Motion Planning in Unknown Environments

Vikranth Dwaracherla [*], Varsha Sankar [†], Radhika Pramod Patil [‡]

Abstract

Motion planning plays a very crucial role in military, disaster management and rescue . But often in such situations, we do not have a very reliable map of the surroundings. In this project, we propose a method to navigate the robot in unknown environments using local information obtained from surroundings. Method proposed uses Neural Networks and Q-learning. Q- learning is a reinforcement learning algorithm. Description of the proposed method is as follows. Introduction, related work is explained in sec 1 , sec 2 followed by Problem formulation which is explained in 3 and the method used for data generation for training neural network is described in sec 4, Implementation and Simulation results are presented in sec 5, 6 and conclusions and future work are stated in sec 7.

1. Introduction

Autonomous Navigation in unknown environments is one of the most widely researched areas as it finds application in a lot of fields, such as military, search, rescue and exploration. Since navigating in unknown environment means that the robot does not have the complete or reliable map of the area, it should be inherently intelligent to be able to make optimal decisions at each time step and move towards its goal with limited local information from the sensors about the immediate surroundings, thus making it a challenging problem. This was our motivation for choosing the problem. The problem fits well into the realm of Reinforcement learning. We have devised a method for the robot to move to any desired destination using a Q-learning approach. The inputs (state) are the local sensor data which give information about the obstacles in the immediate surroundings and the location of the robot, destination. Based on this, an optimal decision (action), i.e, velocity and angular velocity are chosen such that they maximize the reward (Q-value) for that (state, action) pair. So, in this case Q-value could be seen as a metric that evaluates how good a given (state, action) pair is. The Q-value is obtained from function approximation by using a Neural network, whose input consists of the (state, action) pair and outputs an estimate of Q-value.

2. Related Work

We have looked at some previous work on autonomous navigation algorithms which were based on machine learning approaches. [1] deals with training a guidance function to give the robot greater visibility into unknown parts of the environment based on the value of the future observations in terms of the expected cost to goal, instead of exploration techniques to observe the map and this results in high speed navigation. [2] uses reinforcement learning approach using neural networks to store Q-value. In [3], an online algorithm is presented to guarantee the safety of the robot through an emergency maneuver. [4] presents high-speed navigation in unknown environments by modeling the problem as a POMDP and at the same time discusses difficulty of this approach as there is no accurate probability distribution of the real-world environment. There has been work on extending these algorithms to 3-dimensional space (for air vehicles), and similar work is presented in [5]. The proposed method is

[*] vikranth@stanford.edu
[†] svarsha@stanford.edu
[‡] radhikap@stanford.edu

motivated mainly by the ideas from [1] and [2].

3. Problem Formulation

Our goal is to navigate the robot to a desired destination (G) using the local information obtained from its surroundings. Here we consider that the robot has range based sensors (like Ultrasonic sensors/ RADAR/ LIDAR/ SLAM, etc), which gives the minimum distance to an obstacle along a particular direction w.r.t the heading (θ) of the robot. Here measurements are minimum of distance to the obstacle and maximum possible range of the sensor. This is to mimic the realistic measurements obtained from range based sensors. Robot also knows the desired location so, it computes desired heading angle (α) which tells us the direction along which robot needs to move in a free space i.e. if there were no obstacles. Now, the robot needs to decide on the action i.e speed(v) and angular velocity(w) based on this information. The robot's motion is non-holonomic robot and governed by the equations below:

$$\begin{aligned} \dot{x} &= v\sin\theta \\ \dot{y} &= v\cos\theta \\ \dot{\theta} &= \omega \end{aligned}$$

Here, v,w is the speed and angular velocity along counter clockwise direction. x,y represent x and y co-ordinate of the current position of the robot. S denotes the array of the sensor values where, each element is minimum distance to an obstacle along the corresponding direction. Sensor placement considered is shown in the fig 1

Based on the local information of the robot, we would like to predict an optimal action which takes the robot to the desired destination in least amount of time. We developed a reinforcement based approach in order to solve this problem. Neural network is used to predict the Q-value of a state and an action pair $Q(s,a)$. This $Q(s,a)$ value of state s and taking an action a, tells us how good our action is when the robot is at state s. The optimal action at time t, is given by $\hat{a_t} = \arg\max_a Q(s_t,a_t)$. But, we need to come up with such a Q-value to generate data points, as neural network needs supervised data to train it. One of the metric which we used was reward $R(s_t,a_t)$ Eq 1, which tells us how good our action a_t is when state of the robot is s_t.

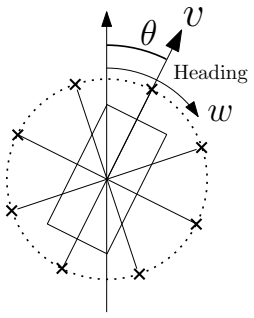

Figure 1: Configuration of the robot showing the sensor placement, Heading angle θ

4. Data for training Neural Network

Input to the neural network is s,a. Where s consists of sensor measurements S, current heading θ and desired heading α. a represents a possible action (v,w). Therefore input to neural network is S,θ,α,v,w (12 dimensional vector in our case). Output is reward estimate $R(s,a) = Q(s,a)$. But, in general, we can evaluate the action only after it reaches the desired goal or fails to reach after some iterations.

The proposed method gives a metric based on local information which tells us how good an action a is at state s. It is given by Eq 1.

$$\begin{aligned} Reward(s,a) = &-\rho\left(\frac{d+D_n-D_c}{D_c}\right) - \gamma\left(dt - \frac{d}{V_{max}}\right) \\ &- \beta|\omega| - \delta(1-S(0)) \end{aligned} \tag{1}$$

Here d is the distance traveled by the robot in that particular time step by taking action a, D_c, D_n are shortest distances to the goal before and after taking action a. $S(0)$ is the distance of the obstacle directly in front of the robot after taking action a. dt is the time step or discretization factor. $Vmax$ is the maximum speed of the robot. ρ,γ,β and δ are hyper parameters which gives importance to how

well various actions were performed. The idea of constructing a metric to inform us regarding performance of an action was taken from [1].

In [1], used only first term in Eq 1 was used for performance evaluation. But this poses a problem as $v = 0$ is the maximizer for the reward which attains a maximum value of 0 at $v = 0$. In [1], they have overcome this by mentioning the paths explicitly. ρ is the weight or importance assigned for choosing an optimal path in the map on which the robot was trained on based on current state. γ gives importance for choosing the max velocity, as we would like to choose max attainable velocity whenever possible. β is used to penalize the paths which have unnecessary turns. It also acts as a parameter which control the rate at which heading angle changes. δ is used to penalize the paths in which bot goes for head on collision with the obstacles. We have observed that on addition of δ term the improvement was minimal. So, we have chosen a lower value of $\delta \approx 0$.

This formulation allowed us to explicitly give a metric for an action at any state. This enables us to train a neural network by posing the problem of learning the Q value as a supervised learning problem. In the data generation phase, we randomly pick a point from the feasible locations in the map and calculate the reward values for actions picked from the set of feasible actions randomly. These are our $Data_{in}$ and $Data_{out}$ for training neural network. This is done at the starting to generate a lot of data. This is similar to a *warm-start* approach. Once the neural network is trained, we use the neural network to make an optimal decision to obtain the desired path. The value of metric is also generated so that we can use these to train neural network after robot reaches destination or fails to reach within a maximum number of iterations. Neural network structure is shown in fig 2

5. Method of Implementation

Implementation of the system requires the equations to be discretized. The time step for discretization is dt . The kinematic equations governing the motion of the robot are (2)and (3)

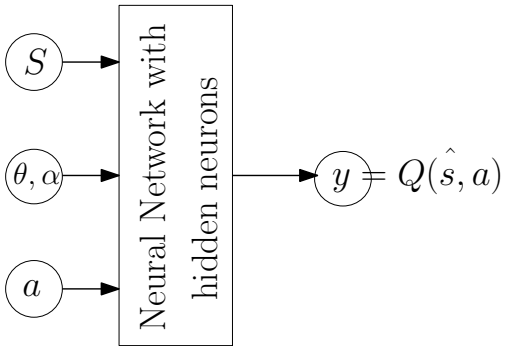

Figure 2: Neural Network

$$dx = \mathbb{1}_{\{\omega \neq 0\}} \frac{v}{\omega}(\cos(\theta + \omega dt) - \cos\theta) + \mathbb{1}_{\{\omega=0\}} v \sin\theta dt \quad (2)$$

$$dy = \mathbb{1}_{\{\omega \neq 0\}} \frac{v}{\omega}(\sin\theta - \sin(\theta + \omega dt)) + \mathbb{1}_{\{\omega=0\}} v \cos\theta dt \quad (3)$$

Since it is not practical to check Q value for all possible v, ω values we restrict them to $v \in [0, V_{max}]$, $\omega \in (-\pi, \pi]$(which are still infinitely many points), we discretize the values of v, ω into a finite number of points. Since, neural network prediction doesn't depend on the value of discretization, we can discretize into any number of values as per our convenience. This is one of the advantage of using the neural networks as they have capability to interpolate. 8 sensor range based measurements were considered at a step of $\pi/4$ in interval $(-\pi, \pi]$.

At the t^{th} step, we are at s_t. A_t represents the set of all feasible actions at the state s_t. We calculate $Q(s_t, a), \forall a \in A_t$ using neural network and pick the action by $a_t = \arg\max_{a \in A_t} Q(s_t, a)$.

By using the kinematic model described in Eq 2, Eq 3 we compute the location of the robot at $t + 1^{th}$ step and repeat the process until the desired goal or maximum number of steps is reached. The complete algorithm structure is shown in fig 3

6. Simulation Results

Simulations are carried out on MATLAB. For the simulations $dt = 0.1$, $V_{max} = 25$, $\omega_{max} = \pi/2$

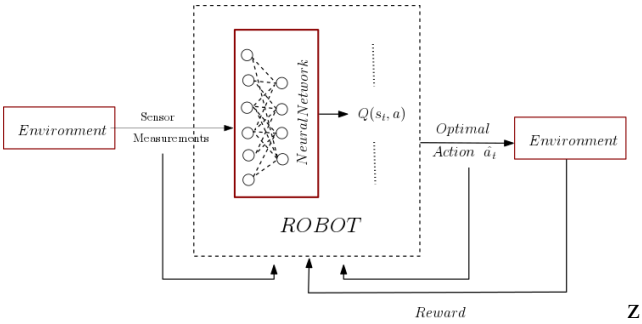

Figure 3: Flowchart for the algorithm

z

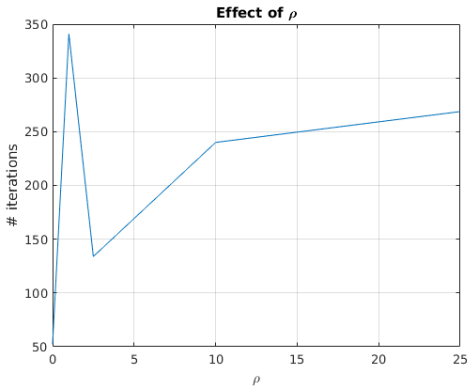

Figure 4: Effect of ρ on performance

were used.

We have analyzed the affect of hyper-parameter on the performance. Here, performance is decided based on the number of iterations taken to reach goal. Higher the number of iterations, more is the time required and this implies lower performance. Lower the number of iterations, less time and this means better performance.

We have analyzed the performance of ρ while keeping the other hyper-parameters chosen to be constant as $\beta = 0.1$, $\gamma = 2.5$. The performance can be observed in fig 4. We can observe that by choosing a higher value of ρ or lower value of ρ the performance deteriorates. The optimal value of ρ was chosen as $\rho \approx 2$.

We have then analyzed the performance of γ while the other hyper-parameters chosen to be $\beta = 0.1$, $\rho = 2$. The performance can be observed in fig 5. We can observe that by choosing a higher value of γ or lower value of γ the performance deteriorates. The optimal value of γ was chosen as $\gamma \approx 2.5$.

Performance of β while keeping $\gamma = 2.5$, $\rho = 2$ constant was analyzed. The performance can be observed in fig 6. We can observe that by choosing a higher value of β or lower value of β the performance deteriorates. The optimal value of β was chosen as $\beta \approx 0.1$.

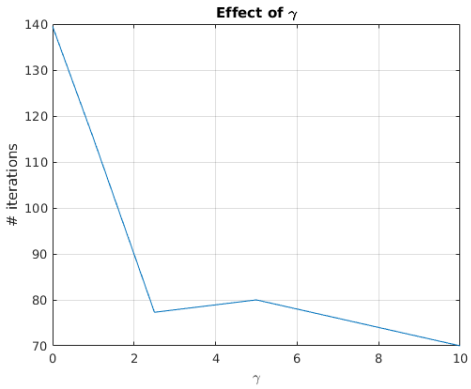

Figure 5: Effect of γ on performance

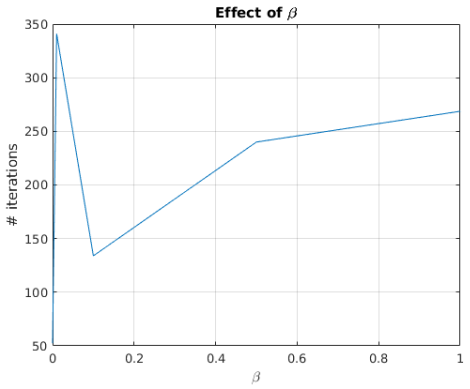

Figure 6: Effect of β on performance

We initially tried a neural network with multiple hidden layers but observed that the performance is similar to a single hidden layer one. Hence, we have chosen a single layer neural network. Since, the number of inputs were 12,

we analyzed the effect of varying the number of neurons from 1 to 12. We have observed that performance is similar after number of neurons were greater than 6 as shown in Fig 7. This suggests to us that the hidden latent variables are around 6. So we have chosen a size of 7 neurons for hidden layer.

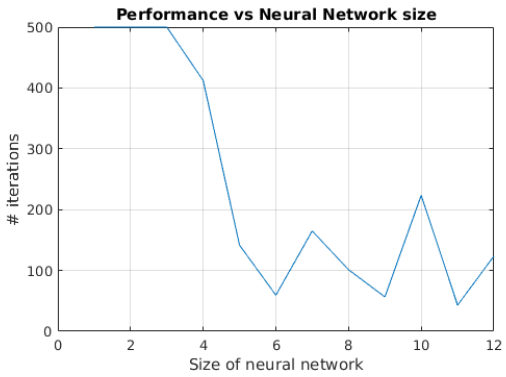

Figure 7: Effect of size of hidden layer on performance

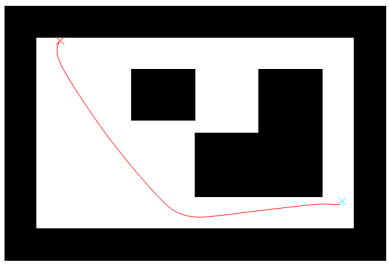

Figure 8: Path is obtained by giving a higher value of δ and β in the Eq 1

Simulations are carried out with default values of $\rho = 2$, $\beta = 0.1$, $\gamma = 2.5$.

In some cases, when the map trained on and map on which the algorithm is tested are entirely different then the robot fails to reach its destination and gets trapped at some points. The main reason for this is that we are using only local information. In order to over come this, we have observed that on adding a exploratory behavior by choosing a randomized action after few iterations enables the robot to reach the desired destination.

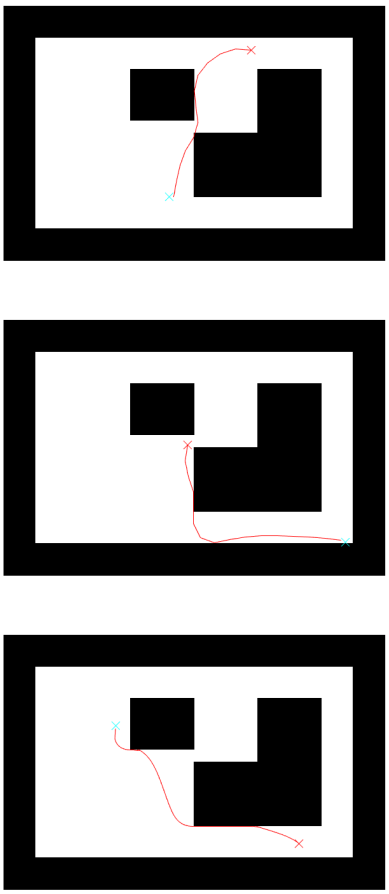

Figure 9: Results obtained on using the proposed algorithm

7. Conclusion and Future work

We have proposed a method to navigate a robot in an unseen environment using neural networks and $Q-learning$ based approach, using only the local information and desired destination. In order to speed up the process, a warm-start procedure is used. Since, prediction of a neural network is just a matrix multiplication, it is incredibly fast and can be employed for real-time applications.

Future work, would involve practical implementation of the method on a robot. We also need to look at the characteristic of the various sensors for range measurements, localization and adjust our metric appropriately.

References

[1] Richter, Charles and Roy, Nicholas *Learning to Plan for Visibility in Navigation of Unknown Environments*, Birkhäuser, 2013.

[2] Huang, Bing-Qiang, Guang-Yi Cao, and Min Guo. *"Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance." 2005 International Conference on Machine Learning and Cybernetics.* Vol. 1. IEEE, 2005.

[3] Arora, Sankalp, et al. *"Emergency maneuver library-ensuring safe navigation in partially known environments." 2015 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2015.

[4] Richter, Charles, William Vega-Brown, and Nicholas Roy. *"Bayesian Learning for Safe High-Speed Navigation in Unknown Environments."* Proceedings of the International Symposium on Robotics Research (ISRR 2015), Sestri Levante, Italy. 2015.

[5] Bachrach, Abraham, et al. *"RANGERobust autonomous navigation in GPSdenied environments."* Journal of Field Robotics 28.5 (2011): 644-666.