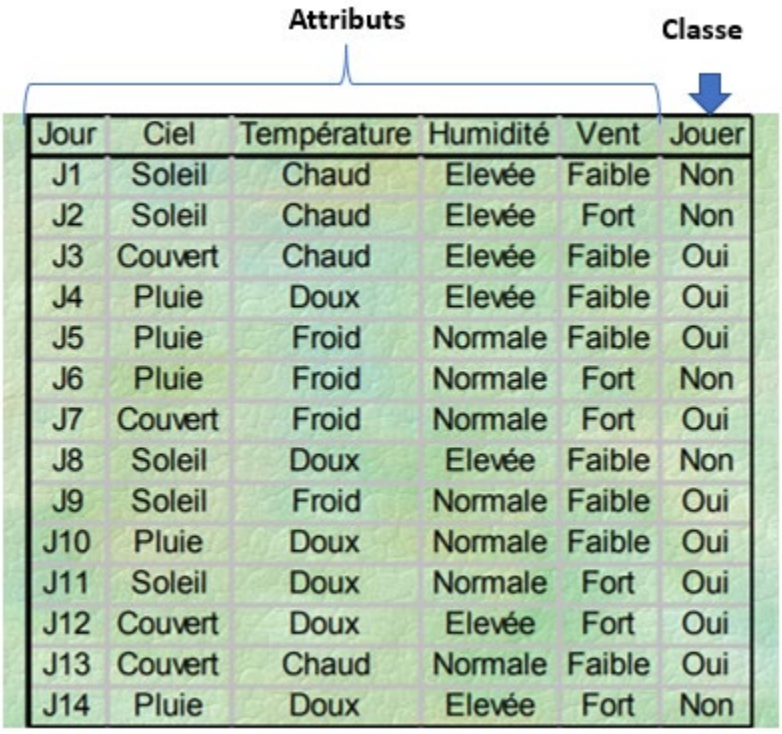
Objectif : extraire et exploiter automatiquement l'information présente dans un jeu de données.

6.4 Types d'apprentissage

L'apprentissage automatique se divise principalement en trois catégories :

6.4.1. Apprentissage supervisé

Dans ce type d'apprentissage, Un expert doit préalablement étiqueter des exemples, c'est-à-dire que chaque exemple d'entraînement est associé à une réponse correcte. L'objectif est que le modèle puisse prédire la sortie correcte pour de nouvelles entrées sachant que les classes sont prédéterminées et les exemples connus [7].



Jour	Ciel	Température	Humidité	Vent	Jouer
J1	Soleil	Chaud	Elevée	Faible	Non
J2	Soleil	Chaud	Elevée	Fort	Non
J3	Couvert	Chaud	Elevée	Faible	Oui
J4	Pluie	Doux	Elevée	Faible	Oui
J5	Pluie	Froid	Normale	Faible	Oui
J6	Pluie	Froid	Normale	Fort	Non
J7	Couvert	Froid	Normale	Fort	Oui
J8	Soleil	Doux	Elevée	Faible	Non
J9	Soleil	Froid	Normale	Faible	Oui
J10	Pluie	Doux	Normale	Faible	Oui
J11	Soleil	Doux	Normale	Fort	Oui
J12	Couvert	Doux	Elevée	Fort	Oui
J13	Couvert	Chaud	Normale	Faible	Oui
J14	Pluie	Doux	Elevée	Fort	Non

Figure 6.1 : Exemple données dans l'apprentissage supervisé

On peut citer les exemples suivants qui peuvent être résolus par l'apprentissage supervisé :

- Classification d'e-mails en spam / non-spam
- Reconnaissance de chiffres manuscrits
- Prédiction de la température du lendemain
- Diagnostic médical basé sur des symptômes

Autrement, on peut dire que l'apprentissage supervisé peut traiter les problèmes de :

- **Classification** (ex. : chat ou chien)

- **Régression** (ex. : prédire le prix d'une maison)

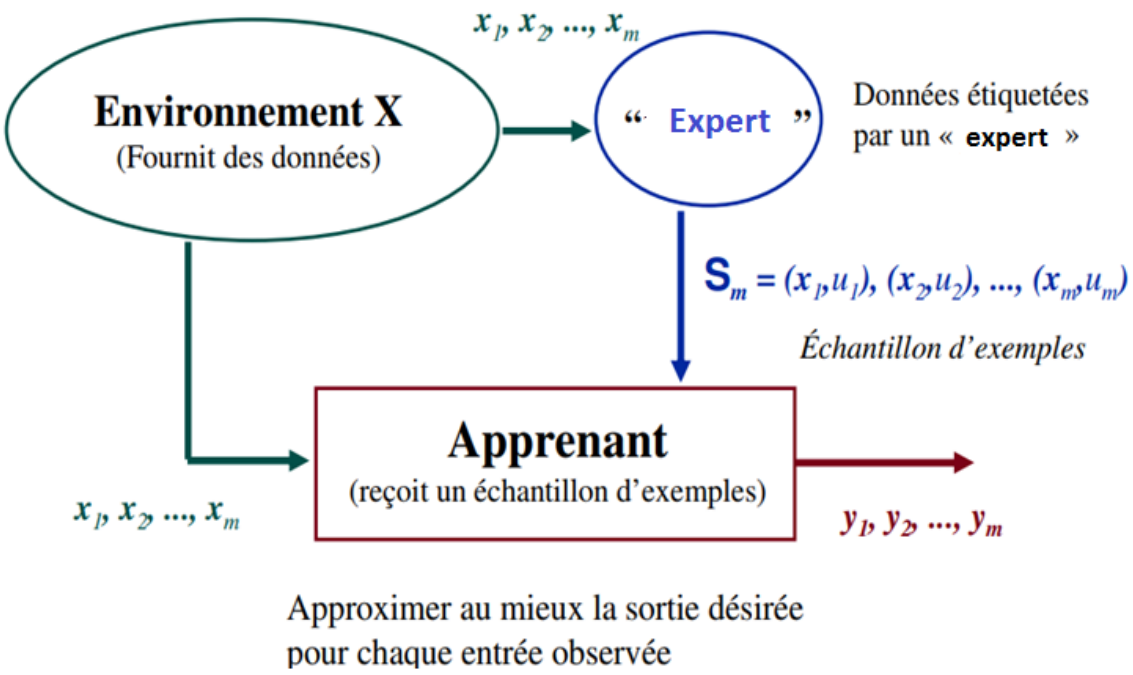


Figure 6.2 : Apprentissage supervisé

- **Exemples d'algorithmes** :
  - Régression linéaire
  - Arbres de décision
  - Machines à vecteurs de support (SVM)
  - Réseaux de neurones

6.4.2 Apprentissage non supervisé

Ici, l'algorithme reçoit des données sans étiquettes et doit découvrir des structures ou des motifs cachés. Il est souvent utilisé pour le regroupement ou la réduction de dimensions.

Dans ce type d'apprentissage, aucun expert n'est requis et le système ne dispose que d'exemples, mais non d'étiquettes, et le nombre de classes et leur nature n'ont pas été prédéterminés. L'algorithme doit découvrir par lui-même la structure plus ou moins cachée des données.

On peut citer les exemples suivants qui peuvent être résolus par l'apprentissage non supervisé :

- Regroupement de clients selon leur comportement d'achat
- Segmentation d'images (par couleurs)
- Détection d'intrusions dans un réseau

Autrement, on peut dire que l'apprentissage non supervisé peut traiter les problèmes de :

- **Clustering** (regroupement non supervisé)
- **Réduction de dimensionnalité** (ex. : PCA)

Le clustering consiste à séparer les données en k-groupes. Pour faire ce classement on va optimiser deux choses :

  - Maximiser l'homogénéité des données au sein des groupes, c'est à dire faire en sorte que les observations contenues dans un même groupe se ressemblent le plus possible ;
  - Maximiser l'hétérogénéité entre les groupes, c'est à dire faire en sorte que des observations appartenant à des groupes distincts soient les plus différentes possible.

Exemples d'algorithmes :

- K-moyennes (K-means)
- Analyse en composantes principales (PCA)
- Algorithmes de clustering hiérarchique

6.4.3 Apprentissage par renforcement

L'algorithme apprend en interagissant avec un environnement et en recevant des récompenses ou des pénalités en fonction de ses actions. Le but est de maximiser la récompense cumulée.

L'agent intelligent observe les effets de ses actions, déduit de ses observations la qualité de ses actions et améliore ses actions futures.

L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d'apprentissage. L'agent intelligent décide d'effectuer une action en fonction de son état pour interagir avec son environnement. L'environnement lui renvoie un renforcement sous la forme d'une récompense positive ou négative. Charge ensuite à l'agent de maximiser ce renforcement.

On peut citer les exemples suivants qui peuvent être résolus par l'apprentissage par renforcement :

- Un robot qui apprend à marcher.
- Un programme qui apprend à jouer à un jeu (comme AlphaGo).

Autrement, on peut dire que les composants de l'apprentissage par renforcement sont :

- **Agent** : celui qui apprend.
- **Environnement** : le monde avec lequel l'agent interagit.

- **État** : la situation actuelle.
- **Action** : décision prise par l'agent.
- **Récompense** : retour numérique indiquant si l'action est bénéfique.

L'apprentissage par renforcement est utilisé lorsqu'il y a de l'incertain dans la manière dont l'environnement évolue. Il se distingue des autres approches d'apprentissage par plusieurs aspects : L'apprentissage se fait sans supervision Il repose sur le principe d'essai/erreur.

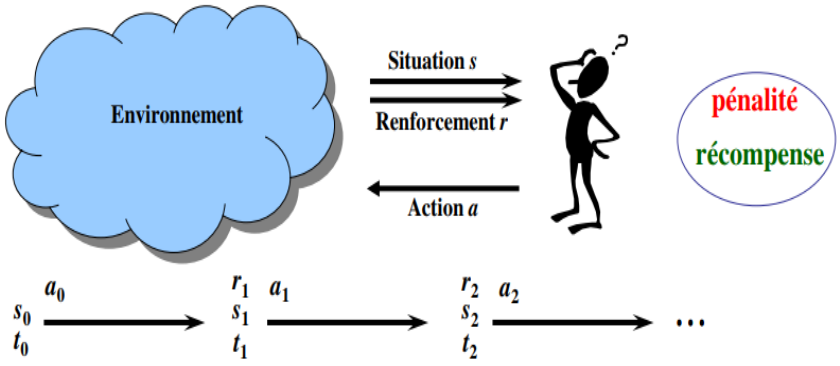


Figure 6.3 : Apprentissage par renforcement

Exemples d'algorithmes :

- Q-learning
- Deep Q-Networks (DQN)
- Policy Gradient Methods

Il existe aussi d'autres types d'apprentissage automatique, en dehors du supervisé, non supervisé et renforcement qui sont les plus connus. On peut citer [10] :

6.4.4 Apprentissage semi-supervisé (Semi-supervised Learning)

C'est un mélange entre supervisé et non supervisé. On dispose de peu de données étiquetées et de beaucoup de données non étiquetées. Le modèle utilise les deux pour apprendre plus efficacement. Ce type d'apprentissage permet de réduire les coûts d'annotation tout en profitant des avantages du supervisé.

6.4.5 Apprentissage auto-supervisé (Self-supervised Learning)

L'algorithme crée lui-même ses étiquettes à partir des données brutes. Il s'agit d'un sous-domaine en forte croissance, notamment en traitement du langage (NLP) et vision par ordinateur**.** Ce type d'apprentissage est utilisé pour le pré-entraînement de modèles sur de très grandes bases.

6.4.6 Apprentissage en ligne (Online Learning)

L'apprentissage se fait progressivement, exemple par exemple, sans avoir besoin de tout le jeu de données à l'avance. Ce type d'apprentissage est idéal pour les flux continus de données ou **l'**adaptation en temps réel**.**

6.4.7 Deep learning

Le deep learning (apprentissage profond) est une méthode d'apprentissage automatique qui repose sur des réseaux de neurones artificiels à plusieurs couches (d'où "profond").

Le tableau 6.1 représente les différences majeures entre machine learning et deep learning.

Machine Learning (classique)	Deep Learning
Nécessite d'extraire manuellement les caractéristiques	Apprend automatiquement les caractéristiques
Moins gourmand en données	Nécessite de grandes quantités de données
Fonctionne bien avec des données structurées (tableaux)	Excelle avec données non structurées (images, son, texte)

Tableau 6.1 : Machine learning vs deep learning

6.5 Données d'apprentissage

Dans tout processus d'apprentissage automatique, la qualité et l'organisation des données jouent un rôle fondamental dans la performance du modèle. Avant même d'entraîner un algorithme, il est essentiel de diviser les données disponibles en ensembles distincts**,** chacun ayant une fonction bien précise dans le cycle de modélisation.

Cette structuration permet non seulement de former efficacement le modèle, mais aussi de mesurer sa capacité de généralisation et d'**éviter** les biais tels que le surapprentissage (overfitting).

Le sur-apprentissage se produit lorsque le modèle s'adapte trop étroitement aux données d'entraînement, capturant le bruit au lieu des motifs sous-jacents. Cela entraîne une mauvaise performance sur de nouvelles données.

Il faut savoir qu'en général, l'erreur d'apprentissage diminue à chaque étape et que si l'erreur réelle (de validation) **diminue**, se **stabilise** et puis **augmente**, on parle alors du phénomène de sur-apprentissage qu'il faut absolument éviter, et pour cela il faut généralement utiliser plus de données, simplifier le modèle et appliquer des techniques de régularisation.

Avant de commencer le traitement des données, une étape de prétraitement est parfois nécessaire qui consiste à nettoyer, normaliser ou standardiser les données

Ainsi, on distingue généralement trois ensembles complémentaires :

- L'ensemble d'apprentissage **:** constitue l'ensemble des candidats ou exemples (images, attributs, DB, …) utilisés pour générer le modèle d'apprentissage.
- L'ensemble de validation **:** peut être utilisé lors de l'apprentissage (comme sous population de l'ensemble d'apprentissage) afin de valider (intégrer) le modèle et d'éviter le sur apprentissage.
- L'ensemble de test **:** est constitué des candidats sur lesquels sera appliqué le modèle d'apprentissage (pour tester et corriger l'algorithme).

Ces ensembles sont utilisés à différents stades du processus pour entraîner**,** valider**,** puis évaluer la robustesse du modèle de manière objective.

6.6 Critères de performance

L'évaluation de la performance d'un classifieur se fait par le taux de bonnes classifications.

Par exemple, si pour 100 exemples de tests, 89 ont été prédits correctement par notre modèle de prédiction, on pourra dire que ce modèle a une précision de 89% (souvent écrit 0,89).

Mais la précision n'est pas le seul critère à prendre en compte, notamment pour les problèmes à deux classes. En effet, lorsque l'on travaille avec des modèles de prédiction binaires, de nouveaux critères peuvent entrer en jeu (par exemple, dans le domaine médical).

Pour évaluer un modèle d'apprentissage automatique, plusieurs critères sont utilisés :

- **Précision** : proportion de prédictions correctes.
- **Rappel** : capacité à identifier toutes les instances positives.
- **F-mesure** : moyenne harmonique de la précision et du rappel.
- **Courbe ROC** : représente le compromis entre le taux de vrais positifs et de faux positifs.

Matrice de confusion

La matrice de confusion est un outil d'évaluation utilisé pour mesurer les performances d'un modèle de classification. Elle permet de comparer les prédictions du modèle aux valeurs réelles (attendues) afin de visualiser les types d'erreurs commises.

La matrice de confusion est un outil d'évaluation des performances d'un modèle de classification supervisée. Elle permet de visualiser les résultats des prédictions du modèle par rapport aux vraies classes des données.

C'est un tableau 2x2 (dans le cas d'un problème binaire) qui décompose les prédictions en quatre catégories

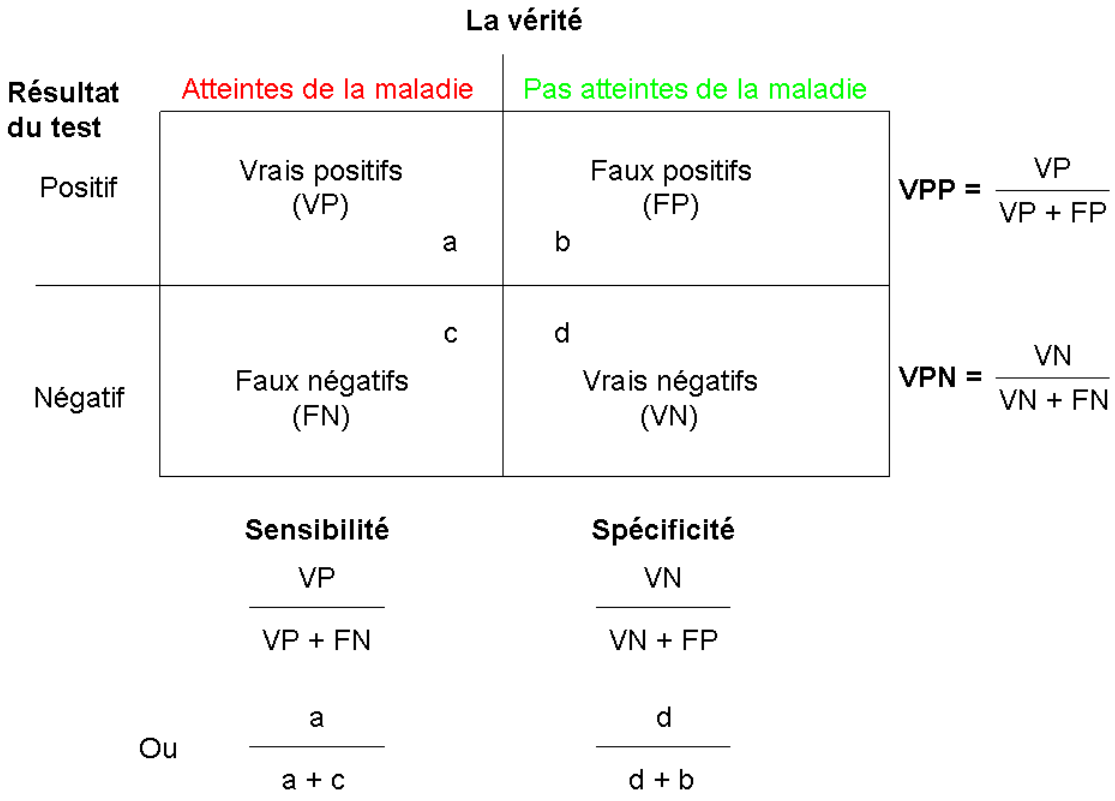


Figure 6.4 : Matrice de confusion

- **VP (Vrai Positif)** : le modèle a correctement prédit un cas positif.
- **VN (Vrai Négatif)** : le modèle a correctement prédit un cas négatif.
- **FP (Faux Positif)** : le modèle a prédit un cas positif à tort (erreur de type I).
- **FN (Faux Négatif)** : le modèle a raté un cas positif (erreur de type II).

À partir de la matrice, on peut calculer plusieurs indicateurs utiles :

- La **sensibilité** représente la proportion de personnes vraiment atteintes de la maladie, dans la population ciblée, qui sont identifiées par le test de dépistage comme étant atteintes de la maladie (c'est-à-dire qu'elles ont des résultats élevés).

  La sensibilité indique la probabilité que le test diagnostiquera correctement un cas.

- La **spécificité** représente la proportion de personnes *non atteintes de* la maladie qui a des résultats peu élevés sur le test de dépistage: la probabilité que le test identifiera correctement une personne n'étant pas atteinte de la maladie.

- L'exactitude (Accuracy) représente le taux de bonne classification :$acc = tbc = TP + TN\ TP + FN + TN + FP \approx TP + FN * tpr + FP + TN * 1 - fpr$
- La sensibilité et la spécificité d'une classification doivent toujours être données ensemble. On peut d'ailleurs définir l'indice de Youden :

$$IYouden = Se + Sp - 1$$

Cet indice révèle l'efficacité de la prédiction : S'il est négatif ou nul, la classification est inefficace. Elle est d'autant plus efficace qu'il se rapproche de 1.

6.7 Conclusion

L'apprentissage automatique permet aux systèmes informatiques de déduire automatiquement des modèles à partir de données, sans être explicitement programmés pour chaque tâche. Grâce à différentes approches : supervisées, non supervisées ou par renforcement ; les algorithmes peuvent effectuer des prédictions, des classifications ou encore détecter des anomalies avec une efficacité croissante.

Cependant, malgré leur performance, les méthodes classiques d'apprentissage automatique atteignent parfois leurs limites face à des données complexes, non structurées ou de très grande dimension, comme les images, le son ou le langage naturel.

C'est dans ce contexte qu'émergent les réseaux de neurones, inspirés du fonctionnement du cerveau humain. Ces modèles constituent une classe puissante d'algorithmes capables d'extraire automatiquement des représentations hiérarchiques des données, ouvrant la voie à des avancées majeures dans des domaines tels que la vision par ordinateur, la traduction automatique, ou encore la reconnaissance vocale.

Le chapitre suivant sera donc consacré à l'étude de ces réseaux neuronaux artificiels, en abordant leur architecture de base, leur fonctionnement, ainsi que les différentes variantes qui ont révolutionné le domaine de l'intelligence artificielle.

Chapitre 7 :

Les Réseaux de Neurones Artificiels

7.1 Introduction

L'inspiration derrière les réseaux de neurones artificiels vient du cerveau humain. L'idée est de reproduire, de façon simplifiée, le fonctionnement des neurones biologiques pour permettre à une machine d'apprendre, raisonner et reconnaître des motifs complexes à partir de données.

Aujourd'hui, les réseaux de neurones sont la base du deep learning et sont utilisés dans une multitude d'applications : reconnaissance d'images, traitement du langage, conduite autonome, etc.

7.2 Définition d'un réseau de neurone artificiel

Un neurone artificiel est une unité de calcul. Il reçoit des entrées pondérées, effectue une somme, applique une fonction d'activation, et produit une sortie.

Les réseaux de neurones artificiels sont un moyen de modéliser le mécanisme d'apprentissage et de traitement de l'information qui se produit dans le cerveau humain.

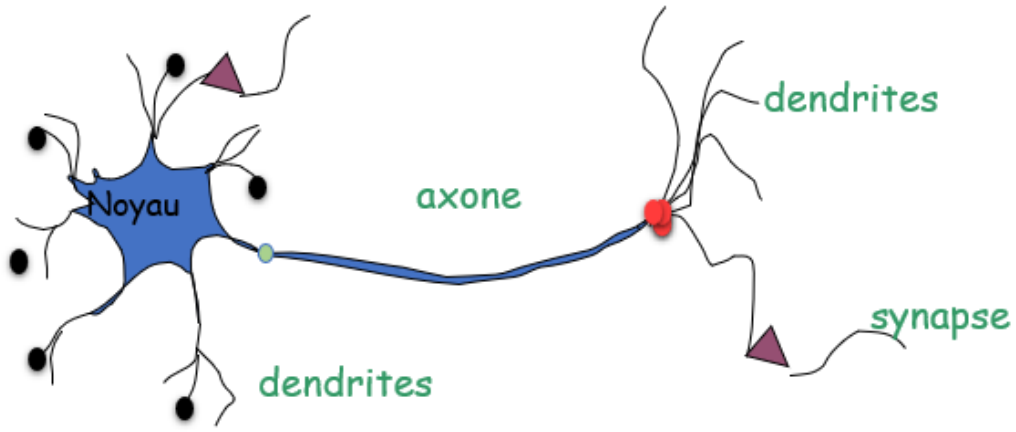


Figure 7.1 : Neurone biologique

Le neurone artificiel est la modélisation mathématique du neurone biologique.

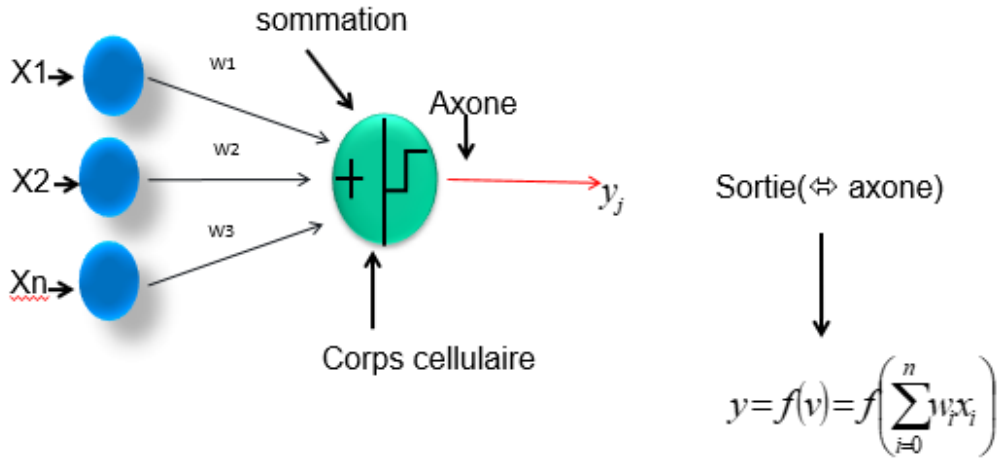


Figure 7.2 : Neurone artificiel

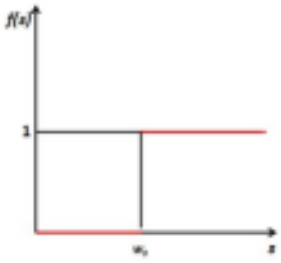
Tels que :

- (x1 , x 2 ,…, xm) : sont les entrées du neurone (signaux qui lui parviennent).
- (w1 , w2 , ..., w n): les poids associés à chaque connexion (importance de chaque signa).
- b : le seuil d'activation (biais).
- S : la somme pondérée des entrées (potentiel d'activation).

$$S = \sum_{i=1}^{n} w_i x_i + b$$

- ŷ = f(S) : la sortie du neurone (réponse du neurone « activé ŷ =1, ou non activé ŷ =0»).
- ŷ = { 1 $si\ S > 0$

  0 $si\ S < 0$}



7.3 Structure d'un réseau de neurones

Un réseau de neurones est composé de **plusieurs couches de neurones** :

- **Couche d'entrée** : reçoit les données.
- **Couches cachées** : effectuent des transformations successives.
- **Couche de sortie** : donne le résultat final.

Par exemple, pour une image de chiffre manuscrit (28x28 pixels), chaque pixel est une **entrée**, et la **sortie** est la prédiction du chiffre (de 0 à 9).

Les neurones sont organisés en couches successives : chaque neurone reçoit une information (une entrée) issue des neurones de la couche qui précède. Chacune de ces informations est pondérée : elle est multipliée par une valeur qui lui confère un "poids" wi particulier. Les entrées ainsi pondérées sont additionnées.

Elles sont traitées par une fonction objective dont le but est d'adapter la valeur de sortie à une plage de valeurs. La valeur de sortie issue de cette fonction constitue l'entrée de l'ensemble des neurones de la couche suivante.

7.3.1 Le perceptron

C'est un des premiers réseaux de neurones, conçu en 1958 par Rosenblatt. Il consiste en un seul neurone qui possède un seuil θ ainsi qu'un vecteur de poids synaptiques ajustables (w1 ,w2 ,…,wn) tout comme le modèle de neurone de McCulloch & Pitts [4].

Il permet de classifier correctement des objets (représentés par les vecteurs X(x1 ,x 2 , …,x n)) appartenant à deux classes (C1 et C2) linéairement séparables.

La sortie du neurone ne peut prendre que deux états (-1 et 1). La fonction de signe, définie comme fonction d'activation du neurone. La sortie du neurone vaut alors +1 si l'objet X appartient à la classe C1, et -1 dans le cas contraire (X appartient à la classe C2).

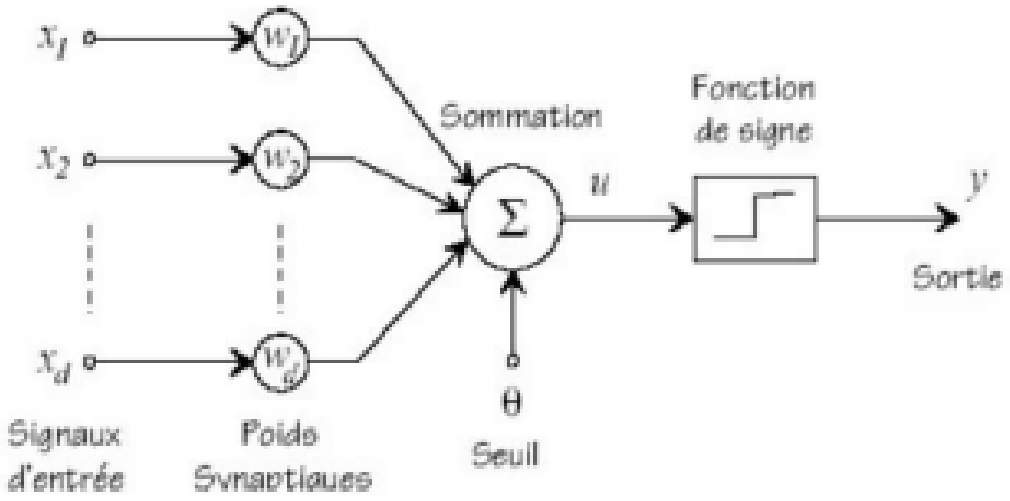


Figure 7.4 : Neurone de perceptron

La reconnaissance de plusieurs classes est rendue possible par la mise en parallèle de plusieurs neurones (une couche de neurones).

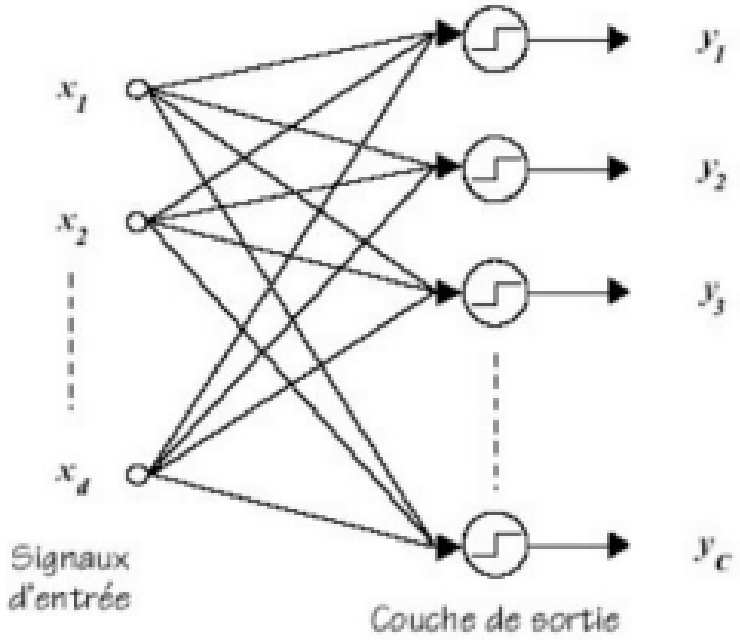


Figure 7.5 : Réseau de perceptron monocouches

7.3.2 Le perceptron multicouche (Multi Layer Perceptron)

Le perceptron multicouche est un réseau de neurones organisé en couches successives : On utilise le plus souvent des réseaux particuliers organisés en couches. L'influx d'information va toujours des couches d'entrées aux couches de sorties. Ces réseaux peuvent être appris par descente de gradient. Ils sont adaptés aux données de tailles fixes, comme des images.

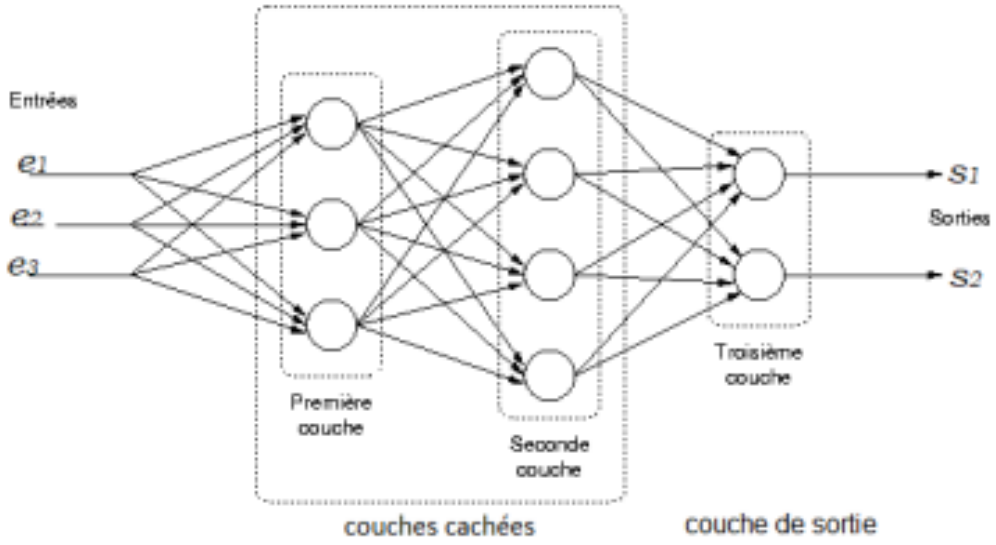


Figure 7.6 : Perceptron multicouches

7.4 Apprentissage d'un réseau : la rétropropagation

Le réseau apprend en ajustant les poids grâce à l'algorithme de rétropropagation [4][9]de l'erreur (backpropagation) :

1. **Propagation avant (forward)** : on calcule la sortie du réseau.
2. **Calcul de l'erreur** : différence entre la sortie et la cible.
3. **Rétropropagation** : on propage l'erreur en arrière pour ajuster les poids.
4. **Mise à jour des poids** : via un optimiseur (ex. : **gradient descendant**, Adam).

Ce processus est répété sur des milliers d'exemples.

7.5 Fonctions d'activation courantes

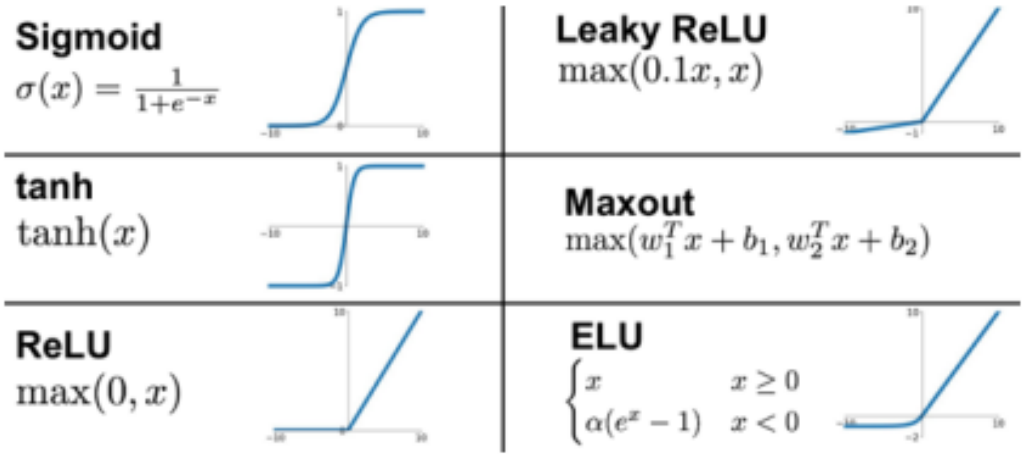


Figure 7.7 : Fonctions d'activation

La fonction sigmoide est utilisée lorsque la sortie est binaire 0 ou 1, par contre la fonction Tanh est utilisée lorsque les données sont centrées autour de 0. Dans les réseaux profonds, on utilise souvent la fonction ReLu tandis que la fonction Softmax est utilisée dans la dernière couche en classification.

7.6 Réseaux profonds (Deep Neural Networks)

Un réseau devient "profond" lorsqu'il contient plusieurs couches cachées [10]. Ces couches permettent au modèle d'apprendre des représentations hiérarchiques : Les premières couches détectent des motifs simples (lignes, bords…)

Les couches profondes combinent ces motifs pour identifier des formes complexes (visages, objets…)

Les Composants clés d'un réseau profond sont :

- Neurone artificiel
- Réseau de neurones multicouches (MLP)
- Fonctions d'activation (ReLU, sigmoid)
- Rétropropagation de l'erreur (backpropagation)
- Optimiseur (SGD, Adam)

7.6.1 Réseaux convolutifs (CNN – Convolutional Neural Networks)

Les CNN sont utilisés dans les images, vision par ordinateur et fonctionnent avec des filtres qui capturent les motifs visuels locaux.

7.6.2 Réseaux récurrents (RNN – Recurrent Neural Networks)

Les RNN sont utilisés pour des données séquentielles (texte, audio) et ils ont une mémoire des états précédents (utile pour les phrases, la musique, etc.)

7.6.3 Réseaux Transformers

Les transformers sont utilisés pour le texte, traduction, chatbots et ils font un traitement parallèle et ont performances exceptionnelles (modèles comme BERT, GPT).

7.7 Applications des réseaux de neurones

- Reconnaissance faciale sur smartphones
- Traduction automatique (ex. : Google Translate)
- Détection de fraudes bancaires
- Analyse médicale (ex. : détection de tumeurs)
- Voitures autonomes
- Génération de texte ou d'images (comme DALL·E ou ChatGPT)

Le tableau 7.1 suivant, nous donne plusieurs exemples de problèmes et le type de réseau de neurones qu'on peut utiliser pour le résoudre.

Exemple	Type de données	Différentes couches	Type de réseau
Classification 2D	Coordonnées (x, y)	**Entrées** : (x, y) – deux coordonnées **Sortie attendue** : 0 (bleu) ou 1 (rouge)	Perceptron / MLP
Reconnaissance de chiffres	Images (28x28)	**Entrées** : images de chiffres manuscrits **Sortie** : classe parmi {0, 1, ..., 9}	MLP / CNN
Reconnaissance vocale	Audio	**Entrée** : signal audio (converti en spectrogramme) **Sortie** : séquence de mots	RNN / Transformer

Diagnostic médical	Images médicales	**Entrée** : image (scanner, IRM) **Sortie** : binaire (malade / sain) ou score de probabilité	CNN
Génération de texte	Texte	**Entrée** : quelques mots (ex. : "Aujourd'hui je vais…") **Sortie** : suite du texte	Transformer

Tableau 7.1 : Exemples de types de réseau de neurones

7.8 Conclusion

L'apprentissage automatique est un domaine dynamique qui permet aux machines d'apprendre et de s'adapter à partir de données. En comprenant ses types, ses méthodes d'évaluation et ses défis, on peut concevoir des systèmes intelligents capables de résoudre des problèmes complexes dans divers domaines.

Les réseaux de neurones artificiels ont révolutionné le champ de l'intelligence artificielle. Grâce à leur capacité à apprendre des motifs complexes dans des données brutes, ils permettent de résoudre des tâches autrefois réservées à l'intelligence humaine.

Ils forment la colonne vertébrale du deep learning, et leur évolution continue promet des innovations spectaculaires dans les années à venir.

Chapitre 8 :

La Cybernétique et ses liens avec l'Intelligence Artificielle

8.1 Introduction

Avant que l'intelligence artificielle ne devienne ce que nous connaissons aujourd'hui, un courant scientifique a jeté les bases d'une réflexion sur les machines capables de "penser" ou de "s'autoréguler" : la cybernétique.
Fondée dans les années 1940 par Norbert Wiener, la cybernétique est une discipline à la croisée de l'informatique, de la biologie, des mathématiques, de la physique et de la psychologie. Son objectif : étudier les mécanismes de communication et de contrôle dans les systèmes vivants et artificiels.

8.2Définition de la cybernétique

La cybernétique est une science des systèmes de contrôle et de régulation, qu'ils soient biologiques, mécaniques ou informatiques.

Elle a introduit la notion de "système intelligent", c'est-à-dire capable d'agir, de s'adapter, voire de se corriger.

Elle est appliquée dans plusieurs domaines tels que :

- **Robotique adaptative** : robots capables d'ajuster leurs comportements (comme les drones ou exosquelettes).
- **Systèmes autonomes** : véhicules ou usines qui s'auto-régulent en temps réel.
- **Neurosciences computationnelles** : modélisation du cerveau comme système de contrôle.

8.3 Enjeux éthiques et limites de l'intelligence artificielle

L'intelligence artificielle ouvre des perspectives impressionnantes, mais elle soulève aussi de nombreuses interrogations. Peut-on faire confiance à une machine pour prendre une décision qui nous concerne ? Comment éviter les dérives ? Qui est responsable en cas d'erreur ? Ces questions ne sont pas secondaires : elles sont au cœur des débats actuels sur l'IA [9].

8.3.1 Biais et discriminations

L'un des problèmes majeurs des systèmes d'IA est le biais algorithmique. Une IA apprend à partir de données… or ces données ne sont jamais neutres. Si l'historique utilisé pour entraîner un modèle contient des préjugés, des déséquilibres ou des discriminations (souvent inconscientes), l'IA risque de les reproduire — voire de les amplifier.

Par exemple, un algorithme de recrutement pourrait désavantager les femmes si les données passées favorisaient systématiquement les candidatures masculines. Des études ont aussi montré que certains logiciels de reconnaissance faciale étaient beaucoup moins précis pour

les personnes à la peau foncée. Ces biais peuvent avoir des conséquences graves, notamment lorsqu'il s'agit de décisions médicales, juridiques ou financières.

8.3.2 Transparence et explicabilité

De nombreux systèmes d'IA — notamment ceux fondés sur des réseaux de neurones profonds — sont souvent qualifiés de boîtes noires. Ils donnent des résultats, parfois très justes, mais sans que l'on sache vraiment *comment* ils sont parvenus à leur conclusion. Cela pose un problème évident dans des contextes sensibles : un médecin, un juge, un enseignant ou un assureur doivent pouvoir expliquer une décision. Or, il est difficile de justifier une action prise par un modèle qu'on ne comprend pas entièrement.

Ce défi a donné naissance à un champ de recherche spécifique : **l'**explicabilité de l'IA (*explainable AI* ou XAI). L'objectif est de rendre les systèmes plus transparents, compréhensibles et auditables.

8.3.3 Automatisation et emploi

L'IA automatise de plus en plus de tâches : diagnostic médical, contrôle qualité, rédaction de rapports, gestion de la relation client, etc. Cela peut libérer du temps pour les humains… mais cela soulève aussi une inquiétude bien réelle : la disparition de certains emplois.

Des métiers répétitifs ou à faible valeur ajoutée sont les premiers concernés. Mais à plus long terme, même des professions qualifiées pourraient être partiellement remplacées ou transformées par l'IA. Cela oblige à repenser l'éducation, la formation continue, et l'organisation du travail. L'enjeu n'est pas seulement technique, il est profondément social.

8.3.4 Responsabilité et sécurité

Si une IA prend une mauvaise décision, qui est responsable ? Le concepteur ? L'utilisateur ? L'entreprise qui l'a déployée ? Cette question juridique est encore floue dans de nombreux pays. Elle devient d'autant plus critique avec l'émergence des systèmes autonomes, comme les véhicules sans conducteur ou les drones intelligents.

Par ailleurs, certains systèmes d'IA peuvent être détournés pour des usages malveillants : désinformation automatisée, surveillance de masse, cyberattaques, etc. Il est donc crucial de développer une IA de confiance, encadrée par des principes de sécurité, d'éthique et de droit.

8.3.5 Déshumanisation des interactions

Un autre risque, plus subtil mais tout aussi réel, est celui d'une perte de lien humain. Si l'on remplace progressivement les enseignants, les conseillers, les médecins ou les artistes par des intelligences artificielles, que devient la relation humaine dans ces échanges ? Même si l'IA peut simuler l'empathie, elle ne la ressent pas. L'humain a besoin d'écoute, de compréhension authentique, de contact. Il ne faut pas que la technologie nous fasse perdre cette dimension essentielle.

8.4 Vers une IA responsable

L'éthique de l'IA n'est pas un sujet réservé aux philosophes ou juristes. C'est un enjeu technique, social et moral. Chaque développeur ou utilisateur de modèles IA doit se poser des questions :

- Est-ce que mon modèle traite tout le monde de manière juste ?
- Quelles données j'utilise, et d'où viennent-elles ?
- Que faire si l'IA se trompe de manière critique ?

L'idée n'est pas d'interdire l'IA, mais de la concevoir de manière plus consciente, inclusive et juste.

8.5 Conclusion

L'Intelligence Artificielle est devenue un levier essentiel de transformation dans de nombreux secteurs. Ses applications améliorent l'efficacité, la sécurité et la qualité de vie. Cependant, il est crucial d'accompagner ces avancées technologiques d'une réflexion éthique et d'une régulation appropriée pour garantir un développement responsable et équitable de l'IA.

L'IA n'est pas infaillible, ni impartiale par nature. Elle hérite de nos biais et peut même les aggraver si l'on n'y prend pas garde. Reconnaître cette réalité est une première étape vers une intelligence artificielle éthique et équitable. En tant que concepteurs, chercheurs ou utilisateurs, nous avons un rôle de responsabilité.

La cybernétique est l'un des piliers théoriques de l'intelligence artificielle moderne. Si l'IA actuelle s'appuie surtout sur les données et les algorithmes d'apprentissage, la cybernétique rappelle qu'un système intelligent est aussi un système capable d'équilibre, de contrôle et de régulation. Revisiter cette discipline enrichit notre compréhension des racines de l'IA et de ses perspectives interdisciplinaires.

Exercices applicatifs

Exercice 1 :

Pour chaque exemple ci-dessous, indique s'il s'agit d'un système classique ou d'un système intelligent. Justifie ta réponse en une ou deux phrases.

Exemple	Type de système	Justification
Un ascenseur qui monte ou descend selon l'étage demandé		
Une voiture autonome qui s'adapte au trafic en temps réel		
Une machine à laver qui lance un cycle programmé		
Un chatbot qui comprend des demandes en langage naturel		
Un feu de circulation avec minuterie fixe		
Un système de recommandation sur YouTube		

Exercice 2:

Parmi trois personnes Ahmed, Bachir et Choukri il y a un qui ment et le reste dit toujours la vérité. On sait que:

- Ahmed dit la vérité si Bachir la dit aussi,
- Bachir ment si Choukri le fait aussi.

Qui est le menteur ?

Exercice 3:

Trois collègues, Ahmed, Bachir et Choukri dinnent ensemble chaque Vendredi. Les affirmations suivantes sont vraies :

- Si Ahmed commande un dessert, Bachir en commande un aussi.
- Chaque jour, soit Bachir, soit Choukri, mais pas les deux, commandent un dessert.
- Ahmed ou Choukri, ou les deux, commandent chaque jour un dessert.
- Si Choukri commande un dessert, Ahmed fait de même.

Exprimer les données du problème comme des formules propositionnelles

Exercice 4 :

Soit les propositions suivantes:

p : « l'enfant sait lire »
q : « l'enfant sait écrire »

Donner la traduction dans le langage courant des propositions suivantes :

- $p \wedge q$;
- $p \wedge (\neg q)$;
- $(q \rightarrow p)$;
- $(\neg p) \vee (\neg q)$;
- $(\neg p) \wedge (\neg q)$

Exercice 5:

Exprimez sous forme de formule syntaxiquement correcte les phrases suivantes:

a. Il existe des entiers qui ne sont pas pairs.
b. Tout entier est pair.
c. Un entier est soit pair, soit premier.
d. Le double d'un nombre premier est non premier.
e. Il existe un entier non nul supérieur ou égal à 1.
f. Le double du successeur d'un entier pair est premier.
g. Tout entier premier peut être majoré par un entier pair et par une autre entier premier.
h. Si un entier x est inférieur à l'entier y alors tout entier supérieur à y est aussi supérieur à x

Exercice 6:

Formalisez les affirmations suivantes dans la logique des prédicats du premier ordre. Pour chaque formule. Vous disposez des quatre prédicats suivants :

beau(j) qui est vrai s'il fait beau le jour j.
mer(x,j) qui est vrai si x va à la mer le jour j.
voisin(x,y) qui est vrai si x est voisin de y.
suivant(j1,j2) qui est vrai si j2 est le jour qui suit j1.

Vous disposez aussi de deux constantes moi et aujourd'hui.

a. Aujourd'hui il fait beau.
b. Les jours où il fait beau, je vais à la mer
c. Je ne vais jamais à la mer les jours où il ne fait pas beau

d. Au moins un de mes voisins ne va jamais à la mer le même jour que moi
e. Les jours où il fait beau, tous les voisins de mes voisins vont à la mer
f. Aucun de mes voisins ne va à la mer deux jours de suite.
g. Il ne fait jamais beau plus de deux jours de suite.

Exercice 7 :

Soit les données suivantes :

mostafa est un homme	lylia est une enfant
ilyes est un homme	samir aime tom&jerry
samir est un homme	mostafa et ilyes aiment sara
sara est un femme	ilyes aime lylia

Q1- traduisez les énoncés ci-dessus en faits avec prolog et enregistrez sous lenom ex1.pl

Q2- lancez SWI-prolog et chargez votre programme à travers le menu File→Consult

Q3- Testez votre programme. Exemple : femme(sara).

Q4- utilisez la requête : ? -**listing(homme)**. Que fait cette requête ?

Q5- Lancez la requête permettant de retourner toutes les personnes aimées par ilyes.

Q6- En utilisant le variable inconnue (_), lancez la requête permettant de testeruniquement si mostafa aime quelqu'un sans retourner ce dernier.

Q7- Ajoutez la règle suivante à la base de connaissances :

X est adorable s'il existe quelqu'un qui aime X, et X est une femme,homme, ou enfant.

Remarque : après chaque modification, rechargez votre programme avec File→Reload modified files

Q8- Lancez une requête qui permet de retourner toutes les personnes adorablessans répétition.

Q9- Ajoutez la règle suivante : X est jaloux de Y si X et Y aime la même femme. Utilisez le symbole (\=) (diffèrent).

Q10- utilisez : ? - **listing(jaloux).** Que fait listing dans cette requête?

Q11- affichez les hommes jaloux.

Exercice 8 :

Définissez la famille ci-dessous dans l'éditeur. Attention à ne pas mettre de majuscules aux prénoms ni aux prédicats. N'oubliez pas les points.

```
homme(ahmed).
homme(djamel).
homme(boumediene).
homme(bilal).
homme(ilyes).
homme(yassine).
femme(lila).
femme(chahra).
femme(sihem).
femme(meriem).
femme(sophia).
pere(ahmed,djamel).
pere(djamel,boumediene).
pere(boumediene,bertran).
pere(boumediene,sophia).
pere(djamel,sihem).
pere(ilyes,yassine).
mere(lila,djamel).
mere(chahra,sihem).
mere(chahra,boumediene).
mere(sihem,yassine).
mere(meriem,bilal).
mere(meriem,sophia).
```

A. Questions

Traduire les questions suivantes en Prolog et vérifier les réponses :

Q1. Est-ce que boumediene est un homme ?

Q2. Est-ce que yassine est une femme ?

Q3. Qui est une femme ?

Q4. Qui est un homme ?

Q5. Est-ce que meriem est la mère de sophia ?

Q6. Est-ce que meriem est la mère de yassine ?

Q7. Qui est la mère de djamel ?

B. Définition de prédicats

Définir les prédicats suivants :

- parent(X,Y) : X est un parent de Y, père ou mere ;
- fils(X,Y) : X est le fils de Y ;
- fille(X,Y) : X est la fille de Y ;
- grand_pere(X,Y) : X est le grand-père de Y ;
- grand_mere(X,Y) : X est la grand-mère de Y ;
- frere(X,Y) : X est le frère de Y ;
- soeur(X,Y) : X est la soeur de Y.

Q1. Quels sont les enfants de boumediene ?

Q2. Quels sont les hommes qui sont pères ?

Exercice 9 :

Adam aime les pommes.
Inès aime les carottes.
Karim aime les oranges.
Les pommes sont des fruits.
Les oranges sont des fruits.
Les carottes sont des légumes.

Ceux qui aiment les fruits sont en bonne santé.

Q1. Formalisez ces faits et règles en PROLOG.

Q2. Quelle est la requête pour savoir qui est en bonne santé ?

Q3. Quelle est la requête pour “Qui aime les pommes?” ?

Q4. Comment savoir les fruits que connaît le programme ?

Exercice 10 :

1- Traduire en Prolog l'énoncé suivant :

- Les étudiants qui ont la grippe doivent prendre un congé de maladie.
- Les étudiants qui ont de la fièvre et qui toussent ont la grippe.
- Ceux qui ont une température supérieure à 38° ont de la fièvre.
- Mohammed tousse et a une température supérieure à 38°.

2- La problématique à résoudre :

- Est-ce que Mohamed doit prendre un congé de maladie ?

Exercice 11: Résoudre le problème suivant par chaînage avant et arrière:

Base de règles :	Base de faits initiale :
• **R1 :** Si A et B alors C • **R2 :** Si F et D alors A • **R3 :** Si D et E alors B • **R4 :** Si B et D alors F • **R5 :** Si E et F alors D	• E • F

➢ On cherche à démontrer C

Exercice 12:

Dans le domaine de la botanique, considérons la base de règles suivante :

1. **Si** fleur **et** graine **alors** phanérograme
2. **Si** phanérograme **et** graine nue **alors** sapin
3. **Si** phanérograme **et** 1-cotylédone **alors** monocotylédone
4. **Si** phanérograme **et** 2-cotylédone **alors** dicotylédone
5. **Si** monocotylédone **et** rhizome **alors** muguet
6. **Si** dicotylédone **alors** anémone
7. **Si** monocotylédone **et non** rhizome **alors** lilas
8. **Si** feuille **et** fleur **alors** cryptogame
9. **Si** cryptogame **et non** racine **alors** mousse
10. **Si** cryptogame **et** racine **alors** fougère
11. **Si non** feuilles **et** plante **alors** thallophyte
12. **Si** thallophyte **et** chlorophylle **alors** algue
13. **Si** thallophyte **et non** chlorophyle **alors** champignon
14. **Si non** feuille **et non** fleur **et non** plante **alors** colibacille

Déterminez une plante ayant les caractéristiques suivantes :
{rhizome, fleur, graine, 1-cotylédone}. Ce sera l**a base de faits initiale**.

Exercice 12: Soit le système expert "Ville"

Base de règles	Base des faits
• **R1 : Si** belle ville et très bon restaurants **alors** ville méritant le voyage. • **R2 : Si** ville historique **alors** ville méritant le voyage. • **R3 : Si** autochtones accueillants et traditions folkloriques **alors** ville méritant le voyage. • **R4 : Si** monuments et végétation abondante **alors** belle ville. • **R5 : Si** tradition culinaire **alors** bons restaurants. • **R6 : Si** restaurants 3 étoiles **alors** très bons restaurants. • **R7 : Si** restaurants 3 toques **alors** très bons restaurants. • **R8 : Si** musées et ville ancienne **alors** ville historique. • **R9 : Si** Provence et bord de mer **alors** autochtones accueillants. • **R10 : Si** parcs verdoyants et avenues larges **alors** végétation abondante.	• Parcs verdoyants • Avenues larges • monuments • restaurants 3 toques • ville ancienne

- On cherche à vérifier si la ville est une ville qui mérite le voyage.
- Construire la solution sous forme d'un graphe.

Exercice 13:

base de connaissances	Base des faits
• **R1 :** Si B et D et E alors F • **R2 :** Si G et D alors A • **R3 :** Si C et F alors A • **R4 :** Si B alors X • **R5 :** Si D alors E • **R6 :** Si X et A alors H • **R7 :** Si C alors D • **R8 :** Si X et C alors A • **R9 :** Si X et B alors D	• B • C

Peut-on obtenir le fait H ? Construire la solution sous forme de graphe.

Exercice 14 :

Nous considérons un monde avec 4 pions (A,B,C,D) non superposables. Ils peuvent être arrangés dans n'importe quel ordre, sauf A qui ne peut pas être plus à droite que D. Par exemple, **ABCD** et **CBAD** sont deux états possibles du monde, tandis que **DCBA** et **CDAB** ne sont pas possibles.

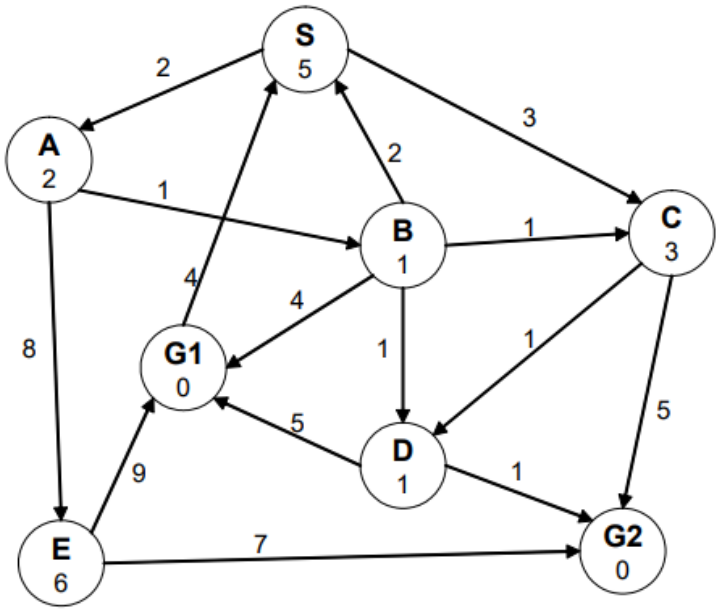
Le monde peut être manipulé par une action de la forme echange(x,y) qui échange les pions des positions x et y. Par exemple echange(1,2) transforme BCAD dans CBAD. Seules les actions echange(1,2), echange(2,3) et echange(2,4) sont autorisées. Ils donnent un successeur uniquement si la situation atteinte est possible.

1. Dessinez le graphe d'états.
2. On suppose que l'état de départ est **ADBC** et l'état que l'on veut atteindre est **CBAD**. On suppose que chaque action coûte 1.
   - ❖ Donnez une bonne heuristique **h** admissible (différente de 0 pour les nœuds non-finaux) pour ce problème. Le principe de l'heuristique devrait être suffisamment général pour pouvoir s'appliquer à des problèmes similaires.
3. Appliquez la recherche A∗ avec votre heuristique. Si vous n'avez pas trouvé d'heuristique, utilisez l'heuristique h = 0. Ne considérez pas les nœuds déjà développés. En cas d'égalité, choisissez un nœud à développer au hasard.

Exercice 15:

Dans l'espace de recherche suivant, l'état S est l'état de départ et les états G1 et G2 sont des états qui satisfont le test de but. Le nombre au dessus d'un arc représente le coût pour le parcourir. La valeur de la fonction heuristique h est inscrite dans le cercle. Pour chacune des méthodes de recherche suivantes : indiquez quel but est atteint et donnez la liste, dans l'ordre, de tous les états qui ont été choisis pour être explorés.

- ❖ Profondeur d'abord
- ❖ Largeur d'abord
- ❖ Algorithme glouton
- ❖ A*



Corrigé exercice 1

Exemple	Type de système	Justification
Un ascenseur qui monte ou descend selon l'étage demandé	**Classique**	Suit un programme fixe, réagit uniquement à la commande donnée.
Une voiture autonome qui s'adapte au trafic en temps réel	**Intelligent**	Utilise des capteurs et de l'IA pour analyser son environnement et s'adapter aux conditions de circulation.
Une machine à laver qui lance un cycle programmé	**Classique**	Fonctionne selon une séquence d'étapes prédéfinies, sans adaptation.
Un chatbot qui comprend des demandes en langage naturel	**Intelligent**	Capable de traiter le langage humain et d'adapter ses réponses grâce à l'IA.
Un feu de circulation avec minuterie fixe	**Classique**	Son fonctionnement est basé uniquement sur une minuterie, sans prise en compte du trafic réel.
Un système de recommandation sur YouTube	**Intelligent**	Analyse le comportement des utilisateurs et propose du contenu personnalisé grâce à l'apprentissage automatique.

Corrigé exercice 2 :

Nous utilisons les propositions suivantes :

A – Ahmed dit la vérité

B – Bachir dit la vérité

C – Choukri dit la vérité

Les deux affirmations peuvent être exprimées comme B → A et ¬C → ¬B.

Nous construisions une table de vérité en remarquant qu'il suiffera de considérer que trois lignes (comme un seul garçon ment)

A	B	C	$B \rightarrow A$	$\neg C \rightarrow \neg B$
1	1	0	1	0
1	0	1	1	1
0	1	1	0	1

On constate que uniquement l'interprétation A = 1, B = 0, C = 1 satisfait des formules. Par conséquent, le menteur c'est Bachir.

Corrigé exercice 3:

On introduit des variables propositionnelles a, b et c qui représentent le fait que:

Ahmed (a), Bachir (b) et Choukri (c) prennent un dessert. On traduit ainsi le problème :

- $a \Rightarrow b$
- $(b \wedge \neg c) \vee (\neg b \vee c)$
- $a \vee c$
- $c \Rightarrow a$

Corrigé exercice 4:

- L'enfant sait lire et écrire
- L'enfant sait lire mais il ne sait pas écrire
- Si l'enfant sait écrire alors il sait lire
- L'enfant ne sait pas lire ou il ne sait pas écrire
- L'enfant ne sait pas lire et il ne sait pas écrire

Corrigé exercice 5 : (je précise qu'il peut y avoir d'autres solutions)

a. $\exists x \quad entier(x) \wedge \neg\, pair(x)$
b. $\forall x \quad entier(x) \wedge \ pair(x)$
c. $entier(x) \wedge (pair(x) \vee premier(x))$
d. $(double(x) \wedge premier(x)) \Rightarrow \neg\, premier(x)$
e. $\exists x \quad entier(x) \wedge \neg\, nul(x) \wedge (x>=1)$
f. $(entier(x) \wedge \ pair(x) \wedge \ double(x+1)) \Rightarrow premier\ (y) \wedge (y=x+1)$
g. $\forall x \ (entier(x) \wedge \ premier(x)) \ \Rightarrow \wedge \ \exists y,z \ (majore(x,y) \wedge \ entier(y) \ \wedge$ $entier(z) \ \wedge pair(y) \ \wedge premier(z))$
h. $(entier(x) < entier(y)) \ \Rightarrow \forall z \ (entier(z) \wedge (z>y) \ \wedge (z>x))$

Corrigé exercice 6: (je précise qu'il peut y avoir d'autres solutions)

a. beau(aujourd'hui)
b. $beau(j) \Rightarrow mer(moi,j)$

c. $\neg beau(j) \Rightarrow \neg mer(moi,j)$
d. $\exists x\ voisin(x,moi) \wedge \neg mer(x,j) \wedge mer(moi,j)$
e. $\forall x,y\ voisin(x,moi) \wedge voisins(x,y) \wedge beau(j) \wedge mer(y,j)$
f. $\forall x, \neg voisin(x,moi) \wedge mer(x,j) \wedge suivant(j,j1)$
g. $\neg beau(j) \wedge suivant(j,j1)$

Corrigé exercice 7 :

Q3. ?- femmme(sarra).
Résultat : true.

Q4. ?- listing(homme).
Résultat : homme(mostafa).
homme(ilyes).
homme(samir).
Résultat : true.

- La requête ?- listing(homme).permet de trouver tous les hommes de la base de connaisances.

Q5. ?- listing(aime(ilyes,X)).
Résultat : aime(ilyes, sara).
aime(ilyes, norhane).
true.

Q6. ?- aime(mostafa,_).
Résultat : true.

Q7. adorable(X):-(aime(_,X),(femme(X);homme(X);enfant(X))).

Q8. setof(X,adorable(X),R).
Résultat : R = [norhane, sara].

Q9. jaloux(X,Y):-(aime(X,Z),aime(Y,Z),femme(Z),X\=Y).

Q10. ?-listing(jaloux).
jaloux(X, Y) :-
aime(X, Z),
aime(Y, Z),
femme(Z),
X\=Y.

- Cette requête permet de consulter la définition de la règle jaloux.

Q11. ?- jaloux(X, Y).
X = mostafa,
Y = ilyes

Corrigé exercice 8 :

Q1. ? - homme(Boumediene).
Résultat : : true.

Q2. ?- femme(yassine).
Résultat :false.

Q3. ? - listing(femme).
Résultat : femme(Lila).
femme(chahra).
femme(sihem).
femme(meriem).
femme(sophia).

Q4. ?- listing(homme).
Résultat : homme(Ahmed).
homme(Djamel).
homme(Boumediene).
homme(Bilal).
homme(Ilyes).
homme(yassine).
true

Q5. ?- mere(meriem,sophia).
Résultat : true

Q6. ?- mere(meriem,yassine).
Résultat : false

Q7.?- mere(X,Djamel).
Résultat :X = Lila.

C. Prédicats

```
/*X est un parent de Y, pere ou mere */
        parent(X,Y) :- mere(X,Y).
        parent(X,Y) :- pere(X,Y).
 /* X est le fils de Y */
```

```
                fils(X,Y) :- mere(Y,X),homme(X).
                fils(X,Y) :- pere(Y,X),homme(X).
/* X est la fille de Y */
                fille(X,Y) :- mere(Y,X),femme(X).
                fille(X,Y) :- pere(Y,X),femme(X).
/* X est le grand-pere de Y */
                grand-pere(X,Y) :- parent(P,Y), pere(X,P).
/* X est la grand-mere de Y */
                grand-mere(X,Y) :- parent(P,Y), mere(X,P).
/* X est le frere de Y */
                frere(X,Y) :- homme(X), pere(P,X), pere(P,Y), mere(M,X),
                mere(M,Y), X \== Y.
/* X est la soeur de Y */
                soeur(X,Y) :- femme(X), pere(P,X), pere(P,Y), mere(M,X),
                mere(M,Y), X \== Y.
```

Q1. ?- listing(pere(Boumediene,X)).
Résultat : pere(Boumediene, Bilal).
pere(Boumediene, sophia).
true.

Q2. ?- homme(X),pere(X,_).
Résultat : X = Ahmed ;
X = Djamel ;
X = Djamel ;
X = Boumediene ;
X = Boumediene ;
X = Ilyes ;
false.
Remarque : il faut ajouter le ;

D. Partie supplémentaire

On peut tester les règles par ces requêtes

```
?- listing(fils(X,Y)).
fils(X, Y) :-
mere(Y, X),
homme(X).
fils(X, Y) :-
pere(Y, X),
homme(X).
true.

?- listing(parent(X,Y)).
parent(X, Y) :-
mere(X, Y).
parent(X, Y) :-
```

```
pere(X, Y).
true.

?- listing(fille(X,Y)).
fille(X, Y) :-
mere(Y, X),
femme(X).
fille(X, Y) :-
pere(Y, X),
femme(X).
true.

?- listing(frere(X,Y)).
frere(X, Y) :-
homme(X),
pere(P, X),
pere(P, Y),
mere(M, X),
mere(M, Y),
X\==Y.
true.

?- listing(soeur(X,Y)).
soeur(X, Y) :-
femme(X),
pere(P, X),
pere(P, Y),
mere(M, X),
mere(M, Y),
X\==Y.

?- setof(X,frere(X,Y),R).
Y = sihem,
R = [Boumediene] ;
Y = sophia,
R = [Bilal
```

Corrigé exercice 9 :

Q1.

```
aime(adam, pommes).
aime(ines, carottes).
aime(karim, oranges).
fruits(pommes).
```

fruits(oranges).
legumes(carottes).
bonnesante(X):-aime(X,Y), fruits(Y).

Q2. bonnesante(X). adam ; karim
Q3. aime(X, pommes). adam
Q4. fruits(X). pommes, oranges

Corrigé exercice 10 :

Q1.

conge(X) :-grippe(X).
fievre(X) :- temp(X,T),
sup(T,38).
grippe(X) :-fievre(X),tousse(X).
tousse(mohamed).
temp(mohamed, t).
sup(t,38).

Q2.

Requête : conge(X). réponse : mohamed

Corrigé exercice 11:

- Chainage avant

  - E, F + D avec **R5**
  - D, E, F + B avec **R3**
  - B, D, E, F + A avec **R2**
  - A, B, D, E, F + C avec **R1**

Base finale : A, B, D, E, F, C

Il y a **réussite**, le fait C a été inclus dans la base

- Chainage arrière

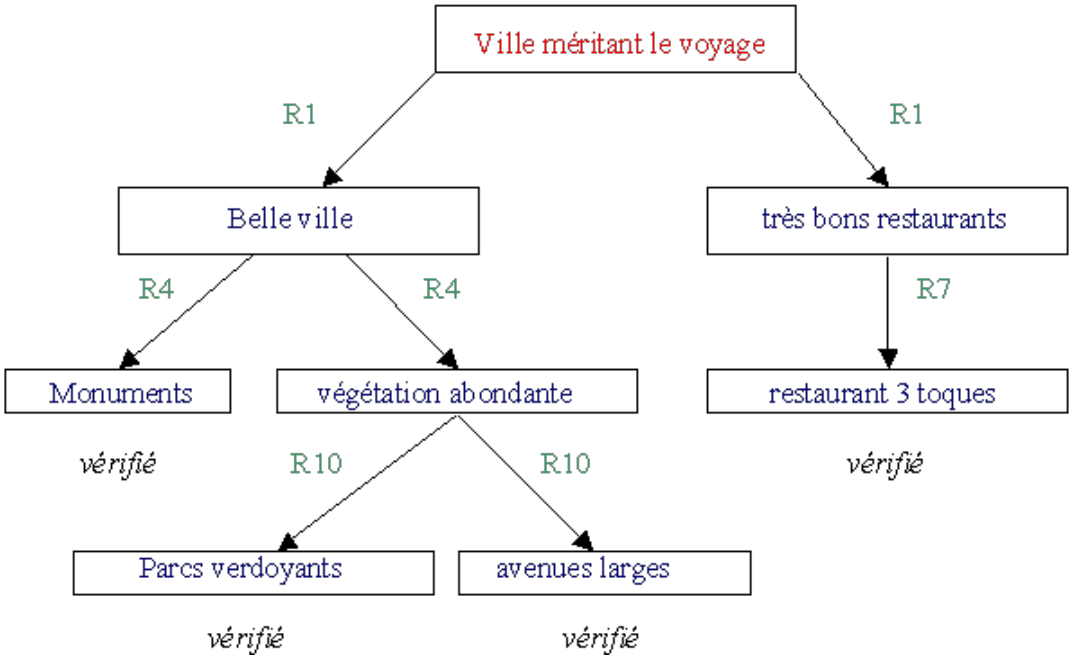
  - E, F, **A, B** avec **R1**

- E, F, **D, B** avec **R2**
- E, F, **D** avec **R3**

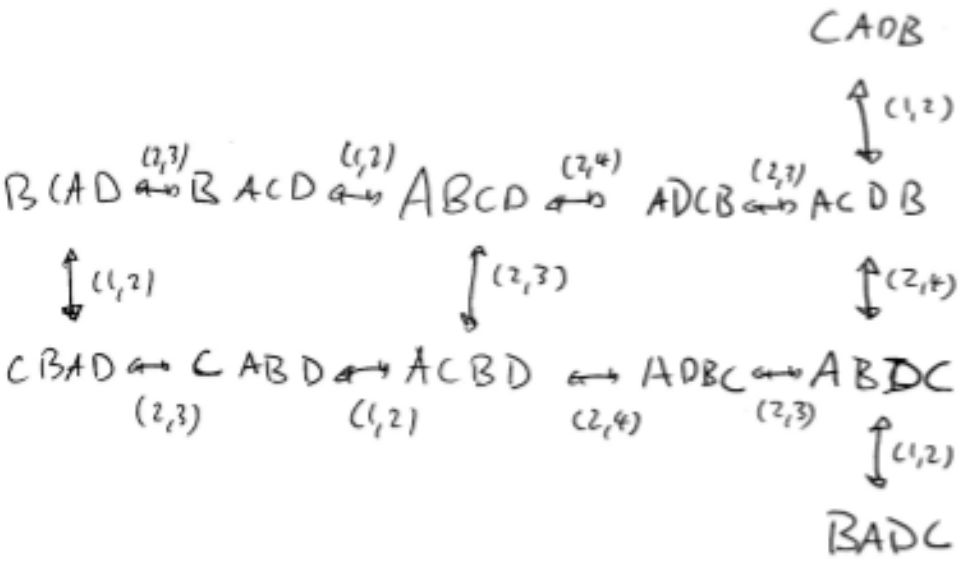
Base finale : E, F

Il y a **réussite**, il n'y a plus de faits à démontrer en base de fait

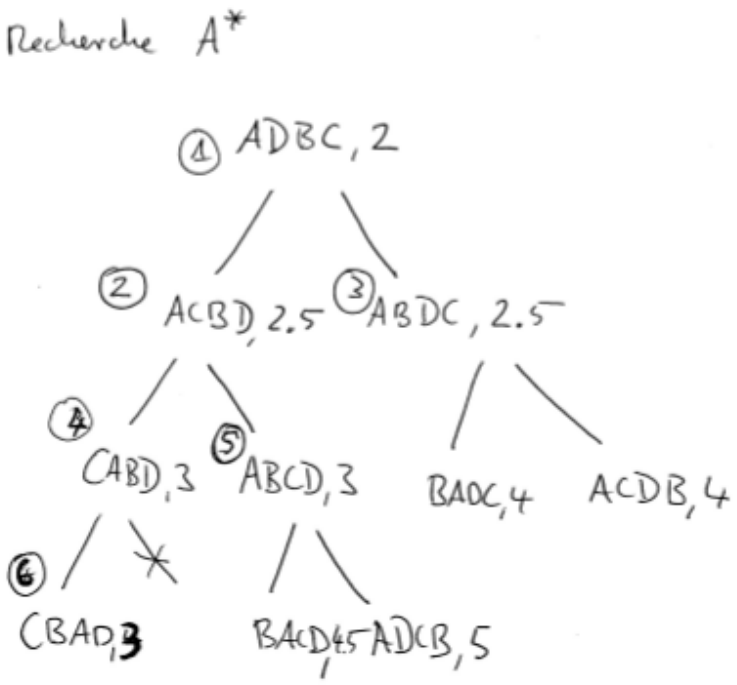
Corrigé exercice 13: Système expert Ville



Corrigé exercice 14:



- Heuristique proposée: le nombre de pions mal placés par rapport à CBAD



Corrigé exercice 15:

a) Profondeur d'abord: S,A,E,G2
1. Largeur d'abord: S,A,C,E,B,D,G2
2. Algorithme glouton: S,C, G2
3. A* S, A, B, D, G2

Liste des références

[1] A. M. Turing, “Computing Machinery and Intelligence,” *Mind*, 1950.

[2] J. McCarthy, M. Minsky, N. Rochester, and C. Shannon, “Proposal for the Dartmouth Summer Research Project on Artificial Intelligence,” 1956.

[3] R. Bellman, *An Introduction to Artificial Intelligence*. North-Holland, 1978.

[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2020.

[5] McCulloch, W.S. and Pitts, W.H. (1943) A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 5, 115-133.

[6] J. Giarratano and G. Riley, *Expert Systems: Principles and Programming*, 4th ed. Thomson, 2005.

[7] Université de Tlemcen, “Cours d’Intelligence Artificielle – Université de Tlemcen, Plateforme Moodle,” [En ligne]. Disponible sur : https://elearn.univ-tlemcen.dz/course/view.php?id=333. [Consulté: Juin. 2025].

[8] R. Barták, “Introduction to Constraint Programming,” [En ligne]. Disponible sur: https://www.kti.ms.mff.cuni.cz/~bartak/. [Consulté: Juin. 2025].

[9] P. C. L. Hui, éd., *Artificial Neural Networks - Recent Advances, New Perspectives and Applications*. IntechOpen, 2020.

[10] M. H. Landscovic, *Apprentissage automatique : Une Introduction Facile Au Machine Learning*. Auto-édition, 2024.

Annexe A

Outils et bibliothèques pour l'apprentissage automatique et le deep learning

1. Langages de programmation populaires :

- **Python** : Le plus utilisé car il est simple, lisible et a un écosystème très vaste (NumPy, pandas…)
- **R** : Fort en statistiques et visualisation et il est très utilisé en data science.
- **Java / C++** : Utilisés souvent pour des applications industrielles ou embarquées.

2. IDE et environnement interactifs :

- **Jupyter Notebook** : idéal pour exploration et prototypage.
- **Google Colab** : utilise Jupyter dans le cloud, aucun besoin d'installation en local.
- **VS code/PyCharm** : environnement de développement pour débogage et gestion de projets.

3. Bibliothèques et frameworks pour le machine learning

Le tableau A.1 décrit les bibliothèques utilisées en machine learning avec leur description.

Outil	Description	Utilisation typique
scikit-learn	Bibliothèque Python très accessible pour ML classique	Régression, classification, clustering
XGBoost / LightGBM	Algorithmes d'arbres très performants (gradient boosting)	Compétitions Kaggle, data science
CatBoost	Version optimisée d'arbres de décision pour données catégorielles	Big data tabulaire

Tableau A.1 : Bibliothèques pour le machine learning

Le tableau A.2 décrit les frameworks utilisés pour le deep learning avec leurs avantages.

Framework	Description	Points forts
TensorFlow	Développé par Google, très puissant	Support industriel, déploiement facile
Keras	Interface simplifiée pour TensorFlow	Facile à apprendre, très pédagogique
PyTorch	Développé par Meta, très utilisé en recherche	Flexibilité, dynamique, communauté forte
JAX	De Google, pour calcul différentiel rapide	Utilisé pour recherche avancée
MXNet	Par Amazon, supporte de nombreux langages	Bon pour cloud, mais moins populaire

Tableau A.2 : Frameworks pour le deep learning

Bien sûr, on doit obligatoirement parler des plateformes cloud pour entrainer et déployer les modèles de l'IA comme : Google Colab, AWS SageMaker, Microsoft Azure ML et Kaggle Kernels.

Annexe B

Syntaxe de Prolog

Un programme Prolog est constitué d'un ensemble de clauses ; une clause est une affirmation portant sur des atomes logiques ; un atome logique exprime unerelation entre des termes ; les termes sont les objets de l'univers.

A. Termes : toute variable ou constante est un terme.

1. Constantes : elles commencent par une lettre minuscule et peuvent êtreune chaîne de caractères ou un nombre (entier ou flottant) (ex. nabil version_2_3 '21 ans' -123 12.56).
2. Variables : elles commencent par une lettre majuscule (ex. Var, X, Var_longue_2) ou par « _ » (ex. _objet, _21). Une variable qui commence par « _ » est une variable anonyme (elle représente un objetdont on ne souhaite pas connaître la valeur).

Remarque : On peut avoir un terme composé (appelé aussi foncteur). Un terme composé avec la forme : foncteur(t_1, ..., t_n). Il commence par une minuscule,

et t_1, ..., t_n sont des termes (variables, constantes ou termes composés). Lenombre d'arguments n est appelé *arité du terme.*

Par exemple, adresse(18,"rue Meghili Mounir",Ville) est un terme composé defoncteur adresse et d'arité 3, dont les deux premiers arguments sont des constantes et le troisième argument est la variable Ville.

B. Prédicats (atomes) : expriment une relation entre les données.Exemple :

mère(farah, amine) fils(amine, farah)

C. Fait : est un prédicat qui termine par un « . ». Il exprime uneconnaissance acquise (connaissance qui est toujours vraie).

Exemple1 : etudiant(amine).

Exemple2 : habite(amine, tlemcen). inscrit(yassine, 'fac des sciences').

Un fait peut être défini avec des variables.Exemple : égale(X, X). plus_petit(0, X).

D. Règle : c'est une clause de Horn qui exprime la façon dont de nouvelles connaissances peuvent être déduites.

Exemple 1 : personne(X) :- homme(X).

personne(X) :- femme(X).

Exemple 2 : personne(X) :- (homme(X) ; femme(X)).

Ces deux règles ont la même signification qui se lisent ``pour tout X,personne(X) est vrai si homme(X) est vrai ou femme(X) est vrai"

Exemple 3 : mortel(X) : - humain(X).

Cette règle se lit : pour tout X, mortel(X) est vrai si humain(X).Exemple 4 :

frangin(X, Y) :- père(Z, X), père(Z, Y).

Cette règle se lit : pour tout X, pout tout Y, frangin(X, Y) est vrai s'il existe un Ztels que Z est le père de X et Z est le père de Y.

E. Requête (question ou but) :

Elle est composée par un ou plusieurs prédicats et qui se termine par un « . ».

?- prédicat_1(…), prédicat_2(…),…. .

1. Requête booléenne : qui ne contient pas de variables.

Exemples :

?- fils(yassine, med).

?- fils(yassine, med), frère(yassine, sara).

2. Requête existentielle : elle est définie avec une ou plusieurs variables.
   Exemple : Si on a dans la base de connaissances

fils(nabil,otman).

soeurs(fatima, nabil).

Requêtes :

?- fils(X, otman).

?- soeurs(X, nabil).

Requêtes utiles

?- listing.

permet d'afficher le contenu de la base de connaissance courante. Ce prédicat peut également prendre en argument le prédicat dont on souhaite connaître la définition courante. Par exemple : listing(femme).

Glossaire

Algorithme
Suite d'instructions logiques permettant de résoudre un problème ou d'exécuter une tâche.

Apprentissage automatique (Machine Learning)

Branche de l'intelligence artificielle qui permet aux systèmes d'apprendre à partir des données sans être programmés explicitement.

Apprentissage profond (Deep Learning)

Technique d'apprentissage utilisant des réseaux de neurones à plusieurs couches pour traiter de grandes quantités de données.

Automatisation
Utilisation de machines ou de programmes pour exécuter des tâches sans intervention humaine directe.

Base de données

Ensemble structuré d'informations stockées et organisées pour faciliter l'accès, la gestion et la mise à jour.

Big Data

Ensemble volumineux et complexe de données nécessitant des méthodes spécifiques pour le traitement et l'analyse.

Chatbot
Programme informatique capable de dialoguer avec un utilisateur en langage naturel.

Classification
Technique d'apprentissage supervisé consistant à attribuer une catégorie à des données.

Clustering
Méthode d'apprentissage non supervisé qui regroupe des données similaires en classes (ou clusters).

Données
Informations brutes collectées à partir d'observations, de capteurs ou d'interactions.

Explicabilité
Capacité d'un système intelligent à expliquer de manière compréhensible son raisonnement ou ses décisions.

Feu de circulation (classique)

Système fonctionnant selon une minuterie fixe, sans adaptation au trafic réel.

IA (Intelligence artificielle)

Discipline visant à développer des systèmes capables de simuler certaines facultés humaines comme la perception, le raisonnement et la prise de décision.

Intelligent (système)

Système capable de s'adapter, d'apprendre et de traiter des situations non prévues par simple programmation.

Itération
Répétition d'une opération ou d'un ensemble d'opérations dans un processus.

Jeu de données

Collection organisée de données utilisée pour entraîner ou tester un modèle d'IA.

Machine à laver (classique)

Exemple de système automatique suivant un cycle programmé, sans adaptation.

Modèle (en IA)

Représentation mathématique ou statistique utilisée par une machine pour faire des prédictions ou prendre des décisions.

Neurone artificiel

Unité de base des réseaux de neurones, inspirée du fonctionnement du cerveau humain.

Optimisation
Processus consistant à améliorer un système ou un modèle pour atteindre la meilleure performance possible.

Prédiction
Résultat produit par un modèle d'IA pour anticiper une valeur ou un événement futur à partir de données d'entrée.

Programme
Suite d'instructions codées permettant à un ordinateur d'exécuter une tâche.

Raisonnement
Capacité d'un système intelligent à établir des conclusions logiques à partir d'informations.

Réseau de neurones

Modèle d'IA composé de couches de neurones artificiels interconnectés, utilisé notamment en apprentissage profond.

Recommandation (système de)

Outil intelligent qui suggère du contenu personnalisé (ex. YouTube, Netflix) selon les préférences des utilisateurs.

Régression
Méthode statistique ou d'apprentissage supervisé visant à prédire une valeur numérique.

Robot autonome

Machine capable d'exécuter des tâches et de prendre des décisions sans contrôle humain direct.

Séquence
Suite ordonnée d'opérations ou d'étapes exécutées par un système.

Supervisé (apprentissage)

Type d'apprentissage automatique où les données utilisées pour l'entraînement comportent des exemples avec réponses attendues.

Système
Ensemble organisé d'éléments en interaction pour atteindre un objectif précis.

Système classique

Système déterministe dont le fonctionnement suit des règles ou séquences fixes, sans adaptation.

Traitement du langage naturel (NLP)

Branche de l'IA qui permet aux machines de comprendre et de générer du langage humain.

Voiture autonome

Exemple de système intelligent capable de percevoir son environnement et de s'adapter en temps réel au trafic.

QCM récapitulatif

1. **Quel algorithme de recherche utilise une heuristique pour guider la recherche ?**
   - **A.** Algorithme de recherche informée
   - **B.** Algorithme de recherche non informée
   - **C.** Algorithme de recherche en profondeur
   - **D.** Aucune réponse

2. **Qu'est-ce que le metaverse ?**
   - **A.** Un univers virtuel en 3D où les utilisateurs peuvent interagir
   - **B.** Un système expert utilisé pour prendre des décisions complexes
   - **C.** Un algorithme de recherche heuristique basé sur des règles
   - **D.** Aucune réponse

3. **La connaissance est :**
   - **A.** Une mesure brute
   - **B.** Une donnée interprétée
   - **C.** Une information comprise
   - **D.** Aucune réponse

4. **Qu'est-ce que le chaînage arrière ?**
   - **A.** Une technique pour résoudre des problèmes en partant du résultat souhaité
   - **B.** Une technique pour résoudre des problèmes en partant des données disponibles
   - **C.** Une technique pour résoudre des problèmes en partant de la réponse de l'utilisateur
   - **D.** Aucune réponse

5. **Quelle est la principale différence entre l'IA forte et l'IA faible ?**
   - **A.** A. L'IA forte a une compréhension réelle et consciente, l'IA faible simule l'intelligence sans conscience.
   - **B.** B. L'IA forte imite parfaitement l'intelligence humaine, l'IA faible ne fait que suivre des instructions programmées.
   - **C.** C. L'IA forte peut résoudre tous les problèmes humains, l'IA faible est limitée à certaines tâches.
   - **D.** D. Aucune réponse.

6. **Qu'est-ce que l'apprentissage par renforcement ?**
   - **A.** Un processus d'apprentissage où un agent cherche à minimiser une récompense.
   - **B.** Un processus d'apprentissage où un agent cherche à maximiser une pénalité.
   - **C.** Un processus d'apprentissage où un agent cherche à minimiser le temps de calcul.
   - **D.** Aucune réponse.

7. **Combien de couches cachées peut avoir un réseau de neurones ?**
   - **A.** Exactement 3 couches cachées.
   - **B.** Exactement 2 couches cachées.
   - **C.** Au maximum 2 couches cachées.
   - **D.** Aucune réponse.

8. **Dans l'apprentissage non supervisé :**
   - **A.** Les classes ne sont pas connues au préalable
   - **B.** Le processus se passe en deux phases (apprentissage et test)
   - **C.** Un expert est requis
   - **D.** Aucune réponse

9. **Les mesures de performances en apprentissage automatique sont :**
   - **A.** Exactitude et fidélité
   - **B.** Sensibilité et spécificité
   - **C.** Particularité et spécificité
   - **D.** Aucune réponse

10. **La réalité augmentée :**
   - **A.** Permet de mener une activité visuelle, sonore et tactile dans un monde créé numériquement
   - **B.** Désigne les différentes méthodes qui permettent d'incruster de façon réaliste des objets virtuels dans une séquence d'images.
   - **C.** Insère des images de synthèse sur les images du monde réel grâce à l'appareil photo d'un téléphone portable ou à des lunettes vidéo spéciales.
   - **D.** D. Aucune réponse

11. **Qu'est-ce qu'un domaine dans un CSP ?**

A. Un ensemble de valeurs possibles pour une variable
B. Une fonction qui évalue la satisfaction des contraintes
C. Une relation logique entre les variables
D. Aucune réponse

12. **En théorie des jeux, l'équilibre de Nash représente :**

A. Une situation où il y a égalité entre les joueurs
B. Une solution unique
C. Une situation où chacun adopte la meilleure réponse compte tenu du choix des autres
D. Aucune réponse

Complétez le tableau suivant

Français	English	عربي
		الأنظمة الخبيرة
	Data Warehouse	
	Machine learning	
Fouille de données		

Correction QCM

1. **Quel algorithme de recherche utilise une heuristique pour guider la recherche ?**
   - **E.** Algorithme de recherche informée
   - **F.** Algorithme de recherche non informée
   - **G.** Algorithme de recherche en profondeur
   - **H.** Aucune réponse

2. **Qu'est-ce que le metaverse ?**
   - **E.** Un univers virtuel en 3D où les utilisateurs peuvent interagir
   - **F.** Un système expert utilisé pour prendre des décisions complexes
   - **G.** Un algorithme de recherche heuristique basé sur des règles
   - **H.** Aucune réponse

3. **La connaissance est :**
   - **E.** Une mesure brute
   - **F.** Une donnée interprétée
   - **G.** Une information comprise
   - **H.** Aucune réponse

4. **Qu'est-ce que le chaînage arrière ?**
   - **E.** Une technique pour résoudre des problèmes en partant du résultat souhaité
   - **F.** Une technique pour résoudre des problèmes en partant des données disponibles
   - **G.** Une technique pour résoudre des problèmes en partant de la réponse de l'utilisateur
   - **H.** Aucune réponse

5. **Quelle est la principale différence entre l'IA forte et l'IA faible ?**
   - **E.** A. L'IA forte a une compréhension réelle et consciente, l'IA faible simule l'intelligence sans conscience.
   - **F.** B. L'IA forte imite parfaitement l'intelligence humaine, l'IA faible ne fait que suivre des instructions programmées.
   - **G.** C. L'IA forte peut résoudre tous les problèmes humains, l'IA faible est limitée à certaines tâches.
   - **H.** D. Aucune réponse.

6. **Qu'est-ce que l'apprentissage par renforcement ?**
   - **E.** Un processus d'apprentissage où un agent cherche à minimiser une récompense.
   - **F.** Un processus d'apprentissage où un agent cherche à maximiser une pénalité.
   - **G.** Un processus d'apprentissage où un agent cherche à minimiser le temps de calcul.
   - **H.** Aucune réponse.

7. **Combien de couches cachées peut avoir un réseau de neurones ?**
   - **E.** Exactement 3 couches cachées.
   - **F.** Exactement 2 couches cachées.
   - **G.** Au maximum 2 couches cachées.
   - **H.** Aucune réponse.

8. **Dans l'apprentissage non supervisé :**
   - **E.** Les classes ne sont pas connues au préalable
   - **F.** Le processus se passe en deux phases (apprentissage et test)
   - **G.** Un expert est requis
   - **H.** Aucune réponse

9. **Les mesures de performances en apprentissage automatique sont :**
   - **E.** Exactitude et fidélité
   - **F.** Sensibilité et spécificité
   - **G.** Particularité et spécificité
   - **H.** Aucune réponse

10. **La réalité augmentée :**
    - **E.** Permet de mener une activité visuelle, sonore et tactile dans un monde créé numériquement
    - **F.** Désigne les différentes méthodes qui permettent d'incruster de façon réaliste des objets virtuels dans une séquence d'images.
    - **G.** Insère des images de synthèse sur les images du monde réel grâce à l'appareil photo d'un téléphone portable ou à des lunettes vidéo spéciales.
    - **H.** D. Aucune réponse

11. **Qu'est-ce qu'un domaine dans un CSP ?**
    - **E.** Un ensemble de valeurs possibles pour une variable
    - **F.** Une fonction qui évalue la satisfaction des contraintes
    - **G.** Une relation logique entre les variables
    - **H.** Aucune réponse

12. **En théorie des jeux, l'équilibre de Nash représente :**
    - **E.** Une situation où il y a égalité entre les joueurs
    - **F.** Une solution unique
    - **G.** Une situation où chacun adopte la meilleure réponse compte tenu du choix des autres
    - **H.** Aucune réponse

Français	English	عربي
Systèmes experts	Expert Systems	الأنظمة الخبيرة
Entrepôt de données	Data Warehouse	مستودع البيانات
Apprentissage automatique	Machine learning	تعلم الآلة
Fouille de données	Data mining	التنقيب في البيانات