ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research

REACHING THE TARGET WITH A SHORT DISTANCE AND TIME USING Q LEARNING ALGORITHM

Mohammed Nasif Mustafa

DOI: 10.5281/zenodo.55192
Keywords: Robots, Q Learning.

ABSTRACT
My project include a short abstract for robotic history and what's the robotic , what's the advantage and disadvantage of robotic , the type of robotic , the application of robotic , then I choose one method from robotic process methods which is Q–learning method for reaching the target with a short distance and time.

CHAPTER ONE *ROBOTS*

Introduction to Robots

. What is the first thing that comes to mind when you think of a robot?

For many people it is a machine that imitates a human—like the androids in Star Wars, Terminator and Star Trek the Next Generation. However much these robots capture our imagination, such robots still only inhabit Science Fiction. People still haven't been able to give a robot enough 'common sense' to reliably interact with a dynamic world. However, Rodney brooks and his team at MIT Artificial Intelligence Lab is working on creating such humanoid robots.

The type of robots that you will encounter most frequently are robots that do work that is too dangerous, boring, onerous, or just plain nasty. Most of the robots in the world are of this type. They can be found in auto, medical, manufacturing and space industries. In fact, there are over a million of these types of robots working for us today.

Some robots like the Mars Rover sojourner and the upcoming Mars Exploration Rover, or the underwater robot Caribou help us learn about places that are too dangerous for us to go. While other types of robots are just plain fun for kids of all ages. Popular toys such as Tekno , Polly or AIBO ERS-220 seem to hit the store shelves every year around Christmas time.

But what exactly is a robot?

As strange as it might seem, there really is no standard definition for a robot. However, there are some essential characteristics that a robot must have and this might help you to decide what is and what not a robot is. It will also help you to decide what features you will need to build into a machine before it can count as a robot.

A robot has these essential characteristics:

-Sensing: First of all your robot would have to be able to sense its surroundings. It would do this in ways that are not non similar to the way that you sense your surroundings. Giving your robot sensors: light sensors (eyes), touch and pressure sensors (hands), chemical sensors (nose), hearing and sonar sensors (ears), and taste sensors (tongue) will give your robot awareness of its environment.

-Movement: a robot needs to be able to move around its environment. Whether rolling on wheels, walking on legs or propelling by thrusters a robot needs to be able to move. To count as a robot either the whole robot moves, like the Sojourner or just parts of the robot moves, like the Canada Arm.

-Energy: a robot needs to be able to power itself. A robot might be solar powered, electrically powered, battery powered. The way your robot gets its energy will depend on what your robot needs to do.

-Intelligence: A robot needs some kind of "smarts." This is where programming enters the pictures. A programmer is the person who gives the robot its 'smarts.' The robot will have to have some way to receive the program so that it knows what it is to do.

ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research

So what is a robot?

Well it is a system that contains sensors, control systems, manipulators, power supplies and software all working together to perform a task. Designing, building, programming and testing robots is a combination of physics, mechanical engineering, electrical engineering, structural engineering, mathematics and computing. In some cases biology, medicine, chemistry might also be involved. A study of robotics means that students are actively engaged with all of these disciplines in a deeply problem-posing problem-solving environment.

The Advantages of Industrial Robots:

1. **Quality:** Robots have the capacity to dramatically improve product quality, Applications are performed with precision and high repeatability every time. This level of consistency can be hard to achieve any other way.
2. **Production: R**obots , throughput speeds increase , which directly impacts production. Because robots have the ability to work at a constant speed without pausing for breaks, sleep, vacations, they have the potential to produce more than a human worker.
3. **Safety:** Robots increase workplace safety. Workers are moved to supervisory roles, so they no longer have to perform dangerous applications in hazardous settings.
4. **Savings:** Greater worker safety leads to financial savings. There are fewer healthcare and insurance concerns for employers. Robots also offer untiring performance which saves valuable time .Their movements are always exact, so less material is wasted.

The Disadvantages of Industrial Robots:

1. **Expense:** The initial investment of robots is significant, especially when business owners are limiting their purchases to new robotic equipment the cost of automation should be calculated in light of a business' greater financial budget. Regular maintenance needs can have a financial toll as well.
2. **Role:** Integration industrial robots do not guarantee results. Without planning, companies can have difficulty achieving their goals.
3. **Expertise:** Employees will require training in programming and interacting with the new robotic equipment. This normally takes time and financial output.
4. **Safety:** Robots may protect workers from some hazards, but in the meantime, their very presence can create other safety problems. These new dangers must be taken into consideration.

CHAPTER TWO Q-LEARNING ALGORITHM

Abstract

Q-learning (Watkins, 1989) is a simple way for agents to learn how to act optimally in controlled Markovian domains. It amounts to an incremental method for dynamic programming which imposes limited computational demands. It works by successively improving its evaluations of the quality of particular actions at particular states.

Introduction

Q-learning (Watkins, 1989) is a form of model-free reinforcement learning. It can also be viewed as a method of asynchronous dynamic programming (DP). It provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build maps of the domains.

Learning proceeds similarly to Sutton's (1984; 1988) method of temporal differences (TD): an agent tries an action at a particular state, and evaluates its consequences in terms of the immediate reward or penalty it receives *and* its estimate of the value of the state to which it is taken. By trying all actions in all states repeatedly, it learns which best overall, judged by long-term discounted reward are. Q-learning is a primitive (Watkins, 1989) form of learning, but, as such, it can operate as the basis of far more sophisticated devices.

Examples of its use include Barto and Singh (1990), Sutton (1990), Chapman and Kaelbling (1991), Mahadevan and Connell (1991), and Lin (1992), who developed it independently. There are also various industrial applications.

ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research

Q Learning Algorithm:
Q-Learning uses an action-value function that calculates the expected utility of performing a speciation in a discrete state and following axed policy thereafter. Three die rent functions are involved: memorization, exploration and updating (figure 1).

In response to the present situation, an action is chosen. In order to ensure that both exploration and exploitation are performed, a set probability value is used alongside a randomly generated number to determine whether the agent will explore or exploit. If the random number is above the probability threshold, the optimal action yielding the highest q-value
Is selected (exploitation). Otherwise, a random action is selected (exploration). This promotes both aspects of exploitation and exploration which are necessary to ensure the success of the learning technique .

Each time the agent performs an action a in states at time t, the agent gets a reward r which represents how close it is to solving the task. The utility
for performing this action is then updated according to the update rule:

$$Q(s,\alpha) = Q_t(s,\alpha) + \alpha[r + \gamma . \max Q(\acute{s},\alpha') - Q(s, \alpha)] \ldots\ldots 1$$

Where s' is the next state, a' is the optimal action in s', α is the learning rate and γ is the discount factor. After applying this rule, all the other en- tries in the table remain unchanged. Since there is an expected reward for each action in a multitude of states, the total size of the table (state-space)
Grows exponentially as the complexity of the problem increases. This rapid growth results in dismal scalability in addition to physical space limitations in terms of data storage.

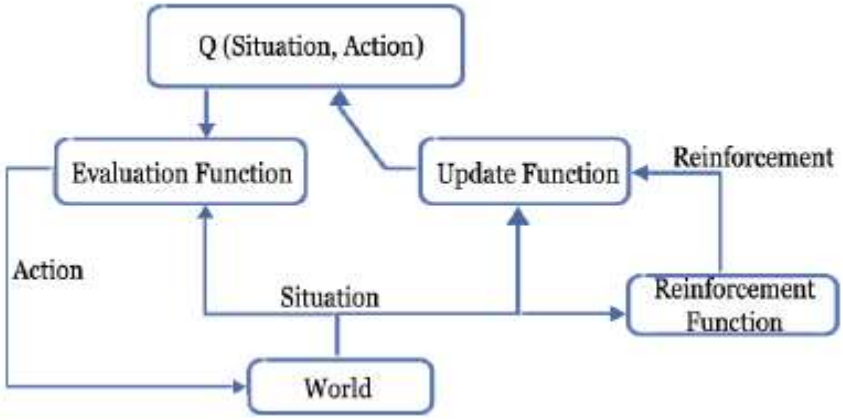


Figure 1: The 3 functions of q-learning

The task for Q-learning:
Consider a computational agent moving around some discrete, finite world, choosing one from a finite collection of actions at every time step. The world constitutes a controlled Markov process with the agent as a controller. At step *n*, the agent is equipped to register the state *xn* (Є X) of the world, and can choose its action *an* (Є 2)1 accordingly. The agent receives a probabilistic reward *rn*, whose mean value (*RXn (an)* depends only on the state and action, and the state of the world changes probabilistically to *yn* according to the law:

$$\text{Prob}\{y_n = y | x_n, a_n\} = P_{x_n y}[a_n].$$

The task facing the agent is that of determining an optimal policy, one that maximizes total discounted expected reward. By discounted reward, we mean that rewards received *s* steps hence are worth less than rewards received now, by a factor of *ys* (0 < *y* < 1). Under a policy T, the value of state *x* is:

$$V^{\pi}(x) \equiv \mathcal{R}_x(\pi(x)) + \gamma \sum_{y} P_{xy}[\pi(x)] V^{\pi}(y$$

ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research

because the agent expects to receive *(Rx(r(x))* immediately for performing the action *I* recommends, and then moves to a state that is 'worth' *V*(y)* to it, with probability Pxy [>(*)]. The theory of DP (Bellman & Dreyfus, 1962; Ross, 1983) assures us that there is at least one optimal stationary policy T* which is such that

$$V^*(x) \equiv V^{\pi^*}(x) = \max_a \left\{ \mathcal{R}_x(a) + \gamma \sum_y P_{xy}[a] V^{\pi^*}(y) \right\}$$

is as well as an agent can do from state *x*. Although this might look circular, it is actually well defined, and DP provides a number of methods for calculating *V** and one T*(assuming that *(Rx(a)* and Pxy[a] are known. The task facing a Q, learner is that of determining a *f** without initially knowing these values.

There are traditional methods (e.g., Sato, Abe & Takeda, 1988) for learning *(Rx(a)* and Pxy[a] while concurrently performing DP, but any assumption of certainty equivalence, i.e., calculating actions as if the current model were accurate, costs dearly in the early stages of learning (Barto & Singh, 1990). Watkins (1989) classes ^-learning as incremental dynamic programming, because of the step-by step manner in which it determines the optimal policy. For a policy T, define Q, values (or action-values) as:

$$\mathcal{Q}^\pi(x, a) = \mathcal{R}_x(a) + \gamma \sum_y P_{xy}[\pi(x)] V^\pi(y).$$

In other words, the 3, value is the expected discounted reward for executing action *a* at state *x* and following policy T thereafter. The object in Q-learning is to estimate the Q, values for an optimal policy. For convenience, define these as *Q*(x, a) = Q**(x, a), Vx, a.*
It is straightforward to show that *V*(x)* = maxa *Q*(x, a)* and that if a* is an action at which the maximum is attained, then an optimal policy can be formed as *r*(x) = a**. Herein lies the utility of the Q, values—if an agent can learn them, it can easily decide what it is optimal to do.
Although there may be more than one optimal policy or *a**, the 5* values are unique.
In Q-learning, the agent's experience consists of a sequence of distinct stages or *episodes.*
In the nth episode, the agent:
• observes its current state *xn,*
• selects and performs an action *an,*
• observes the subsequent state *yn,*
• receives an immediate payoff *rn,* and
• adjusts its Qn-1 values using a learning factor an, according to:

$$\mathcal{Q}_n(x, a) = \begin{cases} (1 - \alpha_n)\mathcal{Q}_{n-1}(x, a) + \alpha_n[r_n + \gamma V_{n-1}(y_n)] & \text{if } x = x_n \text{ and } a = a_n, \\ \mathcal{Q}_{n-1}(x, a) & \text{otherwise,} \end{cases} \quad (1)$$

Where:

$$V_{n-1}(y) \equiv \max_b \{\mathcal{Q}_{n-1}(y, b)\} \quad (2)$$

is the best the agent thinks it can do from state *y.* Of course, in the early stages of learning, the *Q,* values may not accurately reflect the policy they implicitly define (the maximizing actions in equation 2). The initial Q, values, *Q,0(X, a),* for all states and actions are assumed given.
Note that this description assumes a look-up table representation for the *Q,n(x, a).*
Watkins (1989) shows that ^-learning may not converge correctly for other representations.
The most important condition implicit in the convergence theorem given below is that the sequence of episodes that forms the basis of learning must include an infinite number of episodes for each starting state and action. This may be considered a strong condition on the way states and actions are selected—however, under the stochastic conditions of the theorem, no method could be guaranteed to find an optimal policy under weaker conditions.

ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research

Note, however, that the episodes need not form a continuous sequence—that is the *y of* one episode need not be the *x* of the next episode.
The following theorem defines a set of conditions under which *Qn(x, a)* - *Q*(x, a)* as *n* - o. Define *ni(x, a)* as the index of the ith time that action *a* is tried in state *x*

Theorem
Given bounded rewards \rn\ < (R, learning rates 0 < an < 1, and

$$\sum_{i=1}^{\infty} \alpha_n{}^{i(x,a)} = \infty, \quad \sum_{i=1}^{\infty} [\alpha_n{}^{i(x,a)}]^2 < \infty, \quad \forall x, a, \qquad (3)$$

then *Qn(x, a)* - *G*(x, a)* as *n* - o, Vx, a, with probability 1.

CHAPTER THREE

-Working methods of the Robot:

A: Candidate elimination.

The **candidate elimination algorithm** incrementally builds the version space given a hypothesis space ***H*** and a set *E* of examples. The examples are added one by one; each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example. The candidate elimination algorithm does this by updating the general and specific boundary for each new example. This is described below.

1: **Procedure** CandidateEliminationLearner(*X*,*Y*,*E*,***H***)
2: **Inputs**
3: *X*: set of input features, $X=\{X_1,...,X_n\}$
4: *Y*: target feature
5: *E*: set of examples from which to learn
6: ***H***: hypothesis space
7: **Output**
8: general boundary $G \subseteq \boldsymbol{H}$
9: specific boundary $S \subseteq \boldsymbol{H}$ consistent with *E*
10: **Local**
11: *G*: set of hypotheses in ***H***
12: *S*: set of hypotheses in ***H***
13: Let *G={true}*, *S={false}*;
14: **for each** $e \in E$ **do**
15: **if** (*e* is a positive example) **then**
16: Elements of *G* that classify *e* as negative are removed from *G*;
17: Each element *s* of *S* that classifies *e* as negative is removed and replaced by the minimal generalizations of *s* that classify *e* as positive and are less general than some member of *G*;
18: Non-maximal hypotheses are removed from *S*;
19: **else**
20: Elements of *S* that classify *e* as positive are removed from *S*;
21: Each element *g* of *G* that classifies *e* as positive is removed and replaced by the minimal specializations of *g* that classifies *e* as negative and are more general than some member of *S*.
22: Non-minimal hypotheses are removed from *G*.
23:
24:

B: Tree search:

In computer science, **tree traversal** (also known as **tree search**) is a form of graph traversal and refers to the process of visiting (checking and/or updating) each node in a tree data structure, exactly once. Such traversals are classified by the order in which the nodes are visited. The following algorithms are described for a binary tree, but they may be generalized to other trees as well.

ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research

Types of tree search:
1. *Depth-first search:*
A. Pre-order.
B. In-order.
C. Post-order.
D. Generic tree.

2. *Breadth-first search*
3. *Other types*

C: Find-s algorithm:
This algorithm tries to find the most specific target concept. To do so, it assumes that the target concept exists in the hypothesis class and that the most specific representation will yield the best results (it is the best Representation).
How does it work? We begin by initializing our hypothesis *h* to the most specific hypothesis in our hypothesis class *H* [1]. We then proceed through our training examples. If we find a negative example, we discard it, making no changes to *h*. If, however, we have a positive example, we then alter *h* so that it accepts this new example (as well as all previous training examples) but remains as specific as possible. We then continue this process until we have seen all of the training data.
There are some shortcomings to this method, however. Although it will find a hypothesis consistent with the data, there is no way to determine that this hypothesis is the only target concept consistent with the data. Further, there is no way to determine how many consistent hypotheses exist within the hypothesis class. In addition; the Find-S Algorithm is very sensitive to incorrect data (or noise) that may be in the training examples. Further, the only contributors to the hypothesis are the positive examples in the training data.

D: A* Searching algorithm
A* (pronounced 'A-star') is a search algorithm that finds the shortest path between some nodes S and T in a graph.

E: Q- Learning algorithm.
And I will choose the last method (Q learning) that we explain it in chapter two.
Work procedures:
1- Name space Q learning
2-the program include four matrix [16*16]
3- The matrix use from the program is the following:-

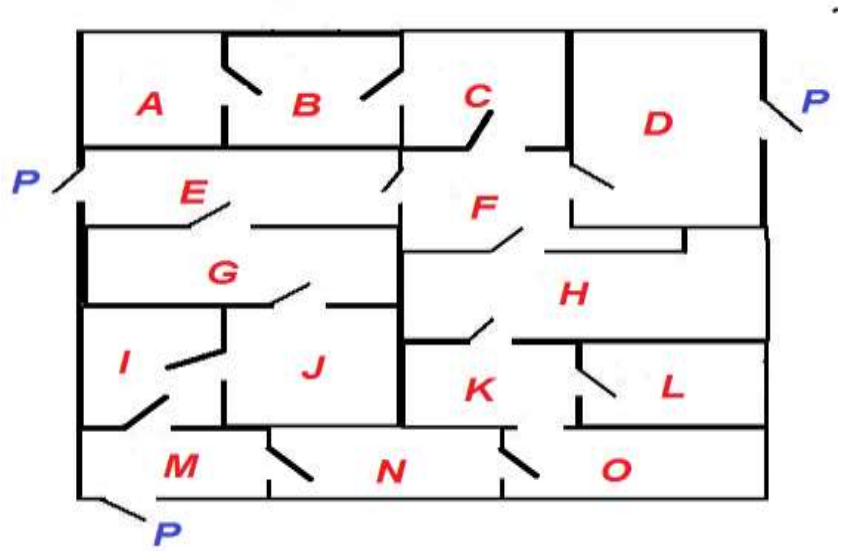
* R matrix [16*16]
*Q matrix [16*16]
* Temp Q matrix [16*16]

4- R matrix: - is a basic matrix using for serving data for calculation.
5- Q matrix: - is a matrix using for calculate operation.
6- Temp Q matrix: saving the calculate operation that is execute.
7- R matrix: - saving the data like:-
If there is a direct path from first area to the second area we put (0) symbol, and if there is no direct path we put (-) symbol, and we put (100) symbol when it reach the target (goal) directly.

Distributed method-
0 → direct path
- → No direct path
100 → is the goal
Matrix distributed method contain of [16] column and [16] Row.

ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research



1- From A to A no path (-)
. From A to B direct path (0)
. From A to C no direct path (-)
. From A to D no direct path (-)
. From A to F no direct path (-)
From A to E no path (-)
. From A to G direct path (0)
. From A to H no direct path
. From A to I no direct path (-)
. From A to J no direct path (-)
.From A to K no path (-)
. From A to L direct path (0)
. From A to M no direct path
. From A to N no direct path (-)
. From A to O no direct path (-)
.From A to P no path (-)
. From A to Q direct path (0)
. From A to R no direct path
. From A to S no direct path (-)
. From A to T no direct path (-)
. From A to U no direct path (-)
. From A to V no direct path (-)
. From A to W no direct path (-)
. From A to X no direct path (-)
. From A to Y no direct path (-)
. From A to Z no direct path (-)

2- from B to A direct path (0)
. From B to B no direct path (-)
. From B to D no direct path
. From B to C direct path (0)
. From A to D no direct path (-)
. From A to F no direct path (-)
From A to E no path (-)
. From A to G direct path (0)
. From A to H no direct path

ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research

. From A to I no direct path (-)
. From A to J no direct path (-)
. From A to K no path (-)
. From A to L direct path (0)
. From A to M no direct path
. From A to N no direct path (-)
. From A to O no direct path (-)
. From A to P no path (-)
. From A to Q direct path (0)
. From A to R no direct path
. From A to S no direct path (-)
. From A to T no direct path (-)
. From A to U no direct path (-)
. From A to V no direct path (-)
. From A to W no direct path (-)
. From A to X no direct path (-)
. From A to Y no direct path (-)
. From A to Z no direct path (-)

3- From D to A no direct path (-)
. From D to B no direct path (-)
. From D to C no direct path (-)
.From D to F direct path (0)
. From D to P goal path (100)
. etc.

4 – From M to A no direct path (-)
. From A to I direct path (0)
. From A to P goal path (100)
. From E to G to J to I to goal (100)
. From E to F to D to goal (100)
. From F to E to G to J to I to goal (100)
. From F to D to goal (100)
. From F to E to goal (100)
. From G to J to I to goal (100)
. From G to E to F to D to goal (100)
. From G to E to goal (100)

If we assume the Robot is stop IN a place like [F] THE ROBOT IS going TO calculate the way that - lead to the target and it choose a short way in according to the available calculation from the program
1- F → D →P = 800
2- F →H →k → o→ N →M → P = 1500
3- F →E → P = 820
4- F →E →G → J → T →M →P =1600
SO it choose the short path that represents (F →D→P) OF the process above.

REFERENCES

1. **The Internet Website:**

http://prime.jsc.nasa.gov/ROV/types.html
http://www.ai.mit.edu/people/brooks/index.shtml
http://www.robotbooks.com/robot-news.htm
http://www.aibo.com
http://www.tekno-robot.com/index1.htm1

ISSN 2349-6193
Impact Factor: 2.556

IJESMR
International Journal OF Engineering Sciences & Management Research

http://prime.jsc.nasa.gov/rov/images
http://www.thetech.org
http://www.militaryandiology.org
http**://www.robots.com/arrivals.php**
http://www.doc.ic.ac.uk/~ajd/Robotics/ for up to date information.
http://mechatronics.poly.edu
http://artint.info/html/ArtInt_193.html
https://en.wikipedia.org/wiki/Tree_traversal.

2. Dr. Bob Williams, "**An Introduction to Robotics**" © 2016 Dr. Bob Productions.
3. CH J. C .H. WATKINS, and P. Dayan, " **Machine Learning** 8, 279-292 (1992), Kluwer Academic Publishers, Boston. Manufactured in The Netherlands. Centre for Cognitive Science, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9EH, Scotland.
4. I. Karabegović, E. Karabegović, and E. Husak**.** " **INDUSTRIAL ROBOTS AND THEIR APPLICATION IN SERVING CNC MACHINE TOOLS**" TMT 2011, Prague, Czech Republic, 12-18 September 2011.
5. COELLO, C.A., PULIDO, G.T. and LECHUNGA, M.A. (2004), **Handling multiple objectives with particle swarm optimization**, 8(3), PP. 256-279, In proceedings of the 2002 Congress on Evolutionary Computation, In IEEE Transactions on Evolutionary Computation,8(3).
6. 15th International Research/Expert Conference **"Trends in the Development of Machinery and Associated Technology"** TMT 2011, Prague, Czech Republic, 12-18 September 2011.
7. Isak Karabegović, Edina Karabegović, Ermin Husak, **"INDUSTRIAL ROBOTS AND THEIR APPLICATION IN SERVING"** CNC MACHINE TOOLS **,** Machine Learning, 8, 279-292 (1992).
8. CHRISTOPHER J.C.H. WATKINS **"Q,-Learning"** *25b Framfield Road, Highbury, London N5 IUU, England*
9. PETER DAYAN, *Centre for Cognitive Science, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9EH, Scotland*
10. Steve Dini and Mark Serrano "**Combining Q-Learning with Artificial Neural Networks in an Adaptive Light Seeking Robot**" May 6, 2012**.**