الجمهورية الجزائرية
الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و
البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة أبي بكر
بلقايد– تلمسان –
Université Aboubakr Belkaïd– Tlemcen –

Polycopié de cours

Apprentissage Artificiel

En : Génie Biomédical

Spécialité : Informatique Biomédicale

Niveau : 3ème Année Licence

Module : Apprentissage Artificiel (GB526)

Présenté Par : EL HABIB DAHO Mostafa

Etablissement de rattachement : Université de Tlemcen

Faculté : de Technologie

Département : Génie Biomédical

ANNEE UNIVERSITAIRE 2017 / 2018

Table des matières

Avant-propos

Au cours des 20 dernières années, l'apprentissage automatique statistique a connu une évolution considérable et est aujourd'hui une branche majeure de l'Intelligence Artificielle. Dans ce polycopié de cours, l'idée est de décrire les concepts fondamentaux et les avancées majeures de cette décennie. Il est illustré d'exemples applicatifs pris dans différents domaines et plus particulièrement en Biomédical.

Dans ce cours, il sera question d'initier les étudiants de troisième année en Informatique Biomédicale au monde de l'Intelligence Artificielle (IA) et plus particulièrement à l'Apprentissage Artificiel (AA) ou Machine Learning (ML) en anglais.

A travers ce cours d'Apprentissage Artificiel GB526, l'étudiant va pouvoir passer en revue : l'historique de l'IA, les types de l'apprentissage Artificiel ainsi que quelques méthodes utilisés pour l'amélioration de l'apprentissage. Plus important encore, ce cours va lui permettre de démarrer rapidement avec des applications pratiques et parfois amusantes d'algorithmes d'apprentissage artificiel. Le cours contient des pseudocodes et des exemples de données à utiliser en apprentissage. J'encourage également les étudiants à explorer leurs propres bases de données en utilisant des algorithmes d'apprentissage supervisé, non supervisé ou semi-supervisé.

Pour mettre en pratique les connaissances acquises dans cette matière, l'étudiant doit suivre les modules GB545 (TP Caractérisation des données) et GB546 (TP Techniques de classification) qui sont des travaux pratiques sur la programmation des techniques intelligentes.

Ce polycopié de cours est inspiré de plusieurs livres et manuels que vous trouverez dans la liste des références.

Objectifs de l'apprentissage

A l'issue du module, l'étudiant sera capable de :

- Comprendre les différents types d'apprentissage Artificiel (Supervisé, non supervisé, semi-supervisé et par renforcement).
- Construire des modèles intelligents pour la classification des données.
- Evaluer et améliorer l'apprentissage d'un système basique d'aide au diagnostic médical.

Descriptif et structure

Ce module se compose des trois chapitres suivants :

I. Introduction
- Généralités
- Historique

II. Les différents types d'apprentissage
- Apprentissage supervisé
- Apprentissage non supervisé
- Apprentissage semi supervisé
- Apprentissage par enforcement

III. Amélioration de l'apprentissage
- Les méthodes évolutionnistes (Algorithme Génétique, Particle Swarm Optimization)

Prérequis

Maitrise de l'algorithmique et des connaissances des bases en :
- Langage de programmation Matlab (vous n'avez PAS besoin d'être des experts en Matlab pour comprendre ce cours).
- Mathématiques,
- Probabilités et statistiques.

CHAPITRE I

Introduction à l'intelligence artificielle et l'apprentissage artificiel

Ce chapitre retrace l'évolution de l'intelligence artificielle au fils des années avec les définitions des concepts de base de l'IA.

1. Introduction

La science en général et l'informatique en particulier ont évolué au point où cela commence à coïncider avec la science-fiction.

Une machine peut-elle rêver ? OUI, selon Google, un ordinateur peut rêver, le géant de l'informatique a développé un algorithme de reconnaissance des images qui se base sur l'apprentissage profond (Deep Learning) et qui peut interpréter les images. Ce réseau de neurones artificielles est capable de générer lui-même des images à partir de bruit d'image (ou bruit numérique). Les figures 1 et 2, montrent quelques exemples des images générées par le logiciel de Google [1] :

Figure 1 : Image générée par le logiciel DeepDream

Figure 2 : Image générée par nous-même en utilisant le logiciel DeepDream[1], à gauche l'image originale.

[1] Vous pouvez tester ce logiciel sur le site : https://deepdreamgenerator.com/

Avant d'arriver à ce niveau avancé, l'intelligence artificielle est passée par plusieurs étapes qu'on citera dans ce chapitre. Pour cela nous allons tout d'abord commencer par des définitions et des généralités sur l'intelligence artificiel et l'apprentissage artificiel.

2. Généralités

2.1. C'est quoi l'intelligence artificielle ?

L'intelligence artificielle, souvent abrégée avec le sigle « IA », est définie par l'un de ses créateurs, Marvin Lee Minsky, comme : « *la construction de programmes informatiques qui s'adonnent à des tâches qui sont pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critiquée* ». [2]

Nous y trouvons donc le côté « artificiel » atteint par l'usage des ordinateurs ou de processus électroniques élaborés et le côté « intelligence » associé à son but d'imiter le comportement intelligent. Cette imitation peut se faire dans le raisonnement, par exemple dans les jeux ou la pratique de mathématiques, dans la compréhension des langues naturelles, dans la perception : visuelle (interprétation des images et des scènes), auditive (compréhension du langage parlé) ou par d'autres capteurs.

Le but de l'Intelligence Artificielle (IA) est de concevoir des systèmes capables de reproduire le comportement de l'humain dans ses activités de raisonnement. L'IA se fixe comme but la modélisation de l'intelligence prise comme phénomène (de même que la physique, la chimie ou la biologie qui ont pour but de modéliser d'autres phénomènes).

Le domaine de l'intelligence artificielle est influencé par plusieurs disciplines à savoir :

- Informatique, génie (**comment programmer et implanter l'IA ?**)
- Mathématiques (**statistique (limites théoriques de l'IA ?)**
- Neurosciences (**comment le cerveau fonctionne ?)**
- Psychologie cognitive (**comment l'humain réfléchit ?)**
- Économie, théorie de la décision (**comment prendre une décision rationnelle ?)**
- Linguistique (**quelle est la relation entre le langage et la pensée ?)**
- Philosophie (**quel est le lien entre le cerveau et l'esprit ?)**

Une machine ou un logiciel sont dit « intelligent » quand ils sont capables d'imiter un comportement ou une action intelligente, mais cette tâche ne peut pas être accomplie sans passer par un apprentissage artificiel ou automatique comme le fait d'ailleurs l'être humain.
Les questions qui se posent :

- Que veut dire l'apprentissage artificiel ou automatique ?
- Comment une machine peut-elle apprendre ?

Pour répondre à ces questions nous allons tout d'abord reprendre la notion d'apprentissage naturel.

2.2. Apprentissage naturel

Dès sa naissance, **un enfant** apprend à reconnaitre l'odeur de sa mère, sa voix et plus largement l'ambiance du lieu où il vit. Ensuite, il apprend à **coordonner** ses perceptions, comme sa vue ou son toucher, avec ses mouvements. Par des essais gratifiants ou pénalisants, il apprend plus tard à **marcher**, manifestant une grande capacité à **intégrer** des signaux différents : la vue, le sens de l'équilibre, la proprioception, la coordination motrice. Il apprend en même temps à **segmenter** et **catégoriser** des sons et à les associer à des significations. Il apprend aussi la structure de sa langue maternelle et acquiert simultanément un répertoire organisé de connaissances sur le monde qui l'entoure.

Les modalités de l'apprentissage naturel sont donc multiples : apprentissage par cœur, par instruction, par généralisation, par découverte, apprentissage impliquant des catégorisations voire la formation de théories, apprentissage plus ou moins supervisé ou autonome, etc.

Ces diverses formes d'apprentissage auront-elles une contrepartie lorsqu'il s'agira d'apprentissage par des **machines** ?

2.3. Apprentissage Artificiel

Pour résoudre un problème sur un ordinateur, nous avons besoin d'un algorithme. Un algorithme est une séquence d'instructions qui doit être effectuée pour transformer l'entrée en sortie. A titre d'exemple, on peut concevoir un algorithme de tri où l'entrée est un ensemble de nombres et la sortie est leur liste ordonnée.
Pour une même tâche, il peut y avoir plusieurs algorithmes et nous pouvons être amené à trouver le plus efficace, nécessitant le moins d'instructions et/ou de mémoire.
Pour certaines tâches, cependant, nous n'avons pas d'algorithme, par exemple, pour détecter la présence d'une pathologie chez un malade. Nous savons ce qu'est l'entrée : un ensemble de caractéristique ou de mesures biomédicales et nous savons ce que la sortie devrait être : une sortie oui / non indiquant si le sujet est malade ou pas. Mais nous ne savons pas comment transformer l'entrée en sortie, car, par exemple, ce qui peut être considéré comme une maladie change à travers le temps et d'un individu à un autre.

Ce qui nous manque dans la connaissance, nous le compensons dans les données. Nous pouvons facilement compiler des milliers d'exemples dont nous savons qu'ils sont des malades et ce que nous voulons, c'est « apprendre » ce qui les relie. En d'autres termes, nous aimerions que l'ordinateur (machine) extraie automatiquement l'algorithme pour cette tâche. Il n'y a pas besoin d'apprendre à trier les nombres, nous avons déjà des algorithmes pour cela ; mais il y a beaucoup d'applications pour lesquelles nous n'avons pas d'algorithme mais nous avons des exemples de données.

Avec les progrès de la technologie informatique, nous avons actuellement la capacité de stocker et de traiter de grandes quantités de données, ainsi que d'y accéder à partir d'emplacements physiquement éloignés sur un réseau informatique. La plupart des dispositifs d'acquisition de données sont maintenant numériques et enregistrent des données fiables. Pensez, par exemple, à une chaîne de supermarchés qui compte des

centaines de magasins dans tout le pays et vend des milliers de produits à des millions de clients. Les terminaux de point de vente enregistrent les détails de chaque transaction : la date, le code d'identification du client, les biens achetés et leur montant, l'argent total dépensé, etc. Cela représente généralement des giga octets de données chaque jour. Ce que veut la chaîne de supermarchés, c'est de pouvoir prédire quels sont les clients potentiels d'un produit. Encore une fois, l'algorithme pour cela n'est pas évident ; cela change dans le temps et selon l'emplacement géographique. Les données stockées ne deviennent utiles que lorsqu'elles sont analysées et transformées en informations que nous pouvons utiliser, par exemple, pour faire des prédictions.

Nous ne savons pas exactement quelles personnes sont susceptibles d'acheter cette saveur de crème glacée, ou le prochain livre de cet auteur, ou voir ce nouveau film, ou visiter cette ville, ou cliquez sur ce lien. Si nous le savions, nous n'aurions besoin d'aucune analyse des données ; nous irions de l'avant et écririons le code. Mais parce qu'on ne le sait pas, nous ne pouvons que recueillir des données et espérer extraire les réponses à ces questions et à d'autres similaires à partir des données. Nous croyons qu'il existe un processus qui explique les données que nous observons.

Bien que nous ne connaissions pas les détails du processus qui permet la génération de la connaissance (par exemple, le comportement du consommateur) nous savons que ce n'est pas complètement aléatoire. Les gens ne vont pas dans des supermarchés pour acheter des choses au hasard. Quand ils achètent des boissons gazeuses ils achètent du pain, de la glace en été et des légumes secs en hiver. Il y a certains modèles dans les données.
Nous ne sommes peut-être pas en mesure d'identifier le processus complètement, mais nous croyons que nous pouvons construire une approximation bonne et utile. Cette approximation peut ne pas tout expliquer, mais peut toujours être en mesure de rendre compte d'une partie des données.

Nous croyons que même si l'identification du processus complet peut ne pas être possible, nous pouvons toujours détecter certains modèles ou régularités. C'est le créneau de l'apprentissage automatique.

L'application de méthodes d'apprentissage automatique à de grandes bases de données s'appelle l'exploration de données. L'analogie est qu'un grand volume de terre et de matière première est extrait d'une mine qui, une fois traitée, conduit à une petite quantité de matière très précieuse ; De même, dans l'exploration de données, un grand volume de données est traité pour construire un modèle simple avec une utilisation précieuse, par exemple, ayant une grande précision prédictive.

Ses domaines d'application sont nombreux : en plus de la vente au détail, les banques financières analysent leurs données passées pour construire des modèles à utiliser dans les applications de crédit, la détection de la fraude et le marché boursier. Dans la fabrication, les modèles d'apprentissage sont utilisés pour l'optimisation, le contrôle et le dépannage. En médecine, les programmes d'apprentissage sont utilisés pour le diagnostic médical et le traitement de signal. Dans les télécommunications, les schémas d'appels sont analysés pour

l'optimisation du réseau et la maximisation de la qualité de service. En science, de grandes quantités de données en physique, en astronomie et en biologie ne peuvent être analysées assez rapidement que par les ordinateurs. Le World Wide Web est énorme ; il ne cesse de croître et la recherche d'informations pertinentes ne peut pas être effectuée manuellement.

Cependant, l'apprentissage automatique n'est pas seulement un problème de base de données ; c'est aussi une partie de l'intelligence artificielle. Pour être intelligent, un système qui évolue dans un environnement doit avoir la capacité d'apprendre. Si le système peut apprendre et s'adapter à de tels changements, le concepteur du système n'a pas besoin de prévoir et de fournir des solutions pour toutes les situations possibles. L'apprentissage automatique nous aide également à trouver des solutions à de nombreux problèmes de vision, de reconnaissance vocale et de robotique. Prenons l'exemple de la reconnaissance des visages : c'est une tâche que nous faisons sans effort. Chaque jour, nous reconnaissons les membres de la famille et les amis en regardant leurs visages ou leurs photos, malgré les différences de posture, d'éclairage, de coiffure, etc. Mais nous le faisons inconsciemment et nous sommes incapables d'expliquer comment nous le faisons. Parce que nous ne sommes pas en mesure d'expliquer notre expertise, nous ne pouvons pas écrire le programme informatique. En même temps, nous savons qu'une image de visage n'est pas seulement une collection aléatoire de pixels ; un visage a une structure. C'est systématique, il y a les yeux, le nez, la bouche, qui sont situés à certains endroits du visage. Le visage de chaque personne est un modèle composé d'une combinaison particulière de ceux-ci. En analysant des images de visage échantillon d'une personne, un programme d'apprentissage capture le motif spécifique à cette personne, puis reconnaît en vérifiant ce motif dans une image donnée. Ceci est un exemple de reconnaissance de formes.

L'apprentissage automatique consiste à programmer des ordinateurs pour optimiser un critère de performance en utilisant des exemples de données ou des expériences passées. Nous avons un modèle défini jusqu'à certains paramètres, et l'apprentissage est l'exécution d'un programme informatique pour optimiser les paramètres du modèle en utilisant les données d'apprentissage ou l'expérience passée. Le modèle peut être prédictif pour faire des prédictions dans le futur, ou descriptif pour acquérir des connaissances à partir de données, ou les deux. L'apprentissage automatique utilise la théorie des statistiques pour construire des modèles mathématiques, car la tâche principale consiste à faire des inférences à partir d'un échantillon. Le rôle de l'informatique est double :

- Premièrement, en formation, nous avons besoin d'algorithmes efficaces pour résoudre le problème d'optimisation, ainsi que pour stocker et traiter l'énorme quantité de données que nous avons en général.
- Deuxièmement, une fois qu'un modèle a été appris, sa représentation et sa solution algorithmique pour l'inférence doivent également être efficaces. Dans certaines applications, l'efficacité de l'algorithme d'apprentissage ou d'inférence, à savoir sa complexité spatiale et temporelle, peut être aussi importante que sa précision prédictive.

Nous allons voir, par la suite quelques exemples d'applications plus en détail pour mieux comprendre les types et les utilisations de l'apprentissage automatique.

2.4. Les règles d'association

Dans le cas d'aide au diagnostic médical par exemple, une application de l'apprentissage automatique est l'analyse des variables (features), qui consiste à trouver des associations entre les caractéristiques du patient.

Pour trouver une règle d'association, nous souhaitons apprendre une probabilité conditionnelle de la forme P (Y | X) où Y est la caractéristique que nous aimerions conditionner sur X, qui est un facteur de la maladie ou l'ensemble des facteurs dont nous savons que le patient les a déjà. [3]

Exemple : nous calculons que P (Diabétique | Hypertension) = 0.7. Ensuite, nous pouvons définir la règle :

70% des patients qui sont atteint de l'hypertension souffrent également du Diabète.

2.5. Classification

Un crédit est une somme d'argent prêtée par une institution financière, par exemple une banque, pour être remboursée avec intérêt, généralement par versements échelonnés. Il est important que la banque puisse prédire à l'avance le risque associé à un prêt, c'est-à-dire la probabilité que le client fasse défaut et ne rembourse pas le montant total. Il s'agit à la fois de s'assurer que la banque réalisera un profit et de ne pas gêner un client avec un prêt au-dessus de sa capacité financière.

En notation de crédit, la banque calcule le risque compte tenu du montant du crédit et des informations sur le client. Les informations sur le client comprennent les données auxquelles nous avons accès et sont pertinentes pour le calcul de sa capacité financière - à savoir, le revenu, l'épargne, les garanties, la profession, l'âge, les antécédents financiers et ainsi de suite. La banque a un dossier de prêts passés contenant de telles données client et si le prêt a été remboursé ou non. A partir de ces données d'applications particulières, il s'agit de déduire une règle générale codant l'association entre les attributs d'un client et son risque. C'est-à-dire que le système d'apprentissage automatique adapte un modèle aux données antérieures pour pouvoir calculer le risque d'une nouvelle application et décide ensuite de l'accepter ou de la refuser en conséquence.

Ceci est un exemple d'un problème de classification où il existe deux classes : les clients à faible risque et les clients à haut risque. Les informations sur un client constituent l'entrée du classificateur dont la tâche est d'affecter l'entrée à l'une des deux classes.

Après une formation avec les données passées, une règle de classification apprise peut être de la forme :

SI revenue > ϑ_1 ET économies > ϑ_2 ALORS faible risque SINON risque élevé

Ceci est un exemple de discriminant ; c'est une fonction qui sépare les exemples de différentes classes. [3]

Avec une règle comme celle-ci, l'application principale est la prédiction : Une fois que nous avons une règle qui correspond aux données du passé, si le futur est similaire au passé, nous pouvons faire des prédictions correctes pour les nouvelles instances. Compte tenu d'une nouvelle application avec un certain revenu et des économies, nous pouvons facilement décider si elle est à faible risque ou à haut risque.

Pour le diagnostic médical, les entrées sont les informations pertinentes que nous avons sur le patient et les classes sont les maladies. Les entrées contiennent par exemple l'âge du patient, son sexe, ses antécédents médicaux et ses symptômes actuels. Certains tests peuvent ne pas avoir été appliqués au patient, et ces entrées seraient donc manquantes. Les tests prennent du temps, peuvent être coûteux et peuvent incommoder le patient, nous ne voulons donc pas les appliquer à moins que nous croyions qu'ils nous donneront des informations précieuses. Dans le cas d'un diagnostic médical, une mauvaise décision peut conduire à un traitement erroné ou inexistant, et en cas de doute, il est préférable que le classificateur rejette et reporte la décision à un expert humain.

2.6. Régression

Disons que nous voulons avoir un système qui peut prédire le taux d'injection de l'insuline pour les patients diabétiques qui portent des pompes à insuline. Les entrées sont les attributs du malade (taux de glycémie dans le sang, taux d'insuline, âge, ..) qui, selon l'expert, affectent le taux d'insuline à injecter. La sortie est la quantité d'insuline nécessaire pour équilibrer le taux de glycémie dans le sang. De tels problèmes où la sortie est un nombre sont des problèmes de régression. [3]
Soit X les attributs du malade et Y la quantité d'insuline à injecter. Une fois de plus, en examinant les injections passées, nous pouvons collecter des données d'apprentissage et le programme d'apprentissage automatique adapte une fonction à ces données pour apprendre Y en fonction de X.

Exemple d'une fonction de régression simple :

$$Y = W.X + W_0$$

La régression et la classification sont des problèmes d'apprentissage supervisé où il y a une entrée, X, une sortie, Y, et la tâche est d'apprendre le mappage de l'entrée à la sortie.

L'approche dans l'apprentissage automatique est que nous supposons un modèle défini jusqu'à un ensemble de paramètres :

$$Y = g(X \mid \theta)$$

où g (·) est le modèle et θ sont ses paramètres. Y est un nombre en régression et est un code de classe (par exemple, 0 ou 1) dans le cas d'une classification. g (·) est la fonction de régression ou de classification, c'est la fonction discriminante séparant les instances de classes différentes. Le programme d'apprentissage automatique optimise les paramètres, θ, de sorte que l'erreur d'approximation est minimisée, c'est-à-dire que nos estimations sont aussi proches que possible des valeurs correctes données dans l'ensemble d'apprentissage.

3. Historique

Nous avons vu plutôt que même si les machines ont un cœur de pierre, elles peuvent apprendre. En tant que sous-ensemble de l'intelligence artificielle (IA), les algorithmes d'apprentissage automatique permettent aux ordinateurs d'apprendre des données, et même de s'améliorer, sans être explicitement programmés. C'est ainsi que votre commande vocale **Siri** (assistance vocale de l'iPhone) communique avec vous, ou comment votre super parking (logiciel de stationnement automatique intégré dans les voiture intelligentes) se gare, ou, en partie, comment votre banque approuve votre demande de prêt.

Dans cette partie, nous proposons un voyage très rapide dans le temps pour passer en revue quelques moments de l'histoire de l'apprentissage automatique et de son évolution.

C'est dans les années 1940 que le premier système informatique à commande manuelle, ENIAC (Electronic Numerical Integrator And Computer), a été inventé [4]. A cette époque, le mot "ordinateur" était utilisé comme nom pour un humain avec des capacités de calcul numérique intensives, donc, ENIAC a été appelé une machine de calcul numérique. Dès le début, l'idée était de construire une machine capable d'imiter la pensée et l'apprentissage humains.

Dans les années 1950, nous voyons le premier programme de jeu informatique prétendant être en mesure de battre le champion du monde des dames. Ce programme [5] a aidé les joueurs de dames à améliorer leurs compétences. À peu près à la même époque, Frank Rosenblatt a inventé le Perceptron [6] qui était un classifieur très, très simple mais quand il a été combiné en grand nombre, dans un réseau, il est devenu un outil puissant. La puissance est relative à l'époque et à ce moment-là, c'était une véritable percée. Ensuite, nous voyons plusieurs années de stagnation du champ de réseau neuronal en raison de ses difficultés à résoudre certains problèmes.

Des techniques spécifiques à l'informatique ont été développées à partir des années 1980 :

Les réseaux de neurones [7] qui simulent l'architecture du cerveau humain (une nouvelle aire pour les réseaux de neurones artificiels), et La programmation logique inductive [8] qui fait « marcher à l'envers » le processus habituel de déduction.

Avant l'années 1990, l'IA s'est essentiellement focalisée sur les théories et techniques permettant la réalisation d'intelligences individuelles. Dans la nature, il existe toutefois une autre forme d'intelligence collective, comme les êtres multicellulaires simples, les colonies d'insectes sociaux, les sociétés humaines. Ces sources d'inspiration montrent qu'une forme d'intelligence supérieure peut résulter de l'activité corrélée d'entités plus simples. Dans les systèmes artificiels, ce champ porte le nom d'« IA distribuée» ou de « systèmes multi-agents ». [9]

Grâce aux statistiques, l'apprentissage automatique est devenu très célèbre dans les années 1990. L'intersection de l'informatique et des statistiques a donné naissance à des approches probabilistes en IA. Ceci a déplacé le champ plus loin vers des approches axées sur les données. Ayant des données à grande échelle disponibles, les scientifiques ont commencé à construire des systèmes intelligents capables d'analyser et d'apprendre à partir de grandes quantités de données. Le système Deep Blue [10] d'IBM a battu le champion du monde des échecs, Garry Kasparov.

Figure 3 : Deep Blue Vs Garry Kasparov

L'arrivée d'Internet a ouvert la voie au partage et à la communication de connaissances. A cette époque, beaucoup de questions étaient posés : Comment organiser et traiter ces gigantesques masses d'information ? Comment en extraire les connaissances pertinentes pour les problèmes posés ?

Les moteurs de recherche comme Google ont intégré dans leurs tâches des techniques avancées de recherche d'information (information retrieval) et d'intelligence artificielle (data mining).

Parmi les avancées technologiques de l'époque, en 1999, un agent artificiel et intelligent de la NASA est arrivé à piloter un satellite au-delà de la planète Mars pendant une journée entière sans aucune aide en provenance de la Terre.

Durant les années 2000, beaucoup de progrès ont était senti, surtout en ce qui concerne l'intégration de l'apprentissage artificiel sur des services web. Il existe de plus en plus de systèmes de recommandation de produits ou de services sur le Web : films, livres, cours, restaurants, voyages, trajets d'autobus ou de train. Ils sont tous basés sur des techniques d'IA comme le raisonnement à base de cas, le filtrage de contenu (content filtering) ou le filtrage collaboratif (collaborative filtering). Certains d'entre eux prennent en compte les données démographiques mais aussi les habitudes d'achat du client ainsi que le comportement de navigation sur le web (web usage mining).

Le système Captcha, développé à Carnegie Mellon University, s'occupe de différencier les humains des machines. Captcha génère des tests que seuls les humains peuvent passer, afin de combattre les spams et les actions malicieuses de certaines machines.

Figure 4 : modèles de système Captcha

L'équipe américaine de Sargur Srihari, directeur du Center of excellence for document analysis and recognition (CEDAR, State University of New York at Buffalo), a mis au point en 2002 un logiciel capable de distinguer avec 98% de certitude si deux documents ont été écrits par la même personne ou non. Les travaux ont été effectués sur un échantillon de 1500 personnes [11].

En robotique, Wakamaru, est un modèle qui a était développé en 2005 par Mitsubishi Heavy Industries, il est doté de la parole, est principalement conçu pour veiller sur les personnes âgées [12]. Destiné à s'insérer dans la vie familiale de tout un chacun, il aura pour mission de prévenir l'hôpital ou les services de santé en cas de besoin.

Figure 5 : le robot Wakamaru

La particularité de ces dernières années est l'invention des machines dont la puissance de traitement permet de traiter des données massives. Avec cette avancé technologique, des algorithmes très puissants et très gourmands en même temps ont revu le jour, le plus connu est le Deep Learning [13] qui est largement utilisé depuis l'année 2010 (utilisé par Google, Facebook, Amazon, Apple, Netflix,..).

L'apprentissage Profond (en anglais deep learning, deep structured learning, hierarchical learning) est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures de réseaux de neurones.

Une application du *deep learning* à la santé publique est le projet Horus de la société Eyra inventé en 2016 et commercialisé en 2017. Il s'agit d'un appareil portable utilisant la plate-forme NVidia Jetson [14], qui aide les mal-voyants ou les aveugles à s'orienter et à reconnaître des personnes ou des objets, en retranscrivant en audio une image captée par une caméra.

Figure 6 : Projet Horus

De nos jours, l'apprentissage artificiel est devenu un outil courant dans presque toutes les tâches qui nécessitent l'extraction d'informations à partir de grands ensembles de données. Nous sommes entourés d'une technologie basée sur l'apprentissage automatique : les moteurs de recherche apprennent à nous apporter les meilleurs résultats (tout en plaçant des annonces rentables), le logiciel anti-spam apprend à filtrer nos e-mails, et les transactions par carte de crédit sont sécurisées par un logiciel qui apprend à détecter les fraudes. Les appareils photo numériques apprennent à détecter les visages et les applications intelligentes d'assistance personnelle sur les téléphones intelligents apprennent à reconnaître les commandes vocales. Les voitures sont équipées de systèmes de prévention des accidents construits à l'aide d'algorithmes d'apprentissage automatique. Les maisons équipées de capteurs pour optimiser la consommation énergétique et assurer le confort et la sécurité des habitants. Les hôpitaux modernes avec les équipements derniers génération (IRM connecté avec système d'aide au diagnostic médical). L'apprentissage automatique est également largement utilisé dans des applications scientifiques telles que la bio-informatique, la médecine et l'astronomie.

Sans apprentissage, l'intelligence artificielle ne serait jamais arrivée à un tel point de développement car les outils d'apprentissage automatique visent à doter les programmes de la capacité « d'apprendre » et de « s'adapter ».

4. Conclusion

Dans ce chapitre nous avons vu les définitions de l'intelligence artificielle, l'apprentissage automatique ainsi qu'un bref historique de l'évolution de ce domaine.
Dans le chapitre suivant, nous allons détailler les différents types d'apprentissage artificiel.

CHAPITRE II

Les types d'apprentissage

Dans ce chapitre nous nous intéressons aux différents types d'apprentissage automatique sur des problèmes de classification, ainsi qu'aux méthodes d'évaluation de ces classifieurs.

1. Apprentissage supervisé

L'apprentissage est dit supervisé si les différentes étiquettes, ou classes, sont connues et assigné préalablement à chaque élément de la base de données.
Dans l'apprentissage supervisé, les données sont composées d'une collection de N objets (variables d'entrée) décrits par M caractéristiques (attributs). Ces caractéristiques fournissent des informations sur l'étiquette de chaque objet (variable de sortie ou cible). Lorsque la sortie est un nombre réel, la tâche d'apprentissage automatique est appelée régression alors que si la sortie est discrète (binaire ou catégorique), nous parlons de classification.
Un ensemble de données peut être défini comme une matrice (voir Figure 7), où nous notons X l'espace des vecteurs d'entrée de la dimension M. De même, Y désignera l'espace de sortie.

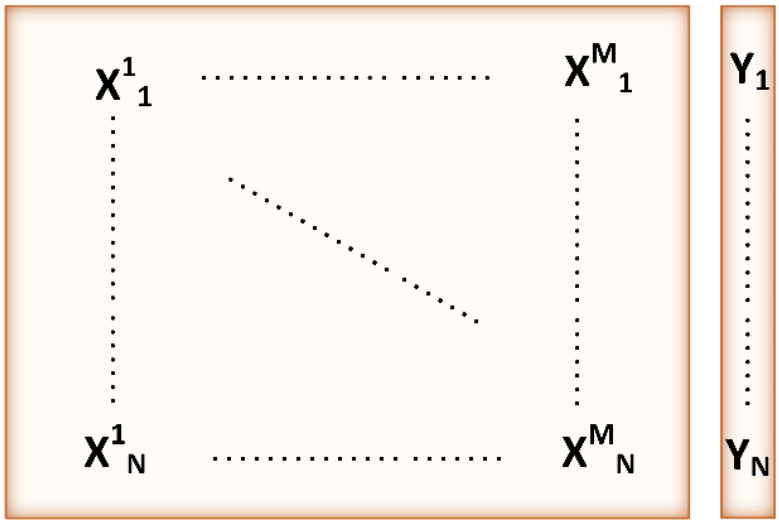


Figure 7 : représentation matricielle d'une base de données

Par exemple, nous considérons le problème bidimensionnel de la figure 8 comme un problème de classification médicale, où chaque point de cet espace peut représenter un patient décrit par deux variables (M1 et M2) et supposons que le but de l'apprentissage est de trouver une fonction qui sépare au mieux les données dans deux classes (malade ou en bonne santé). Ici, l'objectif de l'apprentissage supervisé est de fournir une règle qui aide un médecin à prédire la classe d'un nouveau patient.

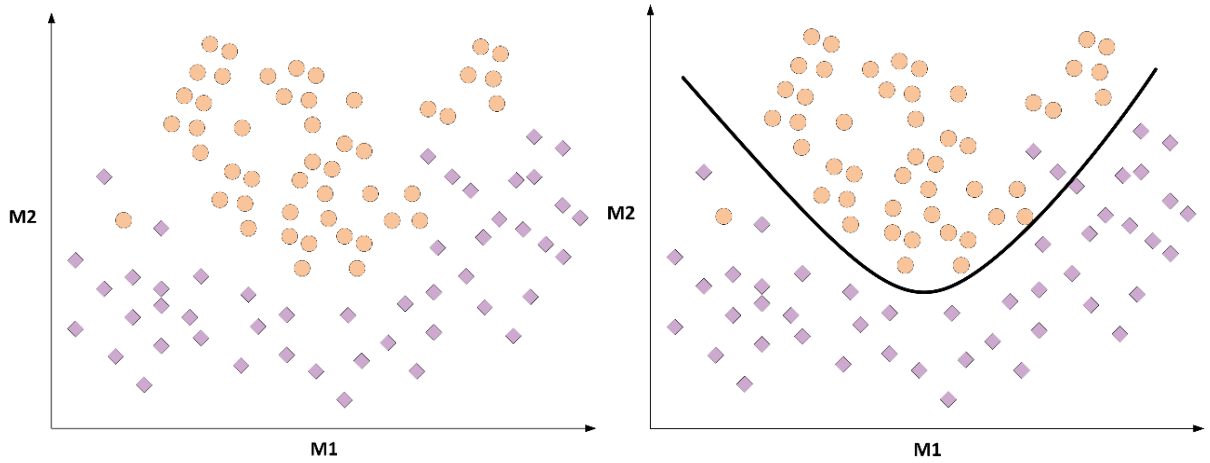


Figure 8: A gauche, un échantillon correspondant à un problème de classification simple. À droite, une séparation de ce problème

Un algorithme d'apprentissage peut être défini comme suit : C'est un algorithme qui prend en entrée un ensemble de données contenant les informations nécessaires pour caractériser un problème donné et renvoie un modèle qui représente les concepts caractérisant ces données et qui doit être capable de prédire l'étiquette de nouveaux objets en fonction de leurs valeurs d'entrée. Donc, pour avoir un modèle, nous devons passer par une phase d'apprentissage et pour le valider, une phase de test. [3]

1.1. Phase d'apprentissage

Dans cette phase (figure 9), un algorithme d'apprentissage automatique est entrainé sur une base de données étiquetées, au cours de cette étape, le modèle peut améliorer ses performances en se guidant avec une sous partie de la base d'apprentissage appelée base de validation. [3]

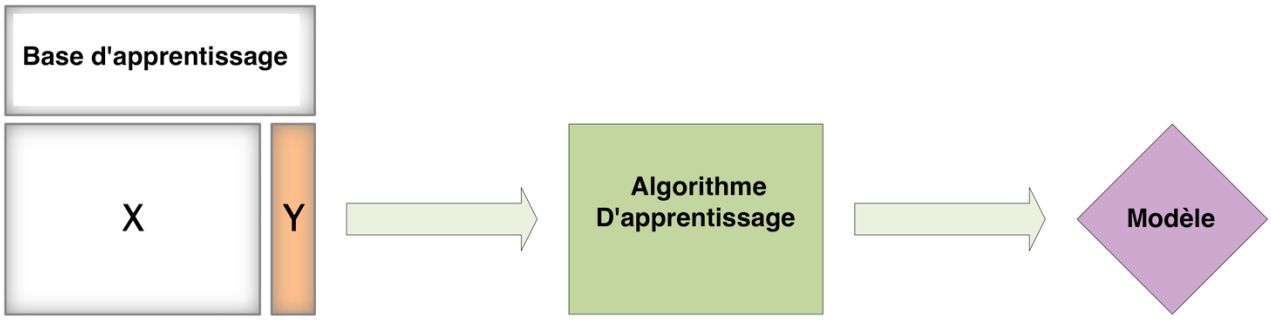


Figure 9: Apprentissage d'un modèle à partir d'un ensemble d'apprentissage

1.2. Phase de test

Dans les problèmes de classification, à l'étape du test, le modèle devrait être en mesure de donner une classe à une instance de test en fonction de sa valeur d'entrée (Figure 10) en suivant les règles apprises dans l'étape d'apprentissage. [3]

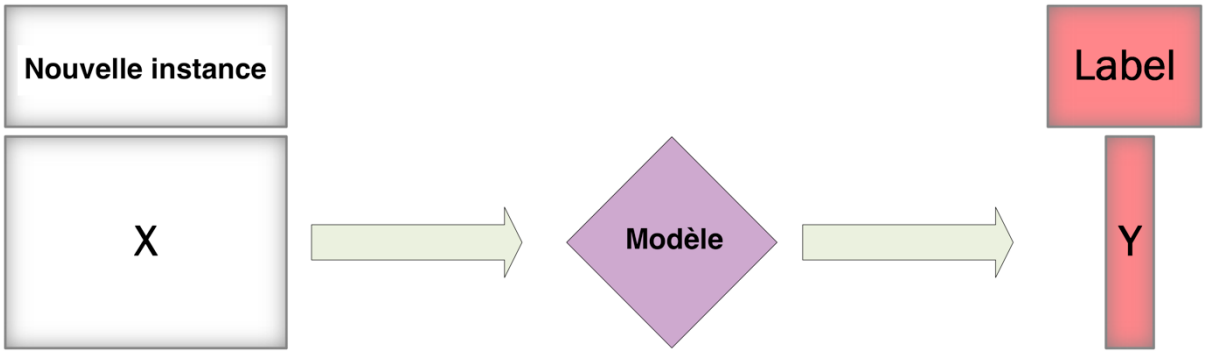


Figure 10: prédiction de la classe d'une nouvelle instance

Généralement, pour former et tester un modèle, les bases de données utilisées sont divisées en trois parties, la première pour l'apprentissage, la seconde pour la validation et la dernière pour tester. Cependant, nous pouvons également utiliser toute la base de données pour les phases d'apprentissage et de test. Cette technique est appelée validation croisée.

1.3. La validation croisée

La validation croisée (K-fold cross validation) est une méthode utilisée pour l'évaluation de la fiabilité des résultats obtenus par un classifieur donné. Cette méthode est basée sur un mécanisme d'échantillonnage.
Par exemple, dans un système d'aide au diagnostic, cette technique mesure le taux d'erreur d'un modèle de classification en utilisant toutes les données disponibles (base de données complète), à la fois en apprentissage et en test. La méthode de validation croisée est utile lorsque le nombre de données n'est pas vraiment important (petits jeux de données). Son principe de fonctionnement est résumé dans les points suivants :

- Les données disponibles sont divisées en K blocs disjoints (voir la figure 11, exemple de 5-fold cross validation)
- Le modèle de classification est entraîné en utilisant les données des blocs K-1 (LS pour Learning Set).
- Le test est effectué en utilisant le bloc de données restant (TS pour Test Set).
- L'apprentissage et le test sont répétés K fois (K expériences) puisque tous les blocs servent d'échantillons d'apprentissage et de test.
- Le taux d'erreur final du système est une moyenne des erreurs commises dans toutes les expériences K.

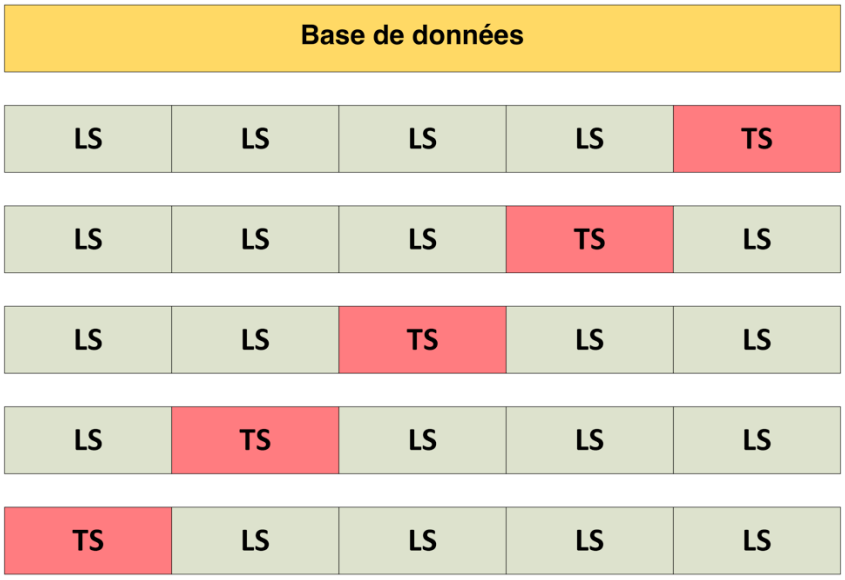


Figure 11: Exemple de validation croisée avec 5 blocs (5-fold cross validation)

Il existe de nombreuses applications qui sont le résultat d'un apprentissage automatique et qui permettent d'assister les cliniciens dans leurs procédures de diagnostic, car ils peuvent fournir un diagnostic plus précis et réduire les erreurs dues à la fatigue et aux doutes du médecin.

Ces techniques consistent à prédire la classe des nouvelles données observées en utilisant des modèles de classification comme les arbres de décision, les réseaux bayésiens, les réseaux de neurones, les k-plus proches voisins, etc …

Cependant, avant d'utiliser ces applications, le médecin doit être sûr de leurs fiabilités, pour cela il faudrait appliquer un certain nombre de tests et de méthodes d'évaluation des classifieurs.

1.4. Méthodes d'évaluation des classifieurs

L'évaluation de la performance d'un modèle de classification est une étape très importante car elle montre à quel point le classifieur est bon. Dans la littérature scientifique, nous trouvons que le plus grand critère utilisé par les chercheurs pour évaluer les classifieurs est le taux de bonne classification (les exemples correctement classés dans la phase de test) ou le taux d'erreur. Cependant, la seule utilisation de ce paramètre pour l'évaluation est très insuffisante. Dans la section suivante, nous présenterons un aperçu des différents critères d'évaluation que les étudiants doivent utiliser pour valider leurs classifieurs dans les séances des TPs (mesures scalaires et mesures graphiques telles que la courbe ROC (Receiver Operating Characteristic)).

1.4.1. Erreur de généralisation

Le taux d'erreur de généralisation (TE) ou de classification (TC) (TE = 100-TC) est la technique la plus utilisée pour évaluer les performances des classifieurs dans lesquelles les classes données par le classifieur sont comparées aux vraies étiquettes de l'ensemble de test (voir Figure 12). Le TC est un paramètre simple qui est calculé comme suit :

TC = Nombre d'échantillon bien classés / Nombre total d'exemples

Le TC est un paramètre relativement important pour l'évaluation d'un classifieur puisqu'il ne prend pas en compte la distribution des classes. Notant que la répartition des classes dans différentes bases de données médicales peut être déséquilibrée.

Si l'on considère par exemple une base de données avec 1000 patients dont seulement 10 sont malades, et que notre classifieur décide que tous les 1000 patients étaient normaux pendant la phase de test, la valeur du TC est égale à 99%.

Pour un classifieur, les taux d'erreur peuvent signifier que le système fonctionne très bien, ce qui est loin d'être le cas pour l'exemple ci-dessus.

Cela signifie que le TC n'est pas suffisant pour évaluer notre système. Ainsi, nous sommes obligés d'ajouter d'autres paramètres pour évaluer les algorithmes proposés. [3]

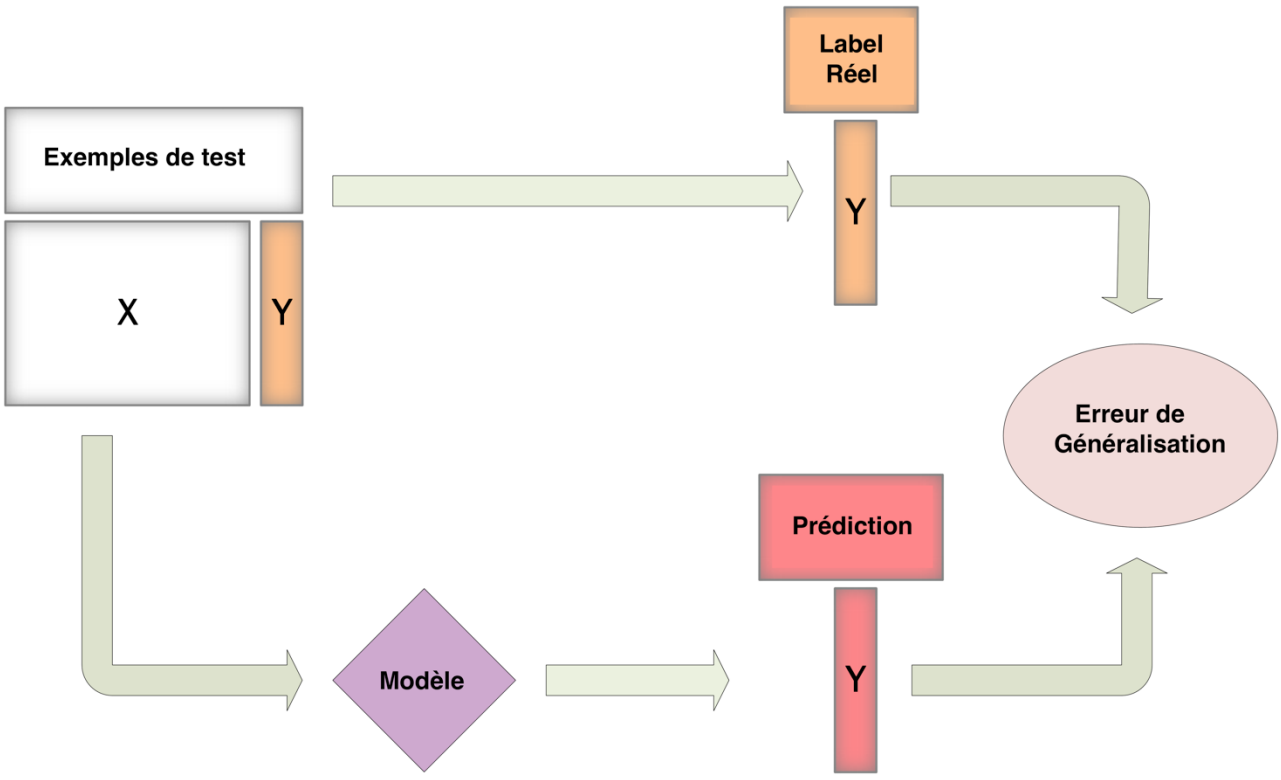


Figure 12: Évaluation de l'erreur de généralisation

1.4.2. Matrice de confusion

La matrice de confusion relie les décisions du classifieur et les étiquettes des échantillons. C'est un outil pour mesurer la qualité d'un système de classification. Comme le montre la figure 13, sur la diagonale de la matrice de confusion, nous trouvons les exemples bien classés, les autres sont mal classés. La matrice est un paramètre d'évaluation qui prend soin de la bonne classification et de la distribution des différentes classes.

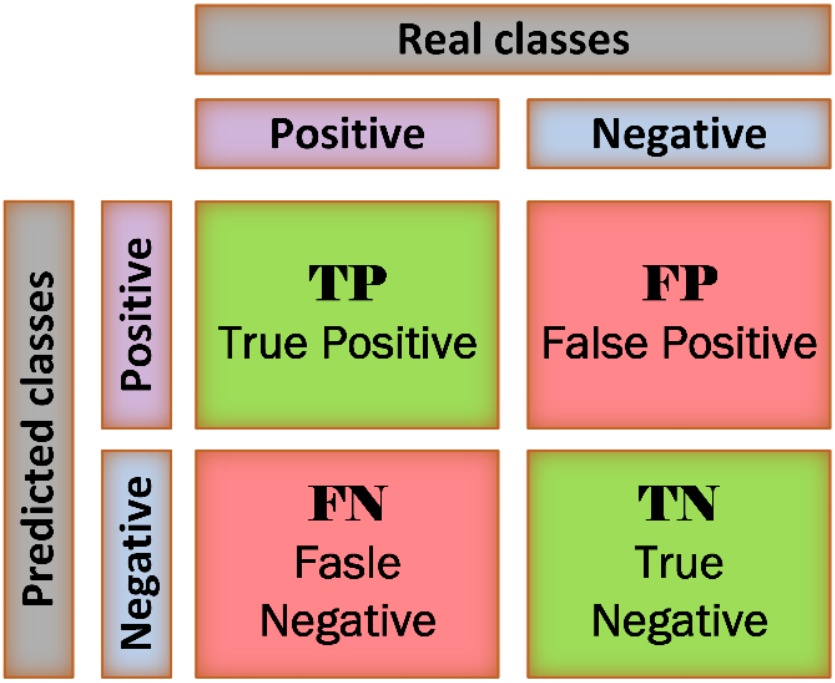


Figure 13: Matrice de confusion

Sachant que, par exemple, dans le problème de diagnostic médical :

- VP (TP en anglais) : représente le nombre de personnes malades classifiées malades.
- FP (FP en anglais) : représente le nombre d'individus en bonne santé classés comme malades.
- FN (FN en anglais) : représente le nombre d'individus malades classés comme patients en bonne santé.
- VN (TN en anglais) : représente le nombre d'individus en bonne santé classés en bonne santé.

Les matrices de confusion sont conçues pour donner plus de détails sur la classification des échantillons d'une classe donnée. A partir d'une matrice de confusion, nous pouvons calculer des mesures statistiques telles que la sensibilité et la spécificité. [3]

1.4.3. Sensibilité

La sensibilité (également appelée taux de vrai positif) d'un test de diagnostic quantifie sa capacité à identifier correctement les sujets atteints de la maladie (par exemple, le pourcentage de personnes malades qui sont correctement identifiées comme ayant la maladie). C'est la proportion de vrais positifs qui sont correctement identifiés par le test, donnée par [3] :

Sensibilité (Se) = Vrais positifs (VP) / (Vrais positifs (VP) + Faux négatives (FN))

1.4.4. Spécificité

La spécificité (parfois appelée le taux de vrai négatif) est la capacité d'un test à identifier correctement les sujets sans la maladie (par exemple le pourcentage de personnes en bonne santé qui sont correctement identifiées comme ne présentant pas la maladie). C'est la proportion de vrais négatifs qui sont correctement identifiés par le test :

Spécificité (Sp) = Vrais négatifs (VN) / (Faux positifs (FP) + Vrais négatives (VN))

Un prédicteur parfait serait décrit comme étant 100% sensible (c.-à-d. Prédisant que toutes les personnes du groupe malade qui sont malades) et 100% spécifique (c'est-à-dire ne prédisant pas quelqu'un du groupe sain comme étant malade) ; Cependant, théoriquement, tout prédicteur commettra une erreur minimale. [3]

1.4.5. Courbe ROC

ROC (Receiver Operating Characteristic) est une méthode de représentation graphique qui mesure les performances d'un classifieur binaire d'un côté et mesure la pertinence de ses différents descripteurs de l'autre côté. Cette méthode d'évaluation a été inventée pendant la seconde guerre mondiale pour déterminer une séparation de seuil entre le signal radar et le bruit. Depuis plusieurs années, son utilisation est devenue indispensable comme méthode d'évaluation des systèmes d'aide à la décision. Pour la représentation de la courbe ROC, plusieurs moyens (choix des axes de la courbe) basées sur la matrice de confusion sont possibles :

- Le taux de vrais positifs (VP) (Sensibilité) en ordonnée et le taux de faux positifs (FP) (1 - Spécificité) en abscisse.
- Le taux des vrais négatifs (VN) en ordonnée et le taux des vrais positives (VP) en abscisse.
- Le taux de faux positifs (FP) en ordonnée et faux négatif (FN) en abscisse.

Afin de déterminer la validité d'un test, le calcul de l'aire sous la courbe (Area Under the Curve (AUC)) est requis. La valeur AUC est utilisée pour évaluer le classifieur [15]. La figure 14 représente un exemple de l'espace ROC où :

- Le point a à (0,0) représente le seuil 1, en d'autres termes toutes les prédictions sont négatives.
- Le point b en (0,1) représente la situation idéale où le taux de vrai positif est maximal et le taux de faux positif est minimal.
- Le point c en (1,1) représente le seuil 0, toutes les prédictions sont positives.
- La ligne pointillée représente une courbe ROC pour les prédictions aléatoires, l'aire sous cette courbe est égale à 0, 5.
- La ligne rouge représente un exemple de courbe ROC.
- La zone verte ombrée représente l'aire sous la courbe ROC (AUC).

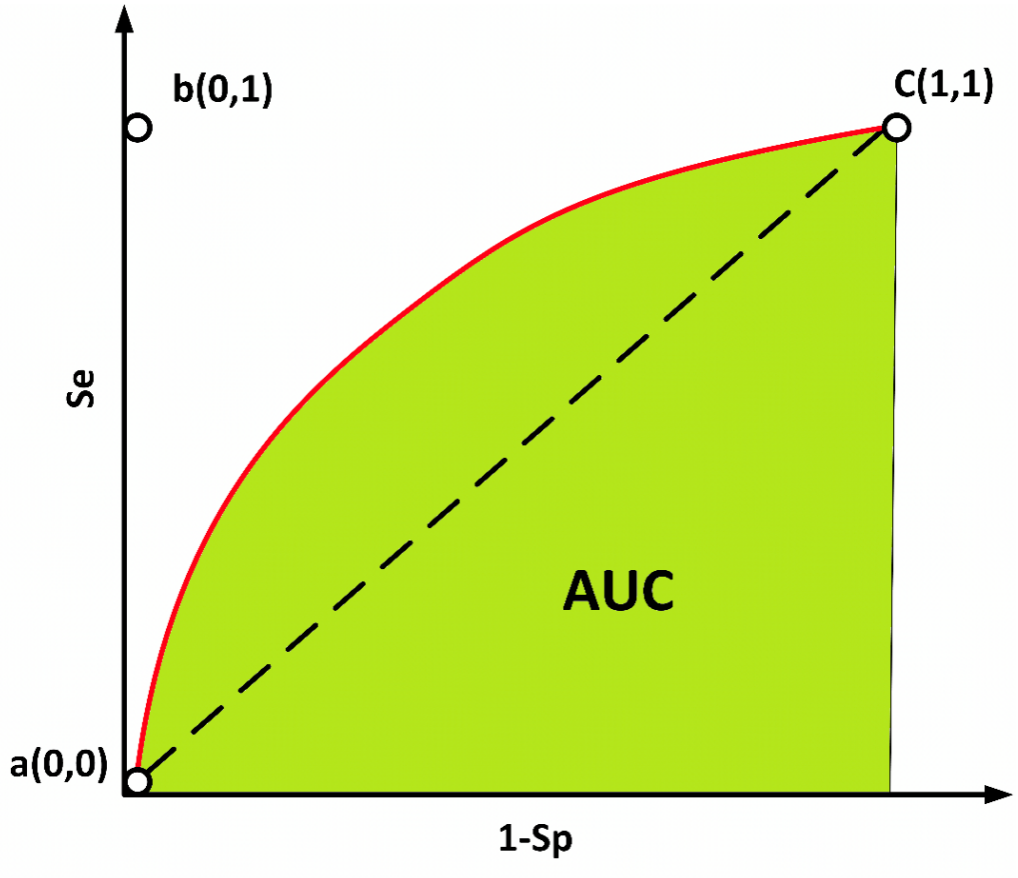


Figure 14: Courbe ROC

Après avoir présenté l'apprentissage supervisé avec son vocabulaire, les notations et les procédures standard utilisées pour évaluer les résultats des classifieurs, nous allons voir les techniques les plus utilisés de cette famille.

1.5. Les algorithmes de base

Dans cette partie du cours, nous examinerons principalement le problème de la classification supervisée. Dans ce qui suit, les principaux algorithmes de classification supervisée proposés dans la littérature seront présentés. Ce n'est pas une présentation exhaustive de toutes les méthodes mais seulement pour identifier les méthodes les plus conventionnelles qui peuvent être utilisées dans les séances de travaux pratiques en fonction de leurs propriétés spécifiques.

1.5.1. K-plus proches voisins

Dans la reconnaissance de formes, l'algorithme du k-plus proches voisins (k-Nearest Neighbors en anglais) (ou k-NN) est une méthode non paramétrique utilisée pour la classification supervisée et la régression [16]. Dans les deux cas, l'entrée consiste en les k exemples d'entraînement les plus proches dans l'espace des caractéristiques. k-NN est un type d'algorithme d'apprentissage basé sur des instances, où la fonction est seulement approximée localement et tout calcul est différé jusqu'à la classification. L'algorithme k-NN est l'un des plus simples algorithmes d'apprentissage automatique.

Le principe de l'algorithme k-NN est que les objets les plus proches sont plus susceptibles d'appartenir à la même catégorie. Ainsi, avec le KNN, les prévisions sont basées sur un ensemble de prototypes d'échantillons, qui sont utilisés pour prédire de nouvelles données, sur la base du vote majoritaire des K prototypes les plus proches. L'algorithme standard peut être résumé en deux étapes :

1. Déterminer (Vk (x)) l'ensemble des k voisins les plus proches de x
2. Sélectionnez la classe de x en utilisant un vote majoritaire dans (Vk (x)) (Figure 15)

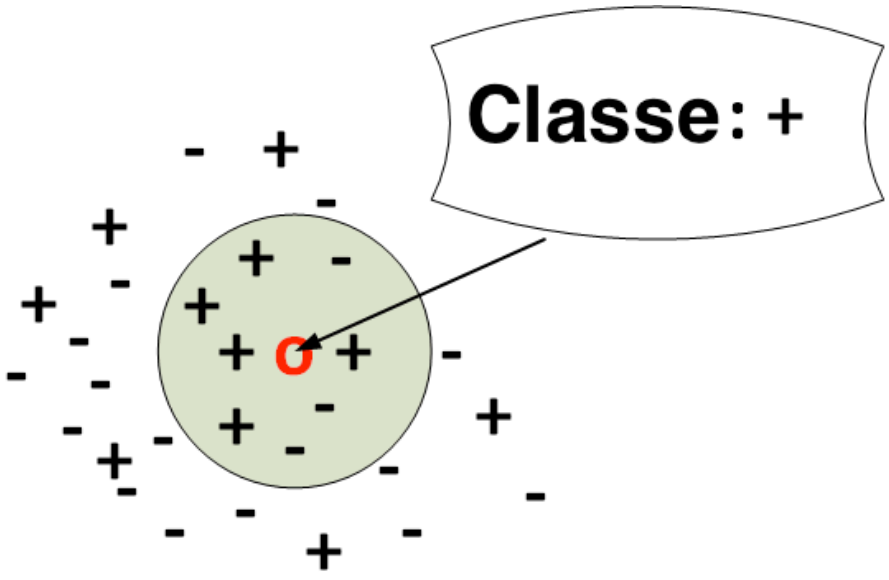


Figure 15: K-plus proches voisins

Les exemples d'apprentissage sont des vecteurs dans un espace de caractéristiques multidimensionnel, chacun avec une étiquette de classe. La phase d'apprentissage de l'algorithme consiste seulement à stocker les vecteurs de caractéristiques et les étiquettes de classe des échantillons d'apprentissage. Dans la phase de classification, un vecteur non étiqueté (une requête ou un point de test) est classé en affectant l'étiquette la plus fréquente parmi les k échantillons d'apprentissage les plus proches de ce point de requête (k est une constante définie par l'utilisateur). Une mesure de distance couramment utilisée pour les variables continues est la distance euclidienne. Pour les variables discrètes, telles que la classification de texte, une autre mesure peut être utilisée, telle que la mesure de recouvrement (ou distance de Hamming). Souvent, la précision de classification de k-NN peut être améliorée de manière significative si la métrique de distance est apprise avec des algorithmes spécialisés tels que Large Margin Nearest Neighbor ou Neighborhood components analysis. Le choix d'une valeur K appropriée est important pour les performances du classifieur K-Nearest Neighbor. Si la valeur K est trop grande, comme le montre la figure 16, le classifieur de plus proche voisin peut classer incorrectement l'instance de test car sa liste de voisins les plus proches peut inclure des points de données éloignés de son voisinage. D'autre part, si la valeur de K est trop petite, alors le classifieur du plus proche voisin peut être susceptible de souffrir du sur-apprentissage à cause du bruit dans l'ensemble de données d'entraînement.

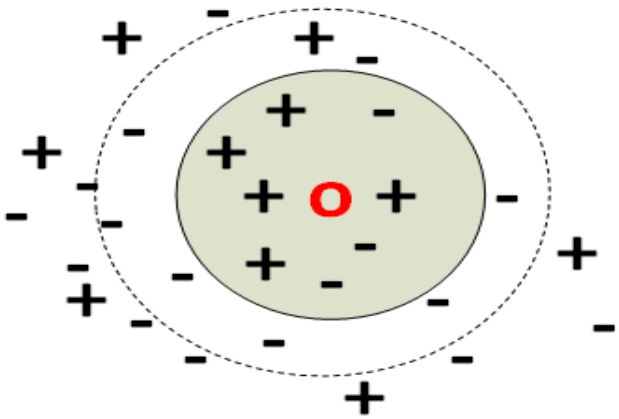


Figure 16: Choix du voisinage du K-plus proches voisins

Avantage :

- Apprentissage rapide
- Méthode facile à comprendre
- Adapté aux domaines où chaque classe est représentée par plusieurs prototypes et où les frontières sont irrégulières (ex. Reconnaissance de chiffre manuscrits ou d'images médicales)

Inconvénients :

- Prédiction lente car il faut revoir tous les exemples à chaque fois
- Méthode gourmande en espace mémoire
- Sensible aux attributs non pertinents et corrélés
- Particulièrement vulnérable au fléau de la dimensionnalité

1.5.2. Les arbres de décision

Un arbre de décision est une structure arborescente (voir Figure 17) dans laquelle chaque nœud interne représente un test sur un attribut, chaque branche représente le résultat du test et chaque nœud feuille est étiqueté avec une classe (décision prise après le calcul de tous les attributs). Un chemin de la racine à la feuille représente les règles de classification.

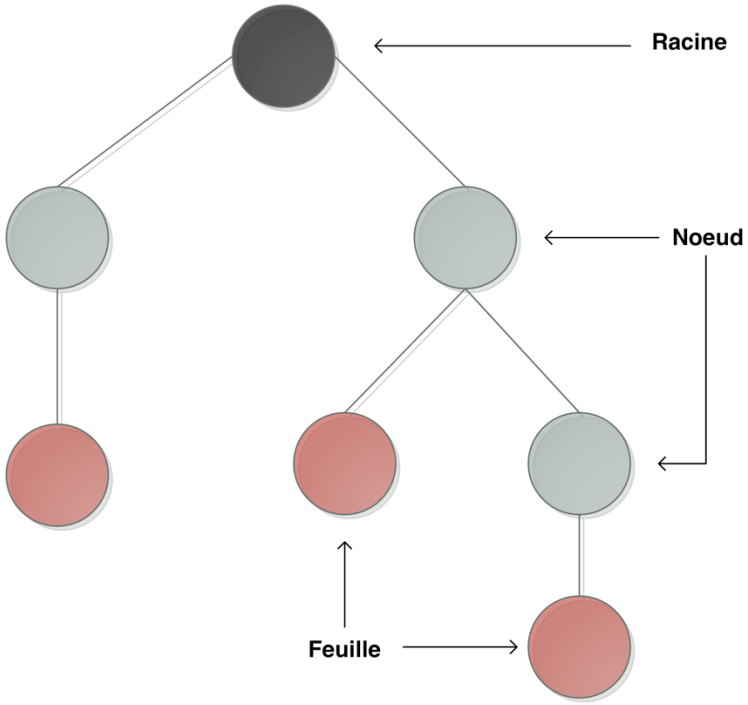


Figure 17: Arbre de décision

Dans l'analyse de décision, les arbres de décisions sont utilisés comme outil d'aide à la décision visuelle et analytique, où les valeurs attendues (ou l'utilité attendue) des alternatives concurrentes sont calculées. Un arbre de décision est une méthode que nous pouvons utiliser pour faire les bons choix, en particulier les décisions qui impliquent des coûts et des risques élevés. Les arbres de décision utilisent une approche graphique pour comparer des alternatives concurrentes et assigner des valeurs à ces alternatives en combinant les incertitudes, les coûts et les gains en valeurs numériques spécifiques. Les arbres de décision trouvent une utilisation dans un large éventail d'applications, ils sont utilisés dans de nombreuses disciplines différentes, y compris le diagnostic médical, les sciences cognitives, l'intelligence artificielle, la théorie des jeux, l'ingénierie et l'exploration de données.

L'apprentissage par arbre de décision est l'une des techniques les plus efficaces pour l'apprentissage supervisé de la classification. Pour cette partie du cours, supposons que toutes les entités ont des domaines discrets finis et qu'il existe une seule entité cible appelée classification. Chaque élément du domaine de la classification est appelé une classe. Un arbre de décision ou un arbre de classification est un arbre dans lequel chaque nœud interne (non-feuille) est étiqueté avec une entité en entrée. Les arcs provenant d'un nœud étiqueté avec une entité sont étiquetés avec chacune des valeurs possibles de l'entité. Chaque feuille de l'arbre est étiquetée avec une classe ou une distribution de probabilité sur les classes.

Un arbre peut être appris en divisant l'ensemble source en sous-ensembles basés sur un test de valeur d'attribut. Ce processus est répété sur chaque sous-ensemble dérivé d'une manière récursive appelée partitionnement récursif. La récursion est terminée lorsque les sous-ensembles de chaque nœud ont tous la même valeur de la variable cible (on l'appelle aussi feuille pure), ou lorsque la division n'ajoute plus de valeur aux prédictions. Ce processus d'induction descendante des arbres de décision est un exemple d'algorithme glouton, et c'est de loin la stratégie la plus courante pour l'apprentissage des arbres de décision à partir des données.

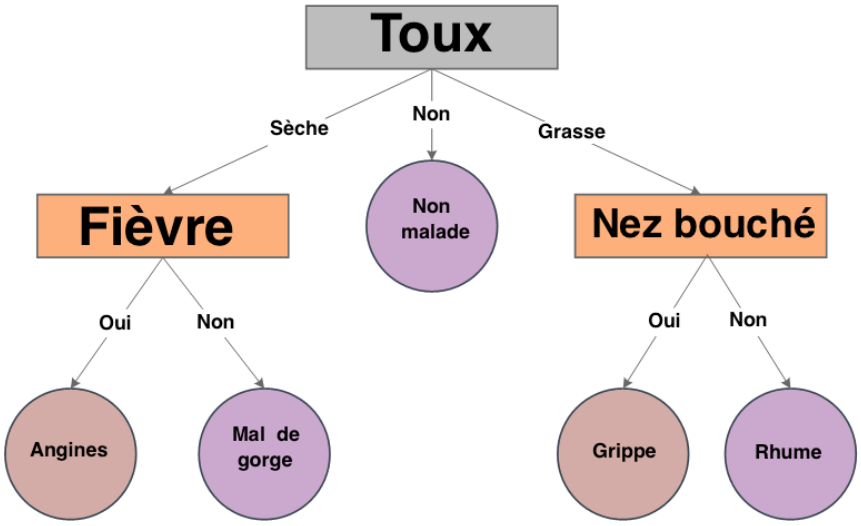


Figure 18: Exemple d'un arbre de décision

La figure 18 fournit un exemple d'arbre de décision pour une tâche de prédiction. Dans ce cas, les nœuds de test sont décorés de questions sur les maladies, tandis que les feuilles sont associées à des informations sur la variable cible de sortie Malade (ici par exemple : Angines, Mal de gorge, Grippe ou Rhume). Cette représentation graphique est souvent facile à comprendre et à interpréter par des experts humains.

Les arbres de décision utilisés dans l'exploration de données sont de deux types principaux :

- **Arbre de classification :** c'est quand le résultat prédit est la classe à laquelle appartiennent les données.
- **Arbre de régression :** c'est quand le résultat prévu peut être considéré comme un nombre réel (par exemple la durée de séjour d'un patient dans un hôpital).

Il existe de nombreux algorithmes spécifiques d'arbre de décision, les plus connus étant :

- l'arbre de classification et de régression (CART) qui a été introduit par Breiman en 1984 [17] (les arbres utilisés pour la régression et les arbres utilisés pour la classification ont des similitudes mais aussi des différences, comme la procédure de division).
- ID3 (Iterative Dichotomiser 3) [18]
- C4.5 (successeur de ID3) [19].

Les algorithmes de construction des arbres de décision fonctionnent généralement de haut en bas, en choisissant une variable à chaque étape qui scinde le mieux l'ensemble des éléments. Ce choix dépend des métriques (entropie logarithmique ou de Shannon, impureté de Gini, variance de la régression, test de Chi-2…) utilisées pour mesurer le « meilleur » score entre les attributs.

Les mesures de score que nous utiliserons dans ce cours sont basées sur l'impureté de Gini. Utilisé par l'algorithme CART, l'impureté de Gini est une mesure du désordre (ou d'inégalité de répartition), pour le choix d'un test dans défini une position de l'arbre. L'indice de Gini atteint son minimum (zéro) lorsque tous les cas dans le nœud tombent dans une seule catégorie cible.

$$\text{Gini} = 1 - \sum_{i=1}^{k} \mathrm{P_i}^2(\mathrm{i})$$

Où P_i est la fréquence relative de la classe i au nœud. Un nœud avec une seule classe (un nœud pur) a l'indice de Gini nul, sinon l'indice de Gini est positif.

Une des questions qui se pose dans un algorithme d'arbre de décision est la taille optimale de l'arbre final. Un petit arbre peut ne pas capturer les informations structurelles importantes sur l'espace échantillon. Un arbre trop grand risque de surcharger les données d'apprentissage et de mal généraliser aux nouveaux échantillons (sur-apprentissage). Le sur-apprentissage est une difficulté pratique importante pour les modèles d'arbre de décision et de nombreux autres modèles prédictifs. Le sur-apprentissage se produit lorsque l'algorithme d'apprentissage continue de développer des hypothèses qui réduisent l'erreur de l'ensemble d'apprentissage mais qui augmente l'erreur de l'ensemble de test (erreur de généralisation). Il y a plusieurs approches pour éviter le sur-apprentissage dans la construction des arbres de décision, la plus utiliser est l'élagage (pruning en anglais).

L'élagage devrait réduire la taille d'un arbre d'apprentissage sans réduire la précision prédictive mesurée par un ensemble de tests ou en utilisant la validation croisée. Il existe de nombreuses techniques pour l'élagage des arbres qui diffèrent dans la mesure utilisée pour optimiser les performances. On distingue deux manières de faire l'élagage dans les arbres de décisions :

- **Pré-élagage** : qui arrête de faire pousser l'arbre plus tôt, avant qu'il ne classe parfaitement l'ensemble d'entraînement.
- **Post-élagage** : qui permet à l'arbre de classer parfaitement l'ensemble d'entraînement, puis le tailler.

Pratiquement, la deuxième approche (post-élagage) est plus réussie car il n'est pas facile d'estimer précisément quand arrêter de faire pousser l'arbre. L'étape importante de l'élagage des arbres consiste à définir un critère permettant de déterminer la taille d'arbre finale correcte en utilisant l'une des méthodes suivantes :

- Utilisez un ensemble de validation pour évaluer l'effet des nœuds post-élagage sur les performances de l'arbre.
- Construisez l'arbre en utilisant l'ensemble d'apprentissage, puis appliquez un test statistique, tel l'estimation d'erreur ou Test de signification (par exemple, test du Chi-2), pour estimer si l'élagage ou l'expansion d'un nœud particulier est susceptible de produire une amélioration.
- Minimum Description Length: (principe de la longueur minimale de description) Utiliser une mesure explicite de la complexité pour coder l'ensemble d'apprentissage et l'arbre de décision, en arrêtant la croissance de l'arbre lorsque cette taille de codage est minimisé.

La première méthode est l'approche la plus courante. Dans cette approche, les données disponibles sont séparées en deux ensembles : un ensemble d'apprentissage, utilisé pour construire l'arbre de décision, et un ensemble de validation, utilisé pour évaluer l'impact de l'élagage sur l'arbre.

Avantages :

- Facilité à manipuler des données « symboliques »
- Multi-classe par nature
- Interprétabilité de l'arbre
- Peut-être utiliser pour Identification des variables importantes
- Classification très efficace (en particulier sur entrées de grande dimension)

Inconvénients :

- Sensibilité au bruit et points aberrants
- Stratégie d'élagage délicate

1.5.3. Les réseaux de neurones

Un réseau neuronal artificiel (RNA ou ANN : Artificial Neural Networks en anglais) est un modèle de calcul dont la conception est schématiquement inspirée du fonctionnement des neurones biologiques (du cerveau humain). Les réseaux neuronaux sont généralement optimisés par l'apprentissage des méthodes probabilistes, ils sont placés dans la famille des méthodes statistiques d'une part, et dans la famille des méthodes d'intelligence artificielle d'autre part car elles fournissent un mécanisme perceptuel des propres idées du programmeur.

En tant que tels, les RNA contiennent des « nœuds » interconnectés (neurones) capables de traiter et de transmettre des informations d'un nœud à un autre. Les RNA les plus basiques

décrits par McCulloch et Pitts en 1943 [20] (figure 19), transmettent par anticipation l'information dans une seule direction. Les propriétés du réseau et la transmission de l'information au sein du réseau sont régies par l'architecture et la façon dont ces interconnexions sont modélisées.

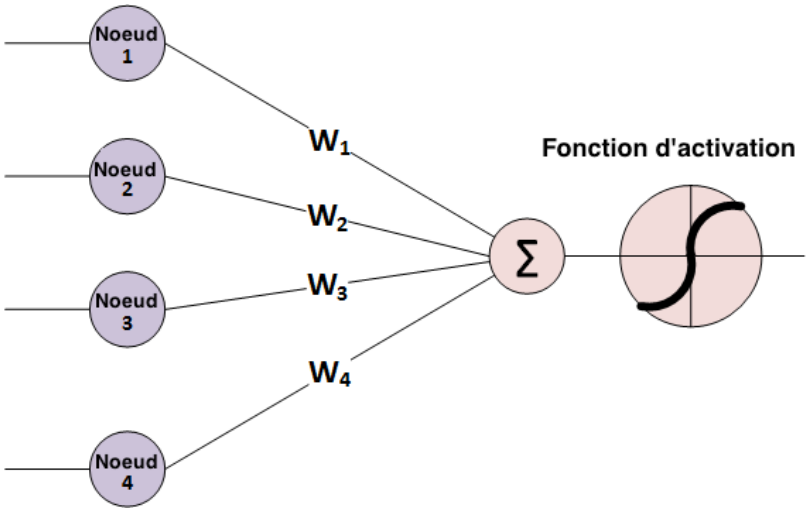


Figure 19: Perceptron

La rétro-propagation (back-propagation en anglais) est une technique basée sur l'apprentissage supervisé est utilisé pour l'apprentissage de réseaux de neurones artificiels. Cette technique a été décrite par P.Werbos en 1974 et développée plus tard par Rumelhart en 1986 [21]. Le principe du processus de rétro-propagation est itératif, l'ensemble d'échantillons d'apprentissage est injecté dans le réseau avec l'étiquette des classes réelles. Pour minimiser l'erreur quadratique moyenne entre la prédiction du réseau et la classe elle-même, les poids sont modifiés pour chaque échantillon d'apprentissage.

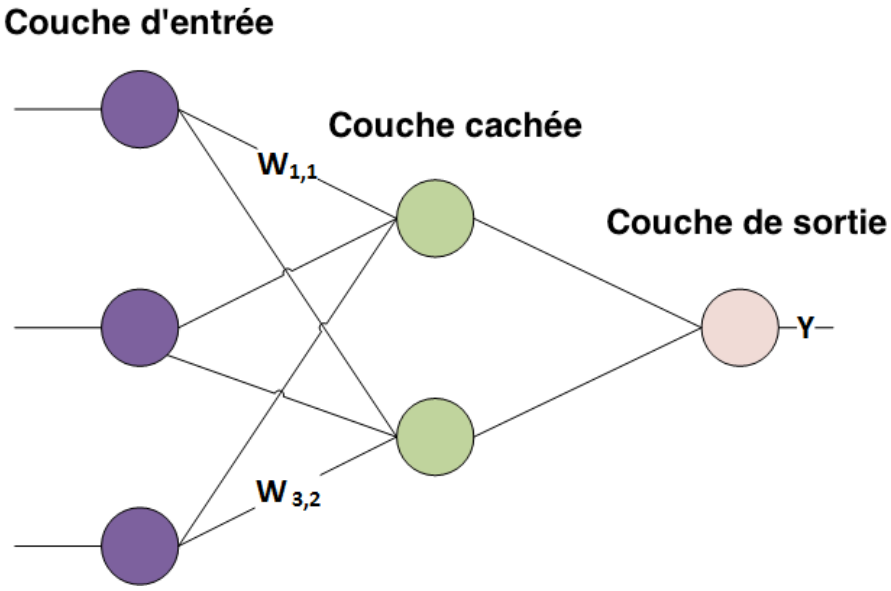


Figure 20: Perceptron multicouches

Bien que l'erreur soit minimisée localement, la technique peut converger au minimum et donner de bons résultats. Dans la plupart des cas, peu de problèmes dus aux minima locaux sont rencontrés. Cependant, il y a encore deux problèmes qui peuvent être rencontrés dans une application réelle, d'une part la lente convergence si le "taux d'apprentissage" n'est pas bien choisi et d'autre part le risque possible de converger vers un minimum local plutôt que vers la surface d'erreur globale.

Pour cela, plusieurs chercheurs ont tenté d'optimiser les RNA en les combinant avec d'autres techniques telles que les algorithmes génétiques ou l'essaim à particule. De plus, les réseaux de neurones sont considérés comme des systèmes de type « boîte noire » puisqu'ils ne sont pas interprétables. Plusieurs études ont tenté de résoudre ce problème en les combinant avec des méthodes telles que la logique floue, d'où les modèles neuro-flous (tel que Anfis et Nesclass).

Avantages :

- Capacité de représenter n'importe quelle fonction, linéaire ou pas, simple ou complexe.
- Faculté d'apprentissage à partir d'exemples représentatifs, par « rétro propagation du gradient ».
- L'apprentissage (ou construction du modèle) est automatique
- Résistance au bruit ou au manque de fiabilité des données
- Comportement moins mauvais en cas de faible quantité de données (comparé aux arbres de décision)

Inconvénients :

- L'absence de méthode systématique permettant de définir la meilleure topologie du réseau et le nombre de neurones à placer dans la ou les couches cachées.
- Le choix des valeurs initiales des poids du réseau et le réglage du pas d'apprentissage
- Les réseaux de neurones ne sont pas interprétables.

1.5.4. Machine à vecteur de support

Les machines à vecteurs de support (SVM : Support Vector Machines en anglais) ont été introduites en 1995 par Cortes et Vapnik [22]. Ils sont utilisés dans de nombreux problèmes d'apprentissage : reconnaissance de formes, catégorisation de texte ou même diagnostic médical. Les SVM s'appuient sur deux concepts : la marge maximale et la fonction du noyau. Ils résolvent les problèmes de discrimination non linéaire. La marge est la distance entre la limite de séparation et les échantillons les plus proches appelés vecteurs de support.

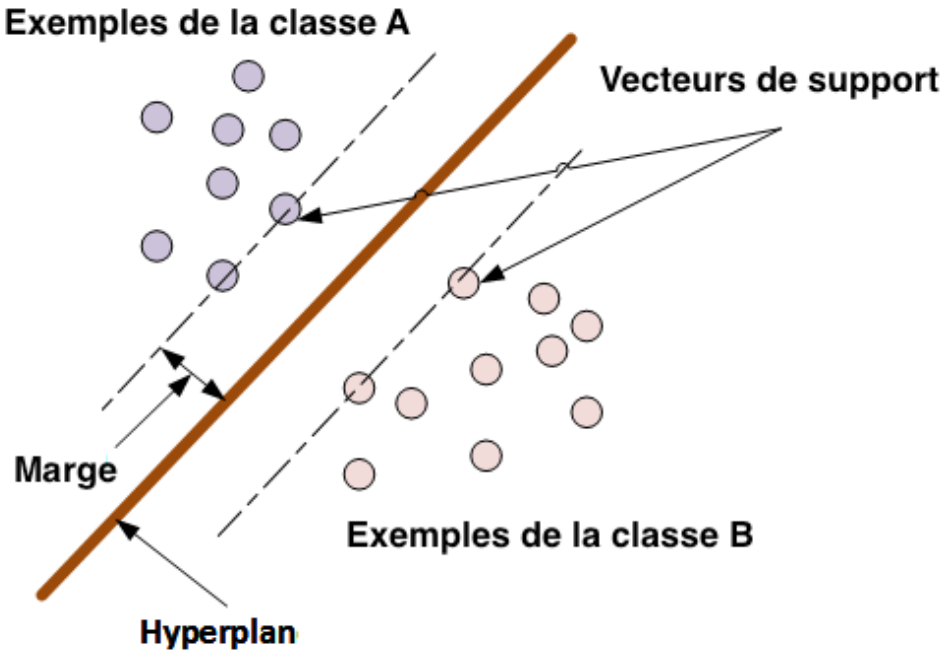


Figure 21 : Machine à vecteur de support

Dans un problème linéairement séparable, le SVM trouve un séparateur qui maximise la marge. Dans le cas d'un problème non linéaire, une fonction noyau est utilisée pour projeter les données dans un espace de grande dimension où elles sont linéairement séparables (Figure 22).

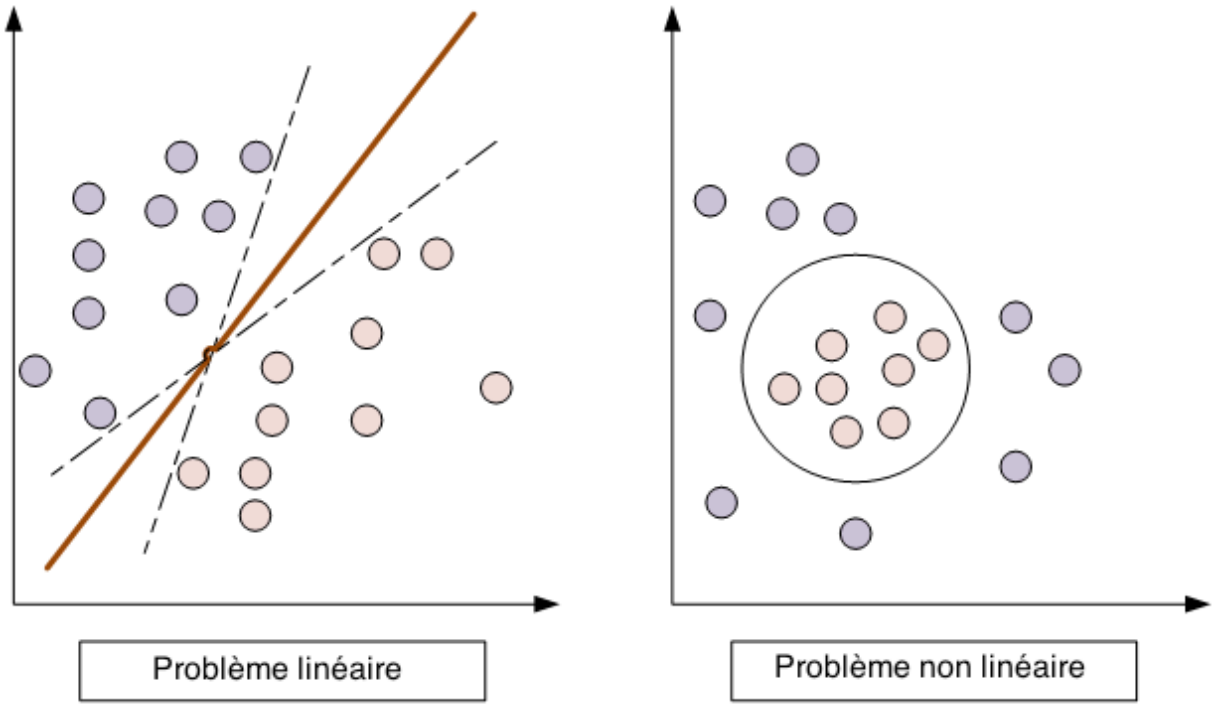


Figure 22: Différents problèmes de séparation

Plus formellement, une machine à vecteurs de support construit un hyperplan ou un ensemble d'hyperplan dans un espace de grande dimension, qui peut être utilisé pour la classification. Intuitivement, une bonne séparation est obtenue par l'hyperplan qui a la plus grande distance jusqu'au point d'apprentissage le plus proche (vecteur de support) de n'importe quelle classe (marge dite fonctionnelle), car plus la marge est grande, plus l'erreur de généralisation du classifieur est faible. Les SVM, sont une méthode de classification supervisée particulièrement adaptée au traitement de données de grande dimension. En ce qui concerne les techniques d'apprentissage classiques, SVM ne dépend pas de la dimension de l'espace de représentation des données. En utilisant une fonction noyau, ils permettent une classification non-linéaire. Le SVM est une technique d'apprentissage qui se développe depuis plus de 20 ans mais loin d'avoir atteint ses limites.

Avantages :

- Les SVM possèdent des fondements mathématiques solides.
- Les exemples de test sont comparés juste avec les supports vecteurs et non pas avec tous les exemples d'apprentissage
- Décision rapide.

Inconvénients :

- Choix empirique d'une fonction noyau appropriée pour le problèm
- Temps de calcul élevé lors d'une régularisation des paramètres de la fonction noyau.
- Grande quantité d'exemples en entrées implique un calcul matriciel important
- Difficulté à identifier les bonnes valeurs des paramètres (et sensibilité aux paramètres)
- Difficulté d'interprétations
- Classification binaire

Bien que les méthodes supervisées ont connu un grand succès dans le monde de l'apprentissage artificiel, leur plus grande limite réside dans le fait qu'elles nécessitent beaucoup de moyen humain pour l'étiquetage des données, d'où l'apparition des méthodes non supervisé aussi appelé méthode de regroupement ou clusturing en anglais.

2. Apprentissage non supervisé

Le regroupement est l'une des techniques les plus largement utilisées pour l'analyse exploratoire des données. On l'appelle aussi « apprendre sans enseignant » ou « apprendre par corrélation ». Ce type d'apprentissage est utilisé dans les cas où nous avons une base d'apprentissage dont les classes ne sont pas définies à l'avance.

Dans toutes les disciplines, des sciences sociales à la biologie en passant par l'informatique, les gens essaient d'obtenir une première intuition de leurs données en identifiant des groupes significatifs parmi les points de données. Par exemple, les biologistes computationnels regroupent les gènes sur la base de similitudes dans leur expression dans différentes expériences ; les détaillants regroupent les clients, sur la base de leurs profils de clients, à des fins de marketing ciblé ; et les astronomes regroupent les étoiles en fonction de leur proximité spatiale.

Le premier point que l'on devrait clarifier est, naturellement, qu'est-ce que le regroupement? Intuitivement, le regroupement consiste à grouper un ensemble d'objets de sorte que des objets similaires se retrouvent dans le même groupe et que des objets dissemblables soient séparés en différents groupes. Ce type d'apprentissage permet la construction automatique des classes sans l'intervention de l'expert. Cependant, cette approche nécessite une bonne estimation du nombre de classes.

Comme pour le monde supervisé, dans le non-supervisé il existe des méthodes et des algorithmes pour la classification des données, nous allons citer quelque uns.

2.1. Classification hiérarchique

La classification hiérarchique implique la création de clusters ayant un ordre prédéterminé du haut vers le bas. Par exemple, tous les fichiers et dossiers du disque dur sont organisés dans une hiérarchie. Il existe deux types de classification hiérarchique, Classification par division ou par agglomération.

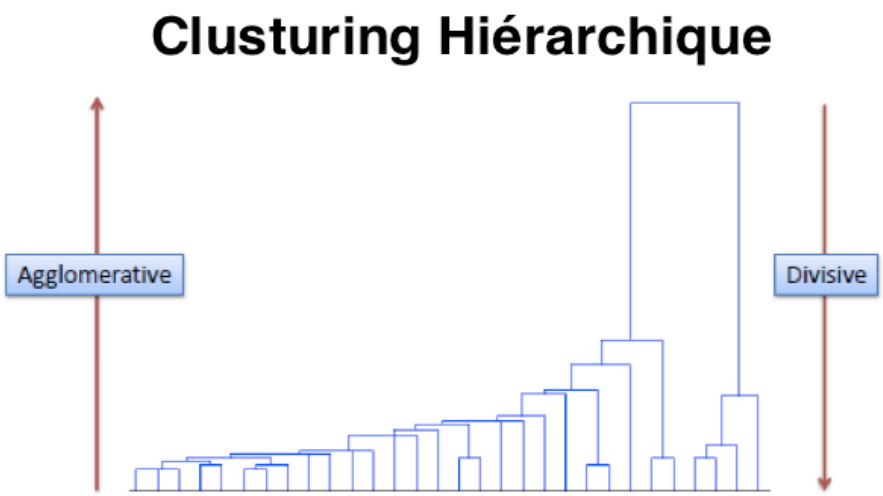


Figure 23: classification hiérarchique

- **Méthode par division :** Dans cette méthode, nous affectons toutes les observations à un seul cluster, puis partitionnez le cluster en deux clusters les moins similaires. Enfin, nous procédons récursivement sur chaque cluster jusqu'à ce qu'il y ait un cluster pour chaque observation.

- **Méthode d'agglomération :** Dans cette méthode, nous assignons chaque observation à son propre cluster. Ensuite, nous calculons la similarité (par exemple, la distance) entre chacun des clusters et nous joignons les deux clusters les plus similaires. Enfin, nous répétons les étapes 2 et 3 jusqu'à ce qu'il ne reste qu'un seul cluster.

Avant de commencer le regroupement, il est nécessaire de déterminer la matrice de proximité contenant la distance entre chaque point à l'aide d'une fonction de distance. Ensuite, la matrice est mise à jour pour afficher la distance entre chaque cluster.

2.2. K-moyenne

L'algorithme K-Means a pour objectif de partitionner N objets en K clusters dans lesquels chaque objet appartient au cluster avec la moyenne la plus proche. [23]
Cette méthode produit exactement K clusters différents de la plus grande distinction possible.

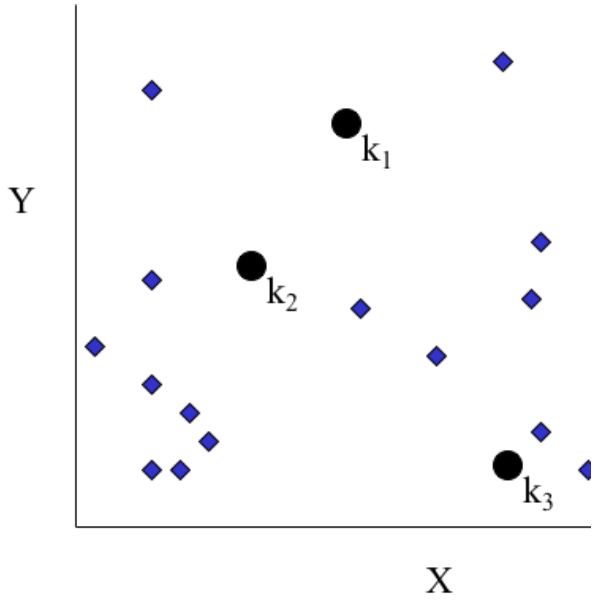
Le meilleur nombre de clusters K conduisant à la plus grande séparation (distance) n'est pas connu a priori et doit être calculé à partir des données. L'objectif du clustering K-Means est de minimiser la variance intra-cluster totale ou la fonction d'erreur quadratique.

L'algorithme k-means passe par ces étapes :

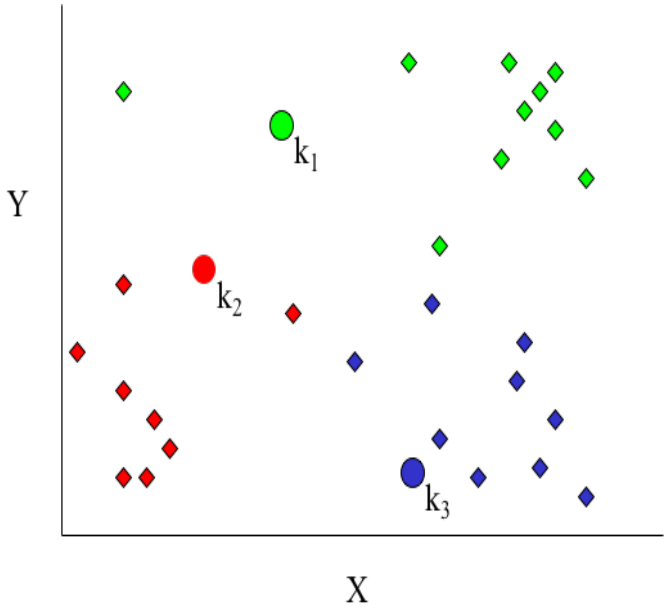
1. Choisir au hasard le centre de chacune des K classes,
2. Affecter chaque élément à la classe dont le centre lui est le plus proche (en utilisant par exemple une distance euclidienne),
3. Déplacer chaque centre vers la moyenne des éléments de la classe,
4. Répéter de 2 à 3 jusqu'à convergence.

Exemple de déroulement de l'algorithme avec 3 clusters :

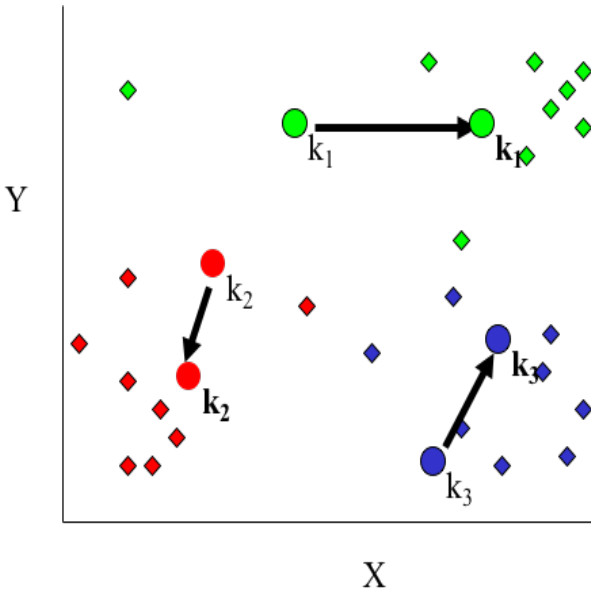
Dans la première étape, choisir 3 centres de classes (aléatoirement) (K1, K2 et K3)



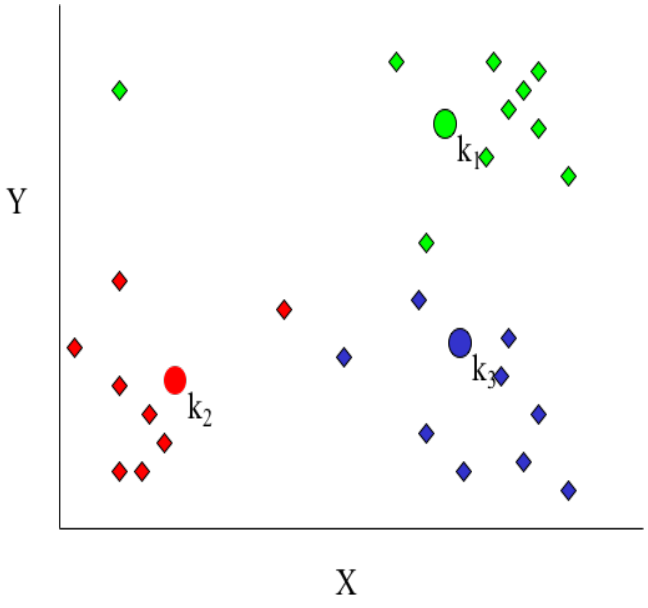
Etape 2 : Affecter chaque point à la classe dont le centre est le plus proche

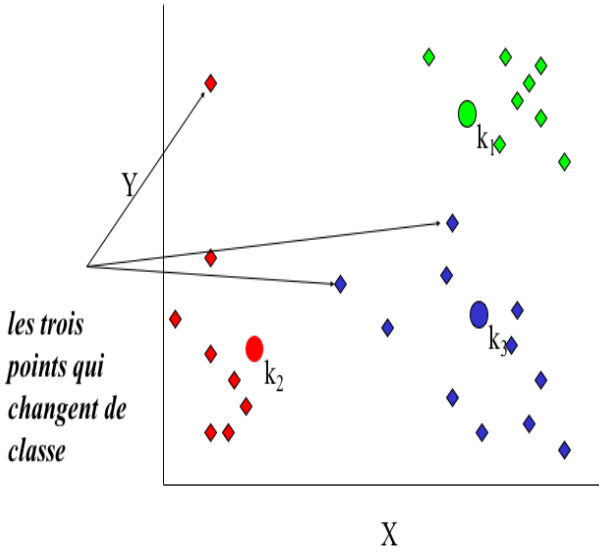


Etape 3 : Déplacer chaque centre de classe vers la moyenne de chaque classe.

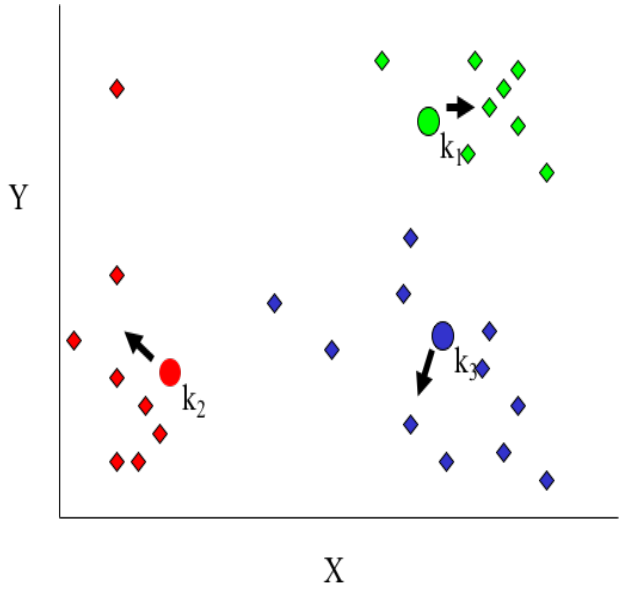


Etape 4 : Réaffecter les points qui sont plus proches du centre d'une autre classe

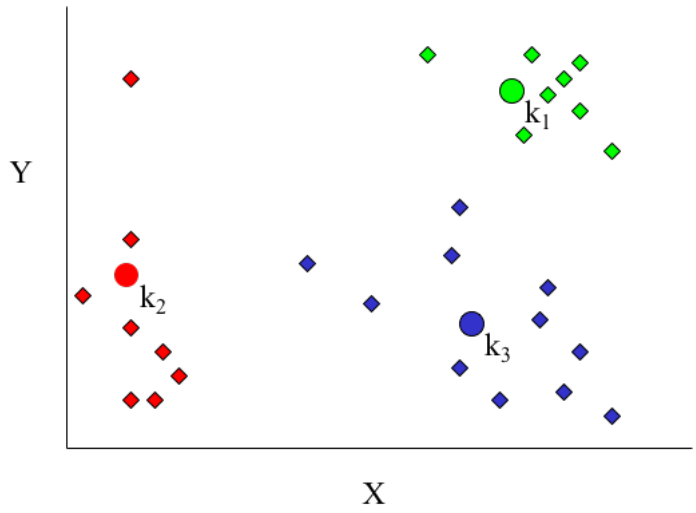




Puis, nous recalculons les moyennes des classes.



Déplacement des centres des classes vers les moyennes.



L'algorithme s'arrête quand il n'y aura plus de changement de centre. Comme toutes les autres techniques, K-means présente quelques avantages et inconvénients.

Avantages :

- Simple, compréhensible
- Éléments affectés automatiquement aux classes

Inconvénients :

- Nombre de classes fixé à l'avance
- Chaque élément doit être dans une classe
- Très sensible aux éléments marginaux (intérêt d'une normalisation)

3. Apprentissage semi-supervisé

Les méthodes modernes d'acquisition automatique permettent d'obtenir de nombreuses variables sur de nombreux individus pour un faible coût.

Dans le domaine médical, le praticien dispose souvent d'un grand échantillon de données non étiquetées et d'un plus petit échantillon de données étiquetées. Exemple, avec le développement des techniques d'analyse biologique, il est facile d'obtenir des millions de gènes. Cependant, la classification de ces gènes dans des classes est une tâche longue et difficile. Des situations similaires sont valables pour la télédétection, le traitement de signal et l'imagerie médicale.

Vu la disponibilité des données non étiquetées et la difficulté de leur annotation, les méthodes d'apprentissage semi-supervisé ont acquis une grande importance. A la différence de l'apprentissage supervisé, l'apprentissage semi-supervisé vise les problèmes avec relativement peu de données étiquetées et une grande quantité de données non étiquetées.

Plusieurs sortes de techniques ont été développées pour réaliser la tâche d'apprentissage semi-supervisé. Il existe principalement trois paradigmes qui abordent le problème de la combinaison des données labellisées et non labellisées afin d'améliorer les performances. De ce fait, nous citons en bref ces catégories : apprentissage semi-supervisé, apprentissage transductif et l'apprentissage actif.

L'Apprentissage semi-supervisé (SSL) renvoie à des méthodes qui tentent d'exploiter soit les données non étiquetées pour l'apprentissage supervisé où les exemples non étiquetés sont différents des exemples de test (Classification semi-supervisée); soit d'exploiter les données étiquetées pour l'apprentissage non supervisé (Regroupement semi-supervisé).

L'Apprentissage Transductif : regroupe les méthodes qui tentent d'exploiter les exemples non étiquetés. Il fournit le label uniquement pour les données disponibles non labellisées.

L'Apprentissage inductif : se réfère à des méthodes qui sélectionnent les exemples non étiquetés les plus importants, et un oracle peut être proposé pour l'étiquetage de ces instances. Il produit non seulement des labels pour données non labellisées, mais aussi un classifieur.

3.1. Classification semi-supervisée

Le principe de la classification semi-supervisé est l'utilisation d'un algorithme supervisé et l'entraîner sur des données avec labels (peu) puis exploiter les données sans labels (beaucoup) pour améliorer l'apprentissage. Il existe plusieurs techniques de classification semi-supervisé développées, parmi ces méthodes :

- Auto apprentissage (Self-Training).
- Co-training (Co-apprentissage)
- S3VM : séparateur semi-supervisé à vaste marge
- T-SVM : Transductive Support Vector Machines.

Comme exemple dans ce cours, nous allons voir l'algorithme de Self-Training.

3.1.1. Self-Training

C'est un algorithme très répandu dans le monde de la classification semi-supervisé. Le classifieur est d'abord entraîné sur les données étiquetées puis utilisé pour classer les données non étiquetées.
Parmi ces données nouvellement labélisées, l'algorithme sélectionne ceux qui ont le plus grand degré de confiance pour les réintégrer dans une nouvelle phase d'apprentissage supervisé (renforcer la base d'apprentissage avec plus de données labélisées). La procédure est répétée jusqu'à satisfaire un critère d'arrêt. Cette technique est appelée Self-Training car elle utilise ses propres prédictions pour s'améliorer à chaque itération.

3.1.2. Algorithme d'auto-apprentissage

Entrée D_E : Données étiquetées, D_{NE} : Données non-étiquetées
Répéter :
Faire apprendre une hypothèse supervisée h à partir de D_E
Appliquer h sur D_{NE} pour la prédiction des étiquettes
Ajouter les éléments les plus confiants de D_{NE} à D_E
Jusqu'à satisfaction du critère d'arrêt (nombre d'itération par exemple)

3.2. Regroupement semi-supervisé

L'apprentissage semi-supervisée par regroupement (ou Semi-Supervised Clustering en anglais), introduit des connaissances expertes partielles dans un algorithme de clustering.
Contrairement à la classification semi-supervisée, où l'apprentissage est fait d'une manière supervisée lorsqu'on a un nombre de données labélisées plus ou moins suffisant pour un premier apprentissage, le clustering semi-supervisé est utilisé lorsque que la quantité des données étiquetées est tellement faible ou partielle qu'il est impossible d'appliquer des techniques supervisées.
Dans ce cas-là, les connaissances a priori se présentent soit sous la forme d'étiquettes de classe, soit sous la forme de contraintes sur des paires de données (exemple : si deux données sont similaires et doivent alors être regroupées ensemble, ce sont des contraintes "must-link", ou bien, si elles sont différentes et donc ne doivent pas être regroupées ensemble, ce sont des contraintes "cannot-link"). Elles sont ensuite utilisées pour guider le processus de clustering dans l'espace des solutions.
Ils existent plusieurs algorithmes dans cette famille, nous nous intéressant dans ce cours à l'algorithme k-means dans sa version semi-supervisé.

3.2.1. Seeded K-Means

Les données étiquetées fournies par l'utilisateur sont utilisées pour l'initialisation : le centre initial pour le cluster i est la moyenne des points labélisés ayant l'étiquette i.
Les points de départ ne sont utilisés que pour l'initialisation et non pour les étapes suivantes (donc peuvent changer de classe durant l'apprentissage).

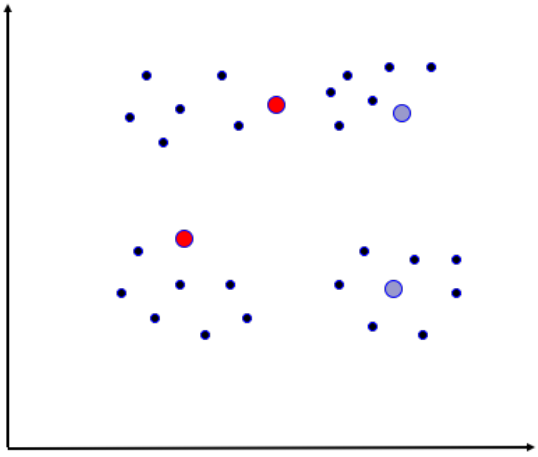
3.2.2. Constrained K-Means

Les données étiquetées fournies par l'utilisateur sont utilisées pour initialiser l'algorithme K-Means. Les étiquettes de ces données restent inchangées dans les étapes d'attribution de classes et seules les étiquettes des données non labélisées sont recalculées.

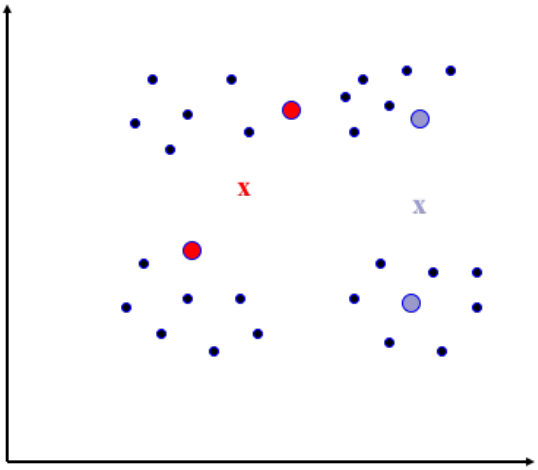
3.2.3. COP K-means

C'est l'algorithme K-Means avec les contraintes must-link (doit être dans le même cluster) et cannot-link (ne peut pas être dans le même cluster). Les centres de cluster sont choisis au hasard, mais au fur et à mesure qu'une donnée est affectée au cluster le plus proche, l'algorithme doit vérifier que cette affectation ne viole aucune des contraintes imposées par les données labélisées.

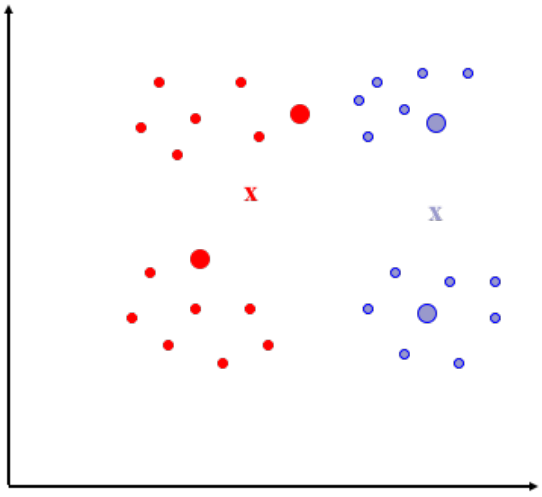
Exemple de l'algorithme Constrained K-means:



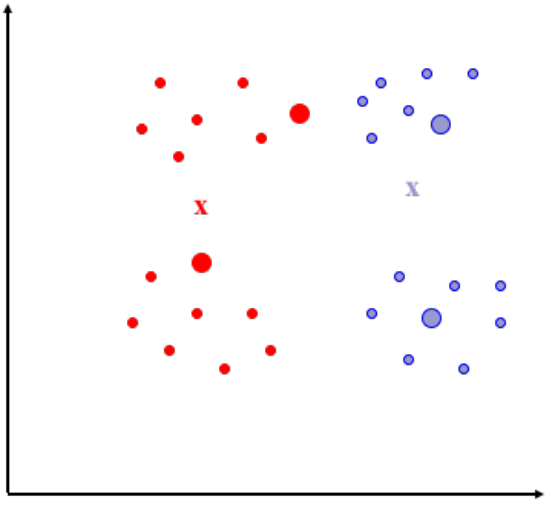
Etape 1 : Ajout des données labélisées



Etape 2: Calcul du centre de chaque classe en prenant compte que les données labélisées



Etape 3 : Affection des données aux clusters



Etape 4 : calcul des nouveaux centres et répétition (étape 2,3) jusqu'à la convergence

4. Apprentissage par renforcement

L'apprentissage par renforcement est un autre type d'apprentissage automatique qui permet aux machines et aux agents logiciels de déterminer automatiquement le comportement idéal dans un contexte spécifique, afin de maximiser leurs performances. La rétroaction simple de récompense est exigée pour que l'agent apprenne son comportement ; c'est ce qu'on appelle le signal de renforcement.

Il existe de nombreux algorithmes différents pour résoudre ce problème. En fait, l'apprentissage par renforcement est défini par un type spécifique de problème, et toutes ses solutions sont classées en tant qu'algorithmes d'apprentissage par renforcement. Dans le problème, un agent est supposé décider de la meilleure action à sélectionner en fonction de son état actuel. Lorsque cette étape est répétée, le problème est connu comme un processus de décision de Markov.

L'apprentissage par renforcement permet à l'agent d'apprendre son comportement en fonction des réactions de l'environnement. Ce comportement peut être appris une fois pour toutes, ou continuer à s'adapter au fil du temps. Si le problème est modélisé avec soin, certains algorithmes d'apprentissage par renforcement peuvent converger vers l'optimum global; C'est le comportement idéal qui maximise la récompense.

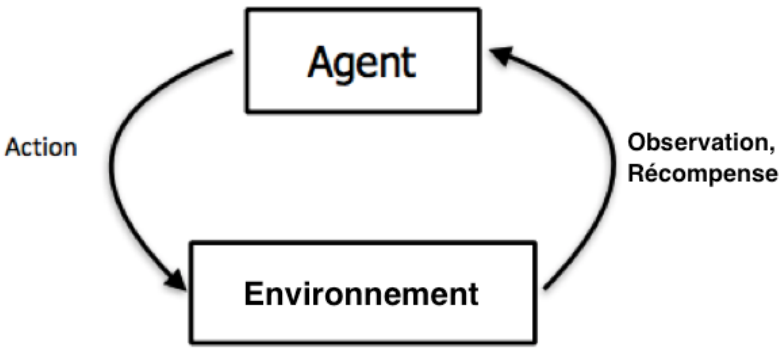


Figure 24 : Apprentissage par renforcement

Comme indiqué sur la figure, l'agent observe l'environnement, prend une action pour interagir avec l'environnement et reçoit une récompense positive ou négative.

5. Conclusion

Dans ce chapitre, nous avons vu les différents types d'apprentissage artificiel avec quelques algorithmes souvent utilisés dans la littérature. Le choix de l'algorithme à utiliser dépend du type des données en notre possession. Par exemple si nous avons une base de données entièrement labélisées le meilleur choix sera de faire un apprentissage supervisé. Si nous avons une base non labélisée, nous devons faire un choix pour une méthode non supervisée. Dans le cas où nous avions une base avec des données partiellement labélisées, si la majorité de ces données sont étiquetées on opte pour la classification semi-supervisée, dans le cas contraire le regroupement semi-supervisé est vivement conseillé.

CHAPITRE III

Amélioration de l'apprentissage

Ce chapitre introduit les techniques évolutionnaires utilisées pour l'améliorations des classifieurs.

1. Introduction

Le minimum local dans une fonction (comme la fonction de l'erreur) est la plus petite valeur dans un espace bien défini et non pas dans tout l'espace de recherche (solution) de la fonction (le minimum global est la valeur la plus petite de tout l'espace).

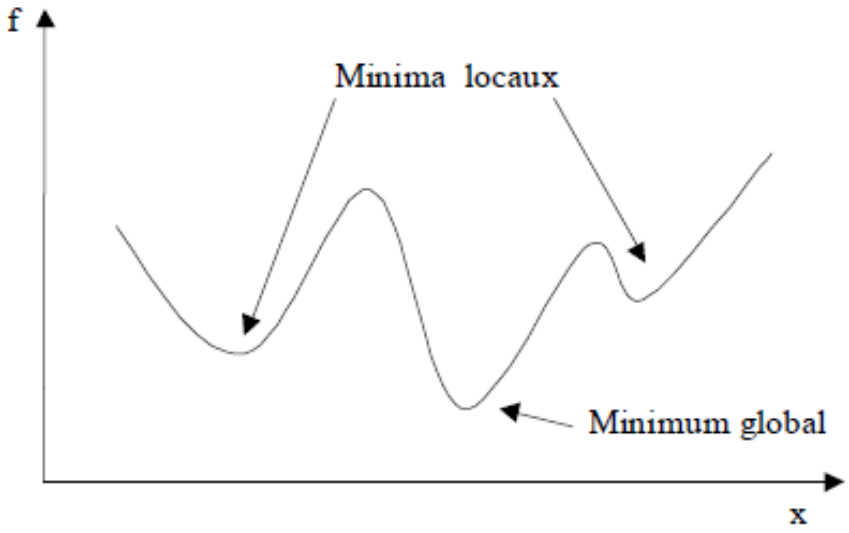


Figure 25 : Minimum local et global

Quelques classifieurs classiques tel que les réseaux de neurone, et malgré leurs performances approuvées, souffrent du problème du minimum local vu qu'ils utilisent la descente du gradient.

Pour atteindre le minimum global, les algorithmes tels que les réseaux de neurone, doivent être exécuter plusieurs fois en utilisant différentes initialisations des poids ou bien les combiner avec des méthodes d'optimisation évolutionnaires comme les algorithmes génétiques (AG) ou les algorithmes d'optimisation par essaimes particulaires (PSO).

2. Les Algorithmes Génétiques

Les algorithmes génétiques sont un type d'algorithme d'optimisation, ce qui signifie qu'ils sont utilisés pour trouver la ou les solutions optimales d'un problème de calcul donné qui maximise ou minimise une fonction particulière. Les algorithmes génétiques représentent une branche du domaine d'étude appelée calcul évolutif.

Un algorithme génétique est une heuristique de recherche inspirée de la théorie de l'évolution naturelle de Charles Darwin [24]. Cet algorithme reflète le processus de sélection naturelle où les individus les plus aptes sont sélectionnés pour la reproduction afin de produire la progéniture de la génération suivante.

2.1. Notion de sélection naturelle

Le processus de sélection naturelle commence par la sélection des individus les plus aptes d'une population. Ils produisent des descendants qui héritent des caractéristiques des parents et seront ajoutés à la prochaine génération. Si les parents ont une meilleure condition physique, leur progéniture sera meilleure que celle des parents et aura une meilleure chance de survivre (figure 26). Ce processus continue d'itérer et à la fin, une génération avec les individus les plus aptes sera trouvée.

Figure 26 : Processus de sélection naturelle

Cette notion peut être appliquée pour un problème de recherche. Nous considérons un ensemble de solutions pour un problème et sélectionnons l'ensemble des meilleures d'entre elles.

Cinq phases sont considérées dans un algorithme génétique :

1- Création de la population initiale,
2- Evaluation des individus (fonction Fitness),
3- Sélection,
4- Croisement (Crossover),
5- Mutation.

2.2. Population initiale

Le processus commence avec un ensemble d'individus appelé Population. Chaque individu est une solution au problème que vous voulez résoudre.

Un individu est caractérisé par un ensemble de paramètres (variables) connus sous le nom de gènes. Comme indiqué dans la figure 27, les gènes sont joints dans une chaîne pour former un chromosome (solution).

Dans un algorithme génétique, l'ensemble des gènes d'un individu est représenté à l'aide d'une chaîne, en termes d'un alphabet. Habituellement, les valeurs binaires sont utilisées (chaîne de 1 et 0). Nous disons que nous codons les gènes dans un chromosome.

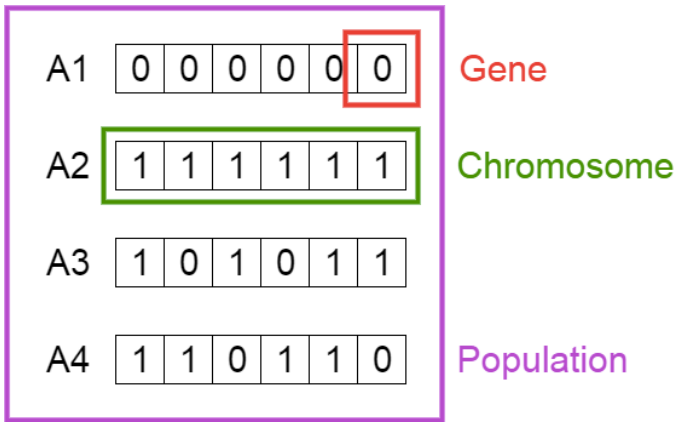


Figure 27 : Population d'un AG

2.3. Fonction Fitness

La fonction de Fitness détermine la capacité d'un individu (la capacité d'un individu à rivaliser avec d'autres individus). Chaque individu a un score de fitness et la probabilité qu'un individu soit sélectionné pour la reproduction est basée sur ce score.

2.4. Sélection

L'idée de la phase de sélection est de sélectionner les individus les plus aptes et de les laisser transmettre leurs gènes à la génération suivante.
Deux paires d'individus (parents) sont sélectionnées en fonction de leurs scores Fitness. Les personnes ayant un score élevé ont plus de chance d'être sélectionnées pour la reproduction.

2.5. Croisement

Le croisement est la phase la plus significative dans un algorithme génétique. Pour chaque paire de parents à accoupler, un point de croisement est choisi au hasard parmi les gènes. Par exemple, considérez le point de croisement comme étant 3, comme indiqué ci-dessous dans la figure 28.

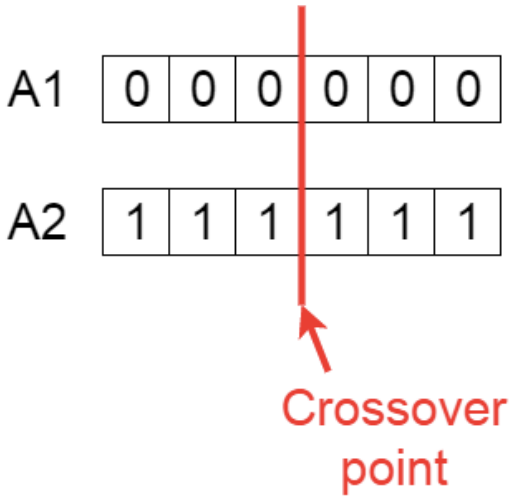


Figure 28 : Point de croisement

Les descendants sont créés en échangeant les gènes des parents entre eux jusqu'à ce que le point de croisement soit atteint (figure 29).

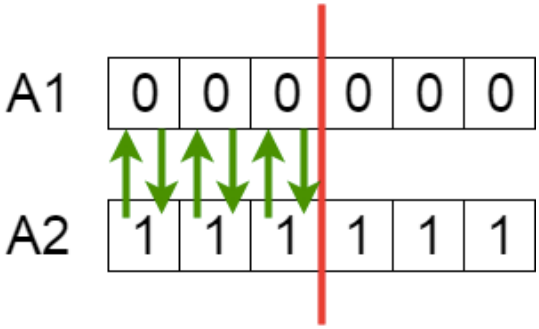


Figure 29 : Échange des gènes entre les parents

La nouvelle progéniture est ajoutée à la population (figure 30).

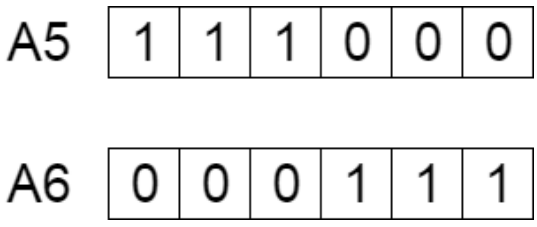


Figure 30 : Nouvelle progéniture

2.6. Mutation

Chez certains nouveaux descendants formés, certains de leurs gènes peuvent être soumis à une mutation avec une faible probabilité aléatoire. Cela implique que certains gènes du chromosome seront modifiés (figure 31).

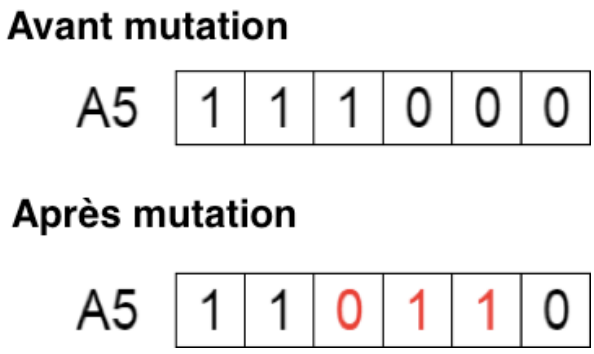


Figure 31 : Mutation d'un chromosome

La mutation se produit pour maintenir la diversité au sein de la population et empêcher la convergence prématurée de l'algorithme.

2.7. Terminaison

L'algorithme se termine si la population a convergé (ne produit pas de descendance qui sont significativement différentes de la génération précédente). Ensuite, il est dit que l'algorithme génétique a fourni un ensemble de solutions à notre problème.

Notons bien que la population a une taille fixe. Au fur et à mesure que de nouvelles générations se forment, les individus les moins bons seront supprimés, offrant ainsi de l'espace pour de nouveaux enfants.

La séquence des phases est répétée pour produire des individus de chaque nouvelle génération qui sont meilleurs que la génération précédente.

2.8. Le pseudo code de l'algorithme AG

```
DÉBUT
    Générer la population initiale
    Calculer le score
    REPETER
        Calcul de fitness
        Sélection
        Croisement
        Mutation
    JUSQU'À ce que la population ait convergé
ARRÊTEZ
```

Avantage :

Les algorithmes génétiques présentent les avantages :

- D'êtres des méthodes robustes à l'initialisation (c'est-à-dire que leurs convergences ne dépendent pas de la valeur initiale.
- Permettent de déterminer l'optimum global d'une fonctionnelle ou de s'en approcher même sur des problèmes complexes.
- Ils sont parallélisables (nous pouvons diminuer le temps de calcul si nous parallélisons le programme).

Inconvénients :

L'inconvénient majeur des algorithmes génétiques réside dans le nombre important d'évaluations nécessaires et leur temps de convergence.

3. Optimisation par Essaims Particulaires (PSO)

PSO a été proposé par Eberhart et Kennedy en 1995 [25], développé par la suite dans des milliers d'articles scientifiques, et appliqué à de nombreux problèmes, par exemple la formation des réseaux neuronaux, l'exploration de données (Data Mining), le traitement du signal et l'optimisation.

3.1. Principe de base

PSO est une méta-heuristique de l'intelligence de l'essaim inspirée du comportement de groupe des animaux, par exemple les troupeaux d'oiseaux. De même que pour les algorithmes génétiques, il s'agit d'une méthode basée sur la population, c'est-à-dire qu'elle représente l'état de l'algorithme par une population, modifiée itérativement jusqu'à ce qu'un critère de terminaison soit satisfait.

Dans les algorithmes PSO, la population P = {p1, ..., pn} des solutions réalisables est souvent appelée un essaim. Les solutions réalisables p1, ..., pn sont appelées particules. La méthode PSO considère l'ensemble R^d de solutions comme un « espace » où les particules « bougent ». Pour résoudre des problèmes pratiques, le nombre de particules est habituellement choisi entre 10 et 50.

3.2. Topologie de l'essaim

Chaque particule i a son voisinage N_i (un sous-ensemble de P). La structure du voisinage est appelée la topologie d'essaim, qui peut être représentée par un graphique. Les topologies habituelles sont : topologie entièrement connectée, topologie partiellement connectée (avec un nombre prédéfini de voisin) et topologie de cercle.

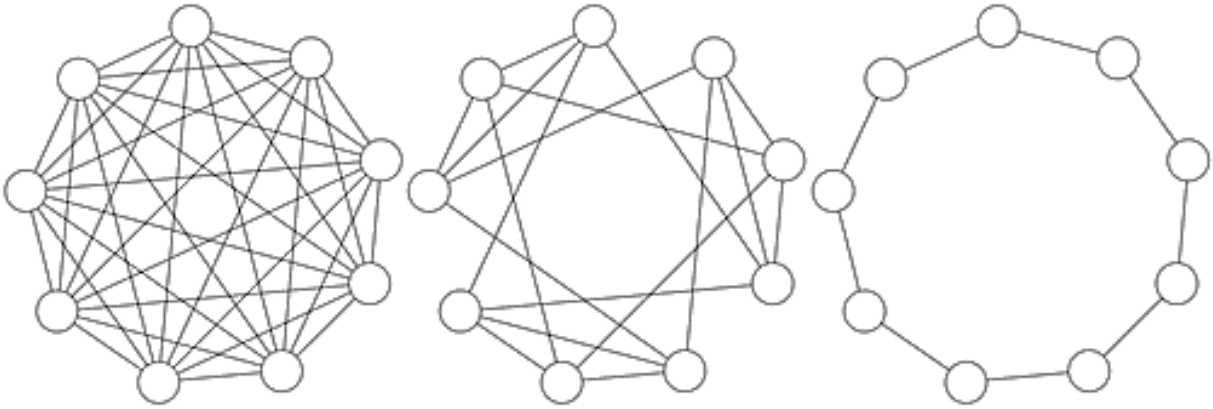


Figure 32 : les topologies les plus utilisées

3.3. L'algorithme PSO

Comme indiqué précédemment, PSO simule les comportements du flocage des oiseaux. Supposons le scénario suivant : un groupe d'oiseaux recherche aléatoirement de la nourriture dans une zone. Il n'y a qu'un seul morceau de nourriture dans la zone fouillée. Les oiseaux ne savent pas où est la nourriture mais savent à chaque itération, à quelle distance ils sont de la nourriture. Alors, quelle est la meilleure stratégie pour trouver la nourriture? L'approche la plus efficace consiste à suivre l'oiseau le plus proche de la nourriture.

L'algorithme du PSO reprend ce procédé pour résoudre les problèmes d'optimisation. Dans PSO, chaque solution unique est un "oiseau" dans l'espace de recherche. Nous l'appelons "particule". Toutes les particules ont des valeurs de score qui sont évaluées par la fonction de fitness pour être optimisées, et ont des vitesses qui dirigent le vol des particules. Les particules volent à travers l'espace du problème en suivant les particules optimales actuelles.

PSO est initialisé avec un groupe de particules aléatoires (solutions), puis recherche des optima en mettant à jour les générations. Dans chaque itération, chaque particule est mise à jour en suivant deux "meilleures" valeurs. La première est la meilleure solution (fitness) obtenue jusqu'à présent. Cette valeur est appelée pbest (meilleure position de p). Une autre "meilleure" valeur qui est suivie par l'optimiseur d'essaim de particules est la meilleure valeur obtenue jusqu'ici par n'importe quelle particule de la population. Cette meilleure valeur est appelée gbest (pour global best). Quand une particule prend une partie de la population comme ses voisins topologiques, la meilleure valeur est un meilleur local et s'appelle lbest (pour local best).

Après avoir trouvé les deux meilleures valeurs, la particule met à jour sa vitesse et ses positions avec l'équation suivante (a) et (b).

$$v_i^{(t+1)} = w\, v_i^{(t)} + \phi_1 u_1 (pbest_i^{(t)} - x_i^{(t)}) + \phi_2 u_2 (gbest_i^{(t)} - x_i^{(t)}) \qquad (a)$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \qquad (b)$$

Caractéristiques de la particule i à l'itération t:

xi (t) est la position (un vecteur d-dimensionnel)
vi (t) est la vitesse ; c'est la taille de pas entre xi (t) et xi (t + 1)
w (t) est le poids d'inertie ; un facteur d'amortissement qui diminue pendant le calcul généralement d'environ 0,9 à environ 0,4.
ϕ_1, ϕ_2 sont les coefficients d'accélération ; généralement entre 0 et 4.

Les symboles u1 et u2 dans la formule de vitesse (a) représentent des variables aléatoires avec la distribution U (0,1). La première partie de la formule de la vitesse est appelée « inertie », la seconde «la composante cognitive (personnelle) », la troisième est «la composante sociale (de voisinage) ». La position de la particule i change selon la formule (b)

3.4. Le pseudo code de l'algorithme PSO

```
Pour chaque particule
   Initialiser la position de la particule
Fin

Faire
   Pour chaque particule
         Calculer la valeur de fitness
         Si la valeur de fitness est meilleure que la meilleure valeur de fitness (pBest)
         définir la valeur actuelle comme le nouveau pBest
   Fin
   Choisissez la particule avec la meilleure valeur de fitness de toutes les particules comme
   le gBest
   Pour chaque particule
         Calculer la vitesse des particules selon l'équation (a)
         Mettre à jour la position des particules selon l'équation (b)
   Fin
Tant que les itérations maximales ou les critères d'erreur minimum ne sont pas atteints
```

3.5. Contrôle des paramètres PSO

Dans le cas ci-dessus, nous pouvons remarquer qu'il existe deux étapes clés lors de l'application de PSO à des problèmes d'optimisation : la représentation de la solution et la fonction de Fitness.

Il n'y a pas beaucoup de paramètres à régler dans PSO. Voici une liste avec leurs valeurs typiques.

- **Le nombre de particules (population)** : la taille typique est 20 - 50. En fait, pour la plupart des problèmes, 10 particules sont assez satisfaisantes pour obtenir de bons résultats. Pour certains problèmes difficiles ou spéciaux, cela peut aller jusqu'à 100 ou 200 particules.

- **Dimension des particules :** elle est déterminée par le problème à optimiser,

- **Facteurs d'apprentissage (coefficient d'accélération) :** ϕ_1, ϕ_2 sont généralement égaux à 2. Cependant, d'autres paramètres ont également été utilisés dans différents articles. Mais généralement ϕ_1 est égal à ϕ_2 et va de [0,4].

- **La condition d'arrêt :** le nombre maximal d'itérations exécutées par le PSO et l'exigence d'erreur minimale.

3.6. Version globale vs version locale

Il existe deux versions de PSO. Version globale (utilise gbest) et locale (utilise un voisinage et donc lbest). La version globale est plus rapide mais pourrait converger vers l'optimum local pour certains problèmes. La version locale est un peu plus lente mais pas facile à piéger dans les optimums locaux. Nous pouvons utiliser la version globale pour obtenir un résultat rapide et utiliser la version locale pour affiner la recherche.

3.7. Variantes de PSO

Il existe un grand nombre de variante différentes des PSO, qui habituellement modifie la formule pour le changement de vitesse (par exemple, au lieu de u1 et u2 ils utilisent les matrices diagonales U1 et U2, dans d'autres variantes ils n'utilisent pas d'inertie, mais imposent une limite supérieure sur la vitesse des particules, il y a le PSO dit "entièrement informé" une modification populaire utilisant un "coefficient de constriction").
Il existe des variantes de PSO pour l'optimisation contrainte, pour l'optimisation discrète et pour l'optimisation multi-objective.

Avantages

- Très simple et facile à programmer
- Trouve le minimum global s'il existe
- PSO présente l'avantage d'être efficace sur une grande variété de problèmes, sans pour autant que l'utilisateur ait à modifier la structure de base de l'algorithme

Inconvénients

- Il n'y a pas de théorie générale de la convergence applicable aux problèmes pratiques et multidimensionnels.
- Pour obtenir des résultats satisfaisants, il est parfois nécessaire de régler les paramètres d'entrée et d'expérimenter différentes versions de la méthode PSO.
- Certaines versions de la méthode PSO dépendent du choix du système de coordonnées des particules.

4. Comparaisons entre l'algorithme génétique et PSO

La plupart des techniques évolutives ont la procédure suivante :

1. Génération aléatoire d'une population initiale
2. Calcul d'une valeur de fitness pour chaque sujet. Cela dépendra directement de la distance à l'optimum.
3. Reproduction de la population basée sur les valeurs de fitness.
4. Si les exigences sont satisfaites, arrêtez. Sinon, retournez à 2.

De la procédure, nous pouvons remarquer que PSO partage de nombreux points communs avec AG. Les deux algorithmes commencent avec un groupe d'une population générée aléatoirement, les deux ont des valeurs de fitness pour évaluer la population. Les deux mettent à jour la population et recherchent l'optimum avec des techniques aléatoires. Les deux systèmes ne garantissent pas le succès.

Cependant, PSO n'a pas d'opérateurs génétiques comme le croisement et la mutation. Les particules se mettent à jour avec la vitesse interne. Ils ont aussi de la mémoire, ce qui est important pour l'algorithme.

Comparé aux algorithmes génétiques (GA), le mécanisme de partage d'informations dans PSO est significativement différent. Dans les GA, les chromosomes partagent des informations entre eux. Ainsi, toute la population se déplace comme un groupe vers une zone optimale. Dans PSO, seul gBest (ou lBest) donne l'information aux autres. C'est un mécanisme de partage d'information à sens unique. L'évolution ne cherche que la meilleure solution. Par rapport à GA, toutes les particules ont tendance à converger rapidement vers la meilleure solution, même dans la version locale dans la plupart des cas.

5. Conclusion

Dans ce chapitre, nous avons vu qu'il existe une autre forme d'intelligence dite collective qui est le fruit d'une collaboration entre plusieurs individus. Les Algorithme génétique et les méthodes d'optimisation par essaimes particulaires sont les modèles les plus utilisés dans la littérature. Ces méthodes peuvent êtres utiliser seules ou bien combiner avec des algorithmes classiques comme les réseaux de neurones.

Conclusion générale

Nous avons commencé ce polycopié de cours d'apprentissage artificiel par des définitions et des généralités sur l'intelligence artificielle en général et l'apprentissage automatique en particulier avec un historique sur les moments fort de l'évolution de ce domaine.

Dans le deuxième chapitre, nous avons expliqué les différentes familles d'apprentissage (supervisé, non supervisé, semi supervisé et par renforcement) avec des illustrations et des algorithmes très connu dans l'état de l'art ainsi que les méthodes d'évaluation des classifieurs (Taux de classification, spécificité, sensibilité et la courbe ROC).

Dans le dernier chapitre, nous avons introduit les méta-heuristiques basées sur l'intelligence collective (méthodes évolutionnaires) en étudiant deux des algorithmes les plus connus à savoir les algorithmes génétiques et le PSO.

A l'issue de la lecture de ce polycopié et la participation aux cours, l'étudiant sera en mesure de comprendre les bases de l'apprentissage artificiel et faire la différence entre ses types. Il sera également en mesure de construire des classifieurs intelligents qui rependent eux besoins de l'utilisateur en prenant compte de la particularité de la base de données fourni. Aussi le fait d'évaluer son modèle en utilisant les techniques vues dans ce cours, l'étudiant aura la possibilité d'améliorer son modèles en utilisant les méta-heuristiques et ainsi voir l'impact de cette amélioration.

Bibliographie

[1] Deep Dream, https://deepdreamgenerator.com, site consulté le 05 janvier 2018.

[2] M. Minsky, The Society of Mind, New York: Simon & Schuster, 1986.

[3] J. Han, M. Kamber et J. Pei, Data Mining: Concepts and Techniques, Third edition, The Morgan Kaufmann Series in Data Management Systems, 2011.

[4] ENIAC, https://www.britannica.com/technology/ENIAC, Site consulté le 08 septembre 2017.

[5] L. Rougetet, Un ordinateur champion du monde d'Échecs : histoire d'un affrontement homme-machine, Sciences du jeu [En ligne], 5 | 2016, mis en ligne le 24 février 2016, consulté le 01 mars 2017.

[6] F. Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, In: Palm G., Aertsen A. (eds) Brain Theory. Springer, Berlin, Heidelberg, 1986.

[7] F. Muller et W. Taylor, «A parallel processing model for the study of possible pattern recognition mechanisms in the brain,» *Pattern Recognition,* vol. 8(1), pp. 47-52, 1976.

[8] K. Kawai, R. Mizoguchi, O. Kakusho et J. Toyoda, «A framework for ICAI systems based on Inductive Inference and Logic Programming,» chez *Proc. of 3rd Logic Programming Conference*, 1986.

[9] J. Ferber, Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence, Addison-Wesley, 1999.

[10] M. Campbell, A. J. Hoane et F.-h. Hsu, «Deep Blue,» *Artificial Intelligence,* vol. 134(1), pp. 57-83, January 2002.

[11] S. N. Srihari and S. Lee, «Automatic Handwriting Recognition and Writer Matching on Anthrax-related Handwritten Mail,» chez *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, Niagara-on-the-lake, Ontario, Canada, 2002.

[12] P. J. Springer, Military Robots and Drones: A Reference Handbook (Contemporary World Issues), ABC-CLIO , 2013.

[13] Yann LeCun, Yoshua Bengio et Geoffrey Hinton, «Deep learning,» *Nature,* vol. 521(7553), p. 436.

[14] Horus, «https://horus.tech/?l=en_us,» p. consulté le 5 janvier 2018.

[15] A. P. Bradley, «The use of the area under the ROC curve in the evaluation of machine learning algorithms,» *Pattern Recognition,* vol. 30(7), pp. 1145-1159, 1997.

[16] J. Oja et E. Laaksonen, «Classification with learning k-nearest neighbors,» chez *IEEE International Conference on Neural Networks*, 1996.

[17] L. Breiman, H. Friedman, J., J. Olshen et R. A. Stone, «Classification and Regression Trees,» 1984.

[18] K. Ayman, E. Idrees, A. M. et E. S. Ahmed, «Enhancing Iterative Dichotomiser 3 algorithm for classification decision tree,» *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 6(2), pp. 70--79, 2016.

[19] J.-B. Lamy, A. Ellini, V. Ebrahiminia, J.-D. Zucker et H. F. a. A. Venot, «Use of the C4.5 machine learning algorithm to test a clinical guideline-based decision support system,» *tud Health Technol Inform,* vol. 136, p. 223–228, 2008.

[20] W. S. McCulloch et W. Pitts, «A logical calculus of the ideas immanent in nervous activity,» *The bulletin of mathematical biophysics,* vol. 5(4), p. 115–133, 1943.

[21] D. E. Rumelhart, G. E. Williams et R. J. Hinton, «Learning internal representations by error propagation,» *In Parallel distributed processing: explorations in the microstructure of cognition,* vol. 1, pp. 318-362, 1986.

[22] C. Vapnik et V. Cortes, «Support-Vector Networks,» *Machine Learning,* vol. 20(3), p. 273–297, 1995.

[23] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. Piatko et R. S. a. A. Y. Wu, «An Efficient k-Means Clustering Algorithm: Analysis and Implementation,» 2000.

[24] K. Deb, «An introduction to genetic algorithms,» *Kalyanmoy Deb,* vol. 24, p. 293–315, 1999.

[25] J. Eberhart et R. Kennedy, «Particle swarm optimization,» chez *EEE International Conference on Neural Networks, Perth, WA*, 1995.