PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

IBN KHALDOUN UNIVERSITY - TIARET

THESIS

Introduced to

MATHEMATICS AND COMPUTER SCIENCE FACULTY
DEPARTMENT OF COMPUTER SCIENCE

For the graduation of

MASTER

Specialty: Computer Engineering

By

ACHIR Mohamed Amine
ARABI Slimane

On the subject of

Multi-Agent Machine Learning :A Reinforcement Approach

Publicly Defended on .. / 09 / 2021 in Tiaret in front of the jury composed of :

Mr. ALEM Abdelkader	MAA	University of Tiaret	President
Mr. MOKHTARI Ahmed	MAA	University of Tiaret	Supervisor
Mr. BENAOUDA Habib	MAA	University of Tiaret	Examiner

2020 - 2021

بسم الله الرحمن الرحيم

والصلاة والسلام على أشرف المرسلين سيدنا و حبيبنا محمد ﷺ

الحمد لله الذي هدانا و ما كنا لنهتدي لولا أن هدانا الله

نحمده و نشكره على توفيقه لنا لإنجاز هذا العمل

وليس هناك افضل من كلمة شكر تنبع من القلب وتحمل في طياتها الاعتراف بالجميل فألف شكر للأستاذ مختاري أحمد الذي رافقنا في طيلة مراحل عملنا هذا

جزيل الشكر أيضا للأساتذة أعضاء لجنة المناقشة

الأستاذ عالم عبد القادر و الأستاذ بن عودة حبيب

و كل الطاقم العلمي لجامعة ابن خلدون

Acknowledgements

All thanks and praises go to Allah, the Almighty and the Merciful, for granting us the wisdom and the health to complete this work during this challenging time.

We would like to acknowledge the help and the supervision provided to us by ***Mr. MOKHTARI Ahmed*** *for his availability, recommendations, and confidence. His office door was always open whenever we ran into a trouble spotor questioned our project. His direct involvement and encouragement throughout the study influenced the positive progress and made us stronger and better students.*

A Special Thanks to ***Mr. ALEM Abdelkader*** *and* ***Mr. BENAOUDA Habib*** *, It is a great pleasure to have you as the jury for this work. We want to express our deep gratitude to you having kindly accepted to examine our modest work.*

We would like to express our appreciation to Google for its search engine for many useful inputs and valuable comments.

Big thanks to the teachers and the Computer Science and Mathematics faculty administration who watched over our training and our follow-up in our studies.

Finally, we address our thanks to all those who contributed to this work's outcome with their advice, constructive feedback, or encouragement.

Dedication

First, I thank God for granting me success in this work

I have the honor to dedicate this work to my kind parents who did not spare me financially or morally support and accompanied me as much as possible throughout the completion of this work.

Also to my brothers Islam and Achraf , the spoiled younger sister Fatouma and all the family of Achir and Abdelhadi.

I also dedicate the work to my friends DEGHMICHE Oussama and MADANI Youcef, all my classmates and everyone I know from near or far

Thank you all

Achir Mohamed Amine

Dedication

Dedication to my family. A special feeling of gratitude to my loving parents, whose words of encouragement and push for tenacity ring in my ears. My brothers Youcef , Abdelbasset , and Mohamed have never left my side and are very special. In addition, I will not forget my teacher Othman who has always been a lamp for us in this life.

To my friends who supported me throughout the process. I will always appreciate everything they did. I will not mention their names so as not to forget some of them. Thank you all.

Slimane Arabi.

Abstract

Learning is a process of improving the performance of a system based on its past experiences. This method intervenes when the problem seems too complicated to solve in real time, or when it seems impossible to solve the problem in a classic way. As an example of learning methods we cite reinforcement learning.

This method of learning is often used in the field of robotics and agents. It aims to determine a control law for a mobile robot or agent in an unknown environment. This kind of technique applies when it is assumed that the only information on the quality of the actions performed by the robot is a scalar signal that presents a reward or a punishment, the learning procedure aims to improve the choice of actions in order to maximize the rewards.

One of the most used algorithms for solving this learning problem is the Q-Learning algorithm that is based on the Q-Function, and to ensure the generation of this last function and the proper functioning of the learning system. , the action performed by the mobile robot in its environment is ensured by the use of a selection function, this action is evaluated by rewards and punishments.

Key words: Learning, Machine Learning, Multi Agent System, Agent, Reinforcement Learning, Q-Learning, reward, Punishment.

الملخص

التعلم هو عملية تحسين أداء النظام بناءً على خبراته السابقة. تتدخل هذه الطريقة عندما تبدو المشكلة معقدة للغاية بحيث لا يمكن حلها في الوقت الفعلي، أو عندما يبدو من المستحيل حل المشكلة بطريقة كلاسيكية.كمثال على طرق التعلم نذكر التعلم المعزز.

غالبًا ما تستخدم طريقة التعلم هذه في مجال الروبوتات والوكلاء. يهدف إلى تحديد قانون تحكم لروبوت متحرك أو وكيل في بيئة غير معروفة. ينطبق هذا النوع من التقنية عندما يُفترض أن المعلومات الوحيدة عن جودة الإجراءات التي يقوم بها الروبوت ، هي إشارة قياسية تقدم مكافأة أو عقوبة ، ويهدف إجراء التعلم إلى تحسين اختيار الإجراءات من أجل تعظيم المكافآت.

واحدة من الخوارزميات الأكثر استخداما لحل مشكلة التعلم هذه هي خوارزمية Q-Learning التي تعتمد على وظيفةQ ، ولضمان إنشاء هذه الوظيفة الأخيرة والتشغيل السليم لنظام التعلم. ، يتم ضمان الإجراء الذي يقوم به الروبوت المحمول في بيئته من خلال استخدام وظيفة التحديد ، ويتم تقييم هذا الإجراء من خلال المكافآت و العقوبات.

الكلمات المفتاحية: التعلم ، التعلم الآلي ، النظام متعدد الوكلاء ، الوكيل ،روبوت، التعلم المعزز ، خوارزمية التعلم Q،مكافأة، عقوبة.

Contenents

list of Acronymes

ML:	Machine Learning
RL:	Reinforcement Learning
MDP:	Markov Decision Process
MAS:	Multi Agent System
MARL:	Multi Agent Reinforcement Learning
QL:	Quality Learning

list of Equations

list of Figures

List of Tables

Introduction

Reinforcement learning (RL) and multi-agent systems (MAS) are promising tools in the field of artificial intelligence: the first allows to design the behavior of intelligent entities (agents) using simple rewards , and the second is based on the idea that intelligent behavior can “emerge” from the collaboration of a group of agents.

The problem posed is that of an agent placed in an environment within which he must achieve a goal, be it an end goal or a goal of "maintaining" a state. A first method is to plan your behavior in advance, knowing a sufficiently complete model of the environment. As this is often unpredictable, it may be necessary to revise the plan regularly. In the event that no sufficient model is known, the planning is no longer usable and it will be necessary to learn by trial and error a behavior that will solve the problem posed to the agent. This is how reinforcement learning is defined. As part of the decision-making process in Q learning, use will be made of statistical techniques and dynamic programming methods.

Different problems arise in such learning type :

1. Should we learn a behavior directly or is it through adaptation to the model of environment and its results? or conversely, learn a model and deduce the best possible behavior from it?
2. Knowing that there is a gain to be maximized, how to balance the use of the learning? already carried out(for immediate gain) and exploring possible behaviors (to improve future gain)?
3. How to design a single agent model that works with reinforcement learning methods?
4. How to adapt the Q-Learning algorithm to work in a multi agent system?
5. What is the more efficient model between our proposed models?

In a nutshell, we will present the theoretical aspects of the solution, then implement it and see the proposed solution in a comprehensive way from all sides.

Chapter 01

Machine Learning (ML)

Introduction

Machine learning is programming computers to optimize a performance criterion using example data or experience. We need learning in cases where we cannot directly write a computer program to solve a given problem, but need example data or experience. One case where learning is necessary is when human expertise does not exist, or when humans are unable to explain their expertise.

Another case is when the problem to be solved changes in time, or depends on the particular environment. We would like to have general purpose systems that can adapt to their circumstances, rather than explicitly writing a different program for each special circumstance.

Already, there are many successful applications of machine learning in various domains ,this is what we will discuss in this chapter, and we will also talk about the types of algorithms used in machine learning, especially reinforcement learning, which is one of the most prominent point of our project.

1. What is machine learning [1]

Machine learning is a field of study concerned with giving computers the ability to learn without being explicitly programmed.

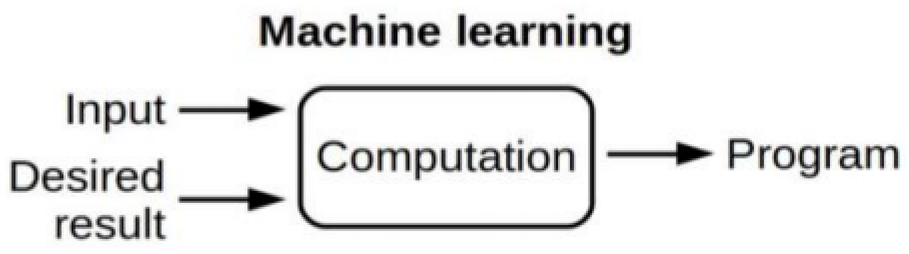


Figure 01: Machine-learning process

2. Applications of Machine Learning [2]

So, what can you do with machine learning? Quite a lot, really. This section breaks things down and describes how machine learning is being used at the moment.

2.1 Software

Machine learning is widely used in software to enable an improved experience with the user. With some packages, the software is learning about the user's behavior after its first use. After the software has been in use for a period of time it begins to predict what the user wants to do.

2.2 Stock Trading

Many platforms aim to help users make better stock trades.

These platforms have to do a large amount of analysis and computation to make recommendations. From a machine learning perspective, decisions are being made for you on whether to buy or sell a stock at the current price. It takes into account the historical opening and closing prices and the buy and sell volumes of that stock.

2.3 Medicine and Healthcare

The race is on for machine learning to be used in healthcare analytics. A number of startups are looking at the advantages of using machine learning with big Data to provide healthcare professionals with better-informed data to enable them to make better decisions.

2.4 Robotics

Using machine learning, robots can acquire skills or learn to adapt to the environment in which they are working. Robots can acquire skills such as object placement, grasping objects, and locomotion skills through either automated learning or learning via human intervention. With the increasing amount of sensors within robotics, other algorithms could be employed outside of the robot for further analysis.

In addition to this, there are many other uses for machine learning, which we can summarize in the following figure

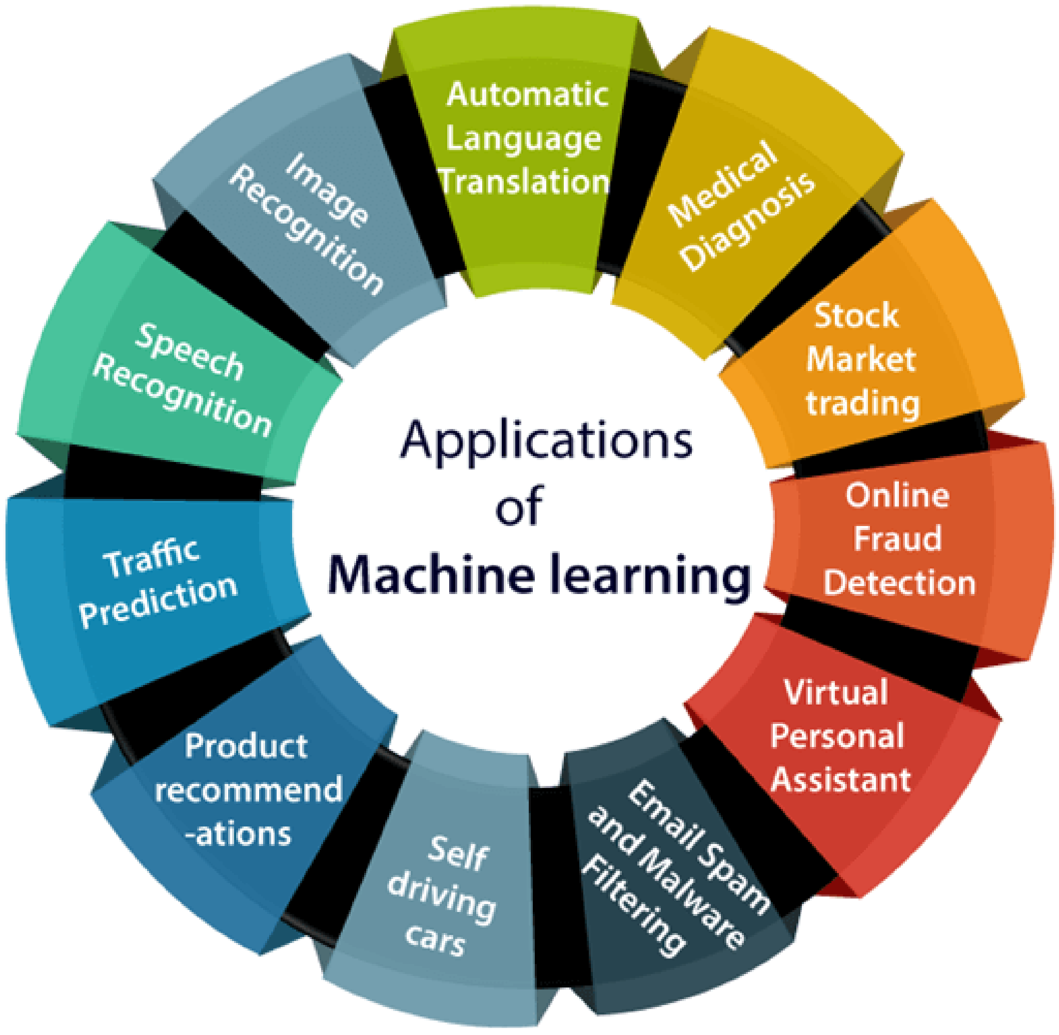

Applications of Machine learning
Automatic Language Translation
Medical Diagnosis
Stock Market trading
Online Fraud Detection
Virtual Personal Assistant
Email Spam and Malware Filtering
Self driving cars
Product recommend -ations
Traffic Prediction
Speech Recognition
Image Recognition

Figure 02: Applications of Machine Learning

3. Algorithm Types for Machine Learning

We can use many algorithms in machine learning. The required output is what decides which to use.

Machine learning algorithms characteristically fall into one of the following types:

- Supervised learning;
- Unsupervised learning;
- Reinforcement learning

3.1 Supervised learning

Supervised learning refers to working with a set of labeled training data. For every example in the training data, you have an input object and an output object.

Therefore, the algorithm generates a function that maps inputs to desired outputs. One standard formulation of the supervised learning task is the classification problem: the learner is required to learn (to approximate the behavior of) a function which maps a vector into one of several classes by looking at several input-output examples of the function.

In its most abstract form, supervised learning consists in finding a function $\mathbf{f : X \rightarrow Y}$ that takes as input $\mathbf{x \in X}$ and gives as output $\mathbf{y \in Y}$ (X and Y depend on the application): $\mathbf{y = f(x)}$**.**

In supervised learning, the aim is to learn a mapping from the input to an output whose correct values are provided by a supervisor.

An example would be classifying Twitter data. Assume you have the following data from Twitter; these would be your input data objects:

- Really loving the new St Vincent album!
- #fashion I'm selling my Louboutins! Who's interested?
- I've got my Hadoop cluster working on a load of data.

In order for your supervised learning classifier to know the outcome result of each tweet, you have to manually enter the answers; for clarity, I’ve added the resulting output object at the start of each line.

- **Music** Really loving the new St Vincent album!
- **Clothing** #fashion I'm selling my Louboutins! Who's interested? #louboutins
- **Big data** I've got my Hadoop cluster working on a load of data. #data

Obviously, for the classifier to make any sense of the data, when run properly, you have to work manually on a lot more input data. What you have, though, is a training set that can be used for later classification of data [2][3].

We can describe the process of this algorithm as bellow:

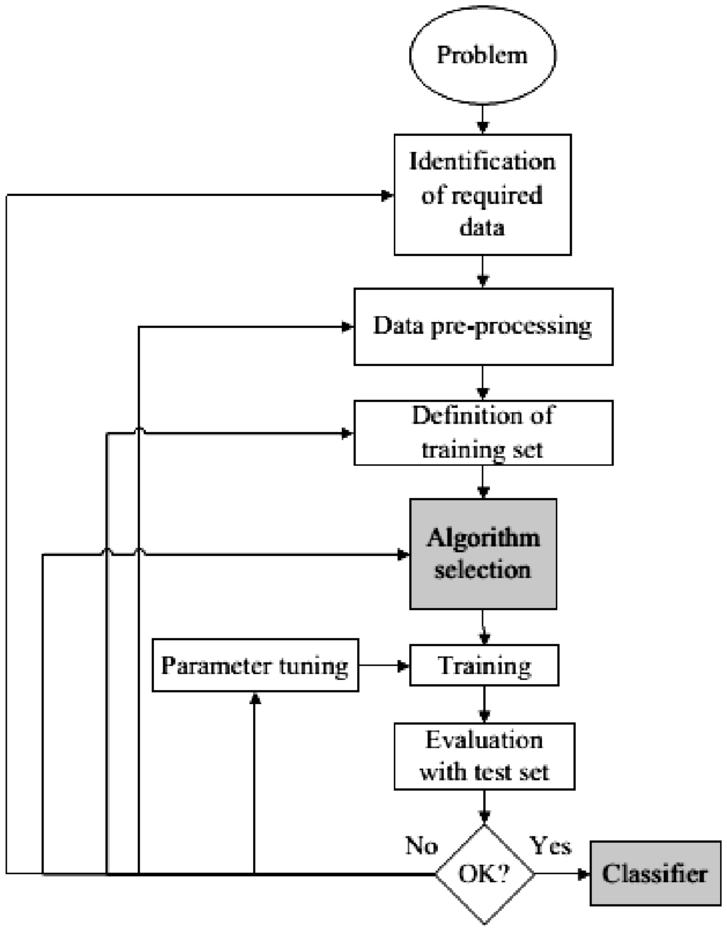


Figure 03: Supervised learning algorithm process [4]

3.2 Unsupervised learning

Unsupervised learning is a branch of machine learning that learns from data that do not have any label. It relates to using and identifying patterns in the data for tasks such as data compression or generative models.[3]

Therefore, unsupervised learning is when you let the algorithm find a hidden pattern in a load of data. With unsupervised learning there is no right or wrong answer; it's just a case of running the machine learning algorithm and seeing what patterns and outcomes occur.[3]

In unsupervised learning, there is no such supervisor and we only have input data. The aim is to find the regularities in the input. There is a structure to the input space such that certain patterns occur more often than others, and we want to see what generally happens and what does not. In statistics, this is called density estimation, and one from its most used and known methods is clustering also called cluster analysis.[3]

3.3 Reinforcement learning

3.3.1 Definition: "a way of programming agents by reward and punishment without needing to specify how the task is to be achieved" [5]

Reinforcement Learning is an approach through which intelligent programs, known as agents, work in a known or unknown environment to constantly adapt and learn based on giving points. The feedback might be positive, also known as rewards, or negative, also called punishments. Considering the agents and the environment interaction, we then determine which action to take.

Briefly, Reinforcement Learning is based on rewards and punishments.

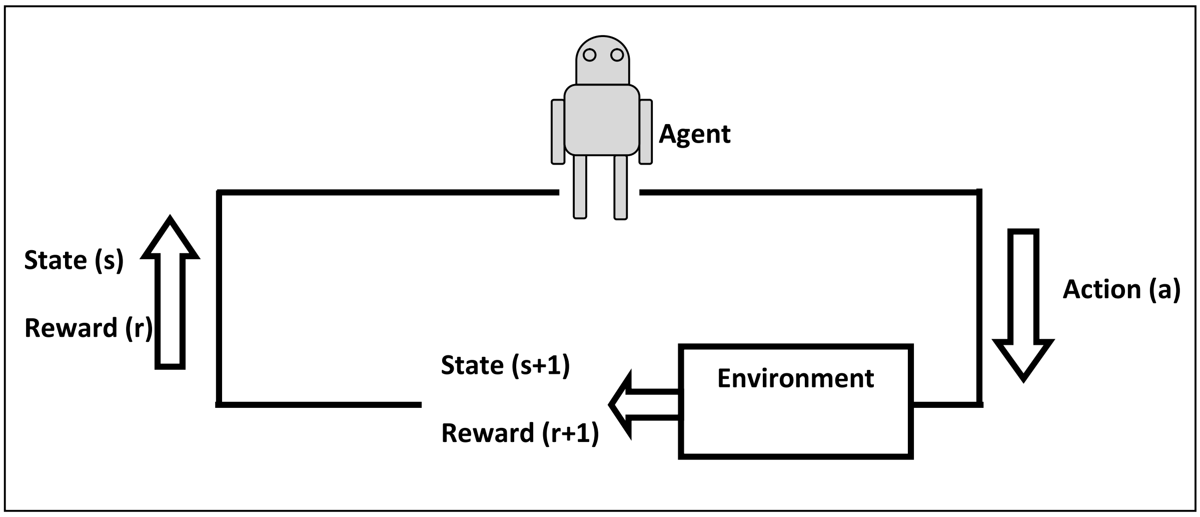


Figure 04: Reinforcement learning process[6]

3.3.2 Reinforcement learning elements

Beyond the agent and the environment, one can identify four main sub elements of a reinforcement learning system: a policy, a reward, a value function, and, optionally, a model of the environment.

- **An agent:** physical or software entity;
- **An environment:** the area which the agent runs in and reacts with;
- **A policy** defines the learning agent's way of behaving at a given time;
- **A reward** signal defines the goal in a reinforcement learning problem;
- **A value function** specifies what is good in the long run;
- **A model of the environment** which is something that mimics the behavior of the environment, or more generally, that allows inferences to be made about how the environment will behave. [6]

3.3.3 Reinforcement learning aims

- Improve the strategies used to solve any problem continuously by relying on the feedback received.
- Maximize the rewards while taking steps to solve the problem
- Achieve optimal steps that maximize the rewards to solve the problem at hand. [10]

3.3.4 Building units for RL [11]

- Policy Function π(a|s): Probabilistic function for actions depending on states, indicating how to act in a certain situation.

- Return G: Cumulative sum of future rewards in time, scaled by discount factor γ. It is defined as:

$$G_t = \sum_{i=t+1}^{\infty} \gamma^{i-t-1} r_i = r_{t+1} + \gamma G_{t+1} \text{ ....(1)}$$

- Value Function V(s|π): Expected return when policy π is followed at state s, defined as:

$$V^{\pi}(s) = \mathrm{E}[G_t | s_t = s, \pi] \text{...........(2)}$$

- Action-Value Function Q(s,a|π): Expected return when policy π is followed, except action a at first step at state s, defined as:

$$Q^{\pi}(s,a) = \mathrm{E}[G_t | s_t = s, a_t = a, \pi] \ldots\ldots\ldots\ldots(3)$$

3.3.5 Bellman Equation

The goal of RL at some time t = 0 is to find a policy that maximizes the accumulated sum of rewards over time $R_{\mathrm{t}} = \sum_{\mathrm{t}=0}^{\infty} \gamma^{\mathrm{t}} \mathrm{r}_{\mathrm{t}+1}$, where $\gamma \in [0,1]$ is called the discount factor and determines how strongly immediate rewards are weighted compared to rewards in the future. A discount factor $\gamma < 1$ guarantees that the future discounted return R_t is always a finite number if the immediate reward is bounded. Since the state transition process is random, the actually observed accumulated (discounted) reward R_t might be different from the expected return $\mathbb{E}[R_t|\pi, s_0]$ that the agent gets on average applying policy π starting from some initial states_0.The return has the recursive property $\sum_{\mathrm{t}=0}^{\infty} \gamma^{\mathrm{t}} \mathrm{r}_{\mathrm{t}+1} = \mathrm{r}_0 + \gamma \sum_{\mathrm{t}=1}^{\infty} \gamma^{\mathrm{t}} \mathrm{r}_{\mathrm{t}+1}$.Its expectation conditioned on the current state s and the policy π is called the value V^{π} of state [12]

$$V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^{\mathrm{t}} \mathrm{r}_{\mathrm{t}+1} | \pi, s_0 = s] = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} p_{s,s'}^{q} \left(R_{s;s'}^{a} + \gamma V^{\pi}(s')\right) \ldots\ldots\ldots\ldots(4)$$

3.3.6 Monte Carlo Methods [11]

Monte Carlo Methods uses statistical sampling to approximate Q function. In order to use such methods, one must wait until simulation ends because cumulative sum of rewards in the future is used for each state.

Once state s_t is visited and action a_t is taken, return G_t is calculated from its definition using instant and future rewards waiting to end of episode. Monte Carlo methods aim to minimize the gap between $Q(s_t,a_t)$ and target value G_t for all possible samples.

With a predetermined learning rate α, Q function is updated as;

$$Q(s_t, a_t) \leftarrow \alpha(G_t - Q(s_t, a_t)) \ldots\ldots\ldots\ldots(5)$$

3.3.7 Markov Decision Process (MDP)

MDP is the mathematics foundation of RL and if we wanted to fully understand RL algorithms, we always need to start with MDP. [07]

MDP is basically a framework for decision making under uncertainty. It can provide a way to compute an optimal decision policy

The MDP consists to the following [08]

- State space S, as a set of all possible states
- Action space A, as a set of all possible actions
- Model Function T(s'|s,a), as a state transition probabilities.
- Reward Function R(s), as rewarding mapping from state, action, next state tuple to reward.
- Discount Factor $\gamma \in [0,1]$, a real number determining importance of future rewards for control objective.

The work will be in three phases:

- Take note of what state you're in.
- Take an action based on your policy and receive a reward.
- Take note of the reward you received by taking that action in that state.

3.3.8 The Markov Property

- **Transition** : Moving from one state to another is called Transition.
- **Transition Probability**: The probability that the agent will move from one state to another is called transition probability. [08]

The Markov Property state that : **"Future is Independent of the past given the present"**

Mathematically we can express this statement as :

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots \dots, S_t] \dots\dots\dots\dots (6)$$

S[t] denotes the current state of the agent and S[t+1] denotes the next state. What this equation means is that the transition from state S[t] to S[t+1] is entirely independent of the past. [08]

3.3.9 State Transition Probability

As we now know about transition probability we can define state Transition Probability to another State .For Markov State from S[t] to S[t+1] i.e. any other successor state , the state transition probability is given by [08]:

$$P_{ss'} = \mathrm{P}\left[S_{t+1} = s'^{|S_t = s}\right] \dots\dots\dots\dots (7)$$

4. Difference between ML algorithms

The main difference is about the inputs and outputs of each one from the three algorithms previously discussed, the figure bellow shows this difference.

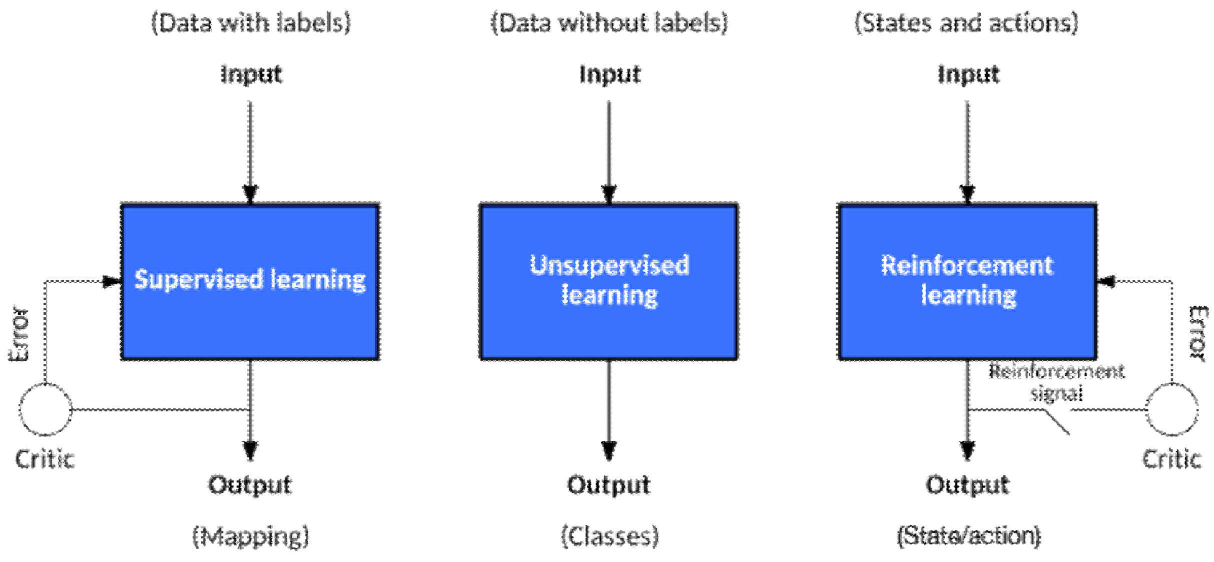


Figure 06: The difference between ML algorithms

Conclusion

Through what we discussed in this chapter, reinforcement learning is one of the most prominent methods used in machine learning in our time, and it has been very popular in the last twenty years

Reinforcement learning helps find the situation that needs action and also provides the learning agent with a reward function and enables it to discover the best way to obtain large rewards, and is useful in discovering the action that produces the highest reward over the longest period.

Because of all these characteristics, many techniques are used in reinforcement learning to eliminate many intractable problems.

Chapter 02

Multi Agent Systems (MAS)

Introduction

Multi-agent systems are a problem-solving tool in the field of artificial intelligence. The starting point is that some problems can be effectively solved by a set of agents rather than just one, intelligent behavior emerging from the combination of simple behaviors. It is therefore possible to use agents of low complexity for problems which are a priori difficult.

If different elements of the problem physically require the intervention of more than one agent, the multiplicity of agents can be all the more useful.

In MAS, each agent often has only incomplete information or insufficient capacity to solve the problem at hand. In addition, there is no overall control system: everyone makes their own decisions, possibly after having come to an understanding with their peers. These principles make it possible to use agents that are simple enough not to be too specialized, and therefore to be reusable.

In nature, the most visible multi-agent systems are animal societies: ants, men, sheep, even animal or plant associations which, by their common action, create group or more simply cooperative behavior. We are therefore interested in seeing how from a set of rather simple agents intelligent behavior emerges.

A multi-agent system is considered in its environment. Different elements are examined in the behavior of MAS: interactions with the environment, communication between agents, and the organization of society.

1. Agent

1.1 Definition

“An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors”. [13]

According to this definition an agent is any entity (physical or virtual one) that senses its environment and acting over it.

- **Physical entities** that could be considered as agents are, in the case of a power system, simple protection relay or any controller that controls directly particular power system component or part of the system.
- **Virtual entity** that can be considered as an agent is a piece of software that receives inputs from an environment and produces outputs that initiate acting over it.
- Often an agent is a combination of physical (computation architecture) and virtual one (a piece of software running on the computational architecture).[15]

1.2 Agents types

Agents can be classified according to various criteria. The most common separation into two categories is next :

1.2.1 Cognitive agents

Each agent is specialized in a field and knows how to communicate with others. They have explicit goals and plans to enable them to accomplish their goals. [14]

- Explicit representation of self, environment and other agents.
- Explicit organization.
- Explicit and elaborate interaction.

1.2.2 Reactive agents

Agents without intelligence (without anticipation, without planning) which react by stimulus-response to the current state of the environment. Intelligent behaviors can emerge from their association. [14]

- No explicit representation.
- Implicit / induced organization.
- Communication via the environment.

1.3 Properties of Agents

- **Reactive:** Responds in a timely fashion to changes in the environment
- **Autonomous:** Exercises control over its own actions
- **Goal-oriented:** Does not simply act in response to the environment
- **Temporally continuous** :is a continuously running process

- **Communicative:** communicates with other agents
- **Learning:** changes its behavior based on its previous experience
- **Mobile:** able to transport itself from one machine to another (this is associated manly with software agents)
- **Flexible:** actions are not scripted

These properties are often considered also as a type of agent. [15]

2. Multi Agent System (MAS)

2.1 Definition

“ A multi-agent system is a loosely coupled network of problem-solving entities (agents) that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity (agent)”[16] .

2.2 MAS composants

- **Eenvironment:** that is to say a space generally having a metric.
- **A set of objects:** These objects are located, that is to say that, for any object, it is possible at a given moment, to associate a position in the environment. These objects are passive, that is to say that they can be perceived, created, destroyed and modified by agents.
- **A set of agents:** which represent the active entities of the system.
- **A set of relations:** that unite objects and agents to each other.
- **A set of operations** allowing agents A to perceive, produce, consume, transform and manipulate the objects.
- **Operators** responsible for representing the application of these operations and the reaction of the world to this attempted modification, which will be called the laws of the universe. [14]

2.3 MAS conception

To design an MAS, it is necessary to define: [14]

MAS model

- the model of each of the agents that will come into action (microscopic level);
- define their environment and their interactions (macroscopic level);
- define the social organizations (macro level) which structure them.

A concrete model of MAS

- which creates, initializes the agents,
- set up their organization and;
- launches the agents who must intervene for a particular execution.

2.4 Architecture of multi-agent system

Based on the internal architecture of the particular individual agents forming the multi-agent system, it may be classified as two types:

- Homogeneous structure
- Heterogeneous structure

2.4.1 Homogeneous Structure

In a homogeneous architecture, all agents forming the multi-agent system have the same internal architecture. Internal architecture refers to the Local Goals, Sensor Capabilities, Internal states, Inference Mechanism and Possible Actions [17] . The difference between the agents is its physical location and the part of the environment where the action is done. Each agent receives an input from different parts of the environment. There may be overlap in the sensor inputs received. In a typical distributed environment, overlap of sensory inputs is rarely present [18] .

2.4.2 Heterogeneous Structure

In a heterogeneous architecture, the agents may differ in ability, structure and functionality [19] . Based on the dynamics of the environment and the location of the particular agent, the actions chosen by agent might differ from the agent located in a different part but it will have the same functionality. Heterogeneous architecture helps to make modelling applications much closer to real-world [20] .Each agent can have different local goals that may contradict the objective of other agents. A typical example of this can be seen in the Predator-Prey game . Here both the prey and the predator can be modelled as agents. The objectives of the two agents are likely to be in direct contradiction one to the other [21].

2.5 MAS types

A typology of cooperation from [22] seems the simplest and here we start with this typology as the basis for MAS classification. The typology is given in Figure 07.

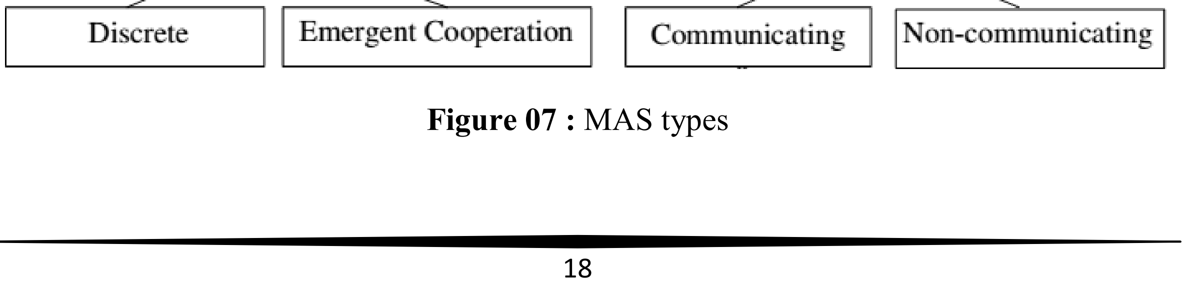


Figure 07 : MAS types

A MAS is independent if each individual agent pursues its own goals independently of the others. A MAS is discrete if it is independent, and if the goals of the agents bear no relation to one another. Discrete MAS involve no cooperation. However agents can cooperate with no intention of doing so and if this is the case then the cooperation is emergent[15]

2.5.1 Independent discrete MAS

This type of MAS is encountered in the environments that permit decoupling or decomposition (spatial, temporal). As an example of this type of MAS we use two agents (controllers) controlling synchronous generator in a power system: automatic voltage regulator (AVR) and speed governor. They have different goals that bear no relation to one another. AVR goal is to keep terminal voltage at predefined value and speed governor goal is to keep angular speed at synchronous value. [15]

2.5.2 Independent MAS with emergent cooperation

In this MAS agents are designed independently and each individual agent pursues its own goals independently of the others. It is more important to stress that in this MAS the individual agents are not aware of existence of other agents and each agent considers others as a part of the environment. Since agents exist in the same environment they can affect each other indirectly and the cooperation can emerge with no intention of doing so. The cooperation among independent agents can emerge in two ways:

- individual agents receive as sensory inputs (directly or they estimate them) the inputs or control of one or more other agents in the environment,
- individual agents, by their actions, change sensory inputs of another agent in a cooperative way without explicit intention of doing so. [15]

2.5.3 Cooperative MAS

Cooperative MAS is presented from two dimensions: agent heterogeneity and amount of communication among individual agents within a MAS. With respect to these two dimensions cooperative MAS can be classified in four groups: [15]

- Homogeneous non-communicating MAS,
- Homogeneous communicating MAS,
- Heterogeneous non-communicating MAS,
- Heterogeneous communicating MAS.

2.5.3.1 Homogeneous non-communicating MAS

In this MAS there are several different agents with identical structure. All individual agents have the same goals, domain knowledge, and possible actions. They also have the same procedure for selecting among their actions. The only differences among individual agents are their sensory inputs and the actual actions they take, or in other words they are situated (placed) differently in the environment. [15]

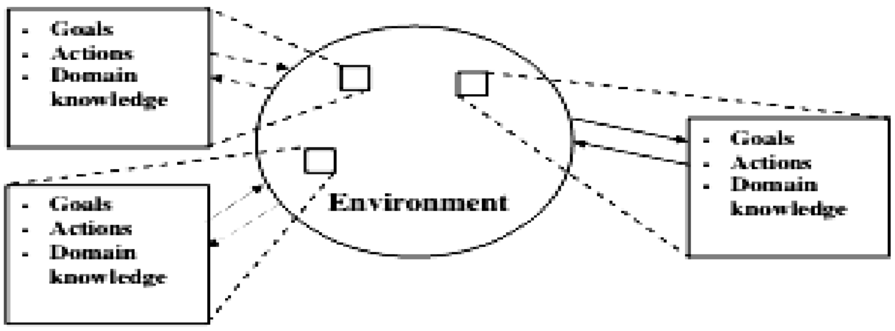


Figure 08: Homogeneous non-communicating MAS[15]

2.5.3.2 Homogeneous communicating MAS

In this MAS individual agents can communicate with each other directly. With the aid of communication, agents can coordinate more effectively. [15]

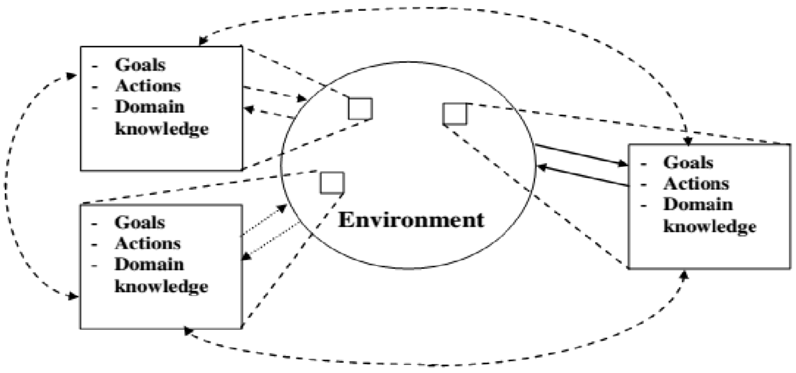


Figure 09: Homogeneous communicating MAS[15]

2.5.3.3 Heterogeneous non-communicating MAS

Individual agents within a MAS might be heterogeneous in a number of ways, from having different goals to having different domain knowledge and actions. Adding the possibility of heterogeneous agents in a MAS adds a great deal of potential power at the price of added complexity. [15]

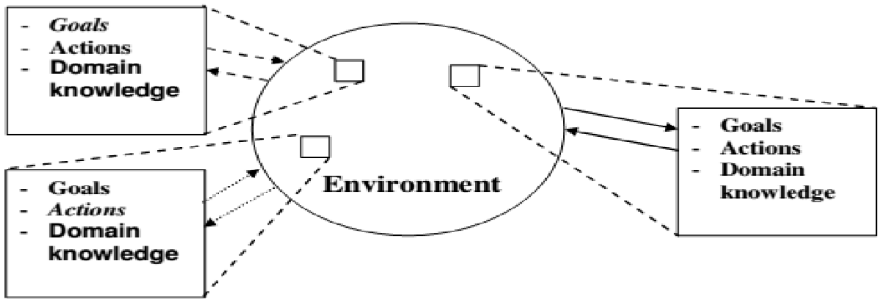


Figure 10: Heterogeneous non-communicating MAS[15]

2.5.3.4 Heterogeneous communicating MAS

In this type of MAS, ndividual agents (with different goals, actions, and /or domain knowledge) can communicate with one another. This MAS inherits the issues of communicating from homogeneous communicating MAS but heterogeneity brings additional issues to the communication. Two most important issues are: communication protocols and theories of commitment. Also, the issue of benevolence vs. competitiveness becomes more complicated for this type of MAS. [15]

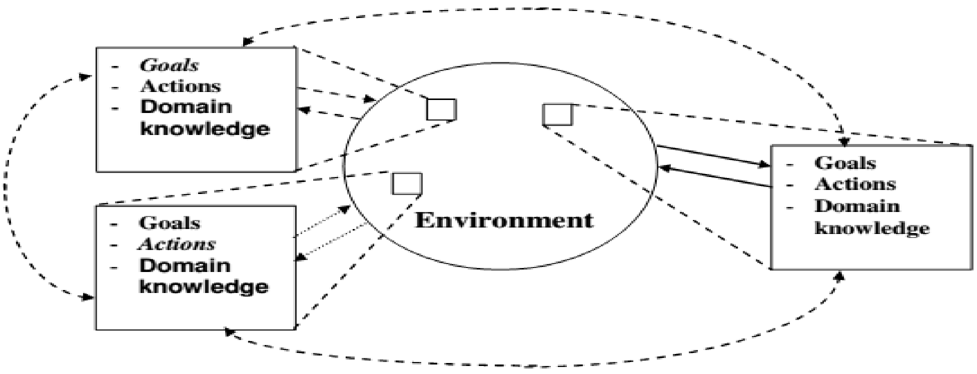


Figure 11: Heterogeneous communicating MAS[15]

2.6 Learning in MAS

The learning of an agent can be defined as building or modifying the belief structure based on the knowledge base, input information available and the consequences or actions needed to achieve the local goal [38]. Based on the above definition, agent learning can be classified into three types.

- Active learning
- Reactive learning
- Learning based on consequence

2.6.1 Active Learning

Active learning can be described as a process of analysing the observations to create a belief or internal model of the corresponding situated agent's environment. The active learning process can be performed by using a deductive, inductive or probabilistic reasoning approach.

In the deductive learning approach, the agent draws a deductive inference to explain a particular instance or state-action sequence using its knowledge base. Since the result learned is implied or deduced from the original knowledge base which already exists, the information learnt by each agent is not a new but useful inference. The local goal of each agent could form a part of the knowledge base. In the deductive learning approach, the uncertainty or the inconsistency associated with the agent knowledge is usually disregarded. This makes it not suitable for real-time applications. [41]

In an inductive learning approach, the agent learns from observations of stateaction pair. These viewed as the instantiation of some underlying general rules or theories without the aid of a teacher or a reference model. Inductive learning is effective when the environment can be presented in terms of some generalized statements. Well known inductive learning approaches utilize the correlation between the observations and the final action space to create the internal state model of the agent. The functionality of inductive learning may be enhanced if the knowledge base is used as a supplement to infer the state model. The inductive learning approach suffers at the beginning of operation as statistically significant data pertaining to the agent may not be available. [41]

The probabilistic learning approach is based on the assumption that the agent knowledge base or the belief model can be represented as probabilities of occurrence of events. The agent's observation of the environment is used to predict the internal state of the agent. One of the best examples of probabilistic learning is that of the Bayesian theorem. According to the Bayesian theorem, the posterior probability of an event can be determined by the prior probability of that event and the likelihood of its occurrence. The likelihood probability can be calculated based on observations of the samples collected from the environment and prior probability can be updated using the posterior probability calculated in the previous time step of the learning process. [41]

2.6.2 Reactive Learning

The process of updating a belief without having the actual knowledge of what needs to be learnt or observed is called as Reactive Learning. This method is particularly useful when the underlying model of the agent or the environment is not known clearly and are designated as black box. Reactive learning can be seen in agents which use connectionist systems such as neural networks. Neural networks depend on the mechanism which maps the inputs to output data samples using inter-connected computational layers. Learning is done by the adjustment of the synaptic weights between the layers. In [39], reactive multi-agent based feed forward neural networks have been used and its application to the identification of non-linear dynamic system have been demonstrated. In [40] many other reactive learning methods such as accidental learning, go-with-the-flow, channel multiplexing and a shopping around approach have been discussed. Most of these methods are rarely employed in a real application environment as they depend on the application domain. [41]

2.6.3 Learning Based on Consequences

Learning methods presented in the previous sections were concerned with understanding the environment based on the belief model update and analysis of patterns in sample observations. This section will deal with the learning methods based on the evaluation of the goodness of selected action. This may be performed by reinforcement learning methods.[41]

2.7 Roles of MAS

2.7.1 Simulation of complex phenomena

Multi-agent systems are used to simulate interactions between autonomous agents. We try to determine the evolution of this system in order to predict the resulting organization. For example, in sociology, we can configure the different agents that make up a community. By adding constraints, we can try to understand which will be the most efficient component to achieve an expected result (construction of a bridge). They even make it possible to experiment with scenarios that would not be feasible on real populations, whether for technical or ethical reasons [23].

2.7.2 Solve a problem in a distributed way

Agent actions are object transformations related to the description of a problem.[14]

2.7.3 Manage and maintain a work environment

The physical or social actions carried out by agents are real actions, they evolve over time and modify the world: football robots, agents negotiating a meeting with the user's profile.[14]

2.7.4 Program conception

At the same time, software engineering has evolved into increasingly autonomous components. MASs can be seen as the meeting of software engineering and distributed artificial intelligence, with a very important contribution from distributed systems. Compared to an object, an agent can take initiatives, can refuse to obey a request, can move around ... Autonomy allows the designer to concentrate on a humanly understandable part of the software [24]

2.8 Communication in Multi-Agent System

Communication is one of the crucial components in multi-agent systems that needs careful consideration. Unnecessary or redundant intra-agent communication can increase the cost and cause instability. Communication in a multi-agent system can be classified as two types. This is based on the architecture of the agent system and the type of information which is to be communicated between the agents[18]. The various issues arising in MAS system with homogeneous and heterogeneous architecture has been considered and explained by using a predator/prey and by the use of robotic soccer games[16]. Based on the information communication between the agents [25] , MAS can classified as local communication or Blackboard. Mobile communication can be categorized into class of local communication

2.8.1 Local Communication

Local communication has no place to store the information and there is no intermediate communication media present to act as a facilitator. The term message passing is used to emphasize the direct communication between the agents. Figure 12 shows the structure of the message passing communication between agents. In this type of communication, the information flow is bidirectional. It creates a distributed architecture and it reduces the bottleneck caused by failure of central agents[41]. This type of communication has been used in [26] [27] [28].

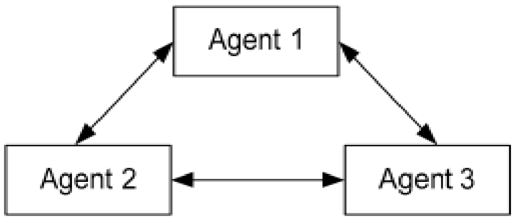


Figure 12: Message Passing Communication between agents[41]

2.8.2 Blackboards

Another way of exchanging information between agents is through Blackboards Agent-based blackboards, like federation systems, use grouping to manage the interactions between agents. There are significant differences between the federation agent architecture and the blackboard communication.

In blackboard communication, a group of agents share a data repository which is provided for efficient storage and retrieval of data actively shared between the agents. The repository can hold both the design data as well as the control knowledge that can be accessed by the agents. The type of data that can be accessed by an agent can be controlled through the use of a control shell. This acts as a network interface that notifies the agent when relevant data is available in the repository. The control shell can be programmed to establish different types of coordination among the agents. Neither the agent groups nor the individual agents in the group need to be physically located near the blackboards. It is possible to establish communication between various groups by remote interface communication. The major issue is due to the failure of blackboards. This could render the group of agents useless depending on the specific blackboard. However, it is possible to establish some redundancy and share resources between various blackboards. Figure 13. shows a single blackboard with the group of agents associated with it. Figure 14. shows blackboard communication between two different agent groups and also the facilitator agents present in each group. [41]

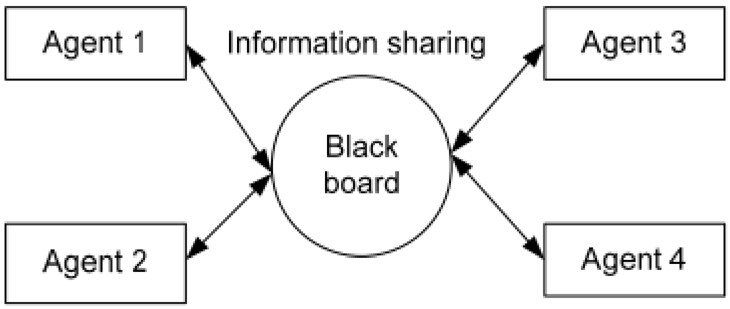


Figure 13: Blackboard type communication between agents. [41]

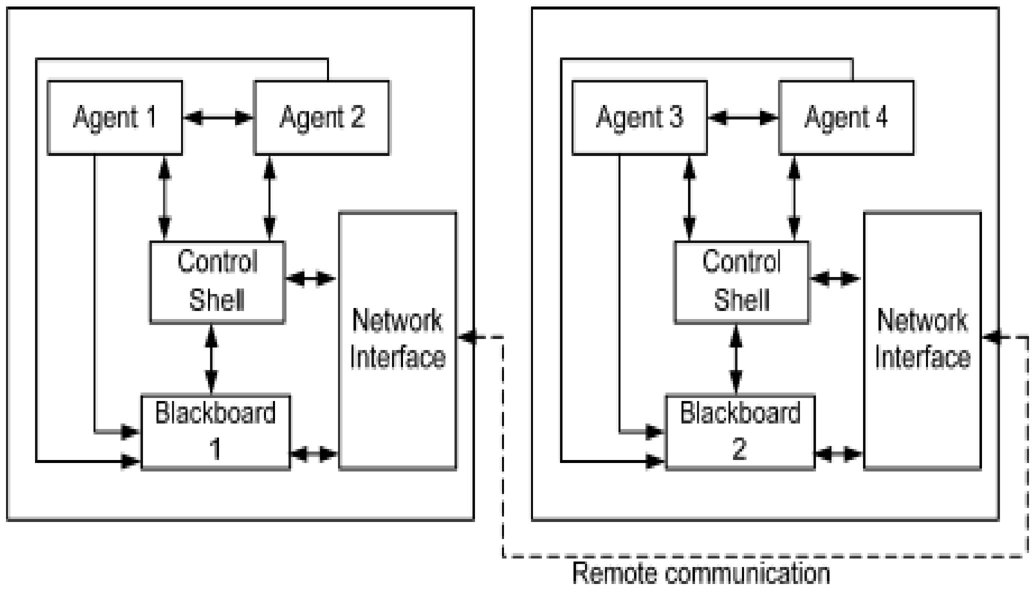


Figure 14: Blackboard communication using remote communication between agent groups. [41]

2.9 Advantages of MAS [15]

Main advantages of MAS are robustness and scalability.

- Robustness refers to the ability that if control and responsibilities are sufficiently shared among agents within MAS, the system can tolerate failures of one or more agents.
- Scalability of MAS originates from its modularity. It should be easier to add new agents to a MAS.

Conclusion

In this chapter, we talked about the agent, its types and characteristics, a brief survey of the existing architectures in multi agents, communication requirements, roles of MAS.

All this allowed us to take an expanded view of the agents, their method of work and their communication, especially in the event of a multiplicity of them, which will motivate us in further discoveries such as discovering the method of their work and their cooperation after contact and knowing the extent of their achievement of the desired goals.

Chapter 03

Multi Agent Reinforcement Learning

(MARL)

Introduction

In the field of machine learning (ML), reinforcement learning (RL) has attracted the attention of the scientific community owing to its ability to solve a wide range of tasks by using a simple architecture and without the need for prior knowledge of the dynamics of the problem to solve.

RL has found uses in many applications, from finance and robotics, to natural language processing and telecommunications.

The core of a RL system is the agent that operates in an environment that models the task that it has to fulfill. In all of the above applications, the RL agents interact with the environment via a trial and error approach, within which they receive rewards (reinforcement) for their actions.

This mechanism, similar to human learning, guides the agent to the improvement of its future decisions in order to maximize the upcoming rewards. Despite the success of this approach, a large number of real-world problems cannot be fully solved by a single active agent that interacts with the environment; the solution to that problem is the multiagent system (MAS), in which several agents learn concurrently how to solve a task by interacting with the same environment.

In Figure 15 , we show the representation of the RL structure for a single (a) agent and for an MAS (b).

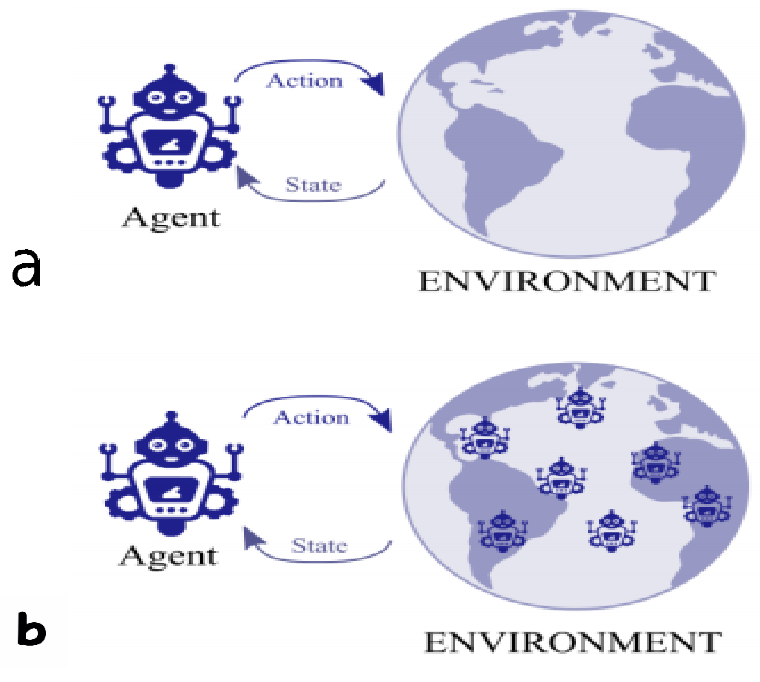


Figure 15: RL structure for a single agent and for an MAS

1. What is MARL?

Definition: Multi-agent reinforcement learning is an extension of reinforcement learning concept to multi-agent environments. Reinforcement learning allows to program agents by reward and punishment without specifying how to achieve the task. Formally agent-environment interaction in multi-agent reinforcement learning is presented as a discounted stochastic game. [30]

In the multi-agent scenario, much like in the single-agent scenario, each agent is still trying to solve the sequential decision-making problem through a trial-and-error procedure. The difference is that the evolution of the environmental state and the reward function that each agent receives is now determined by all agents' joint actions. As a result, agents need to take into account and interact with not only the environment but also other learning agents. A decision-making process that involves multiple agents is usually modeled through a stochastic game (Shapley, 1953), also known as a Markov game (Littman, 1994).

Therefore, MARL is the generalization of the Markov decision process to the multi-agent systems case, which known as stochastic game [31].

1.1 Definition: A stochastic game is a tuple $\langle X, U_1, \dots, U_n, f, \rho_1, \dots, \rho_n \rangle$ [31]

where

- n is the number of agents
- X is the finite set of environment states
- $U_1, \dots, U_n$ are the finite sets of actions available to the agents, yielding the joint action set
- $U = U_1 \times \dots \times U_n$, $f: X \times U \times X \rightarrow [0,1]$ is the state transition probability function.
- $\rho_i: X \times U \times X \rightarrow \mathbb{R}, \mathrm{i} = 1 \dots \mathrm{n}$ are the reward functions of the agents.

We assume that the reward functions are bounded. In the multi-agent case, the state transitions are the result of the joint action of all the agents, $u_{k=}\left[u_{1,k}^T, \dots, u_{n,k}^T\right]^T, u_k \in U, u_{i,k} \in U_i$, where T denotes vector transpose.

The policies $h_i: X \times U_i \rightarrow [0,1]$ form together the joint policy $\boldsymbol{h}$. Because the rewards $r_{i,k+1}$ of the agents depend on the joint action, their returns depend on the joint policy:

$$R_i^h(x) = E\{\textstyle\sum_{k=0}^{\infty} \gamma^k r_{i,k+1} \,|x_0 = x, h\} \text{..............(8)}$$

The Q-function of each agent depends on the joint action and on the joint policy, $Q_i^h: X \times U \rightarrow \mathbb{R}$ with

$$Q_i^h(x, i) = E\{\textstyle\sum_{k=0}^{\infty} \gamma^k r_{i,k+1} \,|x_0 = x, u_0 = u, h\} \text{ ....(9)}$$

2. MARL categories

MARL algorithms can be coarsely divided into three groups depending on the kind of reward given by the environment: fully cooperative, fully competitive, and mixed cooperative–competitive. [32]

2.1 Cooperative: All agents working towards a common goal. [32]

2.2 Competitive: Agents competing with one another to accomplish a goal[32]

2.3 Some mix of the two: Group of coordinating agents in competition with an other group of coordinating agents. [32]

The best way to understand this category is the following example: Think a 5 vs 5 basketball game, where individuals on the same team are coordinating with one another, but the two teams are competing against one another. [32]

3. Benefits of MARL

Experience sharing can help RL agents with similar tasks learn faster and reach better performance. For instance, the agents can exchange information using communication [33] , skilled agents may serve as teachers for the learner[34], or the learner may watch and imitate the skilled agents [35].

A speed-up can be realized in MARL thanks to parallel computation, when the agents exploit the decentralized structure of the task.

When one or more agents fail in a multi-agent system, the remaining agents can take over some of their tasks. This implies that MARL is inherently robust. Furthermore, by design most multi-agent systems also allow the easy insertion of new agents into the system, leading to a high degree of scalability.

Existing MARL algorithms often require some additional preconditions to theoretically guarantee and to exploit the above benefits . Relaxing these conditions and further improving the performance of MARL algorithms in this context is an active field of study [35].

4. Limits of MARL

The transition from single-agent to multi-agent settings introduces new challenges that require a different design approach for the algorithms, the most important limits are:

4.1 Non stationarity

The environment in a multi-agent setting can be modified by the actions of all agents; thus, from the single-agent perspective, the environment becomes non-stationary. The effectiveness of most reinforcement learning algorithms is tied to the Markov property, which does not hold in non-stationary environments [36]. Policies created in a non-stationary environment are deemed to have become outdated. Despite the loss of theoretical support, algorithms designed for the single-agent setting have been applied in multi-agent settings, such as independent learners (IL), occasionally achieving desirable results [33]

4.2 Scalability

As the number of agents increases, there is a growth in the joint action space. For this reason, centralized approaches, in which an observer selects the actions after receiving the action–state information of every agent, require large amounts of computational resources and memory to work with more than a couple of agents. A possible solution to the curse of dimensionality in MARL is to use independent learners, but as we have seen, this approach is unable to obtain consistent results in a non-stationary environment. A third model of agent connection is the decentralized setting with networked agents. In this setting, every agent is able to interact with the environment and to exchange information with few other agents (those in its vicinity), creating a time-varying communication network between all the agents. Algorithms developed for this setting are scalable to a massive number of agents and more real-world-oriented applications, as the absence of a central controller and uncertainty in communication links are typical requirements in a large number of applications. [30]

5. Multi-agent reinforcement learning goals

5.1 Convergence

Convergence to equilibria is a basic stability requirement . It means the agents' strategies should eventually converge to a coordinated equilibrium. [30]

5.2 Alternative to rationality

An alternative to rationality is the concept of no-regret, which is defined as the requirement that the agent achieves a return that is at least as good as the return of any stationary strategy, and this holds for any set of strategies of the other agents. This requirement prevents the learner from ‘being exploited’ by the other agents. [30]

5.3 Optimality/Compatibility/Safety

Targeted optimality/compatibility/safety are adaptation requirements expressed in the form of bounds on the average reward . Targeted optimality demands an average reward, against a targeted set of algorithms, which is at least the average reward of a best-response. Compatibility prescribes an average reward level in self-play, i.e., when the other agents use the learner’s algorithm. Safety demands a safety-level average reward against all other algorithms. An algorithm satisfying these requirements does not necessarily converge to a stationary strategy. [30]

5.4 Stability and adaptation

MARL can also be related to stability and adaptation. For instance, opponent-independent learning is related to stability, whereas opponent-aware learning is related to adaptation. An opponent-independent algorithm converges to a strategy that is part of an equilibrium solution regardless of what the other agents are doing. An opponent-aware algorithm learns models of the other agents and reacts to them using some form of best-response. Prediction and rationality are related to stability and adaptation, respectively. Prediction is the agent's capabilityto learn accurate models of the other agents. An agent is called rational if it maximizes its expected return given its models of the other agents. [30]

6. MARL algorithms

MARL algorithms can be classified along the categories or task type previously cited

The type of task considered by the learning algorithm leads to a corresponding classification of MARL techniques into those addressing cooperative, competitive, or mixed stochastic games.

A significant number of algorithms are designed for static (stateless) tasks only.

The following table summarizes the MARL algorithms by task type (category)

Cooperative		competitive	Mixed	
Static	Dynamic		Static	Dynamic
• JAL • FMQ	• Team-Q • Distributed-Q • OAL	• Minimax-Q	• Fictitious Play • MetaStrategy • IGA • WoLF-IGA • GIGA • GIGA-WoLF • AWESOME • Hyper-Q	• Single-agent RL • Nash-Q • CE-Q • Asymmetric-Q • NSCP • WoLF-PHC • PD-WoLF • EXORL

Table 01: Breakdown of MARL algorithms by the type of task they address. Reproduced from [37]

Conclusion

Multi-agent reinforcement learning (MARL) is a young, but active and rapidly expanding field of research. MARL aims to provide an array of algorithms that enable multiple agents to learn the solution of difficult tasks, using limited or no prior knowledge. To this end, MARL integrates results from single-agent RL , game theory, and direct policy search.

This chapter has provided an extensive overview of MARL . we have presented what is MARL and its categories, the main benefits and challenges of MARL , as well as the different viewpoints on defining the MARL learning goal. Then, we have gave in a summarized way a set of MARL algorithms for fully cooperative, fully competitive, and mixed tasks.

Chapter 04

Our proposed models

Introduction

As we have seen previously, multi Agent systems (MAS) and reinforcement learning can solve specific problems using multiple agents.

The MAS and RL "mix", using simple agents, makes it possible to solve complex problems. We will try to combine the advantages of both tools to create a flexible and efficient system.

In this section, that represents our work, we will design four models of reinforcement learning: one in the case of single agent and three in multi agent case using the Q-Learning algorithm.

First, we will represent our agents in a way that permits to them to work in different environments and easy to manipulate, by using "Threads". This way of representation will help us to adapt the Q-Learning algorithm to work in a multi agent system; because the Q-learning is a single agent reinforcement learning algorithm.

Then, we will present our models separately starting by the basic one: the single agent model for giving a general view of reinforcement learning and seeing the behavior of the agent during its interaction with the environment.

After that, we will enter in the multi agent framework starting by a competitive model, then a cooperative model, and finally a communicative model.

Finally, we will give a statistical comparison between our models according to the number of attempts and the time token to achieve the goal in the same environment conditions and see which model is more efficient.

1.Q learning

1.1 What is "Q"?

The "Q" in Q-learning stands for quality. Quality here represents how useful a given action is in gaining some future rewards.[41]

1.2 The Q-Learning algorithm in RL

Of all the types of algorithms available in reinforcement learning, ever wondered why this Q-learning has always been the sought-after one? Here is the answer:

<< Q-learning is a model-free, value-based, off-policy learning algorithm >>

- **Model-free:** The algorithm that estimates its optimal policy without the need for any transition or reward functions from the environment.
- **Value-based:** Q learning updates its value functions based on equations, (say Bellman equation) rather than estimating the value function with a greedy policy.
- **Off-policy:** The function learns from its own actions and doesn't depend on the current policy

In the Q-Learning algorithm, the goal is to learn iteratively the optimal Q-value function using the Optimality function.

To do so, we store all the Q-values in a table that will be updated at each time step using the Q-Learning iteration.[41]

1.3 Optimality function

Recall the definition of the (optimal) Q-function:

$$Q(s,a) \triangleq r(s,a) + \gamma \textstyle\sum_{s'} p(s'|s,a)V^*(s') \text{ ....(10)}$$

The optimality equation is then

$$V^*(s) = max_a Q(s,a), s \in S \text{ ....(11)}$$

1.4 Value iteration function

The value iteration function is given by:

$$V_{n+1}(s) = max_a\{r(s,a) + \gamma \textstyle\sum_{s'} p(s'|s,a)V_n(s')\}, s \in S \text{ ....(12)}$$

With $V_n \leftarrow V^*$ it can be reformulated as:

$$Q_{n+1} = r(s,a) + \gamma \textstyle\sum_{s'} p(s'|s,a) max_{a'} Q_n(s',a') \text{ ....(13)}$$

Then the update function become as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma . max_{\alpha'} Q(s',a') - Q(s,a)],\ s \leftarrow s' \text{ ....(14)}$$

1.5 Rewards

Since the agent is reward motivated and going to learn how to control the moves by trial experiences in the environment, we need to decide the **rewards** and/or **penalties** and their magnitude accordingly. Here a few points to consider:

- The agent should receive a high positive reward for a successful move because this behavior is highly desired.
- The agent should be penalized if it moves to an obstacle state.

1.6 State Space

The State Space is the set of all possible situations our agent could inhabit. The state should contain useful information the agent needs to make the right action.

1.7 Action Space

In each case, the agent takes action .The action in our case can be to move in the direction or to determine the announcement of the achievement of the desired goal.

This is the action space: the set of all the actions that our agent can take in a given state.

1.8 Q-learning hyper parameters

The hyper parameters of your model are the parameters that are external to the model that you will set yourself (think of the value in an algorithm such as -nearest neighbors; this is a hyper parameter. You have to set it yourself as there is nothing inherent in the data that determines what should be).[08]

The three most important hyper parameters for your agent are as follows:

Alpha: The learning rate

Gamma: The discount rate

Epsilon: The exploration rate

1.8.1 Alpha – deterministic versus stochastic environments

The agent's learning rate “alpha” ranges from zero to one. Setting the learning rate to zero will cause your agent to learn nothing, all of its exploration of its environment and the rewards it receives will affect its behavior at all, and it will continue to behave completely randomly.

Setting the learning rate to one will cause your agent to learn policies that are fully specific to a deterministic environment. One important distinction to understand is between deterministic and stochastic environments and policies.

Briefly, in a deterministic environment, the output is totally determined by the initial conditions and there is no randomness involved. We always take the same action from the same state in a deterministic environment.

In a stochastic environment, there is randomness involved and the decisions that we make are given as probability distributions. In other words, we don't always take the same action from the same state.[08]

1.8.2 Gamma – current versus future rewards

Let us discuss the concept of current rewards versus future rewards. Your agent's discount rate gamma has a value between zero and one, and its function is to discount future rewards against immediate rewards.

Your agent is deciding what action to take based not only on the reward it expects to get for taking that action, but on the future rewards it might be able to get from the state it will be in after taking that action.

One easy way to illustrate discounting rewards is with the following example of a mouse in a maze collecting cheese as rewards and avoiding cats and traps (that is, electric shocks):

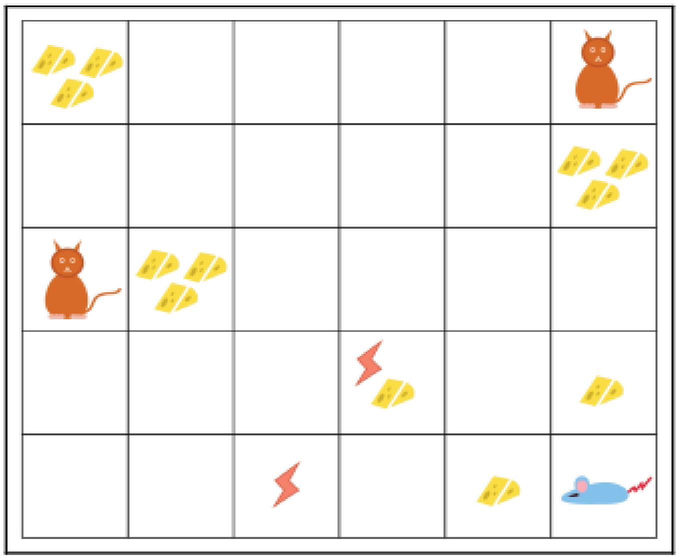


Figure 16: Reward illustration

The rewards that are closest to the cats, even though their point values are higher (three versus one), should be discounted if we want to maximize how long the mouse agent lives and how much cheese it can collect. These rewards come with a higher risk of the mouse being killed, so we lower their value accordingly. In other words, collecting the closest cheese should be given a higher priority when the mouse decides what actions to take.

When we discount a future reward, we make it less valuable than an immediate reward (similar to how we take into account the time value of money when making a loan and treat a dollar received today as more valuable than a dollar received a year from now).

The value of gamma that we choose varies according to how highly we value future rewards:

- If we choose a value of zero for gamma, the agent will not care about future rewards at all and will only take current rewards into account
- Choosing a value of one for gamma will make the agent consider future rewards as highly as current rewards[08]

1.8.3 Epsilon – exploration versus exploitation

Your agent's exploration rate epsilon also ranges from zero to one. As the agent explores its environment, it learns that some actions are better to take than others, but what about states and actions that it hasn't seen yet? We don't want it to get stuck on a local maximum, taking the same currently highest-valued actions over and over when there might be better actions it hasn't tried to take yet.

When you set your epsilon value, there will be a probability equal to epsilon that your agent will take a random (exploratory) action, and a probability equal to 1-epsilon that it will take the current highest Q-valued action for its current state.

As the agent gets more and more familiar with its environment, we want it to start sticking to the high-valued actions it's already discovered and do less exploration of the states it hasn't seen. We achieve this by having epsilon decay over time as the agent learns more about its environment and the Q-table converges on its final optimal values. There are many different ways to decay epsilon, either by using a constant decay factor or basing the decay factor on some other internal variable. Ideally, we want the epsilon decay function to be directly based on the Q-values that we've already discovered.[08]

1.9 Q-learning algorithm stages

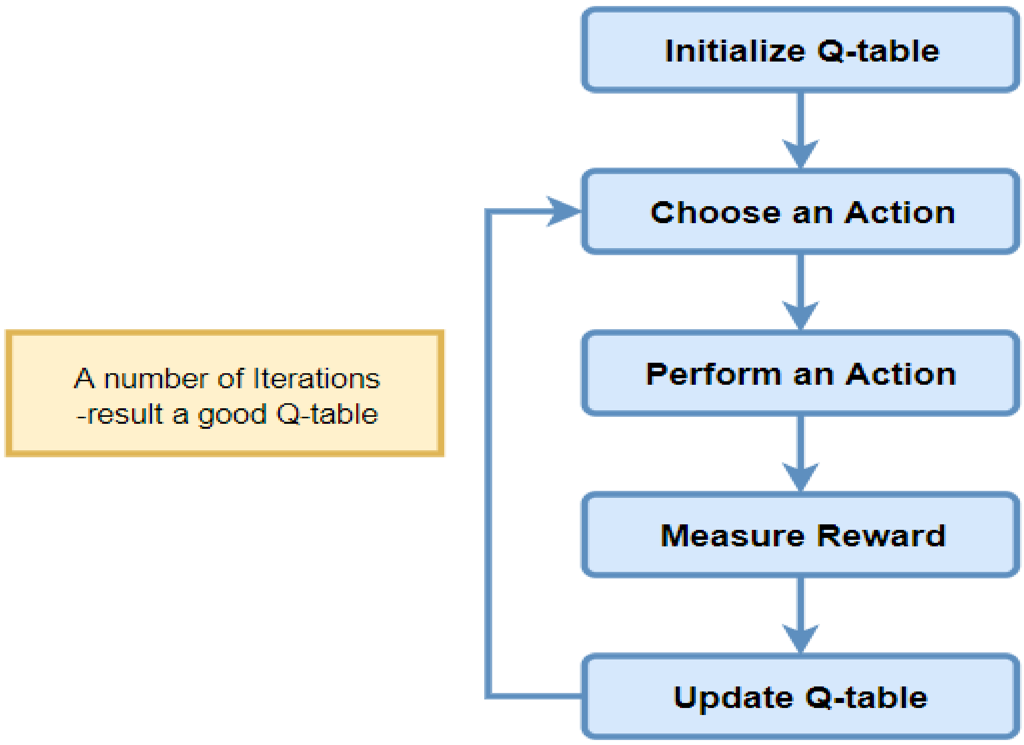


Figure 17: Q-learning Algorithm steps [08]

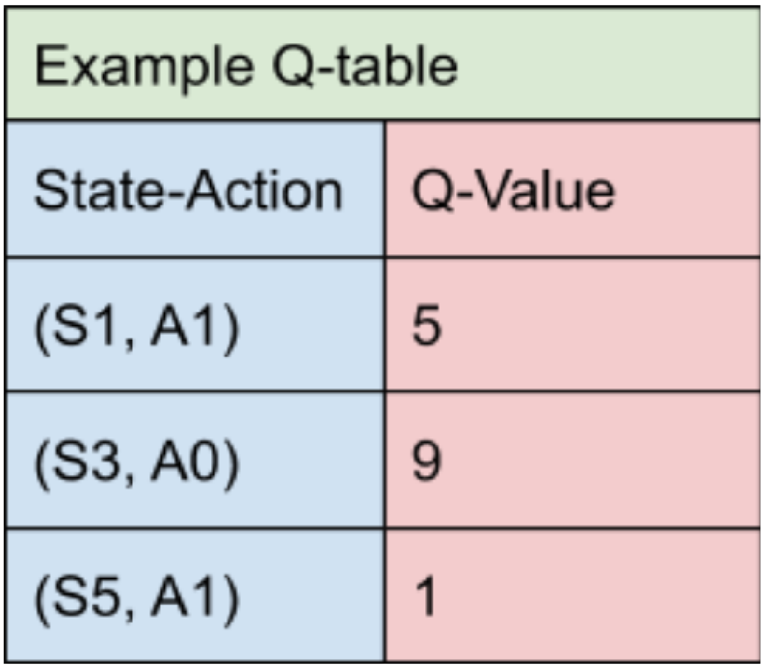
1.10 The procedural form of Q-learning algorithm

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s using policy derived from Q
         (e.g., ε-greedy)
      Take action a, observe r, s'
      $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{\alpha'} Q(s', a') - Q(s, a)]$
      s <-- s';
   until s is terminal
```

1.10.1 Step 1: Initialize the Q-table:

The Q-table is a simple data structure that we use to keep track of the states, actions, and their expected rewards. More specifically, the Q-table maps a state-action pair to a Q-value (the estimated optimal future value) which the agent will learn. At the start of the Q-Learning algorithm, the Q-table is initialized to all zeros indicating that the agent doesn't know anything about the world. As the agent tries out different actions at different states through trial and error, the agent learns each state-action pair's expected reward and updates the Q-table with the new Q-value. Using trial and error to learn about the world is called Exploration.

One of the goals of the Q-Learning algorithm is to learn the Q-Value for a new environment. The Q-Value is the maximum expected reward an agent can reach by taking a given action A from the state S. After an agent has learned the Q-value of each state-action pair, the agent at state S maximizes its expected reward by choosing the action A with the highest expected reward. Explicitly choosing the best known action at a state is called Exploitation.[08]



Example Q-table	
State-Action	Q-Value
(S1, A1)	5
(S3, A0)	9
(S5, A1)	1

Figure 18: Example Q-table mapping states and actions to their corresponding Q-value

1.10.2 Step 02: Choose an action using the Epsilon-Greedy Exploration Strategy:

A common strategy for tackling the exploration-exploitation tradeoff is the Epsilon Greedy Exploration Strategy.

- At every time step when it's time to choose an action, roll a dice

- If the dice has a probability less than epsilon, choose a random action
- Otherwise take the best known action at the agent's current state

Note that at the beginning of the algorithm, every step the agent takes will be random which is useful to help the agent learn about the environment it's in. As the agent takes more and more steps, the value of epsilon decreases and the agent starts to try existing known good actions more and more. Note that epsilon is initialized to 1 meaning every step is random at the start. Near the end of the training process, the agent will be exploring much less and exploiting much more.[08]

1.10.3 Step 03: Measure Reward:

He takes action and waits for the reward result.[08]

1.10.4 Step 04: Evaluation:

Needs job update the function Q(s,a).

This process is repeated again and again until the learning is stopped. In this way the Q-Table is been updated and the value function Q is maximized. Here the Q(state, action) returns the expected future reward of that action at that state.[08]

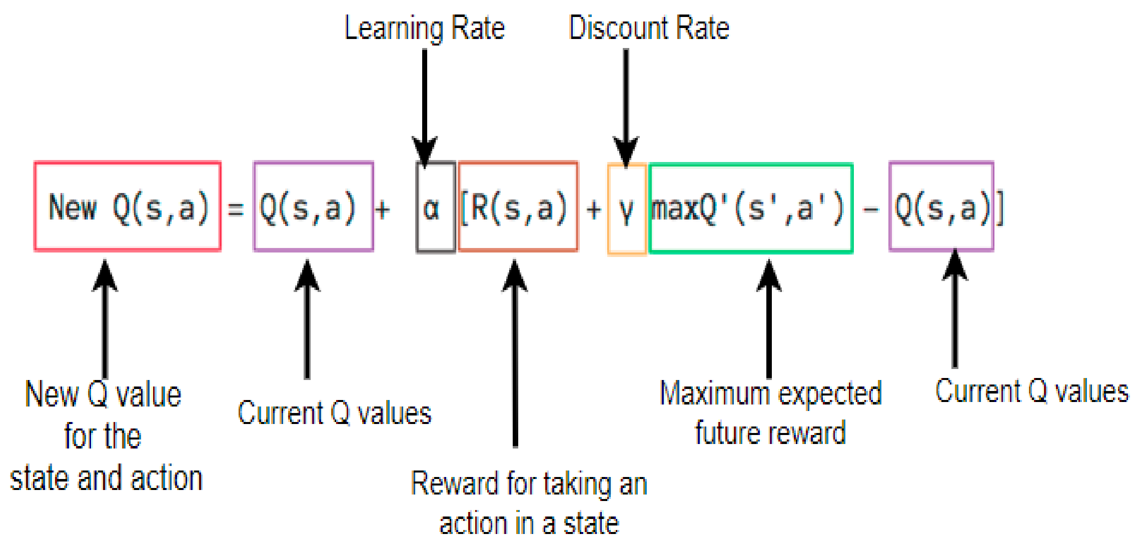

Learning Rate
Discount Rate
New Q(s,a) = Q(s,a) + α [R(s,a) + γ maxQ'(s',a') - Q(s,a)]
New Q value for the state and action
Current Q values
Reward for taking an action in a state
Maximum expected future reward
Current Q values

Figure 19: Q-learning update function

2. Our case: scenario of robots self drive

In the context of teaching robots to self-driving, each robot makes decisions on its own, outside of the user's control, as these decisions will be appropriate for the environment.
After launch, the robots must reach a specific location.

In order to ensure maximum efficiency, the robots will need to know the path between the launch area and the target area without hitting some of the obstacles in place.

We will be using Q-Learning to get this job done!

2.1 Define the Environment

The environment consists of states, actions, and rewards. States and actions are inputs for the Q-learning agent, while the possible actions are the agent's outputs.

2.1.1 States

The states in the environment are all of the possible locations within the environment. Some of these locations are obstacles (black squares), while other locations are aisles that the robot can use to travel (white squares). The green square indicates the goal state.

The bottom line used as robot's start positions. Let us take a simple example of 10x10 states The agent starts in the white squares of the bottom line as per his ID for example the agent "1" will start in the position (10,1)

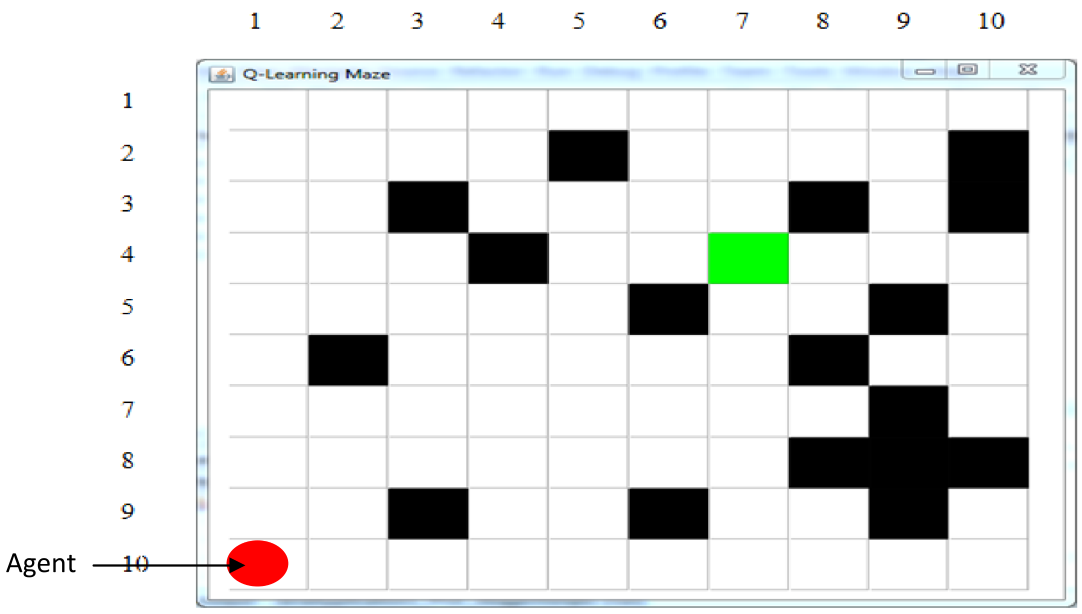


Figure 20 : States illustration

The agent's objective is to know the path between the starting area and the arrival area, passing through the permitted areas only.

As shown in the image above, there are 100 possible cases (locations). These states are arranged randomly in a grid containing 10 row, and 10 columns. Each location can thus be identified by its row and column index ,for example the goal position is (4,7).

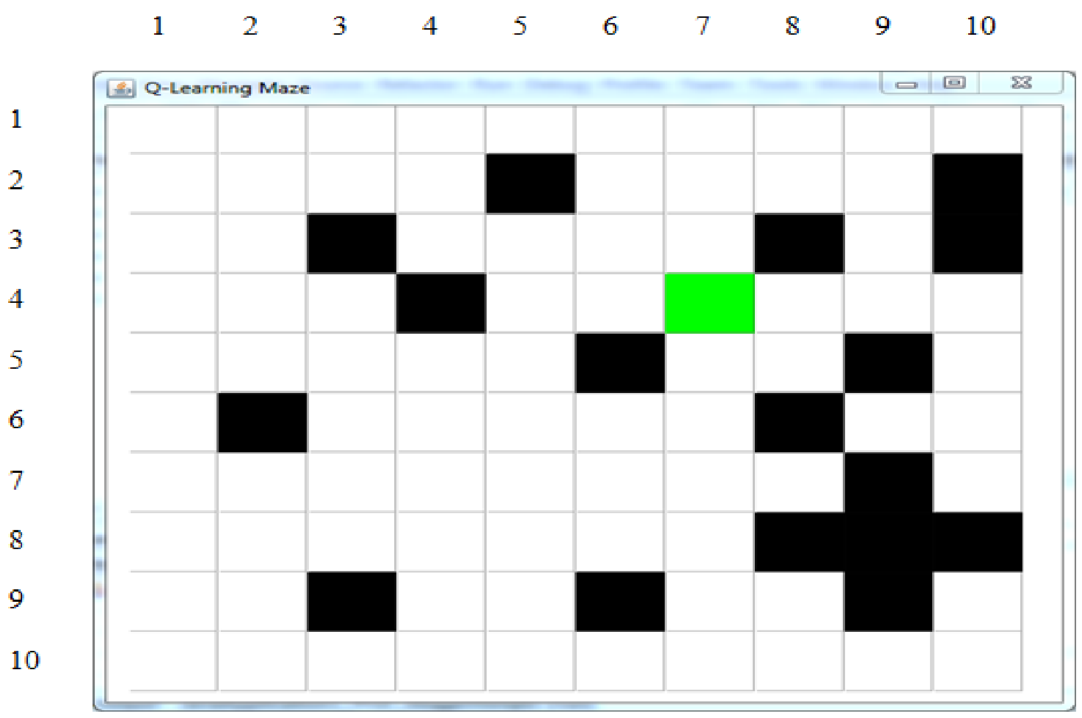
2.1.2 Actions

The actions that are available to the agent are to move the robot in one of four directions:

- Up
- Right
- Down
- Left

2.1.3 Rewards

The final component of the environment that we need to identify is rewards. To help the agent learn, each state (location) is assigned a value. The figure bellow shows rewards value for each state.



🟩 Reward = 1

⬜ Reward = 0

⬛ Reward = -1

Figure 21: Reward values

Via its cumulative rewards (by reducing cumulative penalties), the agent can find the shortest paths between the access area (green square) and all other locations where the robot is allowed to travel (white squares). The agent will also need to learn how to avoid hitting any of the obstacles (black squares)!

2.2 Our models

2.2.1 Single Agent model

The agent is alone in the environment and will search for the target by following the steps of the Q-learning previously described.

2.2.2 Competitive model (non centralized)

In this model, the number of agents will be greater or equal to two, where each agent has its own Q-table and is independent of the others, so each agent will search for the target alone, so there is something of competition.

2.2.3 Cooperative model (centralized)

- The same characteristics as the previous model, but this time the Q-table will be shared by all agents, and therefore it will be a kind of cooperation between them in finding the way to the goal
- In the event that one of the agents finds the way, it will be followed by the other agents even though they do not find the way
- This model will allow to shorten the time due to the advantage of cooperation in it.

2.2.4 Communicative model

- After reaching a specific number of operations without reaching the goal, we will start at this models.
- In this model, there will be communication between agents in making decisions to move from one state to another.
- This communication is based on asking questions by an agent in a current state to other agents about the decision they made when they were in that state **.**

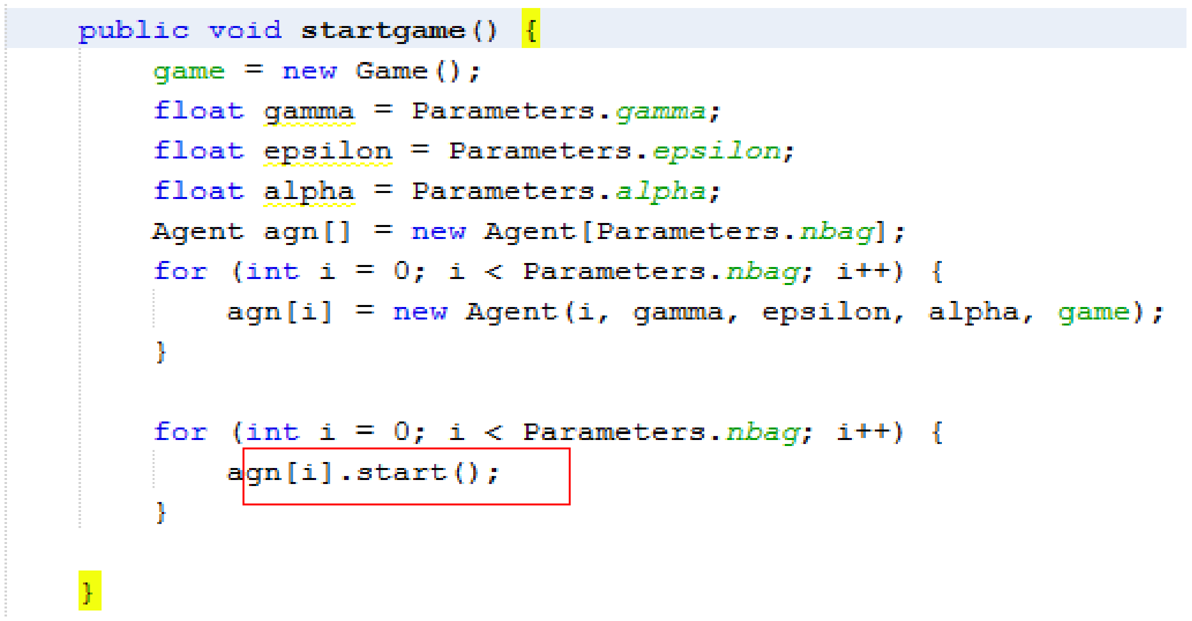
2.3 Adapt the Q-learning algorithm to work in MAS

Every one know that Q-Learning algorithm is intended to work in non-multi-agent systems, so in order to adapt it to work in a multi-agent system, we proposed a simple approach which is to use threads.

We used threads as agents and each one has an identifier that distinguishes him from the others

2.3.1 Using threads

The use of threads in our application was in the function <startgame> which is responsable to start agent to work by the command < . start > as the Figure shows



```
public void startgame() {
    game = new Game();
    float gamma = Parameters.gamma;
    float epsilon = Parameters.epsilon;
    float alpha = Parameters.alpha;
    Agent agn[] = new Agent[Parameters.nbag];
    for (int i = 0; i < Parameters.nbag; i++) {
        agn[i] = new Agent(i, gamma, epsilon, alpha, game);
    }

    for (int i = 0; i < Parameters.nbag; i++) {
        agn[i].start();
    }

}
```

Figure 22: code part for thread usage

2.3.2 The purpose behind using threads

The goal behind using threads is that they are easy to manipulate by virtue of our study of it over the previous years, in addition to the fact that the operating system monitors it automatically through the division of time and priorities, and any other matters related to it

2.4 Train the Models

All models are implemented through the same steps, but with a slight difference according to each model, such as the number of agents and the "common or private" Q-table.

Our next task is for our agent to learn about their environment by applying the Q-Learning model. The learning process will follow these steps:

1. Choose an infinite random state (white square) for the worker to start this new loop.
2. Choose an action (move up, right, down, or left) for the current position. Actions will be determined using the greedy Epsilon algorithm. This algorithm will usually choose the most promising action of the agent, but will sometimes choose a less promising option to encourage the agent to explore the environment.
3. Perform the chosen action, go to the next state (i.e. go to the next location).
4. Get the bonus of moving to the new state.
5. Update the Q value of the previous state and action pair.
6. If the new (current) state is a terminal state, then go to #1. Otherwise, go to #2.
7. This entire process will be repeated by a number of user-defined loops. This will give the agent enough opportunity to know the paths where the robot is allowed to walk, while avoiding hitting any of the obstacles at the same time!

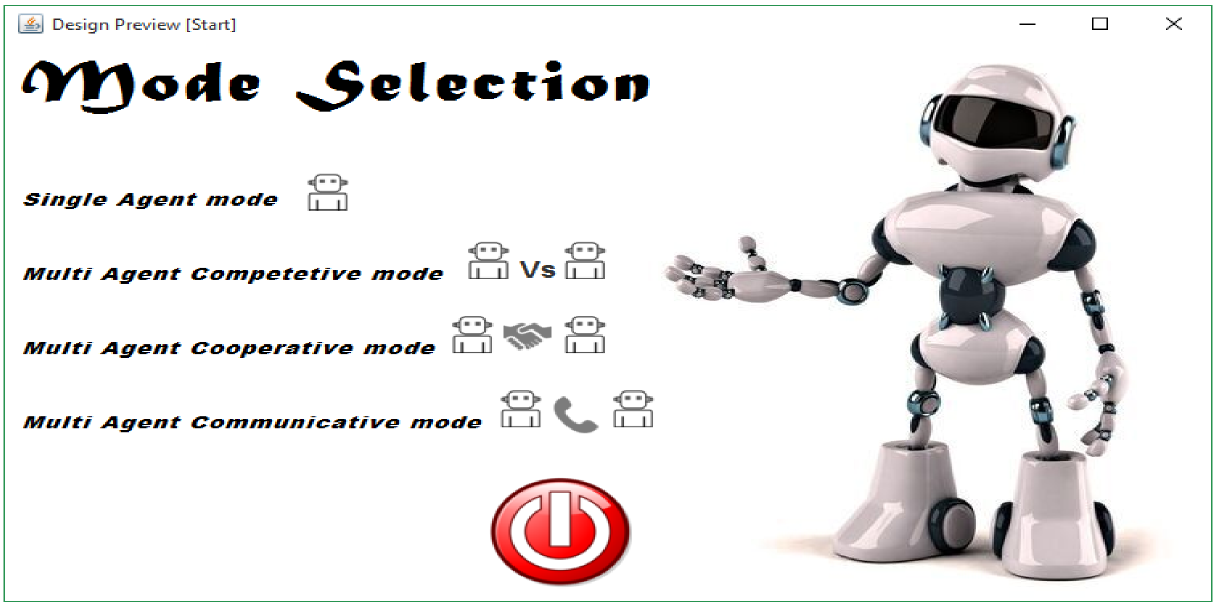

Design Preview [Start]
Mode Selection
Single Agent mode
Multi Agent Competetive mode
Vs
Multi Agent Cooperative mode
Multi Agent Communicative mode

Figure 23: Initial window

2.4.1 Single Agent model training

- After choosing single agent model, its window will appear
- The user enters the number of episodes in addition to the rest of Q-Learning's hyper parameters, maze height and width and goal position.
- We click "save" in order to save these parameters, a confirmation message will appear.
- Then click start button.
- The agent will seek to find the target, and after finding it, the results will appear in the output field
- To stop the model running click "stop" button and "back" to return at initial window

The figure bellow shows the operations cited above

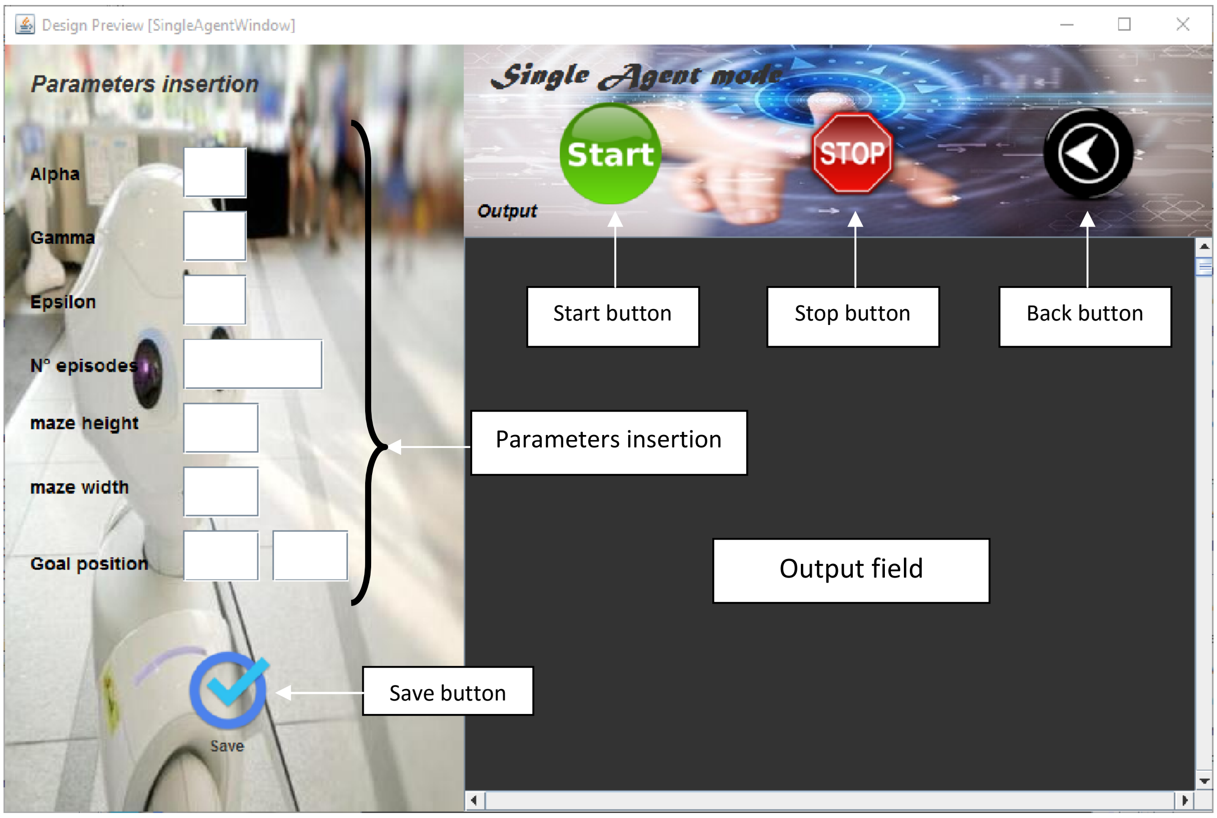


Figure 24: Single Agent model window

2.4.2 Multi agent competitive model training

- After choosing multi agent competitive model, its window will appear.
- The user enters number of episodes in addition to the rest of Q-Learning's hyper parameters, maze height and width and goal position, but this time user should enter the number of agents also.
- We click "save" in order to save these parameters, a confirmation message will appear.
- Then click start button.
- Agents will seek to find the target, and after finding it, the results will appear in the output field
- To stop the model running click "stop" button and "back" to return at initial window

NB: In this model, agents enter into a kind of competition, so each agent will search for the way alone without resorting to others.

What confirm this are the simulation results, where we often find that each agent takes his own path, except in some rare cases in which they take the same path, but this does not mean that they cooperate in searching for it, but rather the results of their search were similar. In addition to the results of the table for each agent where we find the results are not the same

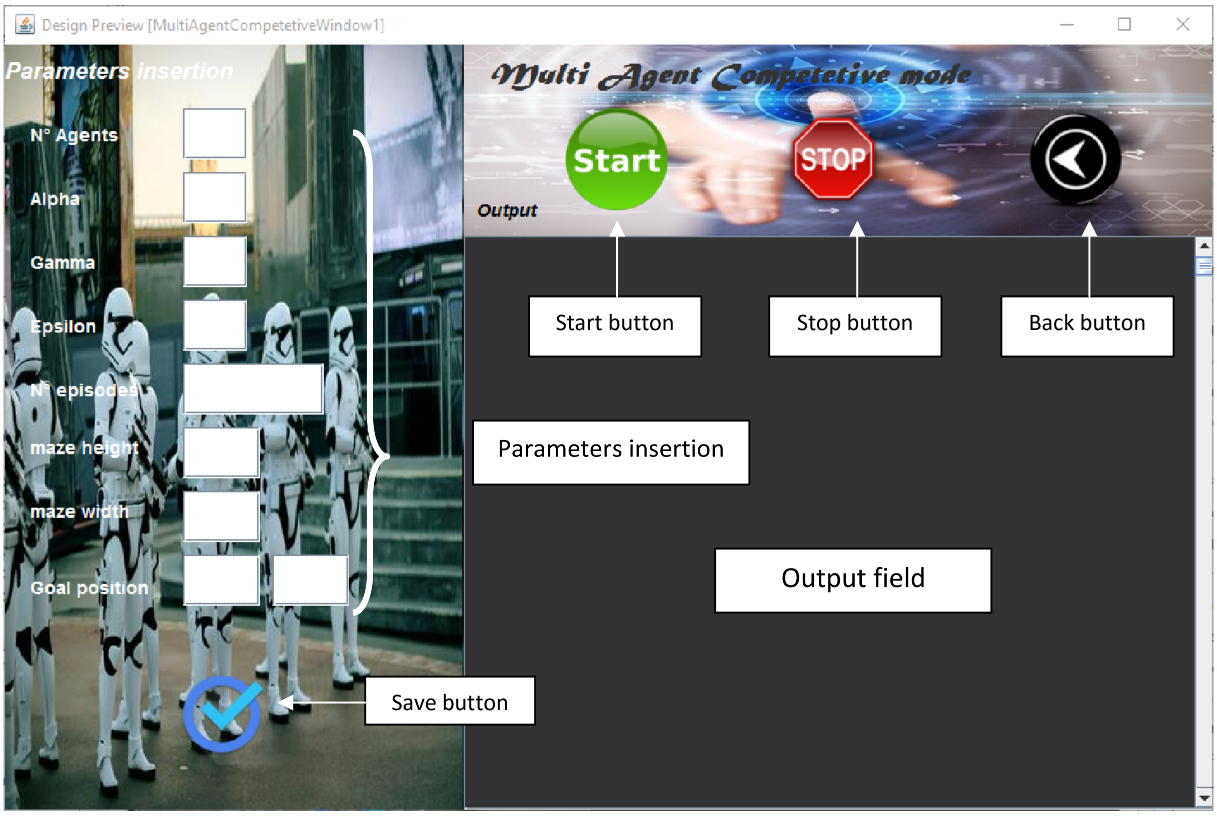


Figure 25: Multi Agent Competitive model window

2.4.3 Multi agent cooperative model training

- After choosing multi agent cooperative model, its window will appear.
- The user enters the number of agents, number of episodes in addition to the rest of Q-Learning's hyper parameters, maze height and width and goal position.
- We click "save" in order to save these parameters, a confirmation message will appear.
- Then click start button.
- Agents will seek to find the target, and after finding it, the results will appear in the output field
- To stop the model running click “stop” button and “back” to return at initial window

---

NB: In this model, agents cooperate to find the target.

What confirm this are the simulation results, where we find that all agents take the same path.

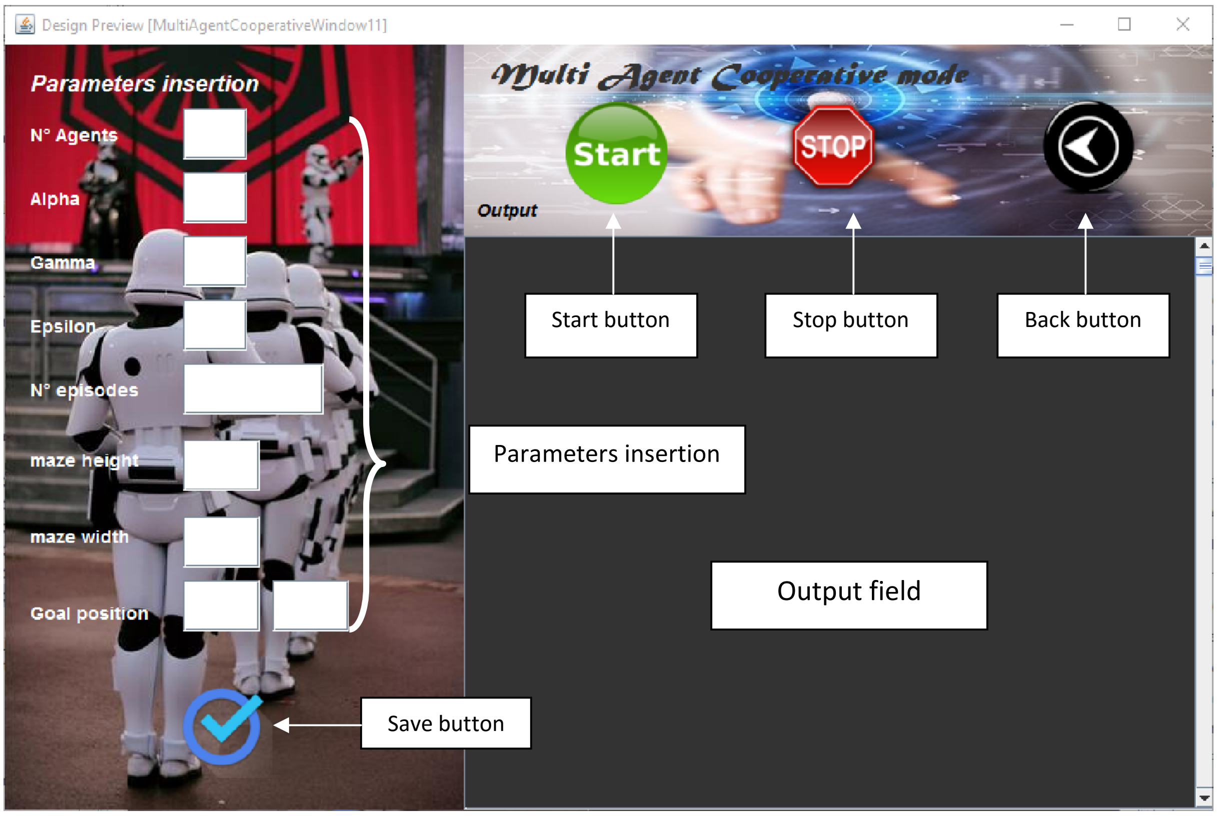


Figure 26: Multi Agent Cooperative model window

2.4.4 Multi agent communicative model training

- After choosing multi agent communicative model, its window will appear.
- The user enters the number of agents, number of episodes in addition to the rest of Q-Learning's hyper parameters, maze height and width and goal position.
- We click "save" in order to save these parameters, a confirmation message will appear.
- Then click start button.
- Agents will seek to find the target, and after finding it, the results will appear in the output field
- To stop the model running click “stop” button and “back” to return at initial window

NB: In this model, agents make communication between them to find the target.
What confirm this are the simulation results, where we find that some agents waiting in their current states to get information from the others in order to take the best decision to move.

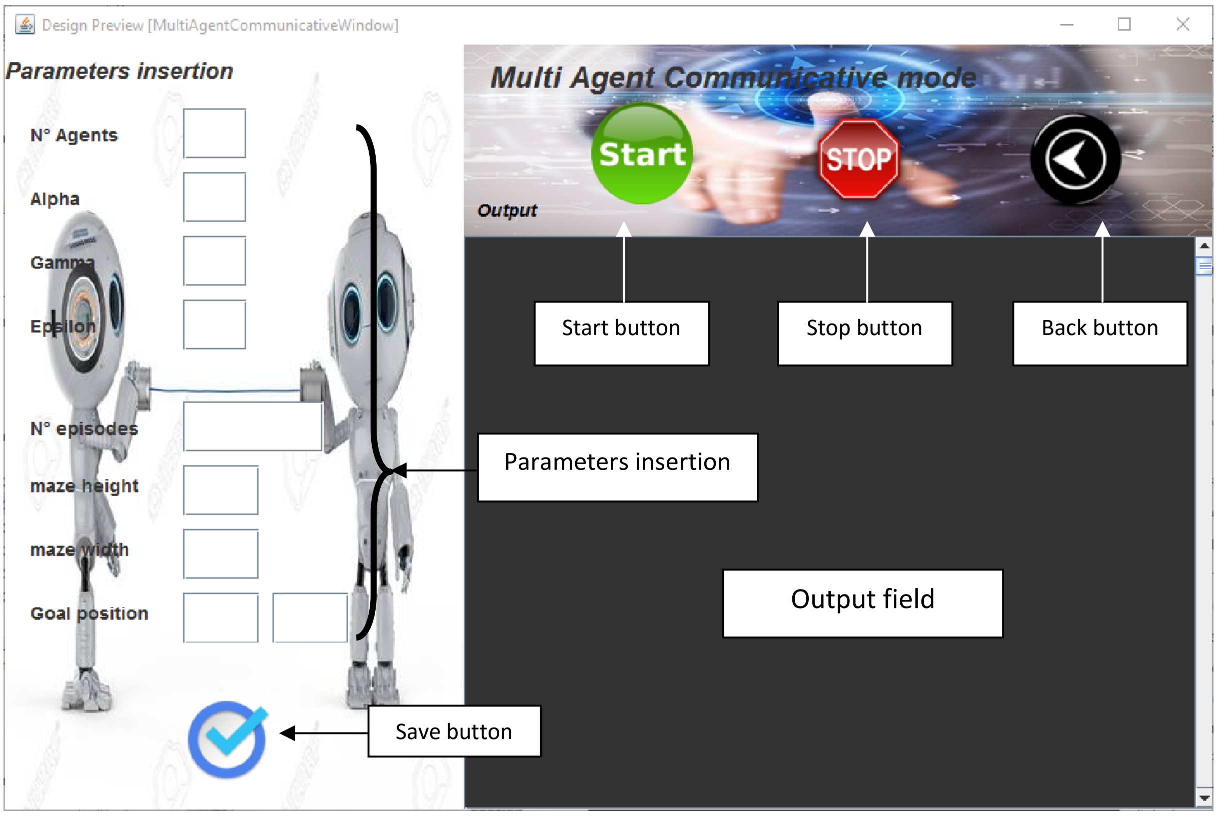


Figure 27: Multi Agent Communicative model window

3. Comparative study between models

In order to know which model was more efficient, we present a comparative study between our models. This study is based on two factors: the number of attempts and the time token by all agents to rich the goal state.

The following table represents the results of eight simulations for each model

Model	Simulation	Attempts	Time (ms)	Average attempts	Average time (ms)
Single agent	01	75345	4500	72828	4349,68
	02	72743	4345		
	03	73043	4363		
	04	72390	4324		
	05	74060	4423		
	06	73009	4360		
	07	72003	4300		
	08	70032	4183		
Multi agent competitive	01	75013	4480	76593	4574,38
	02	76103	4545		
	03	77752	4644		
	04	79068	4722		
	05	75639	4518		
	06	75107	4486		
	07	75871	4531		
	08	78194	4670		
Multi agent cooperative	01	12331	1980	12726	2043,35
	02	14956	5613		
	03	12553	2016		
	04	12907	2072		
	05	12091	1941		
	06	13339	2142		
	07	12936	2077		
	08	10691	1717		
Multi agent communicative	01	159555	25407	158262	25201,15
	02	160092	25493		
	03	157242	25039		
	04	154667	24629		
	05	164989	26272		
	06	160074	25490		
	07	153579	24455		
	08	155900	24825		

Table 02: Statistic comparative table between models

3.1 Observation

Through the simulation results, we note that the agents in the cooperative model made the least number of attempts in the least time, in contrast to the communicative model, which took a large number of attempts and time during the agents' search for the goal. As for the individual and competitive models, their results were very conflicting.

3.2 Interpretation

The results of the previous simulation can be interpreted as follows:

- **Cooperative model:** The search time and the number of attempts were few due to the presence of direct cooperation between the agents and their participation in the search for the target.
- **Communicative model:** the length of time can be explained by the presence of a time margin in the process of communication between agents. As for the high number of attempts, it is because the agent was originally trying some states, and when he communicates, he will try the states suggested by other agents.
- **Single and Competitive models:** the convergence of the results of these two models is because they are two models with the same architecture: the competitive model is a group of individual models that compete.

3.3 Result

From the foregoing, we conclude that the cooperative model was more effective in accomplishing tasks, followed by the individual and competitive models, and then the communicative model.

Conclusion

In this chapter, we have presented the implementation of the Q-Learning algorithm in different types of systems

First, we have presented the Q-Learning algorithm, its theoretical concepts and how to adapt it to work in multi agent system.

. We have opted for choices of tools and techniques that we have used.

Then, we have exposed the basic functionalities offered in our models through examples.

Finally, we have presented a statistic comparative study between our proposed models in order to know which model was more efficient.

Conclusion

As mentioned in the introduction, the point of view of artificial intelligence that we have adopted throughout this thesis is to design systems that operate in a rational manner (i.e. to achieve a goal), a system decided by the term "agent". This let us to take a first look at machine learning and specifically the field of reinforcement learning, which fits this definition well. But since we are also interested in the results that can be obtained from emerging phenomena, we also decided to work in the field of multi-agent systems starting with a single agent and applying inter-agent communication models.

Therefore, this thesis was an opportunity to use simultaneously the above propositions via the Q learning algorithm.

The mentioned algorithm is one of the most commonly used algorithms to solve the learning problem, which is based on the Q function, to ensure the creation of the last function and the proper operation of the system.

For the individual agent, it is designed in such a way that immediate behavior is taken, adapted to the environment and rewarded for taking the best possible comportment with it.

As for the multi-agents, a unique model that has always been difficult to achieve by using Q-learning, so we adapted the Q Learning algorithm via multi-threaded programming, which greatly helped us to implement this model and made it easy to adapt the algorithm to competitive and collaborative multi-Agent.

As for the communicative model, we found some obstacles, especially with regard to finding a way for agents to communicate effectively, so we proposed our own method that proved to be remarkably effective. It is worth noting that there is no unified or 100% effective communication system so far, as all communication systems are just ssuggested by developers

Finally, and as a result of the statistical comparative study between models, we arrived to know that the cooperative model was the more efficient one.

[01] Artificial Intelligence and Machine Learning Fundamentals by Zsolt Nagy

[02] Machine Learning Hands-On for Developers and Technical Professionals by Jason Bell

[03] An Introduction to Deep Reinforcement Learning Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare and Joelle Pineau (2018)

[04] Supervised-Learning-[SB-Kotsiantis]

[05] Kaelbling, Littman, & Moore, 96

[06] Reinforcement Learning: An Introduction Richard S. Sutton and Andrew G. Barto2014, 2015

[07] https://web.stanford.edu/class/cme241/lecture_slides/rich_sutton_slides/5-6-MDPs.pdf

[08] Hands-On Q-Learning with Python: Practical Q-learning with OpenAI Gym (English Edition) by Nazia Habib

[09] C.J.C.H. Watkins and P. Dayan. Q-learning. Machine Learning, 8(3–4):279–292, 1992

[10] Dr. Sunil Kumar Chinnamgari - R Machine Learning Projects_ Implement supervised, unsupervised, and reinforcement learning techniques using R 3.5

[11] ğurcan Özalp, Artificial Intelligence Engineer /Researcher- ugurcanozalp.medium.com

[12] Reinforcement Learning in a Nutshell V. Heidrich-Meisner, M. Lauer, C. Igel and M. Riedmiller,January 2007

[13] S. Russel, P. Norvig, “Artificial intelligence – A modern approach”, Prentice Hall, 1995.

[14]Introduction aux Systèmes Multi-Agents C. HANACHI, C. SIBERTIN-BLANC Université Toulouse I & IRIT.

[15] Agents and Multi-Agent Systems: A Short Introduction for Power Engineers -Technical ReportDr. Mevludin Glavic May, 2006

[16] P. Stone, M. Veloso, “ Multiagent systems: A survey from a machine learning perspective” , Autonomous Robots, vol. 8, no. 3, 2000.

[17] Tien C.Hsia and Michael Soderstrand, “Development of a micro robot system for playing soccer games,” In Proceedings of the Micro-Robot World Cup Soccer Tournament, pp. 149-152, 1996

[18] Balaji P.G and D.Srinivasan, “Distributed multi-agent type-2 fuzzy architecture for urban traffic signal control,” In IEEE Internationa Conference on Fuzzy Systems, pp. 1624-1632, 2009

[19] Lynne E.Parker, “Heterogeneous multi-robot cooperation,” PhD Thesis, Massachusetts Institute of Technology, 1994

[20] Lynne E.Parker, “Life-long adaptation in hetergeneous multi-robot teams:response to continual variation in robot performance,” Autonomous Robots, vol. 8, no. 3, 2000

[21] Rafal Drezewski and Leszek Siwik, “Co-evolutionary multi-agent system with predatorprey mechanism for multi-objective optimization,” In Adaptive and Natural Computing Algorithms, LNCS, vol. 4431, pp. 67-76, 2007

[22] J. Doran, S. Franklin, N. R. Jenkins, T. J. Norman, “ On cooperation in multi-agent systems” , In UK Workshop on Foundations of Multi-agent Systems, Warwick, 1996.

[23] Nigel Gilbert, « Computational Social Science », Agent-Based Modelling and Simulation in the Social and Human Sciences, Bardwell Press, 2007

[24] Ferber, P.52-54

[25] Budianto, “An overview and survey on multi agent system,” in Seminar Nasional “Soft Computing, Intelligent Systems and Information Technology” , SIIT 2005

[26] Choy, M C, D Srinivasan and R L Cheu, "Neural Networks for Continuous Online Learning and Control," IEEE Transactions on Neural Networks, vol. 17, no. 6, pp. 1511-1531, 2006

[27] M.C.Choy, D.Srinivasan and R.L.Cheu, “ Cooperative, hybrid agent architecture for realtime traffic signal control,” IEEE Trans. On Systems, Man and Cybernetics-Part A: Systems and Humans, vol. 33, no. 5, pp. 597-607, 2003

[28] Balaji P.G, D.Srinivasan and C.K.Tham, “ Coordination in distributed multi-agent system using type-2 fuzzy decision systems,” in Proceedings of IEEE International Conference on Fuzzy Systems, pp. 2291-2298, 2008

[29] Susan E.Lander, “Issues in multiagent design systems,” IEEE Expert, vol.12, no. 2 , pp. 18-26, 1997

[30] Multi-Agent Reinforcement Learning Algorithms Natalia Akchurina Dissertation in Computer Science submitted to the Faculty of Electrical Engineering, Computer Science and Mathematics University of Paderborn in partial fulfillment of the requirements for the degree of doctor rerum naturalium (Dr. rer. nat.) Paderborn, February 2010

[31] L. Bus¸oniu, R. Babuˇ ska, and B. De Schutter, “Multi-agent reinforcement learning: An overview,” Chapter 7 in Innovations in Multi-Agent Systems and Applications – 1 (D. Srinivasan and L.C. Jain, eds.), vol. 310 of Studies in Computational Intelligence, Berlin, Germany: Springer, pp. 183–221, 2010.

[32] Pierre Haou Multi-Agent Reinforcement Learning (MARL) and Cooperative AI

[33] Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. (1993)

[34] Clouse, J.: Learning from an automated training agent. In: W orking Notes Workshop on Agents that Learn from Other Agents, 12th International Conference on Machine Learning (ICML-95). Tahoe City, US (1995)

[35] Price, B., Boutilier, C.: Accelerating reinforcement learning through implicit imitation. Journal of Artificial Intelligence Research 19, 569–629 (2003)

[36] Sutton, R.S.; Barto, A.G. Reinforcement Learning I: Introduction; MIT Press: Cambridge, MA, USA, 1998

[37] Bus¸oniu, L., Babuˇ ska, R., De Schutter, B.: A comprehensive survey of multi-agent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Reviews 38(2), 156–172 (2008)

[38] L. Lhotska, "Learning in multi-agent systems: Theoretical issues," Computer Aided Systems Theory – EUROCAST'97, LNCS-1333, pp.394-405, 1997

[39] Gomez.F, Schmidhuber.J and Miikkulainen.R, "Efficient non-linear control through neuro evolution," Proceedings of ECML 2006, pp. 654-662

[40] Jiming Jiu, Autonomous Agents and Multi-agent Systems, World Scientific Publication

[41] P.G. Balaji and D. Srinivasan An Introduction to Multi-Agent Systems Department of Electrical and Computer Engineering National university of Singapore

[42] A beginners Guide to Q-Learning, Model Free Reinforcement Learning By Chathurangi Shyalika, 2019.