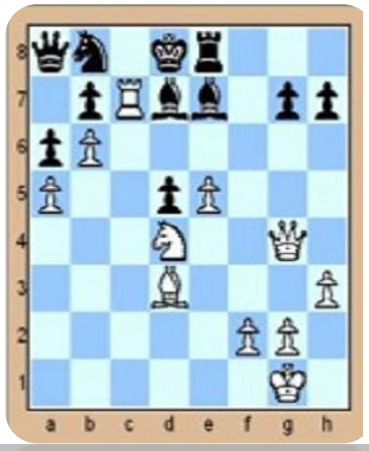
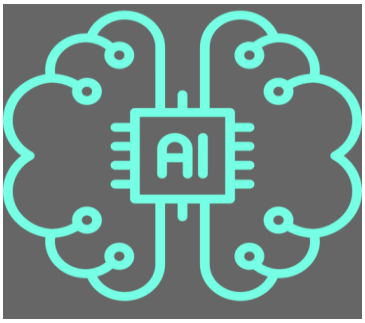
Intelligence artificielle



Artificial Intelligence (IA)
Applied to Electrical Engineering



Cours enseigné par
Ecole Nationale Polytechnique de Constantine (ENPC)

2016-2024

Intelligence artificielle appliquée à l'électrotechnique

Sommaire

1. **Introduction et définition de l'intelligence artificielle**
2. **L'intelligence artificielle dans la commande et le contrôle des systèmes électriques**
3. **Classification supervisée et non supervisée**
4. **Logique floue et applications en électrotechnique**
5. **Réseaux de neurones et applications en électrotechnique**
6. **Algorithmes génétique et applications en électrotechnique**
7. **Algorithmes modernes d'optimisation**

Summary

1. **Introduction and definition of artificial intelligence**……..... 1

1.1.AI introduction……. 3
1.2. AI History……. 3
1.3. Some AI areas …….. 8
1.4. AI research areas……... 8
1.5. AI applications…….. 9
1.6. Biblio Other Forum……... 9

2. Artificial intelligence in the command and control of electrical systems… 11
2.1. Introduction…….. 11
2.2. The Samurai robot…… 12
2.2.1. Automatic speaking processing……. 13
2.2.2. Representation, reasoning, development of plans……. 13
2.2.3. Location, cartography, forms recognition……. 14
2.2.4. Contemporary complement: the semantic web…….. 14
2.3. Algorithmic and learning……..... 14
2.3.1. Learning algorithmic…….. 16
2.3.2. Learning challenges……... 16
2.3.3. Multi-agent systems……... 16
2.3.4. Complexity, logic…….. 17
2.4. Robotics, real world confrontation and bio-inspiration…… 17
2.5. Recognition of forms, excavation and search for images……. 18
2.5.1. ANR projects …… 19
2.6. Automatic language and speech processing……. 19
2.7. Knowledge, Web, Semantic Web……. 19
2.8. AI APPLICATIONS…… 19
2.8.1. Example 1: Discover how artificial intelligence systems are used in the world of transportation and integrated into autonomous vehicles…….. 19 20
2.8.2. Example 2: ABS braking system with different controllers (P,PI, PID and Fuzzy)…….. 22 24
2.8.3. Example 3: AI and traffic monitoring…… 24
2.8.4. Example 4: Object detection (LIDAR) …… 25
2.8.5. Example 5: How do autonomous vehicles work?..... 26
2.9. References …… 27

3. Supervised and unsupervised classification……... 27
3.1. Introduction…….. 27
3.2. Useful concepts and definitions…… 27
3.3. Supervised or unsupervised classification…… 28
3.3.1. Supervised classification…… 29
3.3.2. Unsupervised clustering……. 31
3.3.2.1.The steps of an automatic classification……

3.3.2.2. Automatic classification algorithm…….. 31
3.3.2.3. Applications of clustering……. 31
3.4. Aggregation criterion…… 32
3.5. Similarity measures…….. 32
3.6. Evaluation and validity criteria……. 34
3.6.1. Determination of the number of clusters…….. 34
3.6.2. Core Concepts of Cluster Validity…… 35
3.7. Evaluation of a classification system…… 36
3.7.1. Test corpus (supervised case)…… 36
3.7.2. Unsupervised case……. 38
3.8. Classification techniques ……. 38
3.9. Hierarchical algorithm……. 38
3.10. Algorithms per partition…… 38
3.11. Density-based algorithms……. 38
3.12. Classification based on grid quantification…….. 38
3.13. Other methods…….. 38
3.14. The K-means method and its variants…….. 39
3.14.1. Presentation……..... 39
3.14.2. Characteristics of the *K*-means algorithm …… 39
3.14.3. Idea……..... 39
3.14.4. Example……..... 39
3.14.5. Algorithm……..... 42
3.14.6. Advantages and disadvantages……. 42
3.15. Hierarchical clustering……. 42
3.15.1. Intuitions and principles…… 42
3.15.2. CAH algorithm…….. 43
3.15.3. Graphical example……. 44
3.15.4. Dendrogram……..... 44
3.15.5. Idea…… 44
3.15.6. Example in dimension 2…… 44
3.15.7. Advantages and disadvantages…… 46
3.16. *k* nearest neighbors……. 46
3.16.1. Presentation…… 46
3.16.2. Idea …….. 47
3.16.3. Example in dimension 2, *k*=3…… 47
3.16.4. Exemple en dimension 2, *k*=10 ……. 48
3.16.5. Advantages and disadvantages……. 49
3.17. Centroid approach……. 49
3.17.1. Idea……..... 49
3.17.2. Example……..... 49
3.18. References……. 51

4. Fuzzy logic and applications in electrical engineering…….. 52
4.1. Introduction…….. 52

4.2. History of fuzzy logic…….. 52
4.3. Why Fuzzy Logic: Limits of Classical Logic…….. 53
4.4. Fuzzy set theory…… 53
4.4.1. Notion of partial membership…… 53
4.4.2. Membership functions…….. 54
4.4.3. Fuzzy logic operators…….. 56
4.4.4. Fuzzy rules……. 58
4.5. Fuzzy Control…… 64
4.5.1. Example 1: Fuzzy command of an automatic watering system…… 65
4.5.2. Example 2: Fuzzy speed control of a separately excited DC motor [23]…….. 67
4.6. References……. 70

5. Neural networks and applications in electrical engineering……. 71
5.1. Introduction…….. 71
5.2.1. Historical elements…… 71
5.2.2. The first successes……. 71
5.3. Apps……..... 72
5.4. View of several biological neurons……. 72
5.5. Real neuron……. 73
5.6. Formal neuron……. 73
5.7. Advantage of Neural Networks……. 76
5.8. Disadvantages……. 76
5.9. Architecture and operating principles……. 76
5.10. Input data type…… 77
5.11. Exploitation of results……. 77
5.12. Hidden Layer Parameterization……. 77
5.13. Values of nodes and links…….. 77
5.14. The sigmoid function…….. 78
5.15. SEC……. 79
5.16. Backpropagation……. 79
5.17. Stopping criteria…….. 81
5.18. Model with multiple target variables…….. 82
5.19. Interpretation of results: sensitivity analysis…….. 82
5.20. Definitions…….. 82
5.21. Learning…….. 83
5.21.1. Hebb's law, an example of unsupervised learning……. 84
5.21.2. The Perceptron learning rule, an example of supervised learning…… 86
5.22. Different neural architectures……. 88
5.23. Neural network applications……... 92
5.24. References……... 96

6. Genetic algorithms and applications in electrical engineering……. 97
6.1. Introduction……. 97
6.2. Evolutionary algorithms…… 98
6.3. Genetic algorithms for optimization pbs……... 99
6.4. How genetic algorithms work?..... 100
6.5. Genetic algorithms areas……... 100
6.6. Genetic algorithms paradigm…… 101
6.7. Genetic algorithms form……... 101
6.8. Genetic algorithms processor…… 101
6.8.1. Coding…….. 101
6.8.2. The selection operator…… 103
6.8.3. The crossover or crossover operator……. 104
6.8.4. The mutation operator…….. 106
6.8.5. The replacement operator……. 107
6.8.6. Elite replacement ……. 108
6.9. Genetic algorithms steps programming…… 108
6.10. Example 1: calculate the maximum of a real function implementing all the operators…….. 110
6.11. Example 2 find the set of parameters (w_1: w_6???) that maps the following inputs to the outputs of *y* function ……. 112
6.12. Example 3: Adjusting the parameters of a PID regulator by optimization method…….. 115
6.13. References…… 117

7. Modern optimization algorithms…… 118
7.1. Introduction…….. 118
7.2. Definition…….. 118
7.3. WEIERSTRASS theorem……. 119
7.4. FERMAT's theorem ……. 119
7.5. Nature of a critical point: direct study…….. 120
7.6. Example…… 120
7.7. Related Extrema…… 121
7.7.1. Problem…….. 122
7.7.2. Definition Linked Optimization…… 122
7.7.3. Example……. 122
7.7.4. LAGRANGE multiplier theorem……. 123
7.8. Optimize $f(x, y)$ under the constraint $g(x, y) = 0$……. 124
7.8.1. Example……. 125
7.9. Modern optimization method based Particle Swarm Optimization (PSO)……. 126
7.9.1. Particle Swarm Optimization (PSO): Particle Swarm Optimization is a method……... 126
7.9.2. How does this algorithm work?..... 126
7.9.3. Example 1 find the maximum of real function based

PSO…… 127
7.9.4. Example 2: particle swarm optimization applied to the power flow computation…… 130
7.10. References …… 132

Unité d'enseignement Fondamentale UEM 1.3
Matière: Intelligence Artificielle
Code: IA

I. Objectifs de la matière

L'intelligence artificielle (IA) ou la science de la machine a pris un développement très rapide, surtout avec l'apparition du programme informatique ChatGPT en 28 Avril 2023. Chacun veut l'intégrer dans ses différents projets.

Ce module introduit le concept d'intelligence artificielle et lève les ambiguïtés et les questions que les étudiants posent. Comment fonctionne l'intelligence artificielle ? Quelles tâches l'intelligence artificielle peut-elle gérer ? Quelles sont les conséquences de l'intelligence artificielle sur nos vies ? Comment bien se préparer avant de lancer un projet IA ? A travers diverses techniques telles que la logique floue, les réseaux de neurones, les génétiques algorithmes, les méthodes de l'apprentissage automatique…*ect* qui permettent d'introduire ce concept dans de nombreux domaines.

L'Objectifs d'IA

- Présenter quelques exemples d'applications d'IA (p. *ex.*: le robot Sofia, système de freinage automatique, un système d'arrosage…) ;
- Comprendre les principes et les mécanismes de base exploités dans les techniques d'intelligence artificielle ;
- Savoir reconnaitre les forces et les faiblesses des approches fondées sur les techniques d'intelligence artificielle ;
- Être capable d'identifier les situations où les techniques d'IA sont candidates pour la résolution de problèmes complexes ;

II. Compétences visées

En fin d'UE les étudiants devraient être capables de :

- apprendre les principes de base de l'IA.
- Identifier les principes généraux de l'apprentissage automatique;
- Contraster ses capacités et ses limites;
- Articuler des opinions sur des nouvelles et des événements en IA.
- Apprendre le fonctionnement théorique et pratique de l'IA.
- Reconnaître les algorithmes typiques utilisés en apprentissage automatique supervisé;
- Paraphraser comment les concepts mathématiques sont utilisés dans la théorie de l'apprentissage supervisé; Les probabilités et statistiques et L'optimisation.
- être appelées à lancer et gérer un projet en IA.
- Déduire si l'IA est appropriée pour une tâche ou une application industrielle;
- Formuler des attentes réalistes;
- Se questionner sur les conditions pour que le projet soit un succès et, pour ce faire, les préparatifs nécessaires.

III. Connaissances préalables recommandées

Pour réussir cette matière, les étudiants devraient avoir des compétences en Math et en programmation.

IV. Contenu

1. **Chapitre 1 : Introduction et définition de l'intelligence artificielle (01 semaines)**
   1.1. Introduction
   1.2. Historique de l'IA avec des exemples
   1.3. Quelques domaines de l'IA
   1.4. Domaines de recherche en IA
   1.5. Applications de l'IA
2. **Chapitre 2 : L'intelligence artificielle dans la commande et le contrôle (02 semaines)**
   2.1. Introduction
   2.2. Le robot Samuraï
   2.3. Algorithmique et apprentissage
   2.4. La Robotique, La Confrontation Au Monde Réel Et La Bio-Inspiration
   2.5. Reconnaissance de formes, fouille et recherche d'images
   2.6. Traitement automatique de la langue et de la parole
   2.7. Connaissances, Web, Web Semantique
   2.8. Les applications de L'IA
3. **Chapitre 3 : Classification supervisée et non supervisée (03 semaines)**
   3.1. Introduction
   3.2. Concepts et définitions utiles
   3.3. Classification supervisée ou non supervisée
   3.4. Critère d'agrégation
   3.5. Les mesures de similarité
   3.6. Évaluation et critères de validités
   3.7. Evaluation d'un système de classification
   3.8. Techniques de classification
   3.9. Quelques approches classiques
   3.10. La méthode des K-moyennes et ses variantes (Algorithme & exemple)
   3.11. Clustering hiérarchique (Algorithme&exemple)
   3.12. k plus proches voisins (Algorithme & exemple)
   3.13. Approche de centroide (Algorithme & exemple)
4. **Chapitre 4 : Logique floue et applications (03 semaines)**
   4.1. Introduction
   4.2. Historique de la logique floue
   4.3. Pourquoi la logique floue h limites de la logique classique
   4.4. Théorie des ensembles flous
   4.5.Commande Floue h exemple système d'arrosage automatique
5. **Chapitre 5 : Réseaux de neurones et applications (03 semaines)**
6. **Chapitre 6 : Algorithmes génétiques et applications (02 semaines)**
7. **Chapitre 7 : L'optimisation (02 semaines)**

Preface

Artificial intelligence (AI) applied to electrical engineering

Artificial intelligence (AI) is a discipline with vague outlines. If we leave aside strong artificial intelligence, which is still distant, which aims to simulate human behavior in all fields of knowledge and interactions, so-called weak artificial intelligence is a family of methods used to solve problems comes from the resemblance to human reasoning.

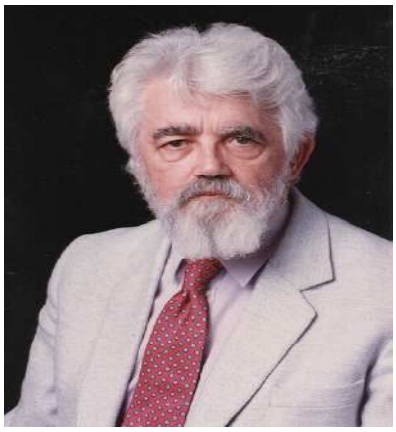
Before, so-called intelligent systems quickly calculated solutions to problems based on behavioral laws coded by engineers specializing in problem in question. This ranged from an electronic chess game for beginners to a system of load shedding of the electricity network. Since the appearance of the first artificial neurons inspired by our biological neurons, the resemblance with human intelligence has gone down to the level of learning.

Each advance of AI, its victories in chess then in the game of go, its musical or pictorial creations, its instant translations, its medical diagnoses raise questions about professions that would be inaccessible to it and about the nature of human intelligence.

AI making it possible to process complex systems that are difficult to describe by equations or trees (images, human voices, etc.), neural networks are a very promising tool also for engineering sciences, whether by using them in their preferred fields (image processing, voice processing, language processing, diagnosis …) or directly for machine control or material modeling. This explains their appearance in engineering training, particularly in the electrical engineering sciences program. In addition to the advantages offered and good performances of electrical systems by fuzzy logic in control compared to classical controllers.

The objective of this lecture is to provide the necessary bases to implement AI algorithms in electrical engineering sciences. This includes seven chapters from presentation of concepts (Introduction and definition of artificial intelligence, artificial intelligence in the command and control of electrical systems, supervised and unsupervised classification, fuzzy logic and applications in electrical engineering, neural networks and applications in electrical engineering, genetic algorithms and applications in electrical engineering, modern optimization algorithms) with applications examples of AI industry and research. Finally exercises and applications work using Matlab for some, Python libraries for others.

Chapter 1: Introduction and definition of artificial intelligence (IA)



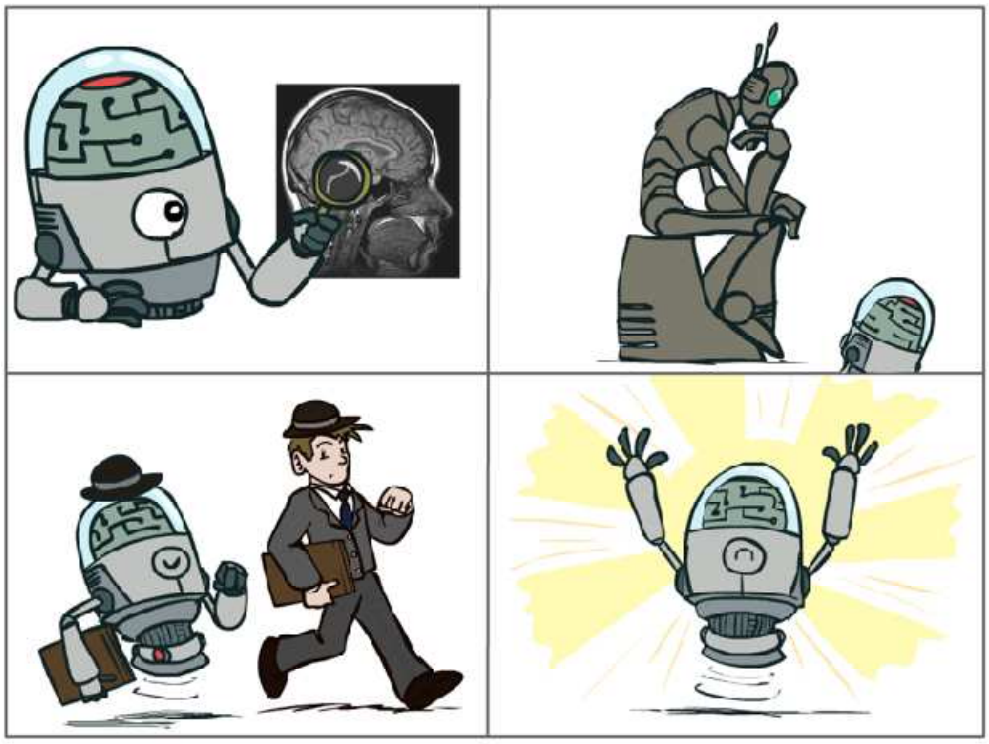
John McCarthy
[https://nationalmedals.org/laureate/john-mccarthy/]

Workshop organisé à Dartmouth College en 1956 par John McCarthy et formellement proposé par McCarthy, Marvin Minsky, Nathaniel Rochester et Claude Shannon. Le terme IA a été forgé par McCarthy à cette occasion.

"We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer." (McCarthy et al. 1955)

Artificial intelligence (AI) is the science of machines that:

Think humanely



Think rationally

Act humanely

Act rationally

Figure 1.1. Intelligent machine characteristics [1].

Acting humanely: necessary skills?

- Natural language processing to communicate at a human level;
- Knowledge representation to record human-level information and knowledge;
- Automatic reasoning to draw relevant conclusions from the information provided;
- Learning to adapt to new circumstances and extrapolate from cases already seen.

There is a more advanced version of the Turing test called the Total Turing Test in which the interrogator can send images and provide objects to the subject.

Thinking humanly: the cognitive models approach

- How do humans think? Two methods of investigation: introspection psychological experiments;
- If we have been able to create a theory of human thought, we can try to make a computer model of it;
- If the computer model holds reasoning analogous to human reasoning, then there is a good chance that the model is correct;
- In GPS (General Problem Solver), Simon and Newell were less interested in the fact that the program found the right answer than in the trace of the reasoning followed – unlike other researchers at the same time;
- Cognitive sciences try to build models of human reasoning based on computer models derived from AI and on the results of experimental psychology.

Thinking rationally: the approach by the laws of thought

Socrates is a man, all men are mortal, and therefore Socrates is mortal.

- Aristotle: correct thought, syllogisms are patterns that always provide correct conclusions if the premises are correct;
- Birth of logic as a discipline;
- Development of formal logic towards the end of the 19th and the beginning of the 20th century;
- Principle of resolution of Robinson in 1965;
- Logician stream in AI: using logic to represent knowledge and reasoning;
- Some obstacles to this approach: difficult to represent all knowledge in logical form, the logic of 1st order predicates is certainly not enough difference between solving problems in theory and in practice: enormous computing skills are needed;
- Advantages: you master the theory used.

The problem with AI is common sense reasoning. All you have to do is create it!

Rational decisions: Term rationality used in a very technical sense:

- seek to achieve the predefined objectives as much as possible;
- be concerned only with the decisions that are made, not the process that leads to them;
- the goals are characterized in terms of the usefulness of the results;
- being rational means maximizing your expected utility ⇒ artificial intelligence = rationality by calculation;

Acting rationally: the rational agent approach

- Act rationally ≡ try to achieve goals given beliefs;
- An agent is something capable of perceiving and acting;
- Encapsulates the needs of two other approaches:
    1. acting rationally may require thinking rationally, but that is not enough;

2. the capacities exhibited in the Turing test are useful for an agent to communicate and behave in adequacy with its environment.

Two advantages of studying AI from this angle:

1. more general than the laws of thought approach, does not close doors on other techniques to fill gaps;
2. more suited to a “hard” scientific approach than models based on human behavior, the rationality sought is clearly defined;
3. We will focus on building rational agents and the principles behind them;
4. Achieving absolute rationality (perfect, always making the best decision) is almost always impossible, the necessary resources being too great. But we will still seek to understand how to find this best decision.

Artificial intelligence strong and weak:
Artificial intelligence strong: is project to create a machine capable not only simulating intelligent behavior, but experiencing real self-awareness, “real feelings” and an understanding of its own reasoning. Engine of the discipline, but arouses many debates.

Artificial intelligence weak: engineer pragmatic approach. Seeking to build increasingly autonomous systems, algorithms capable of solving problems of a certain complexity. The machine seems to act as if it were intelligent. It is mainly on the basis of this assumption that most current artificial intelligence techniques are used.

1.1. AI introduction

The goal of Artificial Intelligence (AI) is to design systems capable of reproducing the behavior of humans in their reasoning activities.
AI sets itself the goal of modeling intelligence as a phenomenon (as well as physics, chemistry or biology, which aims to model other phenomena).

- A machine will be considered intelligent if it reproduces the behavior of a human being in a specific domain or not;
- A machine will be considered intelligent if it models the functioning of a human being.

1.2. AI History

Prehistory 1945-1955:

Automatic language translation => knowledge representation, extraction problem and generic writing problem.

Artificial Intelligence in Science Fiction Literature and Film (Films: Kubrick's 2001 Odyssey, Spielberg's AI)

Appearance of the word robot for the first time in 1923 in the play “R.U.R” (Rossum’s Universal Robots) written by Karel Capek.

In 1950, Isaac Asimov (author of Science fiction with a scientific background) proposed his three Laws of robotics.

- A robot must not attempt the life of a human,
- A robot must obey the commands of a human unless it contradicts the first law,

- A robot must preserve its own existence unless it contradicts the first two laws) (Movie I, Robot with Will Smith, 2004),

The beginnings 1955-1970

- Artificial intelligence term appeared in 1956 when Minsky, McCarthy, Newelle and Simon met at Dartmouth College (New Hampshire, USA).

- Era of absolute enthusiasm (Simon in 1958): in less than ten years a chess program will reach the level of a world champion and an automatic theorem demonstration program will discover a mathematical theorem. Yet Kasparov was only beaten by the Deep Blue machine in 1997!

- Work development: chess games, demonstration of theorems in geometry

- Appearance of the first program, LOGIC THEORIST (automatic theorem demonstration) in 1956 and of the IPL1 language. Appearance of the Lisp languages in 1960 by MacCarthy, and Prolog in 1971 by Alan Colmerauer.

- Eliza was built at MIT in 1965, an intelligent system that dialogues in English and plays the psychotherapist.

- 1970: SCHRDLU, software designed by Terry Winograd. It simulates the manipulation of geometric blocks (cubes, cylinders, pyramids ...) placed on a table. The software automatically generates plans (<< To move the blue cube on the top of the yellow cylinder, I must first remove the pyramid that is on the cube and ...>>) and is provided with an interface in natural language.

Expert systems, including:

- 1969: DENDRAL: analysis of the results of a mass spectrography.
- 1967: MACSYMA (formal calculation software).
- 1977: MYCIN (infectious diseases).
- HEARSAY-II in speech understanding,
- PROSPECTOR in geology.

Specialization 1970-1980 (specialization and theorization)

- AI is the crossroads of several disciplines: computer science, logic, linguistics, neurology and psychology). Birth of the Small talk language in 80;
- Simon received the Nobel Prize in Economics in 1978

Recognition 1980-1990 (credibility and audience)

- Fifth generation project by MITI (3 alphabets for the Japanese: Katakana, Hiragana and Kanji => ideograms) MITI is the former acronym of the name of the Japanese Ministry of Economy replaced today by METI.

From the 1980s

Computer-specific techniques were developed from 1980 : neural networks (NN) to simulate the architecture of the human brain, genetic algorithms (GA) to simulate the process of natural selection of individuals, inductive logic programming that makes "work upside down" the usual process of deduction, the Bayesian networks which are based on the theory of probabilities to choose, among several hypotheses, the most satisfactory.

Late 1980s

AI has essentially to focused on theories and techniques allowing the realization of individual intelligences. In nature, however, there is another form of intelligence – collective this one, such as simple multicellular beings, colonies of social insects, human societies. These sources of inspiration show that a form of higher intelligence can result from the correlated activity of simpler entities. In artificial systems, this field is called “distributed AI” or “multi-agent systems”.

1990-2000

The advent of the Internet has paved the way for knowledge sharing and communication. How to organize and process these gigantic masses of information? How to extract relevant knowledge for the problems posed? Search engines like Google have integrated advanced information retrieval (*information retrieval*) and artificial intelligence (*data mining*) techniques into their tasks.

1990-2000 (continued)

- In 1994, a French team, as part of its research into artificial life, developed «the chance gardens »; These are virtual gardens whose evolution depends on digital data received by modem in real time. They are composed of several families of forms that are born, grow, die and interact with each other following behaviors inspired by life. They constitute ecosystems of artificial life. Object colors change with weather data and with chronological time over days and seasons.

Figure 1.2. Image of the chances garden.

- In 1995, Carnegie Mellon University's ALVINN automatic vision system allowed the automatic driving of a vehicle called *Navlab5* to be driven automatically from Pittsburgh to San Diego, while human operators handled the brake and throttle [1].

- In 1997, in Philadelphia, the world chess champion, Garry Kasparov, was defeated by Deep Blue, an IBM computer, in six rounds. Kasparov won the first, lost the second and played the rest very badly. Furious, he had to bow to the machine. *Kasparov got wiped off the board*, Grand Master Gurevich said.

Figure 1.4. An IBM computer similar to Deep Blue in his 1997 match, Exhibited at the Museum of Computer History in Mountain View, California.

- Also in 1997, *RoboCup* was held for the first time, the championship of robots that play football (or soccer, if you prefer this North American term). This happened in Nagoya, Japan at the IJCAI-97 conference.

- In 1999, an artificial and intelligent NASA agent flew a satellite past the planet Mars for an entire day without any help from Earth.

The 2000s

- The acquisition of knowledge has allowed the creation of ontologies of various kinds. An example is the *Unified Medical Language System*.

- Success in natural language processing, *WordNet* is a lexical database in English and *OpenCyc* is a knowledge base that formalizes common sense knowledge.

- Online learning, or *e-learning*, is on the rise. Thanks to its techniques, AI has made it possible to implement increasingly efficient distance education systems.

There is a better consideration of the learner's profile (cognitive, affective and inferential), a most often collaborative construction of the knowledge base (curriculum), a more intelligent interaction between the system and the learner, *etc*.

- In addition, there are more and more systems for recommending products or services on the Web: films, books, courses, restaurants, trips, bus or metro routes. They are all based on AI techniques such as case-based reasoning, *content filtering* or *collaborative filtering*. Some of them take into account demographic data but also customer purchasing habits as well as web browsing behavior (*web usage mining*).

- The Captcha system, developed at Carnegie Mellon University, deals with differentiating humans from machines. Captcha generates tests that only humans can pass, in order to combat spam and malicious actions by certain machines.

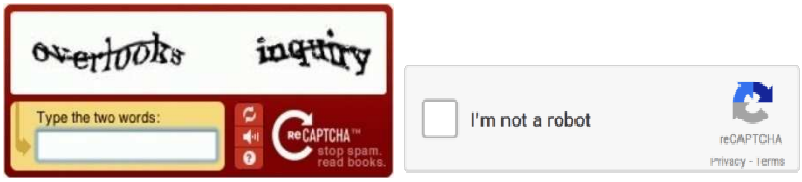


Figure 1.5. Captcha.

- The American team of Sargur Srihari, director of the Center of excellence for document analysis and recognition (CEDAR, State University of New York at Buffalo), developed in 2002 software capable of distinguishing with 98% certainty whether two documents were written by the same person or not. The work was carried out on a sample of 1500 people.

- Featuring an android head and polymer skin, *K-Bot* (designed by David Hanson of the University of Texas) can recognize and track our movements. It has 24 mechanical muscles that allow it to simulate 24 of our facial expressions. This can be of great use to researchers studying human-machine communication.

Figure 1.6. K-Bot.

- *Wakamaru*, a robot developed by Mitsubishi Heavy Industries and endowed with speech, is mainly designed to watch over the elderly. Intended to fit into the family life of everyone, its mission will be to notify the hospital or health services if necessary.

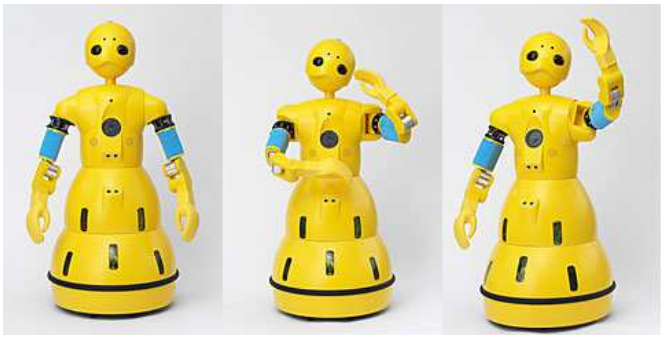


Figure 1.7. Wakamaru robot.

- From January 26 to February 7, 2003, in New York, Garry Kasparov competed in six games of chess against *Deep Junior*, three times software world champion. The match ended in a 3-3 tie. Unlike the famous 1997 tournament, the 39-year-old champion had plenty of time to train beforehand on a PC with the commercial version of *Deep Junior*, which is not far from the level of the current Deep Junior on the same machine. It is able to review three million positions per second!

- ✪ On January 13, 2004, a Quebec firm reported in the *Medical Post* the marketing of a portable cardiac alert system called the *Vital Positioning System* (VPS). Including a cell phone, handheld computer and GPS, this system can detect the approach of a heart attack 8 minutes before the first symptoms are humanly perceptible then automatically call the nearest hospital and specifies the location of the future patient.

1.3. Some AI areas

Expert systems (medicine, financial analysis, device configuration) Task of diagnosis, monitoring or troubleshooting of industrial installations.

Robotics and CAM (computer-aided manufacturing) introduction of robots that acquire information using sensors or cameras in order to move in diverse environments.

Understanding of language and automatic translation Appearance of natural language interfaces=> querying databases in French or English (example: Dune film).

Pattern recognition (speech, image and handwriting recognition) IBM uses an auditory interface that recognizes 20,000 English words from business vocabulary and writes them on the screen.

Learning create programs that generate their own knowledge by modifying themselves from their successes and mistakes.

Artificial emotion (Rosalind Picard's work on emotion).

1.4. AI research areas

***Machine learning*:** This process gives an agent the ability to perform tasks that could not be performed before or to perform more efficiently (faster and more accurately) tasks that it was already performing. *Analytical learning systems* aim to analyze and put into a more effective or "operational" form the knowledge that already exists. *Synthetic learning systems* aim to discover fundamentally new knowledge.

Virtual reality: This field offers new forms of interaction between man and machine. The arrival of more powerful computers, equipped with impressive three-dimensional graphic capabilities, coupled with visualization and interaction peripherals (helmet, glove, *etc.*), makes it possible to provide the sensory information necessary to convince users that they are immersed. Larry Hedges of the Georgia Institute of Technology has long used virtual reality to cure certain phobias such as those of the elevator or those of spiders.

Pattern recognition: Research in this area aims to automate the discernment of typical situations in terms of perception. His methods find numerous applications. These include vision, speech recognition, optical document reading and image synthesis. Advances in video image recognition already allow police to spot a target in a crowd.

Artificial life: This field is interested in the study of ecosystems and the reproduction, by artificial systems, of characteristics specific to living systems (from cellular functioning mechanisms to population dynamics, including models of individual development).

***Robotique*:** An important subfield of AI, robotics can be seen as an intelligent interconnection of perception, action, and the functioning of robots. Used to maintain dynamic representations of their environment, it allows robots to acquire the ability to sense, move, reason and possibly communicate in natural language.

***Multimedia indexing* :** The multimedia resources found on the Web today are numerous, voluminous and sometimes irrelevant. AI therefore offers tools for "searching" databases (*data mining*) in order to extract synthetic knowledge or to discover hidden information, to diagnose situations, or help supervise the operation of systems.

1.5. AI applications

- Medical diagnosis: therapy, device monitoring
- Image synthesis: computer vision
- Natural classifications: (biology, mineralogy, etc.)
- Task planning: (financial forecasts, ...)
- Architecture (computer aided design)
- Fault detection (Sherlock for F16 aircraft)
- Education (Intelligent Tutorial Systems, e-Learning)
- Engineering (verification of design rules)
- Geological prospecting (mining deposits)
- Nuclear power plants, forest fires (real-time systems)
- Flight simulators (CAE, Bombardier, etc.)
- Games (videos)

1.6. Biblio Other Forum

[1] Scott L. Andresen, «Hebert A. Simon: AI pioneer», IEEE Intelligent Systems 16(4): 71–72, 2001.

[2] Marvin Minsky, «*A framework for representing knowledge*», in Patrick Henry Winston, éd. 1975.

[3] Patrick Henry, «*The Psychology of Computer' Vision*», Jan 01, 1975.

[4] Hill, New York, USA, 1975.

[5] <web.media.mit.edu/~minsky/papers/Frames/frames.html>

[6] <www-poleia.lip6.fr/~gortais/pages/Jardin%20des%20Hasards.html>

[7] www.nlm.nih.gov/research/umls

[8] www.cs.waikato.ac.nz

[9] www.opencyc.org

[10] Tim Menzies, «*Guest editor's introduction: 21st-century AI –Proud, not Smug*», IEEE Intelligent Systems 18(3): 18–24, 2003.

[11] www.captcha.net

[12] www.cedar.buffalo.edu/NIJ/

[13] https://www.cs.cmu.edu/~tjochem/nhaa/navlab5_details.html

[14] Ashe Wagshum, Chyi-Yeu Lin and Shih-Hsiang Yen, «*Humanoid Head Face Mechanism with Expandable Facial Expressions*», May 2017 International Journal of Advanced Robotic Systems 13(1):1, 10.5772/62181

[15] Fabrice Popineau Mineure, «*Cours Introduction Intelligence Artificielle*», 2015.

[16] Cyril Terrioux, «*Introduction a l'Intelligence Artificielle*».

[17] Meghyn Bienvenu, «Notes du cours "Introduction à l'Intelligence Artificielle», 2018.

[18] Philippe Beaune, Gauthier Picard, Laurent Vercouter, «*Initiation à l'Intelligence Artificielle*», 2012.

Chapter 2: Artificial intelligence in command and control of electrical systems

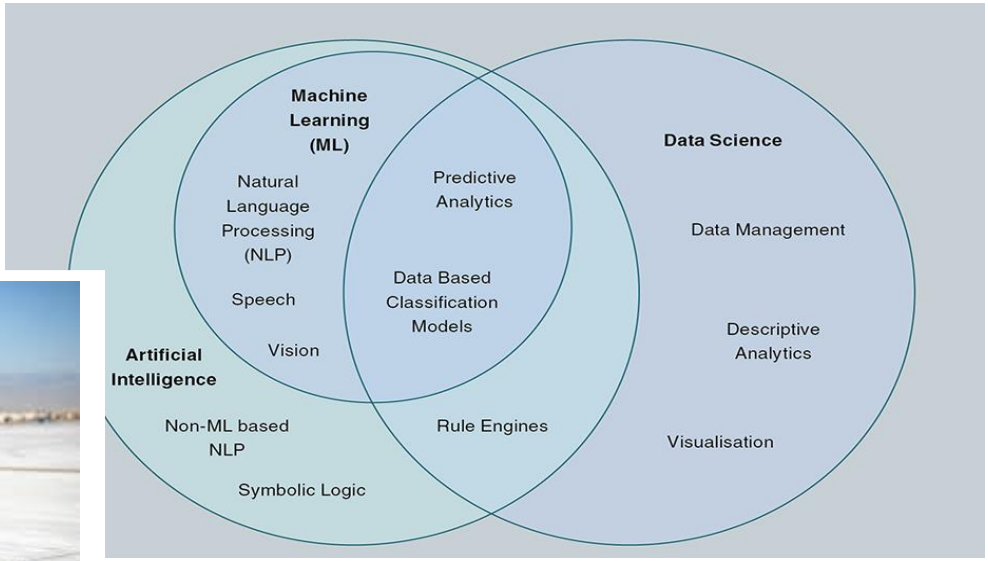


Figure 2.1. Data Science vs Machine Learning and Artificial Intelligence.

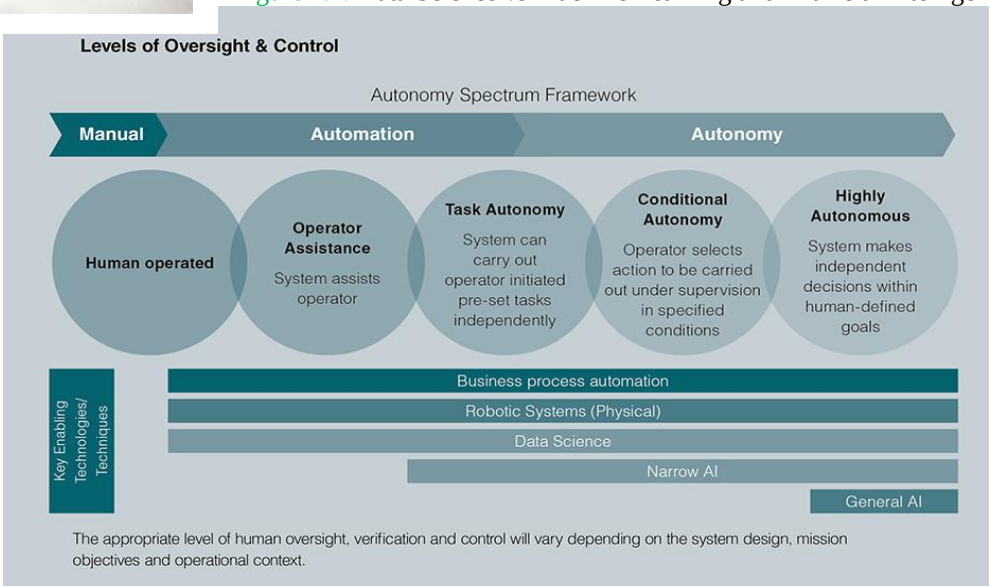


Figure 2.2. Levels of oversights & Control.

Command elements are authority, responsibility, decision making, and leadership.

Control elements are direction, feedback, information, and communication.

2.1. Introduction

Techniques from theories of artificial intelligence have contributed significantly to the new advances made in various other disciplines.

Each application therefore makes it possible to deepen the scientific knowledge of the chosen field. It is useful for researchers, decision -makers but also for a large audience. The importance and diversity of research projects and scientific teams are thus highlighted.
Unpretentious to be exhaustive, the objective is to explain the major issues. It is made to various research projects funded by the ANR. A synthetic presentation of projects is proposed.
The ANR puts in perspective the scientific advances and the results obtained by the high quality projects it supports. This chapter presents a set of projects on the themes of artificial intelligence and robotics, subjects at the heart of information sciences and technologies, where basic research on algorithms, models, methods, rubs shoulders with applications in many sectors such as health, transport, web, or industrial processes.
Artificial and robotic intelligence are also subjects on which the imagination is very rich. It is important that the ANR contributes to demystify research on the subject and highlights the rich scientific content it represents. Among the scenarios, there was in particular that of the replacement of man by artificial systems. If the theory of singularity evokes such a possibility, the objective of the research pursued is to provide assistance to human users, to identify them from painful, dangerous, or repetitive tasks, in order to devote itself to the most interesting and enriching activities. The multidisciplinarity of research in artificial intelligence and robotics is found not only in the diversity of the applications presented, which naturally bring together researchers in stic and specialists in the sectors concerned, but also in the themes covered, for example: Bio-inspired robotics , for which Automaticians and Informatics collaborate with mechanics, biologists, researchers in behavioral sciences; Automatic language processing, which mobilizes linguists, interaction specialists, and researchers in probabilistic algorithmic; Semantic web, for which researchers in representation of knowledge work with sociologists for social networks and folksomies aspects, and philosophers for the development of ontologies. This multidisciplinarity is a great richness, carrying scientific and technological ruptures, and therefore of added value for society and the economy.

2.2. The Samurai robot

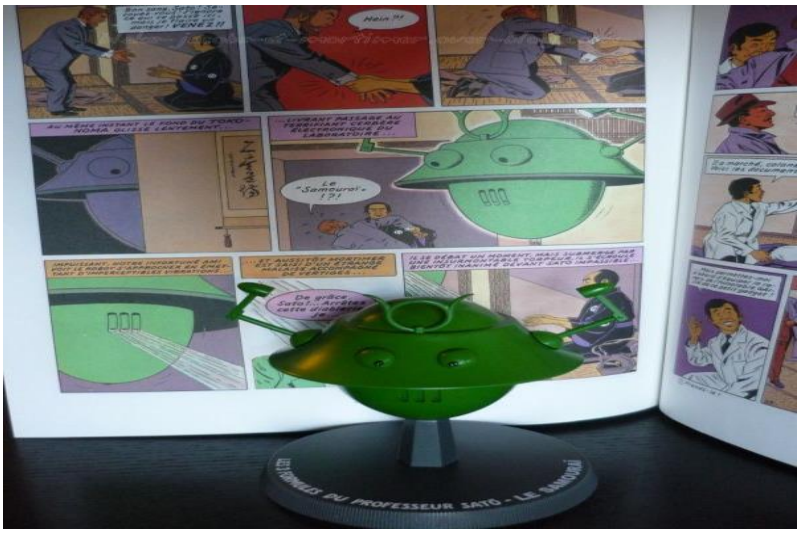


Figure 2.3. Samurai robot.

Record… 220 M.C.!

Consider it and destroy it!

GOES!

These instructions given by Professor Sató in the Samurai robot, intended to destroy the malicking Olrik, are a good illustration of some challenges of artificial intelligence. The artificial intelligence of the robot is constantly constructed in interaction with the external environment (it is said to be located). The robot perceives by its sensors (here, it seems, two artificial eyes and an auditory channel), whose variations will induce a reaction from its part, for example turning when it is called, or modify its actions To take into account new elements. The Samurai robot must understand the word, interpret the orders of its master; One objective is given to him, he will therefore have to build an action plan to reach his goal. Then, to be able to complete it by controlling the smooth running of its execution, the Samurai robot will have to interpret the scenes it will face, automatically recognize forms, people, objects in the series of captured images.

Finally, and this is a specificity of robotics, he will act directly on his environment, here trying to destroy Olrik thanks to the weapons he has.

This anx notebook devoted to intelligent systems and robotics, in its perceptual and cognitive dimensions, constitutes an illustration of recent progress made towards the ambitious objective of making intelligent machines capable of assisting or replacing human operators in complex tasks. The projects gathered there approach many challenges including those that the Samurai robot must take up.

2.2.1. Automatic speaking processing

To understand the word of Professor Sató, the Samurai must transform the audio signal captured by his artificial ears into a series of instructions expressed in a formal language which he can manipulate to conduct reasoning.

First, the teacher's perceptual environment is probably filled with noise from other sources that must be distinguished. If it is a question of sound, an audio signal processing step is necessary in order to clean it, so that there is only the text pronounced by Sató. It will then be necessary to separate the audio signal into phonemes, undoubtedly in reference to a library of pre-recorded sounds, whether by the speaker himself (the professor), or by a set of speakers representing the diversity of possible pronunciations of the Japanese. These phonemes will then have to be assembled in words, here also probably referring to a dictionary and grammar rules. The words must then be gathered in sentences, which uses knowledge of syntax and grammar.

Finally, and this is not the slightest challenge, it will be a question of giving meaning to the sentences, of moving from a syntactic, grammatical representation, to the semantic representation on which the Samurai robot can lead reasoning. The treatment of the natural language, written or spoken, is an important subject of research in AI.

2.2.2. Representation, reasoning, development of plans

From the semantic representation of the objective to be achieved, the robot will lead reasoning in order to define its action plan. For that, he will need to reason on himself, on his state, on the weapons and means of travel he has; He will also have to establish that to destroy Olrik, the most effective is certainly to get closer so that the enemy is within reach. He will undoubtedly need to reason to decide which direction to leave. Finally, once the reasoning has been done, the robot will build a plan to reach the lens in a series of elementary steps, we could note for example:

<Samuraï> <déplace_vers> <location

<Samurai> <xalrik>

(2)
_OLRIK>;
<Samuraï> <dotent> <Olorik> <with> canon_laser>;
<Samuraï> <déplace_vers> <stó>.

All this requires internally a representation of the world in the formable form by an automatic reasoning system: an inference engine capable of carrying out deductions from facts or hypotheses, logical system of evidence, etc. The representation of knowledge, their formal manipulation, the production of logical reasoning systems on his knowledge, have been and are still great challenges of artificial intelligence.

2.2.3. Location, cartography, forms recognition

Locate Orlik, admitting that the robot does it by its visual channel, requires advanced image analysis and scenes. It's not just about spotting the Olrik form in a fixed image even if this one task already poses important research problems. It will also be a question, probably, of mapping the space in which the characters evolve, to locate in this space, to identify objects or special places. If the treatment of fixed 2D image is an arduous task, it is the same for series of images taken by a camera; In this case the Samurai robot has two eyes, we can therefore imagine that its two cameras provide it with stereoscopy, depth information can be used to assess distances at benchmarks. Be that as it may, image processing algorithms and forms recognition will be used in order to provide the Samurai with the information he needs to achieve his goal. One can imagine that the robot has a slam algorithm, the current standard in the matter. All these tasks, forms recognition, location and automatic cartography, are important current research subjects, especially for robotics.

2.2.4. Contemporary complement: the semantic web

The web did not exist in 1971-1972, when Edgar P. Jacobs created this adventure of Blake and Mortimer; In 2011, the author would undoubtedly have given his robot the ability to connect to the web and do research (for example to help the real -time mapping of the places crossed) to better analyze the landmarks , objects encountered on the way. With limited electrical energy, the Samurai would be interested in knowing the location of loading terminals on its career. A simple research on the web is not enough to provide the desired information (for example, a photograph entitled "This is not a loading terminal" would be erroneously selected by most current search engines, which cannot based on the presence of words in the page). To obtain reliable information, it is preferable to make a request in a database where the information is correctly identified and indexed; The purpose of searches on the semantic web is to allow machines to question the web as if it were a huge database, where pages are annotated by category information (semantic information). Automatic web exploitation and the advent of this semantic web also constitute very important and very active areas of today's artificial intelligence.

2.3. Algorithmic and learning

This section is devoted to the design of basic algorithms for the realization of smart systems. At the heart of all these systems, algorithms obviously play an essential role. The quality of the algorithms will depend on performance, both in terms of efficiency (obtain a response in a determined time, and as short as possible), as well as that of the accuracy of the results (measured for example by the number of errors in classification, or the number of non - detection and false positives in alarm treatment).

The algorithms we need are very numerous; The systems conducting reasoning performs research in trees or graphs, the exploration of these structures requires the development of

algorithms and heuristics (empirical problem solving methods) adapted, this was a major concern and An important contribution of artificial intelligence research since its origins over fifty years ago; Numerous decision-making problems, allocation of resources, exploration of complex spaces, resolution of constraints, are NPCOMPLETES (non-deterministic polynomial), that is to say that the time necessary for their resolution grows very quickly Depending on the size of the problem, which makes it almost impossible to solve large problems, even with the most powerful supercomputers: it is important to design algorithms capable of giving a good quality solution, approximated from the solution Ideal, in a limited time.

We can also be interested in so -called "Anytime" algorithms capable of providing an approximate or partial solution at any time of their application, if time constraints are imposed on the problem. In the field of NP-Complet, the SAT problem, which is interested in the satisfability of Boolean formulas, is very studied because it being at the heart of many applications, for example in formal verification or planning.

Algorithmic also plays a very big role for automatic learning problems. This scientific discipline is concerned, according to Wikipedia, “by the development, analysis and implementation of automation methods which allow a machine (in the broad sense) to evolve thanks to a learning process, and thus to fill out Tasks that it is difficult or impossible to fill by more classic algorithmic means ”. It is a question of designing efficient learning algorithms (going on the scale of thousands or even millions of data), reliable, not very sensitive to the noise in the data and capable of generalizing. Important theoretical results on the complexity of the learning task, on the limits and capacities of algorithms, with practical benefits on how to carry out systems, have been obtained for twenty years, such as the theorem From Vapnik-Chervonenkis on the limits of digital learning, but the subject remains a very important field of research.

Many problems can be expressed as optimization problems whether digital, symbolic or hybrid. The optimization algorithmic is very varied, conditioned by the famous theorem of the no free lunch which indicates that no optimization method can be better on all the problems: a method can be better than for a class of given problems. The class of optimization algorithms has purely digital approaches in general with gradient, methods of exploring graphs, stochastic methods such as simulated reception, evolution -based methods such as genetic algorithms, and still others.

The signal processing algorithmic is a vast crucial domain for systems having to carry out formation, classification or classification, or trend analysis. In treatment of handwriting or audio signal, especially for speech, hidden Markov models (Hidden Markov Models, HMM) and their variations are widely used to identify phonemes and pronounced words. In the image field, whether 2D or 3D, the richness of processing algorithms is such that the interested reader will rather refer to the website of the ISIS research group (information, signal, image, vision).

Finally, it is important to be interested in the algorithmic of collaboration between artificial agents: in particular in multi-agent systems, the fact that artificial entities Of different origins must communicate and carry out tasks together has led the community to develop adapted formalisms and methods: for example for questions of confidence between "foreign" systems, or negotiation between entities with different or opposite objectives. To go further than this short introduction to the algorithmic, an excellent starting point is the "course" page of the French association of artificial intelligence which brings together around thirty pointers to courses given by members of The community, whose very good booklet produced in 2001 by Irit "Artificial intelligence, but finally what is it? ».

2.3.1. Learning algorithmic

The few projects presented in this part deal with basic algorithms for automatic learning.
The ASAP project addresses the problem of the best possible representation to make learning more effective, and this by adopting a "deep" approach which consists in building representation as an assembly of elementary bricks. The expected results are new automatic methods for extraction of characteristics. Lampada is interested in the questions of scaling for automatic learning methods for the case of structured data such as biology or web 2.0. He works both on the problem of data representation, and on the improvement of incremental statistical methods (which increment a model learned according to the arrival of new data). The Young Researcher Instruit project is also in the field of statistical learning, by targeting automatic translation problems: learning provides a probabilistic translation of translations from a set of bilingual texts; The table is then to produce the most likely translation of a new text. The results obtained are of excellent quality as evidenced by the many references and publications obtained. Two projects are devoted to classification, that is to say the determination of learning of classification systems from examples, and to storage of examples in the classes thus identified.
Classl addresses the cross classification, which consists in simultaneously obtaining a classification of data according to two dimensions, that of the examples and that of the variables. The project mobilizes researchers from different STIC communities (statistics, data analysis, apprenticeship and IT), with a company interested in exploiting the results on mass distribution data. Finally, the MGA project continues the theoretical advances on probabilistic graphic models, otherwise called Bayesian networks, models widely used in the field of decision -making, with new results on model learning, on estimate, and Inference with Bayesian networks, and applications on tasks that range from word processing to bioinformatics.

2.3.2. Learning challenges

The projects of the previous part put the learning algorithmic at the heart of their concerns. In this part, we bring together four projects which contribute to improving learning technologies by targeting a specific and original application framework, considered to be the bearer of new challenges.
Thus, the white bacon project arises the very generic problem of the learning combined with reasoning and decision -making for an artificial agent like a robot, which captures logical, digital and symbolic information of its environment, and while taking count the concepts of uncertainty on the data.
The Cognilego project is combining supervised learning and self-organizing topological cards (an unopensed system), for handwriting recognition, which is a possible alternative to the Gaussian HMM models mentioned above. In a very different field, Sumacc works on supervised learning by developing active learning -type techniques (the learning system seeks to obtain additional data to improve its performance) for the problem of detecting multimedia concept on the web for the old technology.
The Cosinus Siminole project (simulation, inference, optimization, learning) uses learning to help improve digital simulation systems requiring intensive calculation infrastructure, ultimately targeting the reduction in the number of simulations necessary for identifying A physical process, in this case in particle physics.

2.3.3. Multi-agent systems

The three projects on multi-agent systems algorithmic (SMA) focus on very complementary aspects, which of course represent part of the SMA problem in general. The rest of this

notebook has other approaches using SMAs, but in a less fundamental or not in order to develop generic algorithms.
The SMAs are distributed systems: the FACOMA project was to make an SMA reliable by an agent replication mechanism, that is to say the production of copying of the agents on several machines, this thanks to a specific layer of intergiciel ensuring optimal replication in a dynamic way.
The question of confidence between agents was dealt with by the Fortrust project, the aim of which was to produce a complete formalization of the concepts of trust and reputation, in a logical framework, to allow artificial agents to reason on their reciprocal beliefs, With an example of use for the Wikipedia encyclopedia.
The problem dealt with by the Young Coca Researcher's project is that of competition between several agents for the use of a common resource: this is a classic problem of game theory, which is addressed here by means of methods of 'Combinatory optimization, which can highlight more effective global solutions than letting each agent seek its local optimum independently of others.

2.3.4. Complexity, logic

This part affects algorithmic, with very generic projects that approach essential theoretical points on which it is important to continue to progress. However, we are never far from applications, because the potential benefits of an advance on these base issues are very large.
Thus, the NAFIT project of the 2008 Defis program, collaboration between the Marseille LIF and the Poncelet de Moscow laboratory, works on fundamental questions in information theory, in random, ergodic systems, in complexity of Kolmogoroff. The theoretical results obtained could one day find their application in new cryptography systems, or in new algorithms for very large search engines like Google.
The Iomca project, which brings together French and Taiwanese partners, is interested in improving Monte Carlo Treech techniques, used in planning problems or in games, and taking into account the uncertain. Two notebooks are on the SAT theme (satisfyability): the young PSI researcher project combines a SAT approach with a decomposition approach to Problems to automatically generate programs evidence; In particular, he carried out the interface between the Environment of Evidence Coq, widely used in the field of codes, and a SAT solver.
The UNLOC project has won several awards in recent SAT competitions, working on resolution efficiency and seeking to produce the shortest possible evidence, with new measures that are largely taken up by the international community.
On the sidelines of this section, the LOCI project is interested in the foundations of logic, more precisely in its declination called "playful", based on the notion of play; He confronts it with the observation grounds that are sign language and exchanges on web, in order to design new logical models of interaction.

2.4. Robotics, real world confrontation and bio-inspiration

The domain of robotics has many dimensions and covers a very large scientific and technological field. This discipline associates not only mechanics and mecatronics, electronics (sensors such as perception, vision, actuators), controller, architecture, system engineering, but also communication both between robots and 'With humans by integrating cognition, analysis and expression of behaviors and emotions. Collaboration between artificial entities or with human operators (cobotics) is a very important field of research in which security problems are essential. This is, among other things, one of the fields of sticns where bio-inspiration has taken an important place, many teams seek to "imitate life" to design more flexible, more adaptive robots, equipped with specific features (crawl , swim, fly ...).

Finally, this is an area where applications play a very large role as much from an economic point of view - for example with the vast market of industrial robotics, or with that emerging from personal assistance - than from point of view of Research, because any new science or robotic technology developed is quickly confronted with reality, the robots being located in the real world and in interaction with it. This immediate return of reality inevitably generates new research challenges. The CNRS robotic research group is organized according to the main ones of robotics:

- Medical robotics
- Autonomous vehicles (land and air)
- Robotic manipulation at different scales
- Advanced control architectures of robotic systems
- Interactions between robotic systems and users
- The design of innovative mechanical and mechanical architecture
- Humanoid robots
- Neuro-robotics

As it was specified in the introduction, in this notebook we are only interested in the part "cognition, perception, collaboration" of robotics; The mechanical or technological aspects of the basic system-in particular mechanical, control and control, sensors, actuators, architectures and "low" software platforms-being excluded because it is too distant from the main subject that are intelligent systems. The subdomains concerned are those underlined above, knowing that only part of medical robotics is present in the following projects.
The action of the ANR on robotics was strongly visible from 2006 with the creation of the PSIROB program (interactive and robotic systems program), which funded a small thirty projects on its two years of existence. Subsequently, from 2008, several Stic programs supported research in robotics: arpege (on-board systems and large infrastructures) for mecatronic layers, system, control-command, conting (content and interactions) for layers of perception, Cognition, collaboration, interaction with humans. Some more ruptive robotics projects were supported by the Defis (Emerging Domains) program between 2008 and 2009, as well as in non -thematic programs. This distribution of robotics between two major programs was preserved in the 2011 programming, as illustrated in a very caricatural above, between Contint (digital content and interactions) for the head of robots, and Ins (digital engineering and security), For their legs. Throughout the period, the TECSAN (Health Technologies) program has supported robotics research for the needs of this great application that is health and in particular personal assistance and assistance to the operating gesture . Some other projects on human-robot cooperation have been funded in the human and social science programs.

2.5. Recognition of forms, excavation and search for images

Considerable progress has been made in recent years in the field of computer forms and artificial intelligence around multimedia images and videos data, namely, among other things, recognition and location of objects or classes of classes objects present in an image, indexing and searches of images for classification or categorization of content, and research images by content. This progress has been made possible by the contribution of new concepts and techniques but also the use of many methods and tools coming from applied mathematics (statistical, parametric, SVM, boosting, k-men, etc.) to facilitate data analysis. We can note others the rapid evolution of recognition techniques by automatic learning. These learning techniques will learn the appearance of an object or a category of objects in set images by analyzing its different visual characteristics without definition of formal and explicit model of the desired object. The projects presented here are varied and go from fundamental research

on methods and algorithms to more finalized research with the implementation of information systems.

2.5.1. ANR projects [5]

- Young Researchers' Program - Young Researchers, 2010 edition Eviden project Visualize the functioning of the cell to better to understand.
- Masses of data and ambient knowledge, 2007 edition EFIDIR project How to measure from space deformations of the terrestrial surface.

2.6. Automatic language and speech processing

Automatic Language Processing (TAL) concerns the application of computer programs and techniques capable of automatically processing human language, written or spoken. This field of multidisciplinary research combines linguistics and computer science, and makes more extensive use of pattern recognition, signal processing, statistics, logic, documentary analysis or even psychology to form one of the major fields of artificial intelligence. The applications are very numerous. Some are already in common use: spelling and grammar checkers, monolingual and interlingual search engines, voice dictation, talking GPS, interactive voice servers, automatic translation. Others are beginning to spread, such as querying search engines by voice or automatic transcription and indexing of audiovisual content. However, there are still many challenges to overcome before being able to automatically understand all the subtleties of human language and establish a natural dialogue between man and machine. To advance towards this objective, it will be necessary to further extend the field of models, in particular to fully take into account the semantic and pragmatic levels, paralinguistic information (prosody, expression of emotions) and other communication methods (processing of multimedia documents, communication multimodal). This presupposes not only research work, but also the creation and making available to the scientific community of corpuses of data representative of the phenomena to be studied, as well as the organization of joint evaluations allowing researchers to compare their approaches in order to mutually enrich their knowledge.

2.7. Knowledge, Web, Semantic Web

The web has become in twenty years one of the most frequented places of the planet: initially designed as a platform for exchanging information between scientists, it was first transformed into a vector for the dissemination of scientific, technical and commercial information, before becoming what we know today: the universal system communication used for all our activities whether professional, commercial, social or private. The Web is an asset and a challenge; a wealth due to the quantity and diversity of information stored there, and circulating there; a challenge because this information is massive, multifaceted, heterogeneous, not structured, and from a wide variety of sources. Artificial intelligence researchers have quickly seized the opportunity presented by this universe to apply and develop their tools previously reserved for smaller sectors.

2.8. AI APPLICATIONS

This last section is devoted to projects directly targeting applications of strong societal interest. If new methodological or algorithmic developments are made, they are necessarily in connection with the targeted field of application: environment, transport, security, health, assisted living.

It is very difficult to take stock of the concrete applications of artificial intelligence in our society: as we will have seen in the previous sections, it is a multifaceted field and interacts with many other disciplines, which makes it almost impossible to trace the concrete

consequences. The American Association of Artificial Intelligence organizes an annual conference “Innovative Applications of Artificial Intelligence” which seeks to highlight particularly interesting applications, generally within a framework of cooperation between academic laboratories and companies.
The history of these conferences can be found at http://www.aaai.org/Conferences/IAAI/iaai.php. For example, applications highlighted in 2011 include systems:

- automation of the distribution of short news;
- active learning for military decision-making and planning;
- estimated energy consumption;
- detection of falls by elderly people;
- smart diabetes management;
- *etc*

Since its origins, artificial intelligence has sought to confront the real world, and there are many historical examples: mining research, medical diagnosis, configuration of technical devices, process monitoring, planning of satellite missions, and detection of faults in pieces.
The period of the 1970s-80s saw the development of quantities of expert systems in all economic sectors, in some cases with success, even if it was followed by a period called "AI Winter" of about fifteen years. Due to the inevitable disappointment with the overly high expectations and promises of the initial period. It was then that specialists began to speak of "hidden" or "embedded" artificial intelligence to illustrate the fact that the intelligence component of a system is part of a whole and is no longer part of it. Necessarily the most visible part. This is one of the common characteristics of the twenty projects in this section, which therefore address the various societal challenges mentioned above using artificial intelligence technologies.

This chapter is presented in the form of mini projects so that the student masters, recognizes and covering a diverse number of artificial intelligence applications in control and command.

- environment, transport
- Security
- Health
- Assisted Living…ect

2.8.1. Exemple 1: Discover how artificial intelligence systems are used in the world of transportation and integrated into autonomous vehicles.

Artificial intelligence (AI) may seem like something new. But AI applications have been used in transportation for some time now. Many modern vehicles use a satellite tracking system (GPS). This system uses data from satellites to establish where a vehicle is on Earth. Mapping algorithms use AI to determine the best path to get from point A to point B.

To achieve this, AI systems have learned to predict the best trajectories from immense amounts of data. They then combine this data with real-time information about users. This includes information such as how fast they are traveling on the road. These two types of data can then provide people with accurate and precise information about their movements. AI can even help drivers navigate traffic congestion and avoid road construction sites.

Several safety features in modern vehicles use AI. Driving assistance is an example. Driving assistance systems warn the person driving of possible dangers. This could be, for example, an audible alarm that goes off when the car veers out of its lane. To do this, the car uses various sensors, including cameras and infrared sensors.

Some systems also help with driving. These can be specific functions, such as control systems that adjust the vehicle's speed or direction. It can also be more general functions, such as using machine learning models to make decisions based on different traffic conditions. All of these functions send data to a central data center. This information is then used as training data for future models.

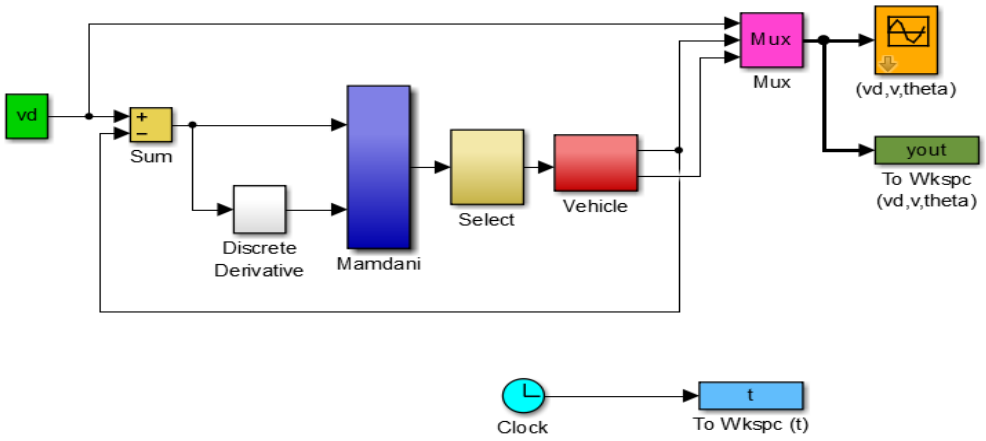


Figure 2.4. VHE control based Fuzzy Control (Vd= 45 m.s^{-1}).

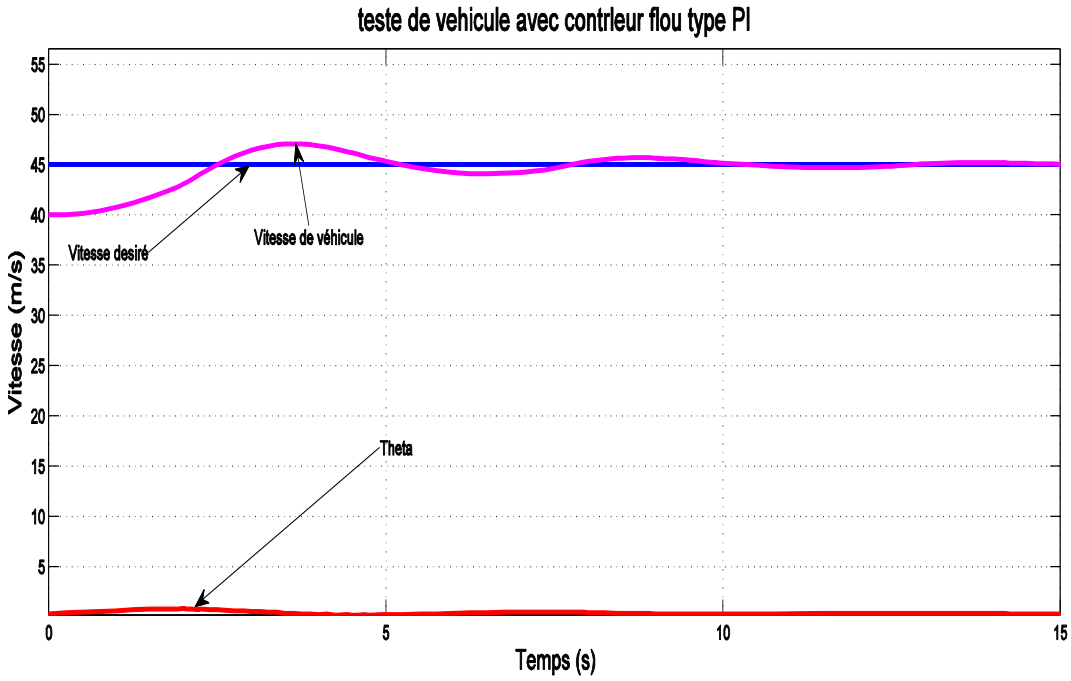


Figure 2.5. Speed Control based floue and PI Control.

Example 2: ABS braking system with different controllers (P,PI, PID and Fuzzy)

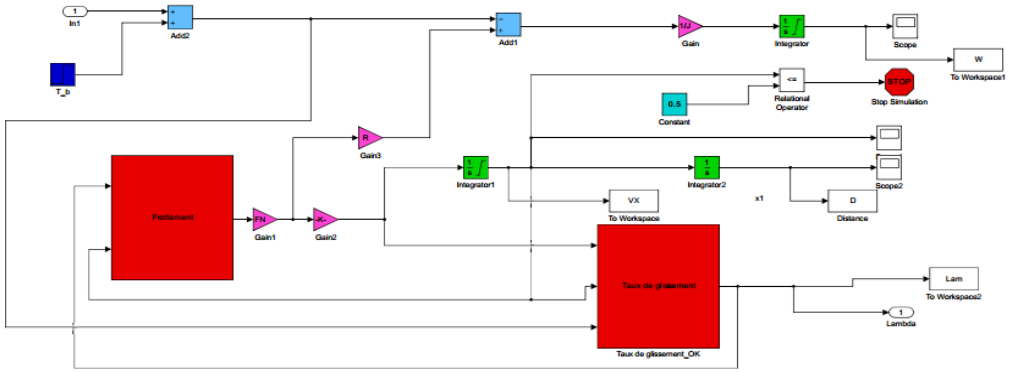


Figure 2.6. ABS model without controller.

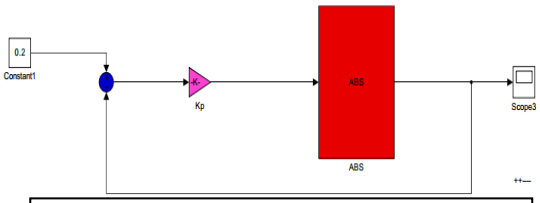


Figure 2.7. ABS model with P controller.

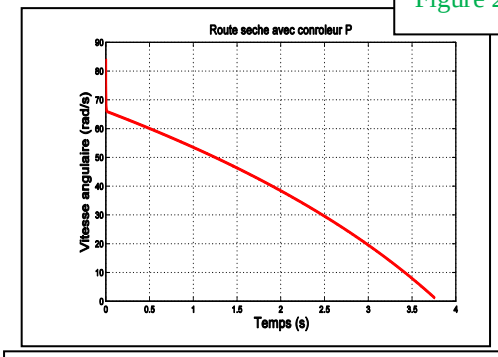


Figure 2.7.1. Angular speed of the vehicle with P controller

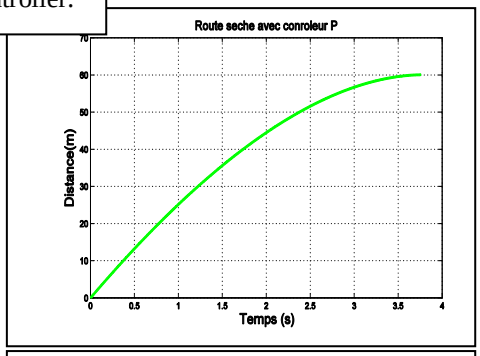


Figure 2.7.2. Braking distance with P controller

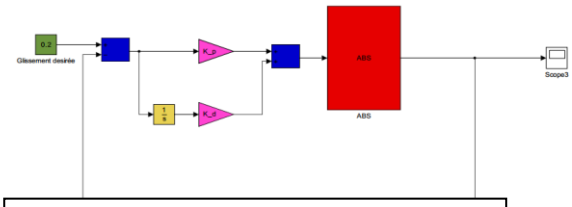


Figure 2.8. ABS model with PI controller.

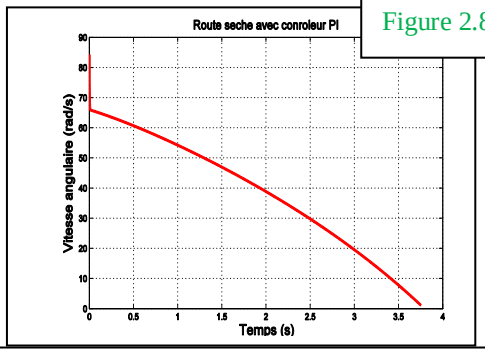


Figure 2.8.1. Vehicle Angular Velocity with PI Controller

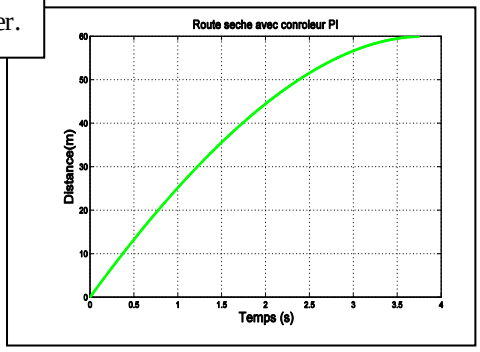


Figure 2.8.2. Braking distance with controller PI

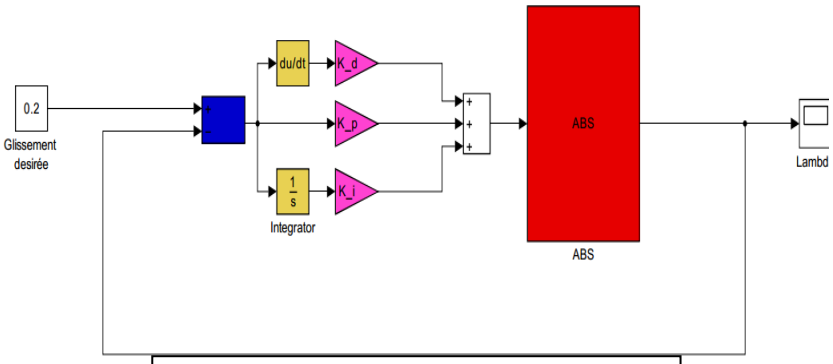


Figure 2.9. ABS model with PID controller

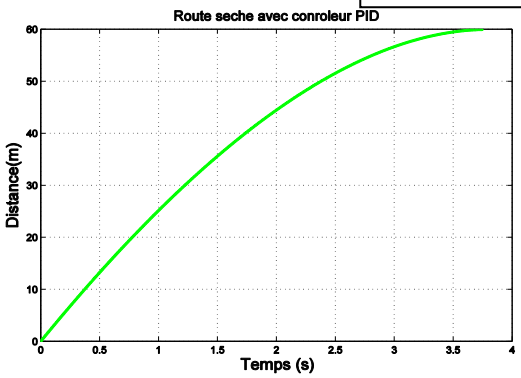


Figure 2.9.1. Braking distance with PID controller

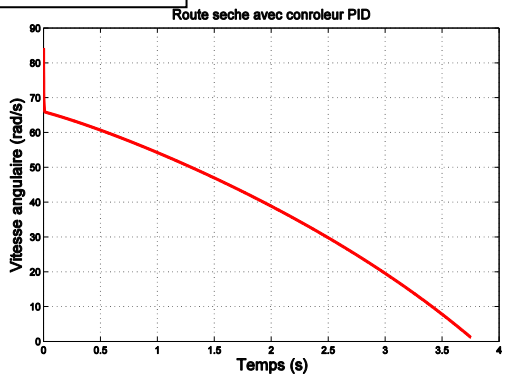


Figure 2.9.2. Vehicle angular speed with PID controller

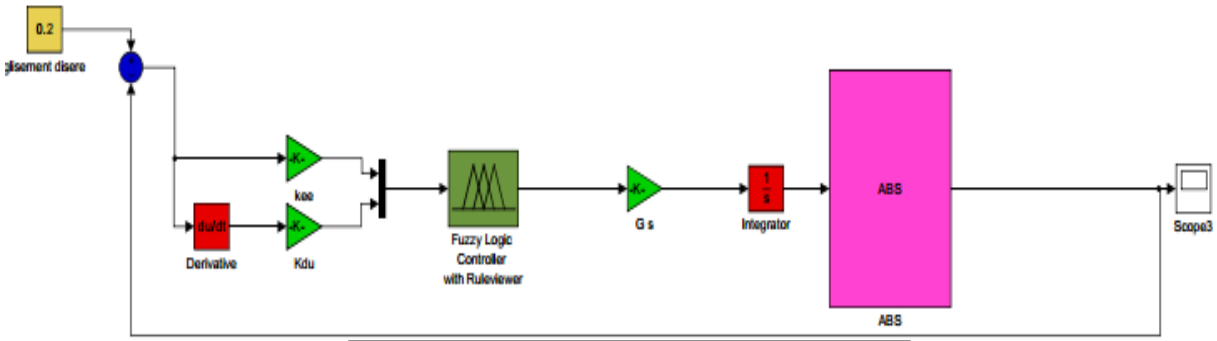


Figure 2.10. ABS model with Fuzzy controller

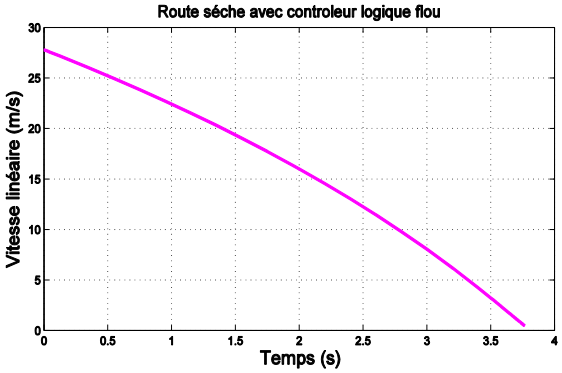


Figure 2.10.1. Vehicle Angular Velocity with Fuzzy Controller

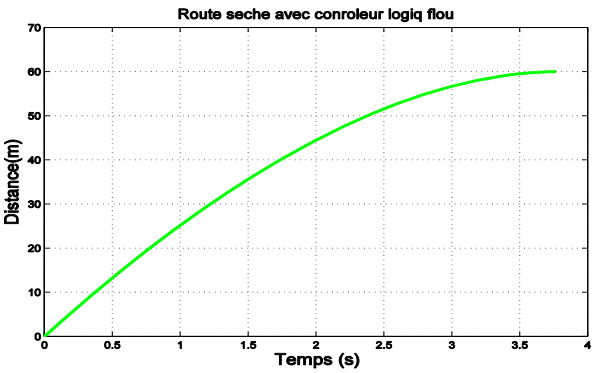


Figure 2.10.2. Braking distance with Fuzzy controller

Example 3: AI and traffic monitoring

Nobody likes to be stuck in traffic! Urban planners are therefore always looking for ways to improve vehicle circulation on the roads. Installing sensors on traffic lights can help. The sensors send data to a large remote database. This data is then used to develop different traffic light control scenarios which are analyzed to determine the best settings. City planners are also using machine learning to design better road systems. This can include ideas like using roundabouts instead of traffic lights.

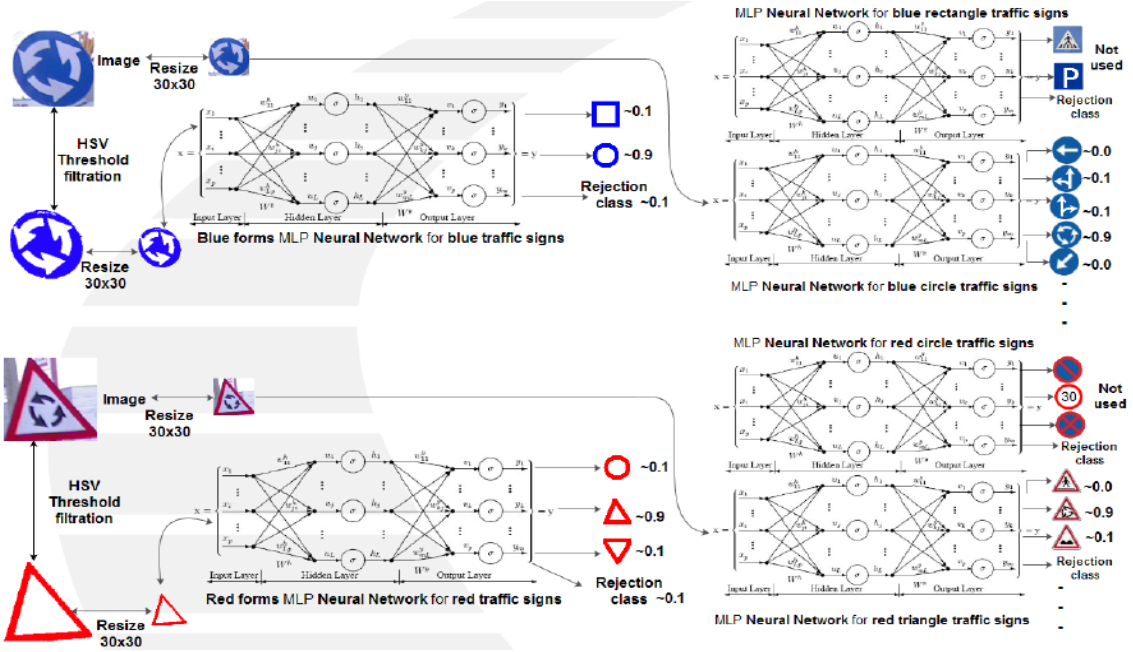


Figure 2.7. Recognition of traffic signs based Neural Network (NN).

Example 4: Object detection (LIDAR) allows you to deduce the distance of surrounding objects, creating a 3D map.

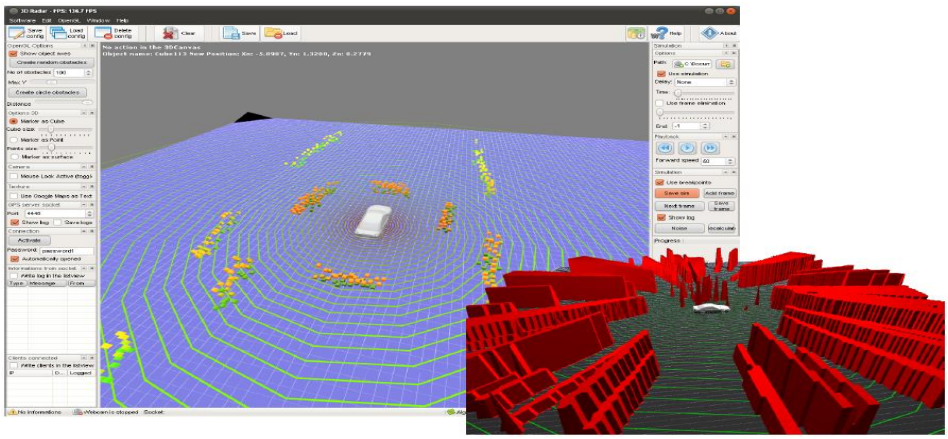


Figure 2.8. LIDAR. [6]

Example 5: How do autonomous vehicles work?

To drive itself, a car needs equipment and software. Equipment is a set of sensors and mechanical parts. It allows the car to perceive its environment and obtain the data necessary for automatic decision-making. It’s like the eyes, hands and legs of the person driving. Software is computer programming. They allow the car's computer to make decisions. It’s like the brain of the person driving.

Self-driving cars use many technologies to perceive their environment. This includes high-definition cameras, ultrasonic sensors, and radar and lidar sensors. These allow the car to detect traffic lights, cyclists, or even a squirrel crossing the street! Radar uses radio waves to detect objects. Lidar works like radar, except it uses pulses of light to detect objects. These last two instruments make it possible to complement the visual information of standard cameras. This is particularly useful when weather conditions reduce visibility.

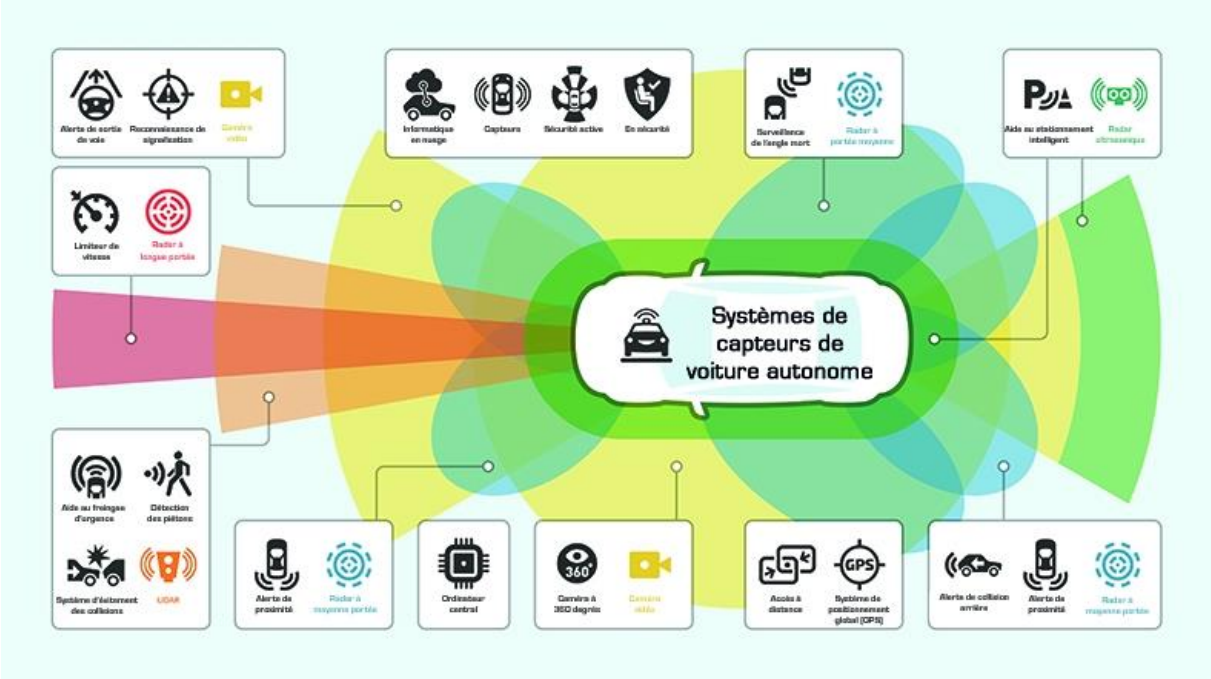


Figure 2.9. Autonomous car detection system. [6]

2.9. References

[19] Intelligence Artificielle et Robotique «Confluences de l'Homme et des STIC», cahier de l'ANR.
[20] https://parlonssciences.ca/ressources-pedagogiques/documents dinformation/intelligence-artificielle-et automobiles

Chapter 3: Supervised and unsupervised classification

3.1. Introduction

The purpose of this chapter is to present a general view of the problem of *CLUSTERING*, by introducing the basic notions and concepts and to highlight the diversity that exists among the different classification methods.

We begin this chapter by reminding some essential concepts and definitions to understand the different methods and tools of automatic classification. We then present the basic steps of the automatic classification procedure as well as the different possible applications of clustering. Then, we discuss the different notions that are used to define the similarity between objects, which constitutes the basis of any clustering method. Finally, we present the existing approaches allowing evaluating the results of clustering algorithms. Further details on the main clustering methods are then presented.

3.2. Useful concepts and definitions

Unsupervised classification or automatic classification - clustering - is an important step in data analysis; whose objective is to identify groups of similar objects called clusters of a data set without knowing their structure beforehand.

- ***Definition of a partition***

Given a finite set of objects denoted *I*, we call partition of *I* any family of non-empty parts *P* disjoint two by two whose union forms the set *I*.

$P = \{C_i,\ i \in I\}$, such that C_i: part of *I* (or a class) having the following properties:

$$\begin{aligned} &1.\forall i \in I, C_i \neq \Phi \\ &2.\forall i \in I, \forall j \in I, \quad i \neq j \rightarrow C_i \cap C_j = \Phi \\ &3.\bigcup_{i \in I} C_i = I \end{aligned} \qquad \ldots\ldots(3.1)$$

- ***What data is processed in classification?***

The data processed in classification can be images, signals, texts, other types of measurements, *etc*.

- ***What is a class?***

Basically, a class (or group) is a set of data formed by homogeneous data (which "resemble" in the sense of a criterion of similarity (distance, probability density, etc.)). For example, a class can be a region in a color image, a particular event in a sound signal, the spam class and non-spam classes in the case of spam detection in an email, *etc*.

- **How many classes?**

The number of groups (denoted by *K*) in prediction is assumed to be fixed (given by the user). This is the case, for example, if one is interested in classifying handwritten letters (number of classes = number of characters of the alphabet), *etc*.

3.3. Supervised or unsupervised classification

The classification of data located in a high-dimensional space is a delicate problem that appears in many sciences whose general objective is to be able to label data by assigning them

a class. There are two types of approaches: supervised classification and unsupervised classification. These two approaches differ in their methods and their purpose.

Classification is the most widely used descriptive technique. There are many classification algorithms. The objective of a classification is to distinguish distinct subsets (or classes) in the starting population.

Remember that classification differs from classification in that the classification criteria are not known a priori (before studying the population). It is the population that determines the criteria.

Classification makes it possible to limit the number of variables per subset. Very discriminating or too little discriminating variables can be eliminated.

Classification makes it possible to find correlations specific to each class and therefore more precise.

Take care: there is no single solution to the problem of classification. In other words, there is no good classification, but several possible classifications.

We talk about class, segment or cluster to talk about both the extension (the individuals) and the intension (the variables and their possible values) of the subsets defined by the classification.

3.3.1. Supervised classification

The classes are known and there are examples of each class. In supervised mode we have a learning sample whereas in unsupervised mode, we do a blind search; the difference between the two situations is the knowledge of the classes.

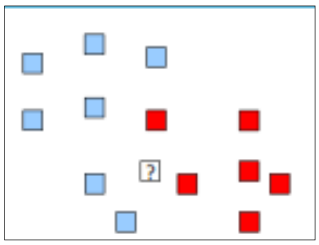
If the possible classes are known and if the examples are provided with the label of their class, we speak of supervised classification (eng. classification), the objective is then to learn using a learning model. from the set of examples (called the learning set) of the rules which make it possible to predict the class of new examples, which amounts to discovering the structure of the classes in order to be able to generalize this structure on a larger set of data.

An example of the application of the supervised classification concerning cars can be taken from (Candillier, 2006), or it can be a question, for example, of determining whether a new car encountered belongs to the class of city cars, intermediate cars or comfortable cars, based on characteristics, and on the known class of cars already encountered (learning examples). Table .1 presents the problem in this context. Each example is associated with the class of car to which it belongs. The objective is then to be able to estimate the most appropriate class for any new example encountered (*e.g.* car 7 in table. 1).

Table3.1-- Example of supervised learning application

Identifier	fuel	cylinders	length	Power	Class
1	gpl	8	186	6000	comfort
2	Essence	4	170	5800	intermediate
3	diesel	6	172	5500	intermediate
4	diesel	4	156	5200	citadin
5	Essence	12	190	5500	comfort
6	Essence	4	175	5800	intermediate
7	diesel	6	170	6000	?

Example: articles in economics, politics, sport, culture...
• We want to classify a new element
Example: assign a label among economics, politics, sport, culture...



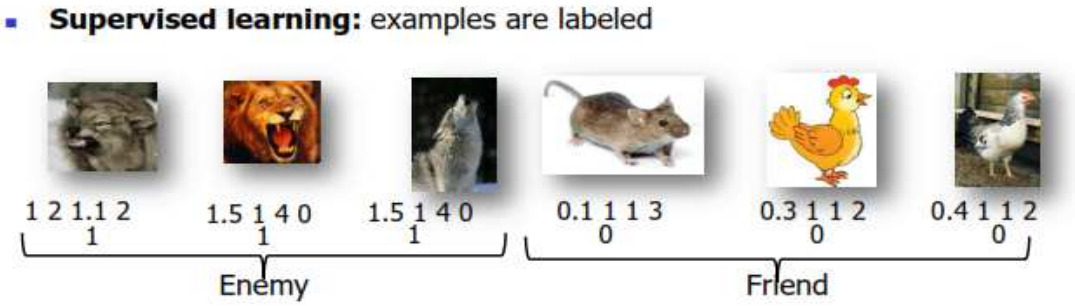


Figure 3.1. Supervised learning exemple.

Applications: Some examples of supervised classification applications are presented below:

- Application 1: An archaeologist seeks to determine whether human remains are those of a man or a woman.
- Application 2: In a bank, a credit committee must decide, based on financial parameters, whether or not to grant a loan to an individual.
- Application 3: Given a set of symptoms, a physician must make a diagnosis.
- Application 4: In industry, we want
  - Identify faces, fingerprints,
  - Identify objects in video sequences,
  - Search for potential customers in databases,
  - Match one or more words in a relevant way to the most relevant text.

3.3.2. Unsupervised clustering

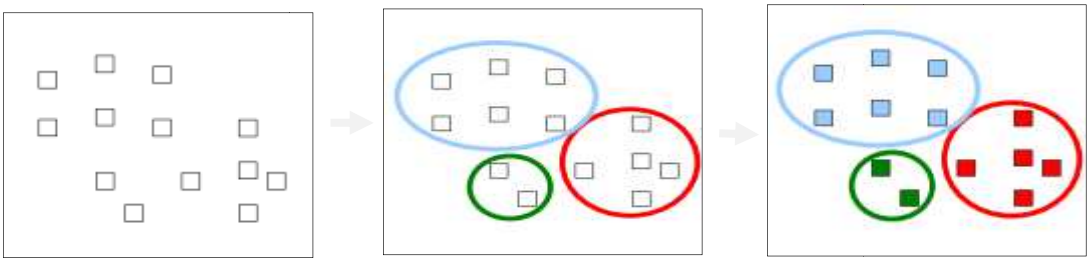
If only the examples, without labels, are available, and if the classes and their numbers are unknown, then we speak of unsupervised classification also called "automatic classification", "clustering" or "grouping".

Unsupervised classification (ang. *clustering*) consists in dividing a set of examples into subsets, called classes (clusters), such that the objects of one class are similar and the objects of different classes are different, in order to understand its structure (Blansché, 2006). In other words, it is a question at this level of seeking the underlying distribution of the examples in their description spaces. As the table 2 shows, there is no class information associated with the examples in this case (Candillier, 2006).

Table. 2--Example of unsupervised learning application

Identifier	fuel	cylinders	length	Power	Class
1	gpl	8	186	6000	?
2	Essence	4	170	5800	?
3	diesel	6	172	5500	?
4	diesel	4	156	5200	?
5	Essence	12	190	5500	?
6	Essence	4	175	5800	?
7	diesel	6	170	6000	?

We have unclassified elements, Example: words of a text
• We want to group them into classes
Example: if two words have the same label, they are related to the same thematic...



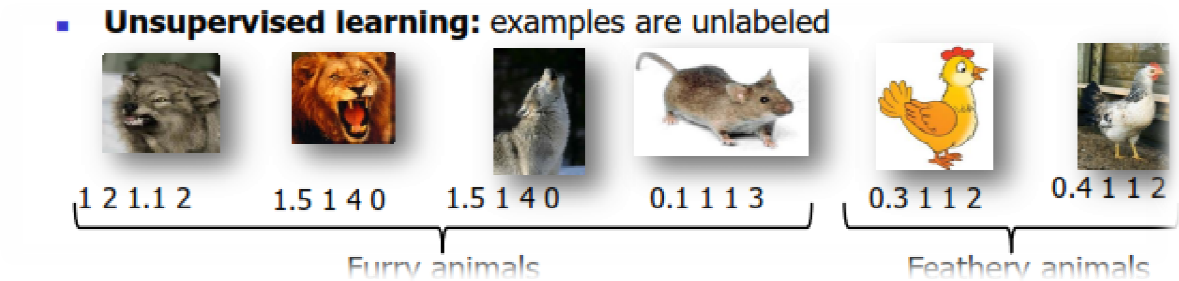


Figure 3.2. Unsupervised learning exemple.

Applications: Some examples of applications of unsupervised classification are presented below:

- Application 1: In biology, we want to group species according to their characteristics and therefore their common origins.
- Application 2: In psychology, we want to classify individuals according to their personality type.
- Application 3: In chemistry, we want to classify compounds according to their properties.
- Application 4: In industry, we want
  - Analyze survey results,
  - Identify potential customers of a company, identify customers likely to compete,
  - Determine sales locations (installation of cash dispensers, *etc.*),
  - Analyze, identify risks (water damage, *etc.*),
  - Analyze textual data.

3.3.2.1. The steps of an automatic classification

The basic steps of the automatic classification process can be summarized as follows: Feature selection/extraction.

- Feature selection is the process of identifying an optimal subset of relevant source features of relevant original features for a previously set criterion for use in clustering. The selection of this subset of characteristics makes it possible to eliminate irrelevant and redundant information according to the criterion used.

- While feature extraction aims at using one or more transformations of the input features to produce new salient features. Either or both of these techniques can be used to obtain an appropriate set of features for use in clustering.

3.3.2.2. Automatic classification algorithm

The clustering or classification step can be done in several ways. Data classification (or clustering) can be hard (a partition of data into groups) or fuzzy (where each model has a degree of membership in each of the output clusters). Thus, our goal through this step is to choose the most appropriate clustering algorithm for clustering the dataset where each clustering algorithm is characterized mainly by a proximity measure and a clustering criterion.

- The proximity measure is a measure that quantifies to what degree any two data points are "*similar*" (feature vectors). In most cases we need to ensure that all chosen variables contribute equally to the calculation of the proximity measure.

- Grouping criterion can be expressed by a cost function or other type of rules. It is necessary to take into account the type of clusters expected by the grouping of the dataset. Thus, we can define a "good" grouping criterion, leading to a partitioning that represents the dataset as best as possible.

- Validation of the results, the grouping algorithms make it possible to extract clusters which are not known a priori. Also, different approaches usually lead to different groups and even for the same algorithm. Therefore a final classification of a data set requires some kind of evaluation in most applications. The accuracy of the results obtained by the clustering algorithms is verified using appropriate techniques and criteria.

- Results interpretation: The ultimate goal of aggregation is to provide users with meaningful insight into the original data, so that they can effectively troubleshoot any issues encountered.

3.3.2.3. Applications of clustering

Unsupervised classification has been used in several fields, ranging from engineering (machine learning, artificial intelligence, pattern recognition, mechanical engineering, electrical engineering), computer science (webmining, spatial database, collection of textual documents , image segmentation), medical and life sciences (genetics, biology, microbiology, paleontology, psychiatry, clinical, pathology), earth sciences (geography, geology, remote sensing), social sciences (sociology, psychology , archaeology, education), and economics (marketing, trade).

This diversity reflects the group's important position in scientific research. On the other hand, this diversity can be a source of confusion, due to the different terminologies and objectives. Clustering algorithms have been developed to solve particular problems, in specific domains, and they are usually based on assumptions and guesses about the data set to be processed. These assumptions inevitably affect the performance of these algorithms in other problems that do not satisfy these assumptions. For example, K-means algorithm based on Euclidean distance and hence it tends to generate hyper spherical clusters. But if the real clusters are in another geometric shape, K-means may not be efficient, leading to look for other clustering algorithms. In the following, we describe some application domains where clustering has been used as an essential step.

- Segmentation (This technique is very useful in image segmentation which can be defined as an exhaustive partitioning of an input image into several regions)
- Pattern recognition and,
- Data mining (Aims to deal with large databases which place additional computational demands on clustering. The birth of clustering in data mining is mainly due to the intense developments in the fields of information retrieval and text mining, space databases e.g. astronomical data, data analysis, web applications, DNA analysis in bioinformatics, and many other specific applications).

3.4. Aggregation criterion

The aggregation criterion makes it possible to compare the classes two by two to select the most similar classes according to a certain criterion. The most classic criteria are the nearest neighbor, the maximum diameter, the average distance and the distance between the centers of gravity.

$$D(C_p, C_q) = \min\{dis(i, j); i \in C_p, j \in C_q\} \quad \text{.....(3.2)}$$

$$D(C_p, C_q) = \max\{dis(i, j); i \in C_p, j \in C_q\} \quad \text{.....(3.3)}$$

$$D(C_p, C_q) = \frac{\sum_{i,j}\{dis(i, j); i \in C_p, j \in C_q\}}{Card(C_p) \times Card(C_q)} \quad (3.4)$$

$$D(C_p, C_q) = dis(G_p, G_q) \quad (3.5)$$

3.5. Similarity measures

The main objective of a classification is to provide homogeneous and well separated groups, in other words groups of objects such as:

- The objects are as similar as possible within a group;
- The groups are as dissimilar as possible.

Due to the variety of feature types and scales, the distance measure (or measures) must be chosen with care. Unfortunately, too often this is an arbitrary choice, sensitive to object representation, and treats all attributes equally. It is more common to calculate the dissimilarity between two objects using a distance measure defined on the feature space (see Table. 3).

An object is described by a set of characteristics, generally represented by a multidimensional vector. The characteristics can be quantitative or qualitative, continuous or binary, nominal or ordinal, which determine the corresponding measurement mechanisms. The distance or dissimilarity function of a data set X is defined to satisfy the following conditions:

1. *Symmetry* $d(x_i, x_j) = d(x_j, x_i)$;
2. *positivity* $d(x_i, x_j) \geq 0$ for all x_i et x_j. If the conditions;
3. *triangulare Inéquality* $d(x_i, x_k) \leq d(x_i, x_j) + d(x_j, x_k)$;
4. Re *flexitivity* $d(x_i, x_j) = 0 \quad si \quad x_i = x_j$ are verified, then it is called a metric.

Likewise, a similarity function is defined to satisfy the following conditions.

1. *Symetry* $S(x_i, x_j) = S(x_j, x_i)$;
2. *positivity* $S(x_i, x_j) \geq 0$ for all x_i et x_j. If the conditions;
3. *triangulare Inéquality* $S(x_i, x_k) \leq S(x_i, x_j) + S(x_j, x_k)$;
4. Re *flexitivity* $S(x_i, x_j) = 0 \quad si \quad x_i = x_j$ are verified, then it is called a metric.

For a dataset with N input objects, we can define a symmetric matrix of size *N X N*, called proximity matrix, whose (*i*,*j*) *th* element represents the similarity or dissimilarity measure for the *ith* and *jth* objects (*i*,*j*=1,...*N*).
Typically, distance functions are used to measure continuous variables, while similarity measures are more important for qualitative variables. Some typical measurements for continuous functions are shown in the table. 3.

Measures	Forms	Comments	Examples and applications
The Minkowski distance	$D_{i,j} = \left(\sum_{l=1}^{d} \left\| x_{il} - x_{jl} \right\|^{1/n} \right)^n$	Metric invariant to any rotation only for *n* = 2 (Euclidean distance). Features with large values and variances tend to dominate other features.	c-means floue
Euclidean distance	$D_{i,j} = \left(\sum_{l=1}^{d} \left\| x_{il} - x_{jl} \right\|^{1/2} \right)^2$	The best known distance, it is only a special case for p = 2 of the Minkowski distance. Tends to form hyper-spherical clusters.	*K*-means
Distance from Manhattan	$D_{i,j} = \sum_{l=1}^{d} \left\| x_{il} - x_{jl} \right\|$	This is a special case for p = 1 of the Minkowski distance. Tends to form hyper-rectangular clusters	ART blurry (Fuzzy)

Chebyshev distance	$D_{i,j} = \max_{1 \le l \le d} \lvert x_{il} - x_{jl} \rvert$	Special case for p=∞ of the Minkowski distance,	fuzzy c-means with Chebyshev distance
Mahalanobis distance	$D_{i,j} = (x_i - x_j)^T S^{-1} (x_i - x_j)$	S is calculated based on all objects. Tends to form hyper-ellipsoid clusters. when the features are uncorrelated, the squared mahalanobis distance is equivalent to the squared Euclidean distance. May cause some computational load.	Ellipsoid ART, hyper ellipsoid clustering algorithm.
Pearson correlation	$D_{i,j} = \frac{1 - r_{ij}}{2}$, $r_{ij} = \frac{\sum_{l=1}^{d} (x_{il} - \bar{x}_i)(x_{jl} - \bar{x}_j)}{\sqrt{\sum_{l=1}^{d} (x_{il} - \bar{x}_j)^2 \sum_{l=1}^{d} (x_{il} - \bar{x}_j)^2}}$	Not a metric. Calculates the distance between an object x_i and a reference point x_r. dir is minimized when a symmetric object exists.	widely used as a measure for gene expression data analysis
Cosine of similarity	$S_{i,j} = \cos\alpha = \frac{X_i^T X_J}{\lVert X_i \rVert \lVert X_J \rVert}$	Independent measurement of vector length. Invariant to rotation, but not to linear transformation.	The most used in the document clustering.

3.6. Evaluation and validity criteria

Clustering is an unsupervised process where data is unlabeled, and no structural information is available. Thus, evaluating the results of clustering algorithms is a very important task. In the clustering process, there are therefore no predefined classes; Moreover most of the algorithms do not provide the means for the validation and the evaluation of the clustering. Therefore, it is difficult to find an appropriate measure to verify the accuracy of the resulting clustering. In this case, several questions can be asked such as:

- What is the optimal number of clusters?
- What is the best cluster?,
- What is the best grouping of data?

In most algorithms, experimental evaluations of the two-dimensional data are used so that the user is able to visually check the validity of the results (i.e. how well the clustering algorithm discovered the clusters of the data set). It is clear that visualization of the data set is a crucial verification of the clustering results. In the case of multidimensional database (eg more than three dimensions) effective visualization of the dataset could be difficult. Furthermore, cluster perception is a difficult task for human beings who are not used to high dimensional spaces.

3.6. 1. Determination of the number of clusters

Automatic classification aims to identify groups of similar objects, thus helping to discover the distribution of objects and interesting correlations in large data sets. However, most clustering algorithms have needed to know the number of classes to search. This is an unsupervised method and in most cases the user will not have prior knowledge of how many

classes are in the dataset, thus leading to separation of data into a number of classes *k* which is usually larger or smaller than the actual number of classes.
If the number of partitioning clusters k is greater than the optimal number of classes (*k'*), then one or more good compact classes can be partitioned. However, if k is less than k', at least one distinct cluster can be merged.

Thus, finding the right number of clusters is a very important problem. For example, if we take the dataset shown in Figure 3.1(a). It is obvious that the optimal number of clusters is three. The partitioning of the previous data set by an unsupervised classification algorithm (e.g. K-means) into four clusters is presented in Figure 3.1(b). In this example the K-means algorithm has found the best four clusters into which the data set can be split. However, this classification is not optimal for the considered database. We define, here, the term "optimal" clustering as the result of running an automatic classification algorithm (i.e., clustering) that best fits the inherent partitions of the dataset.

It is obvious that the clustering shown in Figure 3.1(b) is not an optimal classification of the database. The optimal clustering for this dataset can be obtained by partitioning the data into three clusters. Therefore, if the values assigned to the parameters of a classification algorithm are inaccurate, the clustering method may result in a partitioning that is not optimal for the dataset considered, which leads to false decisions. . The problem of deciding the optimal number of clusters for a dataset as well as evaluating the results of a clustering process have been the subject of several research efforts.

3.6. 2. Core Concepts of Cluster Validity

In this section, we present some basic concepts of clustering validation, as well as the most used validation indices. As we mentioned earlier, the evaluation of clustering results is one of the most important issues in the process of unsupervised classification. It is generally the validity indices that are used to measure the quality of the clustering results.
Two types of indices can be used to assess the quality of a classification: external indices and internal indices. External indices measure the agreement between two partitions where the first partition is known a priori, and the second partition is the result of a clustering process. Internal indices are used to measure the quality of an aggregation structure without any external information. That is, external indices allow the results of a clustering algorithm to be evaluated based on a previously known clustering structure of a data set (or cluster labels).

Whereas, internal indices allow evaluating a partition using inherent quantities and characteristics of the considered data set. The optimal number of clusters, for any classification, is usually determined by one or more internal validity indices. The purpose of such a grouping is to achieve a classification such that the objects within the same class are as similar as possible and the objects of different classes are as dissimilar as possible, in other words, obtain Compact Well Separated Clusters or CBS classes (in English, CWS clusters: Compact Well Separated Clusters):

I. Compactness: several measures based on the variance make it possible to evaluate the compactness of a cluster. Lower variance measure indicates better compactness. Additionally, there are many distance-based metrics to estimate the compactness of a cluster, such as the maximum or average pairwise distance.

II. Separability: Measures the degree to which a cluster is distinct or well separated from other clusters. For example, pairwise distances between cluster centers or minimum pairwise

distances between objects in different clusters are widely used as separation measures. Also, density-based measures are used in some indices.
The general procedure for determining a best partition of a set of objects using internal validation measures is as follows:

- Step 1: initialize the list of classification algorithms that will be applied to the dataset.
- Step 2: For each clustering algorithm, use different combinations of parameters to obtain different clustering results.
- Step 3: calculate the internal validation index corresponding to each partition obtained in step 2.
- Step 4: Choose the best partition and the optimal number of clusters according to the validity indices used.

One can choose a validity index to estimate the optimal number of clusters, where the optimal classification can be extracted among several classifications under different number of clusters. However, the best clustering solution for a classification task depends not only on a validity index, but on the appropriate clustering procedure as well. An obvious case is the use of different classification methods and different result validity indices in different clustering solutions for a specific classification task. Therefore, there is still a lot of complex work to do in the cluster validation process. The principles of some widely used indices for estimating the optimal number of clusters and evaluating the quality of clustering are introduced in the following.

3.7. Evaluation of a classification system

We present here a method for evaluating a supervised classification, and classical techniques for measuring and comparing unsupervised classification systems.

3.7.1. Test corpus (supervised case)

To test the quality of a supervised classification procedure, the classified elements are randomly separated between a reference base (R) and a test base (T). Then, the classification procedure C^f is determined from the examples of the reference base. Then, we use C^f to find the class of the elements of the test base. Finally, the error of the classification procedure is estimated.
To estimate the error rate TE of a classification procedure Cf, a simple method is to calculate the number of misclassified elements over the number of elements to be classified:

$$TE\left(C^f\right)=\frac{1}{card(T)}\sum_{t=1}^{card(T)}\left(C^f\left(\overrightarrow{d}_t\right)\neq C_{dt}\right) \tag{3.6}$$

Where Cdt is the original class of *dt*.
In the case of simple classifications, we may have to calculate the error resulting from a purely random classification to compare it with the error made by our procedure in order to check the performance of our system. Given the frequency Pk (or a priori probability) of the class k in the test base, we call the error TE of the random system:

$$TE_\alpha=1-\sum_{k=1}^{c}\left(P_k\right)^2=1-\sum_{k=1}^{c}\left(\frac{card(C_k|T)}{card(T)}\right)^2 \tag{3.7}$$

Where c is the number of classes and is the number of elements of T that are in class C_k.

3.7.2. Unsupervised case

In the unsupervised case, the classification can be evaluated with respect to some of these characteristics. We distinguish on the one hand, the numerical characteristics: the number of classes obtained, the number of elements per class, the average number of elements per class, the standard deviation of the classes obtained, and on the other hand, the semantic characteristics . For example, if a document is associated with a set of keywords, the semantics associated with a class may consist of the most frequent words in the class:

$$V = \sigma^2 = \frac{1}{c}\sum_{k=1}^{c}\left(card(C_k) - moy\right)^2$$
$$moy = \frac{1}{c}\sum_{k=1}^{c} card(C_k) \qquad (3.8)$$

3.8. Classification techniques

Many methods and approaches have been defined, and it would be difficult to present an exhaustive list here. However, we can distinguish different commonly used methods. Number of possible classes: The number of possible subsets of a set of n elements is called “Bell number” and is given by the following formula:

Relations between the subsets obtained: In general, we obtain disjoint subsets. This is the result of most techniques. We can also consider the case of disjoint subsets with some subsets including others. This is the case with certain so-called “mixed” techniques. Finally, we can also consider the case of non-disjoint subsets but without inclusion, we then speak of fuzzy analysis. We will not discuss this case.

Distance between individuals: The distance between individuals will be given either by a Euclidean metric, the Manhattan metric which takes absolute values rather than squares, the econometric metric, etc.), or by a matrix of similarities (for example the correlation matrix).

Data preparation: It is in our interest to separate ourselves from non-standard individuals.

Choice of classification variables: To distinguish the right classes, it is often necessary to make several “passes”, by removing or adding variables to the classification method. This concerns variables that are too indiscriminate.

Number of classes: Techniques without a priori let the algorithm determine the optimum number of classes. The techniques with a priori oblige to fix a priori the number of expected classes. One can start by using techniques without a priori and then use the results in the techniques with a priori.

We can always impose fewer classes than there are without a priori. It is useful from a practical point of view if the number of classes found without a priori is too high to be operational in practice (a sales department may want to work on 5 segments, and not on the 10 proposed by the classification without a priori). a priori). On the other hand, imposing a greater number of classes than there is without a priori risks leading to arbitrary results.

Choice of a classification: empirical techniques: to validate a classification, several complementary methods can be used:

Some classic approaches;

3.9. Hierarchical algorithms

This class of algorithms consists of creating a hierarchical decomposition of a data table. Two strategies can be considered: bottom-up or top-down. The ascending hierarchical classification proceeds successively by merging smaller clusters into larger ones, the result of the algorithm is a tree of clusters, called the dendrogram, which shows how the clusters are related. While the top-down approach starts with all objects in a single class. At each iteration, a class is decomposed into smaller classes, until there is only one object left in each class, or possibly a stopping condition is verified.

 Ascending hierarchical classification (CHA)
- Clustering Using Representatives (CURE)
- Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)
- Robust Clustering using links (ROCK)

 Descending hierarchical classification (CHD)
-Williams and Lambert
- Tree Structured Vector Quantization (TSVQ)

3.10. Algorithms per partition

Attempts to decompose the dataset directly into a disjoint set of clusters. More specifically, they try to determine an integer number of partitions that optimize an objective function.
- K-means
- K-medoids,
- Partition Around Medoid (PAM).
- Clustering large applications based upon randomized search (CLARANS)
- Clustering LARge Applications (CLARA)

3.11. Density-based algorithms

The main idea of this type of clustering is to group neighboring objects of a dataset into clusters based on density states.

3.12. Classification based on grid quantification

The idea of these methods is to divide the data space into a finite number of cells forming a grid. This type of algorithm is designed for spatial data. A cell can be a cube, a region, a hyper rectangle. These last two types of methods will not be detailed hereafter.

3.13. Other methods…

In the following, we present the main supervised and unsupervised classification algorithms proposed in the literature. It is not a question of making an exhaustive presentation of all the methods but only of specifying the most traditional methods which we will use within the framework of our work according to their particular properties.

3.14. The K-means method and its variants

3.14.1. Presentation

The K-means algorithm is an algorithm for finding classes in data. It is a “non-hierarchical” algorithm: the classes it constructs never maintain hierarchical relationships: a class is never included in another class. It is a widely used algorithm.

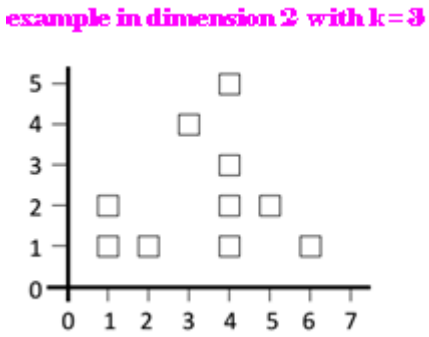
3.14.2. Characteristics of the K-means algorithm (input parameters)

The algorithm works by specifying the number of expected classes. The algorithm calculates the intra-class distance. It therefore works on continuous variables.

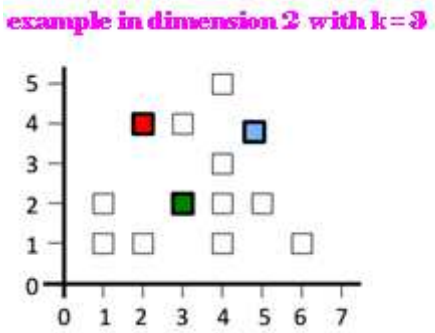
3.14.3. Idea

• Choose k points at random, and consider them as centroids
• Distribute the points in the k classes thus formed according to their proximity to the centroid
• Use the centroids of the classes as new centroids and repeat until there is no more change.
• Distribute the points in the k classes thus formed according to their proximity to the centroid
• Use the centroids of the classes as new centroids and repeat until there is no more change.

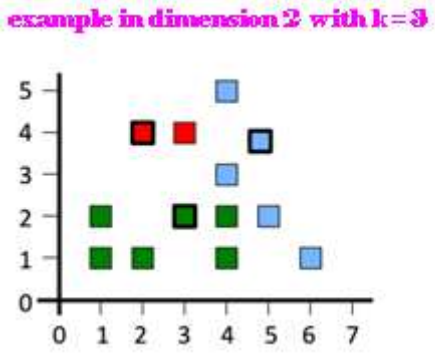
3.14.4. Example



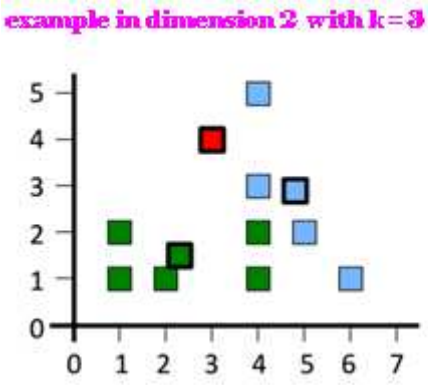
• Choose *k* points at random, and consider them as centroids



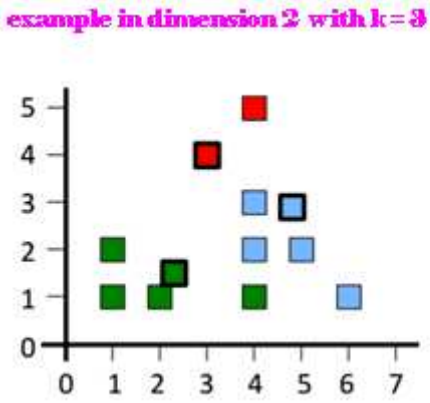
• Distribute the points in the *k* classes thus formed according to their proximity to the centroid



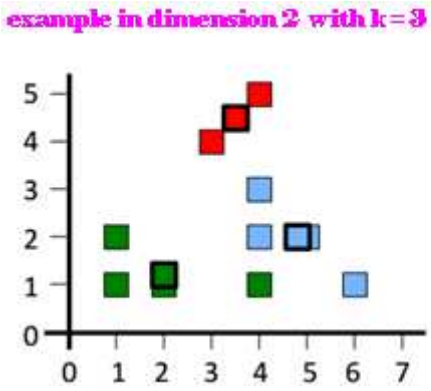
- Use the class centroids as new centroids and repeat until there is no further change.



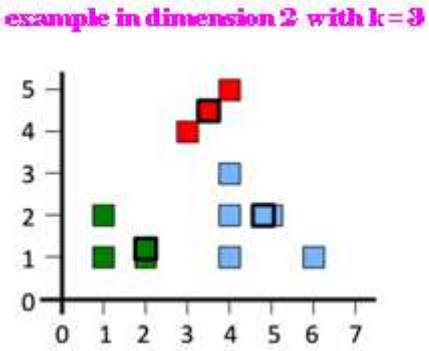
- Distribute the points in the *k* classes thus formed according to their proximity to the centroid.



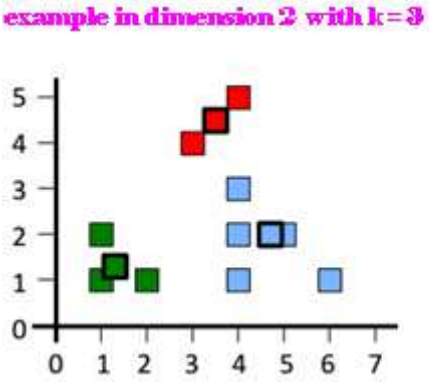
- Use the class centroids as new centroids and repeat until there is no further change.



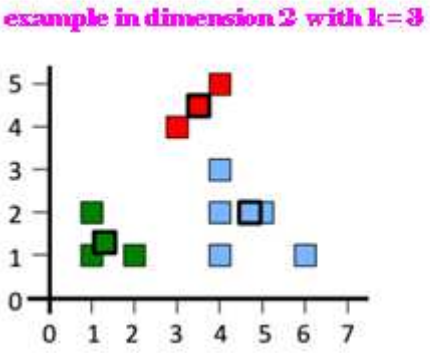
• Distribute the points in the *k* classes thus formed according to their proximity to the centroid.



- Use the class centroids as new centroids and repeat until there is no further change.



- Repeat until there is no further change.



3.14.5. Algorithm

Beginning
K is the number of expected classes;
Initialize the value of the center of these classes with the value of *K* randomly chosen records.
Initialization of intra-class inertia. It could also be initialized to the value of the total inertia.
Repeat
We memorize the "current" intra-class inertia: IAec
For each recording, put it in the nearest class.
Calculate the new center of each class (barycenter).
Calculate the new intra-class inertia: IA
Both IA < IAec
END
We stop when the new intra-class inertia is greater than or equal to the previous one.

3.14.6. Advantages and disadvantages

- Remarks on isolated individuals: out nome, outliers.
- Isolated individuals (outliers, outliers) will always constitute a class.
- The method therefore makes it possible to bring these individuals to light.

You must then eliminate them and restart the method to update the classes. Benefits the calculation time is fast: it is a function of *N* (number of individuals in the starting population).

Disadvantages

- The number of classes is a parameter provided as input.
- The final distribution of classes depends on the first centers chosen.
- The method is especially well suited to spherical classes.

3.15. Hierarchical clustering

In hierarchical classification (CH), the created subsets are hierarchically nested within each other. We distinguish the descending (or divisive) CH which starts from the set of all the individuals and splits them into a certain number of subsets, each subset then being split into a certain number of subsets, and so on. following. And the ascending (or agglomerative) CH which starts from single individuals that are grouped into subsets, which are in turn grouped together, and so on. To determine which classes we will merge, we use the aggregation criterion.

3.15.1. Intuitions and principles

- From a dissimilarity matrix, allows to form groups \step by step:
  - by dividing two groups (descending classification)
  - by agglomeration of groups (ascending classification)
- Create hierarchies between groups (even if the nesting does not make sense from the point of view of interpretation)
- Each level of the hierarchy represents a particular partition of data into disjoint groups
- The hierarchy can be represented as a tree or dendrogram

Iterative data aggregation

- Start with *N* groups, one per observation
- Agglomerate the two most similar groups and recalculate the center
- Repeat until a single group is obtained
- Iterative division of data
- Start with one group
- Divide into two groups as different as possible
- Repeat until *N* groups are obtained
- Similarity measures for agglomerative clustering
- Single link clustering d(Gi; Gj) = min xr□Gi;xs□Gj D(xr; xs)
- Full link clustering d(Gi; Gj) = max xr□Gi;xs□Gj D(xr; xs)

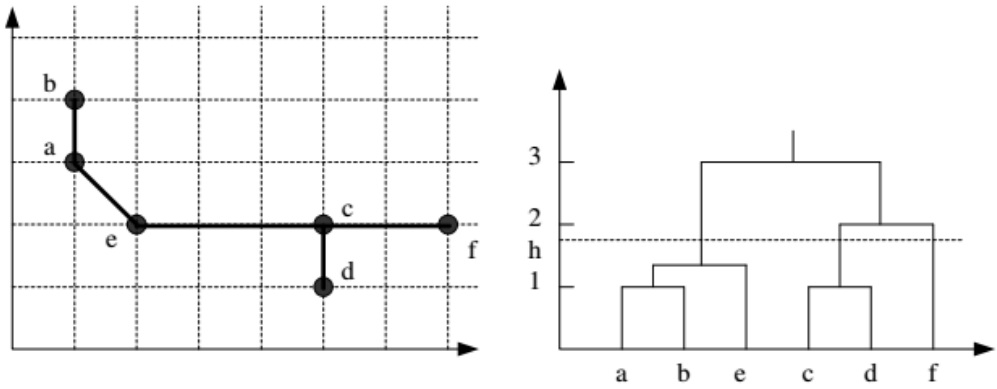


Figure 3.3. Data example and Dendrogram

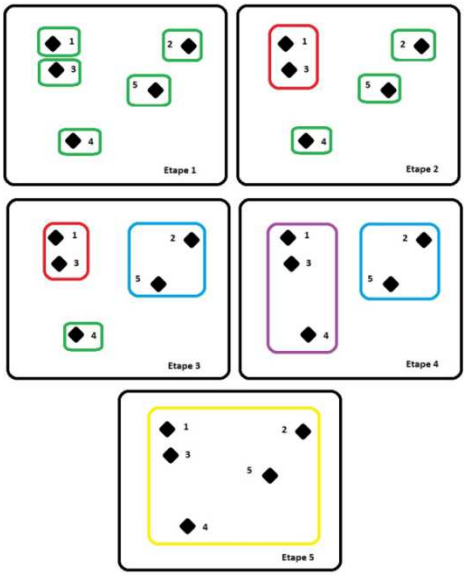
3.15.2. CAH algorithm: The CAH algorithm is described below:

- A deviation is chosen. We construct the table of deviations for the initial partition of the *n* individuals of Γ:

Each individual constitutes an element.

- We go through the table of deviations to identify the pair of individuals with the smallest deviation. The grouping of these two individuals forms a group A. We therefore have a partition of Γ of (n – 1) elements: A and the remaining (n – 2) individuals.
- We calculate the table of deviations between the (n – 1) elements obtained in the previous step and we combine the two elements with the smallest deviation (this can be two of the (n - 2) individuals, or one individual of the (n – 2) remaining individuals with A). We therefore have a partition of Γ of (n – 2) elements.
- Iterate the previous procedure until only two elements remain.
- Combine the two remaining elements. There remains then only one element containing all the individuals of Γ.

3.15.3. Graphical example: Below is a graphical example of the steps of the CAH algorithm:



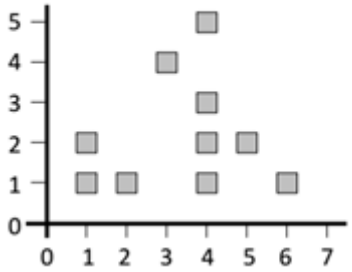
3.15.4. Dendrogram

Dendrogram: The partitions of Γ made at each step of the CAH algorithm can be visualized via a tree called a dendrogram. On one axis appears the individuals to be grouped and on the other axis are indicated the differences corresponding to the different levels of grouping. This is done graphically through branches and nodes. A natural partition is made by cutting the tree at the level of the largest jump of nodes.

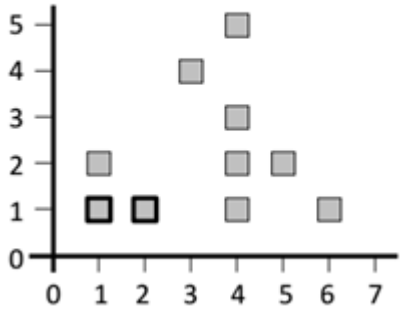
3.15.5. Idea

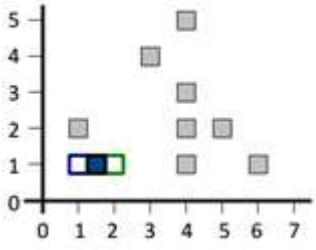
- Take the two closest points, merge them, and then consider their average.
- Repeat until a stopping criterion (for example distance greater than a certain value).
- Or cut the fusion tree to obtain classes.

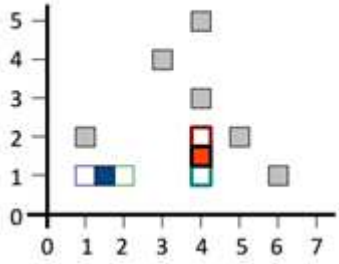
3.15.6. Example in dimension 2

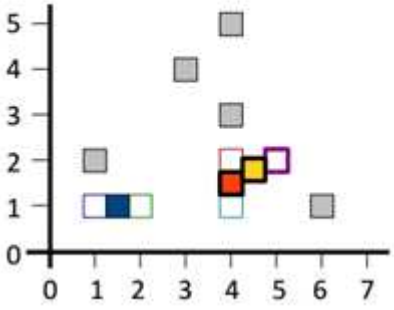


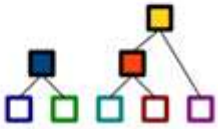
• Take the two closest points, merge them, and consider their average thereafter.


5
4
3
2
1
0
0 1 2 3 4 5 6 7

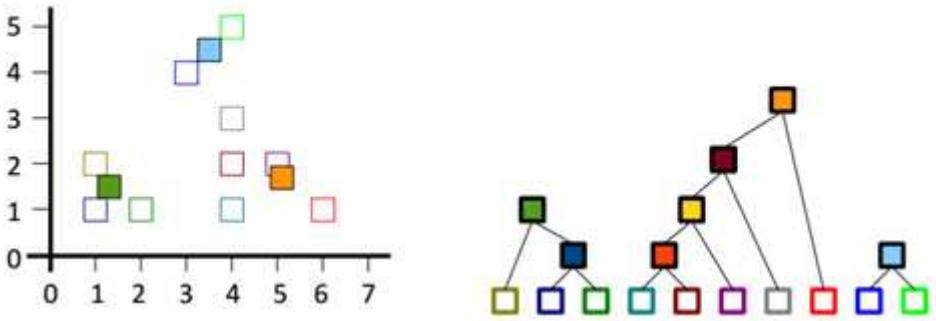

5
4
3
2
1
0
0 1 2 3 4 5 6 7


5
4
3
2
1
0
0 1 2 3 4 5 6 7


5
4
3
2
1
0
0 1 2 3 4 5 6 7



…



3.15.7. Advantages and disadvantages

Advantages: Reading the tree makes it possible to determine the optimal number of classes.

Disadvantages: Computationally expensive.

3.16. *k* nearest neighbors

3.16.1. Presentation

The k-nearest neighbor method (*Knn k*-nearest neighbor) (Indyk and Motwani [1998]) is based on a direct comparison between the feature vector representing the entity to be classified and the feature vectors representing reference entities. The comparison consists of a calculation of distances between these entities. The entity to be classified is assigned to the majority class among the classes of the k closest entities in the sense of the distance used. Let us denote by $Xp = (x_{p1}; x_{p2};\ldots; x_{pN})$ the characteristic vector of the entity p, with N the number of characteristics and by p and q two entities to be compared.
The following distances are usually employed by *Knn* classifiers:

- Euclidean Distance
- Distance from Manhattan
- Minkowski distance
- Chebyshev distance

In figure 3.4, on the left, the classification is simple whatever the number of neighbors: the new object is black.
On the right, on the other hand, everything depends on the number of neighbors chosen and the classification heuristic. For k = 1, the new object is gray. For k = 3, if the three neighbors have the same weight, then the new object is black. On the contrary, if the weight is weighted by the inverse of the distance so the new object can be gray. That amounts to weighting the class assignment with the distance: the further a neighbor is, the more its influence is weak.

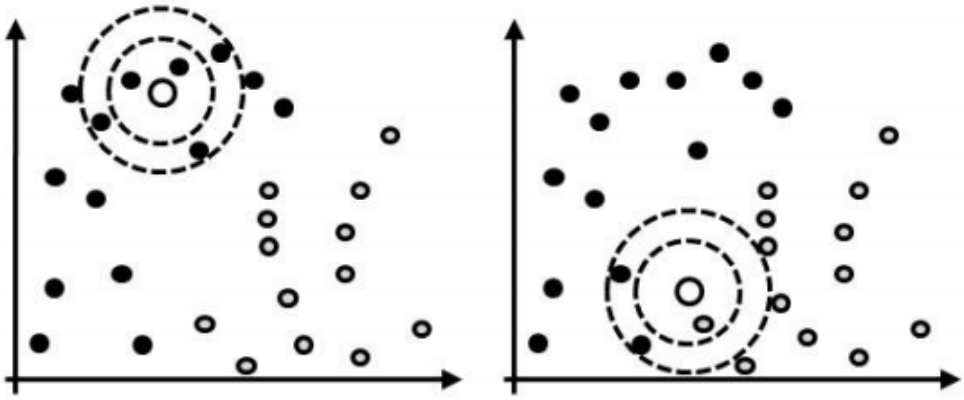
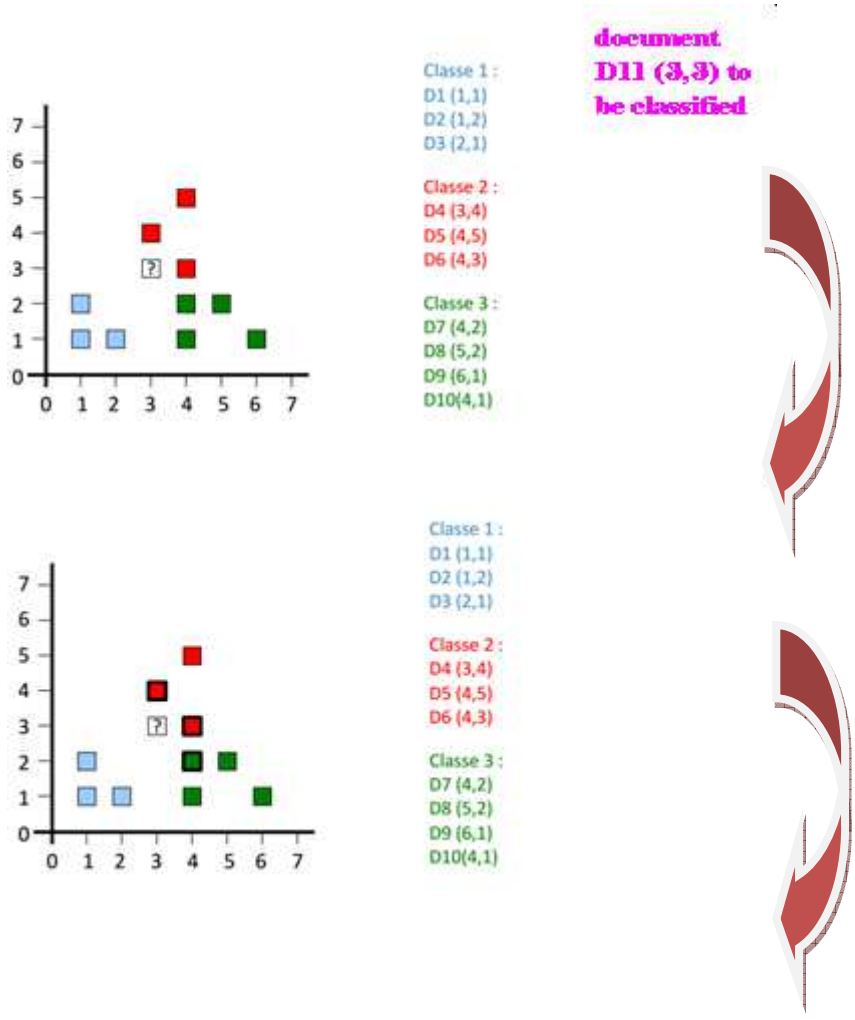


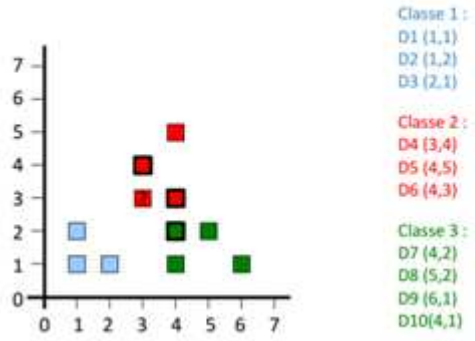
Figure 3.4 Exemple de classification avec les *Knn*.

3.16.2. Idea

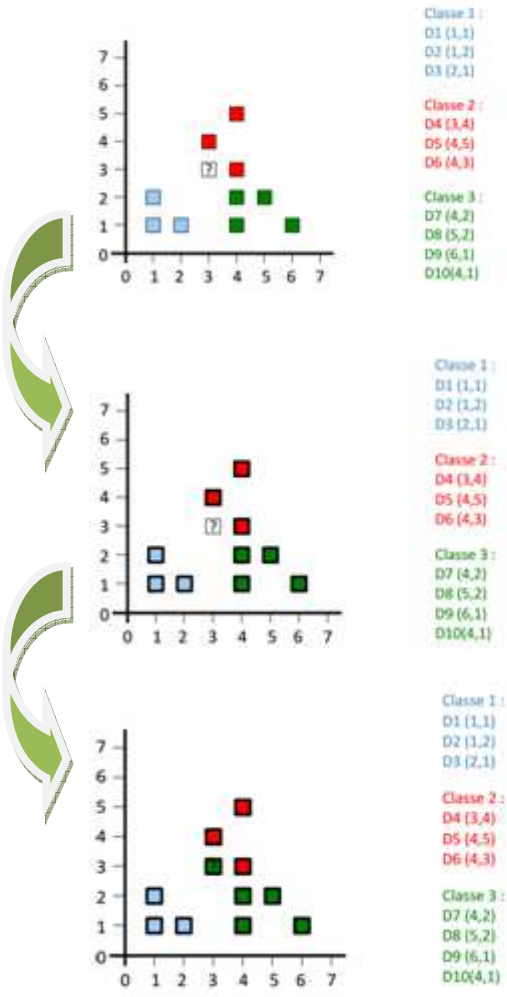
Choose for each vertex the majority class among its *k* nearest neighbors.

3.16.3. Example in dimension 2, *k*=3:





3.16.4. Example in dimension 2, *k*=10 :



Case of equality?

- Increase k by 1? Will work if 2 class classification, risk to fail otherwise.
- Random draw.
- Weighting of neighbors in relation to their distance from the point to be classified.

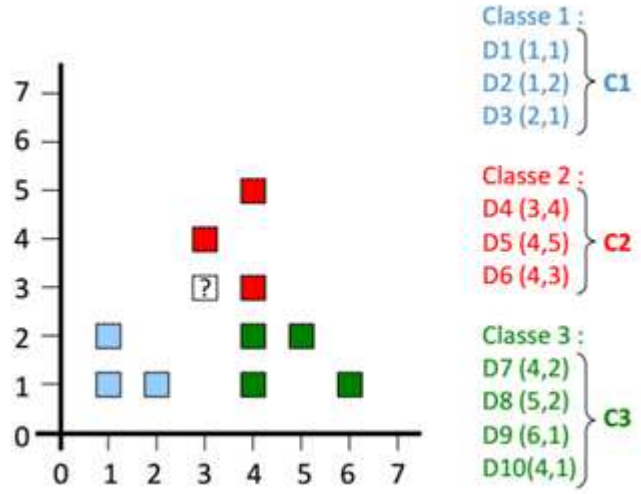
3.16.5. Advantages and disadvantages

The main drawbacks of this method are the number of operations required to classify an entity in the case of a large reference base as well as its sensitivity to the noise present in the training data.

3.17. Centroid approach

3.17.1. Idea: Represent each class by its center and classify the new element according to its distance from the centers.

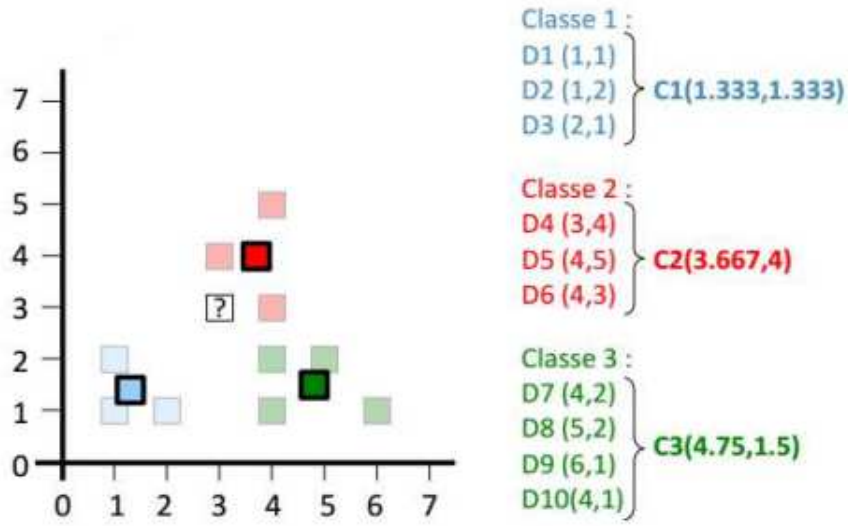
3.17.2. Example

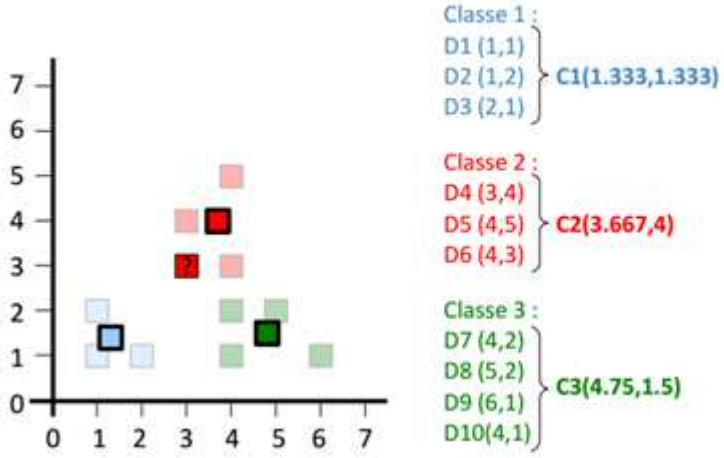


*D*11(3,3) document to be classified

i-ime coordinates of the centroid of class *K*

$$Ck_i = \frac{\sum_{DJ \in classe\,k} Dj_i}{|Class\ k|}$$





*D*11(3,3) → document to be classified

Squares of Euclidean distances to centroids Class 2!

3.18. References

[21] Jure Leskovec and Anand Rajaraman, « *Clustering Algorithms*», Stanford University.
[22] Marc el-bèze, «*Classification locale non supervisée pour la recherche documentaire*», vol . 41, n° 2, pp, 2000.
[23] Catherine Aaron, «*Algorithme em et classification non supervisée*», université Paris.
[24] Christian Gagne, «*Clustering Apprentissage et reconnaissance*», université Laval, 2010.
[25] Marine Campedel, *Classification supervisée*», école national superieur de télécommunication, 2005.
[26] Hassan Chouaib, «*Sélection de caractéristiques: méthodes et applications*», 2011.
[27] Christophe Chesneau, «*Eléments de classification*», Université de Caen, 2016.
[28] Franck Picard, «*Classification non supervisée*», Université de Lyon.
[29] Philippe Gambette, «*Classification supervisée et non supervisée*», Université Marne-la-Vallée (IGM), 2014.

Chapter 4: Fuzzy logic and applications in electrical engineering

4.1. Introduction

Fuzzy logic today: Fuzzy logic, in most applications present, makes it possible to take into account all kinds of qualitative knowledge of designers and operators in system automation. It has aroused media interest in France since the early 1990s. The many applications in household appliances and consumer electronics, particularly in Japan, were the trigger. Tuneless washing machines, anti-shake camcorders, and many other innovations brought the term “fuzzy logic” to a wide audience. In the automobile, automatic transmissions, injection and anti-knock controls, air conditioning are carried out on production vehicles using fuzzy logic. In the field of production processes, both continuous and batch, and in automation (which mainly concerns us) applications have also multiplied.

Fuzzy logic develops there because it is an essentially pragmatic, efficient and generic approach. It is sometimes said that it makes it possible to systematize what is in the domain of empiricism, and therefore difficult to master.

The fuzzy set theory provides a relevant and easily feasible method in real-time applications; it makes it possible to transcribe and make dynamic the knowledge of designers or operators. This adaptable and universal aspect of fuzzy logic makes it possible to tackle the automation of procedures such as start-up, parameter adjustment, for which few approaches existed before. This Chapter presents fuzzy logic and its application in the context of control.

4.2. History of fuzzy logic

Appearance of fuzzy logic: The term fuzzy set appears for the first time in 1965 when Professor Lotfi A. Zadeh, from the University of Berkeley in the USA, published an article entitled “Fuzzy sets”. He has since made many major theoretical advances in the field and was quickly accompanied by many researchers developing theoretical work.

First applications: At the same time, some researchers have focused on the resolution by fuzzy logic of problems deemed difficult. Thus in 1975, Professor Mamdani in London developed a strategy for process control and presented the very encouraging results he had obtained on the operation of a steam engine. In 1978, the Danish company F.L.Smidth carried out the control of a cement kiln. This is the first real industrial application of fuzzy logic.

Growth: It is in Japan, where research is not only theoretical but also very applicative, that fuzzy logic really took off. At the end of the 1980s, it is a real boom that we must speak of. Consumer products, washing machines, cameras and other camcorders stamped "fuzzy logic" do not matter more. In industry, water treatment, port cranes, metros, ventilation and air conditioning systems are affected. Finally, applications exist in very different fields such as finance or medical diagnosis. From 1990, it was in Germany that applications appeared in large numbers as well only to a lesser extent in the USA. Finally in France, fuzzy logic becomes a reality today.

4.3. Why Fuzzy Logic: Limits of Classical Logic

A patient with hepatitis usually presents with the following symptoms:

- The patient has a **high fever**
- His skin has a yellow color
- He is nauseous

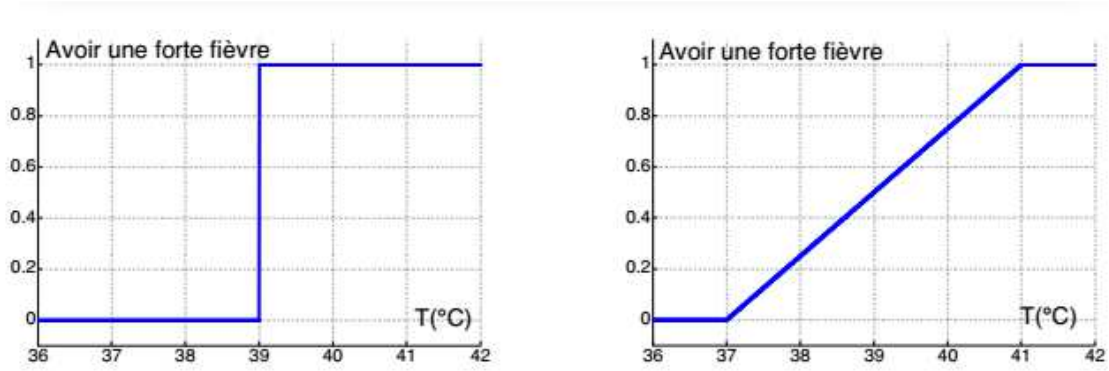


Figure. 4.1 Temperature in classical logic and fuzzy logic

The patient has 38.9°C

- In classical logic: the patient has no high fever => no hepatitis
- In fuzzy logic: the patient has a high fever at 50% => hepatitis at x%

4.4. Fuzzy set theory

4.4.1. Notion of partial membership

In set theory, an element belongs or does not belong to a set. The notion of set is at the origin of many mathematical theories. However, this essential notion does not make it possible to account for situations that are simple and frequently encountered. Among fruits, it is easy to define the set of apples. On the other hand, it will be more difficult to define all the ripe apples. It is clear that the apple ripens gradually... the notion of a ripe apple is therefore gradual. It is to take into account such situations that the notion of fuzzy set was created. Fuzzy set theory is based on the notion of partial membership: each element belongs partially or gradually to the fuzzy sets that have been defined. The contours of each fuzzy set (see fig. 2) are not "sharp", but "fuzzy" or "gradual".

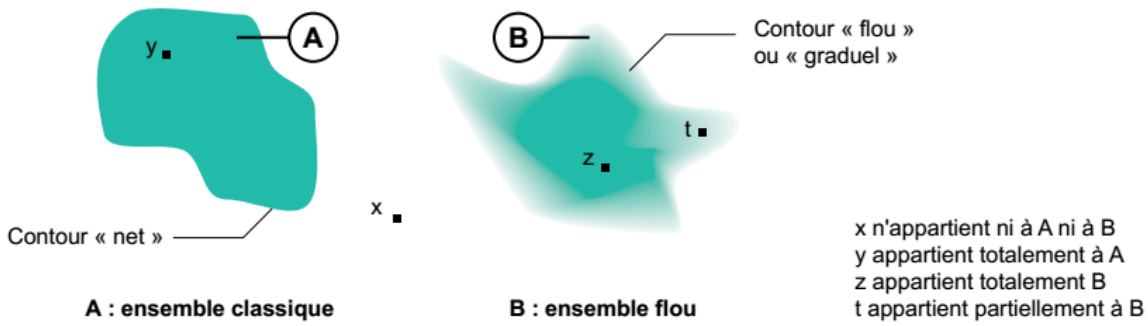


Figure. 4.2: Comparison between a classic set and a fuzzy set [35].

4.4.2. Membership functions

A fuzzy set is defined by its “membership function”, which corresponds to the notion of “characteristic function” in classical logic.

Suppose we want to define the set of "average height" people. In classical logic, we will agree for example that people of average height are those whose height is between 1.60 m and 1.80 m. The characteristic function of the set (see fig. 4.3) gives "0" for sizes outside the interval [1.60 m; 1.80 m] and “1” in this interval.

The fuzzy set of people of “average height” will be defined by a “membership function” which differs from a characteristic function in that it can take any value in the interval [0, 1]. Each possible size will correspond to a “degree of membership” of the fuzzy set of “average sizes” (see fig. 4.4), between 0 and 1.

Several fuzzy sets can be defined on the same variable, for example the “small size”, “medium size” and “large size” sets, concepts each explained by a membership function (see fig. 4.5).

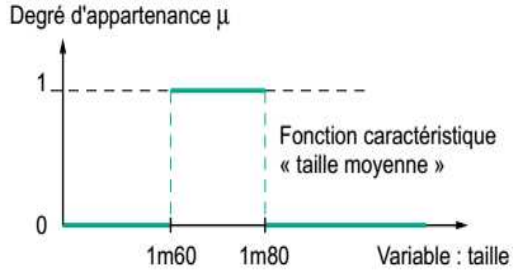


Figure. 4.3: Characteristic function [35].

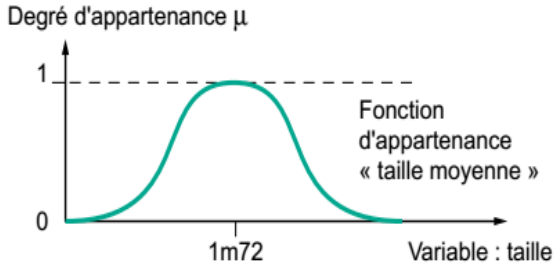


Figure. 4.4: Membership function [35].

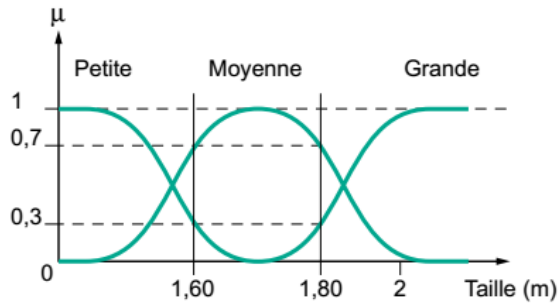


Figure. 4.5 : Membership function, variable and linguistic term [35].

This example shows the graduality that fuzzy logic can introduce. A person 1.80 m tall belongs to the “large size” set with a degree of 0.3 and to the “medium size” set with a degree of 0.7. In classical logic, the transition from medium to large would be abrupt. A person of 1.80 m would for example be of average height while a person of 1.81 m would be tall, which shocks intuition. The variable (for example: size) as well as the terms (for example: medium, large) defined by the membership functions bear the names of linguistic variable and linguistic terms respectively. As will be seen later, variables and linguistic terms can be used

directly in rules. Membership functions can theoretically take any form. However, they are often defined by straight line segments, and called “piecewise linear” (see Figure. 4.6).

The “piecewise linear” membership functions are widely used because:

- They are simple,
- They include points making it possible to define the zones where the concept is true, the zones where it is false, which simplifies the collection of expertise.

In some cases, the membership functions can be equal to 1 for a single value of the variable and equal to 0 elsewhere, and then take the name of “singleton membership functions”. A fuzzy singleton (see figure. 4.7) defined on a real variable (size) is the translation in the fuzzy domain of a particular value (Paul size) of this variable.

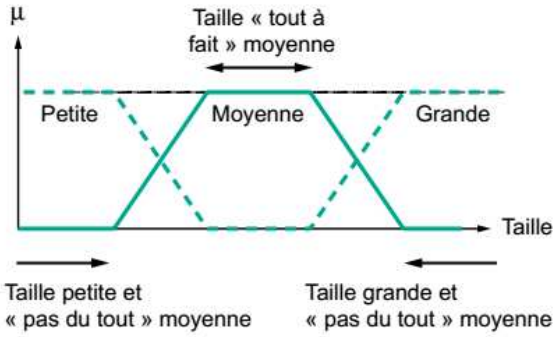


Figure. 4.6: Piecewise linear membership functions [35].

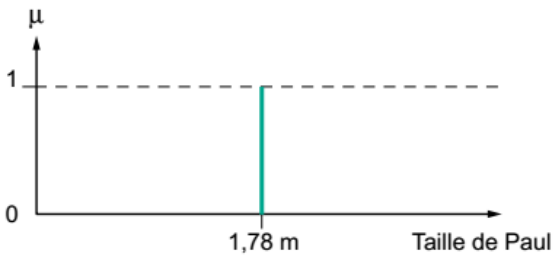


Figure. 4.7: Singleton membership function [35].

Fuzzification - Degree of membership: The fuzzification operation makes it possible to pass from the real domain to the fuzzy domain. It consists in determining the degree to which a value (measured for example) belongs to a fuzzy set. For example (see figure. 4.8), if the current value of the "input" variable is 2, the degree of membership of the "weak input" membership function is equal to 0.4 which is the result of fuzzification.
We can also say that the “weak entry” proposition is true at 0.4. We then speak of the degree of truth of the proposition. Degree of belonging and degree of truth are therefore similar notions.

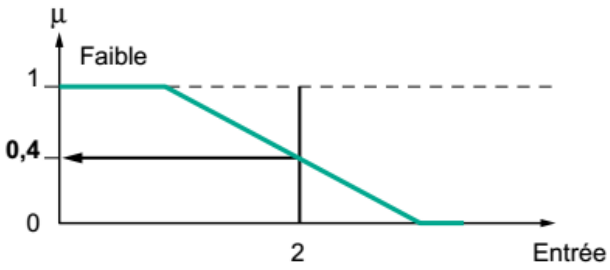


Figure. 4.8 : Fuzzification [35].

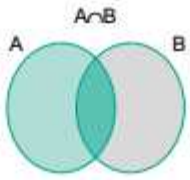
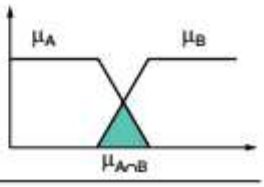
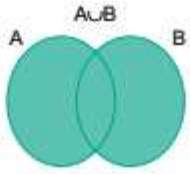
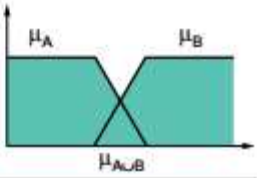
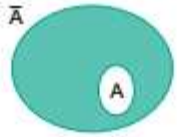
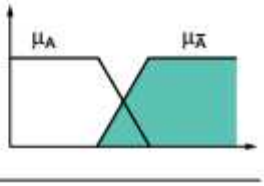
4.4.3. Fuzzy logic operators

These operators make it possible to write logical combinations between fuzzy notions that are to make calculations on degrees of truth. As with classical logic, we can define AND, OR, negation operators. Example: Interesting Apartment = Reasonable Rent AND Sufficient Area.

Choice of Operators: There are many variations in these operators. However, the most common are the so-called "Zadeh" ones described below. Their use will be covered in the didactic example of using a fuzzy rule base. In what follows, the degree of truth of a proposition A will be denoted $\mu(A)$.

Operators between fuzzy sets

The table presents the different ZADEH operators [35].

		ZADEH operator	Logic operator	
Intersection		$\mu_{A\cap B} = MIN(\mu_A, \mu_B)$	AND	
Union		$\mu_{A\cup B} = MAX(\mu_A, \mu_B)$	OR	
Negation		$\mu_{\bar{A}} = 1 - \mu_A$	NO	

Intersection: The logical operator corresponding to the intersection of sets is AND. The degree of truth of the proposition "A AND B" is the minimum of the degrees of truth of *A* and *B*: $\mu(A \text{ AND } B) = \text{MIN}(\mu(A),\mu(B))$

Example: “Low Temperature” is true at 0.7 “Low Pressure” is true at 0.5
“Low Temperature AND Low Pressure” is therefore true at 0.5 = MIN(0.7; 0.5) Note: the AND operator of classical logic is well respected: 0 AND 1 does give 0.

Union: The logical operator corresponding to the union of sets is the OR. The degree of truth of the proposition "A OR B" is the maximum of the degrees of truth of A and B:
$\mu(A \text{ OR } B) = \text{MAX}(\mu(A),\mu(B))$

Example: “Low Temperature” is true at 0.7 “Low Pressure” is true at 0.5 “Low Temperature OR Low Pressure” is therefore true at 0.7.
Note: the OR operator of classical logic is well respected: 0 OR 1 gives 1.

Complement: The logical operator corresponding to the complement of a set is the negation.
$\mu(\text{NOT } A) = 1 - \mu(A)$

Example: "Low Temperature" is true at 0.7 "NO Low Temperature", which will generally be used in the form "NOT Low Temperature", is therefore true at 0.3.
Note: the negation operator of classical logic is well respected: NOT(0) gives 1 and NOT(1) gives 0.

Fuzzy ladder: The ladder language, or contact language, is widely used by automation engineers to write logical combinations. It makes it possible to represent them graphically. Schneider introduced the use of ladder representation to describe fuzzy logic combinations. Here is an example, dealing with air comfort ambient: hot, humid air is uncomfortable (excessive sweating); likewise breathing becomes difficult in cold and too dry air. The most thermally comfortable situations are those in which the air is hot and dry, or cold and humid. This physiological finding can be transcribed by the fuzzy ladder in Figure 4.9 corresponding to the following combination: Good comfort = (Low temperature AND High humidity) OR (High temperature AND Low humidity) It represents a possible definition of the feeling of comfort felt by a person in a thermal environment for which the air is still.

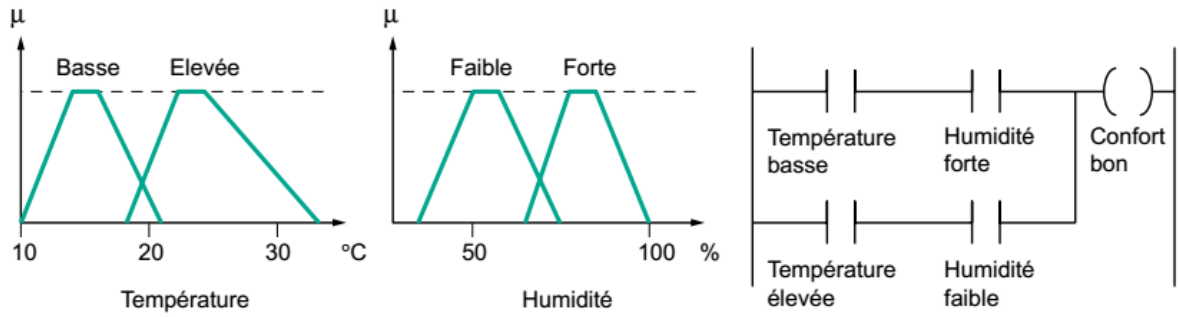


Figure.4.9 : Fuzzy ladder [35].

Fuzzy classification: Classification generally includes two steps:

- Preparatory: determination of the classes to be considered,
- Online: assignment of elements to classes.

The notion of class and set are identical on the theoretical level. There are three types of assignment methods, depending on the result produced:

- Boolean: the elements belong or not to the classes,
- Probabilistic: the elements have a probability of belonging to Boolean classes, such as The probability that a patient has measles given the symptoms he presents (diagnosis),
- Gradual: the elements have a degree of belonging to the sets; for example, a salad belongs more or less to the class of “fresh salads”.

  Classification methods, whether they produce a gradual, Boolean or probabilistic, can be developed from:
- An experience (case of the "fuzzy ladder" mentioned above),
- Examples used for learning (for example in the case of neural network classifiers)
- A mathematical or physical knowledge of the problem (for example the comfort of a thermal situation can be evaluated from heat balance equations).

Gradual (or fuzzy) classification methods make it possible, in particular, to develop control loops. This is the case in the example of the industrial baking of biscuits set out below.

4.4.4. Fuzzy rules

Fuzzy logic and artificial intelligence: Fuzzy logic aims to formalize and implement the way of reasoning of a human being. In this, it can be classified in the field of artificial intelligence. The most used tool in fuzzy logic applications is the fuzzy rule base. A fuzzy rule base is composed of rules that are usually used in parallel, but can also chained in some applications. A rule is of the type: IF “predicate” THEN “conclusion”. For example: “If high temperature and high pressure THEN strong ventilation and valve wide open”. Fuzzy rule bases, like traditional expert systems, work by relying on a knowledge base derived from human expertise. There are, however, great differences in the characteristics and treatment of this knowledge. A fuzzy rule has three functional steps summarized in Figure 4.10.

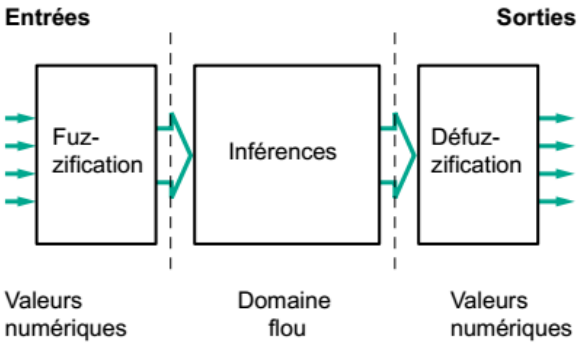


Figure. 4.10 : Fuzzy processing [35].

Predicate: A predicate (also called premise or condition) is a combination of propositions by operators AND, OR, NOT. The propositions “high temperature” and “high pressure” from the previous example are combined by the AND operator to form the predicate of the rule.

Base de règles floues	Base de règles classiques (système expert)
Peu de règles	Beaucoup de règles
Traitement graduel	Traitement booléen
Enchaînement possible mais peu utilisé	Règles enchaînées A OU B ⇒ C, C ⇒ D, D ET A ⇒ E
Règles traitées en parallèle	Règles utilisées une par une, séquentiellement
Interpolation entre règles pouvant se contredire	Pas d'interpolation, pas de contradiction

Figure. 4.11 : Fuzzy rule base and classical rule base [35].

Inference: The most commonly used inference mechanism is the so-called “Mamdani” mechanism. It represents a simplification of the more general mechanism based on “fuzzy implication” and “generalized modus ponens”. These concepts are explained in the appendix. Only “Mamdani” rule bases are used in what follows.

Conclusion: The conclusion of a fuzzy rule is a combination of propositions linked by AND operators. In the previous example, “strong ventilation” and “wide open valve” are the conclusion of the rule. We do not use "OR" clauses in conclusions, because they would introduce uncertainty into the knowledge (the expertise would not make it possible to determine which decision to make). This uncertainty is not taken into account by Mamdani's inference mechanism, which only manages inaccuracies. The fuzzy “Mamdani” rules are therefore a priori not suitable for “medical diagnosis” type diagnosis for which the conclusions are uncertain. The theory of possibilities, invented by Lotfi Zadeh, provides an adequate methodology in these cases. Similarly, negation is a priori prohibited in conclusions for Mamdani rules. Indeed, if a rule had for example the conclusion “Then not average ventilation”, it would be impossible to say if this means “weak ventilation” or “strong ventilation”. This would still be a case of uncertainty.

Mamdani inference mechanism

- Principle: A fuzzy Mamdani rule base therefore includes linguistic rules using membership functions to describe the concepts used (see figure.4.12)

The inference mechanism includes the following steps:

- Fuzzification: Fuzzification consists in evaluating the membership functions used in the predicates of the rules, as illustrated by figure 4.13:

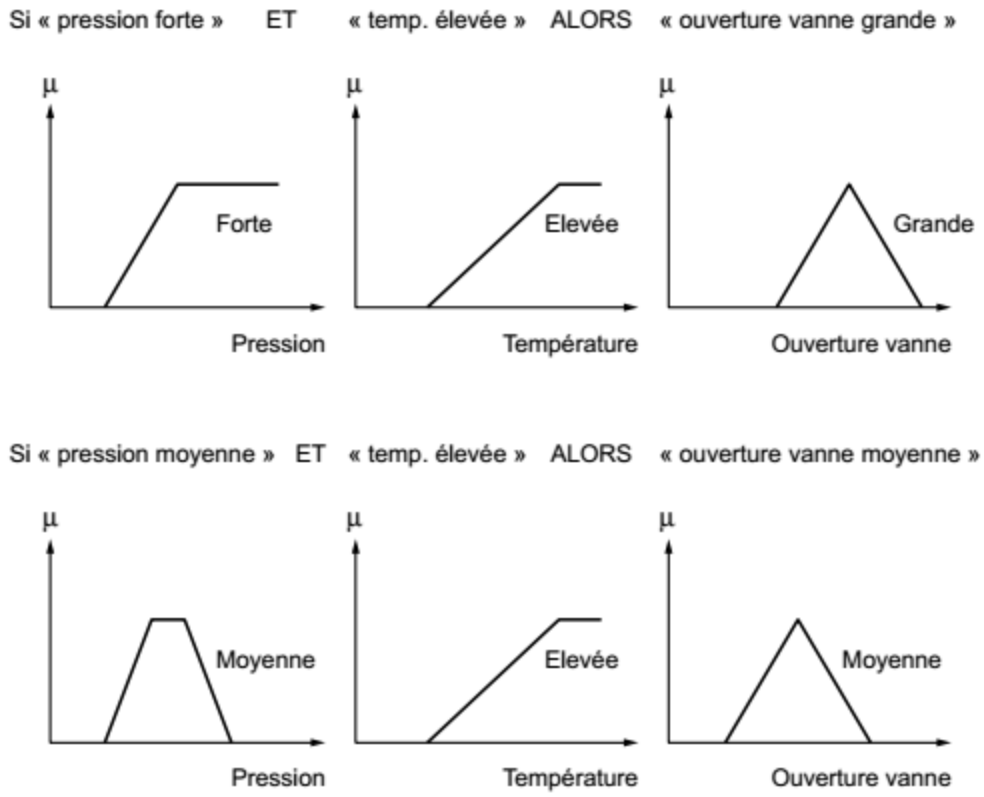


Fig. 4.12 : Implication [35].

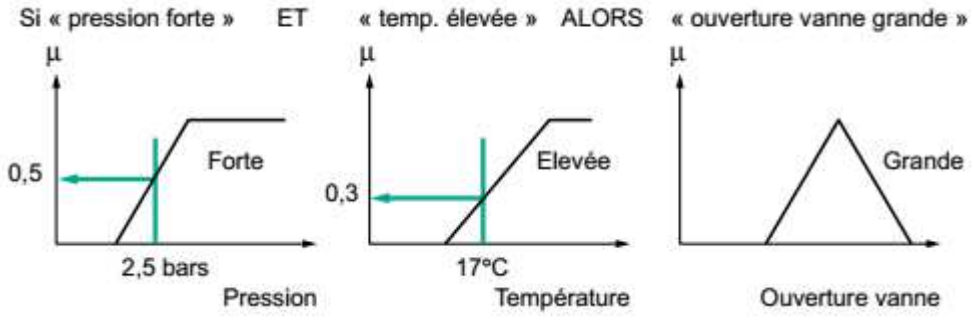


Fig. 4.13 : Fuzzification [35].

Degree of activation: The degree of activation of a rule is the evaluation of the predicate of each rule by logical combination of the propositions of the predicate, as illustrated in figure 4.14. The "AND" is achieved by carrying out the minimum between the degrees of truth of the propositions.

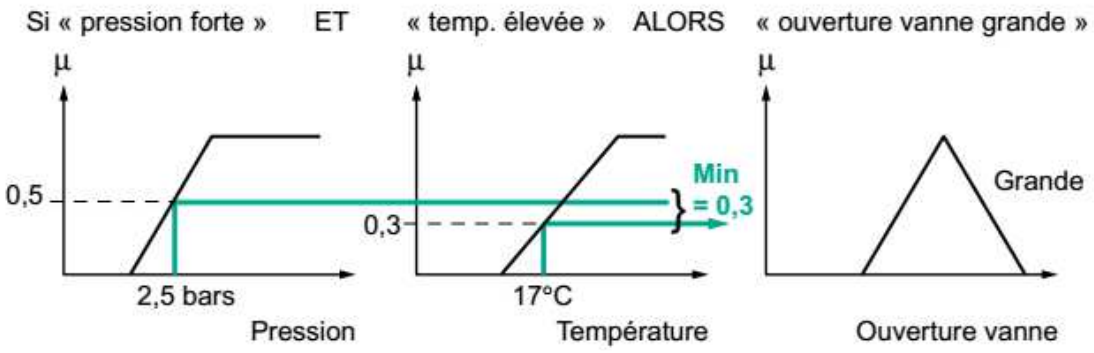


Fig. 4.14 : Activation [35].

Implication: The degree of activation of the rule determines the conclusion of the rule, it is the implication. There are several implication operators (see appendix), but the most used is the “minimum”. The conclusion fuzzy set is constructed by achieving the minimum between the degree of activation and the membership function, a kind of “clipping” of the conclusion membership function (see figure. 4.15).

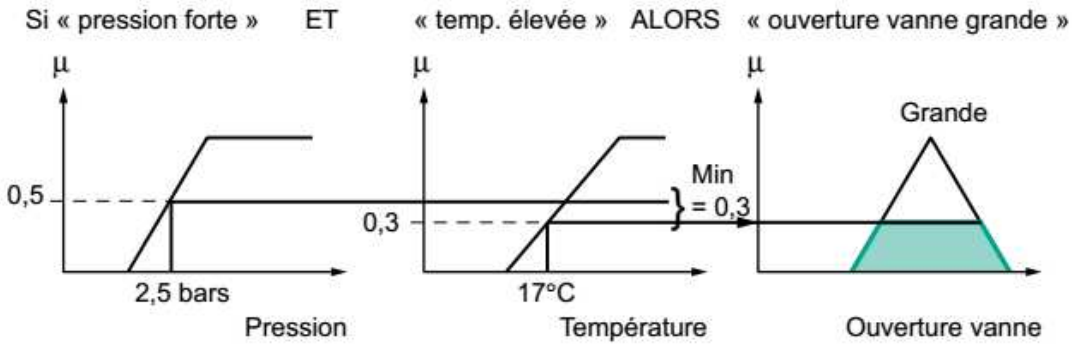


Fig. 4.15 : Implication [35].

Aggregation: The output global fuzzy set is constructed by aggregation of the fuzzy sets obtained by each of the rules concerning this output. The following example presents the case where two rules act on an output. We consider that the rules are linked by a logical “OR”, and we therefore calculate the maximum between the resulting membership functions for each rule (see figure. 4.16).

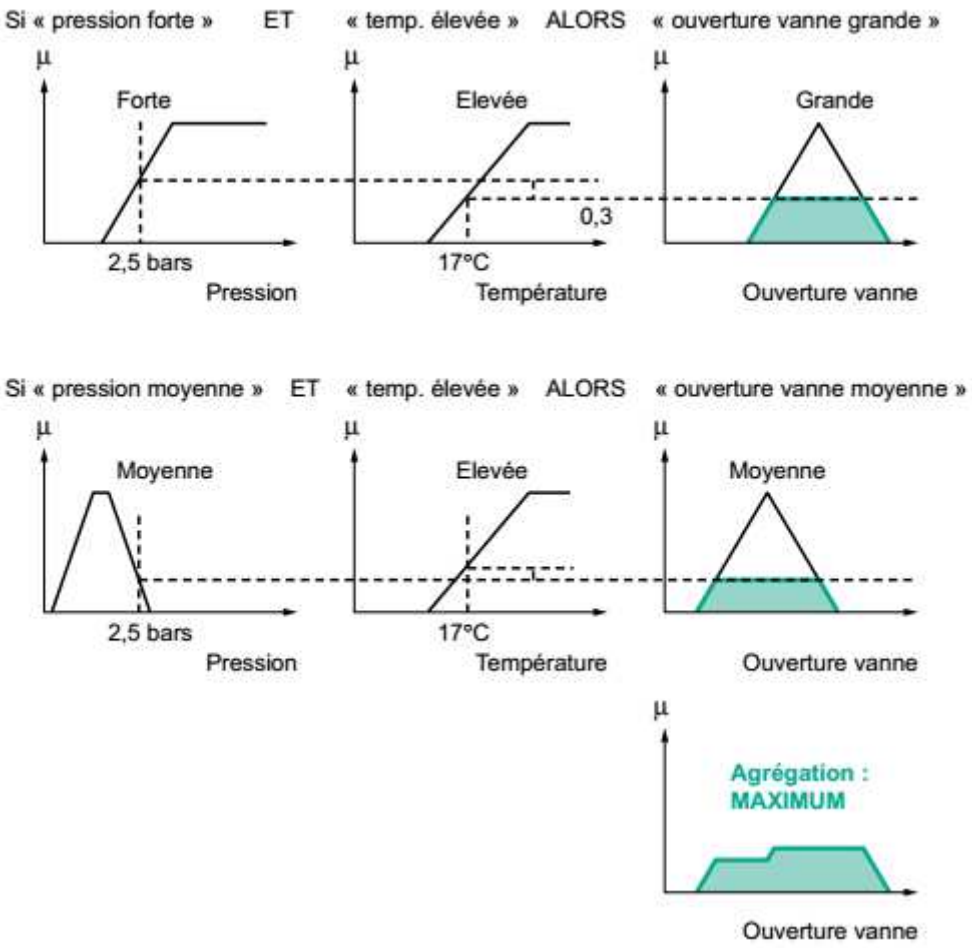


Fig. 4.16 : Rule aggregation [35].

Defuzzification: At the end of the inference, the output fuzzy set is determined but it is not directly usable to give precise information to the operator or to control an actuator. It is necessary to move from the "fuzzy world" to the "real world", this is defuzzification. There are several methods; the most often encountered being the calculation of the “center of gravity” of the fuzzy set (see fig. 4.17).

“Free” and “table” rules: Fuzzy rule bases, in their general case, are therefore defined by membership functions on system variables, and by rules that can be written textually. Each rule uses inputs and outputs that may be different, as shown in the following example:
R1: IF "high temperature" THEN "high output"
R2: IF "medium temperature" AND "low pressure" THEN " average output”
R3: IF “average temperature” AND “pressure high” THEN “low output”
R4: IF “low temperature” AND “high pressure” THEN “very low output”

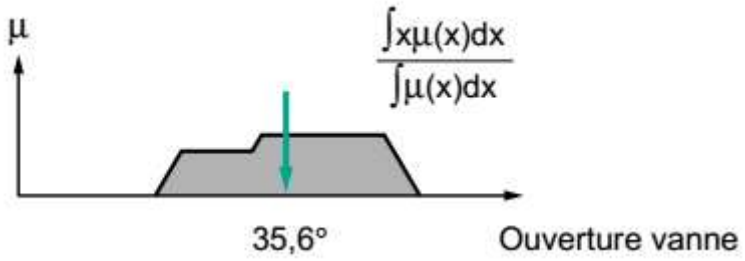


Figure. 4.17 : Defuzzification by center of gravity.

Exemple :

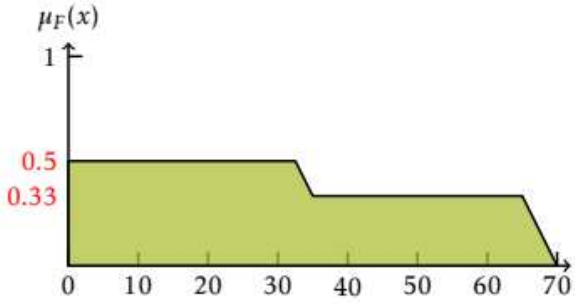


Figure. 4.18 – Aggregation of output rules by cut.

The center of gravity does not need to be calculated very precisely. We can approximate, by calculating every 10, for example. For the following example:

$$CG = \frac{(0+10+20+30)(0.5)+(40+50+60)(0.33)}{0.5+0.5+0.5+0.5+0.33+0.33+0.33} = 26.67 \tag{4.1}$$

The fan must therefore be at 26.67% of its maximum speed.

Schematically, we can represent the “action zones” of the rules and their overlap in the table of figure 4.19.

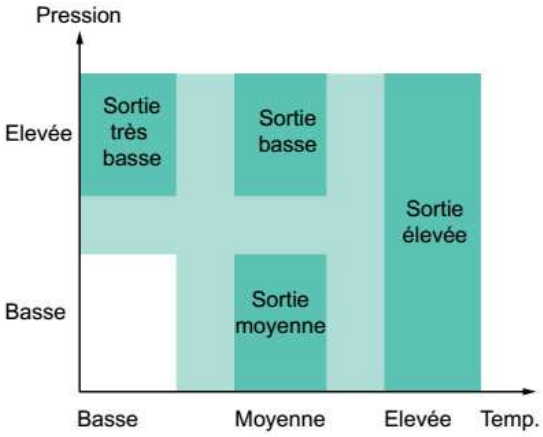


Fig. 4.19 : implication représentée en tableau [35].

We observe that:

- Not all the space is necessarily covered; the “low temperature and low pressure” combination is not taken into account here; the explanation is for example that this combination is not physically possible for this machine, or that it does not interest us; it is better to check it because it may be an oversight;
- The first rule only takes into account the temperature; this situation is completely normal insofar as it correctly reflects the existing expertise. However, many applications define "arrays" of rules. With this in mind, the space is “squared”, and each “box” corresponds to a rule. This approach has the advantage of being systematic, but:
- It does not always make it possible to simply translate (in a minimum of rules) the existing expertise,
- It is only applicable for two or even three inputs, whereas "free" rule bases can be built with a large number of variables.

Remarks

- A fuzzy rule base has a nonlinear static behavior with respect to its inputs.
- Fuzzy rule bases are not dynamic in themselves, although they are often use variables reflecting the dynamics of the system (derivatives, integrals, etc.) or time as inputs.
- The "fuzzy PID" controller, often presented as a didactic example to get an idea of fuzzy logic, has the main interest of producing a non-linear PID, which rarely justifies using it instead of a conventional PID. Moreover, it is difficult here to integrate expertise.

4.5. Fuzzy Control

Fuzzy Logic has been successfully applied to a large number of control applications. The most commonly used controller is the PID controller, which requires a mathematical model of the system. A fuzzy logic controller provides an alternative to the PID controller. The control action in fuzzy logic controllers can be expressed with simple “if-then” rules. Fuzzy controllers are more sufficient than classical controllers because they can cover a much wider range of operating conditions than classical controllers and can operate with noise and disturbances of a different nature.

4.5.1. Example 1: Fuzzy command of an automatic watering system

- inputs :
  - *temperature, air*
  - *ambient humidity*

- outputs : *watering duration*
- universe: temperatures (*U1*); humidity levels (*U2*); durations (*U3*)

- classes (simplified):
  - *U1* (*t*°): cold, hot (2 classes to simplify)
  - *U2* (degree of humidity): dry, wet
  - *U3* (watering duration): short, long

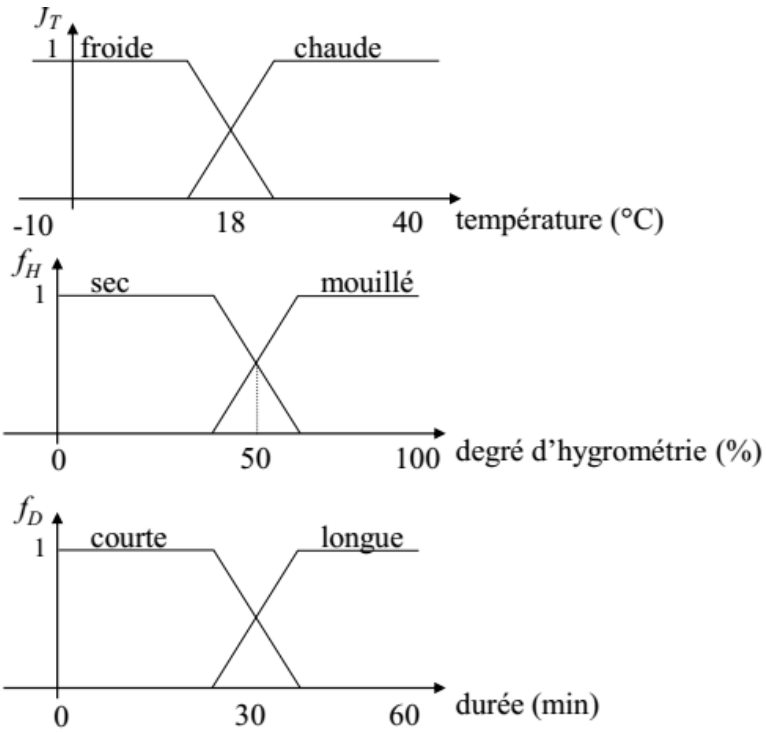


Fig. 4.20 : Inputs (temperature and humidity) and outputs (watering duration).

- Rules:
  - R1 : If the temperature is warm AND the soil is dry THEN the watering duration is long
  - R2 : If the temperature is cold AND the soil is wet THEN watering duration is short
  - R3 : If the temperature is hot AND the soil is wet THEN the watering duration is short
  - R4 : If the temperature is cold AND the soil is dry THEN the watering duration is long
- inputs:
  - measurement of *t*° : $t_0 = 23°C$
  - measurement of humidity in the air : $h_0 = 42$ %

- Fuzzification of inputs
  - measurements of inputs assumed to be exact therefore *singleton*

- Graphical construction of the fuzzy control output

  1) For each rule, define $f_T(t_0)$ et $f_H(h_0)$

  2) Report the minimum of the 2 values on the SEF D of the output (intersection operator: min function)

  3) construct the elementary fuzzy cde of the rule R_i (fuzzy implication and inference mechanism)

  4) take the maximum of elementary solutions (aggregation of rules by use of the union operator)

  5) defuzzify the SEF obtained: obtaining y_0 by equality of the integrals

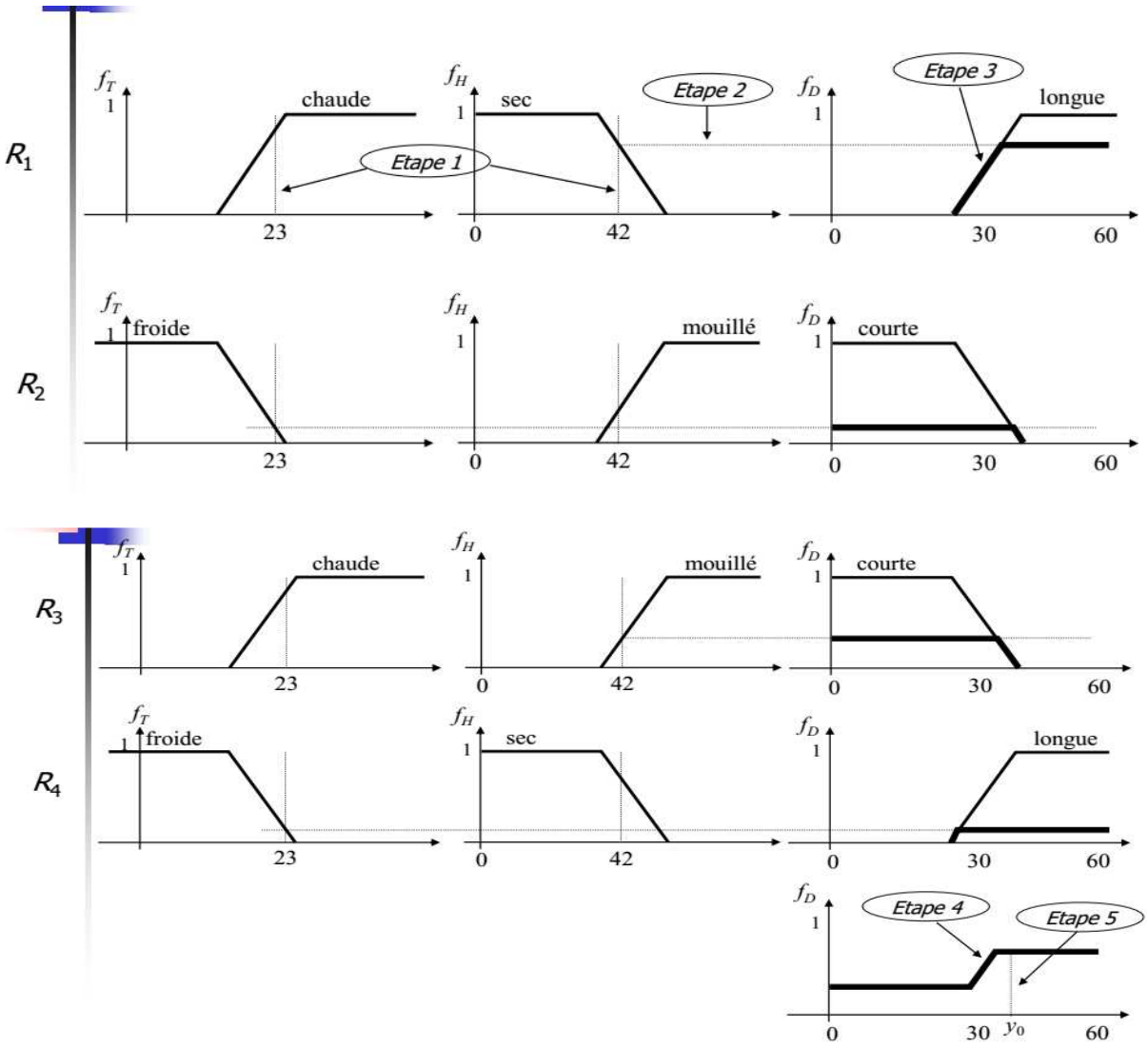


Fig. 4.21 : Rules and aggregation of output rules.

4.5.2. Example 2: Fuzzy speed control of a separately excited DC motor [23]

The objective of this case study is to perform the speed control of a separately excited DC motor (figure 1) using fuzzy logic controller (FLC). The controller will be designed based on the expert knowledge of the system. For the proposed dc motor case, we recommend **7 fuzzy rules** for fuzzy logic controller.

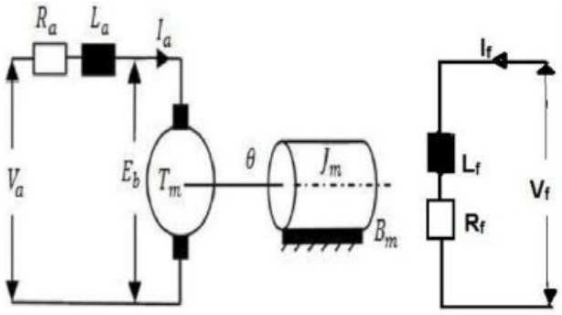


Figure 4.22. Separately excited DC motor.

The structure of the fuzzy logic controller with closed loop (synopsis of all system with fuzzy controller):

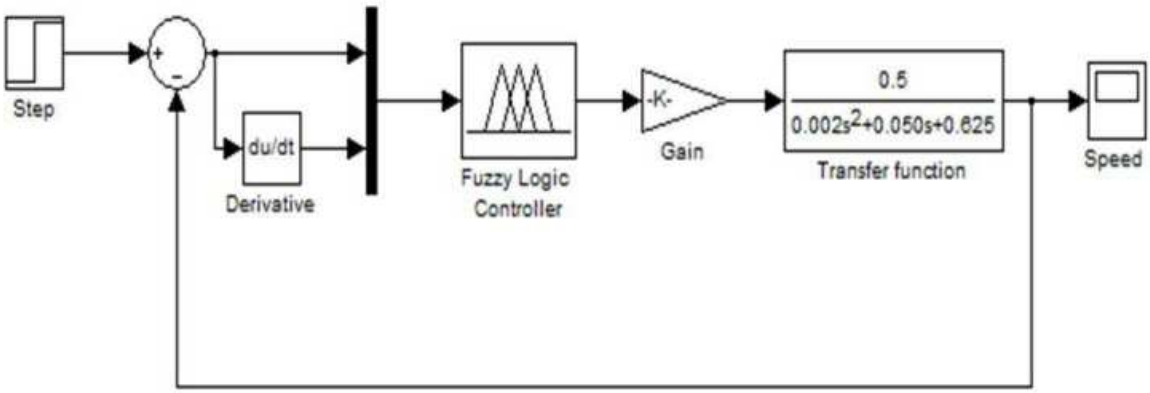


Figure 4.23. Fuzzy controller of separately excited DC motor.

We define the required fuzzy controller inputs and outputs. Then complete the diagram below Figure 4.24:

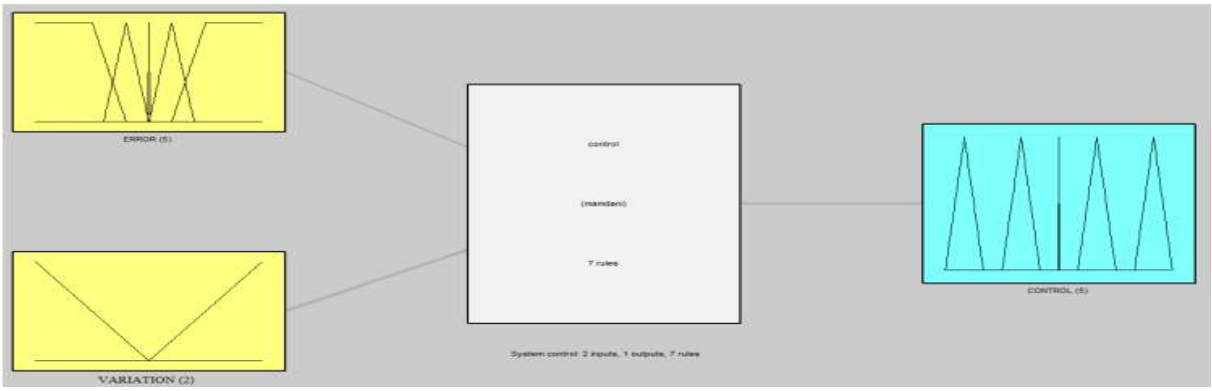
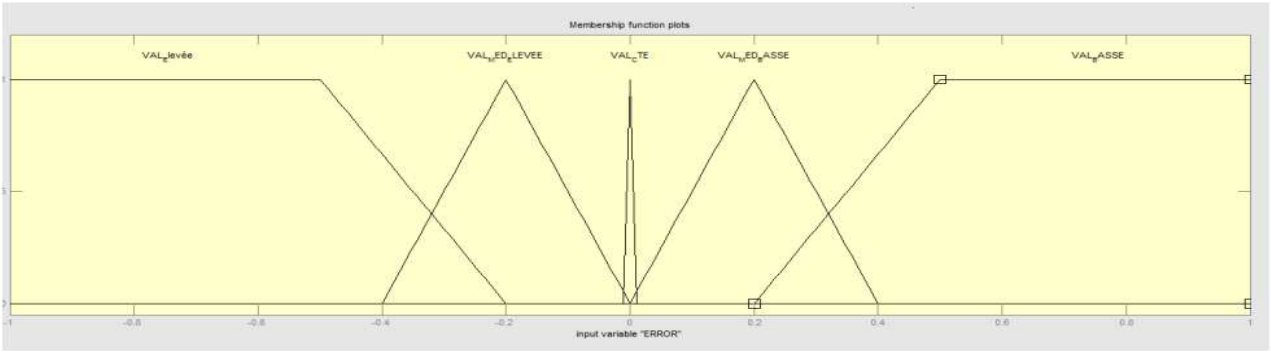


Figure 4.24. Matlab fuzzy interface of separately excited DC motor controller.

Input 1: Error



Input 2 : Variation of Error

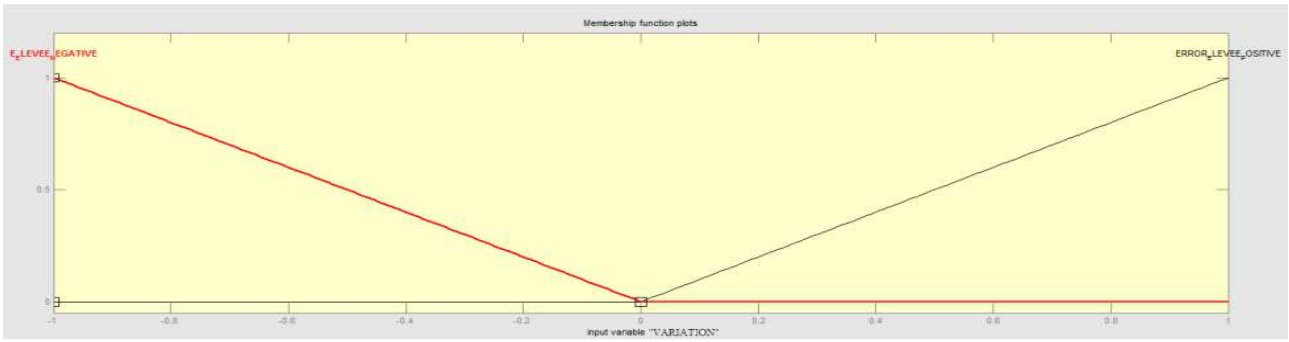


Figure 4.25. Intputs variable control.

Output : CONTROL

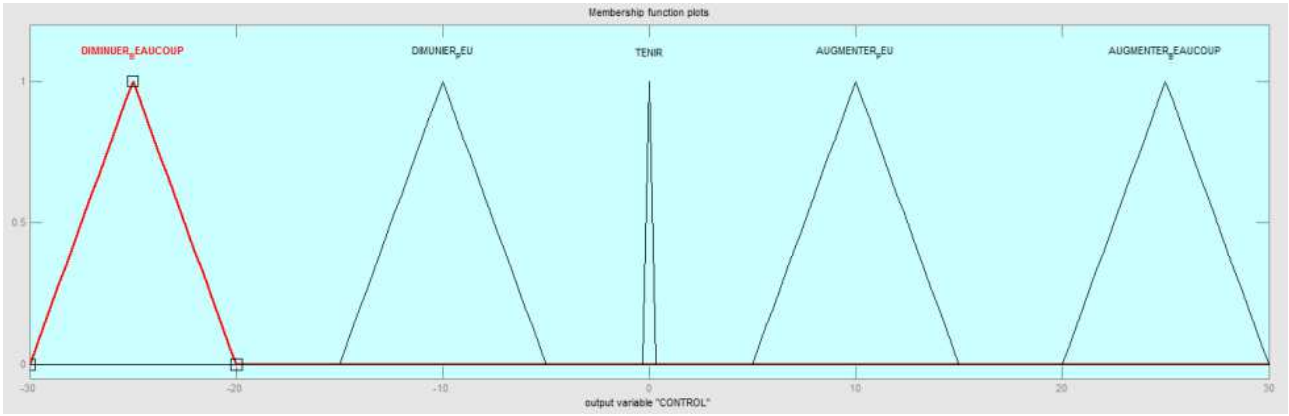


Figure 4.26. Output variable control

7 "If-then" rules necessary for separately excited dc motor speed control:

- If (ERROR is VAL_BASSE) then (CONTROL is AUGMENTER_BEAUCOUP)
- If (ERROR is VAL_Elevée) then (CONTROL is DIMINUER_BEAUCOUP)
- If (ERROR is VAL_CTE) and (VARIATION is E_ELEVEE_NEGATIVE) then (CONTROL is DIMUNIER_PEU)
- If (ERROR is VAL_CTE) and (VARIATION is ERROR_ELEVEE_POSITIVE) then (CONTROL is AUGMENTER_PEU)

- If (ERROR is VAL_CTE) and (CAMBIO is ERROR_ELEVEE_POSITIVE) then (CONTROL is AUGMENTER_PEU)
- If (ERROR is VAL_MED_BASSE) then (CONTROL is AUGMENTER_PEU)
- If (ERROR is VAL_CTE) then (CONTROL is TENIR)

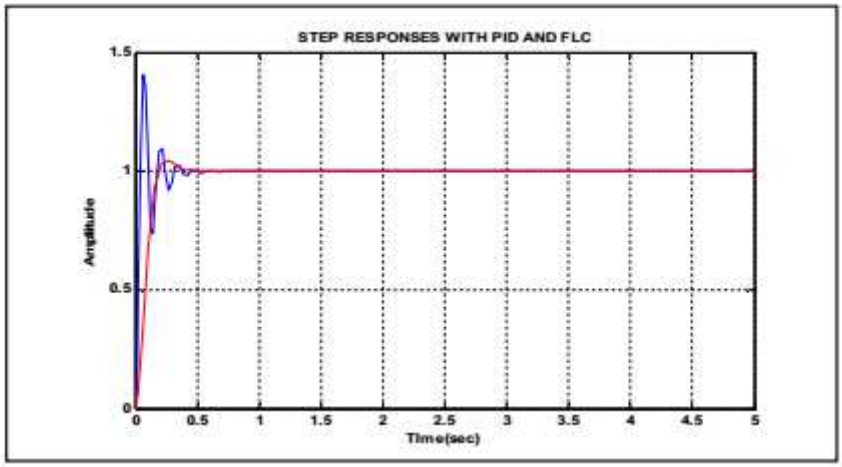


Figure 4.27. Step responses of system using PID and fuzzy logic controller. [23]

Figure 4.27. shows that the response of the system has greatly improved on application of fuzzy logic controller (FLC). The overshoot of the system using FLC has been reduced, settling time, peak time of the system also shows appreciable reduction.

4.6. References

[30] Noureddine Ouadah, Ouahiba Azouaoui, Mustapha Hamerlain, «*Implémentation d'un contrôleur flou pour la navigation d'un robot mobile de type voiture*». 2005
[31] Kailan Shang Zakir Hossen, «*Application de la logique floue à l'évaluation des risques et à la prise de décisions*», 2013.
[32] «Logique Floue», *http://isis.truck.free.fr › Site › Enseignement_files*
[33] «*Utilisation de la logique floue: contrôleur en Simulink*». Fuzzy Logic Toolbox - MATLAB
[34] Antoine Cornuéjols, «*Cours d'I.A. "Introduction à la logique floue*»,
[35] F. Chevrie F. Guély, «*Cahier technique n° 191 La logique floue*»,2015.
[36] Franck Dernoncourt, «*Introduction à la logique floue* », 2012.
[37] Sabeur Elkosantini, «*Simulation du comportement des opérateurs dans les systèmes de production : Application au contrôle de la qualité*», Institut Français de Mécanique Avancée, 2007.
[38] Rekha kushwah, Sulochana Wadhwani, «*Speed Control of Separately Excited DcMotor Using Fuzzy Logic Controller*», *International Journal of Engineering Trends and Technology (IJETT) - Volume4 Issue6*- ISSN: 2231-5381 http://www.ijettjournal.org Page 2518, *June 2013*

Chapter 5: Neural networks and applications in electrical engineering

5.1. Introduction

'Imitate the human brain'

The inspiration behind the technique of «formal neural networks», commonly referred to as a «neural network» comes from the fact that the human brain is a learning system that is not based on the principles of formal logic but on a structure, the human brain, containing approximately 100 billion neurons linked together by 10,000 synaptic contacts, *i.e.* approximately one million billion synapses. Formal neural networks are an attempt to mimic the learning mechanism that occurs in the brain.
Neurons are considered the physical support of intelligence. They fascinate because understanding and knowing how to use intelligence makes it possible to achieve unimaginable goals. For several years, we have been trying to copy neural networks to create intelligent control laws.

5.2.1. Historical elements

- 1890: W. James, famous American psychologist introduces the concept of memory associative, and proposes what will become an operating law for learning on neural networks, later known as Hebb's law.

- 1943: J. Mc Culloch and W. Pitts leave their names to a model of the biological neuron (a neuron with binary behavior). They are the first to show that simple formal neural networks can perform logical, arithmetic and complex symbols (at least at the theoretical level).
As early as 1943, Mac Culloch and Pitt proposed formal neurons mimicking biological neurons and capable of memorizing simple Boolean functions. The artificial neural networks made from these types of neurons are thus inspired by the nervous system. They are designed to reproduce certain characteristics of biological memories by the fact that they are:

- massively parallel;
- able to learn;
- able to memorize information in connections
- able to process incomplete information.

- 1949: D. Hebb, American physiologist explains conditioning in animals by the properties of the neurons themselves. Thus, a Pavlovian-like conditioning such that, feeding a dog at the same time every day causes the animal to secrete saliva at this precise time even in the absence of food. The law of modification of properties connections between neurons that he proposes partly explains this type of result experimental.

5.2.2. The first successes

- 1957: F. Rosenblatt develops the Perceptron model. He built the first neurocomputer based on this model and applied it to the field of pattern recognition.

It should be noted that at that time the means at his disposal were limited and it was a technological feat to succeed in making this machine work correctly for more than a few minutes.

- 1960: B. Widrow, an automation engineer, develops the Adaline model (Adaptive Linear Element). In its structure, the model resembles the Perceptron, however the law learning is different. This is the origin of the backpropagation algorithm of gradient widely used today with multilayer Perceptrons. Adaline-type networks are still used today for certain specific applications. B. Widrow created at this time one of the first firms offering neuro-computers and neuro-components, the "Memistor Corporation". He is now the president of the International Neural Network Society (INNS) to which we will return in the chapter Practical information.

- 1969: M. Minsky and S. Papert publish a book that highlights the limitations perceptron theory. Limitations then known, in particular concerning the impossibility of treat by this model of the nonlinear problems. They implicitly extend these limitations to all models of artificial neural networks. Their objective has been achieved, there is financial abandonment of research in the field (especially in the U.S.A.), researchers are turning mainly to AI and rule-based systems.

5.3. Apps

- statistics: data analysis / forecasting / classification
- robotics: control and guidance of robots or autonomous vehicles
- imagery / pattern recognition
- signal processing
- learning simulation

5.4. View of several biological neurons

Biological neurons

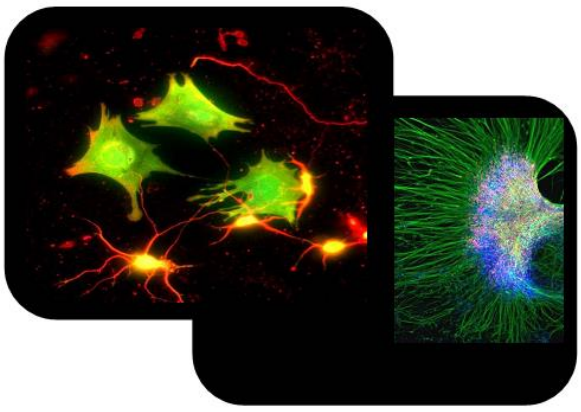


Figure 5.1. Biological neurons.

5.5. Real neuron

Real neurons have three main regions: the cell body, the dendrites relatively short, treelike extensions of the cell body - and the axon, a long, fibrous extension. A neuron uses dendrites to gather input data from other neurons. This input data is combined to produce a response sent to other neurons or other cells. Axons carry impulses from the cell body to other cells (the length of an axon is highly variable; it can reach 1 m in humans and nearly 10 m in giraffes).

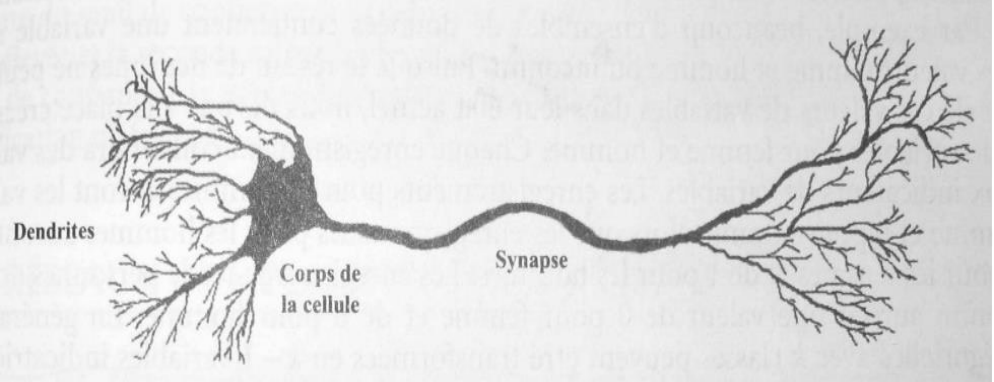


Figure 5.2. Schematic of a real neuron.

5.6. Formal neuron

The input data (x_i) is collected from the upper flow neurons in the data set, and is combined in a combinatorial function such as the sum. This combinatorial function is the input of an activation function which produces a response sent as input to other neurons.

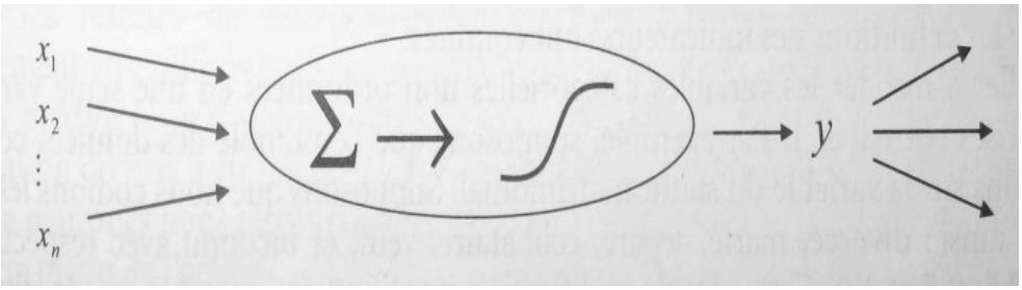


Figure 5.3. Schematic of a formal neuron.

- Principles

- no notion of time,
- synaptic coefficient: real coefficient,
- summation of the signals arriving at the neuron,
- output obtained after application of a transfer function,

- **Exemple**

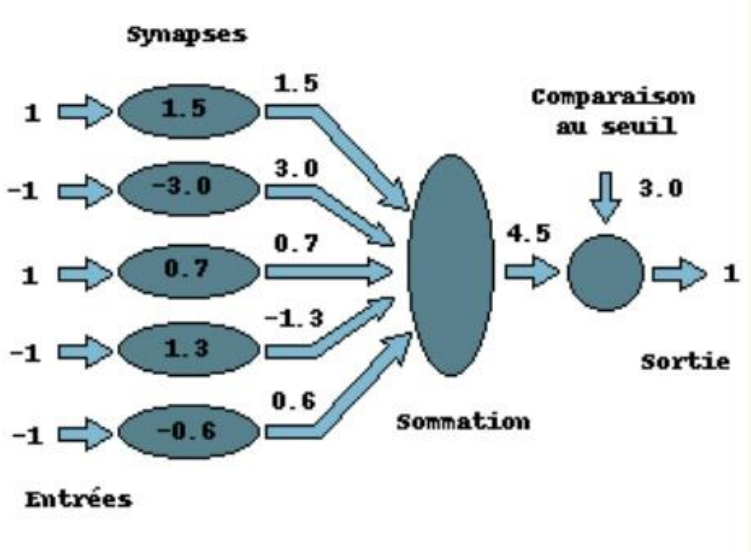


Figure 5.4. Example Neural network structure.

- **Modeling**

The neuron receives the inputs x_1, …, x_i, …, x_n. The activation potential of the neuron $\boldsymbol{p}$ is defined as the weighted sum (the weights are the synaptic coefficients $\boldsymbol{w_i}$) of the inputs. The output $\boldsymbol{o}$ is then calculated according to the threshold $\boldsymbol{\theta}$.

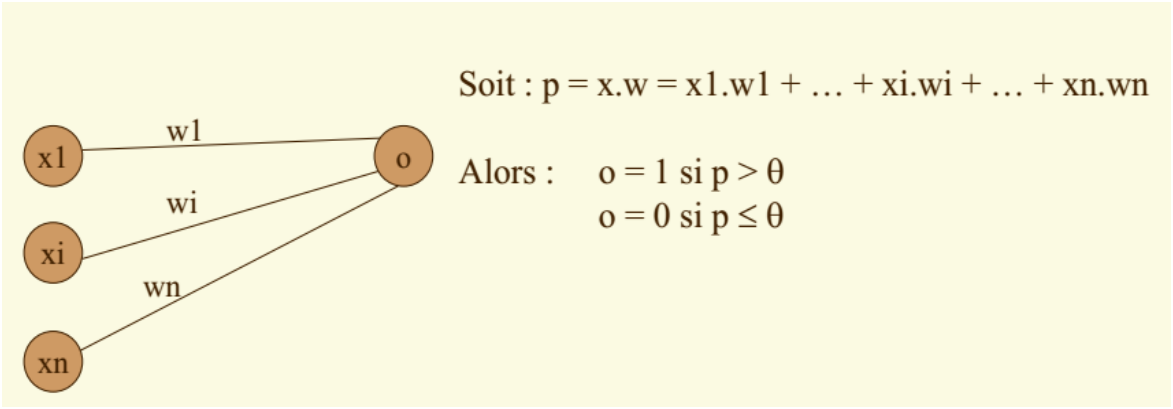


Figure 5.5. Example Neural network output modeling.

- **From biological neuron to formal neuron**

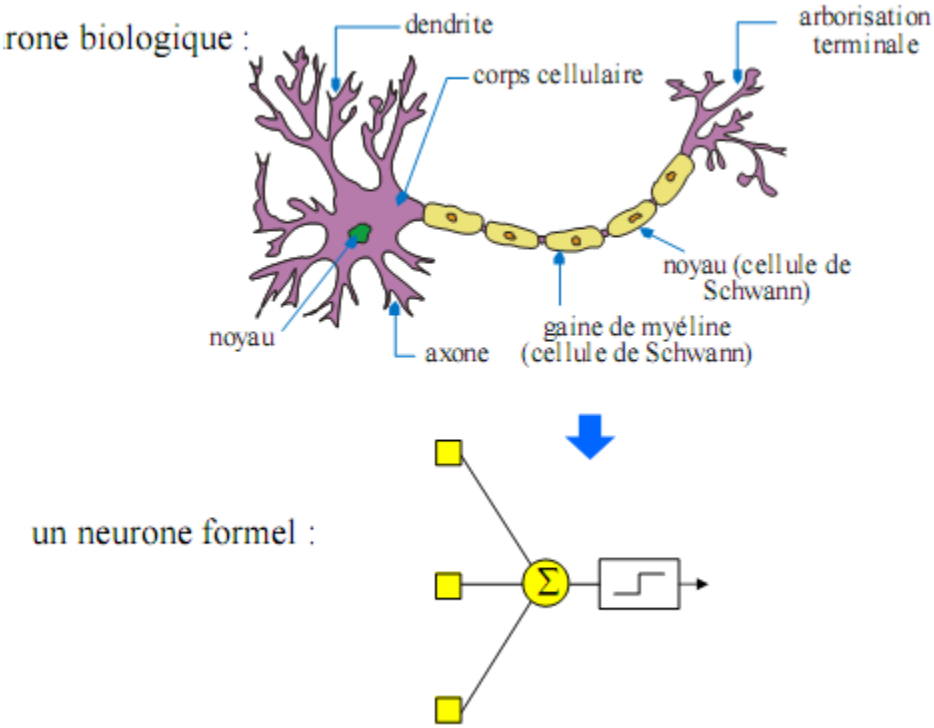


Figure 5.6. From biological neuron to formal neuron.

- **Architecture of a formal neuron**

The formal neuron: example with 3 inputs;

A neuron is a cell that 'focuses', processes them and translates them into a response.

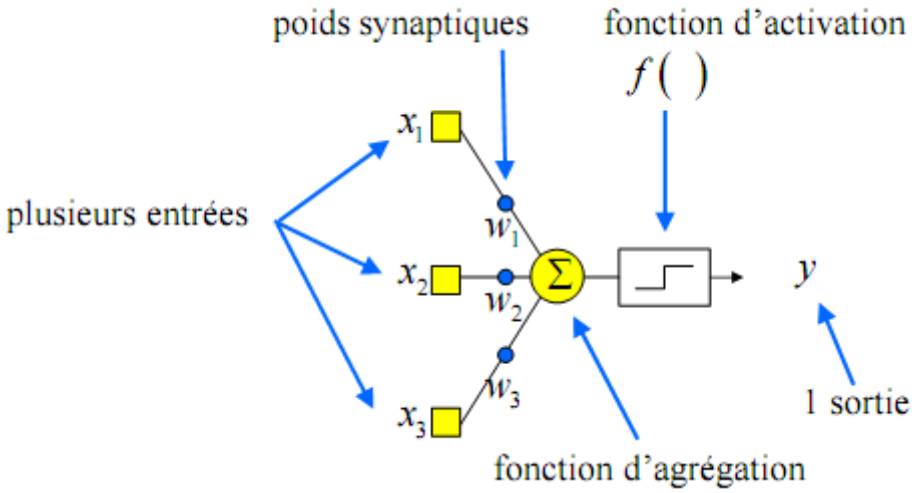


Figure 5.7. Architecture of a formal neuron.

5.7. Advantage of Neural Networks

- Robust to noisy data.
- Allow to model wide varieties of behaviors.

5.8. Disadvantages

- The results are quite opaque, unlike the decision tree method.
- The implementation, which requires an apprenticeship, can be long.

5.9. Architecture and operating principles

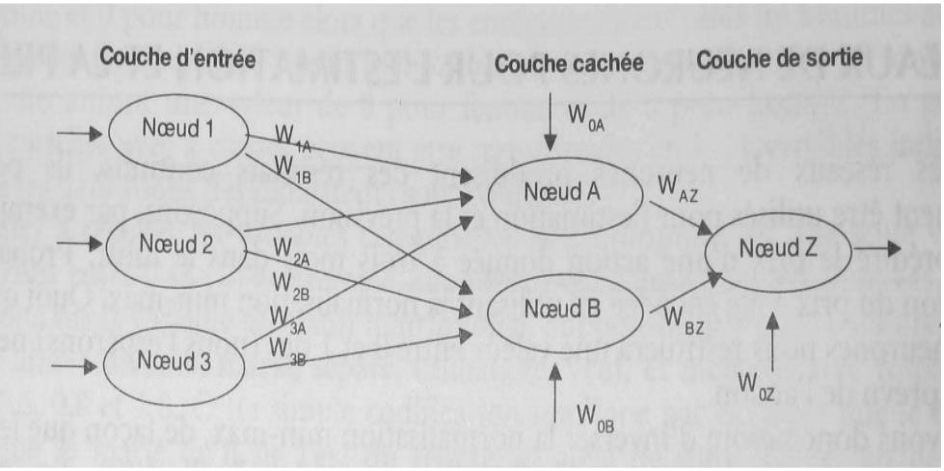


Figure 5.8. Example of a small neural network.

- A formal neural network is arranged in layers of formal neurons.
- Neurons are called “nodes”.
- Most networks consist of 3 successive layers: an input layer, a hidden layer and an output layer. However, there can be 0 or *N* hidden layers.
- From one layer to another, all the nodes of the first layer (“in” nodes) are connected to all the nodes of the second (“out” nodes).
- Each link has a weight: a value between 0 and 1.
- Each node in the hidden and output layers also has a weight: a value between 0 and 1.
- The number of input layer nodes depends on the number of variables taken into account and their type. Simplifying, we can say that we have one node per variable in entrance.
- The number of hidden layers and the number of nodes for each hidden layer is user configurable.
- In general, the output layer contains only one node. However, it can contain more. Simplifying, we can say that this node corresponds to the output variable.

5.10. Input data type

The value of the input and output data must be between 0 and 1.

- **Processing of numeric variables**

A "*min-max*" standardization is applied to the numerical data:

$$x' = (x\text{-}avg(X)) / (max(X)\text{-}min(X))$$

If we apply the results to a population in which the *min* and *max* have changed, erroneous results may be obtained.

- **Treatment of categorical variables**

If they are ordered, each category can be assigned a value between 0 and 1. If they are not ordered, the previous method risks leading to erroneous results due to the creation of unreal neighborhoods. Each category can then be treated as a Boolean variable.

5.11. Exploitation of results

The results are between 0 and 1.

Classification: Classes can be created, the number of classes chosen defining the amplitude.
Example: 4 classes of amplitude 1/4 = 0.25.

Forecast of a variable X: For forecasting, the result will be "denormalized":

$$x' = x * (max(X)\text{-}min(X)) + min(X)$$

Example:
Price forecast of a stock whose *min* is 20, *max* 30 and network exit 0.69:
Forecast = 0.69 * (30-20) + 20 = 26.9

5.12. Hidden Layer Parameterization

One can choose the number of nodes of the hidden layer and the number of hidden layer. The more the number of nodes increases, the more the network is able to identify complex phenomena. However, too many nodes lead to overfitting in the training sample which is ultimately harmful to the test samples.

5.13. Values of nodes and links

- During initialization, a weight is given randomly to each link and to each hidden and output layers node. The adjustment of these weights represents the key to the mechanism of learning by the neural network.
- For an individual, the nodes of the input layer take the normalized value of the model input variables.

- For an individual, the nodes of the hidden and output layers take a value which is a combination (most often a sum) of the linear combinations of the "***in***" nodes and the corresponding weights.

For a given node *j*, we therefore have: *NET j* = *Sum* for *i* from 0 to *N* ($W_{ij} * X_i$)

With

NET: value of the node in the network.

i: ranging from 0 to *N*, *N* being the number of "*in*" nodes.

Wi j: weight of the link between node "*i*" which is "in" and node "*j*" which is "out".

X_i: value of node "*i*", with $X_0 = 1$.

5.14. The sigmoid function

In a real neuron, signals are sent between neurons when the combination of input data exceeds a certain threshold: the activation threshold. The behavior is not linear because the response does not depend linearly on the stimulation increment. The function that models this behavior is called: activation function. It is a nonlinear function. The most common activation function is the sigmoid function:

$$y = 1 / (1 + exp(- x)) \quad (5.1)$$

Either :

$$\text{SIG}(\text{NET}(n)) = 1 / (1 + exp(-\text{NET}(n))) \quad (5.2)$$

With :

exp : exponential function: *exp*(1) = 2,7.

NET(*n*) : *NET* of the node « *n* ».

SIG (NET(*n*)) : sigmoid of *NET* of node « *n* »

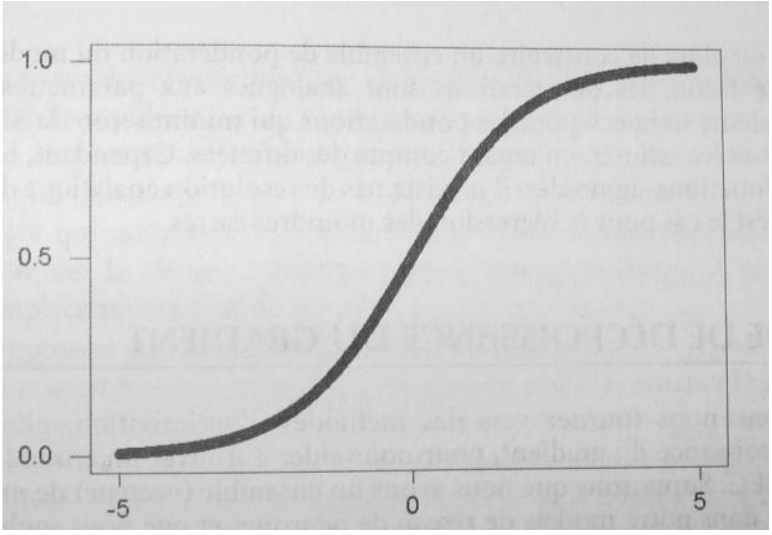


Figure 5.9. Graph of the sigmoid function.

The sigmoid function is such that when the input data is close to the center of the interval, $f(x)$ is linear. When the input data moves away from the center, $f(x)$ is curvilinear. When the data is very far from the center, $f(x)$ becomes almost constant. So increments of the NET of a node produce variable increments of SIG(NET): near the center, an increment of the NET produces a linear increment of the SIG. The further away from the center, the less effect the NET increment has on the SIG. Far from the center an increment of the NET does not produce an increment of the SIG.

The sigmoid function is also called: crushing function: it crushes the extremes. We will apply the sigmoid function to the NET value of each node.

5.15. SEC

Neural networks are a supervised method: a target variable is chosen. Each individual with its input variables passes through the network and provides a result in the output node. This output value is compared to that of the target variable.

$$\textit{Forecast error = actual data value - output value} \quad (5.3)$$

This error is analogous to that of regression models. In general, neural network models calculate a sum of squared errors (SEC):

$$\textit{SEC = Sum for all records (actual data – output data)}^2 \quad (5.4)$$

The problem therefore consists in minimizing the value of SEC according to the set of weight values of nodes and links.

5.16. Backpropagation

Due to the nonlinear nature of the sigmoid function, there is no analytical solution for minimizing the SEC. Backpropagation implements complex mathematical and algorithmic calculations that we do not present here.

We only present the main concepts and parameters that come into play.

- **SEC gradient decay method to adjust weights**

To minimize the SEC, we use the "gradient decay method" which gives the direction in which to adjust the weighting to decrease the DRY.

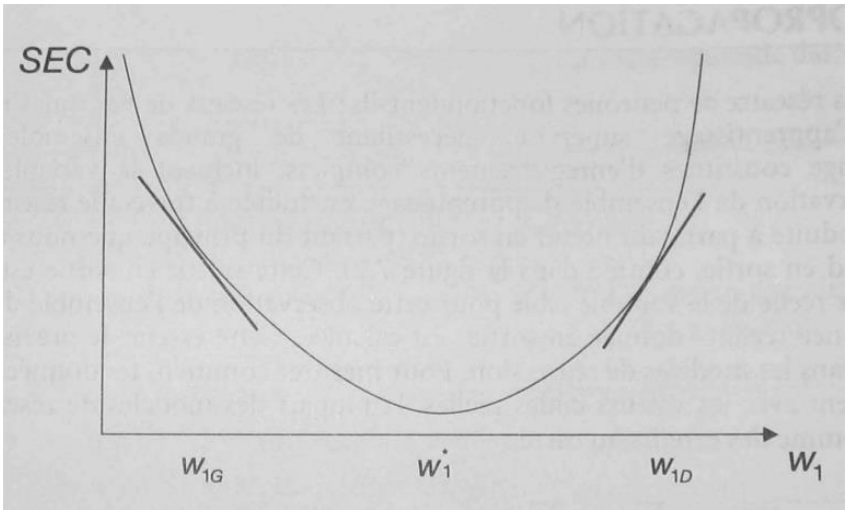


Figure 5.10. Curve of the evolution of the SEC according to weight.

The curve above shows a parabolic evolution of the SEC according to a single weighting. It is a simplification which allows showing that the derivative of the curve gives the slope and tells us which way to adjust the weight.

- **Backpropagation**

Backpropagation consists of adjusting the weights of nodes and links going back from the output layer node to the input layer nodes.

In general, networks update after each calculation of the output value of a registration. This adjustment will depend on:

- The forecast error;
- The learning rate: value between 0 and 1.

- **The learning rate (eta)**

The learning rate is a parameter that favors the evolution of the SEC towards the minimum.

When the learning rate is low, the adjustments are small.

When the learning rate is strong, the adjustments are strong. But too high a learning rate exceeds the optimum SEC. The learning rate may change during learning. In the beginning, it is high to quickly approach the solution. As the network begins to converge, the rate is gradually reduced so as not to exceed the optimum SEC.

- **The moment term (alpha)**

The moment term is an additional parameter that favors the evolution of the SEC towards the minimum. Intuitively, we can understand its operation as follows: the curve of evolution of the SEC according to the weights is not a simple parabola. It contains several minima or

"steps". The term of moment makes it possible to avoid that the search for the best minimum stops at an intermediate level or that it is before or after the best level.

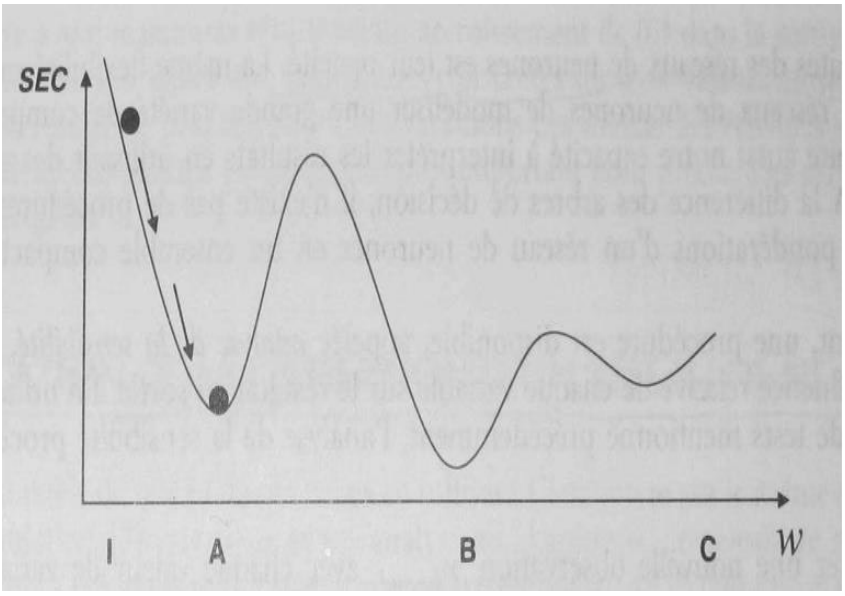


Figure 5.11. The evolution curve of the SEC according to weight.

We can interpret this curve by saying that the moment term favors the fact of not stopping at landing ***A***, not going to landing ***C***, stopping at landing ***B***.

5.17. Stopping criteria

The algorithm can process all records of the training data set an indefinite number of times. Therefore, a stopping criterion must be determined.

- **Time: to be avoided**

Neural network modeling can take several hours! Time can therefore be a stopping criterion if you are in a hurry, but it risks leading to an inefficient model.

- **Minimize the SEC in the learning set: avoid!**

The SEC can be a stopping criterion but it risks leading to over-learning by memorizing idiosyncratic characteristics (specific to individuals regardless of their group).

- **Minimize SEC in the validation set**

At the same time as we minimize the SEC in the training set, we verify that we also minimize it in the validation set. When it increases in the validation set, we are starting to enter the overlearning phase. This is a good stopping criterion.

- Reminder of the problem of overfitting: The model must achieve the minimum error rate for the validation set.

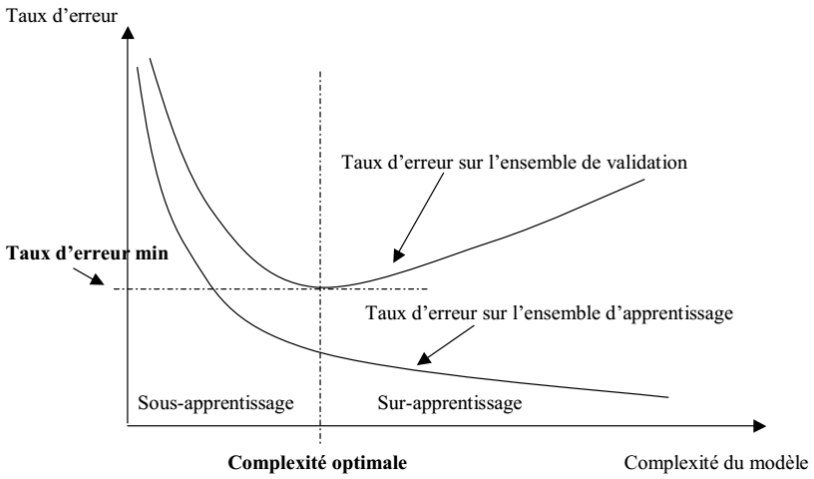


Figure 5.12. SEC function in validation and training set.

5.18. Model with multiple target variables

We can have several target variables in a neural network. The output layer will then have as many nodes as there are target variables. The prediction will be made on the *n*-tulet of value of the target variables.

5.19. Interpretation of results: sensitivity analysis

A disadvantage of neural networks is their opacity. It provides a prediction function but this function is not translatable, as in decision trees, into a set of intuitively understandable rules. However, one can measure the relative influence of each variable on the output result. Sensitivity analysis performs this measurement.

5.20. Definitions

- Determine a neural network = Find the coefficients synaptics.
- We speak of a learning phase: the characteristics of the network are modified until the desired behavior is obtained.
- Learning base: representative examples of the behavior or of the function to model. These examples are in the form of known (input; output) pairs.
- Test base: for any input (noisy or incomplete), calculate the output. The performance of the network can then be evaluated.
- Supervised learning: the synaptic coefficients are evaluated by minimizing the error (between desired output and obtained output) on a learning basis.
- Unsupervised learning: there is no basic of learning. Synaptic coefficients are determined by relation to conformity criteria: general specifications.

- over-fitting: the error is minimized on the learning base at each iteration but the error is increased on the test basis. THE model loses its ability to generalize: this is learning by heart. ⇒ Explanation: too many explanatory variables in the model; we no longer explains the global behavior but the residuals.

5.21. Learning

Learning is probably the most interesting property of networks neural. However, it does not concern all models, but the most used.

Definition:

Learning is a phase in the development of a neural network during which the behavior of the network is modified until the desired behavior is obtained. Neural learning uses examples of behavior.

In the case of artificial neural networks, the learning algorithm is often added to the description of the model. The model without learning is indeed of little interest. In the majority of current algorithms, the variables modified during learning are the weights of the connections. Learning is the modification of network weights in order to tune the network's response to examples and experience. It is often impossible to decide a priori on the values of the weights of the connections of a network for a given application. At the end of learning, the weights are fixed: this is then the phase of use. Some network models are incorrectly called permanent learning. In this case it is true that the learning never stops, however one can always distinguish a phase of learning (in fact updating behavior) and a phase of use. This technique allows the network to maintain an appropriate behavior despite fluctuations in the input data.

At the level of learning algorithms, two major classes have been defined depending on whether the learning is said to be supervised or unsupervised. This distinction is based on the form of the training examples. In the case of supervised learning, the examples are pairs (Input, Associated Output) whereas we only have values (Input) for unsupervised learning. Note however that unsupervised learning models require, before the use phase, a labeling step carried out by the operator, which is nothing more than a part of supervision.

5.21.1. Hebb's law, an example of unsupervised learning

Hebb's law (1949) applies to connections between neurons, as shown in Figure 5.13.

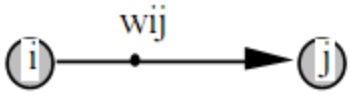


Figure 5.13. *i* the upstream neuron, *j* the downstream neuron and w_{ij} the weight of the connection.

It is expressed as follows: "*If 2 cells are activated at the same time then the strength of the connection increases*". The change in weight depends on the coactivation of the presynaptic and postsynaptic neurons, as shown in Table 1. x_i and x_j are the activation values of neurons *i* and *j* respectively, δw_{ij} (partial derivative of the weight) corresponds to the change in realized weight.

Table 1. Hebb's law.

x_i	x_j	∂w_{ij}
0	0	0
0	1	0
1	0	0
1	1	+

Hebb's law can be modeled by the following equations $w(t+1)$ is the new weight, $w_{ij}(t)$ the old one):

$$w\,(t+1) = w_{ij}\,(t) + \delta w_{ij}\,(t) \qquad (5.5)$$

Where:

$\delta w_{ij}\,(t) = x_i \,.\, x_j$ (coactivity is modeled as the product of the two activation values)

The learning algorithm iteratively (little by little) modifies the weights to adapt the response obtained to the desired response. It is in fact a question of modifying the weights when there is an error only.

(1) Initialization of the weights and the threshold *S* to randomly chosen (small) values.

(2) Presentation of an input $E = (e_1 , \ldots e_n)$ of the learning base.

(3) Calculation of the output obtained x for this input:

(4) $a = \sum (w_i . e_i) - S$ (the threshold value is introduced here in the calculation of the weighted sum) $x = \text{sign}\,(a)$ (if $a > 0$ then $x = +1$ else $a \leq 0$ then $x = -1$)

(5) If the output *x* is different from the desired output d1 for this example of input E1 then modification of the weights (μ is a positive constant, which specifies the step of modification of the weights): $w_{ij}(t+1) = w_{ij}\,(t) + \mu.(x_i.x_j)$

(6) As long as all the examples of the learning base are not processed correctly (*i.e.* modification of the weights), return to step 2.

Example application of Hebb's learning algorithm

Let us choose a binary behavior for the neurons. The inputs e1 and e2 are considered as neurons (Fig. 5.14).

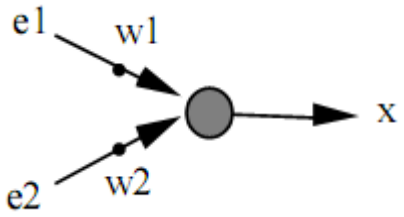


Figure 5.14. Network of 3 neurons (the 2 inputs are considered as two neurons) for the resolution of the problem expressed in table 2.

We are going to learn about a very simple problem. The learning base is described by table 2:

Table 2.

e_1	e_2	x	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

1.1. Initial conditions: μ = +1, the weights and the threshold are zero.

1.2. Let's calculate the value of x for example (1):

1.3. $a = w_1.e1 + w_2.e2 - S = 0.0.\ 1 + 0.0\ .\ 1 - 0.0 = 0$ $a \leq 0 \Rightarrow x = -1$

1.4. The output is false, it is therefore necessary to modify the weights by applying:

$w_1 = w_1 + e_1.x = 0.0 + 1.1 = 1$

$w_2 = w_2 + e_2.x = 0.0 + 1.1 = 1$

2. We move on to the following example (2):

2.1. $a = 1.1 + 1.-1\ -0.0 = 0$ $a \leq 0 \Rightarrow x = -1$

2.2. The output is false, it is therefore necessary to modify the weights by applying:

$w_1 = 1 + 1.1 = 2$

$w_2 = 1 + 1.-1 = 0$

3. The following example (3) is correctly processed: $a = -2$ and $x = -1$ (the output is good). We pass directly, without modification of the weights, to example (4). This too is handled correctly. We then return to the beginning of the learning base: example (1). It is correctly treated, as well as the second (2). The learning algorithm is then complete: the entire learning base has been reviewed without modifying the weights.

5.21.2. The Perceptron learning rule, an example of supervised learning

Hebb's rule does not apply in certain cases, although a solution exists (see exercise in the previous paragraph). Another learning algorithm has therefore been proposed, which takes into account the error observed at the output.

The Perceptron learning algorithm is similar to that used for Hebb's law. The differences are in the modification of the weights.

(1) Initialization of the weights and the threshold S to randomly chosen (small) values.
(2) Presentation of an input E1 = (e_1, ... e_n) of the learning base.
(3) Calculation of the output obtained x for this input:

$$a = \sum (w_i \, . \, e_i) - S$$

$$x = \text{sign}\,(a) \text{ (if } a > 0 \text{ then } x = +1 \text{ else } a \leq 0 \text{ then } x = -1)$$

(4) If the output x of the Perceptron is different from the desired output d1 for this example of input E1 then modification of the weights (μ the step of modification):

$$w_i\,(t+1) = w_i\,(t) + \mu.((d1 - x).e_i) \qquad (5.6)$$

Reminder: d1 = +1 if E is class 1, d1 = -1 if E is class 2 and (d1 - x) is an estimate of the error.

(5) As long as all the examples of the learning base are not processed correctly (*i.e.* modification of the weights), return to step 2.
(6)

Example of operation of the Perceptron learning algorithm

e_1	e_2	d	
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

1.1. Initial conditions: $w_1 = -0.2$, $w_2 = +0.1$, $S = 0.2$, ($\mu = +0.1$)
1.2. $a(1) = -0.2 + 0.1 - 0.2 = -0.3$
1.3. $x(1) = -1$ (desired output d(1) = +1, hence modification of weights)
1.4. $w_1 = -0.2 + 0.1. (1 + 1) . (+1) = 0$
$w_2 = +0.1 + 0.1 . (1 + 1) . (+1) = +0.3$

2.1. $a(2) = +0.3 - 0.2 = +0.1$
2.2. $x(2) = +1$ False
2.3. $w_1 = 0 + 0.1. (-1 - 1) . (-1) = +0.2$
$w_2 = +0.3 + 0.1 . (-1 - 1) . (+1) = +0.1$

2-3/ $a(3) = -0.2 - 0.1 - 0.2 = -0.5$ Ok
2-3/ $a(4) = +0.2 - 0.1 - 0.2 = -0.1$ Ok
2-3/ $a(1) = +0.2 + 0.1 - 0.2 = +0.1$ Ok
2-3/ $a(2) = -0.2 + 0.1 - 0.2 = -0.1$ Ok

5/All the examples in the database have been correctly processed, learning is complete.

5.22. Different neural architectures [https://www.coursera.org/articles/neural-network-architecture]

The architecture of a neural network is the organization of neurons among themselves within the same network. In other words, it's about how they are ordered and connected (Figure 5.15). The majority of neural networks use the same type of neurons. Some rarer architecture is based on dedicated neurons. The architecture of a neural network depends on the task to learn (problem to solve). A neural network is generally made up of several layers of neurons, from inputs to outputs.

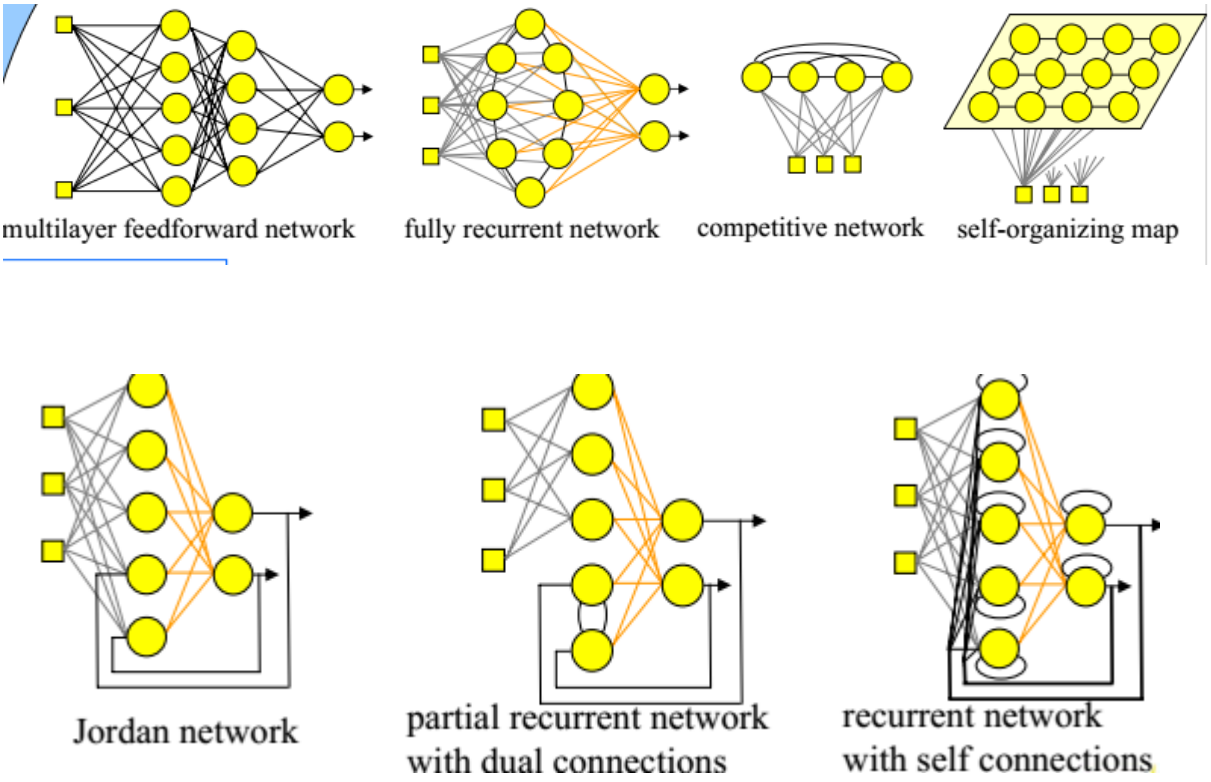


Figure 5.15. Different neural network architectures design.

Neural network architecture refers to the structure of the neural network or the number and types of layers. Let's learn more about these four types of neural networks and their architectures: feedforward neural networks, recurrent neural networks, convolutional neural networks, and generative adversarial networks.

Feedforward neural networks : A feedforward is one of the more basic forms of neural networks, and you can often use the architecture of a feedforward neural network to create more specialized networks. As the name suggests, feedforward neural networks feed data forward from input to output with no loops or circles. Although it's one of the simplest structures for neural networks, the hidden layers between input and output can still be complex. You can use this type of neural network for various tasks, such as pattern and image recognition, regression analysis, and classification.

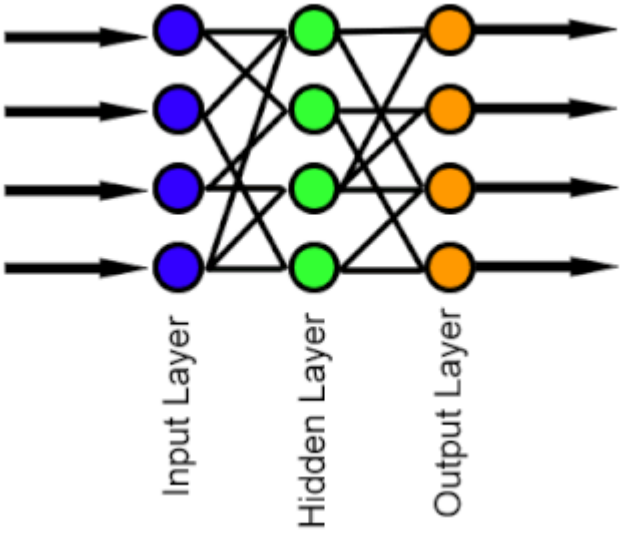


Figure 5.16. In a feedforward network, information always moves one direction; it never goes backwards.

How does a feedforward neural network architecture work? *A feedforward neural network has an input layer, followed by a series of hidden layers, and ends with an output layer. Data flows into the algorithm through the input and passes through the nodes in the first layer. The first layer of nodes computes the data based on the node's weights and passes the calculation to the next layer of nodes. Each node in each layer connects to each node in the next layer, but the data can only flow towards the output.*

Recurrent neural network : A recurrent neural network is a model used for sequential data or time series prediction. For example, a recurrent neural network can make stock market predictions by calculating what is likely to happen in the future based on what happened in the past. You can also use a recurrent neural network for tasks like translation, where the sequence of words changes based on the language, such as a noun before or after an adjective.

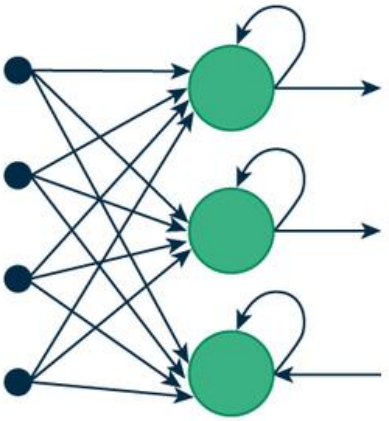


Figure 5.17.

How does recurrent neural network architecture work? In addition to the architecture found in the feedforward neural network, a recurrent network uses loops to circle the data back through the hidden layers before returning an output. Sometimes, recurrent neural networks include specialized hidden layers called context layers, which provide feedback to the neural network and help it become more accurate.

Convolutional neural networks: Convolutional neural networks are particularly skilled at recognizing patterns and images, which makes them important for AI technology like computer vision, among other uses. For example, the US Postal Services uses neural networks to recognize handwritten zip codes. Convolutional neural networks are different from other networks because of their architecture and because the CNN nodes have shared weights and bias values, unlike feedforward or recurrent neural networks. They have shared weight because each node does the same job in a different input area, such as detecting the edge of an image.

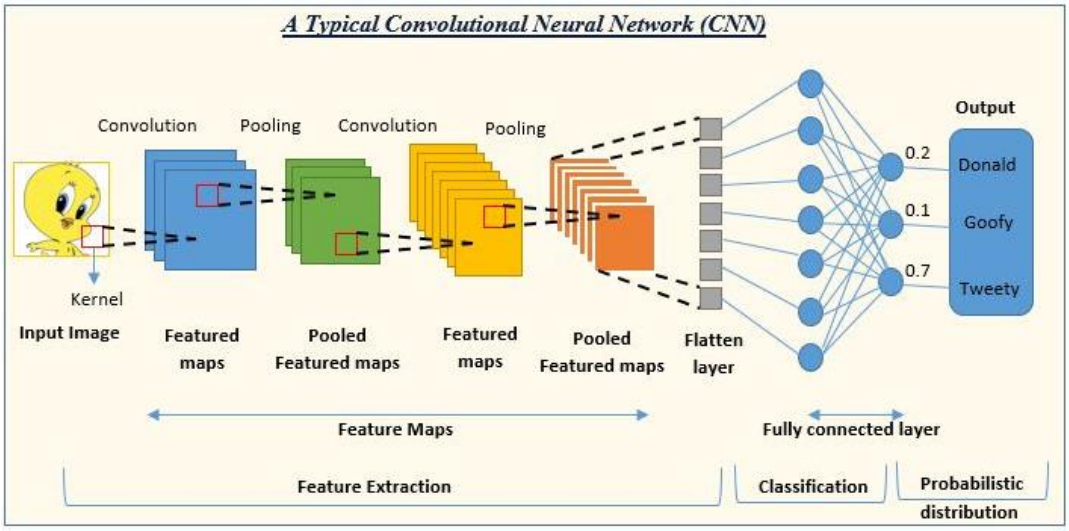


Figure 5.18. Generative adversarial networks.

How does a convolutional neural network architecture work? In addition to input and output layers, convolutional neural networks contain two main types of hidden layers: convolutional and pooling. Convolutional layers filter the input, typically an image, to extract various features. This data then feeds into a pooling layer, simplifying the parameters but keeping important information. The process repeats many times, sometimes including other layers, such as a multilayer perceptron or a rectified linear unit for activation.

Generative adversarial networks: A generative adversarial network differs from the models above because it is actually two separate networks. Working as a team, these two algorithms generate new content based on training data. One of these neural networks, the generator, creates a novel image or text based on training data. The second neural network, the discriminator, judges the generator's work to determine whether it looks real or fake. These two models go back and forth until the discriminator can't tell the difference between the real training data and the generator's fake work. Generative adversarial networks can create 3D models from 2D images, generate images, or create training data sets for other neural networks that are similar but different from existing data sets.

How does a generative adversarial network architecture work? The basic architecture of a generative adversarial network is two distinct neural networks working in tandem to produce an output from the input.

Within this category of neural networks are subtypes that have unique architectures, such as:

- **Vanilla GAN**: This is the basic version of a generative adversarial neural network that needs to be adapted for many specific real-world applications.
- **CycleGAN:** The cycle-consistent generative adversarial network, or CycleGAN, is useful for image-to-image translation, moving an image from one domain to another.
- **DCGAN:** Deep convolutional generative adversarial network, or DCGAN, leverages convolutional neural networks for more powerful image generation.
- **Text-2-image**: A text-2-image generative adversarial neural network can create novel images from text-based descriptions, such as adding specific eye color to a generated face.

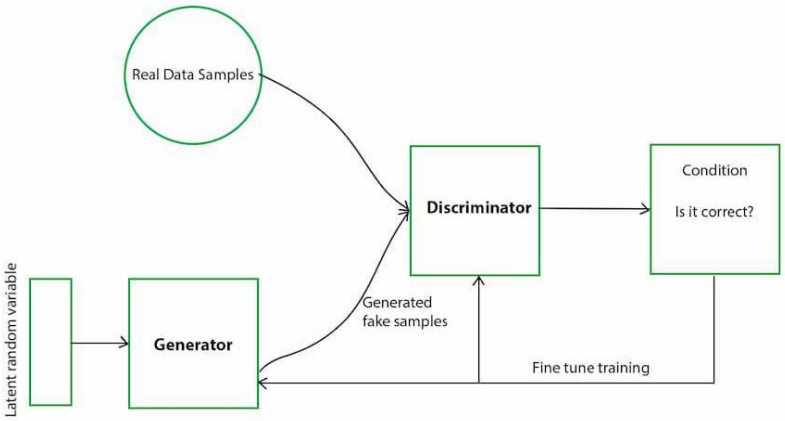


Figure 5.19. Adversarial network.

What are neural networks used for?

Although we have been studying and implementing neural networks since at least the 1940s, advancements in deep learning have guided us to work with the algorithms in new and advanced ways. Today, researchers and scientists can use neural networks for real-world applications in various fields, including the automotive industry, finance, national defense, insurance, health care, and utilities.

- **Automotive**: Self-driving cars use neural networks to make decisions based on the data they receive from their surroundings. Neural networks can also optimize vehicle parts and functions or estimate how many vehicles you need to make to meet demand.
- **Finance**: Neural networks have many uses in the finance industry, from predicting the performance of the stock market or exchange rates between monetary denominations to determining credit scores and default risks.
- **National defense**: The Department of Defense uses neural networks to simulate situational training, such as combat readiness. Other neural network applications in national defense include the ability to develop unmanned aircraft.
- **Insurance**: Insurance providers can use neural networks to model how often customers file insurance claims and the size of those claims.
- **Health care**: In a health care setting, doctors, health care administrators, and researchers use neural networks to make informed decisions about patient care, organizational decisions, and developing new medications.
- **Utilities**: Utility companies can use neural networks to forecast energy demand. Other uses include stabilizing electrical voltage or modeling oil recovery from residential areas.

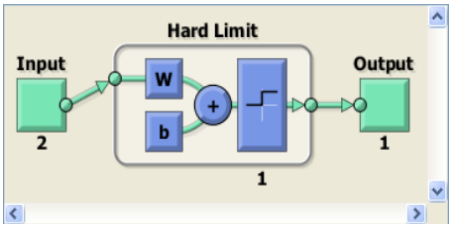
5.23. Neural network applications

Example 1: Classification of linearly separable data with a perceptron

Problem description: Two clusters of data, belonging to two classes, are defined in a 2 dimensional input space. Classes are linearly separable. The task is to construct a Perceptron for the classification of data.

```
close all, clear all, clc, format compact
% number of samples of each class
N = 20; % define inputs and outputs
offset = 5; % offset for second class
x = [randn(2,N) randn(2,N)+offset]; % inputs
y = [zeros(1,N) ones(1,N)]; % outputs
% Plot input samples with PLOTPV (Plot perceptron input/target
vectors)
figure(5.20)
plotpv(x,y);
net = perceptron;
net = train(net,x,y);
view(net);

figure(5.21)
plotpc(net.IW{1},net.b{1});
```



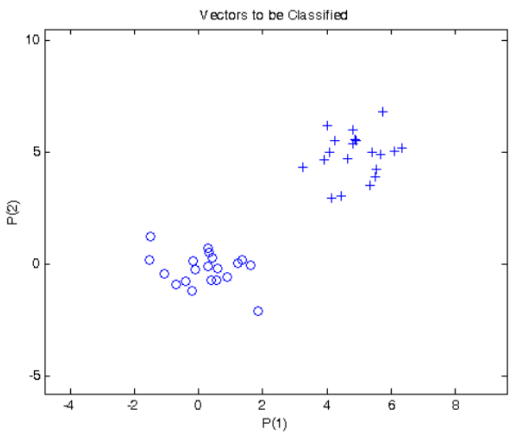


Figure 5.20. Inputs and target network.

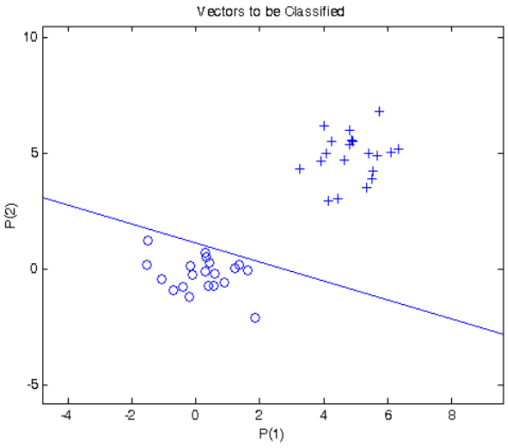


Figure 5.21. Classification results based Perceptron.

Example 2: Artificial neural network based speed control
[https://www.emo.org.tr/ekler/97d404b6119214e_ek.pdf]

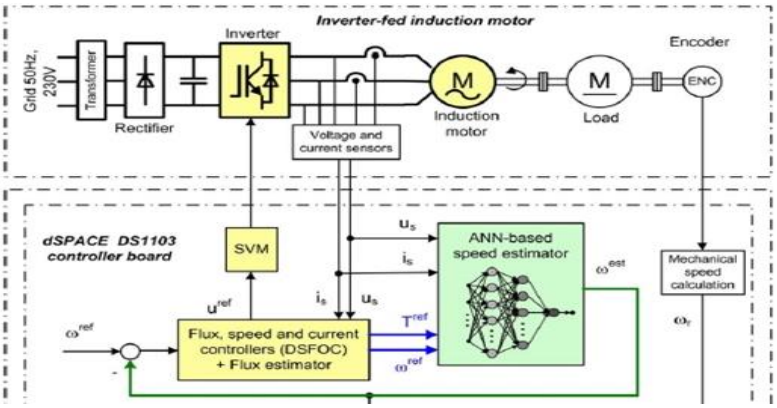


Figure 5.22. Artificial neural network based speed control of induction motor.

A multilayer recursive ANN with the structure 9-10-1 (9 input nodes , 10 nodes in hidden layer, 1 output node) was obtained by trial and error to estimate the instantaneous rotor speed. In figure 5.23, ANN-based speed estimator structure has been showed. The inputs of ANN are the sampled values of stator currents, stator voltages and past value of the rotor speed (usd (k), usd (k-1), usq (k), usq (k-1), isd (k), isd (k-1), isq (k), isq (k-1), wr(k-1)). In proposed scheme a past values of stator voltages and stator currents are used inputs for good performance. Stator currents and voltages are in the stationary reference frame. The output of ANN is rotor speed (wr (k)). When the trained data was applied in checking phase, it gave correct results for those data as well. The activation function of input and hidden layer is tansigmoid. The experiment results showed that when tansigmoid activation function was used instead of logsigmoid activation function the training phase become shorter. The activation function of output layer is linear activation function. Training of ANN was performed with data corresponding the unloaded motor. But when a load was applied to the motor, ANN gave correct results. Activation function, number of hidden layers, number of neurons in each layer has been selected by trial and error. Backpropagation algorithm was used in proposed ANN. The input are (voltages, currents) and output is (rotor speed=target) of training data. For training phase, all inputs and output were normalized between 0-1. Inputs were applied to the ANN and then error was obtained by difference between ANN's output and desired output. Backpropagation algorithm minimizes the error to desired value (sum squared error) by gradient descent method. In this algorithm weights were updated in training phase. In training phase 2800 data was used for each input and output (9 input + 1 output=10*2800=28000 data). Proposed ANN's sum squared error was selected 0.005. and after 168000 epochs the ANN achieved this target. Momentum coefficient was selected 0.95 and learning rate's default value was 0.00001 and it was adaptive. The parameters of motor which was used in experiments are given in [https://www.emo.org.tr/ekler/97d404b6119214e_ek.pdf].
After training phase various test data was applied to the proposed ANN. To show the system's performance, different conditions are applied (the parameter value (stator resistance) was changed and also the motor was loaded). After these changes, the proposed ANN-based estimator gave good results as shown in figure 5.26.

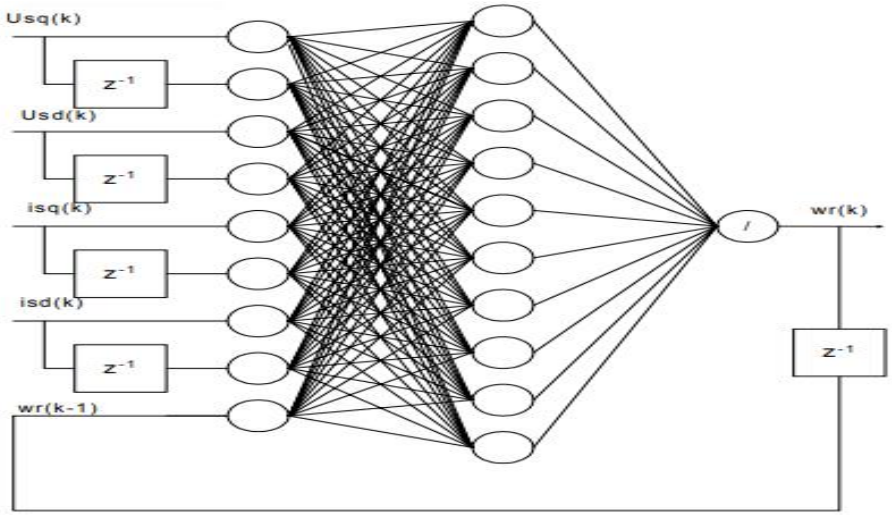


Figure 5.23. Proposed ANN-based speed estimator.

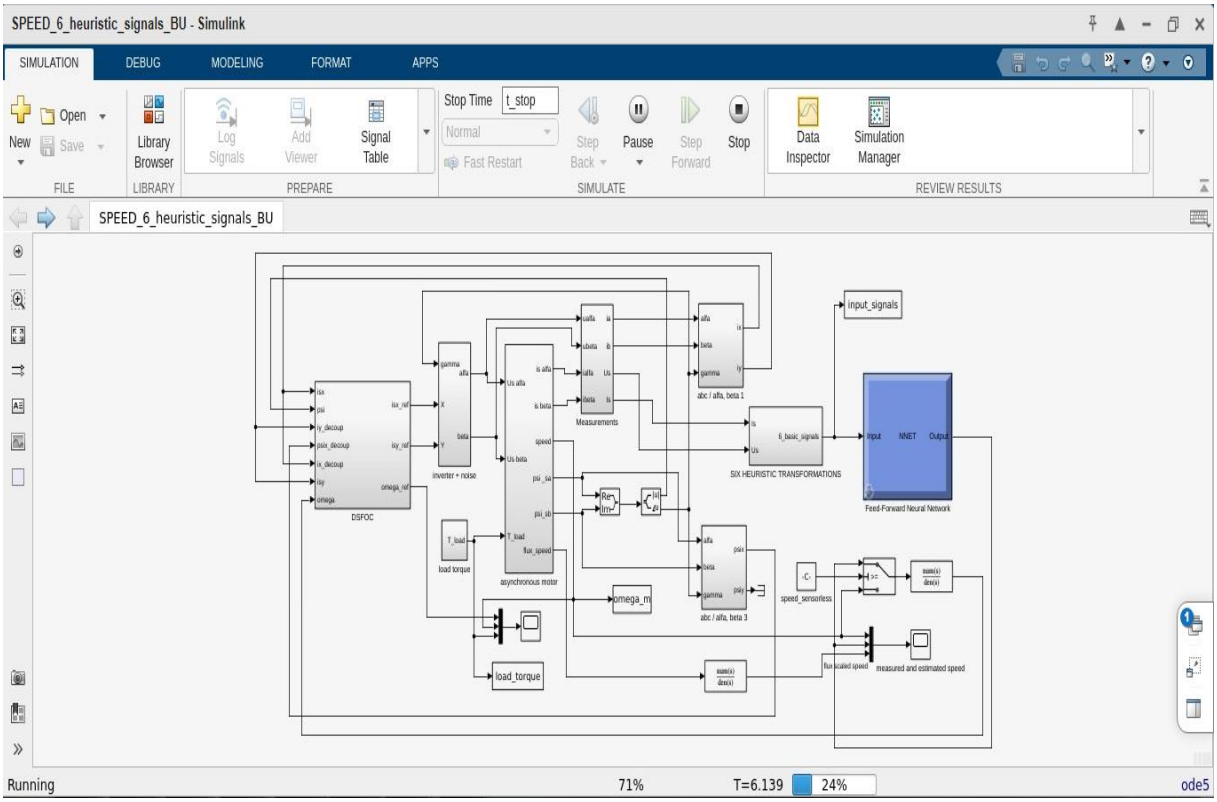


Figure 5.24. Main view.

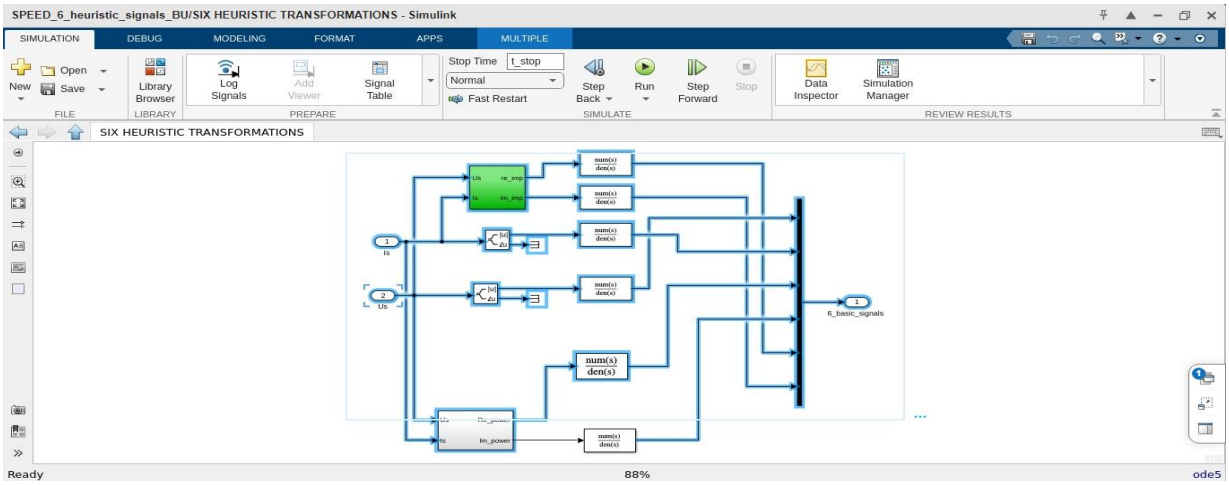


Figure 5.25. Heuristic signals.

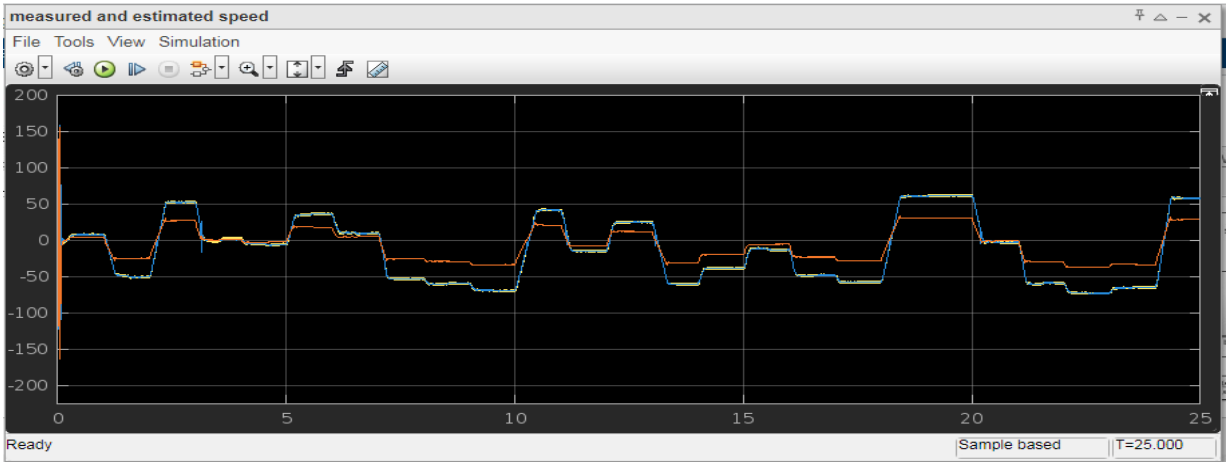


Figure 5.26. Measured and estimated speed.

5.24. References

[39] Marek Rosa, «*General Artificial Intelligence: Good AI and the Future of Human kind*», 2022.
[40] Bertrand LIAUDET, «*Cours de data mining 8 : modélisations réseaux de neurones et de kohonen epf – 4/ 5ème année - option ingénierie d'affaires et de projets – finance*».
[41] Gérald petitjean, «*introduction aux réseaux de neurones*», 2016.
[42] Patrice Wera, «*réseaux de neurones artificielle architecture et application*», 2009.
[43] Patrice Wera, «*Des réseaux de neurones artificiels pour l'identification et le contrôle de systèmes électriques*», 2009.
[44] Ricco Rakotomalala, «*réseaux de neurones artificielle perceptron simple et multi couche, Application des réseaux de neurones à l'apprentissage supervisé*»,
[45] https://www.emo.org.tr/ekler/97d404b6119214e_ek.pdf

Chapter 6: Genetic algorithms and applications in electrical engineering

6.1. Introduction

Genetic algorithms, initiated in the 1970s by John Holland, are optimization algorithms based on techniques derived from genetics and natural evolution mechanisms: crossing, mutation, selection.

It was in 1860 that Charles Darwin published his book titled "*The Origin of Species by Means of Natural Selection*", or the Struggle for Existence in Nature. In this book, Darwin rejects the existence of «fixed natural systems», already forever adapted to all external conditions, and presents his theory of the evolution of species: under the influence of external constraints, living beings gradually adapted to their natural environment through the process of reproduction.

Darwin proposed a theory that clarifies the evolution of species by putting forward four laws:

- The law of growth and reproduction.
- The law of heredity that almost implies the law of reproduction
- The law of variability, resulting from the conditions of existence.
- The law of multiplication of species which leads to the struggle for existence and which has consequence of natural selection.

Almost simultaneously, in 1866, Mendel ("the pitch monk") published the article retracing ten years of hybridization experiments in plants (recombination of genes) and sent it to scientific societies around the world. The reactions are mixed, if not non-existent.

The scientific world is not ready to recognize the quality of its results. It was not until 1900 that the publication of three new articles by Hugo de Vries, Carl Correns and Erich von Tschermark revealed results similar to those of Mendel, and made these first ones recognized.

It was then from the 20^{th} century that the genetic mutation was highlighted. The information processing problems are solved in fixed ways: during its design phase, the system receives all the characteristics necessary for the conditions of operations known at the time of its design, which prevents adaptation to unknown, variable or changing environmental conditions. Computer science researchers are therefore studying methods to allow systems to evolve spontaneously according to new conditions: this is the emergence of evolutionary programming (see Figure 6.1).

In the 1960s, John Holland studied evolutionary systems and, in 1975, he introduced the first formal model of genetic algorithms (the canonical genetic algorithm AGC) in his book Adaptation in Natural and Artificial Systems. He explained how to add intelligence to a computer program with crosses (exchanging genetic material) and mutation (source of genetic diversity). This model will be used as a basis for later research and will be more particularly taken up by Goldberg who will publish in 1989, a work of popularization of genetic algorithms, and added to the theory of genetic algorithms the following ideas:

- an individual is linked to an environment by his DNA code.
- a solution is linked to a problem by its quality index.

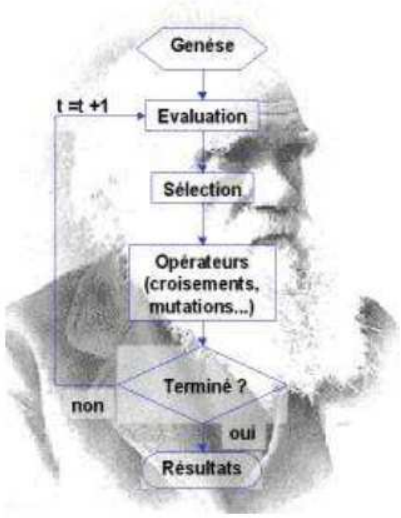


Figure 6.1. Flowchart of an evolutionary algorithm.

6.2.Evolutionary algorithms

Shown above is the flowchart of an evolutionary algorithm. It's about simulating the evolution of a population of various individuals (generally drawn randomly at the start) to which we apply different operators (recombinations, mutations, etc.) and which we submit to selection, at each generation. If selection operates from the adaptation function, then the population tends to improve [Bäck, 1996 and Bäck, 1997]. Such an algorithm does not require any knowledge of the problem: it can be represented by a black box comprising inputs (the variables) and outputs (the objective functions). The algorithm just manipulates the inputs, reads the outputs, manipulates the inputs again to improve the outputs, and so on. [Whitley, 1993] This is how breeders have proceeded for millennia: they have succeeded in modifying, according to their wishes, many animal species without knowledge of genetics or molecular biology.

Evolutionary algorithms constitute an original approach: it is not a question of finding an exact analytical solution, or a good numerical approximation, but of finding solutions that best satisfy various, often contradictory, criteria. If they do not make it possible to find with certainty the optimal solution of the search space, at least we can see that the solutions provided are generally better than those obtained by more traditional methods, for the same computation time.

They are part of the field of artificial life. Artificial life is the study of human-engineered systems that exhibit behaviors similar to natural living systems. It complements the traditional approach to biology, etymologically defined by the study of living beings, by trying to synthesize their behavior on an artificial medium. Modeling, in addition to observation, theory and experiment, is a new scientific tool that has been used since the advent of computers. This can contribute to theoretical biology by placing it in a larger context.

The objective is twofold: on the one hand, the modeling of these phenomena makes it possible to better understand them, and thus highlight the mechanisms which are at the origin

of life; on the other hand, these phenomena can be exploited freely and can therefore be diverse.

The field of artificial evolution has only really expanded in the last 15 years. However, the idea of simulating evolutionary phenomena on computers dates back to the 1950s. Concepts such as the representation of chromosomes by binary strings were already present.

The rise of artificial evolution since the 1980s can be explained by two competing phenomena. Firstly, this growth is mainly due to the exponential increase in the means of calculation made available to researchers, which allows them to display relevant and promising experimental results. The second point is the abandonment of the biologically plausible.

Three types of evolutionary algorithms have been developed in isolation and almost simultaneously, by different scientists: evolutionary programming (L. Fogel 1966), Evolutionary Strategies (J. Rechenberg 1973) and Genetic Algorithms (J. Holland 1975).).

In the 1990s, these three fields began to come out of their isolation and were grouped together under the Anglo-Saxon term of Evolutionary Computation.

We will only deal here with genetic algorithms based on Neo-Darwinism, that is to say the union of the theory of evolution and modern genetics. They rely on different techniques derived from the latter: crosses, mutation, selection...

6.3. Genetic algorithms for optimization pbs

A genetic algorithm searches for the extrema of a function (an optimization bp) defined on a data space. To use it, you must have the following five elements:

1) A population element coding principle. This step associates a data structure with each of the points in the state space. It is generally placed after a phase of mathematical modeling of the problem treated. The quality of data coding conditions the success of genetic algorithms. Binary encodings were widely used originally. Real codings are now widely used, especially in application domains for the optimization of problems with real variables.

2) A mechanism for generating the initial population. This mechanism must be able to produce a non-homogeneous population of individuals which will serve as a basis for future generations. The choice of the initial population is important because it can make the convergence towards the global optimum more or less rapid. In the case where nothing is known about the problem to be solved, it is essential that the initial population be distributed over the entire research domain.

3) A function to be optimized. This returns a value called the individual's fitness or evaluation function.

4) Operators allowing to diversify the population over the generations and to explore the state space. The crossover operator recomposes the genes of individuals existing in the population; the purpose of the mutation operator is to guarantee the exploration of the state space.

5) Sizing parameters: size of the population, total number of generations or stopping criterion, probabilities of application of crossover and mutation operators.

We now know what genetic algorithms are based on. It is now time to deepen the mechanisms of population selection and the notion of diversity that results from it. We will also try to define the operators mentioned in the flowchart of the evolutionary algorithm (see figure 1). Giving a picture that is both global and precise of the main tools of genetic algorithms; this will be our major objective during our second part.

6.4. How genetic algorithms work?

Genetic algorithms provide solutions to problems that do not have analytically or algorithmically computable solutions in a reasonable time.

According to this method, thousands of more or less good solutions (genotypes) are created randomly then are subjected to a process of evaluation of the relevance of the solution mimicking the evolution of species: the most "adapted", *i.e.* the solutions to the problem which are the most optimal survive more than those which are less so and the population evolves by successive generations by crossing the best solutions between them and mutating them, then re-running this process a number of times in an attempt to tend towards the optimal solution.

The mechanism of evolution and selection is independent of the problem to be solved: only three functions change:

- The function which takes care of representing the problem by coding each piece of information characterizing a possible solution according to a very particular coding, each piece of information then represents a gene and all the values that this characteristic can take represent the possible alleles for this gene, and concatenating all these genes to obtain a chromosome that represents a solution to it in its entirety
- The inverse function which, starting from a chromosome makes it possible to obtain a solution by decoding the genome.
- The function which evaluates the adaptation of a solution to a problem, its relevance.

This technique is of general application.

Indeed, when using genetic algorithms, no knowledge of how to solve the problem is required, it is only necessary to provide a function allowing to code a solution in the form of genes (and therefore to do the opposite work) as well as providing a function to assess the suitability of a solution to the given problem.

This therefore makes it a minimal and canonical model for any evolutionary system and for any problem that can be approached from this angle, under this paradigm.

This representation therefore allows us to study properties that are almost impossible to study in their natural environment, as well as to solve problems that have no computable solutions in reasonable time if they are approached under other paradigms, with quantifiable, easily measurable performance that can be compared to other resolution strategies.

6.5. Genetic algorithms areas

Genetic algorithms can be particularly useful in the following areas:

- Optimization: optimization of functions, planning, *etc*. ...
- Learning: classification, prediction, robotics, *etc*...
- Automatic programming: LISP programs, cellular automata, *etc*. ...
- Study of the living, of the real world: economic markets, social behavior, systems immune, *etc*.

6.6. Genetic algorithms paradigm

The main differences of genetic algorithms from other paradigms are:

- We use an information coding: we represent all the characteristics of a solution by a set of genes, that is to say a chromosome, under a certain coding (binary, real, Gray code, etc...), values that we concatenate to obtain a string of characters that is specific to a particular solution (there is a one-to-one relationship between the solution and its coded representation);
- We deal with a population of “individuals”, of solutions: this therefore introduces parallelism.
- The evaluation of the optimality of the system is not dependent with respect to the domain.
- We use probabilistic rules: there is no enumeration of the search space, we explores a certain part of it while being guided by a semi-chance: indeed operators as the evaluation function makes it possible to choose to be interested in a solution which seems represent a local optimum, we therefore make a deliberate choice, then to cross it with a another locally optimal solution, in general the solution obtained by crossing is better or on the same level as his parents, but this is not guaranteed, it depends on the hazards of chance, and this is all the more true for the mutation operator which does not apply only with a certain probability and in case it applies chooses randomly on which locus (loci) to introduce modifications.

6.7. Genetic algorithms form

A generic genetic algorithm has the following form:
1) Initialize the initial population P.
2) Evaluate P.
3) While (Not Convergence) do:
    a) P' = Selection of Parents in P
    b) P' = Apply Crossover Operator on P'
    c) P' = Apply Mutation Operator on P'
    d) P = Replace the Elders of P with their Descendants of P'
    e) Evaluate P
End While

The convergence criterion can be of various natures, for example:

- A minimum rate that we want to achieve of adaptation of the population to the problem,
- A certain calculation time not to be exceeded,
- A combination of these two points.

6.8. Genetic algorithms processor

6.8.1. Coding

Each parameter of a solution is assimilated to a gene, all the values it can take are the alleles of this gene, we must find a way to code each different allele in a unique way (establish a bijection between the "real" allele " and its coded representation).

A chromosome is a sequence of genes, we can for example choose to group together similar parameters in the same chromosome (single-stranded chromosome) and each gene will be identifiable by its position: its locus on the chromosome in question.

Each individual is represented by a set of chromosomes, and a population is a set of individuals.

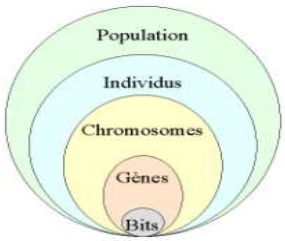


Figure 6. 2. Five levels of organization of a genetic algorithm.

There are three main types of encoding that can be used, and you can switch between them relatively easily:

- Binary coding: this is the most widely used. Each gene has the same binary alphabet {0, 1} A gene is then represented by a long integer (32 bits), the chromosomes which are sequences of genes are represented by gene tables and the individuals of our research space are represented by tables of chromosomes. This case can be generalized to any *n*-ary allelic alphabet allowing a more intuitive coding, for example for the traveling salesman problem one may prefer to use the allelic alphabet $\{c_1, c_2, c_3, ..., c_n\}$ where c_i represents the city of number *i*.

- Real coding: that can be useful in particular in the case where one seeks the maximu of a real function.

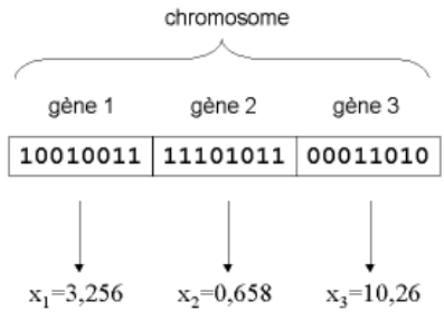


Figure 6. 3. Schematic illustration of the coding of real variables.

- Gray coding: in the case of binary coding, the "Hamming distance" is often used as a measure of the dissimilarity between two population elements; this measure counts the differences of bits of the same rank of these two sequences. And this is where binary coding begins to show its limits. Indeed, two neighboring elements in terms of Hamming distance do not necessarily encode two close elements in the search space. This drawback can be avoided by using "Gray coding": Gray coding is a coding which has the property that between an element *n* and an element *n* + 1, therefore neighboring in the search space, only one bit differs.

6.8.2. The selection operator

This operator is responsible for defining which individuals of *P* will be duplicated in the new population *P'* and will serve as parents (application of the crossover operator). Let *n* be the number of individuals of *P*, we must select *n*/2 of them (the crossover operator allows us to return to *n* individuals).

This operator is perhaps the most important since it allows individuals in a population to survive reproduce or die. As a general rule, an individual's probability of survival will be directly related to its relative effectiveness within the population.

There are basically four different types of selection methods:

- GoldBerg's "skewed lottery" method (roulette wheel),
- The "elitist" method,
- The selection by tournaments,
- Stochastic universal selection.

a) The biased lottery or roulette wheel

This method is the best known and the most used. With this method each individual has a chance of being selected proportional to their performance, so the more the individuals are adapted to the problem, the more likely they are to be selected. To use the image of the "fairground wheel", each individual is assigned a sector whose angle is proportional to his adaptation, his "fitness". We spin the wheel and when it stops spinning we select the individual corresponding to the sector designated by a sort of "cursor", a cursor that points to a particular sector of the wheel after it has stopped spinning.

This method, although widely used, has quite a few drawbacks:

• Indeed, it has a high variance. It is not impossible that out of n successive selections intended to designate the parents of the new generation **P'**, almost all, or even worse all of the ***n*** individuals selected are individuals with really poor fitness and therefore that practically no individual even no individual with strong fitness is among the parents of the new generation. This phenomenon is of course very harmful because it goes completely against the principle of genetic algorithms which wants the best individuals to be selected so as to converge towards the most optimal solution possible.

• Conversely, one can arrive at an overwhelming domination of a “locally superior” individual. This leads to a serious loss of diversity. Imagine for example that we have an individual with a very high fitness compared to the rest of the population, say ten times higher, it is not impossible that after a few successive generations we end up with a population containing only copies of this individual. The problem is that this individual had a very high fitness, but this fitness was all relative, it was very high but only in comparison to other individuals. We are therefore faced with a problem known as "premature convergence; evolution therefore begins to stagnate and we will never reach the optimum, we will remain stuck on a local optimum.

There are some techniques to try to limit this phenomenon, such as "scaling", which consists in making a change of scale so as to increase or forcibly decrease the fitness of one individual compared to another according to their difference in fitness. Nevertheless, it is advisable to opt instead for another method of selection.

b) The elitist method

This method consists of selecting the ***n*** individuals needed for the new generation **P'** by taking the ***n*** best individuals of the population **P** after having sorted it in decreasing order

according to the fitness of its individuals. It is needless to say that this method is even worse than that of the biased lottery in the sense that it will lead to premature convergence even more quickly and above all even more surely than the method of selection of the biased lottery; indeed, the selection pressure is too strong, the variance zero and the diversity non-existent, at least the little diversity that there might be will not result from selection but rather from crossing and mutations. Here too, another method of selection must be chosen.

c) Selection by tournaments

This method is the one with which the most satisfactory results are obtained. The principle of this method is as follows: a draw is made with discount of two individuals of **P**, and we make them "fight". Whoever has the highest fitness wins with a probability **p** between 0.5 and 1. This process is repeated ***n*** times so as to obtain the n individuals of **P'** who will serve as parents. The variance of this method is high and increasing or decreasing the value of **p** respectively decreases or increases the selection pressure.

d) Stochastic universal selection:

This method seems to be used very little and what is more has a low variance, therefore introduces little diversity, we will therefore not go into the details, we will be satisfied to explain its implementation: We take the image of a segment divided into as many sub-segments as there are individuals. The selected individuals are designated by a set of equidistant points.

6.8.3. The crossover or crossover operator

The crossover used by genetic algorithms is the computer transposition of the mechanism that allows, in nature, the production of chromosomes that partially inherit the characteristics of the parents. Its fundamental role is to allow the recombination of information present in the genetic heritage of the population.

This operator is applied after having applied the selection operator on the population P; we therefore end up with a population P' of n/2 individuals and we must double this number for our new generation to be complete.

We will therefore randomly create n/4 pairs and make them "reproduce". The chromosomes (sets of parameters) of the parents are then copied and recombined so as to form two offspring with characteristics from both parents. Let's detail what happens for each couple at the level of each of their chromosomes:

One, two, or even up to $lg - 1$ (where lg is the length of the chromosome) crossing points (loci) are drawn at random; each chromosome is therefore separated into "segments". Then each segment of parent 1 is exchanged with its "homolog" of parent 2 according to a crossing probability $\boldsymbol{p_c}$. From this process results 2 sons for each couple and our population ***P'*** therefore now contains ***n*** individuals.

It can be noted that the number of crossing points as well as the crossing probability $\boldsymbol{p_c}$ make it possible to introduce more or less diversity.

Indeed, the greater the number of crossing points and the higher the probability of crossing, the more there will be exchange of segments, therefore exchange of parameters, information, and the smaller the number of crossing points and the lower the probability of crossing, the less diversity the crossing will bring.

Below, a diagram illustrating a crossing in one point, another for a crossing in two points, and finally a diagram representing a crossing with $lg - 1$ points of crossings (it will be noted besides on this diagram that the exchange of a segment with its counterpart is not always done):

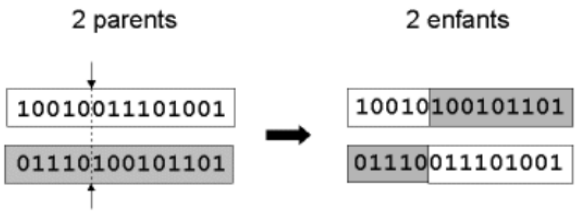


Figure 6.4. Crossing with a crossover point.

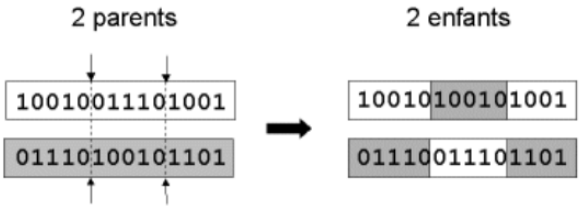


Figure 6.5. Crossover with 2 crossover points.

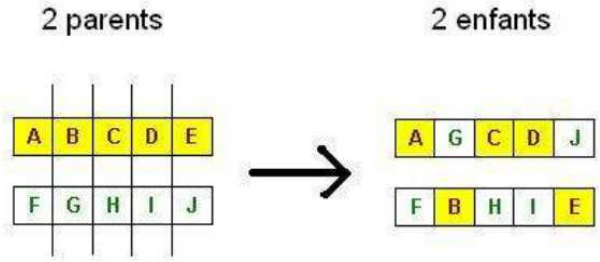


Figure 6.6. Uniform crossing.

One can also cite another method widely used in the case of problems modeled by binary coding, it is uniform crossing. The implementation of this process is very simple; it consists of randomly defining a "mask", that is to say a string of bits of the same length as the chromosomes of the parents to which it will be applied. This mask is intended to know, for each locus, from which parent the first child will have to inherit the gene found there; if facing a locus the mask presents a 0, the child will inherit the gene found there from parent n° 1, and if it presents a 1 it will inherit it from parent n ° 2. The creation of the child n° 2 is done from symmetrically: if for a given gene the mask indicates that son no. 1 should receive it from parent no. 1, then son no. 2 will receive it from parent no. 2, and if son no.1 from parent n.2 then child 2 will receive it from parent n.1.

The crossover operator favors the exploration of the search space. Indeed, consider two genes A and B that can be improved by mutation. It is unlikely that the two improved genes A' and B' appear by mutation in the same individual. But if one parent carries the A' mutant gene and the other the B' mutant gene, the crossover operator will allow to quickly combine A' and B' and thus to create a new individual having this combination, combination thanks to which it is possible that it is even more adapted than its parents. The crossover operator therefore ensures the mixing of genetic material and the accumulation of favorable mutations. In more concrete terms, this operator allows you to create new combinations of component parameters.

In spite of everything, it is possible that the joint action of selection and crossing does not make it possible to converge towards the optimal solution of the problem. Indeed, imagine that we have a population of individuals possessing a single chromosome. Let us consider a particular gene of this chromosome, it will be called G, gene having 2 possible alleles: 0 and 1; if no individual in the initial population possesses the 1 allele for this gene, no possible cross will introduce this allele for our G gene. If the optimal solution to the problem is such that our G gene possesses the 1 allele, it will be impossible for us to reach this optimal solution simply by selection and crossing. It is to remedy this problem, among other things, that the mutation operator is used.

6.8.4. The mutation operator

This operator consists of changing the allelic value of a gene with a very low probability pm, generally between 0.01 and 0.001. We can also take $p_m = 1 / lg$ where lg is the length of the string of bits encoding our chromosome. A mutation simply consists of the inversion of a bit (or several bits, but given the probability of mutation it is extremely rare) found in a very particular locus and also determined randomly; we can therefore summarize the mutation as follows: We use a function supposed to return true with a probability p_m .

For each locus do
    Call the function
    If this function returns true **then**
        the bit at this location is inverted
    End if
End For

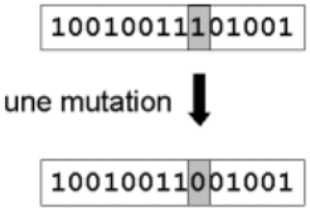


Figure 6.7. The mutation.

The mutation operator thus modifies in a completely random way the characteristics of a solution, which makes it possible to introduce and maintain diversity within our population of solutions. This operator plays the role of a "disturbing element", it introduces "noise" within the population. This operator has 4 major advantages:

- It guarantees the diversity of the population, which is essential for genetic algorithms.
- It avoids a phenomenon known as *genetic drift*. We speak of genetic drift when certain genes favored by chance spread to the detriment of others and are thus present in the same place on all the chromosomes. The fact that the mutation operator can randomly cause changes at any locus avoids the installation of this unfavorable situation.
- It makes it possible to limit the risks of premature convergence caused, for example, by an elitist selection method imposing excessive selective pressure on the population. Indeed, in the case of premature convergence, we end up with a population in which all the individuals are identical but are only local optima. All individuals being

identical, the crossing will not change the situation. Indeed, the exchange of information by crossover between strictly identical individuals is of course completely without consequences; no matter how we choose the crossing method we want, we will always find ourselves exchanging portions of identical chromosomes and the population will not evolve. Evolution being blocked, we will never wait for the global optimum.

The mutation causing random bit inversions makes it possible to reintroduce differences between individuals and therefore to extricate ourselves from this situation.

It is nevertheless useful to keep in mind that this is not a "miracle" solution and that it is of course smarter not to use selection methods known to cause this type of problem.

- The mutation makes it possible to reach the property of ergodicity. Ergodicity is a property ensuring that every point in the search space can be reached. Indeed, a mutation being able to occur randomly at the level of any locus, we have the mathematical certainty that any permutation of our string of bits can appear within the population and therefore that any point of the search space can be reached. Thanks to this property, we are therefore sure of being able to reach the global optimum. It will be noted that the mutation therefore solves the problem exposed at the end of the Section on crossbreeding.

6.8.5. The replacement operator

This operator is the simplest, its work consists in reintroducing the descendants obtained by successive application of the selection, crossover and mutation operators (the population **P'**) into the population of their parents (the population **P**). In doing so, they will replace a certain proportion of these, a proportion that can of course be chosen. The ratio between the number of new individuals going to be introduced into the population **P** and the number of individuals in this population is known as the generation gap.

There are basically 2 different replacement methods:

- Stationary replacement: in this case, the children automatically replace the parents without taking into account their respective performances, and the number of individuals in the population does not vary throughout the simulated evolutionary cycle, which therefore implies to initialize the initial population with a sufficient number of individuals. This method can be implemented in 2 different ways:

1) The first is content to replace the entire population **P** by the population **P'**, this method is known as generational replacement and we therefore have a generation gap which is equal to 1.

2) The second method consists in choosing a certain proportion of individuals from **P'** who will replace their parents in **P** (proportion equal to 100% in the case of generational replacement. This type of replacement generates a population with a large variation and is therefore promotes genetic drift, which is all the more apparent when the population is small. Moreover, in many cases, given that even a child with poor performance necessarily replaces a parent, the best solution is not reached but we are only getting close.

6.8.6. Elite replacement

In this case, we keep at least the individual with the best performance from one generation to the next. In general, it can be assumed that a new individual (child) enters the population only if it fulfills the criterion of being more efficient than the least efficient of the individuals of the previous population. So the children of a generation will not necessarily replace their parents as in stationary replacement and by the same token the size of the population is not fixed over time. This type of strategy improves the performance of evolutionary algorithms in some cases. But also has a disadvantage by increasing the premature convergence rate.

However, finer implementations proceed differently. In this case, the replacement rate is not 100%, the size of the population therefore increases during successive generations, we say that there is overcrowding. We must therefore find a way to select the parents who will be deleted, who will die. De Jong proposed the following solution: imagine that we want to replace 30% of the parents, let n_p be the number of parents corresponding to this percentage, we will replace the n_p relatives closest to their descendants of **P'**. This method therefore allows firstly to maintain diversity and secondly to improve the overall fitness of the population.

6.9. Genetic algorithms steps programming

To properly apply GA, it is necessary to clearly identify the different steps prior to programming.

- Process

To use the GA, you must have six elements:

- A population element coding principle. This step associates a data structure with each of the points of the considered space. It takes place after the phase of mathematical modeling of the problem addressed. Binary encodings were the first to be used. Currently, we use more and more real codings in particular for the optimization of problems with real variables.
- A mechanism capable of generating a non-homogeneous initial population which will serve as a basis for future generations. This choice determines the speed of convergence towards the optimum. In the case where nothing is known about the problem to be solved, it is essential that the initial population be distributed over the entire research domain.
- A function to be optimized. This returns a positive real called the evaluation function (fitness).
- A mechanism for selecting individuals who are candidates for evolution. The casino's "roulette" is generally used to select individuals at random. Each individual occupies a sector on the roulette wheel proportional to his evaluation function: this causes chance to be biased towards the fairest (suitable) elements who are more likely to be selected than the others.
- Operators allowing to diversify the population over the generations and to explore the state space. The crossover operator recomposes the genes of existing individuals. The purpose of the mutation operator is to guarantee the exploration of space.
- Sizing parameters: population size, stopping criterion, probability of application of genetic operators.

This last point raises the question of quantification. In fact, there is no universal setting. However, some values widely used to concretely solve problems deserve to be retained:

- Population size: between 30 and 50 individuals
- Crossing rate: between 70% and 95%
- Mutation rate: 0.5% to 1%.

Framework of the algorithm

```
        p designates the population
        t designates the population
beginning
        t=0
        initialize p(t)
        evaluate p(t)
        while (determination condition is false)
        beginning
                t=t+1
        select p(t) from p(t-1)
        recombine p(t)
        evaluate p(t)
    End
End
```

Coding

In what follows *x****y* will mean *x* to the *y* power; *Pc* crossover probability and *Pm* mutation probability. To find the maximum of a function *f*(*x*), in the interval [*a*, *b*], with a precision of n significant digits, we proceed as follows:
The interval [*a*, *b*] will be subdivided into (*b* - *a*)(10***n*) small intervals which will each represent a chromosome.

Each chromosome will be encoded using *k* bits, with *k* verifying the following inequalities:

with *a* = 0000….00 and *b* = 1111….11.
The code of each chromosome corresponds to its binary value *x'*.
The actual number corresponding to the chromosome is determined by: *x* = a + *x*'(2/((2***k*))

A simple numerical example is given at the end of the chapter.

- Calculations made for each generation:
- Compute the evaluation function *eval*(*vj*) for each chromosome *vj*.
- Calculate the total rating, *F*, of the population (sum of the ratings of each chromosome).
- Calculate the selection probability *pj* for each chromosome *vj*: *pj* = *eval*(*vj*)/F.
- Calculate the cumulative probability *qj* for each *vj* (*qj* = p_1+p_2+….+pj).
- To select, we spin the size_population roulette times as follows: each time, we randomly generate a number *r* in [0, 1],

  If *r* < *q*1 , select v_1,
  Otherwise,
  select *vj* , with 2 =< *j* =< population_size such that *q*(*j*-1) < *r* <*qj*.
- For each chromosome of the new population, we generate, at random, *r* in [0,1], If *r* < *Pc*, select the chromosome for the crossing.
- Cross the chromosomes thus obtained two by two. If the number of chromosomes obtained is odd, we can decide to prune one or take another.
- Generate, at random, *r* in [0, 1] (as many times as there are bits in the whole population). If *r* =< Pm, mute the bit.

6.10. Example 1: calculate the maximum of a real function implementing all the operators

The following examples are chosen very simple to facilitate the understanding of the implementation of the genetic approach.

Maximum of a real function of a real variable

Find the maximum of $f(x) = -(x**2) + 4x$ in the interval [1 , 3] with a precision of 1/10. Analytically, we quickly see that $f'(x) = -2x + 4$, that $f''(x) = -2 < 0$ and that the maximum corresponds to $x = 2$ and $f(x) = 4$.

Let's find the length of the chromosome (number of bits in the string). The length of the interval is 3 – 1 = 2.

Each unit must be subdivided into 10 (precision desired). So the interval is subdivided into 2 * 10 = 20 small intervals. The number of bits required to represent all the reals considered in the interval is k such that $2**(k – 1) =< 20 =< 2**k$. $k = 5$.

To model the problem, let's agree on the following: A population of 4 individuals (chromosomes), each individual encoded on 5 bits (genes). (probability of crossing) $Pc = 0.75$ and (probability of mutation) $Pm = 0.01$.

Let's randomly build the initial generation:

V1	01100	x1' = 12	x1 =1+ x1'(2/ ((2**8)-1) = 1,1	eval(V1) = 3,1
V2	00011	x2' = 3	x2 = 1,0	eval(V2) = 3,0
V3	11011	x3' = 27	x3 = 1,2	eval(V3) = 3,3
V4	10100	x4' = 20	x4 = 1,16	eval(V4) = 3,3

The sum of the ratings is 12.7; the highest rating 3.3 and the average value 3.2. Let's train the first generation.

Selection

Calculating the selection probabilities, we get:

$P1 = 0.2444$	$Q1 = 0.244$
$P2 = 0.2333$	$Q2 = 0.437$
$P3 = 0.259$	$Q3 = 0.736$
$P4 = 0.262$	$Q4 = 1$

We spin the wheel 4 times to generate numbers r in [0, 1], we get:
0.512; 0.710; 0.216; 0.773

$r = 0,51$	$Q2 < 0,512 < Q3$	$V3$ est sélectionné
$r = 0,70$	$Q2 < 0,710 < Q3$	$V3$ …….
$r = 0,282$	$0,216 < Q1$	$V1$ …….
$R = 0,733$	$Q3 < 0,773 < Q4$	$V4$ ……..

La première génération devient :

*V*1’ 11011

*V*2’ 11011

*V*3’ 01100

*V*4’ 10100

Crossing

Assume that randomly, we proceed to the crossing from the second position.
We spin the spinner to generate r numbers in [0, 1].
If $r < 0.75$, the chromosome is selected for crossing over.
We get: 0.82 0.52 0.17 0.35
So *V*2, *V*3, *V*4 are selected. As the number is odd, we drop the last. This gives for the crossing:

V2’	11|011	→ 11100
V3’	01|100	→ 01011

After crossing we get:

***V*1’’** 11011
***V* 2’’** 11100
***V*3’’** 01011
***V*4’’** 10100

Mutation

There are 4x5 = 20 bits.
We spin the wheel 20 times to generate *r* in [0, 1]
If r < 0.01, the bit of this row is muted. Only, at the 18th round, we obtain $r = 0.008$, we mute, then the 18th bit which corresponds to the 3rd bit of the 4th vector.
Finally, the first generation becomes:

***V*1** 11011
***V*2** 11100
***V*3** 01011
***V*4** 01000

Evaluating the first generation, we get:

$x1 = 2.6$ $eval(V1) = 3.6$
$x2 = 2,5$ $eval(V2) = 3,7$
$x3 = 1,7$ $eval(V3) = 3,8$
$x4 = 0,5$ $eval(V4) = 3,7$

Total rating = 14.8 highest value = 3.8 mean value = 3.7
We have just completed an iteration of the “While” loop.

Let's form the second generation By now taking the first generation as the initial population and redoing the loop "as long as" (we apply the selection, crossover and mutation operators) we obtain the second generation:

*V*1 01100	*x*1 = 1.7	*eval*(*V*1) = 3.9
*V*2 11011	*x*2 = 2.6	*eval*(*V*2) = 3.6
*V*3 01100	*x*3 = 1.7	*eval*(*V*3) = 3.9
*V*4 01011	*x*4 = 1.6	*eval*(*V*4) = 3.8

Sum of ratings = 15.2 Highest value = 3.9 (comes 2 times) Average = 3.8
By forming the third generation, starting from the second, we obtain: Sum of the evaluations = 15.5 The greatest value = 3.9 (returns three times) The average = 3.8.

We notice a certain stagnation around $x = 1.7$ and $x = 2.3$ which both give $f(x) = 3.9$.

6.11. Example 2 find the set of parameters (w_1: w_6???) that maps the following inputs to the outputs of *y* function given by :

$$y=w_{1*}\, x_{1+}\, w_{2*}\, x_{2+}\, w_{3*}\, x_{3+}\, w_{4*}\, x_{4+}\, w_{5*}\, x_{5+}\, w_{6*}\, x_{6.}$$

Data

x_1	x_2	x_3	x_4	x_5	x_6	y
4	-2	7	5	11	1	44.1

The goal is to find the set of parameters (w_1: w_6???) that maps the following inputs to its outputs.

$y'=4\, x_1\text{-}2\, x_{2+}\, 7\, x_{3+}\, 5\, x_{4+}\, 11\, x_{5+}\, 1\, x_{6.}$

Solution 1:

$y'=4\, w_1\text{-}2\, w_{2+}\, 7\, w_{3+}\, 5\, w_{4+}\, 11\, w_{5+}\, 1\, w_6$ = 110.3

Error = $|y\text{-}y'| = |44.1\text{-}110.3| = 66.2$

w_1	w_2	w_3	w_4	w_5	w_6
2.4	0.7	8	-2	5	1.1

Solution 2:

$y'=4\, w_1\text{-}2\, w_{2+}\, 7\, w_{3+}\, 5\, w_{4+}\, 11\, w_{5+}\, 1\, w_6$ = 101.1

Error = $|y\text{-}y'| = |44.1\text{-}101.1| = 56$

w_1	w_2	w_3	w_4	w_5	w_6
-0.4	2.7	5	-1	7	0.1

Solution 3:

$y'=4\, w_1\text{-}2\, w_{2+}\, 7\, w_{3+}\, 5\, w_{4+}\, 11\, w_{5+}\, 1\, w_6$ = 13.9

Error = $|y\text{-}y'| = |44.1\text{-}13.9| = 30.2$

w_1	w_2	w_3	w_4	w_5	w_6
-1	2	2	-3	2	0.9

- Use optimization technique such GA
- GA is based on natural evolution of organism (organism → Cells → Chromosome → Génes)
- L'évolution des gènes conduit à l'être humain
- GA : Induviduals → Chromosome → Génes

PB A

SOLUTION

- Gene is anything that is able to enhance the results when changed

Géne 0	Géne 1	Géne 2	Géne 3	Géne 4	Géne 5
w_1	w_2	w_3	w_4	w_5	w_6

- Initial population of solutions (generation 0)

Individuals →

2.4	**0.7**	**8**	**-2**	**5**	**1.1**
-0.4	2.7	5	-1	7	0.1
-1	2	2	-3	2	0.9
4	7	12	6.1	1.4	-4
3.1	4	0	2.4	4.8	0
-2	3	-7	6	3	3

➢ Fitness function $F(c) = 1/\text{error} = 1/|y-y'|$

y'	$F(c)$
110.3	0.015
100.1	0.018
13.9	0.033
127.9	0.012
69.2	0.0389 ≈ 0.04
3	0.024

Depending on the large value of $F(c)$, we choose the individuals

-1	2	2	-3	2	0.9
3.1	4	0	2.4	4.8	0
-2	3	-7	6	3	3

CROSSOVER

-2	3	-7	-3	2	0.9

3.1	4	0	6	3	3

-1	2	2	2.4	4.8	0

MUTATION

-2	3	-7	-3	1	0.9

3.1	4	0	6	1.5	3

-1	2	2	2.4	2.4	0

➢ New population (generation 1)

Old individuals

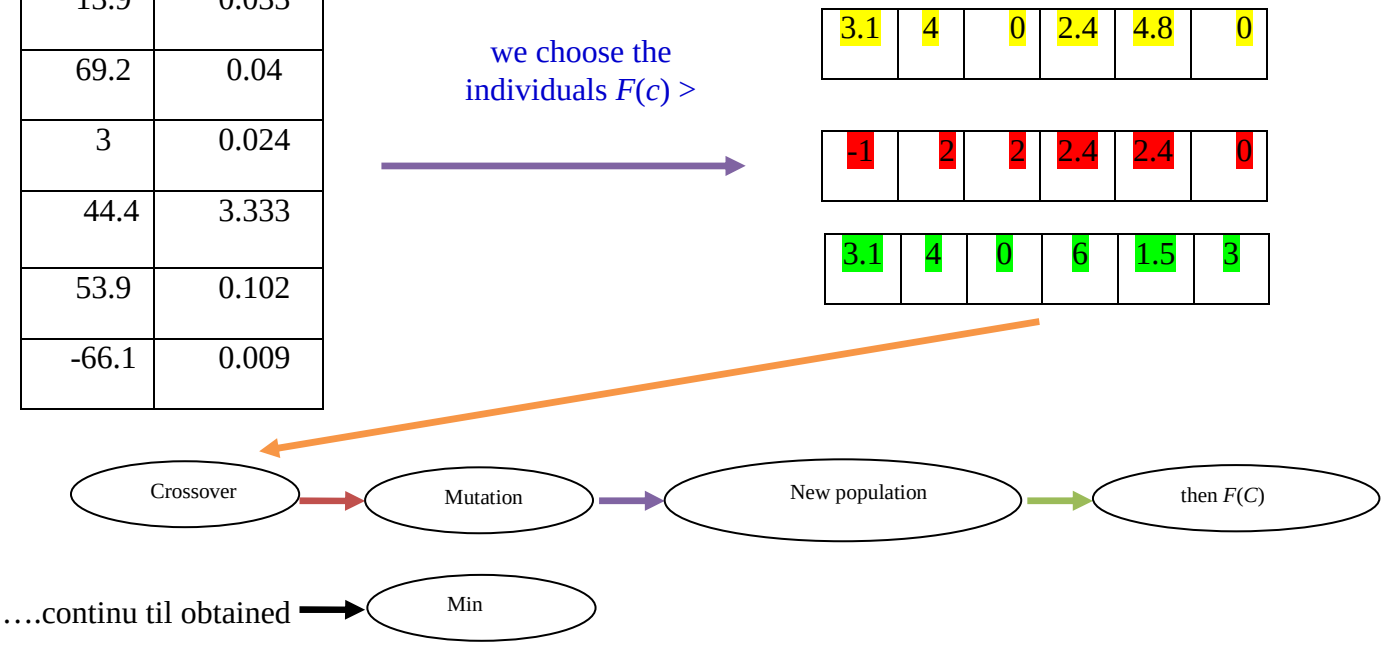
-1	2	2	-3	2	0.9
3.1	4	0	2.4	4.8	0
-2	3	-7	6	3	3

New individuals

-1	2	2	2.4	2.4	0
3.1	4	0	6	1.5	3
-2	3	-7	-3	1	0.9

Remarque

y'	*F* (*c*)
13.9	0.033
69.2	0.04
3	0.024
44.4	3.333
53.9	0.102
-66.1	0.009



6.12. Example 3: Adjusting the parameters of a PID regulator by optimization method

If we consider that "*p*" is a vector which contains the five parameters of the fractional corrector or the three parameters of the classic PID, PI or P, and in order to set and adjust the gains of these correctors, we can call on optimization methods to achieve optimal results. These parameters will therefore be represented by a set of particles or chromosomes, with respect to all the steps of the genetic algorithm described previously. By following the steps of the algorithm, and for a certain number of iterations, we have a strong probability of finding a reliable solution to our optimization problem. This issue being, in our case, defined by a closed loop regulation error criterion. Performance criteria (cost function) are then defined based on the error.

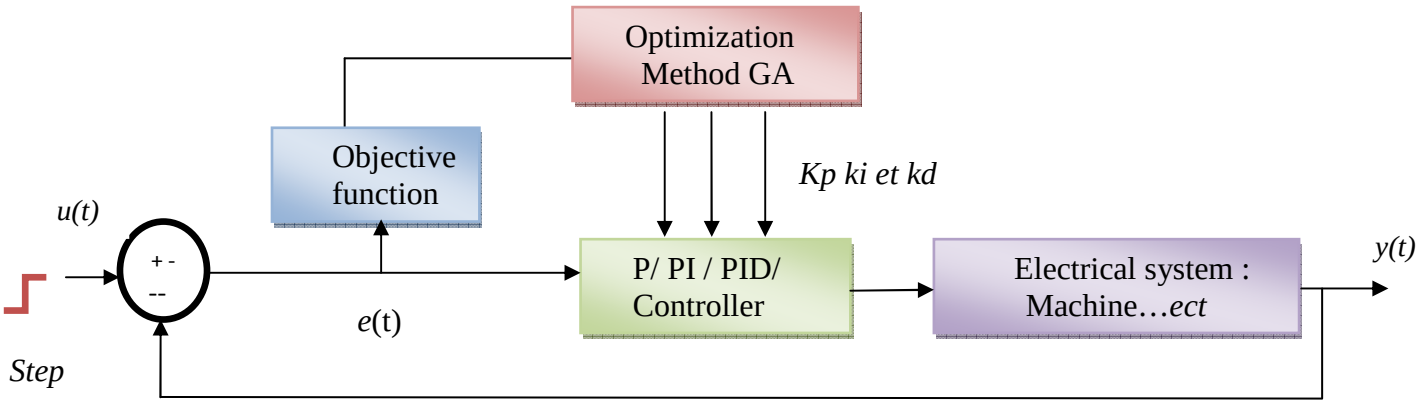


Figure 6.8. Optimization approach for adjusting regulator parameters.

Performance criterion: To have good dynamic precision of a system with one step input, it is necessary that the transient regime is characterized by low overshoot and a response time optimal. For this, the parameters of a regulator are chosen so as to minimize the error dynamic:

$$e(t) = y_{ref}(t) - y_s(t) \tag{6.1}$$

We will use two criteria in this dissertation which are as follows:

- **The integral of the square of the error (ISE)**

$$ISE = \int_0^{+\infty} e^2(t)dt \tag{6.2}$$

It represents the integral of the squared error, it is given by:

- **The integral of the absolute error (IAE)**

$$IAE = \int_0^{+\infty} |e(t)|dt \tag{6.3}$$

It represents the integral of the absolute value of the error.

The IAE criterion is often used for the numerical simulation of systems; however it is inapplicable for analytical work due to the fact that the absolute value of an error function is not always expressible in analytical form. This problem is overcome by the ISE criterion.

6.13. References

[46] G.Menier, «*Introduction aux algorithmes génétiques*», université de Bretagne de sud.
[47] Christelle Reynés, «*Etude des Algorithmes génétiques et application aux données de protéomique*», 2007.
[48] G. Heyen, C. Gerkens, «*Application d'algorithmes génétiques à la synthèse de systèmes de mesures redondants*», 2002.
[49] Thomas Vallée et Murat Yıldızoglu, « *Présentation des algorithmes génétiques et de leurs applications en économie*», 2004.
[50] Souquet amédée rad et francois-gérard, «*Algorithmes génétiques*».
[51] Nicolas Barnier — Pascal Brisset, «*Optimisation par algorithme génétique sous contraintes*», *École Nationale de l'Aviation Civile*, Toulouse, 2004.

Chapter 7: Modern optimization algorithms

Before addressing the topic of modern optimization algorithms, and because our students of ENPC school have not advanced to it previously in their third year engineering study, it is necessary to establish the rules and principles of optimization and give its mathematical model, and then we move to modern methods of optimization.

7.1. Introduction

Among all the subjects covered in this course, the optimization of functions, generally of several variables, is undoubtedly the one that appears most frequently in physical or economic modeling (maximizing profit, customer satisfaction, and productivity or minimize costs, risk, *etc*.).

An optimum or extremum is either a maximum or a minimum, that is to say the highest value or the weaker than the function takes on its definition set or any subset of its definition set.

The topic of multivariate optimization requires a significant conceptual leap. Moreover, taking into account the constraints imposed on variables must be fundamentally reviewed.

Let's take the case of maximizing the function $x \to f(x)$ whose unique variable is subject to a non-negativity constraint ($x \geq 0$), which reflects for example the fact that x is a quantity, a density, a price or any other quantity view of meaning for a negative value. The optimization is then carried out using the usual procedure, the constraint having the effect of restricting the field of study to R^+ and requiring a particular examination for the single point which constitutes the "edge" ($x = 0$). Let us immerse the same problem in a bivariate framework: maximize $f(x, y)$ under the double constraint $x \geq 0$ and $y \geq 0$.

Now, the admissible domain is given by $(R^+)^2$ whose "edge", $\{(x, 0) \mid x \in R^+ \} \cup \{(0, y) \mid y \in R^+\}$.

Obviously includes a infinite points. An individual study of the points in this set becomes technically difficult, so that it appears essential to have optimization methods which immediately integrate the presence of constraints which reveal "edges".

↳ This approach (search for linked extrema), specific to the functions of several variables, will be discussed in this chapter after the presentation of the principles of so-called free optimization, which aims to determine the extrema in open domain, therefore "without edges".

7.2. Definition

Let f be a function of $D \subset R^n$ in R. We say that

➔ f is bounded in D if there exists a real number $M \geq 0$ such that $\forall x \in D, |f(x)| \leq M$;

➔ f admits a global (or absolute) maximum (resp. minimum) in $x_0 \in D$ if $\forall x \in D, f(x) \leq f(x_0)$, (resp. $f(x) \geq f(x_0)$);

➔ f admits a maximum (resp. minimum) local (or relative) at $x_0 \in D$ if there exists a ball of non-zero radius $B(x_0, r)$ such that $\forall x \in D \cap B(x_0, r), f(x) \leq f(x_0)$, (resp. $f(x) \geq f(x_0)$)0.

Given an optimization problem, two questions arise: are there solutions? And how to calculate possible solutions? Optimization theory therefore faces two classic problems in mathematics: that of existence and that of research methods.

A priori, an optimization problem may admit no solution or admit at least one. In general, no mathematical argument guarantees the existence of solution(s). However, we have a sufficient condition thanks to the WEIERSTRASS theorem, which only concerns continuous functions on a compact of R^n, *i.e.* a subset of R^n which is closed and bounded. We recall that a

set is closed if it contains its boundary and that a set is bounded if it is contained in a ball of radius $r \in R$.

For example,

- The square $[-1;1]^2$ is a compact of R^2;
- The set $\{(x, y) \in R^2 \mid x \geq 0, y \geq 0\}$ is an unbounded closed set;
- The disc $\{(x, y) \in R^2 \mid x^2 + y^2 < 1\}$ is a non-closed bounded set;
- The half-plane $\{(x, y) \in R^2 \mid x^2 + y^2 < 1\}$ is an unbounded unclosed set.

7.3. WEIERSTRASS theorem

Let D be a compact of Rn and let f: $D \to R^n$ be a continuous function, then f admits a global maximum and minimum reached at least once, in other words there exist $x_m \in D$ and $x_M \in D$ such that $f(x_m) \leq f(x) \leq f(x_M)$, $\forall(x) \in D$.

Note that the extrema can belong either to the open D° $D \backslash \partial$D or to the boundary ∂D.

The following section presents the determination of extrema in an open domain. The terminology of "free extrema" results from the absence of conditions introducing edges in the domain which are also called "constraints".

To find the extrema, the first idea that comes to mind is to calculate the values that the function f takes for all the values taken by the arguments and then to identify the largest and smallest values taken by the images. This is obviously not the right option if the arguments can take infinite values. If the function to be optimized is a numerical function of a real variable, we can always construct the table of variations or plot the function in the plane as we learned in high school. It is tedious to the extent that we are only interested in the optima and them alone. If the function to be optimized has two real variables, we can, if necessary, ask appropriate software to draw its representative surface or level curves and conclude based on the graphs. This can sometimes be useful, but it is often frustrating to the extent that we do not know a priori where the optima are, which means that we are hard-pressed to give all the details.

In any case, as soon as the function has more than three variables, graphical methods are of no help and you must have a solid theory to determine the optima. The purpose of this chapter is to illustrate certain mathematical results which allow us to answer some of these questions.

It turns out that the mathematical theory of optimization is very complete for functions at least twice continuously differentiable over an open. Solve a maximization (resp. minimization) problem using theorems based on differentiability amounts to searching for local maxima (resp. minima). Indeed, differentiability is a local property in the sense that it is legal to replace a function by a polynomial function in the neighborhood of a point.

The approximation resulting from the differentiation becomes less and less good, or even frankly erroneous, as we move away from the point where it was calculated. Consequently, the most elaborate part of optimization theory gives the properties local aspects of a solution and is unable to characterize the global optima unless the functions have particular properties such as convexity or concavity.

7.4. FERMAT's theorem:

Necessary condition of the first order Let D be an open subset of R^n, x_0 a point contained in D and f: $D \to R$ a function of class C^1 at this point.

If f presents a local extremum then $\nabla f(x_0) = 0$.

Definition Stationary or critical point

Like the functions of a real variable, a point x_0 verifying $\nabla f(x_0)=0$ is called a stationary point or critical point of *f*.

7.5. Nature of a critical point: direct study

The first order condition means geometrically that the plane tangent to the surface of equation $z = f(x, y)$ at the point (x_0, y_0) with coordinates $(x_0, y_0, f(x_0, y_0))$ is horizontal. After having determined a stationary point x_0, we can then determine its nature by studying the sign of the difference $d(h)= f(x_0 +h)- f(x_0)$.

If this difference has a constant sign for *h* close to 0, it is a local extremum (a maximum if $d < 0$, a minimum if $d > 0$). Otherwise, it is a pass point (or saddle point). Better, if the sign is constant for any *h*, then the extremum is global.

Example

We look for the extrema of the function $f(x, y) = x^2 + y^2$ in the open disk centered at (0,0) of radius 1, represented by $\{D = (x, y) \in R^2 \mid x^2 + y^2 < 1\}$.

$\partial f_x (x, y) = 0$ and $\partial f_y (x, y) = 0$. The only unique critical point is (0,0).

$d(h)= f(x_0 +h)- f(x_0)$.

$= f(0 +h)- f(0)$.

$= h^2+y^2-y^2$

$= h^2 > 0$ a global minimum at (0,0).

REMARK

The first order condition only concerns differentiable functions defined on an open set *D*. However, it can naturally happen that a function is not differentiable at a point and nevertheless admits an extremum. Or again, the function can be differentiable on a compact, but the (necessary) condition of the first order applies to the interior points and not to the points of the boundary where *f* can nevertheless pass through an optimum. We then have the following "recipe" for calculating the extrema of a continuous function *f* in a compact set *D*.

"Recipe" for calculating the extrema of a continuous function f in a compact set *D*

- We calculate the value of *f* at the stationary points of *f* in the open $D^\circ = D \setminus \partial D$;
- We calculate the value of *f* at the stationary points of *f* on the edge ∂D (*i.e.* we must study the restriction of *f* to the curve which defines the edge ∂D);
- We calculate the value of *f* at the points of non-differentiability (if any). The largest value gives the global maximum, the smallest the global minimum.

7.6. Example: We want to find the global extrema of the function *f* defined by $f(x, y) = (x - y)^2$ on the closed square $D = [0;1]^2$.

- We calculate the value of *f* at the stationary points of *f* in the open $D^\circ =]0;1[^2$:

$$f(x, y) = (x-y)^2$$

$$, \partial f_x (x, y) = 2.(x-y),$$

$$\partial f_y (x, y) = -2(x- y),$$

and $\nabla f = (0,0)$ if and only if $y = x$ and we have $f(x,x) = 0$.

- We calculate the value of *f* at the stationary points of *f* on the edge ∂D:

- End of equation $y = 0$: let g :]0,1[$\rightarrow R$ such that $g(x) \equiv f(x,0) = x^2$ then $g'(x) \neq 0$ for all $x \in$]0.1[.

- End of equation $y = 1$: let g :]0,1[$\rightarrow R$ such that $g(x) \equiv f(x,1) = (x\text{-}1)^2$ then $g' \neq (x) = 0$ for all $x \in$]0.1[.

- End of equation $x = 0$: let g :]0,1[$\rightarrow R$ such that $g(y) \equiv f(0, y) = (-y)^2$ then $g'(y) \neq 0$ for all $y \in$]0,1[.

- End of equation $x = 1$: let g :]0,1[$\rightarrow R$ such that $g(y) \equiv f(1, y) = (1\text{-}y)^2$ then g'$\neq (y)$ for all $y \in$]0,1[.

Therefore there are no stationary points on the edges of the square.
We calculate the value of f at the points of non-differentiability:

- $f(0,0) = 0$
- $f(1.0) = 1$
- $f(0.1) = 1$
- $f(1,1) = 0$

In summary, the global extrema candidates are the points (0, 0), (0, 1), (1, 0), (1, 1) and (k, k) with $k \in$]0; 1[. We conclude that,

- the global minima are reached at the points (k,k) with $k \in$ [0;1] and are worth 0,
- the global maximum is reached at the point (1, 0) and (0, 1) and is worth 1.

7.7. Related Extrema

The split between free and bound (or "constrained") extrema arose from the impossibility of treating optimization in non-open domains according to the procedure outlined in the previous section. Indeed, the necessary condition does not apply not at the edges of the domain. However, if such points exist, their study on a case-by-case basis is generally not easy. Therefore, mathematical theory offers linked optimization methods. These incorporate directly into the resolution the constraints which define non-open domains. The nomenclature can be misleading. Indeed, at the operational level, it is not the intrinsic presence of constraints in the optimization which leads to abandoning free optimization in favor of linked optimization. Rather, it is the consequences of these restrictions at the level of the topological nature of the domain of definition of the function which guide the user towards one or the other of the techniques. So, in one area very limited, but open, the search for free extrema applies. Conversely, a constraint in the form of a non-strict inequality must always be taken into account to determine the related extrema.

Among the types of constraints that the modeler may find himself confronted with, two classes stand out. On the one hand, those which link the variables of the problem through one or more equations. These so-called equality constraints are applied prehended thanks to the LAGRANGE theorem, which provides a first-order condition formulated from an ad hoc function, called Lagrangian. In this approach, new variables, called multipliers, appear and offer an additional possibility in the analysis of the results. On the other hand, optimization under non-strict inequality constraints, generally treated using the KUHN and TUCKER theorem, will not be studied during this module.

7.7.1. Problem

Determine the extrema of a function of n variables, denoted $f(x)$, mathematically defined in an open domain $D \subset R^n$, but whose variables are subject to the constraints $g_1(x) = 0$, $g_2(x) = 0$, . . ., $g_m(x) = 0$, where the functions g_j are also defined in D. These constraints delimit the subset A of D in which the optimization is carried out:

$A = \{x \in D \mid g_1(x) = 0, \ldots, g_m(x) = 0\}$.

→ The definition of bound extremum results from that of free extrema.

7.7.2. Definition Linked Optimization

The function f, called objective function, admits at $x_0 \in A$ a linked maximum (resp. a linked minimum) under the constraints $g_1(x) = 0$, $g_2(x) = 0,\ldots$, $g_m(x) = 0$, if at this point it admits a free maximum (resp. a free minimum) in the domain A.

R.

It is excluded to have more constraints than variables so that the condition $m < n$ will be systematically imposed.

7.7.3. Example

The consumer maximizes a utility function, denoted $U(x, y)$, which depends on the quantities consumed of two goods, x and y, under a budget constraint $p_1*x+p_2*y = R$, where $p_1 > 0$ and $p_2 > 0$ are the prices of goods. This constraint expresses that the amount allocated to expenditure relating to the two goods considered is fixed at R. In this simple case which includes two variables and a linear constraint, we can easily reduce the optimization linked to the search for a free maximum. Indeed, the budget constraint makes it possible to explain the quantity of one good in relation to the other: $y = \frac{R}{P_2} - \frac{P_1}{P_2}.x$.....(7.1)

and we can define the function of a single variable $\tilde{U}(x) = U\left(x, y(x)\right)$. Under of the necessary condition for the function of a variable obtained after substitution of y as a function of x, we have

$$\tilde{U}(x) = \frac{dU\left(x, y(x)\right)}{dx}, \quad \text{.....(7.2)}$$

$$= \partial_x U(x, y) + \partial_y U(x, y).\, y'(x) \quad \text{.....(7.3)}$$

$$= \partial_x U(x, y) - \frac{p_1}{p_2}\partial_y U(x, y) \quad \text{.....(7.4)}$$

So,

$$\tilde{U}'(x) = 0 \Rightarrow \frac{\partial_x U(x, y)}{p_1} = \frac{\partial_y U(x, y)}{p_2} \quad \text{..... (7.5)}$$

This result indicates that at the optimum, marginal utilities (an economic term which simply means partial derivatives of the utility function in relation to the quantities consumed) weighted by the inverses of the prices equalize and denote this quantity by λ ¸ common. Alternatively, the budget constraint can be reformulated as $g(x, y) = 0$ ou

$$g(x, y) = p_1.x + p_2.y - R \quad \text{..... (7.6)}$$

So that,

$$\begin{aligned} p_1 &= \partial_x g(x, y) \\ p_2 &= \partial_y g(x, y) \end{aligned} \quad \text{..... (7.7)}$$

In other words, if we introduce the function

$$\frac{\partial_x U(x,y)}{p_1} = \frac{\partial_y U(x,y)}{p_2} = \lambda \quad \text{..... (7.8)}$$

$$\begin{cases} \partial_x U(x,y) - \lambda \partial_x g(x,y) = 0 \\ \partial_y U(x,y) - \lambda \partial_y g(x,y) = 0 \end{cases} \quad \text{..... (7.9)}$$

$$\nabla U = \lambda \nabla g \quad \text{..... (7.10)}$$

In other words, if we introduce the function $L(x,y,\lambda) = U(x,y) - \lambda g(x,y)$ at the optimum, $\nabla L = 0$ which corresponds to the necessary condition of existence of a free extrema for the function *L*.

LAGRANGE's theorem generalizes the approach adopted in this resolution. It represents a necessary condition for optimization under equality constraints.

7.7.4. LAGRANGE multiplier theorem

Let the functions f and $g_1, \ldots, g_m$ ($m < n$) of class C^1 in an open $D \subset R^n$. If *f* admits at x_0 an extremum linked under the constraints $g_1(x) = 0, \ldots, g_m(x) = 0$ and if the Jacobian at x_0 (*i.e.* the matrix $\partial_{x_j} g_i(x_0)$) is of rang *m*, then $\exists \lambda = (\lambda_1, \ldots \lambda_m) \in R^m$

Such as $$\nabla f(x_0) = \sum_{i=1}^{m} \lambda_i \nabla g_i(x_0) \quad \text{..... (7.11)}$$

LAGRANGE's theorem can be seen as the necessary condition for the existence of free extrema applied to the function of (n+m) variables called Lagrangian function (or the Lagrangian):

$$\begin{aligned} &L : D \times R^m \to R \\ &(x,\lambda) \to f(x_0) - \sum_{i=1}^{m} \lambda_i g_i(x_0) \end{aligned} \quad \text{..... (7.12)}$$

Attention

The LAGRANGE method just allows us to find the extrema x_0 candidates of *f* under the constraints $g_i(x) = 0$ but does not allow us to conclude on their nature. We must then study the behavior of *f* in the vicinity of each critical point in $\{x \in D \mid g_1(x) = 0, \ldots, g_m(x) = 0\}$. To do this, we study the sign of the distance function defined by $d(h) \equiv f(x_0 + h) - f(x_0)$,

pour $h \approx 0$ et $g_i(x_0 + h) = 0$ pour $i = 1, \ldots, m$.

- The value taken by a multiplier reflects the marginal influence of the level of the corresponding constraint on the value of the objective function at the optimum. We also say that the LAGRANGE multiplier measures the intensity of the constraint. Each multiplier expresses the sensitivity of the objective function to variation in the level of a constraint. For example, a constraint that does not influence the optimization (ineffective or superfluous constraint) is assigned a zero multiplier. On the other hand, a high multiplier corresponds to a constraint which significantly penalizes the optimum.

7.8. Optimize $f(x, y)$ under the constraint $g(x, y) = 0$

Let us rewrite LAGRANGE's multiplier theorem for a function of two variables and a single equality constraint. Let the functions f and g of class C^1 be in an open $D \subset R^2$. If f admits in (x_0, y_0) an extremum linked under the constraint $g(x, y) = 0$ and if $\nabla g\ (x_0, y_0) \neq (0,0)$, then

$$\exists \lambda \in R \text{ tel que } \nabla f(x_0, y_0) = \lambda \nabla g(x_0, y_0) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7.13)$$

We then have two practical methods for determining the linked extrema of f under the constraint g:

Method 1: Lagrangian

Let's form the Lagrangian

$$L : D \times R \to R$$
$$(x, y, \lambda) \to L(x, y, \lambda) = f(x, y) - \lambda . g(x, y) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7.14)$$

Where λ (LAGRANGE multiplier) is an unknown. For this function to have an extremum, the gradient of L must be zero, in other words we are looking for the triples (x, y, λ) such that

$$\begin{cases} \partial_x f(x, y) = \lambda . \partial_x g(x, y) \\ \partial_y f(x, y) = \lambda . \partial_y g(x, y) \\ 0 = g(x, y) \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7.15)$$

Let us denote (x_0, y_0, λ) a solution of this system. If $\nabla g\ (x_0, y_0) \neq 0$, then (x_0, y_0) is a critical point of the function f under the constraint g. These critical points satisfy the constraint, but it is now a matter of ranking these candidates.

$$\Delta(x_0, y_0, \lambda_0) \equiv \partial_{xx} L(x_0, y_0, \lambda_0) \partial_{yy} L(x_0, y_0, \lambda_0) - \left(\partial_{xy} L(x_0, y_0, \lambda_0)\right)^2 \quad \ldots\ldots\ldots \quad (7.16)$$

That is to say the determinant of the submatrix obtained from the Hessian of L by eliminating the last row and the last column.

- if $\Delta(x_0, y_0, \lambda_0) > 0$, $\partial_{xx} L(x_0, y_0, \lambda_0) < 0$ and $\partial_{yy} L(x_0, y_0, \lambda_0) < 0$ we have a local maximum at (x_0, y_0);
- if $\Delta(x_0, y_0, \lambda_0) > 0$, $\partial_{xx} L(x_0, y_0, \lambda_0) > 0$ and $\partial_{yy} L(x_0, y_0, \lambda_0) > 0$ we have a local minimum at (x_0, y_0);
- if $\Delta(x_0, y_0, \lambda_0) \leq 0$ we cannot conclude directly. We then study the sign of the difference $d(h,k) \equiv f(x_0 + h, y_0 + k) - f(x_0, y_0)$.

h and k being linked by the relation $\phi(h,k) \equiv g(x_0 + h, y_0 + k) = 0$.

If this difference has a constant sign for (h,k) close to (0,0), it is a local extremum (a maximum if $d < 0$, a minimum if $d > 0$). Otherwise, f does not present a local extremum at (x_0, y_0).

Method 2: Reduction

This Method is based on the possibility of expressing the constraint in parametric form. For example

- if there exists a function $h(x)$ such that $\{(x,y)\in R^2 | g(x,y)=0\}=\{x\in R | h(x)=y\}$, then optimize the function of two variables $f(x, y)$ under the constraint $g(x, y) = 0$ is equivalent to optimizing the function of a single variable $f(x, y = h(x))$;

- *if* there exists a function $h(y)$ such that $\{(x,y)\in R^2 | g(x,y)=0\}=\{y\in R | h(y)=x\}$, then optimize the function of two variables $f(x, y)$ under the constraint $g(x, y) = 0$ is equivalent to optimizing the function of a single variable $f(x= h(x),y)$;

- *if* there exist two functions $x = x(t)$ and $y = y(t)$ such that $\{(x,y)\in R^2 | g(x,y)=0\}=\{t\in R | g(x(t),y(t))=0\}$, so optimizing the function of two variables $f(x, y)$ under the constraint $g(x, y) = 0$ is equivalent to optimizing the function of a single variable $f(x = x(t), y = y(t))$.

7.8.1. Example

Let us determine the minima and maxima of the objective function $f(x, y) = 5x^2 + 6y^2 - x*y$ under the constraint $x+2y = 24$. To do this, let's construct the LAGRANGE function

$$\begin{aligned} &L: R^2\times R\to R \\ &(x,y,\lambda)\to L(x,y,\lambda)=f(x,y)-\lambda.g(x,y)=5x^2+6y^2-x-\lambda(x+2y-24) \end{aligned} \quad \text{..... (7.17)}$$

and cancel its gradient

$$\nabla L=\begin{bmatrix}0\\0\\0\end{bmatrix}\Leftrightarrow\begin{pmatrix}10x-y+\lambda\\12y-x+2\lambda\\-x-2y-24\end{pmatrix}=\begin{pmatrix}0\\0\\0\end{pmatrix} \quad \text{..... (7.18)}$$

We obtain $x = 6$, $y = 9$ and $\lambda = -51$. Like $\partial xxL(6,9,-51) = 10 > 0$, $\partial yyL(6,9,-51) = 12 > 0$, $\partial xyL(6,9,-51) = -1$ and $\partial xxL(6,9,-51)\,\partial yyL(6,9,-51) - \partial xyL^2(6,9,-51) > 0$, this is a minimum.

In this example we can explain a variable in the constraint, for example $y = 12 - \frac{x}{2}$

Then we can minimize directly the function $f\left(x,12-\frac{x}{2}\right)=7x^2-84x+864$ like $f'\left(x,12-\frac{x}{2}\right)=14x-84$ the minimum is found at $x=6,\quad y=12-\frac{6}{2}=9$ and the function is $f(6.9) = 612$.

7.9. Modern optimization method based Particle Swarm Optimization (PSO)

After introducing the basic and mathematical concepts of optimization, we proceed to apply one of the modern methods of artificial intelligence, for example Particle Swarm Optimization (PSO). Knowing that genetic algorithms are also one of the best methods of optimization, and because we mentioned them previously, we will discuss another method, which is Particle Swarm Optimization.

7.9.1. Particle Swarm Optimization (PSO): Particle Swarm Optimization is a method inspired by biology to solve optimization problems.

Like artificial neural networks, genetic algorithms or ant colony algorithms, Particle Swarm Optimization (PSO) is a bio-inspired algorithm. It is based on the principles of self-organization which allow a group of living organisms to act together in a complex way, based on simple "rules". The PSO is inspired by the model developed by Craig Reynolds to simulate the gregarious movement of certain animals (herds of cattle, flocks of birds, etc.). In this model, each artificial bird, or "boid" (bird-oid object), moves randomly following three simple rules:

- Cohesion: the boids are attracted towards the average position of the group;
- Alignment: the boids follow the same path as their neighbors;
- Separation: to avoid collisions, the boids keep a certain distance between them.

The PSO introduces another principle: the boids do not move randomly, they have an objective to achieve. This is determined by a function to be optimized or "objective function" which is provided by the user, and which depends on the application concerned.

7.9.2. How does this algorithm work? PSO explores the search space through successive tests of body positions, their movements being managed by simple equations. Thus, the location of each boid in the search space represents a potential solution to the optimization problem. And the "quality" associated with each of these solutions is quantified by the objective function, optimized little by little according to the more or less optimal positions.

Concretely, most often, the locations and velocities of the boids are represented as vectors of D-dimensional numbers, the initial positions and velocities often being defined randomly. Then, we repeat the exploration by updating the position of each body then its speed vector until reaching a satisfactory solution. We evaluate the level of quality associated with the

position of each boid using the objective function. We thus determine the best boid and the best position that each boid has encountered up to that moment. Then, the velocity vector of each boid.

- A speed vector starting from X and going towards the best body of the test (red arrow on the diagram);
- A speed vector going towards the best position that the boid visited (green arrow);
- The previous speed vector (blue arrow).

Intuitively, the actions of the different bodies in the test simultaneously allow the search space to be explored and the most promising areas to be exploited. A large number of variations of this algorithm have been developed and are used in various application frameworks.

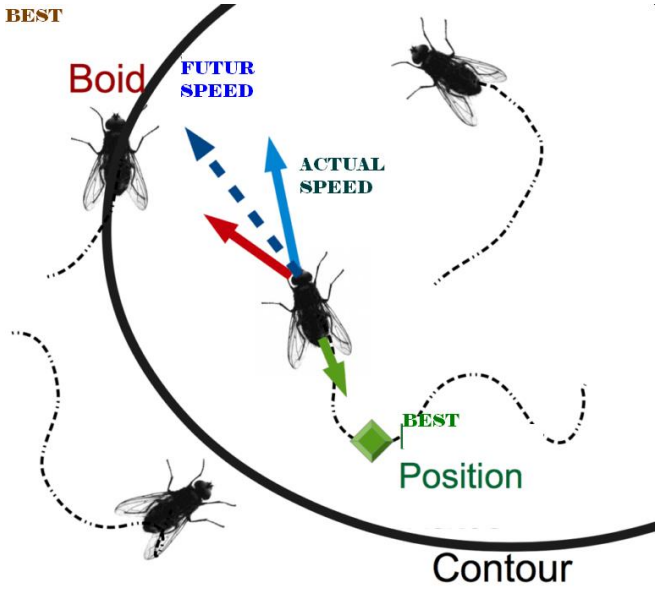


Figure .7.1. PSO illustrative figure.

7.9.3. **Example 1:** find the maximum of real function **based PSO**

- Find the maximum of the following function $F(x)= -x^2+2x+11$ in the interval [-4,4].
- We use four particles (N=4) initial position x_1= -1.5, x_2= 0, x_3 = 0.5 and x_4= 1.25. we assume the initial velocities for each vector is zero, and we intialize weight *as* $c_1=c_2=\theta=1$.

Solution

Step 1: An evaluation of the objective function of the initial position

x_1= -1.5 ⟶ $F(x_1)$= 5.75
x_2= 0 ⟶ $F(x_2)$= 11
x_3 = 0.5 ⟶ $F(x_3)$= 11.75
x_4= 1.25 ⟶ $F(x_4)$= 11.93

- The initial velocities for each vector are zero: v_1=0; v_2=0; v_3=0; v_4=0.
- The iteration number t=1

Step 2: Find the best $\boldsymbol{p_{best,i}}$ of each particles and $\boldsymbol{g_{best}}$

We can set $p_{best,1}$= x_1= -1.5, $p_{best,2}$= x_2= 0 , $p_{best,3}$= x_3 = 0.5 , $p_{best,4}$= x_4= 1.25.

And g_{best} *is the Max of function F(x)* so g_{best}= x_4= 1.25.

Find the position and velocities of each particle

$$v_i(t) = \theta . v_i(t-1) + c_1 \cdot r_1 (P_{best,i} - x_i(t-1)) + c_2 \cdot r_2 (g_{best,i} - x_i(t-1)) \quad \text{…….. (7.19)}$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad \text{…….. (7.20)}$$

$$v_1(1) = 0 + (1*0.3294*(-1.5-(-1.5))) + 1*.0.9542(1.25-(-1.5)) = 2.6241$$
$$v_2(1) = 0 + (1*0.3294*(0-0)) + (1*.0.9542(1.25-0)) = 1.1927$$
$$v_3(1) = 0 + (1*0.3294*(0.5-0.5)) + (1*.0.9542(1.25-0)) = 0.7156$$
$$v_4(1) = 0 + (1*0.3294*(1.25-1.25)) + (1*.0.9542(1.25-1.25)) = 0$$

The new position : $x_i(t) = x_i(t-1) + v_i(t)$

- at t=1 the velocities for each particle are: v_1=2.6241; v_2=1.11927; v_3=0.7156; v_4=0.
- at t=0 the initial position are: x_1= -1.5, x_2= 0, x_3 = 0.5 and x_4= 1.25

$$x_1(1) = -1.5 + 2.6241 = 1.1241$$
$$x_2(1) = 0 + 1.1927 = 1.1927$$
$$x_3(1) = 0.5 + 0.7156 = 1.2156$$
$$x_4(1) = 1.25 + 0 = 1.25$$

Check the new position within search space (the interval [-4,4])???

$$x_1(1) = 1.1241 \rightarrow f(x_1) = 11.9846$$
$$x_2(1) = 1.1927 \rightarrow f(x_2) = 11.9628$$
$$x_3(1) = 1.2156 \rightarrow f(x_3) = 11.9535$$
$$x_4(1) = 1.25 \rightarrow f(x_4) = 11.9375$$

Check the convergence of the current solution. Since the value of $x_i(t)$ did not converge , t=t+1 go to step 2.

Step 2: Find $\boldsymbol{p_{best,i}}$ of each particles and $\boldsymbol{g_{best}}$

$$x_1(0) = -1.5 \rightarrow f(x_1) = 5.75 \quad x_1(1) = 1.1241 \rightarrow f(x_1) = 11.9846 \rightarrow p_{best,1} = 1.1241$$
$$x_2(0) = 0 \rightarrow f(x_2) = 11 \quad x_2(1) = 1.1927 \rightarrow f(x_2) = 11.9628 \rightarrow p_{best,2} = 1.1927$$
$$x_3(0) = 0.5 \rightarrow f(x_3) = 11.75 \quad x_3(1) = 1.2156 \rightarrow f(x_3) = 11.9535 \rightarrow p_{best,3} = 1.2156$$
$$x_4(0) = 1.25 \rightarrow f(x_4) = 11.9375 \quad x_4(1) = 1.25 \rightarrow f(x_4) = 11.9375 \rightarrow p_{best,4} = 1.25$$

As the pb is to find Maximum so $g_{best} = 1.1241 \rightarrow f_{gbest} = 11.9846$

$p_{best,1} = 1.1241,\ p_{best,2} = 1.1927,\ p_{best,3} = 1.2156,\ p_{best,4} = 1.25,\ g_{best} = 1.1241$

Now find the position and velocities such as;

$$v_i(t) = \theta . v_i(t-1) + c_1 \cdot r_1 (P_{best,i} - x_i(t-1)) + c_2 \cdot r_2 (g_{best,i} - x_i(t-1))$$
$$x_i(t) = x_i(t-1) + v_i(t)$$

Iteration t_1:

$x_1 = 1.1241,\quad x_2 = 1.1927,\quad x_3 = 1.2156,\quad x_4 = 1.25$

$v_1 = 2.6241,\quad v_2 = 1.1927,\quad v_3 = 0.7156,\quad v_4 = 0$

$p_{best,1} = 1.1241,\ p_{best,2} = 1.1927,\ p_{best,3} = 1.2156,\ p_{best,4} = 1.25,\ g_{best} = 1.1241$

$$v_1(2) = 2.6241 + (1*0.1482*(1.1241 - 1.1241)) + 1*0.4867(1.1241 - 1.1241) = 2.6240$$
$$v_2(2) = 1.1927 + (1*0.1482*(1.1927 - 1.1927)) + (1*0.4867(1.1241 - 1.1927)) = 1.1593$$
$$v_3(2) = 0.7156 + (1*0.1482*(1.2156 - 1.2156)) + (1*0.4867(1.1241 - 1.2156)) = 0.6711$$
$$v_4(2) = 0 + (1*0.1482*(1.25 - 1.25)) + (1*0.4867(1.1241 - 1.25)) = -0.0613$$

$$x_i(t) = x_i(t-1) + v_i(t)$$

Iteration t_1 : $x_1 = 1.1241,\quad x_2 = 1.1927,\quad x_3 = 1.2156,\quad x_4 = 1.25$

Iteration t_2 : $v_1 = 2.624,\quad v_2 = 1.1593,\quad x_3 = 0.6711,\quad v_4 = -0.0613$

$$x_1(2) = 1.1241 + 2.6240 = 3.7481$$
$$x_2(2) = 1.1927 + 1.1593 = 2.3520$$
$$x_3(2) = 1.2156 + 0.6711 = 1.8867$$
$$x_4(2) = 1.25 + -0.0613 = 1.1887$$

Check the new position within search space (the interval [-4,4]

Check the convergence of the current solution. Since the value of $x_i(t)$ did not converge , t=t+1 go to step 2.

$$\chi_1(2) = 3.7481 \rightarrow f(\chi_1) = 4.448$$
$$\chi_2(2) = 2.3520 \rightarrow f(\chi_2) = 10.172$$
$$\chi_3(2) = 1.8867 \rightarrow f(\chi_3) = 11.213$$
$$\chi_4(2) = 1.1887 \rightarrow f(\chi_4) = 11.964$$

$$g_{best} = 1.1241 \rightarrow f_{gbest} = 11.9846$$

After two iterations the best value found is $f(x)$= 11.9846 at x= 1.1241.

…The best solution can be found after much iteration.

7.9.4. Example 2: particle swarm optimization applied to the power flow computation

This example is well described in the reference [39];

The basis of the PSO algorithm consists in, instant time, analyzing the displacement of each particle in search for the best position and updating its velocity and position using specific equations. The iterative process proceeds until all the particles converge to the obtained global best, which is the adopted solution to the treated problem.

The proposed PSO algorithm is applied to the computational achievement of the load flow solution, based on the minimization of the power mismatches in the system buses. The particles' positions are defined as the voltage modules and angles of the buses. Applying the PSO algorithm, instead of calculating these voltages through the SFLE, initial estimated values are adopted and updated at each process' iteration with the PSO equations, in order to obtain the lowest possible power mismatches.

The particles positions can assume continuing values within the limits specified in the input data. The rule function parameters that will be minimized in the PSO algorithm are defined as *grades*. The grades are defined as the arithmetic mean of the buses apparent power. Each particle has a *local grade*, value obtained by its local best. The *global grade* is the grade related to the best global of all the particles. The *current grade* is the grade obtained by a particle at a given iteration.

The first step of the algorithm is to generate the initial values to the particles positions, velocities, local best parameters and global best parameters. The angles receive a random initial value within the specified boundary. Before the initialization of the module value of

each particle, the bus type needs to be verified and related in the equation. In the case of a PQ bus, the voltage module receives a random value within the specified boundary; for a PV bus, the voltage module receives the related value specified in the input data. The initial velocities are null. The local best parameters receive the particles positions values and the global best parameter receives the first particle value, arbitrarily. The grades are initialized with high values in order to be minimized later. Having that accomplished, the iterations are initialized. The following process is accomplished to each particle of the swarm. Firstly the buses voltages receive the particles positions. The reactive power of the PV buses is calculated using equation (7.1), then the active and reactive power of the slack bus are also calculated using this equation. Finally the power flow in the system lines is calculated in accordance to the equation (7.2).

$$P_i - jQ_i - y_{i1}V_1V_i - y_{i2}V_2V_i - \ldots - y_{in}V_nV_i = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (7.21)$$

$$S_{ij} = P_{ij} + jQ_{ij} = V_i(V_i^{*} - V_j^{*})Y_{ij}^{*} + V_iV_i^{*}Y_{sh,I} \ldots\ldots\ldots\ldots\ldots\ldots \quad (7.22)$$

$$v(t+1) = 1.4\, w.v(t) + 2.6\,(1-w).[r(l(t+1) - x(t+1) + (1-r).((g(t+1) - x(t+1))] \ldots\ldots\ldots (7.23)$$

$$x(t+1) = x(t) - v(t+1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7.24)$$

$$w = 1 - t/ni \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7.25)$$

Thus, once all the power of the buses and of the lines is known, the active and reactive power mismatches of each bus are calculated. They are calculated as the sum of the injected power in the approached bus. The apparent power mismatches arithmetic mean is obtained, and this is the value that is desired to be minimized. The local best is replaced by the current particle position in case of the particle current grade is considered better than the local grade. Thus, after all the particles pass through the described process, a similar criterion is used to the global best updating. Next each particle is verified in the following criteria: whether the local grade or global grade is best, the best global is replaced by the approached best local. The velocities as well as the particles positions are updated according to the equations (7.23), (7.24) and (7.25); which are, respectively: velocities equation, positions equation and inertia weight equation. Such equations are based on the classical PSO equations and have had modifications and coefficients adjusted empirically for an improved efficiency in resolving problems.

7.10. References

[52] Rodolphe Le Riche1*;* Stéphane Mottelet, Eric Touboul, «*Optimisation locale et globale*», Ecole des Mines de Saint-Etienne 2 Université de Technologie de Compiègne, 2010.
[53] Aude Rondepierre & Pierre Weiss, «*Méthodes standards en optimisation non linéaire déterministe*», 2018.
[54] Gérard Verfaillie, «Cours *Optimisation Partie Optimisation Combinatoire 3ieme année ISAE Année*», *ONERA/DCSD/CD*, Toulouse, 2008-2009.
[55] «*Optimisation I Recueil d'exercices corrigés et aide-mémoire*». *http://faccanoni.univ-tln.fr/enseignements.html*, 2018.
[56] Gloria Faccanoni, Camila P. Salomon, Germano Lambert-Torres, Helga G. Martins, Cláudio Ferreira, Cláudio I. A. Costa, «*Load Flow Computation via Particle Swarm Optimization*», Conference Paper , DOI: 10.1109/INDUSCON.2010.5740044 · Source: IEEE Xplore, December 2010.