

Theoretical and Mathematical Concepts Formulation behind the Development of AI-ML Algorithms for Path Planning in Robots

Sanjay Sidu Jadhav*¹, Prashantha GR²

¹Research Scholar, VTU RC, Belagavi, Karnataka

²Jain Institute of Technology, CXM5+48C, Davanagere, Karnataka 577003

* Corresponding author's Email: sanjaysaspade@gmail.com

Abstract: In this paper, a brief insight into the theoretical & mathematical modelling concepts of the various types of algorithm development are presented in a concise manner to achieve the desired task, i.e., design of the robot path to move to the destination from the source to the goal in spite of obstacles. Various theoretical concepts of AI & ML algorithms such as the Reinforcement learning, Q-Learning, Deep Learning, Supervised Learning, Un-supervised learning, etc. are being proposed, which are discussed one after the other in a nutshell, first the theoretical concepts are being presented, this is followed by the algorithm development, then the simulated results. The work is divided into a number of sections in the forth coming research papers, where the theoretical concepts of AI & ML presented here are used to develop the hybrid algorithms for the robot's path planning from the (S) to the ending or (G) point. The article is developed in 2 phases, part I & II. The part-I is presented in this research paper, whereas Part II of this research paper is extended to as another research article. The paper concludes with the overall conclusions of the work done in this research article.

Keywords: AI, ML, DL, Robot, Obstacle, Avoidance, Path, Plan, Goal, Source, Destination, Collision

1. Artificial Intelligence

Artificial Intelligence [AI] is defined as that part of CS & IS with the characteristics with which we associate with intelligence in human behaviour - understanding languages, learning, reasoning, problem solving and so on and is concerned with understanding the nature of intelligent action to perception. It is also defined as the ability of a device to perform functions that are normally associated with human intelligence, such as reasoning, planning, problem solving, pattern recognition, perception, cognition, understanding & learning, possession of sense such as sight, light, sound, touch & feel. The uses of AI are - the collective attributes of a computer, robot or other device capable of performing functions such as learning, decision making, or other intelligent human behaviours [1].

The science and engineering of creating intelligent machines is known as Artificial Intelligence or the AI. One way of interpreting AI is the study of computer systems that aim to model and apply the intelligence of the human mind. AI may also be stated as the ability of a machine to imitate behaviour of human beings in a intelligent manner. AI is made up of two words: artificial and intelligence. Artificial is something that is not natural and was produced by people. The ability to comprehend, reason, and plan is referred to as intelligence. AI is defined as any programming, technology, or algorithm that allows a machine to mimic, develop, or display human cognition or behaviour [2].

The aim of Artificial Intelligence is to develop robotic systems that appear to behave intelligently or think like humans. Often this is described in developing robots that think like us. Modern robot control concepts such as the vision, speech, tactile sensing have their roots in AI research. Robot can be considered as a fully automatic machine (an

artificial human being) working with Artificial Intelligence. We are the natural human beings working with natural intelligence. The more amount of sensors, memory, learning algos we incorporate, the more intelligent a machine or a robot becomes. The examples could be the Crays, androids, humanoids, terminators, walking mechanisms, bipeds, unipeds, Asimov, Honda robots and the AIBO's are the robots working with AI. Thus, AI and robotics always go together. The possible applications of AI are shown in the Fig. 1 & 3 [3].

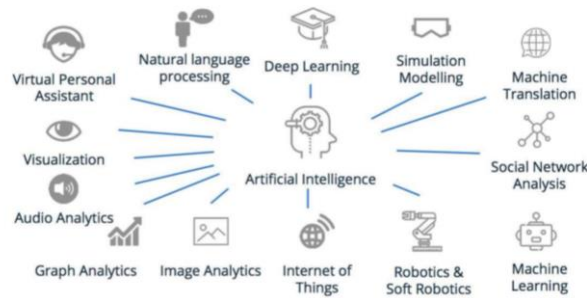


Figure. 1. Possible applications of AI

In Fig. 1, the 15 goals of AI & Robotics are mentioned such as: Planning, Problem solving, Natural intelligence, Machine intelligence, Understanding, Cognitive, Learning, Vision, Speech, Reasoning, Fuzzy logic, Behaviour, Expert systems, Neural networks, Programming, Natural Language Processing [4].

2. Machine Learning

Machine Learning (ML) is a sub-field of AI (Artificial Intelligence) which will provide the system the ability to learn automatically & then to improve the system performance from the learning experience w/o being explicitly programmed (w/o Human Interactions). Often, this will focus on the development of the computer algorithms which could access the data & use it learn for themselves (similar to humans). ML is an AI application which will give the system more stability to learn by automatic means & then to improve from experience. ML enables the computers to tackle tasks that have learnt, until now, only been carried out by people. The basic goal is for computers to learn on their own, without the need for human involvement, and to change their behaviour correspondingly [5]. The conceptual view of ML Algo is shown in the Fig. 2 [6].

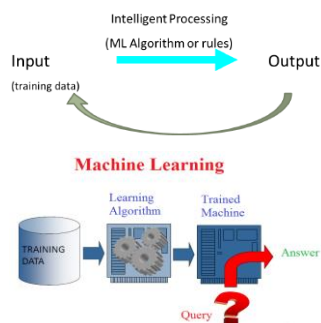


Figure. 2. Conceptual view of ML Algo

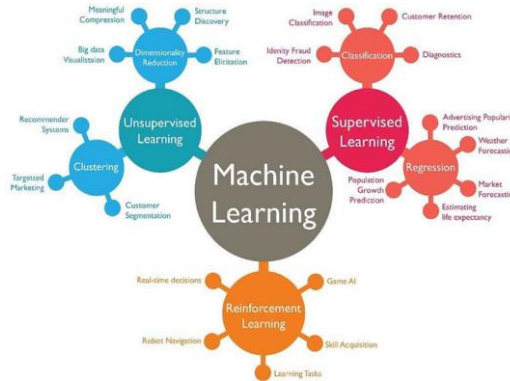


Figure. 3. Applications of Machine Learning of AI & its types

The types of ML concepts are graphically shown in the Figs. 4 & 5 respectively (along with the basic types & the abridged ones) [7].

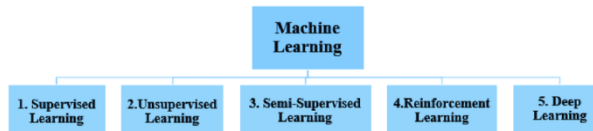


Figure. 4. Abridged (expanded) types of machine learning

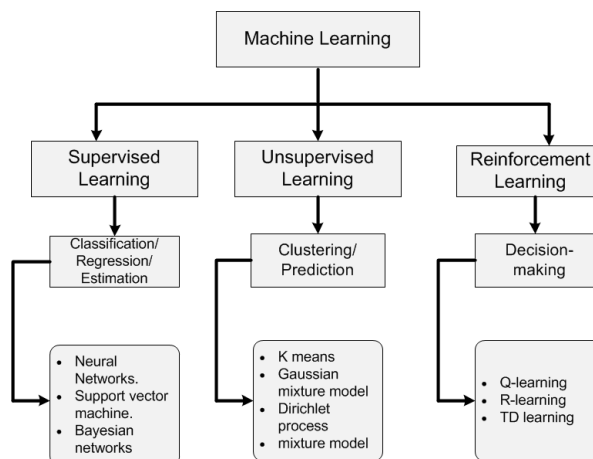


Figure. 5. Three basic types of machine learning algorithms & its sub divisions

Stages of development in machine learning as on date is shown in the Fig. 6. ML is related to the set of programs written in any computer language that will automatically improve their performance through experience. ML is a field of study that gives the computers the ability to learn without explicitly being programmed as stated by Arthur Samuel in his book. In simple terms, ML means making the predictions based on the datas. A program written by the PC or computer will learn from its experiences E w.r.t. some type of classes of task ' T ' its measure of performance ' P ' if the performance of the system at tasks in T , as it is measured by ' P ', thus improving with its experience E is a concise definition of ML as stated by Tom Mitchell. The main advantages of ML are [8].

- Learning and writing an algorithm
- It is simple for a human brain to classify objects, but it is difficult for a computer to do so. It takes time and a large amount of training data for a machine to correctly classify items.
- Implementation and automation

- For a machine, this is simple. A machine, once trained, can process one million photos without tiring out, whereas the human brain cannot, which is why machine learning and big data are such a deadly combination.

The enormous amount of data produced by applications, the rise in computer power in recent years, and the development of stronger algorithms for understanding, cognition, and recognition have all contributed to ML being a buzzword in recent years [9]. (1) Computer Vision, (2) Imitated Learnings, (3) Supervised / Un-Supervised / Reinforcement Learning, (4) Assistive & Medical Technologies, (5) Multi-Agent type of Learning mechanisms.

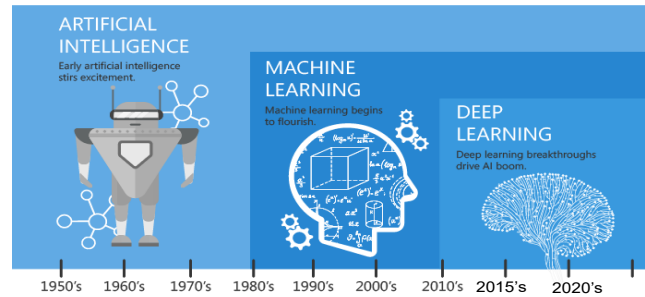


Figure. 6. Stages of development in machine learning as on date

Some examples of ML in use are [10]: *Prediction* — The ML could be employed for predicting the behaviour of the system. In the case of fault in the system, the automated frame work or the system will have to classify the available data into categories in order to determine the chance of a fault. *Picture recognition* — Machine learning may also be used to recognise faces in images. Each person in a database of numerous people has their own category. *Speech Recognizing* – Could be termed as the method of conversion of the audio words into textual matter. It is used in search of voices, audios, songs, speech dialling, routing of the calls, appliance control, etc...could be termed as few of the example of voice user interface. It could also be utilized for entering some simple date & then creating some structured documents. *Medical diagnostics* – ML has been taught to spot malignant tissues. Companies employ machine learning in fraud investigations and credit checks in the financial industry and trading.

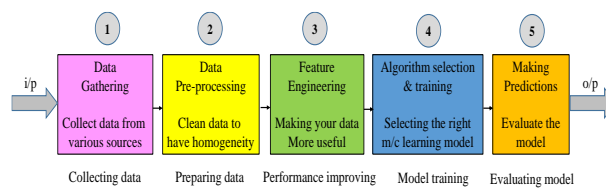


Figure. 7. Five important steps in any ML algorithm development

The 5 steps to solve a machine learning problem is graphically displayed as shown in the Fig. 7. There are 3 main types of ML algorithms, viz. [11], (i) Supervised type of Learning, and (ii) Unsupervised type of Learning & the Reinforcement type of Learning.

3. Supervised Type of Learning (SL)

In supervised type of learning, an Artificial Intelligence module is supplied with a number of information's (datas) which will be labelled, in the sense that every bit of data will be assigned a correct label depending upon its value. Then, the goal will be to estimate the mapping function upto the juncture where one could forecast the output variables (Y) for a new

type of input data (x). Supervisory learning mechanisms are shown in the Figs. 8 –10 respectively [12].

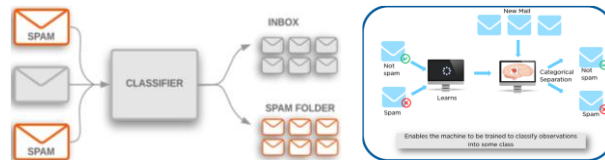


Figure. 8. Supervisory learning mechanisms

We've taken some data and labelled it as 'Spam' or 'Not Spam,' as seen in the following example in Fig. 8. The training supervised model uses this tagged data to train the model. We can test our model after it has been trained by sending it some test new emails and seeing whether it can anticipate the correct outcome. Classification - When the o/p variables are of such a category, such as "red" or "blue" or "illness"& "no disease," then classification problems will exist. &when the o/p variables are of real values, such as the dollar (money) or the weights, then a regression problem exists [13].

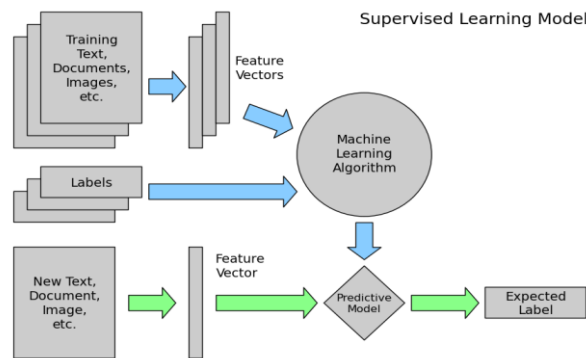


Figure. 9. Supervised learning model

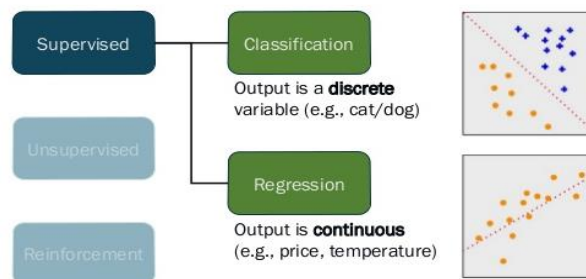


Figure. 10. Types of supervised learning

4. Un-Supervised Learning

Unsupervised learning is when an Artificial Intelligence module is given an un-labelled, un-categorized data of information & its algorithm will act on it w/o any type of initial trainings. Then, the outcome is found out by the type of algo that has been coded. The un-supervised type of learning could be treated as one of the methods of putting the Artificial Intelligence into testing process. We've given our model some characters labelled 'Ducks' and 'Not Ducks.' We will not supply any labels to the concurrent data in our training data. The unsupervised model may distinguish between the characters (ducks and not ducks) based on the type of data and the model the data which rare in the under lying structured distribution

Article Received: 05 September 2021, Revised: 09 October 2021, Accepted: 22 November 2021, Publication: 26 December 2021

where more learning can be done, in the sense lot of training models could be incorporated. There is no o/p that is desired.

The various sorts of USL are as follows: Clustering - This clustering problem will occur when one wants to find the data's natural grouping, i.e., the classifying clients based on their shopping habits. The second difficulty is association. a rule of association When you have a learning problem, one wants to find rule that will describe the substantial chunks of the learnt data, such as persons who will get X &also he will buy Y. Fig. 11 depicts the unsupervised learning model and its various forms [14].

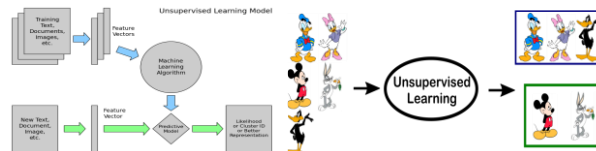


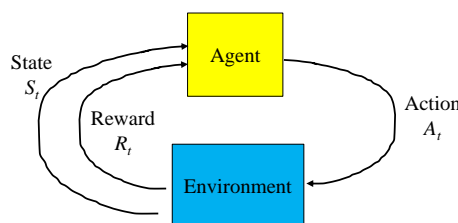
Figure. 11. Un-supervised learning model & its types

5. Deep Learning

It is part of a larger group of ML approaches that combine ANN with representation learning. Un-supervised, semi-supervised, and supervised learning are the three types of learning. Deep learning is Artificial Intelligence techniques that use neural networks to learn un-supervised from un-structured or un-labelled data (deep neural learning or deep neural networks). Machine learning includes 'DL'. By filtering inputs across layers, it teaches a computer to learn how to predict and classify information. Observations can be expressed through images, words, or music. DL was inspired by the way the human brain filters information [15].

6. Reinforcement Type of Learning

In the current AI scenario, the most advanced disciplines of AI is the Reinforcement type of Learning. RL's purpose is to increase anperson's reward by doing a series of behaviour when a response occurs in a complex environment. Reinforcement type of Learning is the method of doing the best judgments based on prior experience. An agent, or reinforcement learning system, learns by interacting with its surroundings. The agent is rewarded for proper performance and penalised for incorrect performance. By maximising reward and decreasing penalty, the agent learns without the need for human interaction. RL is a type of dynamic method of programming that teaches algorithms through a reward and punishment mechanism [16].



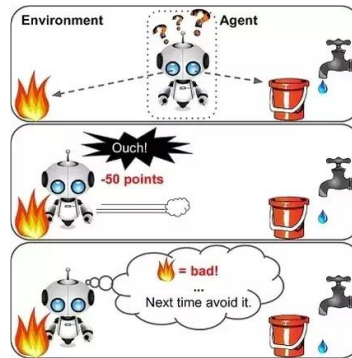


Figure. 12. Reinforcement's basic components agents, learning methodologies in the environment, rewards & actions

We can see that the agent is offered two paths to choose from: one with water and one with fire. A reinforcement algorithm works by rewarding a system; for example, if the agent takes the fire path, the rewards are deducted, and the agent tries to learn that the fire path should be avoided. If it had picked the water way or the safe path, some reward points would have been added, and the agent would then try to figure out which path is safe and which is not. Essentially, the agent increases its environment knowledge to select the next action by leveraging the incentives received. Breaking it down, the Reinforcement Learning process consists of these essential components as shown in Fig. 12, as well as their application shown in Fig. 13 [17].

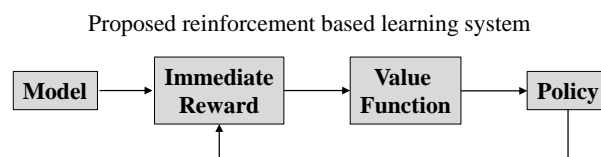


Figure. 13. Working of reinforcement learning

- Observations of the surroundings
- Using a strategy to decide how to act.
- Taking appropriate action.
- Getting a prize or a punishment.
- Applying what we've learned and fine-tuning our strategy.
- Iterate until you find the best strategy.

With RL, there are two sorts of algorithms. They are as follows: 1. model-depended, 2. model-devoids. An algorithm that predicts the best policy without using or estimating environmental dynamics is known as a model-free algorithm (transition and reward functions). An algorithm based on a model, on the other hand, is one that estimates the best strategy using the transition function (and the reward function). It's helpful to have a clear picture of the problem that will be solved by reinforcement learning, in this case Q-Learning. It aids in the identification of the major elements of a reinforcement learning system, such as agent, environments, action, reward, and states. Figures 14 and 15 exhibit examples of reinforcement learning and action taken, respectively [18].

The problem statement is then defined as follows. We're going to make a couple of automatic robot/s for guitar manufacturing plant in a industry. Such type of guitar manufacturing robots will help the guitar workers by delivering the necessary guitar parts for

them to create a guitar. These parts can be found in nine distinct locations around the production warehouse. Guitar parts include polished wood fretboard sticks, polished wood guitar bodies, and guitar pickups, among others. The first priority for the luthiers is the place that contains body woods. They've also supplied priority for other sites, which we'll look at shortly. These are the sites within the factory warehouse [19].

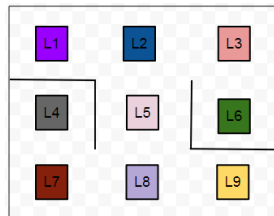


Fig. 14 : Example for RL

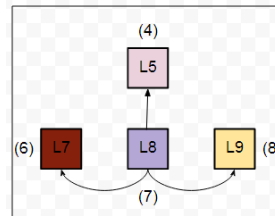


Fig. 15 : Example for action taken in RL

As can be seen, there are minor barriers (represented by smoothed lines) in the way of the places. L6 is the most important site since it houses the polished wood used to prepare guitar bodies. The goal now is to enable the robots to find the shortest route from one site to another on their own. In this example, the agents are robots. The setting is the warehouse of a guitar factory [20].

States include: The states are represented by the locations in table 1. The state of a robot is determined by the position in which it is present at any given point in time. Numbers are better understood by machines than characters. So, let's convert the area codes to numbers [21].



Figure. 16. Grid search using bellman equation

Table 1 : State location table

Location code	State	Location code	State
L-1	0	L-6	5
L-2	1	L-7	6
L-3	2	L-8	7
L-4	3	L-9	8
L-5	4		

The actions: The actions in our case could be stated as one direct destination that any automatic machine (robot) can reach from a given position. Considering the situation when a robo will be at the position no. L8 & the only places it may go are L5, L7, and L9. The diagram below may be helpful in visualising this. The group of action/s that could be set is simply the set of all conceivable states of the robot. Then, the collection of actions that a robot may perform will differ depending on the location. If the robot is in L1, for example, the set of activities will vary [22].

The reward: Now, there are 2 groups as shown in the rewards table 2 as [23].

A group of state/s: $S = 0$ to 8 & $S = 0, 1, 2, 3, 4, 5, 6, 7, 8$.

A set of actions: $A = 0, 1, 2, 3, 4, 5, 6, 7, 8$.

Table 2 : Reward table

Code	L1	L2	L3	L4	L5	L6	L7	L8	L9
L-1	0	1	0	0	0	0	0	0	0
L-2	1	0	1	0	1	0	0	0	0
L-3	0	1	0	0	0	1	0	0	0
L-4	0	0	0	0	0	0	1	0	0
L-5	0	1	0	0	0	0	0	1	0
L-6	0	0	1	0	0	0	0	0	0
L-7	0	0	0	1	0	0	0	1	0
L-8	0	0	0	0	1	0	1	0	1
L-9	0	0	0	0	0	0	0	1	0

Now, a robot will be rewarded if the localized set point (read it as the state) will be directly approachable from another site. Consider the following scenario. L9 can be reached straight from L8. As a result, if a robo advances from position L8 to position L9 and then do it vice-versa, it could be awarded with 1. We do not award any points if a destination is not directly accessible from another (a reward of 0). Yes, the award is simply a number in this case. It allows robots to understand their movements and determine which sites are immediately attainable and which are not. We can create a reward table using this cue that has all of the reward values that map to all of the potential states (locations). Fig. 16 shows the grid search using Bellman's equation [24].

All of the possible prizes that a robot can obtain by travelling between the different states are listed in table 2 above. Now comes a difficult decision. Remember how the luthiers placed L6 at the top of their priority list in the preceding sections? So, how can we include this information in the table above? This is accomplished by linking the highest priority site with a significantly higher reward than the others. Put the number 999 in the cell (L6, L6): We have now explicitly specified all of the critical components of the solution we are pursuing for the aforementioned problem. We'll change gears for a while and look at some of the core concepts in reinforcement learning [25].

7. The Bellman Equation's Design

The AI equation designed by Bellman laid the foundation of reinforcement type of learning & is found across the field. It assists us in resolving MDP. Solving entails determining the best policies & the value f/n's. $V^*(S)$ is the best value f/n because it returns the most value. Then, the given state's value will be equal to the best action's reward in that state/s multiplied by the value of the following state calculated using the Bellman Equation [26].

$$V(S) = \max_a [R(s, a) + \gamma V(s')]$$

Let's break down this math model: $V(s)$ will be the value the particular state in the learning model. After taking action a , $V(s')$ is the value for being in the next state. The reward we receive after taking action a in states is $R(s,a)$. As different actions can be taken, the one with the maximum value is considered because the agent (robot) is being set in the state of optimality. γ could be termed as the discounted factor. The bellman equation in the deterministic

environment is actually different from the non deterministic type of environment or some stochastic type of environments [27].

$$V(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s,a,s') \cdot V(s') \right)$$

Table 3 :
An empty environment

Table 4 :
Sample environment

↑		
↑		
↑	←	A

Table 5 :
Environment with value footprints

1		
1		
1	1	1

When we execute an action in a stochastic environment, it is not guaranteed that we will end up in a specific next state, but there is a chance that we will. The probability of terminating in state s from s by taking action an is P(s,a,s). This results in a total no. of possible states of the future action that could be initiated. Consider one ex., if we take an action (work), one could end up in one of three states: s1, s2, or s3 with probabilities of 0.2 or 0.2 to 0.6. Then, the Bellman formula could be modified as [28].

$$V(S) = \max_a [R(s,a) + \gamma \cdot (0.2 \cdot V(S_1) + 0.2 \cdot V(S_2) + 0.6 \cdot V(S_3))]]$$

The Bellman equation can be solved using a technique known as dynamic type of programming. Now, we think about a square of rooms, which will be similar to our initial problem's actual surroundings but without the obstacles. In table 3, an empty environment is illustrated, followed by a sample environment in table 4, and then an environment with the value of the footprints in table 5 [29].

Assume a robot needs to travel to the green-colored chamber from its current location (A) in the provided direction. How can we programmatically enable the robot to accomplish this? One suggestion is to provide a type of footprint that the robot can follow, such as the one illustrated in Fig. 17. In each of the rooms, a constant value is supplied that will be encountered by the robot if it follows the above-mentioned path. If it begins at place A, it can scan thro' the constant value & then move it forward as necessary. However, this would only work if the robot's orientation is predetermined and it always begins at site A. Consider the following scenario: the robot starts at the position A in Table 6. Then, the environment with further footprints values can be predicted as shown in the table 7 with the values obtained from the Bellman's equation depicted in the table 8 [30].

Table 6 :
Environment with further footprint values

1		
1		
A	1	1

Table 7 :
Environment with further footprint values

1		
*		

Table 8 :
Environment with footprint values using Bellman's equation

1		
0.		
9		

0. 81	0.7 29	Star ting poin t
------------------	-------------------	-------------------------------------

The robot can now see footsteps coming from two directions. As a result, it is unable to choose which path to take in order to reach its target (green room). It occurs mostly because the robot lacks the ability to remember how to proceed. So now it's up to us to give the robot a memory. The Bellman Equation is used in conjunction with the mathematical model provided by the math model as [31].

$$V(S) = \max_a [R(s,a) + \gamma V(s')] \quad V(s) = \max_a (R(s,a) + \gamma V(s'))$$

where, s = a typical states (say, a room), a = action (move between the rooms), s' = robot goes from initial state to the final state, γ = discounted value of the factor (it will be arrived at that state in a fraction of a second), $R(s,a)$ = a rewards f/n which will be taking the state & actions and then will be outputting a particular rewarded values, $V(s)$ = actual asset value of the considered state or the footsteps.

We weigh all of the options and choose the one that provides the most benefit. There is one constraint, however, in terms of the value footprint: the yellow room directly below the green room will always have a value of 1 to indicate that it is one of the closest rooms nearer to the green type of room, which is to ensure that when a robot moves from the yellow to the green chamber, it is rewarded. Let's look at how to interpret the above equation presently. A discount factor of 0.9 will be used. For this room (reading state) what will be the value of $V(s)$? Now, insert the values in the Bellman's equation directly as shown below [32].

$$V(s) = \max_a (0 + 0.9 * 1) = 0.9 \quad V(s) = \max_a (0 + 0.9 * 1) = 0.9$$

Because the robot will not be rewarded for visiting the state (room) designated in the yellow, then, $R(s, a) = 0$. $V(s')$ is 1 because the robo will know the values of the agents in the yellow rooms. We should get [33] if we follow this for the other states:

Here are a few things to keep in mind: The $\max()$ function assists the robot in selecting the condition that provides it with the greatest benefit. The discount factor informs the robot of its location in relation to the destination. The creator of the algorithm that would be implanted in the robot would usually specify this.

The different types of the states could also be assigned some standard parametric values in a similar way with the environmental values with all the footprint values obtained from the Bellman's equation as depicted in the table 9 with the graphical display of the deterministic & non-deterministic searches of the best optimized path as shown in the Fig. 17 [34].

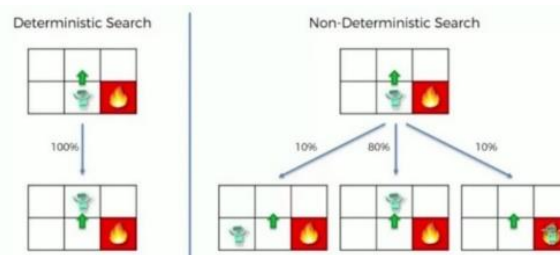


Figure. 17. Deterministic and non-deterministic search using Bellman equation

Table 9. Environment with all footprint values using Bellman equation

	1	0.9
1	0.9	0.81
0.9	0.81	0.729
0.81	0.729	0.66

The robot may now navigate the green room using these valued footprints, even though it will be dropped in any random rooms in the square as shown above in the table. If a robo arrives in the highlighted (sky-blue) rooms now, it will be presented with two possibilities. Because of the way the value imprints are currently put out, either path will eventually be good enough for the robot to take.

Take a note of how the background work of getting to a certain goal starting from a specific starting point was broken a no. of similar sub-tasks, i.e., the complicated problem will be cut down into a no. of smaller problems. It will turn out to be a certain programming concept that would be created specifically for tackling problems with recurring sub-problems. Dynamic programming is the term for it. Richard Bellman, who also created the equation we just looked at, invented it in 1954, therefore the name given to the above equation is the famous ‘Bellman’s Equation’, which is one of the most important equation in the reinforcement type of learning [1].

When we think about it, our surroundings don't always work the way we expect them to. There is usually some element of chance involved. This also applies to a robot. It is possible that the robots internal mechanism will become corrupted at some point. On its journey, the robot may encounter several obstacles that it was not aware of earlier. Even if the robot is aware that it needs to make a right turn, it will not always do so. So, how can we include this element of chance in our situation? processes of Markov decisions [2].

8. Deterministic and non-deterministic search techniques


Deterministic search is the one where the action taken does not involve any randomness for example if the decision taken is to move right then the agent moves right without any randomness. Deterministic search cannot be used in real time application which involves some constraints. Non deterministic search involves randomness in the actions taken and is most often used in real time applications and we are using non deterministic search method in our research work. A Markov process is a random process in which the future will not be dependent on the present as the present value is given. The value of future state depends only on present state but not on how the agent got there (path taken). The entire process could be graphically viewed in the Fig. 17 [3].

$$0.8 * V(s_1') + 0.1 * V(s_2') + 0.1 * V(s_3')$$

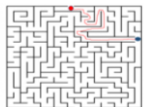
$$V(s) = \max_a (R(s, a) + \gamma \cdot V(s'))$$

$$V(s) = \max_a \left(R(s, a) + \gamma \sum_{s'} P(s, a, s') \cdot V(s') \right)$$

Path 1



Path 2



- Penultimate state
- Starting state

Figure. 18. Path following using Markov’s decision process

An important point to remember – each state within a system is a consequence of its preceding state, which in turn results from its preceding state. However, processing all of this knowledge would become readily infeasible even for environments with short episodes. Considering, each state follows the Markov properties, viz., every state will depend independently on the prior state or level & the transition from that state to the current state. Let’s check out the maze below to better understand how this works [4].

Table 10 : Environment with an agent

		O	

Table 11 : Stochastic environment with an agent

?	?	→	
←	↑	O	?
	↓		
	?		

Table 12 : An environment with an agent (with probabilities)

0.0	0.8	→0
5←	↑	.05
	O	
	↓	
	0.1	
	?	

Now, there are 2 scenarios with 2 different starting points, and the agent crosses various paths to reach the same penultimate state. Let us consider the environment with an agent as shown in the Table 10. Then the stochastic environment with an agent could be obtained as shown in the Table 11. Finally, using these 2 previous tables, the environment with an agent (with probabilities) can be computed as shown in the Table 12. Now it doesn’t matter what direction the agent must take to get to the gold [5].

The next step to get out of the labyrinth and attain the last state is to go right. Of course, we just needed the red / penultimate state knowledge to figure out the next best move which is exactly what the Markov property means. Suppose that the robot is presently in the red area and wants to be transferred to the green room. Consider the possibility that the robot will malfunction and will take to the left position, right, or the bottom turnings instead of the upper turnings for getting to the green type of room from where it started presently (red rooms). Now the challenge is, how can we get the robot to deal with this while it's out in the real world? [6].

This is a circumstance in which the robot's decision on which turn to take is half random and partially under his control. Partly at random since we don't know when the robot may malfunction, & some part under the control of the robot, since it is still deciding to take a turn on its own and with the assistance of the programme contained in it. Markov Decision Processes are defined as follows. A DT control of the stochastic type of process is known as a Markov decision process (MDP) [7].

This will give a mathematical foundation for modelling the different type of decision makings (if, then, jump, else instructions) in the settings where the types of outcomes that are obtained are random in nature and partially controlled by the decision maker. We've now covered the concept of partially random and partially regulated decision type of making the predictions. Then, this idea has to be given a mathematical interpretation form (possibly an equation) so that it may be developed further. It's surprising to learn that any one can do the work with the famous Bellman’s equation & a little simple type of manipulations or variations or adjustments. Then, the original Bellman’s equation could be re-modelled incorporating all the previous parameters as.

$$V(S) = \max a [R(s,a) + \gamma V(s')] \quad V(s) = \max a (R(s,a) + \gamma V(s'))$$

What has to be altered in the above mathematical Bellman's equation is that some type of randomness parameters could be introduced? As far as we don't know when the robot will not take the planned curve, we also don't know where it will end up, but that is nothing, but the location how it moves from the current room to the destination room area. Considering the juncture, w.r.t. the previous Bellman's equation model, no one is sure of the parameter s' , which is possibly the next wait state (rooms, as we are assigning the state). We do, however, know all of the possible paths the robot could take! For incorporating every probability into the mathematical model as stated above, one must assign a probability to each of the turns in order to quantify that the robot has an x percent chance of taking that particular route. If we do so, we get [8],

$$\begin{aligned} V(S) &= \max a \left([R(s,a) + \gamma \cdot \sum s'P(s,a,s')] V(s') \right) V(s) \\ &= \max a \left(R(s,a) + \gamma \cdot \sum s'P(s,a,s') V(s') \right) \end{aligned}$$

Now, step-by-step assigning the new notations to the above math model equations, we get [9]:

- $P(s,a,s')$ - with the action a , the new probability of movement of the robot from one room to another.
- $\sum s'P(s,a,s')V(s') \sum s'P(s,a,s')V(s')$ - Occurrence of the randomness visualizing the situation's expectation that the robot will face during the learning process.

Everything else from the previous two points is identical. Assume the following probability are linked with each of the robot's possible turns while in the room (red), to go towards another room (green). Then, we will have to assign different probability to every turnings (left, right, front, back, etc...), we would be saying that the robot will take the upper turn 80% of the time. When all of the required values are entered into our equation, we receive [10],

$$\begin{aligned} V(s) &= \max a \left(R(s,a) + \gamma \left((0.8V(\text{roomup})) + 0.1V(\text{roomdown}) + \dots \right) \right) V(s) \\ &= \max a \left(R(s,a) + \gamma \left((0.8V(\text{roomup})) + 0.1V(\text{roomdown}) + \dots \right) \right) V(s) + \dots \end{aligned}$$

Because we're introducing stochasticity, the value footprints will shift now. However, we will not calculate those value footprints this time. We'll leave it to the robot to figure it out (more on this in a moment). We haven't thought about awarding the robot for entering a specific room until now. We only pay the robot when it arrives at its destination. To help the robot better assess the quality of its activities, there should be a reward for each activity [11].

The incentives do not have to be consistent. However, having some sort of reward for the acts is preferable than having none at all. The living penalty is the term for this concept. In reality, the new system (rewarded) could be complex in nature & then the modelling sparse rewards are a hot topic in the field of reinforcement learning. If you want to give this issue a try, the following resources may be useful [12]:

- Procrastination & Curiosity in Reinforcement type of Learning.
- Self-supervised Predictions via Curiosity driven Explorations
- Gaining the ability to generalise from sparse and vague rewards.

Rather than focusing on the value of entering a specific room ($V(s)$), we will introduce the concept of action quality in the following section.

9. Transitioning to Q-Learning

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 fonts are preferred. We now have the continuity formula, which gives us a cost for travelling to a specific state (we'll refer to rooms as states from now on) while accounting for the stochasticity of the environment as depicted by [13].

$$V(s) = \max_a (R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s')) V(s) = (R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s'))$$

We also learnt a little about the concept of the living penalty, which entails linking each robot movement with a reward. Rather than establishing the potential worth of the state (value footprint) being moved to, Q-Learning advises evaluating the quality of the activity that was used to get there. Earlier, we had the environmental values with an agent with all possible values of the footprints displayed in the Table 13 as [14]. If we include the concept of evaluating the quality of activities for achieving a specific state s' , then the table with the environment belonging to the agents that could initiate a group of action which could be modified as shown in the Table 14 [15]. Now, the machine has got 4 new states from where it can choose the parametric values, as well as four possible actions to choose from for the current condition. So, here, we explain a method of calculating the value of $Q(s, a)$, viz., the overall quality of the robot's conceivable actions? Let's get started. From the equation as [16]

Table13: The area in which an agent with a possible value of the foot print

0.05 {V(s4)} ←	0.8 {V(s1)} ↑	→ 0.05 {V(s2)}
	O	
	↓	
	0.1 {V(s3)}	

Table14: An safe obstacle collision free zone with a number of agents having a quality of different actions

Q (s4 , a4)} ←	Q (s1 , a1)} ↑	→ Q (s2 , a2)}
	O	
	↓	
	Q (s3 , a3)}	

$$V(s) = \max_a (R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s')) V(s)$$

$$V(s) = \max_a (R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s'))$$

Note that when the max() f/n is discarded or disowned, we get the new math model as [17],

$$R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s') R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s')$$

Fundamentally, we include all possible actions and states (from the present state of the robot) in the equation that yields $V(s)$, and then we take the largest value induced by doing a

given action. The value footprint generated by the calculation above is for only one conceivable action. In reality, we can consider it to be the action's quality [18]:

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') V(s') \\ = R(s, a) + \gamma \sum_{s'} P(s, a, s') V(s')$$

We'll make a small tweak to the previous equation now that we have an expression to quantify the quality of a certain action. Now we may state that $V(s)$ is the highest of all feasible Q values (s, a). Let's use this to our advantage and replace $V(s')$ with a function of $Q()$ [19]:

$$Q(s, a) = R(s, a) + \gamma \left(\sum_{s'} P(s, a, s') \max_{a'} Q(s', a') \right) Q(s, a) \\ = R(s, a) + \gamma \sum_{s'} P(s, a, s') \cdot \max_{a'} Q(s', a')$$

To make our computations simpler, we adopt a new procedure as follows. Because we only have one function to calculate now, $Q()$ (which is also at the heart of the dynamic programming paradigm), and $R(s, a)$ is a quantified measure that generates rewards for going to a specific state. Q -values are the quality of the acts. The value footprints will now be referred to as Q -values [20].

Still now, we had discussed about various issues like the AI, ML, DL, RL, SL, USL, Intro to QL, etc... in this 1st part of the research paper. In the 2nd part of the research article, which is the content of another paper, we briefly discuss about the Q Learning Design, Design of a robot navigational path planner using Q-learning, Q-learning algorithm process, Temporal Difference (TD), Deep Q-learning, Deep Q-networks, Neural network and deep Q-learning, Softmax function, Object detection using Histogram of Oriented Gradients with implementation, Path Planners & Path Planning Problem (Gross Motion), Robot Motion Planning [21].

10. Conclusion

In this research paper, a brief insight into the theoretical & mathematical modelling concepts of the various types of algorithm development was presented in a concised manner to achieve the desired task, i.e., design of the robot path to move to the destination from the source to the goal in spite of obstacles viz., the 1st part & the 2nd part, the material presented in this paper is the 1st part of the concepts proposed & the extension of the materials in Part-I is presented as another research paper. Various theoretical concepts of AI & ML algorithms such as the Reinforcement, Q, Deep, Supervised, Unsupervised Learning, etc. are being proposed, which were discussed one after the other in a nutshell, first the theoretical concepts are being presented. The algorithm work is divided into a number of sections in the forth coming research articles that are going to be published where the theoretical concepts of AI & ML presented here are used to develop the hybrid algorithms for the path planning of the automatic machine (robot/s) from S to G (D).

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

The paper background work, conceptualization, methodology, dataset collection, implementation, result analysis and comparison, preparing and editing draft, visualization have been done by first author. The supervision, review of work and project administration, have been done by second author.

References

- [1] Zhihao Chen, Redouane Khemmar, Benoit Decoux, Amphani Atahouet, Jean-Yves Ertaud, “RealTime Object Detection, Tracking, and Distance and Motion Estimation based on Deep Learning: Application to Smart Mobility”, 2019 Eighth International Conference on Emerging Security Technologies (EST), Colchester, United Kingdom, Jul 2019.
- [2] Z. Zhao, P. Zheng, S. Xu and X. Wu, “Object Detection With Deep Learning: A Review”, IEEE Trans. on Neu. Net. & Learning Syst., vol. 30, no. 11, pp. 3212 to 3232, doi: 10.1109/TNNLS.2018.2876865, Nov. 2019.
- [3] L. Jiao et.al., “A Survey of Deep Learning-Based Object Detection”, IEEE Access Journal, vol. 7, pp. 128837 to 128868, 2019.
- [4] Yakup Demir, “Object recognition and detection with deep learning for autonomous driving applications”, Simulation: Transactions of the Society for Modeling and Simulation International, pp. 1–11.
- [5] MukeshTiwari, Dr. RakeshSinghai , “A Review of Detection and Tracking of Object from Image and Video Sequences”, Int. Jr. Compu. Intelli. Res., ISSN 0973-1873 Vo. 13, No. 5, pp. 745-765, 2017.
- [6] JahanzaibShabbir, and TariqueAnwer , “A Survey of Deep Learning Techniques for Mobile Robot Applications”, Jr. Latex Class Files, Vol. 14, No. 8, Aug. 2015.
- [7] Bae H., Kim G., Kim J., Qian D., Lee S., “Multi-Robot Path Planning Method Using Reinforcement Learning”, Journal of Applied Sciences, vol. 9, issue 15, pp. 30-57, 2019,
- [8] Nguyen, Khang, Huynh, Nhut T., Nguyen, Phat C., Nguyen, Khanh-Duy, Vo Nguyen D., Nguyen, Tam V., “Detecting Objects from Space: An Evaluation of Deep-Learning Modern Approaches”, Journal of Electronics, vol. 9, no. 4, pp. 583, 2020.
- [9] Alessandro Foa, “Object Detection in Object Tracking System for Mobile Robot Application”, Jour. of TRITA-SCI-GRU 2019:090 MAT-E, 2019:46.
- [10] https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e006
- [11] <https://www.educba.com/deep-learning-algorithms>
- [12] <https://pathmind.com/wiki/deep-reinforcement-learning>
- [13] <https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c>
- [14] <https://www.wikipedia.org>
- [15] Dr. T.C.Manjunath, “Fundamental of Robotics”, 5th Edition, Nandu Publishers, Mumbai, Inida, 2005.
- [16] Robert J.S., “Fundamentals of Robotics : Analysis and Control”, PHI, New Delhi., 1992.
- [17] Fu, Gonzalez and Lee, Robotics : Control, Sensing, Vision and Intelligence, McGraw Hill, Singapore, 1995.
- [18] Luh, C.S.G., M.W. Walker, and R.P.C. Paul, “On-line computational scheme for mechanical manipulators”, Jr. Dyn. Meast. Syst. & Contr., Vol. 102, pp. 69-76, 1998.
- [19] Yoshikawa T., “Analysis & Control of Robot Manipulators with Redundancy”, Proc. First Int. Symp. onRobotics Research, Cambridge, MIT Press, UK, pp. 735-748, 1984.
- [20] Whitney DE., The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators, Trans. ASM J. DynamicSystems, Measurements and Control, Vol. 122, pp. 303-309, 1972.
- [21] Whitney DE., Resolved Motion Rate Control of Manipulators and Human Prostheses, IEEE Trans. Syst. Man, Cybernetics, Vol. MMS-10, No. 2, pp. 47-53, 1969.
- [22] Lovass Nagy V, R.J. Schilling, Control of Kinetically Redundant Robots Using {1}-inverses, IEEE Trans. Syst. Man, Cybernetics, Vol. SMC-17, No. 4, pp. 644-649, 1987.

Article Received: 05 September 2021, Revised: 09 October 2021, Accepted: 22 November 2021, Publication: 26 December 2021

- [23] Lovass Nagy V., R J Miller and D L Powers, An Introduction to the Application of the Simplest Matrix-Generalized Inverse in Systems Science, IEEE Trans. Circuits and Systems, Vol. CAS-25, No. 9, pp. 776, 1978.
- [24] D. Xin, C. Hua-hua, and G. Wei-kang, "Neural network and genetic algorithm based global path planning in a static environment," Journal of Zhejiang University Science, vol. 6A, no. 6, pp. 549–554, 2005.
- [25] C. Yang, H. Jianda, and W. Huaiyu, "Quadratic programming-based approach for autonomous vehicle path planning in space," Chinese Journal of Mechanical Engineering, vol. 25, no. 4, pp. 665–673, 2012.
- [26] J.-H. Liang and C.-H. Lee, "Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm," Advances in Engineering Software, vol. 79, pp. 47–56, 2015.
- [27] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz, and N. Pavón, "Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach," Soft Computing- A Fusion of Foundations, Methodologies and Applications, vol. 21, no. 4, pp. 949–964, 2017.
- [28] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1398–1404, Sacramento, CA, USA, April 1991.
- [29] H. Seki, S. Shibayama, Y. Kamiya, and M. Hikizu, "Practical Obstacle Avoidance Using Potential Field for A Nonholonomic Mobile Robot with Rectangular Body," in Proceedings of the 13th IEEE International Conference on Emerging Technologies And Factory Automation, pp. 326–332, Hamburg, Germany, 2008.
- [30] M. Y. Ibrahim and L. McFetridge, "The Agoraphilic algorithm: A new optimistic approach for mobile robot navigation," Proc. of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics Proceedings, pp. 1334–1339, Como, Italy, July 2001.
- [31] J. Borenstein and Y. Koren, "The vector field histogram—fast obstacle avoidance for mobile robots," IEEE Transactions on Robotics and Automation, vol. 7, no. 3, pp. 278–288, 1991.
- [32] B. You, J. Qui, and D. Li, "A novel obstacle avoidance method for low-cost household mobile robot," Proceedings of the 2008 IEEE International Conference on Automation and Logistics (ICAL), pp. 111–116, Qingdao, China, September 2008.
- [33] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," IEEE Robotics and Automation Magazine, vol. 4, no. 1, pp. 23–33, 1997.
- [34] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," Proceedings of the 1999 IEEE International Conference on Robotics and Automation, ICRA99, pp. 341–346, May 1999.
- [35] J. Hong and K. Park, "A new mobile robot navigation using a turning point searching algorithm with the consideration of obstacle avoidance," The International Journal of Advanced Manufacturing Technology, vol. 52, no. 5–8, pp. 763–775, 2011.
- [36] Jadhav, Makrand M. et al. "Machine Learning based Autonomous Fire Combat Turret." Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12.2 (2021): 2372-2381.
- [37] Mulani, Altaf O., and P. B. Mane. "Watermarking and cryptography based image authentication on reconfigurable platform." Bulletin of Electrical Engineering and Informatics 6.2 (2017): 181-187.