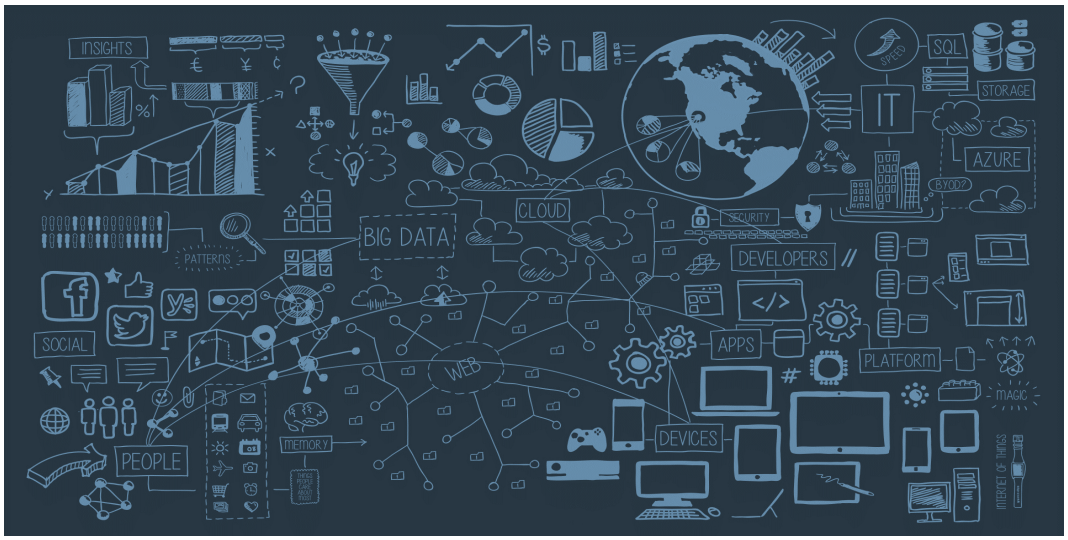
ALDEBARAN
SoftBank Group

Stage de fin d'études

Automatisation de l'analyse d'incidents par l'utilisation du Machine Learning



Étudiant :
Florian Lepont

Tuteur responsable :
M. Emmanuel Nalepa

Hiver 2015

Remerciements

Je tiens tout d'abord à remercier l'entreprise *Aldebaran* de m'avoir accueilli afin de réaliser mon stage ingénieur au sein de l'équipe "Qualification Hardware Pepper".

Je remercie tout particulièrement M Emmanuel Nalepa de m'avoir permis de participer à ce projet captivant et formateur, m'offrant ainsi l'opportunité d'étoffer mes connaissances dans plusieurs disciplines. Sa disponibilité et sa pédagogie ont été des atouts essentiels à mon enrichissement technique.

J'adresse également mes remerciements à toutes les personnes qui m'ont proposé leur aide durant ces vingt-cinq semaines : M. Angelica Lim qui m'a fournie de précieuses informations, notamment dans le domaine du Machine Learning.

Je tiens enfin à remercier l'équipe "Qualification Hardware Pepper" pour l'aide qu'elle m'a apporté et sa très bonne humeur.

Résumé

Mon stage ingénieur, réalisé dans le cadre de ma cinquième année de formation à l'École Nationale d'Ingénieurs de Brest (ENIB), s'est déroulé au sein du département "Qualification Hadware Pepper (QWP) de l'entreprise Aldebaran. La société Parisienne s'est fait connaitre dans le monde des nouvelles technologies et de la robotique humanoïde grâce au développement de son premier produit, "Nao". A l'origine, le robot se prédestine à l'univers de la recherche et aux universités. La société cherche aujourd'hui à conquérir de nouveaux marchés en offrant des produits et des services au monde de l'entreprise et aux particuliers. Cela se traduit notamment par le développement d'un tout nouveau produit : "Pepper".

Cette extension du marché s'accompagne d'une montée en puissance de la fabrication de robots "Pepper". Cela induit le développement d'une nouvelle génération d'outils de production et de post-production. Un des dispositif mis en place est le "Filtering Test" : à la fin de la chaine de production, les robots sont soumis à une série de tests qui visent à mettre à l'épreuve leurs différentes parties mécaniques et électroniques. Lorsqu'une erreur est détectée, les différentes données du robot son enregistrées (e.g. température des fusibles, valeurs de l'accéléromètre, etc.). Afin de déterminer qu'elles sont les causes qui ont entrainé l'apparition de l'anomalie sur le robot, chaque donnée est étudiée minutieusement et des hypothèses sont émises. Cette tâche dite d'investigation peu s'avérer laborieuse, il y'a donc un désir d'automatiser le processus.

Le but de ma présence au sein d'Aldebaran est donc de répondre à ce besoin. En s'appuyant sur l'utilisation de méthodes d'apprentissage automatique (Machine Learning), j'ai donc mis au point un algorithme capable de déterminer automatiquement (après une phase d'apprentissage) les causes ayant entrainées l'apparition d'anomalies sur Pepper. La mise au point de cet outil a été réalisé en trois temps :

1. auto-formation à l'apprentissage automatique et maitrise des outils de développement.
2. conception et développement de l'algorithme.
3. industrialisation du produit, c'est à dire en simplifier l'utilisation et le robustifier.

Ce rapport de stage présente les différentes recherches et effectuées, ainsi que les travaux de développement réalisés pour répondre au mieux à la problématique.

Sommaire

Table des figures

Liste des tableaux

Chapitre 1

Entreprise

1.1 Histoire

Aldebaran (anciennement Aldebaran Robotics) est une société Française de robotique humanoïde, fondée en 2005 par Bruno Maisonnier.

1.1.1 Le premier robot, Nao

Constituée au départ d'une équipe de douze collaborateurs, la toute jeune entreprise se fixe comme objectif de développer des robots humanoïdes et de les commercialiser au grand public, en tant que "nouvelle espèce bienveillante à l'égard des humains". Après trois années de recherche et développement, la société dévoile en 2008 son tout premier produit : Nao. La participation du robot humanoïde à divers évènements internationaux, comme par exemple la RoboCup ou encore l'Exposition Universelle de Shanghai en 2010 participe à sa popularisation auprès des laboratoires de recherche, des universités et des développeurs. Une seconde génération de robot Nao apparait en 2011 (dit Nao Next Gen). L'entreprise dévoile durant la même période un nouveau projet, en partenariat avec différents acteurs de la recherche, visant à créer un véritable robot d'assistance à la personne, Roméo.

1.1.2 La famille s'agrandit

Lors de l'année 2012, Aldebaran Robotics est racheté par SoftBank, société spécialisée dans le commerce électronique au Japon, et devient alors Aldebaran. Débute alors la conception d'un tout nouveau produit, le robot humanoïde Pepper. Dévoilé au grand public en 2014, il est dans un premier temps vendu au Japon auprès des entreprises. Les premiers clients à en bénéficier seront les magasins de téléphonie mobile du groupe SoftBank. Les ventes s'ouvrent dans un second temps aux particuliers Japonais. La société compte aujourd'hui plus de 400 collaborateurs et poursuit le développement de ses trois produits afin de les améliorer et de conquérir de nouveaux marchés (en Europe , en Chine et aux États-Unis).

1.2 Les produits

Aldebaran commercialise à ce jours deux produits : Nao et Pepper. Le robot Roméo est une plateforme de recherche.

1.2.1 Nao

Nao est un robot humanoïde de 58 cm de hauteur. Son public cible est principalement les laboratoires de recherche et le monde de l'éducation (allant des écoles primaires aux universités). Il est actuellement le produit le plus connu de l'entreprise auprès du grand public.

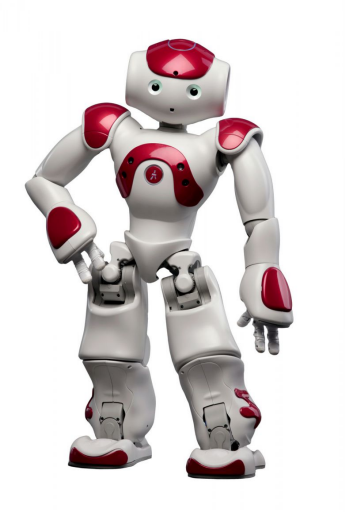


FIGURE 1.1 – Le robot humanoïde Nao.

Caractéristiques techniques Caractéristiques techniques de la dernière version de Nao (V5, Evolution). 1.1

Caractéristiques générales	
Dimensions	574 x 311 x 275 mm
masse	5,4 kg
Degrés de liberté	25
Processeur	Intel Atom Z530 1.6 GHz RAM : 1GB Mémoire flash : 2GB Micro SDHC : 8 GB
Système d'exploitation	Middleware Aldebaran NAOqi basé sur un noyau Linux
Connectivité	Wi-Fi, Ethernet, USB
Batterie	Autonomie : 90 minutes en usage normal Energie : 48.6 Wh
Vision	Deux caméras frontales 2D, 1220p, 30ips
Audio	Sortie : 2 haut-parleurs stéréo 4 microphones directionnels moteur de reconnaissance vocale Nuance
Capteurs	2 capteurs infra-rouges, résistance sensible à la pression, centrale inertielle, 2 systèmes sonars, 3 surfaces tactiles

TABLE 1.1 – Caractéristiques techniques de la dernière version commerciale de Nao [?]

1.2.2 Pepper

Dernier né d'Aldebaran, le robot Pepper est conçu pour vivre au côté des humains. Imaginé au départ pour accompagner et informer les clients dans les magasins de téléphonie du groupe japonais SoftBank, l'entreprise cherche à présent à placer son produit chez les particuliers. Le robot se base sur la structure software et hardware de Nao. Cependant, contrairement à son compagnon Nao, celui-ci se déplace non pas grâce à une paire de jambes, mais via trois roues omnidirectionnelles, facilitant son déplacement. A noter également que Pepper est équipé d'une tablette tactile sur son torse afin de faciliter les interactions Homme-Machine.

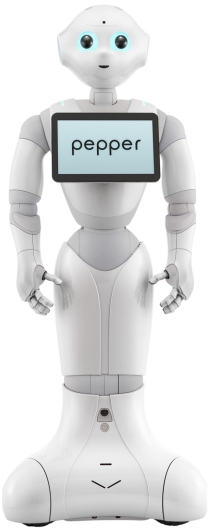


FIGURE 1.2 – Le robot humanoïde Pepper.

Caractéristiques techniques Caractéristiques techniques de la dernière version commerciale de Pepper (V1.7). 1.2

1.2.3 Roméo

Roméo est un nouveau type de robot d'accompagnement et d'assistance à la personne. Cette plateforme de recherche est soutenue par Aldebaran ainsi que d'autres partenaires universitaires et laboratoires de recherche (e.g. INRIA, LAAS-CNRS, ISIR, ENSTA, Telecom, etc.). Roméo est actuellement au stade de prototype et sert principalement de plateforme de tests pour les prochaines innovations majeures d'Aldebaran (e.g. les yeux mobiles, le système vestibulaire, etc.).

Caractéristiques techniques Caractéristiques techniques de la dernière version commerciale de Roméo (V2). 1.3

1.2.4 Le système d'exploitation NAOqi

NAOqi est le système d'exploitation commun aux robots d'Aldebaran. Il se base sur la distribution de Linux Gentoo et contient plusieurs API's afin de commander et contrôler les robots [**?**].

NAOqi Core : Gestion de l'ensemble des fonctions de base des robots (e.g. mémoire, "autonomous Life", comportements du robots, etc.).

Caractéristiques générales	
Dimensions	1210 x 480 x 425 mm
masse	28 kg
Degrés de liberté	17
Processeur	Intel Atom E3845 1.91 GHz RAM : 4 GB Mémoire flash : 8 GB MICRO SDHC : 16Go
Système d'exploitation	Middleware Aldebaran NAOqi, basé sur un noyau Linux
Connectivité	Wi-Fi, Ethernet, USB
Batterie	Énergie : 795 Wh
Vision	2 caméras 2D 1 caméra 3D
Audio	3 microphones directionnels moteur de reconnaissance vocale Nuance
Connectivité	Wi-Fi, Ethernet
Capteurs	6 lasers, 2 capteurs infra-rouges, 1 système sonar, résistance sensible à la pression, 2 centrales inertielles, 3 surfaces tactiles

TABLE 1.2 – Caractéristiques techniques de la dernière version commerciale de Pepper [**?**]

NAOqi Motion : Gestion des mouvements des robots.

NAOqi Audio : Gestion de la partie audio des robots.

NAOqi Vision : Gestion de la partie vidéo des robots

NAOqi People Perception : Ce module est utilisé pour étudier les personnes présentes autour du robot.

NAOqi Sensors : Gestion de l'ensemble des senseurs équipant les robots.

1.2.5 Plateforme de développement

Les robots sont fournis avec une plateforme de développement.

Choregraphe : Il s'agit d'un outil de programmation graphique basé sur une interface prenant la forme de schémas bloc [**?**]. Il permet de façon simple d'interagir avec le robot et de concevoir des applications pour le robot. Il comporte également un environnement de simulation 3D permettant aux développeurs de tester leurs applications sans même posséder un robot. Le logiciel permet également de disposer d'un retour visuel sur ce que le robot perçoit (e.g. vidéo issues des caméras, données des moteurs, etc.)

Kit de développement (SDK) : Il permet de développer des applications pour les robots via plusieurs langages de programmation : C++, Python et Java [**?**].

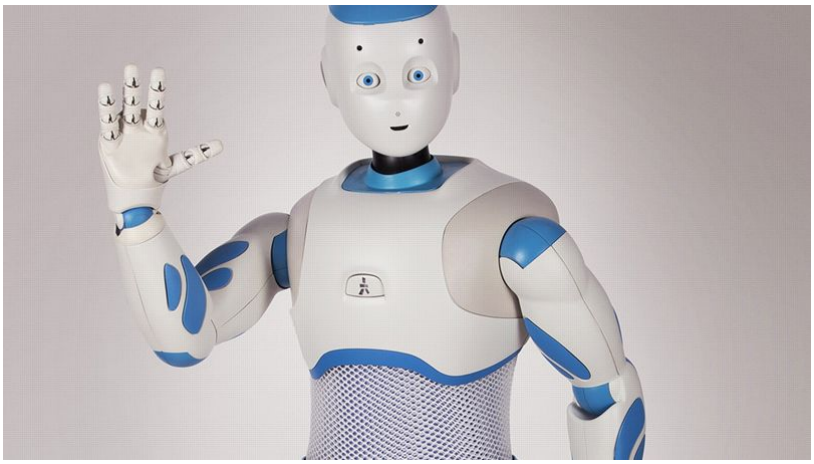


FIGURE 1.3 – Le robot humanoïde Roméo.

Caractéristiques générales	
Hauteur	1467 mm
masse	37 kg
Processeur	Intel ATOM Z530 1.6 GHz RAM : 1 GB Mémoire flash : 2 GB MICRO SDHC : 8 Go
Système d'exploitation	Middleware Aldebaran NAOqi, basé sur un noyau Linux
Connectivité	Wi-Fi, Ethernet
Batterie	Énergie : 795 Wh
Vision	4 caméras 2D 1 caméra 3D
Audio	3 microphones directionnels moteur de reconnaissance vocale Nuance
Connectivité	Wi-Fi, Ethernet
Capteurs	6 lasers, 2 capteurs infra-rouges, 1 système sonar, résistance sensible à la pression, 2 centrales inertielles, 3 surfaces tactiles

TABLE 1.3 – Caractéristiques techniques de la dernière version de Roméo [?]

Chapitre 2

Introduction

2.1 Présentation du produit

On présente ici de manière succincte l'architecture Hardware du robot Pepper afin de se familiariser avec les différents éléments du système avec lesquels nous serons susceptible de travailler durant la mise en œuvre de ce projet.

2.1.1 Les actionneurs

Les moteurs

Le robot Pepper est constitué de 20 moteurs dont il est possible de contrôler la position et la rigidité. 2.1

Tête : 2 moteurs pour les mouvements de lacet (HeadYaw) et de tangage (HeadPitch)

Bras : 4 moteurs par bras, répartis de la manière suivante :

- **Épaule :** 2 moteurs pour les mouvements de tangage (ShoulderPitch) et de roulis du bras (ShoulderRoll).
- **Coude :** 2 moteurs pour les mouvements de lacet(ElbowRoll) et de lacet (ElbowYaw) de l'avant-bras.

Main : 1 moteur pour le mouvement de lacet du poignet (WristYaw) et 1 pour le mouvement d'ouverture et de fermeture de la main (Hand).

Hanche : 2 moteurs pour le mouvement de roulis (HipRoll) et de tangage (HipPitch) du buste.

Genoux : 1 moteur pour le mouvement de tangage (KneePitch) du haut du corps.

Roues : 3 moteurs pour les mouvements en rotation de chacune des 3 roues omnidirectionnelles (WheelB, WheelFR et WheelFL).

Les Leds

Les leds placés sur les épaules de Pepper et autour de ces yeux permettent d'obtenir un certain nombre d'informations sur son état.

2.1.2 Les senseurs

Pepper comporte également une multitude de capteurs. Certains d'entre eux sont utilisés afin de s'assurer du bon comportement des parties électroniques et mécaniques du

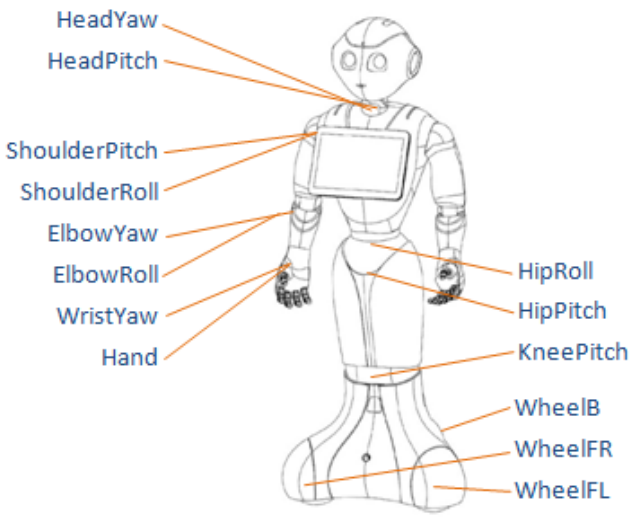


FIGURE 2.1 – Répartition des actionneurs de Pepper

robot, ou pour réaliser du contrôle-commande. D'autres senseurs sont en revanche intégrés au robot pour que l'utilisateur puisse interagir avec. 2.1

2.2 Expression du besoin

L'extension du marché visée par Aldebaran pour Pepper s'accompagne d'une montée en puissance de la production. Afin de la guider, des outils de vérification des produits en fin de ligne de production sont mis en place. Parmi eux, on retrouve le "Filtering Test"2.2 qui consiste à réaliser une série de tests durant six heures. Il vise notamment à stresser l'ensemble des parties mécaniques du robot afin de faire ressortir d'éventuelles erreurs. Si une anomalie survient lors du déroulement du test, les différentes données relatives à l'état des systèmes mécaniques et électroniques du robot sont enregistrées dans un fichier journal (e.g. température des fusibles, valeur des accéléromètres, etc.). Afin d'identifier les causes de l'apparition de problèmes sur Pepper, un certain nombre d'hypothèses sont émises à partir de l'étude des données en sortie du système.

2.2.1 Hiérarchisation des erreurs

Afin de gérer au mieux les anomalies, celles-ci sont hiérarchisées en deux catégories.

Error name Cela correspond à l'erreur visible, c'est à dire de la conséquence liée à une anomalie hardware ou software. Cela peut correspondre par exemple à la chute du robot.

root cause Il s'agit de l'anomalie en elle même, c'est à dire de la cause ayant entraîné l'apparition d'une "error name". Si l' "error name" est la chute d'un robot, la "root cause" peut par exemple correspondre à la détérioration d'un engrenage de la hanche.

En suivant la logique exprimée par ces définitions, une "error name" sera constituée d'une ou plusieurs "root cause".

Senseur	position	description
Capteurs liés au actionneurs	sur les moteurs	Chaque moteur du robot est lié à 3 senseurs donnant des informations sur la valeur du courant délivrée au moteur (A), la température du moteur (C) et la position du moteur.
Senseurs tactiles	1 sur chaque main, 1 sur la tête	Permet à l'utilisateur d'interagir avec le robot par le toucher.
Les boutons	1 sur le buste, 3 bumpers sur la base du robot	Les bumpers permettent au robot de détecter si il rencontre un obstacle à proximité directe. Le bouton du buste permet quant à lui de modifier le mode dans lequel est le robot (autonome, veille).
Centrale inertielle	1 dans le buste, 1 dans la base	Informe sur la position et l'orientation du robot, ainsi que les données qui en découlent (vitesse, accélération).
Sonars	2 sonars à l'avant et l'arrière de la base	Permet de détecter la présence d'un objet situé à partir de 65 cm du robot.
Capteurs batterie	Au niveau de la batterie	Renseigne sur le courant et la tension délivré, le pourcentage de charge et la température.
Capteurs infrarouges	2 sur la base	Permet de détecter la présence d'un objet situé entre à et 50 cm du robot.
Lasers	6 lasers sur la base du robot	Permet de détecter la présence d'un objet

TABLE 2.1 – Les différents capteurs de l'architecture sensorielle de Pepper

2.2.2 Exemple d'analyse d'une anomalie

Observation

- Lors du déroulement du test, le robot tombe à t = 16972 secondes, soit lorsque le robot réalise une séquence de mouvements particulière appelée "Heat Behavior". Les valeurs retournées par l'accéléromètre selon l'axe Z attestent de cette chute. 2.2
- On analyse alors les données liées aux systèmes mécaniques et électroniques du robot pouvant avoir une relation directe ou indirecte avec sa chute. Lorsque l'on étudie la vitesse de rotation du moteur de la hanche, on remarque qu'aux environs de t = 16970 secondes (c'est-à-dire 2 secondes avant la chute du robot), l'information fournie par le senseur ne suit plus la commande envoyée au moteur 2.3. On remarque également que le senseur du genou suit correctement la commande du moteur 2.4.
- On remarque également une augmentation anormale du courant dans le moteur de l'articulation.

Hypothèse émise

Lors de l'exécution de l'animation "Heat Behavior", le robot est amené à réaliser des mouvements amples au niveau de sa hanche, causant un certain stress sur cette partie mécanique. Lorsque l'engrenage de la hanche arrive près de sa butée mécanique, celui-ci ne parvient pas à atteindre sa position zéro. Ce phénomène occasionne une augmentation du courant délivré dans le moteur de l'articulation, ce qui entraine le passage en mode protégé du robot, ce qui désactive sa rigidité. Sans cette rigidité, le robot tombe ("error name").

5 étapes successives sur 6 heures de test			
Activité	Durée (en minutes)	Temps cumulé (en minutes)	Description
Test 1 : succession de divers actions effectué 40 fois durant les 10 heures			
Intro, Danse, Outro	6 à 7	6 à 7	Réalisation des mouvements d'introduction et d'outro
Cycles de WakeUp, Rest	3 à 4	10	Passage successif en mode autonome et veille
Faux dialogues	2	12	Réalisation des mouvements effectués lors de dialogues
Rest	3	15	Réalisation des mouvements lors du passage en mode Rest
Test 12 : enchainement de 8 danses répété 5 fois durant les dix heures			

Table 2.2 – Déroulement d'un Filtering test

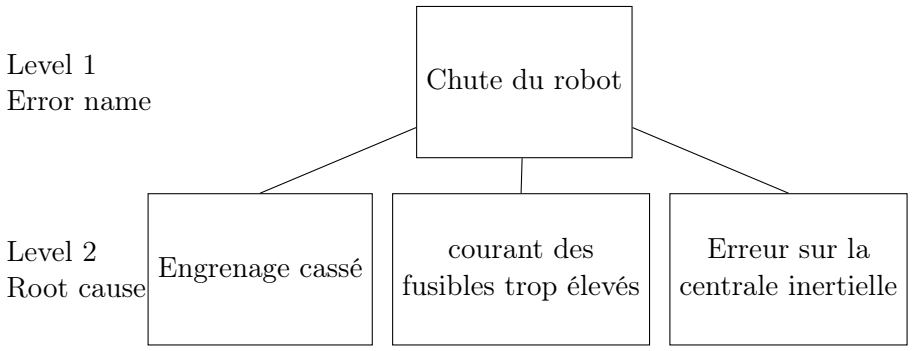


Table 2.3 – Exemple d'un error name et ses root cause

Une étude plus poussée nous apprendra que la "root cause" du problème correspondait à une dent cassée au niveau de l'engrenage de la hanche.

2.3 Solution proposée

De part la quantité d'informations à analyser, cette tâche d'analyse peut rapidement devenir rébarbative, d'où le souhait d'automatiser ce processus d'investigation. De part la variabilité des types de données, on ne peut réduire pas le nombre d'informations à analyser à de simples caractéristiques communes (e.g. moyenne, écart type, etc.). On s'appuiera donc sur des approches algorithmiques plus poussées en utilisant notamment des méthodes d'apprentissages automatiques (plus connues sous le terme anglais de Machine Learning). La multiplicité des modèles englobés dans cette discipline nous permettra de répondre au mieux à la problématique.

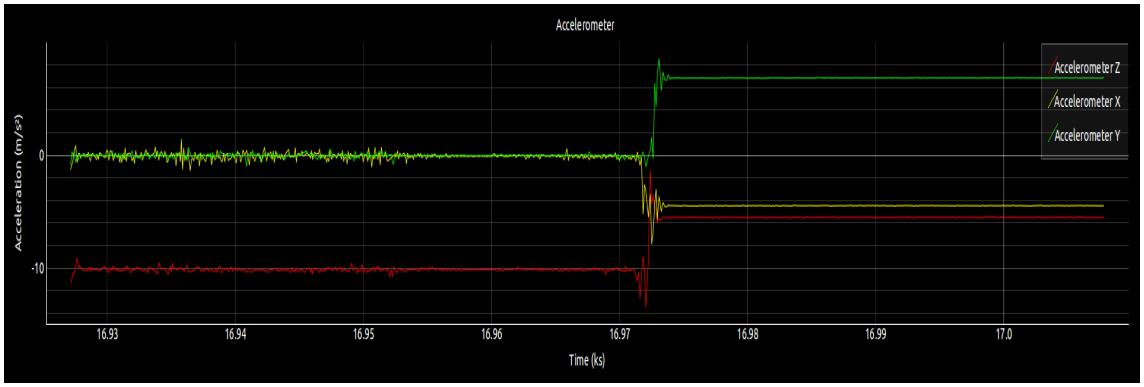


FIGURE 2.2 – Analyse d'une anomalie : accéléromètre

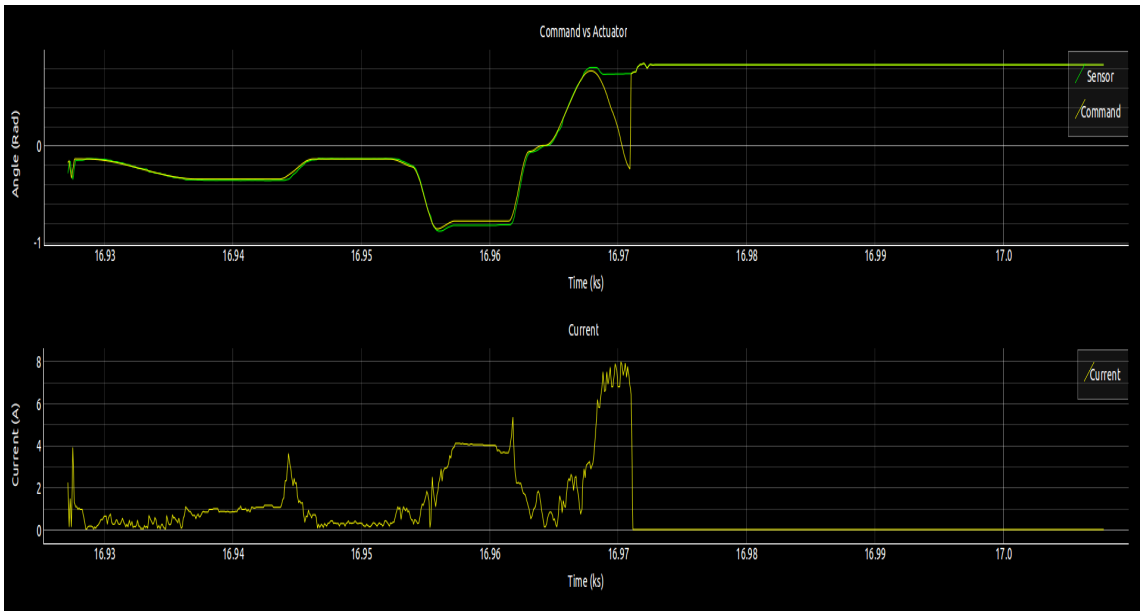


FIGURE 2.3 – Analyse d'une anomalie : les genoux

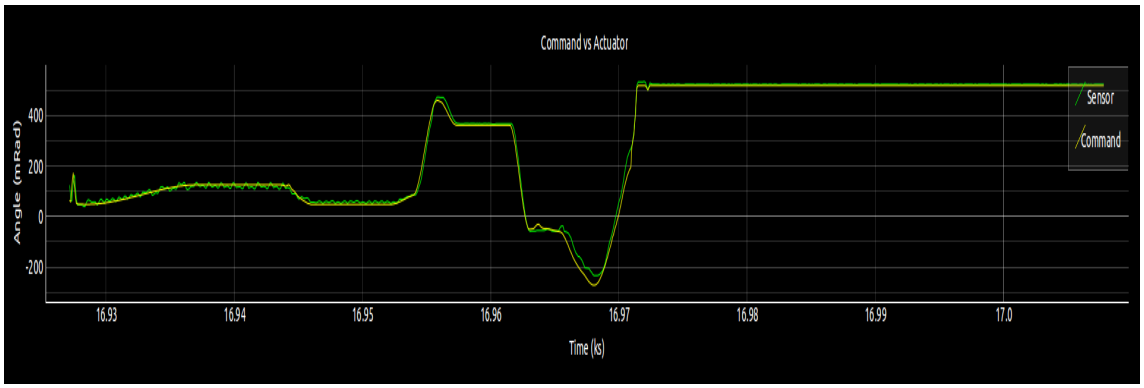


FIGURE 2.4 – Analyse d'une anomalie : la hanche

Chapitre 3

Le Machine Learning

3.1 Généralités sur le Machine Learning

Le Machine Learning (traduire par apprentissage automatique) est une ramification de l'intelligence artificielle. Cependant, son statut de subdivision n'informe en rien sur l'étendu des notions contenues dans cette matière scientifique. Son champ d'étude est vaste et en perpétuelle évolution. Les solutions offertes par cette discipline permettent d'étudier toute sortes de données et d'automatiser une multitude de systèmes. L'apprentissage automatique rencontre un succès croissant qui est corrélé avec l'essor des nouvelles technologies et l'automatisation de l'analyse de volumes conséquents de données utilisateurs (Big Data). Les applications sont donc multiples. En voici quelques exemples :

- Algorithmes des moteurs de recherches (Google Deep Dream, Google TensorFlow)
- Analyse boursière
- Analyse de rapports d'erreurs
- Reconnaissance vocale, biométrie, reconnaissance d'écriture
- Robotique (vision, mouvements, prise de décision, etc.)
- Neurosciences

3.1.1 Définition et principe général du Machine Learning

Le champ d'étude et d'application du Machine Learning étant immense, on propose de redéfinir cette notion en l'adaptant à la résolution de notre problématique (i.e. automatiser l'analyse d'incidents révélés lors du filtering test). On offre ici deux définitions de l'apprentissage automatique : une première dite "High Level" qui le caractérise de manière générale et une seconde qui reflète sa dimension algorithmique.

High Level : Le Machine Learning permet à un système d'évoluer grâce à un processus d'apprentissage et ainsi de remplir des tâches qu'il est difficile, voir impossible, de remplir par d'autres moyens algorithmiques plus classiques.

Mathématique Le Machine Learning fourni les outils pour prédire une/des donnée(s) de sortie Y à partir des données d'entrée X via un processus d'apprentissage.

De nombreuses autres définitions existent, mais elles ne correspondent pas à la dimension recherchée dans le cadre de ce projet.

Au regard des deux définitions stipulées ci-dessus, on peut définir le principe de base de l'apprentissage automatique sous la forme d'un schéma bloc reffig :Schéma fonctionnel haut niveau du Machine Learning.

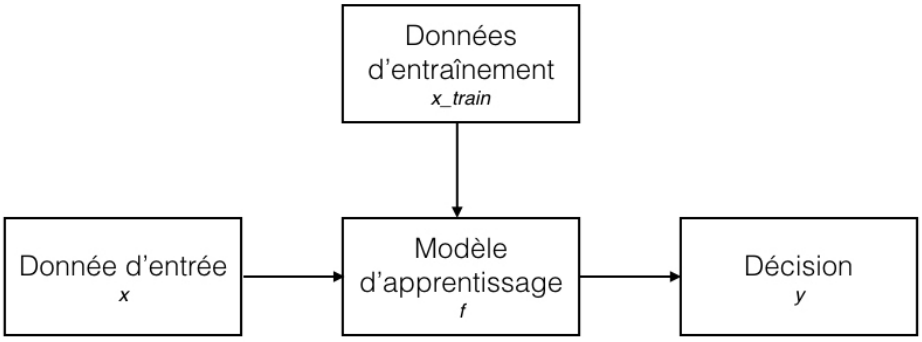


FIGURE 3.1 – Schéma fonctionnel haut niveau du Machine Learning

L'apprentissage automatique peut donc être vu dans sa globalité comme un processus composé de deux étapes successives :

Apprentissage (a) Un ensemble de données est présenté au système (X_train).

(b) A partir de ces informations, le système (f) apprend - s'entraîne- afin d'être par la suite en capacité de prendre décision vis à vis de la tâche qui lui sera demandée.

Prise de décision (a) On a en entrée du système une ou des donnée(s) brutes (X).

(b) Cette donnée est traitée et analysée par le système.

(c) En sortie, une décision est prise quant à la tâche demandée (Y).

Un exemple concret

Afin de présenter de manière plus concrète le processus fonctionnel haut niveau d'un algorithme d'apprentissage, on présente l'exemple suivant :

On cherche à déterminer à quelle période de l'année nous nous trouvons (i.e. printemps, été, automne ou hiver) à partir de l'humidité, la température et la pression atmosphérique d'aujourd'hui.
La première étape de notre processus sera donc d'entrainer notre système afin que celui-ci soit en mesure de prendre une décision vis à vis des données qu'on lui présentera en entrée (i.e. l'humidité, la température et la pression atmosphérique d'aujourd'hui).
Une fois le système entrainé, on attend que celui-ci ai ce type de comportement :
On présente en entrée de mon système une température de -2 degrés, une pression atmosphérique de 1030hPa et un taux d'humidité de 81%. La réponse attendue en sortie du système est : hiver

On peut adapter le schéma fonctionnel haut niveau du Machine Learning à notre exemple 3.2.

Pour caractériser et désigner plus précisément les différents éléments de notre système, on présente ci-dessous le champs lexical utilisé dans le domaine du Machine Learning :

Lexique

les features Le type de données présenté en entrée.
La température, la pression atmosphérique et l'humidité.

échantillons ou exemples Les données permettant d'entrainer le système (x_train).
De nombreux échantillons de température, pression atmosphériques et humidité pris à différents périodes de l'année, sur plusieurs années.

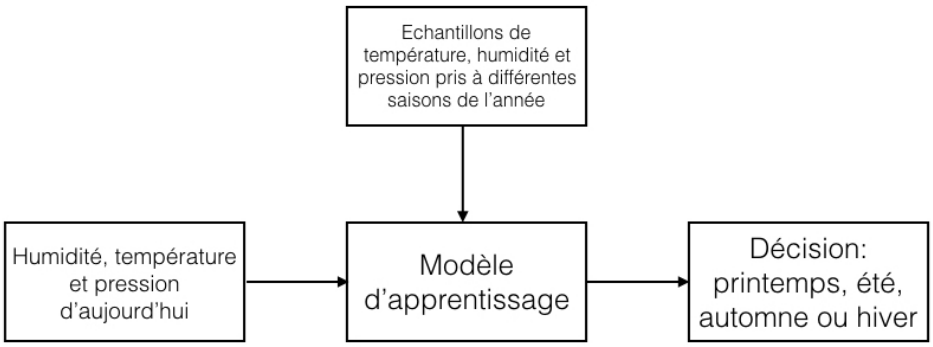


FIGURE 3.2 – Schéma fonctionnel haut niveau du Machine Learning, l'exemple de la prévision saisonnière

le modèle d'apprentissage Le cœur du système décisionnel (f).

La décision La sortie ou réponse du système (Y)

Printemps, été, automne ou hiver.

3.1.2 Les exemples

Les exemples correspondent aux données utilisées pour entraîner mon algorithme d'apprentissage. On parle également d'échantillons. Celles-ci sont regroupées en "features" (terme anglais, traduire par caractéristiques). Pour reprendre l'exemple cité précédemment 3.1.1, nos données sont regroupées en 3 features : la température, l'humidité et la pression atmosphérique. On données sont donc structurées de la manière suivante :

$$\begin{array}{c} \\ Exemple_1 \\ Exemple_2 \\ Exemple_3 \\ ... \\ Exemple_n \end{array} \begin{array}{c} \begin{array}{ccc} temprature(^{\circ}C) & humidite(\%) & pression(HPa) \end{array} \\ \begin{pmatrix} -10 & 85 & 1023 \\ 15 & 80 & 1020 \\ 23 & 65 & 1015 \\ ... & ... & ... \\ 10 & 81 & 1032 \end{pmatrix} \end{array} \tag{3.1}$$

Différents types de données

Il existe deux types de données : les données labellisées et non labellisées.

— Les données labellisées correspondent à des exemples corrélés à une sortie - un label - connue.

— Les données non labellisées ne sont quant à elles pas associées à une sortie.

Pour reprendre l'exemple précédent 3.1.1, on a les jeux de données suivants :

Données labellisées

$$\begin{array}{c} \\ Exemple_1 \\ Exemple_2 \\ Exemple_3 \\ ... \\ Exemple_n \end{array} \begin{array}{c} \begin{array}{ccc} temprature(^{\circ}C) & humidite(\%) & pression(HPa) \end{array} \\ \begin{pmatrix} -10 & 85 & 1023 \\ 15 & 80 & 1020 \\ 23 & 65 & 1015 \\ ... & ... & ... \\ 10 & 81 & 1032 \end{pmatrix} \end{array} \begin{array}{c} \\ hiver \\ automne \\ ete \\ \\ printemps \end{array} \tag{3.2}$$

On connait la sortie qui correspond aux données d'entrée, i.e. que on sait à quelle période de l'année les échantillons ont été prélevé.

Données non labellisées

$$\begin{array}{c} \\ Exemple_1 \\ Exemple_2 \\ Exemple_3 \\ ... \\ Exemple_n \end{array} \begin{array}{c} temprature(^{\circ}C) \\ \left(\begin{array}{c} -10 \\ 15 \\ 23 \\ ... \\ 10 \end{array} \right. \end{array} \begin{array}{c} humidite(\%) \\ \begin{array}{c} 85 \\ 80 \\ 65 \\ ... \\ 81 \end{array} \end{array} \begin{array}{c} pression(HPa) \\ \left. \begin{array}{c} 1023 \\ 1020 \\ 1015 \\ ... \\ 1032 \end{array} \right) \end{array} \begin{array}{c} \\ ?? \\ ?? \\ ?? \\ \\ ?? \end{array} \quad (3.3)$$

On ne connait pas la sortie qui correspond aux données d'entrée, i.e. que on *ne* sait *pas* à quelle période de l'année les échantillons ont été prélevé.

3.1.3 La décision

La sortie de notre système peut être également nommé décision. Toujours selon notre exemple 3.1.1, cela correspond au choix fait par l'algorithme entre les différentes saisons : printemps, été, automne et hiver.

Différents types de sorties

Il existe différents types de sortie : les sorties continues et discrètes.

Les sorties continues peuvent prendre n'importe quelle valeur.
$y \in R$
Déterminer l'évolution de la température en fonction des échantillons enregistrés les mois précédents correspond à une sortie continue.

Les sorties discrètes ne peuvent prendre que des valeurs prédéterminées.
$y \in 1, 2, 3, ..., C$
L'exemple 3.1.1 a une sortie discrète. En effet, la sortie ne peut prendre que des valeurs prédéterminées : printemps, été, automne et hiver.

3.1.4 Le modèle

Il existe différents types d'apprentissages. Le choix d'un modèle en particulier est influencé par le type d'exemples que l'on a en entrée du système et du type de décision que l'on souhaite obtenir en sortie. Nous nous intéresserons à différentes catégories d'apprentissages automatiques :

— Les apprentissages supervisés et non-supervisés. Cette caractéristique dépend du type d'exemples que l'on a en entrée du système.
— Les régression ou les classifications. Cette caractéristique dépend du type de sortie que l'on souhaite obtenir .

apprentissage supervisé et non supervisé

L'apprentissage supervisé nécessite d'avoir des données labellisées en entrée, i.e. que l'on connait le type de décision que l'on aura en sortie du système en fonction des exemples en entrée : il y'a une corrélation entre la sortie et l'entrée. C'est cette notion qui s'exprime au travers du terme *supervisé*. L'apprentissage non supervisé s'appuie quant à lui sur l'utilisation d'une base de donnée non labellisée pour son apprentissage, i.e qu'on ne connait pas le type de décision associé aux exemples en entrée. Dans le cas d'un apprentissage non supervisé, pour parvenir à prendre une décision, l'algorithme devra diviser ce groupes de

données hétérogènes en sous-groupes homogènes d'informations similaires. On appelle ces subdivisions des *clusters*. Afin de matérialiser les différences entres les deux méthodes et les applications possibles pour chacune d'elles, on propose deux exemples 3.1.

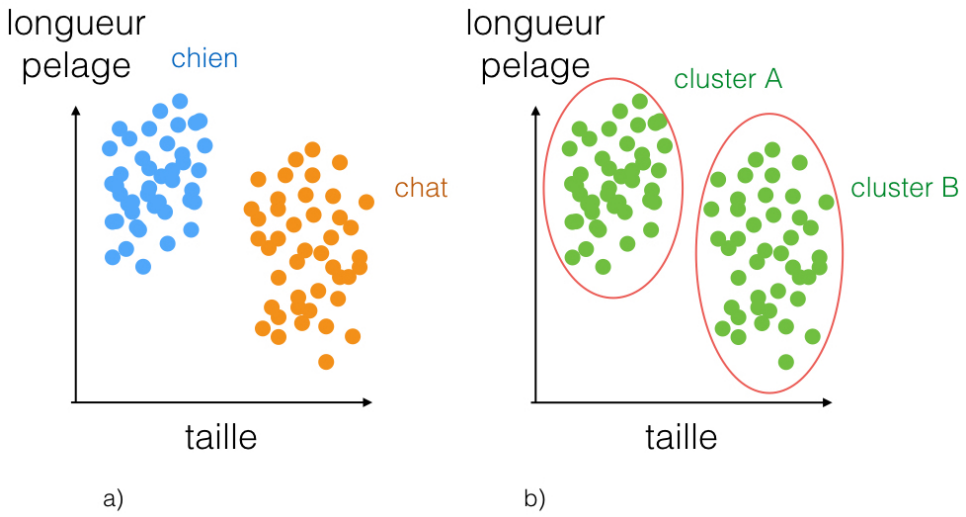


FIGURE 3.3 – Comparaison d'un apprentissage supervisé et non supervisé dans le cadre de l'exemple "apprendre aux humains". On observe sur la figure a) les différents exemples d'entrainement exprimés dans un repère composé des deux features "taille" et "longueur de pelage". On remarque qu'il y'a la formation de deux groupes de données homogènes : les animaux de taille globalement élevée avec un pelage court et les animaux de taille moindre avec un poil globalement plus long. Les données étant labellisées, on sait que le premier groupe correspond à des chiens et le deuxième à des chats.

Il existe deux types d'apprentissage supervisé : la régression et la classification.

Apprentissage supervisé : régression et classification

La régression est un type d'apprentissage avec lequel on souhaite obtenir une sortie continue. Dit de manière différente, la régression implique que l'on souhaite *estimer* ou *prédire* une réponse. La classification est quant à elle un type d'apprentissage avec lequel on souhaite obtenir une sortie discrète, elle peut être vu comme un cas particulier de la régression où les valeurs à prédire sont discrètes. Formulé autrement, la classification implique que l'on souhaite *classer* un exemple parmi différentes catégories. Afin de représenter concrètement les nuances entre les deux méthodes et les applications possibles pour chacune d'elle, on propose l'exemple tableau 3.2).

3.2 Les différents algorithmes d'apprentissage supervisé

Il existe différents algorithmes d'apprentissage supervisé utilisés pour résoudre des problèmes de régression et de classification.

3.2.1 La régression linéaire uni-variable

La régression linéaire cherche à expliquer une variable de sortie y par une fonction affine de x. Cette fonction linéaire affine est appelée *hypothèse* (notée $h(x)$). Exprimé autrement, on a un jeu de données y auquel correspond un jeu de données y, on cherche les valeurs θ_1

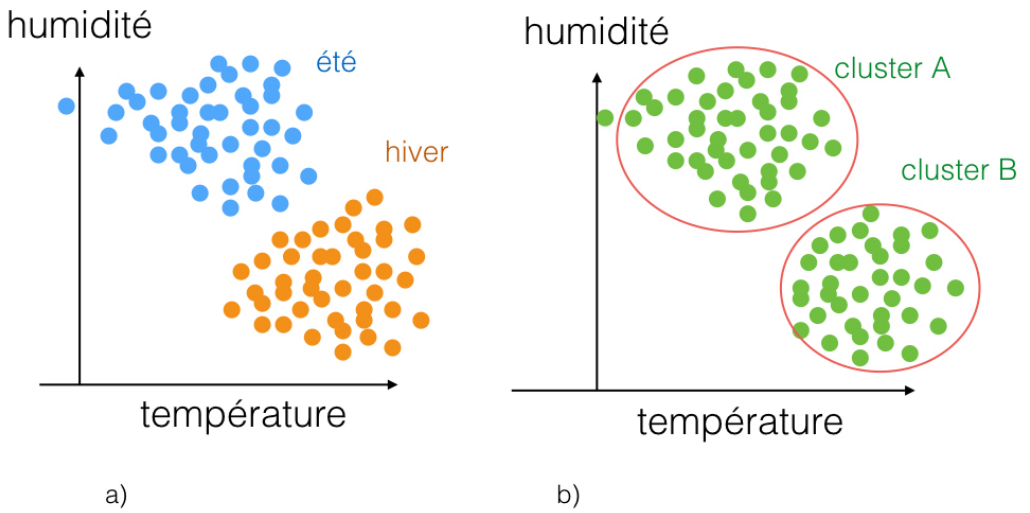


FIGURE 3.4 – Comparaison d'un apprentissage supervisé et non supervisé dans le cadre de l'exemple "prévisions saisonnières". On observe sur la figure a) les différents exemples d'entrainement exprimés dans un repère composé des deux dimensions (features) "humidité" et "température". On remarque qu'il y'a la formation de deux groupes de données homogènes : un où les échantillons sont pris lors de saisons globalement chaudes et sèches etc un autre où les échantillons sont enregistrés lors de périodes globalement froides et humides. Les données étant labellisées, on sait que le premier groupe correspond à l'été et l'autre groupe l'hiver.

et θ_2 permettant de "mapper" les données, tel que :

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{3.4}$$

Exemple de régression linéaire uni-variable

On souhaite déterminer le prix d'un logement en fonction de sa surface au sol en se basant sur les exemples de prix connus du parc immobilier. La surface au sol est donc l'entrée x de notre système et le prix la sortie y. Soit le tableau 3.3 les exemples x, on obtient la représentation graphique en figure 3.6. Grâce à l'expression de l'hypothèse, on est capable de déterminer le prix d'un loyer en fonction de la surface au sol.

Cet exemple est dit uni-variable car un seul jeu de données x (superficie) correspond à un jeu de données y (prix).

La fonction coût

La fonction coût (en anglais cost function) correspond à la comparaison de la moyenne des différences entre les résultats de l'hypothèse $h(x)$ avec les entrées x et les sorties actuelles y. Cela signifie qu'elle cherche à minimiser les valeurs calculées via l'hypothèse et les valeurs réelles (figure 3.7). Soit :

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 \tag{3.5}$$

avec :
— J la fonction coût

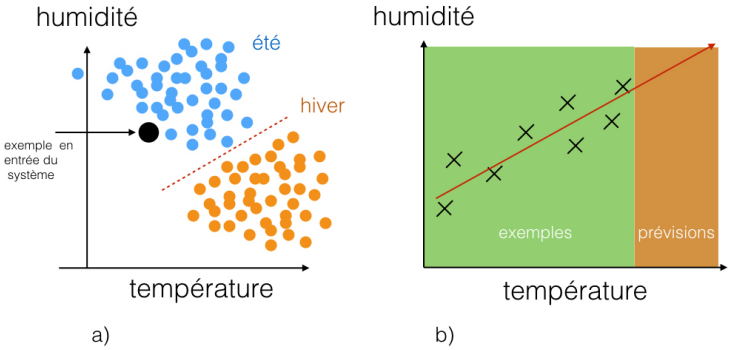


FIGURE 3.5 – Comparaison par l'exemple de la régression et la classification. Sur la figure a), on observe deux jeux de données homogènes (été et hiver) s'exprimant dans un repères en deux dimensions (features humidité et température). Le but est ici de classer la donnée en entrée du système parmi ces deux ensembles, il s'agit donc d'un problème de classification. Dans la figure b), on observe une succession d'exemples exprimés dans un repère en deux dimensions (features température et humidité). On remarque une évolution globalement linéaire de ces exemples, nous permettant ainsi de prédire la température et l'humidité sur les prochains mois : il s'agit d'un problème de régression.

— θ_0 et θ_1 les paramètres de l'hypothèse $h(x)$
— m le nombre d'exemples disponibles pour l'entrainement
— $h_\theta(x)$ l'hypothèse $h_\theta(x) = \theta_0 + \theta_1 x$
— x^i exemple i
— y^i sortie i

Algorithme du gradient

On cherche donc à minimiser la valeur de la fonction coût $J(\theta_0, \theta_1)$ en jouant sur la valeur des paramètres θ_0 et θ_1 de l'hypothèse. Pour cela, on calcule la fonction coût pour différentes valeurs de θ_0 et θ_1 et on cherche la valeur minimale de $J(\theta_0, \theta_1)$ (figure **??**). Dans l'idée, cela revient à choisir une valeur de θ et de "faire un pas" vers la direction la plus basse, il s'agit donc d'un calcul itératif. Mathématiquement, trouver le minimum peut être effectué en appliquant l'algorithme du gradient (en anglais, gradient descent), tel que :

$$\begin{aligned} \theta_0 &= \theta_0 - \alpha \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} \\ \theta_1 &= \theta_1 - \alpha \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} \end{aligned} \tag{3.6}$$

avec α la taille du "pas" que l'on fait vers la direction la plus basse. Si la valeur de α est trop petite, le nombre d'itération sera plus important, augmentant ainsi le temps de calcul. Si la valeur de α est trop grande, on risque de ne pas atteindre le minimum et de diverger.

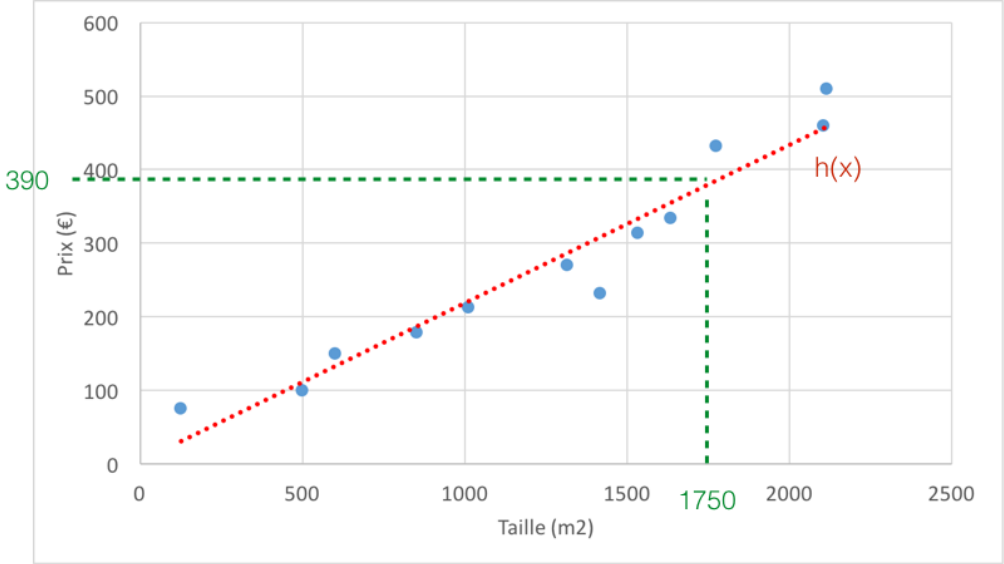


FIGURE 3.6 – évolution du prix de l'immobilier en fonction de la surface. L'ensemble des données semble globalement évoluer de manière linéaire. Cette linéarité est représentée par l'hypothèse $h(x)$. Grâce à celle-ci, on peut déterminer le prix d'un logement en fonction de sa superficie, et inversement. Par exemple, un appartement d'une surface de 1750 m^2 coutera aux alentours de 390€.

3.2.2 La régression linéaire multi-variable

On peut également réaliser de la régression linéaire avec plusieurs variables en entrée.

Exemple de régression linéaire multi-variable

Afin de montrer ce qu'est la régression linéaire multi-variable, on ajuste l'exemple 3.2.1 pour correspondre à un problème multi-variable.

On souhaite déterminer le prix d'un logement en fonction de sa surface au sol, du nombre de pièces et du nombre d'étages en se basant sur les exemples de prix connus du parc immobilier. On a cette fois ci plusieurs entrées (surface, nombre de pièces, nombre d'étages). Soit le tableau **??** des exemples x 3.4

La fonction coût s'exprime donc sous cette forme :

$$h_\theta(x) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 \tag{3.7}$$

Où X_1, X_2 et X_3 correspondent respectivement aux features surface, nombre d'étages et nombre de pièces.

On peut généraliser la fonction coût :

$$h_\theta(x) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 + ... + \theta_n X_n \tag{3.8}$$

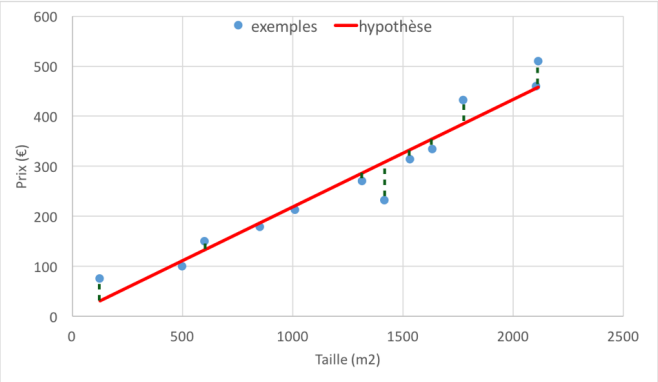


FIGURE 3.7 – Régression linéaire, optimisation de la fonction coût. les valeurs calculées via l'hypothèse et les valeurs réelles. Cela revient à minimiser la distance entre un exemple et l'hypothèse.

Généralisation de la fonction coût

On peut généraliser la fonction coût utilisée lors de la régression linéaire uni-variable pour l'appliquer à un problème de régression linéaire multi-variable, tel que :

$$J(\theta_1, \theta_2, \theta_3, ..., \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 \tag{3.9}$$

Généralisation de l'algorithme du gradient

On peut généraliser l'algorithme du gradient utilisé lors de la régression linéaire uni-variable pour l'appliquer à un problème de régression linéaire multi-variable, tel que :

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta_0, \theta_1, \theta_2, \theta_3, ..., \theta_n)}{\partial \theta_j} \tag{3.10}$$

Où l'on met à jour simultanément l'ensemble des valeurs de θ lors de chaque itération.

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{m}^{i=1} (h_\theta(x^i) - y^i) x_j^i \tag{3.11}$$

avec x_j^i l'entrée j de l'exemple i.

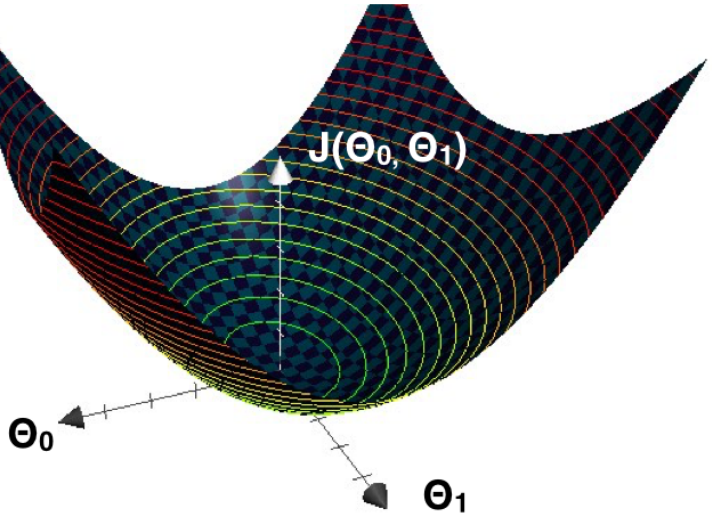


FIGURE 3.8 – Régression linéaire, calcul des paramètres de l'hypothèse. Ce graphique représente la variation des paramètres θ_0 et θ_1 de l'hypothèse en fonction de la valeur de la fonction coût. On cherche les valeurs θ_0 et θ_1 minimisant$J(\theta_0, \theta_1)$. La courbe ayant la forme d'une cuvette, les valeurs optimales de θ_0 et θ_1 correspondent à celles au fond de la cuvette.

Vectorisation des calculs

On peut écrire les exemples en entrée du système sous forme matricielle :

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ ... \\ X_n \end{bmatrix} = X \tag{3.12}$$

De même, on exprime les paramètres θ de l'hypothèse sous forme matricielle :

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ ... \\ \theta_n \end{bmatrix} = \theta \tag{3.13}$$

On peut alors réécrire l'hypothèse (3.8) de la façon suivante :

$$h_\theta(x) = \theta^T X \tag{3.14}$$

L'algorithme du gradient (3.11) s'exprime donc sous cette forme :

$$\theta = \theta - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) X^i \tag{3.15}$$

3.2.3 La régression logistique

La régression logistique permet de résoudre des problèmes de classification. Elle découle de la régression linéaire et s'appuie sur les mêmes concepts. On cherche à séparer des

groupes de données homogènes dans un ensemble hétérogène.

Régression linéaire et classification

On reprend l'exemple de la prévision saisonnière 3.1.1 et on lui applique de la régression linéaire afin de classer automatiquement l'échantillon qu'on lui présente en entrée dans la classe printemps ou hiver en fonction de la température (figure 3.9). Pour cela, on met en place un seuil de classification :

— Si $h_\theta(x) > 0,5$ alors $y = 1$
— Si $h_\theta(x) < 0,5$ alors $y = 0$

Où $y = 0$ correspond à la période hiver et $y = 1$ l' été.

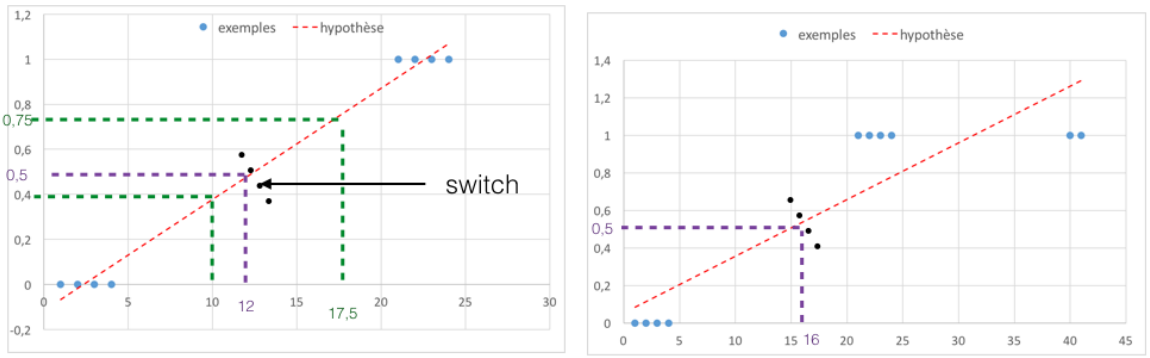


FIGURE 3.9 – Classification via régression linéaire. On observe sur la figure a) que lorsque la température est inférieure à 12,5 °, le système considère que la valeur de sortie est 0 (hiver). Lorsque la température est au dessus, la valeur de sortie devient 1 (été). Ce cas particulier fonctionne car les exemples sont répartis de manière uniforme. Or, dans le cas de la figure b), on remarque que une répartition non uniforme des données entraîne un dysfonctionnement du classement. En effet, le seuil de différentiation entre les deux classes se situe aux alentours de 15 °ce qui n'est pas représentatif des exemples.

Cette méthode fonctionne dans ce cas ci car les exemples sont régulièrement espacés entres eux. Si ils ne l'étaient pas, le seuil serait alors mal placé et la classification serait erronée. La régression linéaire n'est donc pas adaptée pour la classification.

Régression logistique et fonction sigmoïde

Afin de classer des données à partir d'une régression, on modifie la forme linéaire de l'hypothèse utilisée lors de la régression linéaire en une sigmoïde (figure 3.10). On l'appelle également fonction logistique ou de répartition. Du point de vu des probabilités, cela signifie que l'hypothèse sous cette forme représente la probabilité estimée que la sortie y soit égale à 1. Par exemple, si $h_\theta(x) = 0,7$ signifie que l'on a 70% de chance que la variable de sortie soit égale à 1.

soit $g(x) = \frac{1}{1+e^{-z}}$ la fonction sigmoïde, on a l'hypothèse $h_\theta(x)$ suivante :

$$h_\theta(x) = g(\theta^T x) \tag{3.16}$$

soit :

$$h_\theta(x) = \frac{1}{1 + e^{\theta^T x}} \tag{3.17}$$

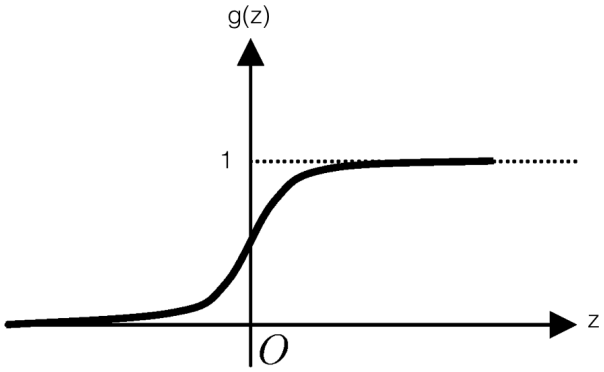


FIGURE 3.10 – Fonction sigmoïde, également appelée fonction logistique.

Ligne de décision

On a la propriété de la fonction sigmoïde suivante :

$$g(z) \geq 0,5 \text{ lorsque } z \geq 0 \tag{3.18}$$

On a donc pour l'hypothèse la propriété suivante :

$$h(x) = g(\theta^T x) \geq 0,5 \text{ lorsque } \theta^T x \geq 0 \tag{3.19}$$

On peut alors émettre les deux affirmations suivantes :

$$\begin{aligned} y = 1 \text{ si } h_\theta(x) \geq 0,5 \text{ soit } \theta^T x \geq 0 \\ y = 1 \text{ si } h_\theta(x) \leq 0,5 \text{ soit } \theta^T x \leq 0 \end{aligned} \tag{3.20}$$

application : Soit $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ avec $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$ on a la condition suivante :

$$y = 1 \text{ si } -3 + x_1 + x_2 \geq 0 \text{ i.e.} x_1 + x_2 \geq 3 \tag{3.21}$$

Cette expression correspond à la ligne (ou frontière) de décision de la régression linéaire (figure 3.11).

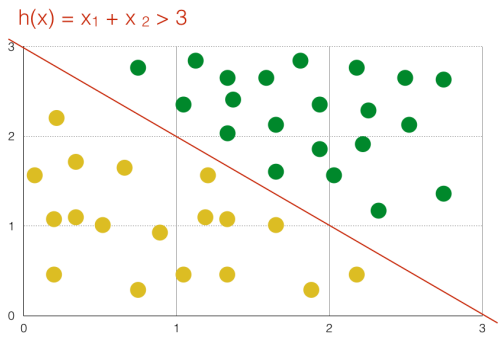


FIGURE 3.11 – Exemple d'une regression logistique

Ligne de décision non linéaire

Il est possible d'utiliser des hypothèses plus complexes. Par exemple, on a $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_1 x_1^2 + \theta_1 x_2^2)$ avec $\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$. Cela revient donc à dire :

$$\begin{aligned} y = 1 \text{ si } & -1 + x_1^2 + x_2^2 \geq 0 \\ & soit x_1^2 + x_2^2 \geq 1 \end{aligned} \tag{3.22}$$

Cette frontière de décision correspond à l'équation d'un cercle. Plus le nombre de polynômes composant l'hypothèse est important, plus on peut créer des lignes de décisions complexes.

Fonction coût

3.2.4 SVM -Support Vector Machine-

3.2.5 Comparaison des algorithmes

	apprentissage supervisé	apprentissage non supervisé
exemple n°1 : apprendre aux humains	Une institutrice souhaite apprendre à ses élèves à différencier un chat d'un chien : c'est la décision qu'on attend d'eux. Pour ce faire, l'éducatrice leur montre différentes photographies de chiens et de chats : ce sont les exemples utilisés pour l'apprentissage. Ces exemples peuvent être segmentés en différentes caractéristiques, comme la taille de l'animal, sa couleur, la longueur du poil, etc : il s'agit des features. Lorsque que l'institutrice leur présente les différentes images, elle stipule clairement si il s'agit d'un chien ou d'un chat : il y'a donc une corrélation entre l'entrée et la sortie de l'apprentissage, il s'agit d'un apprentissage supervisé (figure 3.3,a).	On retrouve cette même institutrice donnant à ses élèves le même exercice à la différence que, contrairement à précédemment, lorsque qu'elle présente les différentes images, elle *ne* stipule *pas* la race de l'animal : il n'y a donc aucune corrélation entre l'entrée et la sortie de l'apprentissage, il s'agit donc d'un apprentissage non supervisé. Pour réussir cet exercice, les enfants devront donc regrouper les animaux en s'appuyant sur leurs similitudes physiques, i.e. leurs features (e.g. taille de l'animal, sa couleur, longueur du poil, etc.). Les élèves ne connaitrons certes pas le nom des deux animaux, mais ils auront su les différencier. C'est la même approche qui est réalisé en apprentissage automatique non supervisé (figure 3.3, b).
exemple n°2 : prévisions saisonnières 3.1.1	On reprend l'exemple dans lequel on souhaite prendre une décision quant à la période de l'année à laquelle on se trouve actuellement, en fonction des features humidité, température et pression atmosphérique. Pour se faire, on prélève des échantillons à différentes périodes de l'année en notant à quelle saison ces données ont été prélevées : se sont des exemples labellisées, i.e. il y'a une corrélation entre les données et la sortie du système. Il s'agit donc d'un apprentissage supervisé (figure 3.4, a).	On reprend le même problème mais cette fois-ci on ne note pas la saison à laquelle les échantillons ont été prélevé. Pour résoudre le problème, l'algorithme doit donc associer les données les plus similaires entre elles et ainsi créer des groupes homogènes d'informations qui correspondront aux 4 décisions possibles (figure 3.4, b).

TABLE 3.1 – Comparaison de l'apprentissage supervisé et non supervisé par des exemples

régression	classification
On souhaite connaitre le temps (température et humidité) qu'il fera pendant les jours suivants. Pour cela, on s'appuie sur les différents échantillons de température et d'humidité enregistrés lors des mois et des années précédentes (exemples labellisés). Le fait de déterminer la température des jours suivants relève de la prédiction (figure 3.2, a).	L'exemple de la prédiction saisonnière 3.1.1 est un problème de classification : on cherche à classer notre donnée d'entrée parmi plusieurs groupes de données homogènes : printemps, été automne ou hiver (figure 3.2, b).

TABLE 3.2 – Comparaison entre l'apprentissage supervisé de type régression et supervisé de type classification

taille (m^2)	prix (€)
2104	460
1416	232
1534	314
852	178
500	100
1012	212
126	75
1775	432
600	150
2114	510
1316	270
1634	334

TABLE 3.3 – exemples du prix des logements en fonction de leur taille

taille (m^2)	prix (€)
2104	460
1416	232
1534	314
852	178
500	100
1012	212
126	75
1775	432
600	150
2114	510
1316	270
1634	334

TABLE 3.4 – exemples du prix des logements en fonction de leur surface, du nombre d'étages et du nombre de pièces

Chapitre 4

Utilisation du Machine Learning pour l'analyse d'incidents

Au regard des recherches exposées précédemment, on propose de réaliser un premier choix quant à la méthode de Machine Learning que l'on souhaite utiliser afin de répondre à la problématique, i.e. d'automatiser l'analyse

4.1 Architecture High Level du système proposé

4.2 Solutions techniques testées

4.3 Solution technique proposée

4.4 Dimensionnement de la solution

Chapitre 5

Industrialisation du produit

5.1 Définition du terme d' "industrialisation"

5.2 Mise en place du process fonctionnel

5.3 Présentation des outils

5.3.1 API

5.3.2 Outils graphiques

chapterConclusion