MS Software Development and Business Information Systems, 2nd year

Iaşi, 2020-2021

Artificial Intelligence for Business

Table of contents:

Chapter 1 - Artificial Intelligence and its application domains

Artificial Intelligence beginnings can be seen immediately after World War II, in the first programs that solve puzzles or play certain games. There were two reasons (besides the fascination it exerts on many) for which games were among the first areas of application for artificial intelligence - firstly, because program performance is easy to measure (most of the time, you either win or lose a game); secondly, that the rules are generally simple and few in number, so they can be easily described and used.

Another area that raised interest in this first period was the demonstration of theorems. This area is similar to that of gaming by the fact that the performances are simple to evaluate (you are subject to a classical theorem). What is to be described to the computer is the set of axioms and the rules of inference (ie, the rules by which new truths are obtained from certain given truths). Several exciting programs have been created in this area, including Newell's „The Logic Theorist'', which demonstrate the theorems in the first chapter of the book „Principia mathematica'' written by Whitehead and Russell, and a program by Gelenter that demonstrates geometry theorems.

Artificial intelligence has become more lucid, more critical of itself and, to a certain extent, more pragmatic. Enthusiasm with psychological flavor-like related to the understanding have faded, and, at the same time, emerged the first efficient and cost-effective expert systems applied in the industry. Interest falls more on a compact and uniform representation of knowledge; people begin to doubt the opportunity of general methods in solving problems.

By the 1980s one of the first expert systems in the industry was created - R1 (now XCON), built at Carnegie Mellon University, S.U.A with the collaboration of DEC (Digital Equipment Corporation). R1 is in charge of configuring computing systems (any VAX computer manufactured by DEC is configured with R1). It has been put into production, and since then the interest of the world for artificial intelligence has increased considerably. As a result, computer scientists in this field have been divided into 'implementers' artificial intelligence techniques and researchers.

1.1 From natural intelligence to artificial intelligence. Definitions

The use of computing to solve complex problems of evaluation, adjustment, diagnosis, forecasting of an economic system has a history that can be considered already long. The current generation of computers manages to implement methods and algorithms that are able to capture the accumulated experience of a wide range of situations, and to even learn from new experiences. These methods are generically grouped under the name of artificial intelligence techniques, and come to support the frantic race of market forces obliging manufacturing firms to launch new generations of products at shorter times.

In order to define and understand artificial intelligence we need to know what natural intelligence is, how the human brain-responsible for natural intelligence works and how intelligence can be measured.

What is natural intelligence? Although natural intelligence is difficult to define, its features can outline this concept:

- the ability to understand, extract the essence of a problem,
- acquiring and memorizing knowledge, gathering information and knowledge, and building a knowledge base,
- the ability to respond quickly and successfully to new problems,
- accessing experience,
- solving new situations / problems based on previous experience,
- establishing decisional alternatives and supporting decision making in this regard.

The concept of artificial intelligence was introduced in 1956 by McCarthy (MIT) and is closely related to two fundamental ideas:

- involves studying human processes of thought to understand what intelligence is;
- deals with the representation of these processes, via computer, robots or other intelligent machines.

Artificial Intelligence is a subdomain of computer science that aims to transform computers into machines capable of acting like humans, namely to reason as they do and to solve problems like them. Artificial intelligence is itself a commercial domain, it is a science and at the same time, a technology based on concepts and ideas arising from research, but they cannot be marketed as such. Artificial Intelligence, however, provides a scientific foundation for many highly profitable business technologies: intelligent systems of all kinds, robots, sensors, smart computers, intelligent instructors, etc.

Some of the most representative definitions of artificial intelligence are:

- Artificial Intelligence is a subdomain of computer science that studies how machines can be determined to act intelligently (Jackson,1996)
- Artificial Intelligence is a derivative of natural intelligence that takes on concrete social forms in the sense that it develops the social intelligence directly and indirectly, as a result of the interests in the economic and social activity (Drăgănescu).

According to the definition given by Elaine Rich, namely " the artificial intelligence is studying how to make electronic computers to execute tasks for which man is today the best", human brain shows a great complexity and effectiveness incomparable in many areas. This efficacy is due to the specific reasoning and learning mechanisms that attempt to model artificial intelligence.

Also very suggestive is Patrick Winston's statement that artificial intelligence deals with the study of ideas that allow computers to do those things that make people look intelligent. The main objectives of artificial intelligence are to make electronic computers more useful and to understand the principles that make intelligence possible.

There are valuable Romanian researchers who have treated this field with recognized results. For example, Luca Dan Serbanati and Cristian Giumale show that "Artificial Intelligence means that part of computer science that designs and constructs intelligent machines, those machines capable of performing functions of the human intellect".

Artificial Intelligence is today designed for the most diverse types of products, where microprocessors and smart programs can be incorporated, not just in the field of computers. It is particularly concerned with "endowing" them with intelligent, human-like behavior.

We consider it appropriate to mention the fundamental questions that artificial intelligence attempts to find answers:

- To what extent will intelligent machines become part of our lives?
- Can equipment be built to possess consciousness?
- Are we able to build such equipment? If so, could we control them?
- What are the main technological obstacles?

1.2 The evolution of artificial intelligence

Although artificial intelligence is a recent field of modern technology, it has a rich history and is closely related to that of information technologies. Here are some of the most significant historical moments of the evolution of artificial intelligence and its specific concepts:

- 1931 – Kurt Godel (Austria) using predicate logic proves that all truth claims are derivable
- 1937 - Alan Turing – halting problem = a limit of intelligent machines (the first machine that solves mathematical problems with a set of rules and an alphabet)
- 1943 – McCullough & Pitts propose a neural network model and establish connection with propositional logic
- 1950 - Alan Turing defines the concept of intelligent machine (Turing test) and develop concepts for machines capable of learning and genetic algorithms
- 1951 - the first electronic computer launched on the market (UNIVAC I). In the same year, Marvin Minsky creates a device equipped with a neural network called SNARK (neural computer)
- 1955 – Arthur Samuel (IBM) is developing a chess program, capable of learning, a chess game that is plays chess better than its creator. In the same year the artificial intelligence language IPL II is developed by A. Newell, J.C.Shaw, H.Simon
- 1956 – McCarthy introduces the term artificial intelligence at a conference held at Dartmouth College. In the same year, Newell & Simon (CMU) presents the Logic Theorist - the first program capable of symbolic processing

- 1958 – McCarthy (MIT) invented the LISP language (self-editing programs) for artificial intelligence
- 1959-1961 - Newell & Simon develop GPS (General Problem Solver) which aims to imitate human thinking process
- 1963 – McCarthy founds the first Artificial Intelligence research lab at Stanford University
- 1965 - DENDRAL - the first expert system to analyze molecular structures (Buchanan, Feigenbaum, Lederberg)
- 1966 - J.Weisenbaum invented Eliza - conversation in natural language (thus creating the basis of chatboot)[1]
- 1967 - the first artificial intelligence lab, created at MIT, SUA; INTERNIST(Pople, Myiers)-diagn.medical = CADUCEUS(1982);
- 1968 – semantic web (Quillian);
- 1970 - PROLOG language(Colmerauer, Roussel) - France;
- 1973 - MYCIN-medical diagnose;
- 1974 – first robot controlled by computer;
- 1982 – R1 (expert system for computer configuration)
- 1986 – rebirth of neural networks -> Nettalk system
- 1990 – developing multiagent systems
- 1997 – IBM Deep Blue, chess computer (plays chess better than Gary Kasparov)
- 2003 – 2011 a multitude of projects are being developed to design and build robots (Japan - RoboCup)
- 2011 - IBM Watson: Smartest Machine ever built

It is necessary to mention that through their work, Newel and Simon popularize the use of rules for human knowledge representation and rule based reasoning, and the inference engine of expert systems corresponds to the cognitive processor proposed by these two titans in the field of artificial intelligence. The model of solving human problems in terms of long-term memory (rules), short-term memory (working memory), and cognitive processor (inference engine) are the basis of current intelligent systems (RBES-Rule Based Expert Systems). Rules-based systems are the most popular method of implementing today's expert systems. The many researches and multiple synthesized achievements are only a summary of the most important developments covering major projects and years of work, all demonstrating that reality has overtaken fiction for the benefit of humanity.

Highlights of the evolution of artificial intelligence:

[1] http://www.manifestation.com/neurotoys/eliza.php3

1. Turing Test – is the long-term ideal of artificial intelligence as a branch of informatics.

- The Turing test starts with a game created by Turing - "the game of imitation" - with three players: a machine (A), a person (B) and a second person (C). A and B are not in the same room as C. C does not know which of the other two players is the machine and cannot see nor speak directly to them. Communication can be done in writing or through a terminal. C's purpose is to differentiate the machine from man based on answers to any kind of questions. If C fails, then the machine can be considered intelligent.
- In 1950, when the article about the test was published, the author predicted that in 50 years (in 2000) it would be possible to have a computer capable of playing the imitation game so well that C's chance to correctly identify man is less than 70% after 5 minutes of play. Today's computer scientists are far less optimistic. In fact, there are even two camps - some who believe in the ability to meet (sometime) Turing's test, and others who are convinced on the contrary.
- An article appeared on June 9, 2014[2] headlines that a computer has passed Turing's test (Computer AI passes Turing test in 'world first').

2. IBM-Deep Blue vs. Garry Kasparov (May, 11th 1997)

This year (2017) marks 21 years since IBM's Deep Blue computer defeated world champion Gary Kasparov on February 10, 1996, in the first of six Philadelphia games. Kasparov went over that failure and then won three games and drew two more, winning a 4-2 duel with advanced technology. The Russian then received a $ 400,000 award for defeating the American computer.

The machine did not say the last word, and in May 1997 a refined version of Deep Blue confronted Kasparov again, who had to admit that he was defeated with the score 2,5-3,5 after six games.

In January 2003, Kasparov accepted a new challenge with Deep Junior for a $ 1 million prize. The man and the machine won one victory each, and three other games ended with a draw. In the last game, Kasparov gained a decent position and requested a draw, accepted by the American team operating the computer. Deep Junior was the first car to defeat Kasparov with black pieces.

1.3 Areas that contributed to the foundation of artificial intelligence

The development of cars with intelligent features involves the use of the most diverse sciences and technologies: linguistics, psychology, philosophy, computer science, mechanics, hydraulics, biology and optics. The intersection between psychology and artificial intelligence is based on areas such as cognitive science and psycholinguistics. Philosophy and artificial intelligence converge to areas such as logic, philosophy of language, and philosophy of thought. Intersections with linguistics include computational linguistics, psycholinguistics and sociolinguistics, and interactions between electrical engineering and artificial intelligence include image processing, control theory, form recognition, and robotics.

[2] http://www.bbc.com/news/technology-27762088

Contributions to artificial intelligence were subsequently brought by management, organizational theory, statistics, mathematics, and others.

It can be seen from these short explanations the variety of disciplines that participate in artificial intelligence either by overlapping or by interaction. However, it is difficult to classify the field of artificial intelligence according to these disciplines (see Figure 1).

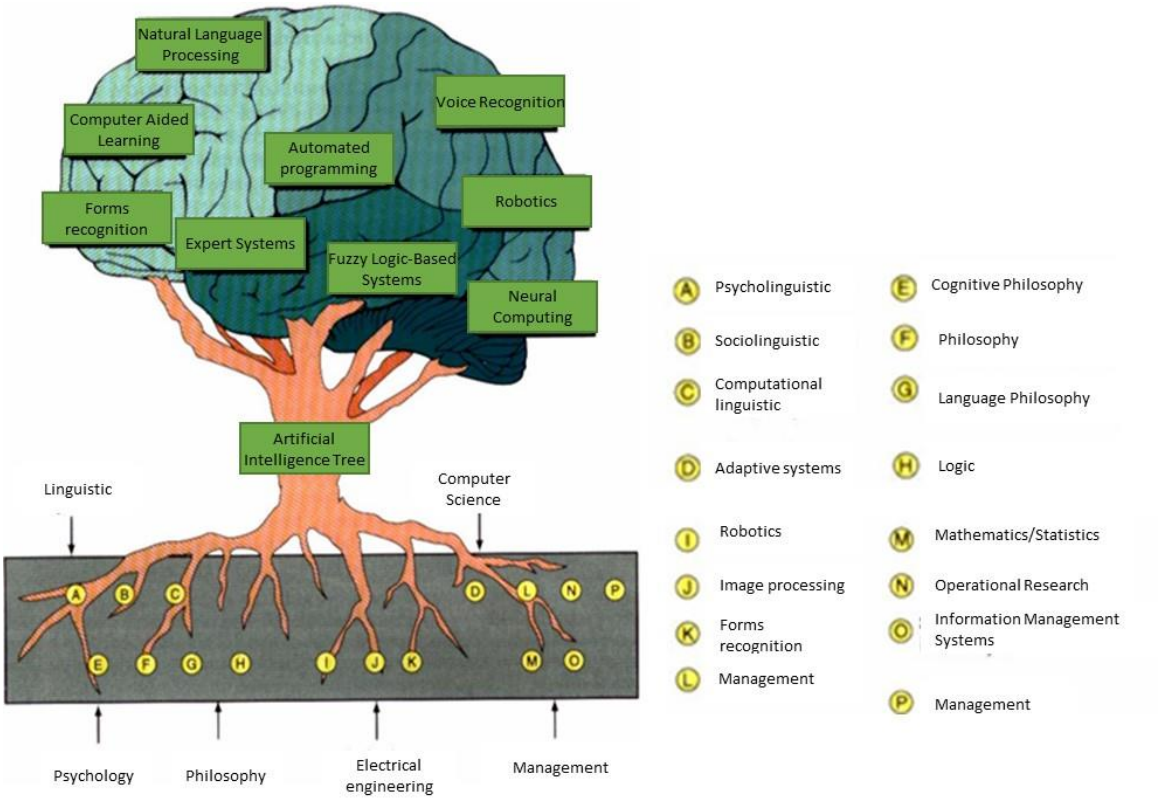


Figure 1 The domains of Artificial Intelligence

1.4 Differences in Intelligent vs. Conventional Computer Programs

Intelligent programs are based on:

- Symbolic representation and manipulation of symbols;
- use of specific concepts and basic techniques such as search and pattern matching;
- algorithms are used to implement search processes, as artificial intelligence is based on a different paradigm of computer programming;
- criteria such as:
    - the type of the processing;

- nature of entries;
- search;
- explanations;
- the main goal;
- the structure;
- the nature of the outputs;
- maintenance and upgrading;
- the ability to reason.

All these characteristics significantly differentiate intelligent programs from conventional ones (see Table 1).

Table 1 Differences in Intelligent vs. Conventional Computer Programs

Criteria	Artificial Intelligence programs	Conventional programs
Processing	**Mainly simbolic**	**Mainly numerical**
Inputs	**May be incomplete**	**Mandatory complete**
Search	**Mainly euristic**	**Algoritmic**
Explanations	**Offfers explanations**	**Usually do not offer explanations**
Main goal	**Knowledge**	**Date and informations**
Structure	**Separation of control knowledge**	**Control is integrated with data and information**
Outputs	**May be incomplete**	**Mandatory accurate and complete**
Maintenance and upgrading	**Relatively easy due to the modularity of knowledge**	**Ussually difficult**
Hardware	**Mainly PCs and workstations**	**All types of computers**
The reasoning capacity	**Yes**	**No**

1.5 Artificial Intelligence in Applications. Categories of intelligent systems

The interest in artificial intelligence has increased in recent years due to spread of applications distributed, meaning that increasingly more businesses and organizations have begun to use this

technology and have specialized in this field. A relatively complete perspective of artificial intelligence applications is provided in the diagram below (Figure 2):

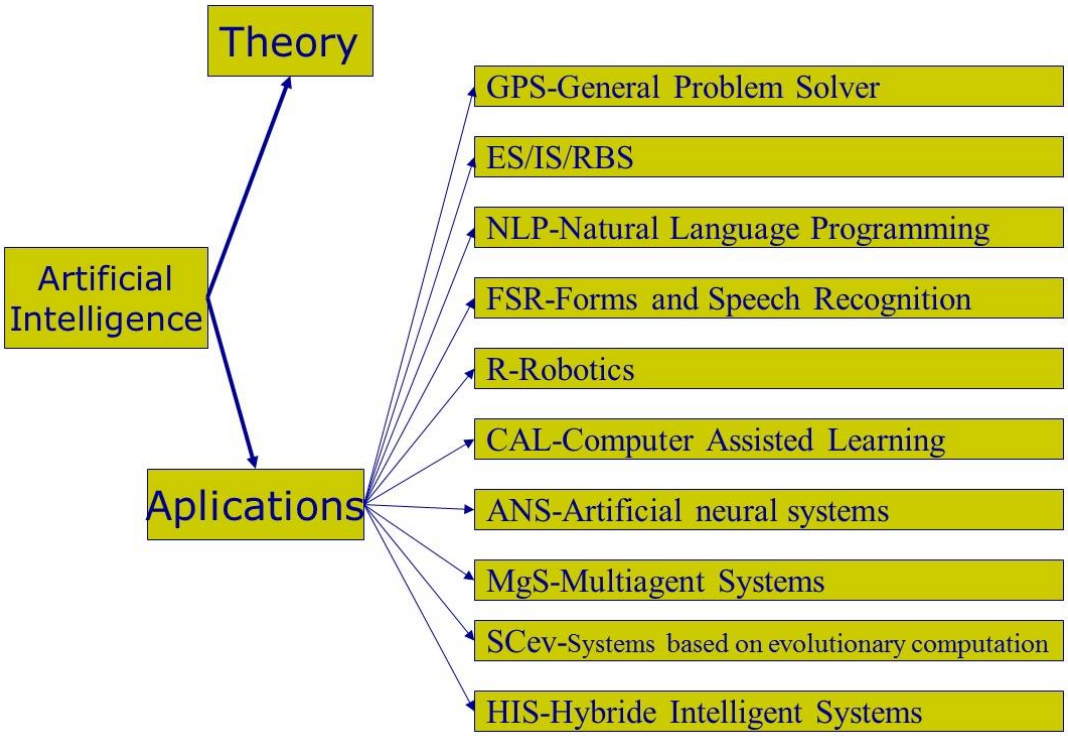


Figure 2 The applications of Artificial Intelligence

GPS

For many years, one of the purposes of artificial intelligence has been to develop techniques to help people, researchers, economists, engineers, etc., to solve problems more quickly and easily. The efforts in the 1950s and 1960s were geared towards achieving a model to solve a broad range of problems. In the 1970s, scientists in artificial intelligence recognize that to develop such a model (General Problem Solver) It requires a better approach to the specific problem. This recognition has catalysed concerns and / or succeeded implementation achievements in the planning and programming directions - two issues of particular interest to businesses and production in general. It has been possible to develop artificial intelligence programs for optimal planning in which the steps and limitations of the problem are communicated within the inputs and the outputs obtained are the final outcome - optimal plans. Also in enterprise programming, which requires the involvement of staff, stocks and other resources, maximizing efficiency and minimizing costs is being pursued throught artificial intelligence programs already in place, able to address the most complex issues.

Another type of problem, which has always been the object of research and design for intelligent systems, is the demonstration of theorems and the reduction of formulas. And for this, there are currently well-developed artificial intelligence techniques capable of verifying the results of lab work or the respective fields.

ES/IS/RBS

- have the best technology available
- use knowledge called expertise, obtained from human experts through the process of acquiring knowledge
- are unable to learn and self-educate
- a special category is based on Case Based Systems Frame Based Systems

NLP

Since the beginning of computer production, communication between computers and people has not been natural from the human factor point of view. Today, attention is increasingly focused on improving human-computer communication. More and more researchers are convinced that the ideal way for communicating with computers is the spoken, natural, everyday language of humans instead of keyboards or other communication tools. Significant results have already been achieved in the use of natural language for communication with electronic computers.

The problems of communication in natural language, however, remain very complex; the natural language is not easy to understand because of multiple meaning (polysemy), depending on the context, and systems developed for the use of natural language are far from yielding a proper return for the full understanding of the spoken language, they are limited to a set of linguistic structures. Natural language understanding systems use one of the two basic techniques: (1) the key word and (2) syntactic / semantic analysis.

Systems using the "key word" technique try to inherit a sense of communication, precisely from the meaning of the key word of communication.

Systems that use "syntactic / semantic analysis" are more promising to producing commercial systems. In these systems a phrase is fragmented (parsed) into parts of speech (subject, predicate) and attempting to infer the meaning of this analysis. Always a large amount of memory is needed for a more comprehensive dictionary - for example the ELIZA system.

Closely related to the processing of natural language, there are already achievements for speech recognition and speech synthesis. Speech recognition systems understand the natural language they accept in voice input. The analog voice signal is converted into numerical signals that can be processed in the computer. These systems are capable of recognizing only the human voice, and only a small number of words.

Speech synthesis systems generate speech in natural language. Attempts are now being made for people to hear and understand the computer's communication in natural language, and the results are already appreciated.

FSR

Forms and Speech Recognition or Computer Vision is a distinct sub-domain of artificial intelligence that deals with the provision of computers with the ability to view and perform certain tasks on that basis. Forms recognition systems are able to identify the traits of real objects or their images, and the

information thus obtained uses them to solve certain problems. In such situations, search and pattern matching techniques are used to retrieve visual information. They can sense details that get rid of the human eye. There are already applications in several areas: robotics, aerial photography, etc.

These systems are very complex and require a large amount of resources, are limited to real-time situations, and further efforts are being made to improve the speed of processing and memory management.

R

Robotic means equipping devices with the ability to move. Robots, at first a fiction, now have a significant number of features and are able to execute a large number of tasks, especially in manufacturing, in a dangerous environment for humans, in repetitive and tiring situation (in the mine, working with explosives etc.). Typically robots have an arm with several joints that are moved under the control of an intelligent program. The purpose of robotics research is to incorporate intelligent programs in robots that allow them to perform tasks with greater "intelligence" and possibly expert systems and voice recognition and synthesis systems.

CAL

Computer-assisted learning is a sub-domain of artificial intelligence in connection with a highly developed user interface and an expert system that controls and adjusts the level of the course according to the student's level and the difficulty, compares the techniques used by the student with those of the expert in the field and helps the student in solving process when he stuck in the rationale he has done.

ANS

Artificial neural systems (also known as artificial neural networks) are networks capable of learning and performing tasks that can not be achieved by conventional computers / programs. These systems use the neural network method to acquire and represent knowledge and can solve problems by training artificial neurons. One of the most "complete" definitions of artificial neural systems has the following statement: ANS is a massively distributed parallel computer that acquires new knowledge based on past experience and makes them available for later use (S.Haykin, 1994).

It is known that these systems are designed on the basis of the natural neuronal system, as a result of the resemblance of SNA with the human brain resulting in the following:

- knowledge is acquired by the neural network through a learning process
- the network requires continuous training
- knowledge is stored in inter-neural connections (synaptic weights)

Artificial neural systems are primarily used to resolve unstructured problems and use patterns instead of If-Then-Else rules. An example of such a network would be to implement a facial recognition system.

MAgS

The multi-agent system is made up of a number of intelligent software agents of the same or different type, specialized, acting separately to solve a problem collectively, interacting through messaging. Individual agents employed in a multi-agent system have different goals or motivations. Multi-agent

systems can solve problems that are too difficult (practically unresolved) for an individual software agent or a monolithic software system (which works in the "mono-agent" mode). For the success of the interaction, agents must have the ability to cooperate, coordinate and negotiate with other agents.

SCev

Systems based on evolutionary computation are inspired by evolutionary processes in nature, based on the principles of heredity and the survival of the best individual (the one that best adapted to the environment). These systems are based on determining the solution of a problem by exploring the space of potential solutions using a population of "seekers" (agents or individuals) and the population elements are coded according to the specificity of the problem (bits, real value vectors, trees, etc.) .

Evolution-based systems use a mechanism inspired by the functioning of biological systems, are being performant in search issues and identifying structures in databases and data mining bases and are easily adaptable to environmental changes and can be used to discover knowledge .

HIS

In the past few years, three categories of intelligent techniques have been proposed: chaining techniques, application's task divisions and multifunctional achievements.

Chained techniques consists in the integration of different methods specific to artificial intelligence, with the goal of eliminating the limitations that each of them suffers individually and obtaining successful implementation solutions for complex systems. Application's task divisions apply to intelligent systems for which no individual technique has the ability to solve all sub-problems imposed by the given application. Multifunctional achievements are hybrids that functionally mimic different processing systems, that is, they offer the possibility of multiple processing of information in a particular architecture.

Depending on the combinations of intelligent techniques, processing architecture, communication requirements, and mode of operation, three classes of hybrid intelligent systems are defined: Functional Replacement Systems, Intercommunication Systems, and Polymorphic Systems. In addition, some combinations with conventional methods such as numerical computation, statistical series, regression techniques, or linear programming have been proposed.

Intelligent Hybrid Functional Replacement Systems implements the main function of an artificial intelligence technique through another intelligent processing method to increase the execution speed and system reliability. The following main functions can be considered: building the knowledge base on expert systems, changing weights in neural networks, or crossing the genetic algorithms. For example, implementing the weighting mechanism in a BP(backpropagation) algorithm using genetic algorithm operators implies that the baseline weights are changed with mutation and crossing operators. The performance of this update method is in some cases better than the BP(backpropagation) algorithms, the number of iterations for achieving convergence towards the good solution being significantly lower.

Another example of hybrid with function replacement is the use of a genetic algorithm that defines fuzzy sets of affiliation. These definitions are usually provided by a subjective reasoning of an expert, which can lead to errors and require careful experimental checks.

Intelligent Hybrid Systems are a set of independent modules for intelligent processing that perform distinct functions but exchange information to generate global solutions. When the problem can be divided into several tasks, each one can be solved using the most appropriate technique. The global solution is achieved through cooperation, based on a control mechanism. For example, if the three tasks of the problem are model recognition, serial reasoning and optimization, then a neural network, an expert system and a genetic algorithm can be chosen for the solution. It can work either sequentially (the control mechanism is made up of rules of the expert system) or in a cooperative competitive environment. Literature signals such achievements, especially in systems for diagnosis.

Intelligent hybrid polymorphic systems use a single functional processing architecture and various intelligent processing techniques. Achieving multifunctionality is obtained by emulating various processing techniques. Unfortunately, research in this area is not convincing. There are a small number of examples of polymorphic hybrids consisting of neural networks used for symbolic reasoning or imitation of genetic search procedures.Alte aplicații ale inteligenței artificiale:

Automatic learning (machine learning). An intelligent system is able to learn to improve its interaction with the environment. He can either learn from mistakes, self-improvement, or be guided by a teacher, either generalizing, or by analogy, or by positive and negative examples. The field of automatic learning formalizes these methods and seeks their application to automated systems.

Automated reasoning and theorem proving (problem solving). The oldest branch of artificial intelligence and the theoretical one of them, uses logic as a formal system for finding demonstrations and generating automatic inferences. Many problems can be formalized as mathematical or logical problems, solving them by reducing to a theorem, starting from an axiom system and using a logical deduction mechanism. As in other cases, the difficulties lie in the extremely high number of opportunities that may arise. To control the explosion of possibilities in search of a solution, heuristic solutions are often used which, although not guaranteeing the optimal solution, can generally lead to the finding of one, at least.

Natural language processing. It is widely accepted that the use of language is a defining feature of intelligence. Although it is so easy for us to communicate through language, an attempt to explain the mechanisms underlying the understanding of the texts or the spoken language proves to be particularly difficult. In the field of understanding natural language, a distinction must first be made between understanding a message communicated by voice and understanding a written message. The first issue deals with the field of speech interpretation (speech processing). The problem of understanding the written language is the subject of two types of concerns. Computational linguistics, on the one hand, as a purely academic domain, discusses computational models of natural language in order to investigate and discover the very nature of human language and human cognitive abilities. Linguistic Engineering, on the other hand, is concerned with developing applications based on the use of natural language, oriented towards industry, commerce, social or educational spheres.

Computer vision. Five senses (sight, hearing, touch, smell, and taste) are at our disposal to interact with the environment to get to know what is outside our body, in our immediate neighborhood. The modeling of these senses on automated systems is certainly a matter of great importance in the field of artificial intelligence. Of the five, the sight occupies a privileged place. Artificial vision systems attempt to decipher the mechanisms of vision and interpretation of static and moving images.

1.6 The Advantages and Limits of Artificial Intelligence

Creating an intelligent machine to mimic the complex performance of human behavior is a real challenge because of the lack of human ability to fully understand the processing mechanism, that is, the power of the human brain. The advantages and disadvantages of an intelligent machine can best be emphasized by making a comparative analysis between human behavior and intelligent systems as can be seen in Table 2 and Table 3, which synthesizes some significant differences.

Table 2 - Advantages of intelligent equipment

Characteristics	**Human ability**	**Machine ability**
Knowledge	Perishable	Permanent
Reasoning	Inconsistent	Consistent
Expertise	high value in monetary units	average value in monetary units
Humans	Mobile	Static
Processing ability	Inconsistent	Highly consistent
Resistance	Limited	Unlimited
Expertise	Restricted	Extensive
Thinking	Mortal	Immortal

Table 3 - Disadvantages of intelligent equipment

Characteristics	**Human ability**	**Machine ability**
Knowledge	Evolutionary	Static
Reasoning	Unlimited	Limited
Expertise	Adaptable	Rigid
Humans	Receptive	Unreceptive
Processing ability	Multiple	Singular
Thinking	Conscious	Unconsciousness
Expertise	Creative	Without inspiration

Among the advantages of artificial intelligence, we mention:

- artificial intelligence is able to solve complex problems;
- computers become more useful when incorporating techniques of artificial intelligence;
- benefits are gained through increased productivity, acquisition of expertise and more secure work environments for users;
- artificial intelligence is not perishable;
- artificial intelligence is cheaper, more solid and more consistent than natural intelligence; Artificial intelligence can be documented by "drawn";
- Artificial intelligence is easy to reproduce.

The limitations or disadvantages of artificial intelligence are:

- is not as creative as natural intelligence;
- can not use the human-specific direct sensory experience;
- is performant only in symbolic processing;
- can not make associations between the qualities of objects, events and processes with the same ease as humans.

References:

Andone, I., Robert J. Mockler, Dorothy G. Dologite, Alexandru Al. Tugui, Dezvoltarea sistemelor inteligente în economie: metodologie și studii de caz, Ed. Economica, Bucuresti 2001

Wolfgang Ertel, Iontroduction to Artificial Intelligence, Springer, London 2011

M. Tim Jones, Artificial Intelligence. A system Approach, Infinity Science Press, Hinghan 2008

Xuan F. Zha, Artificial Intelligence and Integrated Intelligent Information Systems: Emerging Technologies and Applications, Idea Group, London 2007

http://www.creeaza.com/tehnologie/electronica-electricitate/Sisteme-inteligente-hibride577.php

Chapter 2 - Applications of artificial intelligence

2.1. Rules-Based Systems (RBS)

Rule-based systems are the most known intelligent systems, the most numerous and formerly referred to as expert systems.

The following list of definitions had been depicted from the literature for understanding the concepts governing the functionality of Expert Systems, now known as Rules-Based Systems:

• Intelligence - represents the profound, easy understanding of things, especially in the field of culture and science; is the ability to understand, to comprehend phenomena, things, etc.

• Understanding - is the activity of thought that reveals the links between objects and phenomena. In elementary form, understanding is included in the perception process itself. In a more complex form, understanding is involved in discovering the links between cause and effect, the significance of an artistic or scientific work, the motives of human conduct, etc.

• Understanding is mainly involved in the problem-solving process. It is generally based on past experience and its use in a new situation.

• The Expert System - is a form of artificial intelligence. It is designed to reproduce the technique of solving the problem, similar to an expert, in a narrow area of specialization where reasoning is much more justified than the calculation. In specialized literature, expert systems are called intelligent systems, rule-based systems or rule engines.

• Reasoning - is the ability to conceive, or attempt to reach a conclusion based on valid or invalid premises.

• Expert - has expertise in a field and is a qualified person with superior knowledge in a particular field.

2.1.1 Basic concepts and evolutionary milestones

Basic concepts

Expertise is an intensive knowledge, specific to the field of the problem, acquired through training, reading or long experience.

The following types of knowledge are included in the expertise:

1) facts about the problem domain;
2) problem theories;
3) rules and procedures on the subject matter;

4) rules or heuristics about what should be done in a given problematic situation, to solve the problem;

5) global strategies for solving types of problems;

6) metaknowledge.

Experts are difficult to define because they can be people with multiple levels or degrees of expertise. The question is how much expertise does a qualified person have in a field before being considered an expert? Human expertise includes a broad range of expert activities, namely:

1) recognition and formulation of the problem;

2) solving the problem with accuracy and speed;

3) explaining the solution;

4) learning from experience;

5) restructuring of knowledge;

6) fragmentation of rules;

7) determination of relevance;

8) consciousness of its limits.

Some attempts to define the human expert:

1) A person who has knowledge or professional competence in a field.

2) A person recognized for the knowledge and judgments that posess, which also gives public authority. The expert has a long, intense experience, practice or education in a particular field. The person recognized by a professional group or a professional company through certification.

3) A person with a qualification or superior knowledge in a particular field.

All this indicates the ability of experts to transform the data of an arbitrary problem into a form that leads to a quick solution. This is possible due to the ability to learn new things from experience, to fragment the known rules, to determine the relevance of knowledge and its limitation in the field. All these activities need to be deployed efficiently (fast and cost-effectively) and effective (with high quality results). To imitate a human expert, it is necessary to build a system with all these characteristics.

There are two reasons for building an Expert System: replacing an expert(1) and assisting an expert in his work(2).

The main reasons for replacing the human expert are:

- make expertise possible at any time
- automating a routine task that requires an expert
- the expert is very expensive
- expertise is required in environments dangerous to human health.

Assisting human experts by specialized programs is more and more common in large companies and not only. The main reasons for developing expert systems capable of assisting the experts in their work are:

- assists the expert in performing routine tasks to improve his / her productivity
- assists the expert in executing difficult tasks in order to better control complexity
- makes the expert available to information difficult to obtain at the right time.

Over time, the concept of expert system/ rule-based system / rule engine / has come up with a multitude of definitions. Here are just a few of those that outline the most complete features of these systems.

Edward Feigenbaum, from Stanford University, states that "expert systems are designed to solve problems that typically require considerable human expertise." The expert system is a "smart computer program that uses knowledge and problem-solving inferential procedures that are challenging enough to require significant human expertise to solve them."

J. Giarratano and G. Riley (NASA): "An expert system is a system that emulates the ability to make decisions by the human expert. The term “emulates” means that the system is meant to act in every respect as the human expert.

Elaine Rich and Kevin Knight say that "Expert systems are programs that solve problems normally solved by human experts. In order to do so, they require access to a knowledge base, they have to offer different reasoning environments and justify their conclusions".

EM Awad states that "An expert system consists of computer programs that emulate or clone the reasoning of a human expert in the field."

The basic features of the rule-based systems are:

Transfer of expertise is in fact the objective of a rule-based system. The transfer takes place from the human expert to the computer and from here to the expert or nonexpert users. This process involves four activities:

1. acquiring expertise from experts and / or other sources;
2. representing knowledge in the computer;
3. inference based on stored knowledge;
4. transferring knowledge to the user.

Inference (inference process) is a unique, but major feature of the rule-based system, it is in fact the ability to reason. Based on the expertise stored in the knowledge base and the program that can access a database or file, the computer is programmed to make inferences. Inferences are executed by the component called inference engine that has procedures, posess procedural knowledge in relation to problem solving.

Explanatory capacity is a major feature of rule-based systems, in connection with the explanation of the advice they give or the recommendations made for decisional alternatives (see Table 4).

Table 4 – Comparative Situation on Human Expert and Expert System

Criteria	Human Expert	Expert System
Availability	Only on business days	Anytime
Spacial localization	Only local	Anywhere, especially in environments dangerous for human beings
Information security	Unreplaceable	Replaceable
Perishable	Yes	No
Performance	Variable	Permanent consistance
Speed	Variabile	Fast
Cost	High	Acceptable

Here are some of the successive ideas that outline the notion of an expert system:

- from a conceptual point of view, the expert systems aim at reconstituting the reasoning based on the expertise obtained from the human experts;

- expert systems have the knowledge and ability to carry out human intellectual activities;

- expert systems are organized for the acquisition and exploitation of knowledge from a particular domain called the field of the problem;

- expert systems have methods of invoking knowledge and expressing expertise, behaving as a "smart assistant";

- as an automatization accomplishment, the expert systems are based on the principle of separating knowledge (the knowledge base) from the program that deals with it (the inference engine);

- expert systems are capable of memorizing knowledge, establishing links between knowledge, and infering conclusions, solutions, recommendations, advices, and the causes of phenomena and situations based on facts and the processing of uncertain knowledge.

Evolutionary milestones

The existence of expert systems over time is quite recent. The implementation and development of this type of computer systems is due to the desire of researchers in the field of artificial intelligence to create a computer program that can reproduce human thinking and reasoning in solving problems in a similar way to the human one.

The first approach of expert systems was made by three groups of American researchers. Namely:

- Group 1: Newell, Shaw and Simon who developed the General Problem Solver;
- Group 2: Minsky and McCarthy, who, as well as their predecessors, have had significant expertise in the field of expert systems in the 1960s;
- Group 3: made up of Lederberg, Buchanan and Feigenbaum who worked on the DENDRAL program.

The activity of these three groups of scholars has been somewhat successful, but according to the tests carried out, they noticed that systems were performing better only if they were designed to solve problems in a single field.

In the early 1970s, researchers were convinced of the inefficiency of expert systems in solving problems in vast research areas, and found that expert systems technology should be applied to build intelligent applications in well-defined and narrow domains. This, together with technological advances in hardware and software, has led to the crystallization of expert systems technology in the late 1970s and early 1980s, when expert systems managed to migrate from the research laboratories into the commercial environment. This is why some authors understand through the history of expert systems the development of intelligent systems from solving universal problems to specific problems.

The pioneer of intelligent programs was MACSYMA, an algebraic expression simplifier, followed by DENDRAL that identifies molecules based on spectrograms. They have proven their superiority in a specific area of knowledge.

Of all these pioneering programs, the most important is MYCIN, which has been used to diagnose and treat blood infections. These systems that emerged in the mid-1970s are part of the first generation. In the early 1980s, commercial applications of expert systems technology appeared. These include XCON, XSEL, CATS-1 and others and, with these, there are also programs to help accelerate the construction of intelligent systems such as EMYCIN and AGE, for learning from experience such as METADENDRAL and EURISTO or for acquiring knowledge such as EXPERT and KAS.

Some examples of platforms that allow expert systems to be developed are:

- Programming languages: LISP, CLIPS
- Expert Systems Shells: Exsys, Corvid
- Rule based systems:
    - pentru Java: Jess, Drools
    - alte: OPS5
- Web Rule languages family:
    - RuleML and SWRL
    - Interoperability of rules

2.1.2 Architecture of expert systems

The most common structure of expert system is that described in Figure 3. Communication between the system and the user or the human expert is accomplished through a specially designed interface. The other important components of an expert system are the knowledge base, the inference engine, the explanation system, and the knowledge base publisher. In the following paragraphs, each of these components will be briefly described.

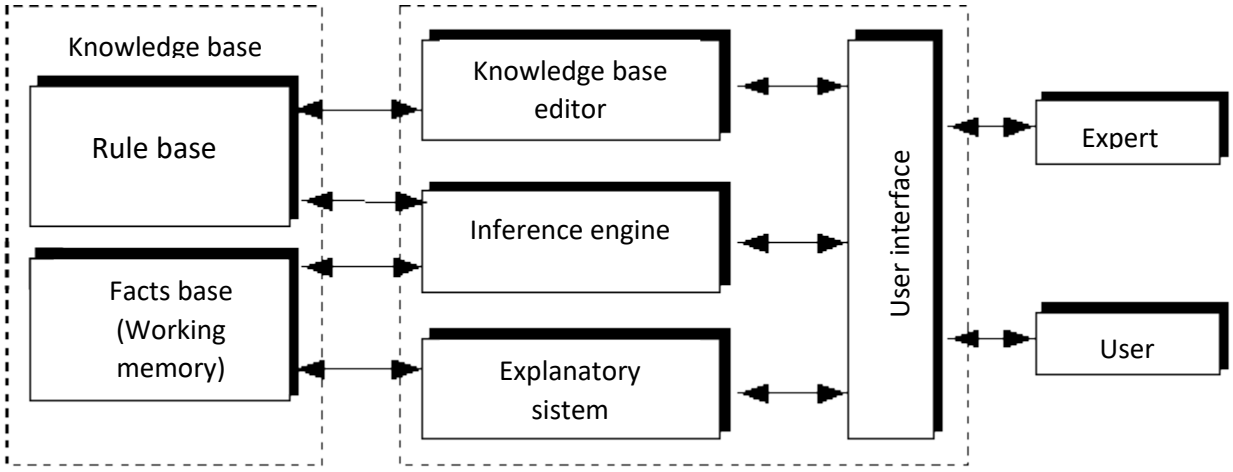


Figure 3 Architecture of expert systems

User Interface. Communication between the operator and the expert system is accomplished through a specialized interface that can use the menus that we are familiar with in Windows applications, dialogue in a natural language or any other form of interaction. In order to simplify communication, it is desirable that the user interface be as friendly and intelligent as possible, respectivley to know how to present the information and to be aware of the user's preferences.

Knowledge base. At the level of an expert system, the knowledge base has two components: the rules base and the facts base. The rule base contains the bulk of the knowledge needed to solve the problems. The more complex expert system uses rule-based rules with a large number of rules ranging from a few hundred to a few thousand. The rule base is the long-term memory of the system because it contains the hard part of the knowledge used by the sistem expert.

On the other part is the short-term memory a.k.a. work memory, which consists of the facts base, and describes the context of the problem and contains the input, output and any intermediate data produced as a result of inferencing by the inference engine.

The represention and organization of the knowledge base are two essential aspects for the proper functioning of the expert system. If the expert system should be developed furthermore after its completion, it is absolutely necessary that the knowledge base to be completely separated from the rest of the system.

The inference engine. The specific mechanism that simulates the human expert's reasoning is inference. According to the dictionary definition (DEX), inference is the logical operation of passing from one statement to another and in which the last statement is deduced from the previous one.

The inference engine controls the way and the sequence in which the knowledge base is applied to the data in the fact base. In fact, the inference engine is a computational program that applies rules to facts in order to generate by induction either new facts that add to the facts base, either confirming or invalidating a hypothesis, or actually addressing the solution of the problem.

In principle, the inference engine contains an interpreter, which analyzes and processes rules-based rules and a scheduler that determines the order in which the rules apply. The inference engine goes through the rules, seeking to identify a correspondence between the facts of the conditions or the consequences of the rules and the information in the facts. When such correspondence is identified, that rule is used to produce a new fact or to confirm a hypothesis.

At the planner level, selecting the rule that applies at one point uses one of the following strategies:

- The selection of the most specialized rule. Of the two rules X and Y, will be selected the one containing the maximum number of conditions, considered to be specialized, with respect to the other, which is general.
- Selecting the most productive rule. Among two rules X and Y, the one containing the maximum number of consequences is selected, assuming that "more is better".
- Heuristic selection. Between many rules that can be applied at a time, it is selected the one that drives the facts base as close as possible to the desired state.
- Select based on trust. Some of the rules are given increased confidence, and when selecting the rule that applies at a certain point, priority is given to the certainty factor wthin the rules.

Knowledge Base Editor. Some expert systems are provided with a knowledge base editor that helps the user, expert or knowledge engineer to update and verify the content of the knowledge base and, in particular, the content of the rule base. The editor's existence also ensures a comfortable development of the system after its deployment.

Expert system functioning

The strength of an expert system is its ability to carry out inferences and to draw conclusions based on premises. In fact, this capability provides intelligence to the expert system. The inference engine task is to deduce some conclusions or verify some assumptions by agreeing the facts-based data with rules-based as illustrated in Figure 4. Fulfilling this task is possible by applying two types of strategies: forward chaining and backward chaining.

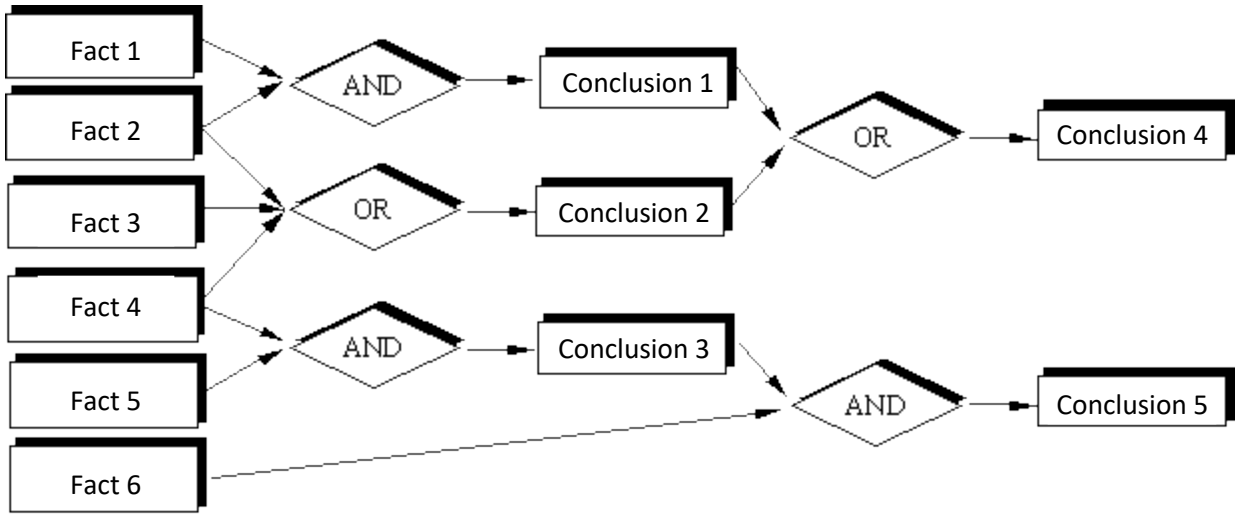


Figure 4 The inference development principle within a rule-based expert system.

In the case of forward chaining, the inference engine examines the current state of the knowledge base, following the identification of the rules whose premises are satisfied by the facts in the fact base. These rules are applied and the resulting facts, as well as their conclusions, are added to the facts. Further, the knowledge base is re-examined and the process described above is repeated until a final conclusion is reached, which may or may not be a solution to the problem. This process is illustrated in Figure 5.

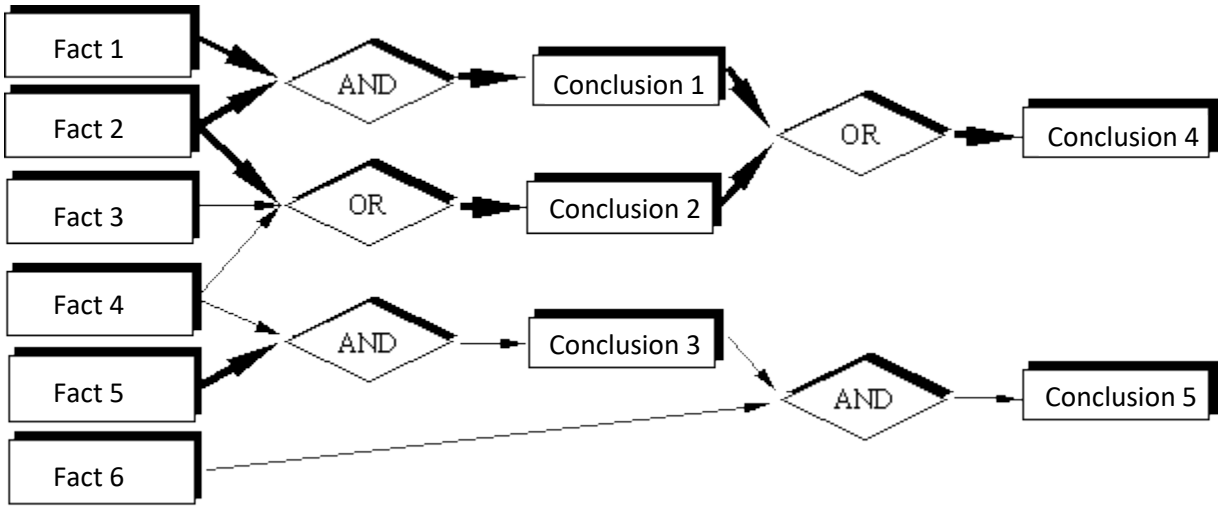


Figure 5 Inference engine functioning according with the forward chaining strategy (with the bold line representing the true facts, and with the straight line, the false ones).

Backward chaining is a purpose-based strategy in the sense of trying to confirm the conclusion of a rule (seen as a goal), demonstrating the validity of all the premises of this rule. These prerequisites can also be the conclusions of other rules (in which case the inferential engine will enter a recursive process, trying to prove the validity of the premise of those rules) or may represent independent facts provided as input to the facts. The pursuit of such a search process is illustrated in Figure 6.

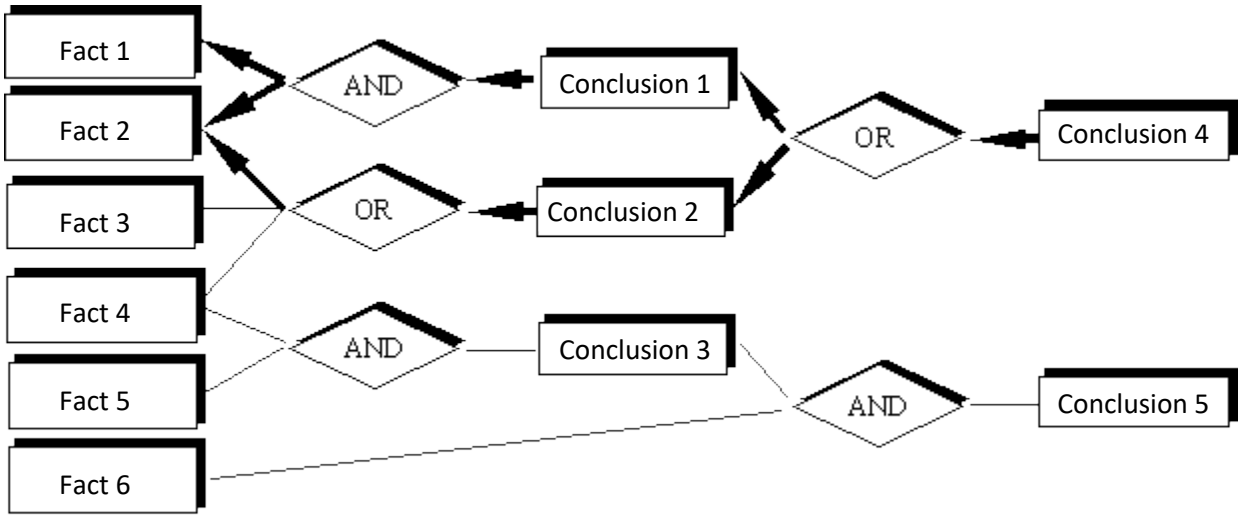


Figure 6 Inference engine operation after backward chaining strategy. Verifying the hypothesis associated with the conclusion 4.

Inferencing by applying forward or backward chaining is equivalent to browsing in one way or another of the decision graph in Figures 5 and 6, by applying one of the known search strategies. On the other hand, it is noted that, in relation to the types of strategies defined above, data-based strategies use forward chaining, while goal-based strategies use backward chaining.

The functioning of an expert system can take place in three ways: (i) knowledge acquisition mode; (ii) consultation mode and (iii) explanation mode.

The expert system is in consultation mode when it is in dialogue with the user to determine a solution to a given problem. The user sends data to the expert system describing the problem, and the system responds using the inference engine to model / simulate reasoning to deduce answers for the user questions. The consultation mode involves the large-scale use of the inference engine, with either of the two inferential mechanisms being applied - forward or backward chaining.

One of the most powerful attributes of an expert system is the ability to explain the inferences it carries out. This capability is based on AND / OR trees that are built during the inference process. Explanation mode provides the user with expert knowledge in an explicit way. In this way, the system explains how it came to a certain conclusion, why it did a certain action or provide answers to a question of "what if?" For example, Figures 7 and 8 show the principle diagrams after which the system answers the questions how and why.

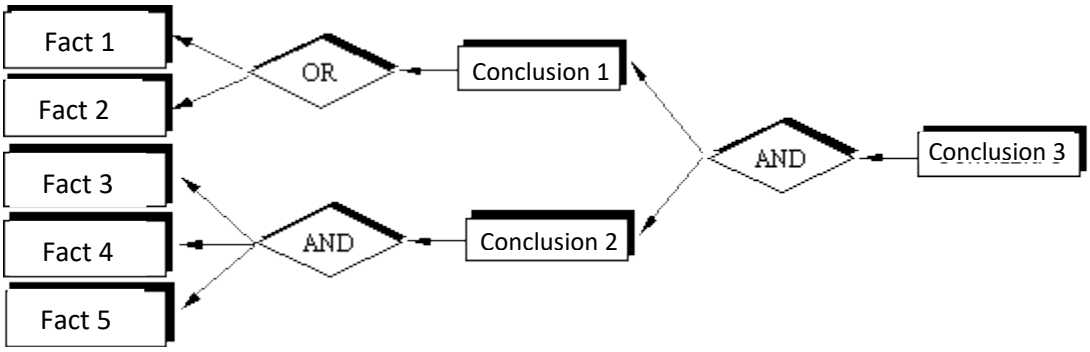


Figure 7 Explaining how a conclusion has been reached.

Thus, in order to explain how a conclusion has been reached, the AND / OR decision tree is browsed in reverse. For example (Figure 7), Conclusion 3 was established as a result of Conclusion 1 and Conclusion 2. In its turn, Conclusion 1 was established because one of Facts 1 or 2 are true.

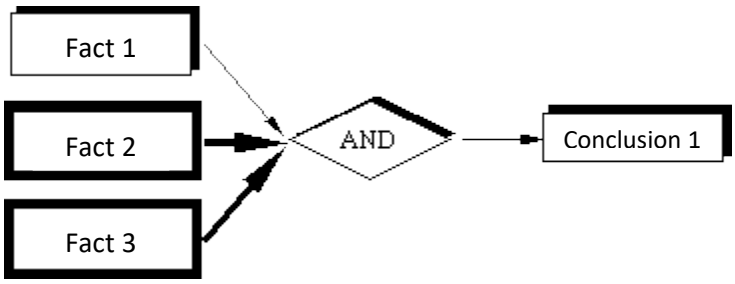


Figure 8 Explaining why a particular action is being taken.

In the case of the scheme in Figure 8, in order to answer why a particular action is being taken, the system must first identify the current objective and justify this objective by checking the conditions leading to its achievement. Thus, if the Facts 2 and 3 are verified, the system may question the user about the validity of Fact 1 in order to determine the validity of Conclusion 1.

Production rules method

Production rules method is a procedural method. The declarative methods include semantic networks, frames/ structured objects and predicate logic (predicate calculus). This is the most commonly used method of developing rule-based systems because it has proven its effectiveness in representing recommendations, guidelines, or strategies when the knowledge to represent relates to the problem-solving experience of a particular field.

The representation of knowledge ensures their formalization and organization. The most commonly used knowledge representation scheme is the production rule, sometimes referred to as the IF - THEN rule. Production rules describe the aptitude and heuristic knowledge currently used by human experts. The set of these rules forms the rule base of an expert system, sometimes referred to as a knowledge base.

A production rule contains a condition or premise, followed by an action or conclusion and takes the form IF <condition> THEN <action>. In general, the part of a rule's action may include: (i) actions with

descriptive effect, eg displaying a message on the monitor screen; (ii) verifying another rule for the case of chaining rules systems, and (iii) adding a new fact to the facts base of the expert system.

A production rule is composed of two parts that establish a relationship that represents the purpose of the rule:

- the IF part (LHS: left-hand-side) and
- THEN part (RHS: right-hand-side)

The general syntax of a production rule is as follows:

IF

condition1 □
condition2 □
....

premises

□ Logic operator (and, or, etc)

THEN

conclusion1 □
conclusion2 □
....

action or consequence

ELSE

conclusion'1 □
conclusion'2 □
.....

The typology of production rules takes into account the great variety of forms of the knowledge to be represented:

- Causality rules
- Recommendation rules
- Directive rules
- Strategic rules
- Heuristic rules
- Rules with variables
- Rules with certainty factors (0-1, -1 - 1, 0-10).

Production rules are a simple way to specify problems and solutions. Unlike coding performed through methods and / or functions, the rules are written in a less complex language; system analysts can quickly verify / track / modify the rules.

The data resides at the level of the business domain, and the logical level is represented by rules. Depending on the type of project / application / system, this separation can be very advantageous.

Drools implementations are scalable, which means that if there are frequent changes to the rules, they can be added without necessarily changing the existing rules.

The rules create a knowledge base that is executable. The knowledge base is a "single point of truth" of a business process. One can have a knowledge base that contains rules on setting discounts, another knowledge base for setting penalties, another knowledge base for cash-flow control, another for determining the need for materials, etc. Ideally, the rules are so easy to understand that they can serve as the source of documentation.

2.1.3. Advantages and disadvantages of using RBS

The main advantages obtained through the use of Rules-Based Systems are:

- easy maintenance;
- easy development;
- portability;
- easy documentation.

Expert systems also have many indirect, non-quantifiable benefits. In all cases, getting benefits is the only critical restriction for introducing expert systems.

Expert systems are a technology like any other, and it is understandable the interest of economic agents in gaining commercial advantages and advancing in competition due to its use. Expert systems technology helps businesses master market changes and occupy a dominant position in the second millennium.

A particular advantage is their relatively low cost, which is quickly depreciated if the frequency of use of expert systems increases. But attention, expert systems are not a panacea, they are profitable only when they are really the best solution to the problem in the field of competence.Among the limits of using RBS we find the folowing:

- knowledge (expertise) is not always readily available;
- experts have limited knowledge on the use of technology;
- the lack of end-user trust is sometimes an additional barrier to using RBS;
- many experts do not have independent means to control situations when their conclusions are unreasonable;
- the vocabulary used by experts is strictly delimited and often difficult to understand by developers;
- the transfer of expertise is still a delicate subject, due to the legal base and the perception of many people.

2.2 Robotics

Living intelligent systems have the ability to find solutions towards navigating in ouuter space, to reach or move objects using their arms. Finding a road on rough terrain (lunar, for example, or on the bottom of the oceans) or in an space that contains obstacles (such as a room) is not a simple matter, not least because of the large number of choices available at each step, complexity that makes impossible the quest for exhaustive search. By trying to model the cognitive processes that take place in living systems, robot planning is a research field that aims to improve the behavior of robots when they are programmed to execute different tasks.

2.3 Intelligent software agents

Intelligent Software Agent is an autonomous IT entity possessing data / knowledge and programs / procedures capable of interacting with other entities by communicating through messages or sharing information.

According to Wooldridge and Jennings (1995), a smart agent is a hardware system but especially software that contains the following properties:

- autonomy
- reactivity
- proactivity
- social ability

The decision-making process of an agent is influenced by the properties of the execution environment. This can be:

- accessible or inaccessible: In an accessible environment, the agent can obtain complete information about the current state of the environment
- deterministic or non-deterministic: in a deterministic environment, any action has only one effect, ie there are no uncertainties about the environment in which the environment will pass after the action
- episodic or none-episodic: In an episodic environment, agent performance depends on a number of discrete episodes, without a link between the performance of the agent in different episodes
- static or dynamic: A static environment remains unchanged except for the effects of the agent's actions.

In short, a smart agent:

- has memory and the ability to work in the field or using memory

- is equipped with sensors to perceive information from its environment
- has action devices created to interact with the outside world
- has the ability to test / check various actions. Thus, it is able to choose the best possible option.
- holds internal memory for its methods and explores the field for which it was developed being guided by knowledge stored in internal memory.

Two agents, under different circumstances, may try to execute the same action, but their behavior may be different. The characteristics of ideal, behavior-independent actions are:

- infallibility: will certainly produce the desired effects when the environment meets its preconditions and is executed correctly
- utility: the usefulness of an action is given by the usefulness of the state in which the action is attained
- cost: different actions may require different resources to be met, such as electricity consumption, spending money, etc.

The above mentioned properties refer to intrinsic purposes, which are not subgoals of another goal already considered, and which can therefore be considered as high level goals:

- autonomous generating goals
- accomplishment goals
- singular or multiple goals
- commitement goals
- purposefulness goals
- meta- goals

A taxonomy attempt for intelligent agents was performed by Keil in 1989 and is presented in Figure 9. Other authors (Caglayan and Harrison, 1997) propose a classification according to the computational environment of the execution area, setting a difference between desktop agents, Internet and Intranet.

Desktop Agents: they function on a PC or workstation and are software agents that are executed locally on the operating system of a personal computer.

Internet Agents: they have emerged from the need to process the increasing amount of information.

Intranet Agents: are software agents located on a corporate server that supervises and manages business on behalf of users.

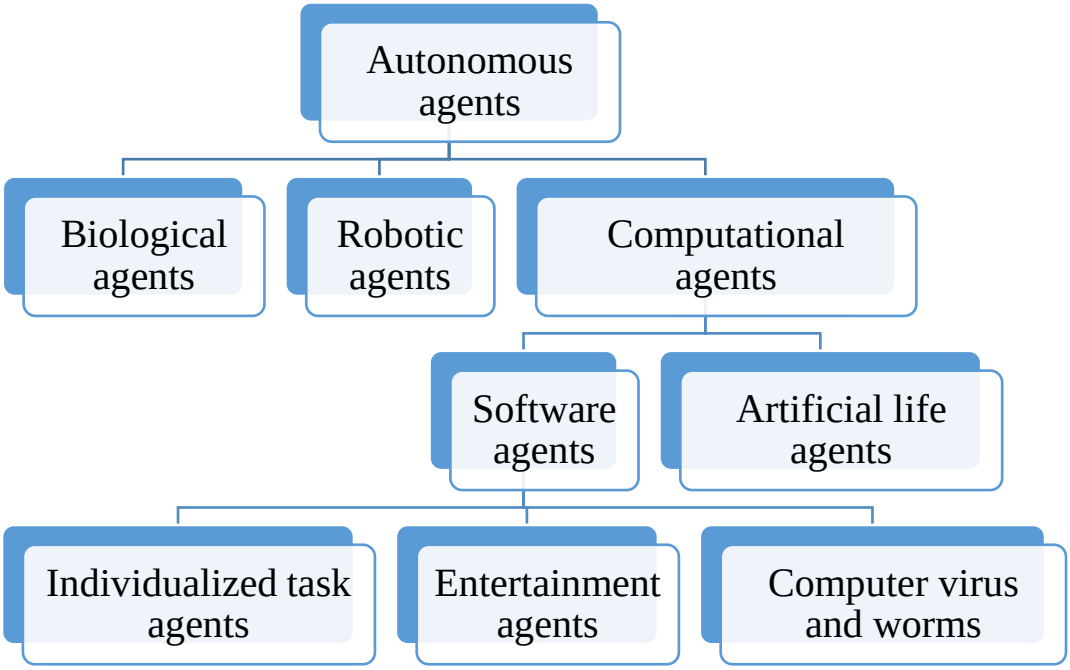


Figure 9 – Agents taxanomy

Intelligent agent development architectures:

- Logic based architectures - where the decision-making mechanism is based on logical deductions;
- Reactive architectures - where implementation of the decision-making process is based on assigning actions to various situations (states);
- Belief-Wishes-Intentions Architectures - where the decision-making process depends on the manipulation of data structures representing the views, requests, and intentions of the agent;
- Layered architecture - the decision-making process is done over several software levels.

In conclusion, we affirm that agents are a smart, functional, flexible solution (relatively independent and adaptable to different platforms and communication systems) to address the ever-increasing needs of a large number of users.

2.4 Fuzzy Sistems /Fuzzy Logic

Fuzzy logic is a generalization of binary logic that extends the concept of value of truth to the partial value of truth. Because it is based on imprecise, ambiguous or vague information, the new type of logic is actually an approximate logic, but it has the great merit of fidel emulation of how people make decisions: thus, based on approximate and imprecise information, precise solutions can be established.

Fuzzy logic is commonly used in automation-specific applications where traditional, often laborious, mathematical models can be replaced by a set of fuzzy rules. They describe the transition from the state desxribed by the input values to the ending state which is reached based on the output values. Such systems are called fuzzy systems.

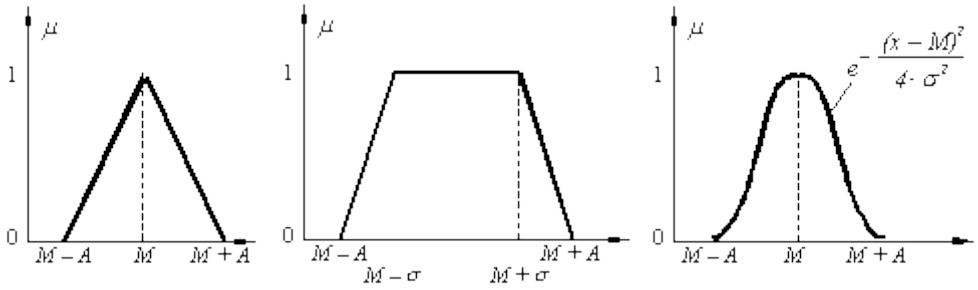
The development of any application that uses fuzzy logic requires three main steps, namely:

- Fuzzification, in which the input values are associated with functions selected according to certain criteria, and based on them, the linguistic values corresponding to the inputs are defined, and they move from the Crisp to the Fuzzy representation.
- Applying the set of fuzzy rules in order to determine the degree of truth for each of the pre-built rules, either based on the knowledge of human experts or on the basis of special techniques for extracting these rules.
- Defuzzification, which uses the degrees of truth established in the previous step for each rule and applies them to their consequences to return from the domain specific to fuzzy quantities in the domain associated with the variable or output variables.

The following paragraphs describe the main features of each of the three mentioned steps.

Fuzzification

In most practical applications of fuzzy systems the input sizes are described by crisp values (possibly characterized by certain imprecisions), defined on a certain domain of the analysis. On the other hand, the output sizes to be obtained describe values proportional to the degrees of fuzzy truth associated with the applicable rules. In this context, fuzzification consists in the process in which the linguistic categories describing the domain of values, the membership functions associated with these values and the degrees of belonging of the inputs to the respective linguistic categories.



(a) (b) (c)

Figure 10 Triangular (a), trapezoidal (b) or Gaussian (c) function.

An important and sensitive component of the fuzzy phase is the building of membership functions. This problem can be addressed in a traditional way or using elements of artificial intelligence, such as neuronal approximations classifiers or evolutionary calculation. The construction of traditional membership functions uses the knowledge of human experts and reasoning in the form presented in Figure 10.

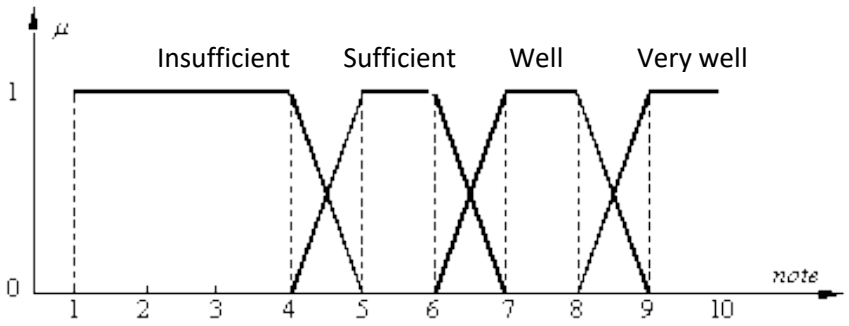


Figure 11 Membership function

Defuzzification

The use of fuzzy logic requires that input values, usually in crisp form, undergo a fuzzy process. The next step is to evaluate the rules governing the fuzzy model system, and the result of these evaluations is also fuzzy (with the exception of Takagi - Sugeno - Kang systems), ie it is presented in the form of a set of values that can be simultaneously assigned to the output value. In practice, however, traditional systems use exact, crisp values. Thus, at the end of any set of fuzzy inferences, it is necessary to apply an inverse, defuzzification process that produces a crisp value based on the fuzzy set describing the result.

The most common method of defuzzification is finding the center of gravity; other methods sometimes used in studying fuzzy systems are height method, the average mean method, or maximum area method. The literature also mentions other alternative methods.

2.5 Artificial Neural Systems

An artificial neural network or system is a mathematical or computational model that is inspired by the structure and functional aspects of a biological neural network. A neural network consists of a group of interconnected artificial neurons arranged on layers. Usually, a connection from one neuron to another has associated a weight.

In most cases, an artificial neural network is an adaptive system that changes its structure according to the information inside or outside that is transmitted through the network during the learning phase.

In the current implementations of neural networks, the biology-inspired approach has largely been abandoned instead of a more practical approach based on statistics and signal processing.

Neural Network Based Methods provide a robust approach to approximating real, discrete and vector functions. For some types of problems, such as learning the interpretation of complex real data obtained from sensors, artificial neural networks are among the most effective known methods. For example, feed-forward backpropagation neural networks, which will be discussed in detail, have proven to be surprisingly good for practical problems such as: autonomous driving, handwriting recognition, robot control, face recognition.

A network neuron has the following form (Figure 12):

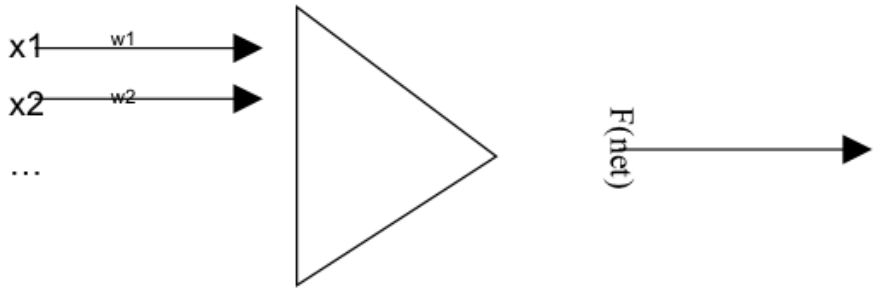


Figure 12 Structure of a neuron

where *x1, x2, ... , xn* are signals received by a neuron, *w1, w2, ... , wn* are the costs or weights for each signal, $net = \sum_{i=1}^{n} w_i x_i$, and *f(net)* is a function that calculates the output from the neuron (activation function). IF *f(net)* is greater than a set threshold for the network, then the neuron sends the signal further.

There are many approaches and algorithms for networking. These include feed-forward backpropagation networks, radial-base networks, recurrent networks (the graph of neurons and synapses also contains cycles), counterpropagation networks. Among the simplest and most used neural networks are those based on the feed-forward with backpropagation algorithm, or short backpropagation. These networks have a classical structure with a layer of input neurons, one exit and several layers hidden between them.

The backpropagation algorithm is divided into 2 phases: the first is to disseminate useful information from the input layer to the output layer, then propagate in the reverse direction of the errors and the second phase consists in updating the weights.

- In the first step, the input data propagates forward over the network to generate the output values, and then the error is calculated and propagated back to the hidden layers and to the input layer to generate the (delta) differences between the obtained values and the actual values for each neuron.
- In the second phase, for each weight, the error gradient is calculated using the delta generated values in the first phase, and updating the weight in the opposite direction to the error decreasing the weight multiplied by a subunit constant called the learning coefficient.

The single hidden layer algorithm looks like this:

```
Initializes network weights // usually random
Execute
  For each example, it's in the training set
```

```
O = output-value(network, e) // scroll forward
T = real value for e
Calculate the eror (T - O) for output neurons
Calculate delta_wh for all weights from the hidden layer
          to the output  // scroll backward
Calculate delta_wi for all weights of the input layer
          To the hidden // continue scroll backward
Updates network weights
Until all examples have been learned correctly or a stop criterion
           has been met,
Return network
```

2.6 Genetic algoritms

Genetic algorithms are inspired by Darwin's evolutionist theory. The idea of creating evolutionary algorithms was introduced in 1960 by Rechenberg I. in his work "Evolution strategies".

Genetic algorithms or Genetic programming is a method based on evolutionary algorithms, inspired by biological evolution, seeking expressions or algorithms that would best solve a particular problem. Structures that are subject to adaptation and evolution in the genetic programming paradigm are hierarchically structured programs whose size, shape and complexity can change throughout the process.

Genetic programming is an automatic learning technique used to optimize a population of programs in accordance with a fitness function determined by the ability of the program to achieve the required task.

Programs handled by genetic programming algorithms are represented by expression trees. Each function call is represented by an internal node in the tree and the arguments of the function are given by the child nodes of the main node. To apply genetic programming to a particular domain, it must be defined a set of primitive functions or operators to work with (ie: sin, cos, +, -, or, and, xor), as well as a set of terminal nodes represented by variables (program input data) or constants.

Algorithms based on genetic programming have the following form:

1) Randomly creates an initial population of individual programs made up of available functions and terminals.
2) It iteratively performs the following steps (one generation) on the population until the stop criterion is satisfied:
    a. Executes every program in the population and calculates its fitness using the defined fitness function.
    b. Choose one or two programs from the population with a fitness-based probability to

participate in the genetic operations of the point c).

c. Creates a new program to be included in the population by applying the following operations to the specified probabilities:
    i. Reproduction: Copy the selected program to the new population.
    ii. Crossover: Creates a new program for the new population, recombining random chunks of two selected programs.
    iii. Mutation: Creates a new program for the new population, modifying a random part of a selected program.

Once the stop criterion has been satisfied, the program with the best result for the fitness function is chosen and returned by the algorithm.

Among the applications of genetic algorithms we find intelligent systems based on genetic algorithms. These systems proved to be performant in:

- problems of searching and identifying specific structures and relationships within databases and voluminous knowledge bases (data mining).
- optimization issues related to staff selection and selection of business portfolios.
- financial affairs, securities trading, credit assessment, fraud detection and bankruptcy prediction.

The mechanism specific to these systems is inspired by the functioning of biological systems in that it encourages candidate solutions capable of solving a problem and penalizing unsuccessful solutions. In this way, after many generations, there are obtained very good solutions for complex optimization problems with a large number of parameters. The basic idea of a genetic algorithm is to start with a population of solutions, each performing more than the previous ones.

The phases of the cycle through which a genetic algorithm operates are:

1. creating a population of "members" (candidate solutions to solve a problem)

2. the selection of the members that best adapted to the needs of the problem to be solved

3. reproduction (using cross-breeding and mutation geneticists to get new members)

4. assessing the extent to which new members respond better to resolving the problem

5. abandoning the old population by replacing it with the new population from the new generation.

Such a cycle is repeated until the best solution to the problem in question is identified.

Chapter 3 - Machine Learning.

Introduction

Arthur Samuel, a pioneer in the field of artificial intelligence and computer gaming, coined the term “Machine Learning”. He defined machine learning as – “Field of study that gives computers the capability to learn without being explicitly programmed”.

Machine Learning(ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance. The process starts with feeding good quality data and then training the machines(computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate.

Example: Training of students during exam.
While preparing for the exams students don’t actually cram the subject but try to learn it with complete understanding. Before the examination, they feed their machine(brain) with a good amount of high-quality data (questions and answers from different books or teachers notes or online video lectures). Actually, they are training their brain with input as well as output i.e. what kind of approach or logic do they have to solve a different kind of questions. Each time they solve practice test papers and find the performance (accuracy /score) by comparing answers with answer key given, Gradually, the performance keeps on increasing, gaining more confidence with the adopted approach. That’s how actually models are built, train machine with data (both inputs and outputs are given to model) and when the time comes test on data (with input only) and achieves our model scores by comparing its answer with the actual output which has not been fed while training. Researchers are working with assiduous efforts to improve algorithms, techniques so that these models perform even much better.

Fig 1. The difference between the so called traditional programming and machine learning

Basic Difference in ML and Traditional Programming?

Traditional Programming : We feed in DATA (Input) + PROGRAM (logic), run it on machine and get output.

Machine Learning : We feed in DATA(Input) + Output, run it on machine during training and the machine creates its own program(logic), which can be evaluated while testing.

In general, any machine learning problem can be assigned to one of two broad classifications: Supervised learning and Unsupervised learning.

How things work in reality:

- Talking about online shopping, there are millions of users with an unlimited range of interests with respect to brands, colors, price range and many more. While online shopping, buyers tend to search for a number of products. Now, searching a product frequently will make buyer's Facebook, web pages, search engine or that online store start recommending or showing offers on that particular product. There is no one sitting over there to code such task for each and every user, all this task is completely automatic. Here, ML plays its role. Researchers, data scientists, machine learners build models on the machine using good quality and a huge amount of data and now their machine is automatically performing and even improving with more and more experience and time.
- Even in health care also, ML is doing a fabulous job. Researchers and scientists have prepared models to train machines for **detecting cancer** just by looking at slide – cell images. For humans to perform this task it would have taken a lot of time. But now, no more delay, machines predict the chances of having or not having cancer with some accuracy and doctors just have to give an assurance call, that's it. The answer to – how is this possible is very simple -all that is required, is, high computation machine, a large amount of good quality image data, ML model with good algorithms to achieve state-of-the-art results.
  Doctors are using ML even to **diagnose patients** based on different parameters under consideration.

How ML works?

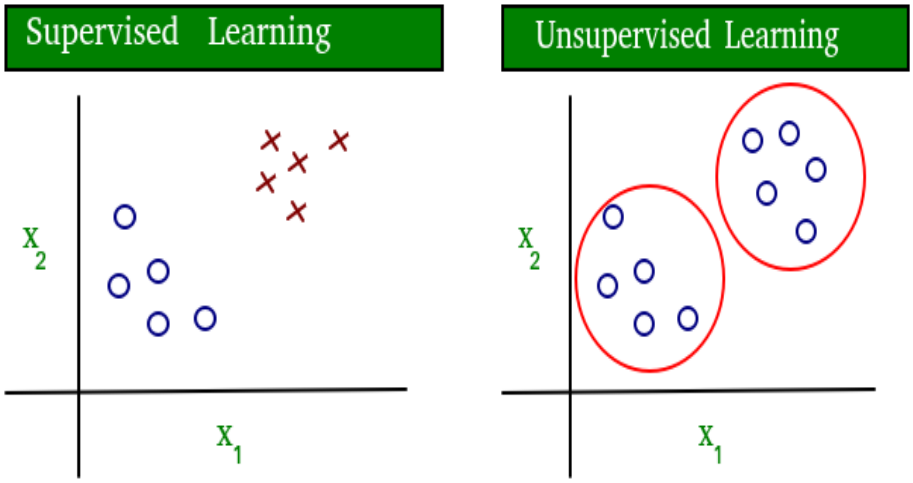
- Gathering past data in any form suitable for processing. The better the quality of data, the more suitable it will be for modeling
- Data Processing – Sometimes, the data collected is in the raw form and it needs to be pre-processed.
  Example: Some tuples may have missing values for certain attributes, an, in this case, it has to be filled with suitable values in order to perform machine learning or any form of data mining. Missing values for numerical attributes such as the price of the house may be replaced with the mean value of the attribute whereas missing values for categorical attributes may be replaced with the attribute with the highest mode. This invariably depends on the types of filters we use. If data is in the form of text or images then converting it to numerical form will be required, be it a list or array or matrix. Simply, Data is to be made relevant and consistent. It is to be converted into a format understandable by the machine
- Divide the input data into training, cross-validation and test sets.
- Building models with suitable algorithms and techniques on the training set.
- Testing our conceptualized model with data which was not fed to the model at the time of training and evaluating its performance using metrics such as F1 score, precision and recall.

Types of machine learning problems

There are various ways to classify machine learning problems. Here, we discuss the most obvious ones.

1. On basis of the nature of the learning "signal" or "feedback" available to a learning system

- **Supervised learning**: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. The training process continues until the model achieves the desired level of accuracy on the training data. Some real-life examples are:
  - **Image Classification:** You train with images/labels. Then in the future you give a new image expecting that the computer will recognize the new object.
  - **Market Prediction/Regression:** You train the computer with historical market data and ask the computer to predict the new price in the future.
- **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. It is used for clustering population in different groups. Unsupervised learning can be a goal in itself (discovering hidden patterns in data).
  - **Clustering:** You ask the computer to separate similar data into clusters, this is essential in research and science.

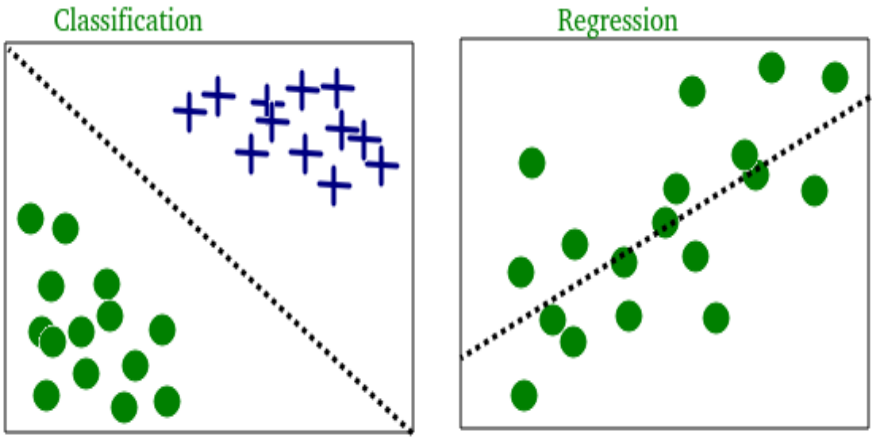


- **Semi-supervised learning**: Problems where you have a large amount of input data and only some of the data is labeled, are called semi-supervised learning problems. These problems sit in between both supervised and unsupervised learning. For example, a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled.
- **Reinforcement learning**: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space.

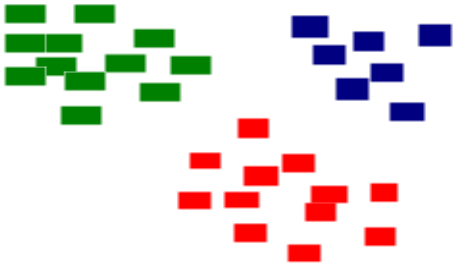
2. On the basis of "output" desired from a machine learned system

- **Classification**: Inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".
- **Regression**: It is also a supervised learning problem, but the outputs are continuous rather than discrete. For example, predicting the stock prices using historical data.

An example of classification and regression on two different datasets is shown below:



- **Clustering**: Here, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
  As you can see in the example below, the given dataset points have been divided into groups identifiable by the colors red, green and blue.



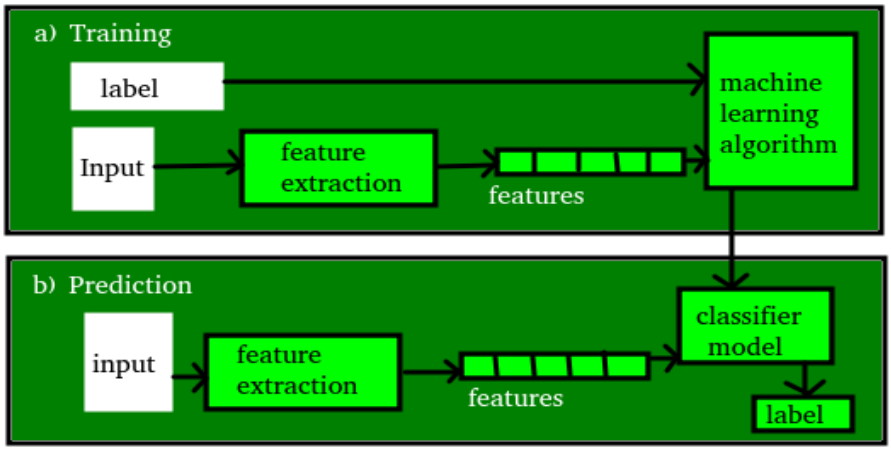
On the basis of these machine learning tasks/problems, we have a number of algorithms which are used to accomplish these tasks. Some commonly used machine learning algorithms are Linear Regression,

Logistic Regression, Decision Tree, SVM(Support vector machines), Naive Bayes, KNN(K nearest neighbors), K-Means, Random Forest, etc.

Terminologies of Machine Learning

- **Model** A model is a **specific representation** learned from data by applying some machine learning algorithm. A model is also called **hypothesis**.
- **Feature** A feature is an individual measurable property of our data. A set of numeric features can be conveniently described by a **feature vector**. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, **etc.**
- **Target (Label)** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the features section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

The figure shown below clears the above concepts:



Within the field of data analytics, machine learning is used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data set(input).

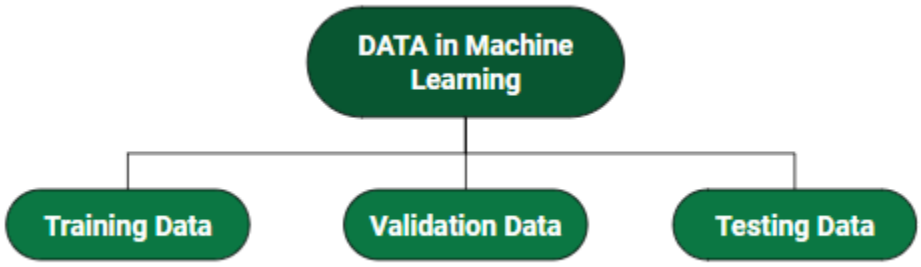
Suppose that you decide to check out that offer for a vacation . You browse through the travel agency website and search for a hotel. When you look at a specific hotel, just below the hotel description there is a section titled "You might also like these hotels". This is a common use case of Machine Learning called "Recommendation Engine". Again, many data points were used to train a model in order to predict what will be the best hotels to show you under that section, based on a lot of information they already know about you.

So if you want your program to predict, for example, traffic patterns at a busy intersection (task T), you can run it through a machine learning algorithm with data about past traffic patterns (experience E) and,

if it has successfully “learned”, it will then do better at predicting future traffic patterns (performance measure P).
The highly complex nature of many real-world problems, though, often means that inventing specialized algorithms that will solve them perfectly every time is impractical, if not impossible. Examples of machine learning problems include, “Is this cancer?”, “Which of these people are good friends with each other?”, “Will this person like this movie?” such problems are excellent targets for Machine Learning, and in fact machine learning has been applied such problems with great success.

How we split data in Machine Learning?

- **Training Data:** The part of data we use to train our model. This is the data which your model actually sees(both input and output) and learn from.
- **Validation Data:** The part of data which is used to do a frequent evaluation of model, fit on training dataset along with improving involved hyperparameters (initially set parameters before the model begins learning). This data plays it’s part when the model is actually training.
- **Testing Data:** Once our model is completely trained, testing data provides the unbiased evaluation. When we feed in the inputs of Testing data, our model will predict some values(without seeing actual output). After prediction, we evaluate our model by comparing it with actual output present in the testing data. This is how we evaluate and see how much our model has learned from the experiences feed in as training data, set at the time of training.



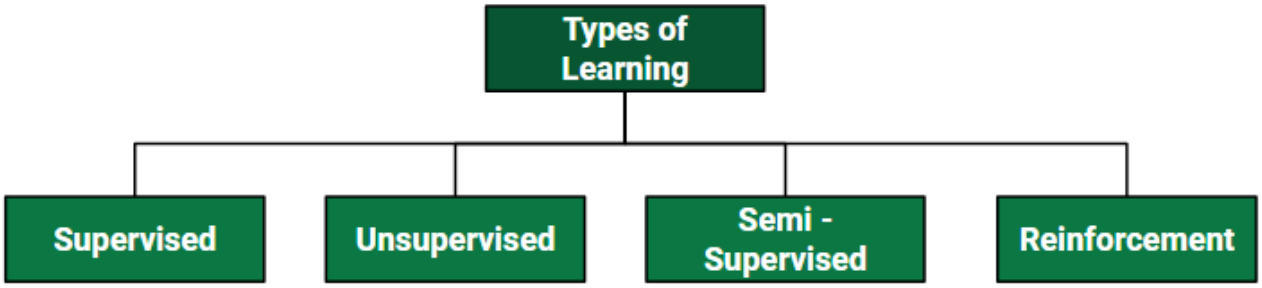
Machine Learning – Applications

Machine learning is actively being used today, perhaps in many more places than one would expect. We probably use a learning algorithm dozens of time without even knowing it. Applications of Machine Learning include:

- **Web Search Engine:** One of the reasons why search engines like google, bing etc work so well is because the system has learnt how to rank pages through a complex learning algorithm.
- **Photo tagging Applications:** Be it facebook or any other photo tagging application, the ability to tag friends makes it even more happening. It is all possible because of a face recognition algorithm that runs behind the application.
- **Spam Detector:** Our mail agent like Gmail or Hotmail does a lot of hard work for us in classifying the mails and moving the spam mails to spam folder. This is again achieved by a spam classifier running in the back end of mail application.

Supervised Learning

What is Learning for a machine? A machine is said to be learning from **past Experiences**(data feed in) with respect to some class of **Tasks**, if it's **Performance** in a given Task improves with the Experience.



Supervised Learning

Supervised learning is when the model is getting trained on a labelled dataset. **Labelled** dataset is one which have both input and output parameters. In this type of learning both training and validation datasets are labelled as shown in the figures below.

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

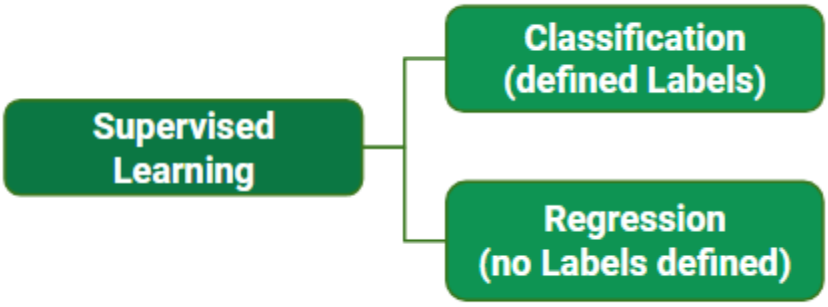
Figure B: REGRESSION

Both the above figures have labelled data set –

- **Figure A:** It is a dataset of a shopping store which is useful in predicting whether a customer will purchase a particular product under consideration or not based on his/ her gender, age and salary. **Input :** Gender, Age, Salary; **Output :** Purchased i.e. 0 or 1 ; 1 means yes the customer will purchase and 0 means that customer won't purchase it.
- **Figure B:** It is a Meteorological dataset which serves the purpose of predicting wind speed based on different parameters. **Input :** Dew Point, Temperature, Pressure, Relative Humidity, Wind Direction
  Output : Wind Speed

Training the system:

While training the model, data is usually split in the ratio of 80:20 i.e. 80% as training data and rest as testing data. In training data, we feed input as well as output for 80% data. The model learns from training data only. We use different machine learning algorithms to build our model. By learning, it means that the model will build some logic of its own.
Once the model is ready then it is good to be tested. At the time of testing, input is fed from remaining 20% data which the model has never seen before, the model will predict some value and we will compare it with actual output and calculate the accuracy.



Types of Supervised Learning:

1. **Classification :** It is a Supervised Learning task where output is having defined labels(discrete value). For example in above Figure A, Output – Purchased has defined labels i.e. 0 or 1 ; 1 means the customer will purchase and 0 means that customer won't purchase. The goal here is to predict discrete values belonging to a particular class and evaluate on the basis of accuracy. It can be either binary or multi class classification. In **binary** classification, model predicts either 0 or 1; yes or no but in case of **multi class** classification, model predicts more than one class. **Example:** Gmail classifies mails in more than one classes like social, promotions, updates, forum.
2. **Regression :** It is a Supervised Learning task where output is having continuous value. Example in above Figure B, Output – Wind Speed is not having any discrete value but is continuous in the particular range. The goal here is to predict a value as much closer to actual output value as our model can and then evaluation is done by calculating error value. The smaller the error the greater the accuracy of our regression model.

Example of Supervised Learning Algorithms:

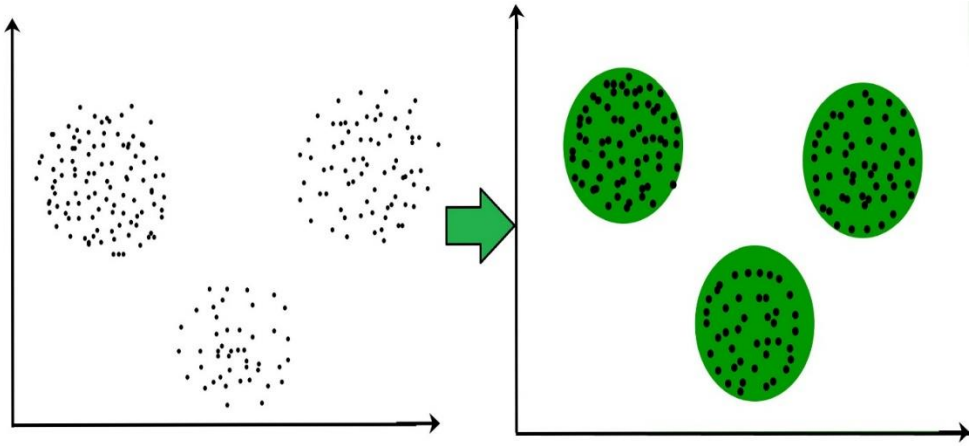
- Linear Regression
- Nearest Neighbor
- Guassian Naive Bayes
- Decision Trees
- Support Vector Machine (SVM)
- Random Forest

Clustering in Machine Learning

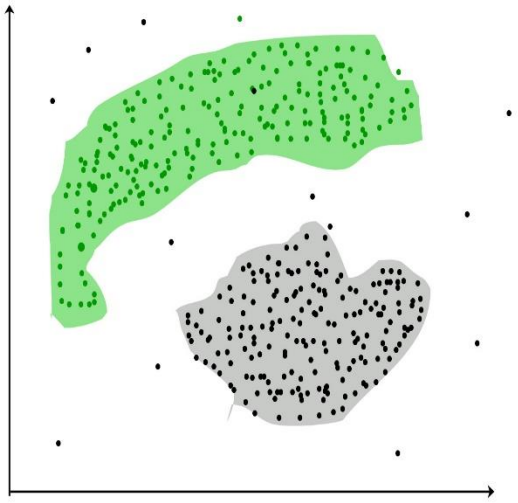
It is basically a type of *unsupervised learning method* . An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For ex– The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.

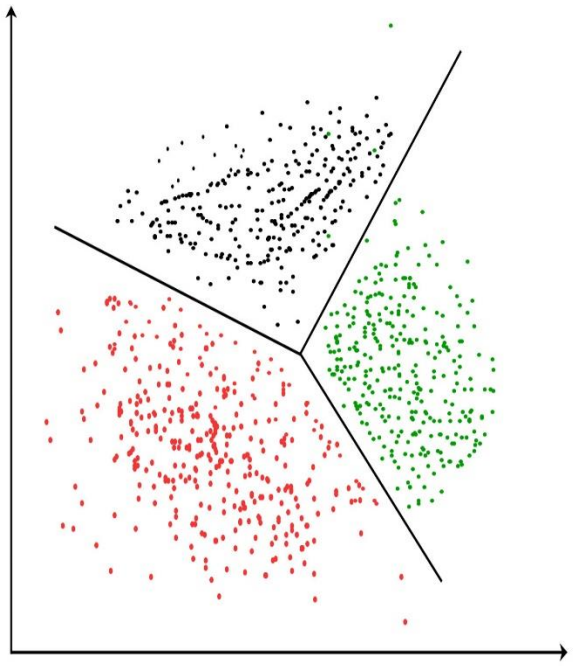


It is not necessary for clusters to be a spherical. Such as :



Clustering Algorithms :

K-means clustering algorithm – It is the simplest unsupervised learning algorithm that solves clustering problem.K-means algorithm partition n observations into k clusters where each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster .



Pandas and scikit-learn libraries

Important features of scikit-learn:

- Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
- Accessible to everybody and reusable in various contexts.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

Loading external dataset: Now, consider the case when we want to load an external dataset. For this purpose, we can use **pandas library** for easily loading and manipulating dataset.
In pandas, important data types are:

Series: Series is a one-dimensional labeled array capable of holding any data type.
DataFrame: It is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object.

1. **Load The Data. Dataset –** Iris data**.** It is famous data used as the "hello world" dataset in machine learning and statistics by pretty much everyone. The dataset contains 150 observations of iris flowers. There are four columns of measurements of the flowers in centimeters. The fifth column is the species of the flower observed. All observed flowers belong to one of three species.

1.1 Import libraries

First, let's import all of the modules, functions and objects to be used.

```
import pandas
```

```
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

1.2 Load Dataset

Data can directly be loaded the UCI Machine Learning repository. Using pandas to load the data and exploring descriptive statistics and data visualization.

Note – names of each column is specified when loading the data. This will help later at the time of exploring the data.

```
url =
"https://raw.githubusercontent.com / jbrownlee / Datasets / master /
iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length',
         'petal-width', 'class']
dataset = pandas.read_csv(url, names = names)
```

2. Summarize the Dataset

Now it is time to take a look at the data.

Steps to look at the data in a few different ways:

- Dimensions of the dataset.
- Peek at the data itself.
- Statistical summary of all attributes.
- Breakdown of the data by the class variable.

2.1 Dimensions of Dataset

```
# shape
print(dataset.shape)
```

2.2 Peek at the Data

```
# head
print(dataset.head(20))
```

2.3 Statistical Summary

This includes the count, mean, the min and max values as well as some percentiles.

```
# descriptions
print(dataset.describe())
```

It is clearly visible that all of the numerical values have the same scale (centimeters) and similar ranges between 0 and 8 centimeters.

2.4 Class Distribution

```
# class distribution
print(dataset.groupby('class').size())
```

2. Data Visualization

Using two types of plots:

1. Univariate plots to better understand each attribute.
2. Multivariate plots to better understand the relationships between attributes.

2.1 Univariate Plots Univariate plots – plots of each individual variable. Given that the input variables are numeric, we can create box and whisker plots of each.

```
# box and whisker plots
dataset.plot(kind ='box', subplots = True,  layout =(2, 2), sharex = False, sharey = False)
plt.show()
```

Creating histogram of each input variable to get an idea of the distribution.

```
# histograms
dataset.hist()
plt.show()
```

It looks like perhaps two of the input variables have a Gaussian distribution. This is useful to note as we can use algorithms that can exploit this assumption.

2.2 Multivariate Plots Interactions between the variables. First, let's look at scatterplots of all pairs of attributes. This can be helpful to spot structured relationships between input variables.

```
# scatter plot matrix
scatter_matrix(dataset)
plt.show()
```

Note the diagonal grouping of some pairs of attributes. This suggests a high correlation and a predictable relationship.

3. Evaluate Some Algorithms

Creating some models of the data and estimate their accuracy on unseen data.

1. Separate out a validation dataset.
2. Set-up the test harness to use 10-fold cross validation.
3. Build 5 different models to predict species from flower measurements
4. Select the best model.

3.1 Create a Validation Dataset One important aspect of all machine learning models is to determine their accuracy. Now, in order to determine their accuracy, one can train the model using the given dataset and then predict the response values for the same dataset using that model and hence, find the accuracy of the model.
But this method has several flaws in it, like:

- Goal is to estimate likely performance of a model on an **out-of-sample** data.
- Maximizing training accuracy rewards overly complex models that won't necessarily generalize our model.
- Unnecessarily complex models may over-fit the training data.

A better option is to split our data into two parts: first one for training our machine learning model, and second one for testing our model.

To summarize:

- Split the dataset into two pieces: a training set and a testing set.
- Train the model on the training set.
- Test the model on the testing set, and evaluate how well our model did.

Advantages of train/test split:

- Model can be trained and tested on different data than the one used for training.

- Response values are known for the test dataset, hence predictions can be evaluated
- Testing accuracy is a better estimate than training accuracy of out-of-sample performance.

Using statistical methods to estimate the accuracy of the models that we create on unseen data. A concrete estimate of the accuracy of the best model on unseen data is taken by evaluating it on actual unseen data.
Some data is used as testing data that the algorithms will not get to see and this data is used get a second and independent idea of how accurate the best model might actually be.
Testing data is split into two, 80% of which we will use to train our models and 20% that we will hold back as a validation dataset.

```
# Split-out validation dataset
array = dataset.values
X = array[:, 0:4]
Y = array[:, 4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split( X, Y, test_size = validation_size, random_state = seed)
```

X_train and Y_train are the training data for preparing models and a X_validation and Y_validation sets can be used later.

Test Harness
Using 10-fold cross-validation to estimate accuracy. This will split our dataset into 10 parts, train on 9 and test on 1 and repeat for all combinations of train-test splits.

```
# Test options and evaluation metric
seed = 7
scoring = 'accuracy'
```

'Accuracy' metric is used to evaluate models. It is the ratio of the number of correctly predicted instances in divided by the total number of instances in the dataset multiplied by 100 to give a percentage (e.g. 95% accurate).

3.2 Build Models- which algorithms would be good on this problem or what configurations to use, is not known. So, an idea is taken from the plots that some of the classes are partially linearly separable in some dimensions.

Evaluating 6 different algorithms:

- Logistic Regression (LR)
- Linear Discriminant Analysis (LDA)
- K-Nearest Neighbors (KNN).
- Classification and Regression Trees (CART).
- Gaussian Naive Bayes (NB).
- Support Vector Machines (SVM).

Algorithms chosen are a mixture of linear (LR and LDA) and nonlinear (KNN, CART, NB and SVM) algorithms. Random number seed are reset before each run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable.

Building and evaluating the models:

```
# Spot Check Algorithms
```

```
models = []
models.append(('LR',    LogisticRegression(solver    ='liblinear',    multi_class
='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma ='auto')))

# evaluate each model in turn
results = []
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits = 10, random_state = seed)
    cv_results = model_selection.cross_val_score(
            model, X_train, Y_train, cv = kfold, scoring = scoring)
    results.append(cv_results)
    names.append(name)
    msg = "% s: % f (% f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

3.3 Select Best Model

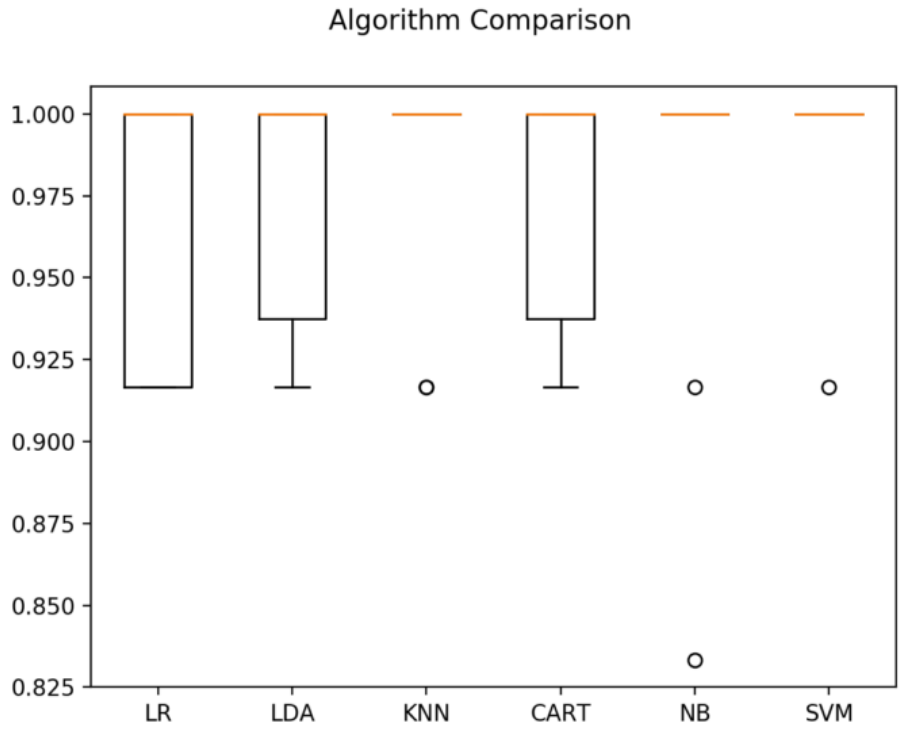
Comparing the models to each other and select the most accurate. Running the example above to get the following raw results:

```
LR: 0.966667 (0.040825)

LDA: 0.975000 (0.038188)

KNN: 0.983333 (0.033333)

CART: 0.975000 (0.038188)

NB: 0.975000 (0.053359)

SVM: 0.991667 (0.025000)
```

Support Vector Machines (SVM) has the largest estimated accuracy score. The plot of the model evaluation results is created and compare the spread and the mean accuracy of each model. There is a population of accuracy measures for each algorithm because each algorithm was evaluated 10 times (10 fold cross-validation).

```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

Box and whisker plots are squashed at the top of the range, with many samples achieving 100% accuracy.



4. **Make Predictions**

   The KNN algorithm is very simple and was an accurate model based on our tests. Running the KNN model directly on the validation set and summarizing the results as a final accuracy score, a confusion matrix, and a classification report.

```
# Make predictions on validation dataset
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

Accuracy is 0.9 or 90%. The confusion matrix provides an indication of the three errors made. Finally, the classification report provides a breakdown of each class by precision, recall, f1-score and support showing excellent results (granted the validation dataset was small).

```
0.9
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]
              precision    recall  f1-score   support
```

Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.85	0.92	0.88	12
Iris-virginica	0.90	0.82	0.86	11
micro avg	0.90	0.90	0.90	30
macro avg	0.92	0.91	0.91	30
weighted avg	0.90	0.90	0.90	30

Important points to note from the above code:

- We create a knn classifier object using:
- knn = KNeighborsClassifier(n_neighbors=3)
- The classifier is trained using X_train data. The process is termed as fitting. We pass the feature matrix and the corresponding response vector.
- knn.fit(X_train, y_train)
- Now, we need to test our classifier on the X_test data. **knn.predict** method is used for this purpose. It returns the predicted response vector, **y_pred**.
- y_pred = knn.predict(X_test)
- Now, we are interested in finding the accuracy of our model by comparing **y_test** and **y_pred**. This is done using metrics module's method **accuracy_score**:
- print(metrics.accuracy_score(y_test, y_pred))
- Consider the case when you want your model to make prediction on **out of sample** data. Then, the sample input can simply pe passed in the same way as we pass any feature matrix.
- sample = [[3, 5, 4, 2], [2, 3, 5, 4]]
- preds = knn.predict(sample)
- If you are not interested in training your classifier again and again and use the pre-trained classifier, one can save their classifier using **joblib**. All you need to do is:
- joblib.dump(knn, 'iris_knn.pkl')
- In case you want to load an already saved classifier, use the following method:
  knn = joblib.load('iris_knn.pkl')

References:

Andone, I., Robert J. Mockler, Dorothy G. Dologite, Alexandru Al. Tugui, Dezvoltarea sistemelor inteligente în economie: metodologie și studii de caz, Ed. Economica, Bucuresti 2001

Michael Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems, 3rd edition (2011), Addison Wesley

Joseph C. Giarratano, Gary D. Riley, Expert Systems: Principles and Programming, 4 edition (2004), Course Technology

Michael Bali, Drools Jbos Rules 5.x Developers Guide (2013), PACKT Publishing

http://iota.ee.tuiasi.ro/~mgavril/Simpe/L2.htm