REINFORCEMENT LEARNING APPLIED TO MOBILE ROBOT NAVIGATION

Navyata Sanghvi

May 14, 2015

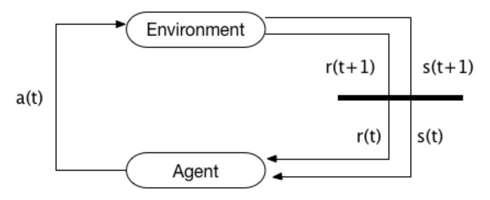
Project Guides:

Dr. P.V. Manivannan (ME, IIT Madras)

Dr. Balaraman Ravindran (CSE, IIT Mardas)

1. Preliminaries
    - Markov Decision Process (MDP)
    - Value Function
    - Q-Learning Algorithm
2. Robot Navigation Tasks
    - Any Goal; No Obstacles
    - Fixed Goal; Fixed Obstacle Configuration
    - Any Goal; Any Obstacle Configuration
3. Conclusion and Future Work

PRELIMINARIES



<u>Definition 1:</u> Markov Decision Process (MDP) $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$

- $\mathcal{S}$: state space
- $\mathcal{A}_s$: action space
- $\mathcal{P}(s'|s, a)$: probability that an action a taken in state s will lead to state s' at the next time step
- $R_a(s, s')$: immediate reward received after transition from state s to s' upon taking action a
- γ: discount factor $\in (0, 1)$

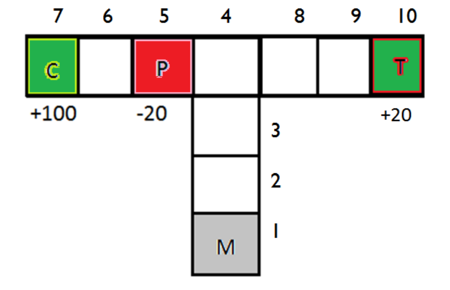
Definition 2: A policy π for an MDP is a mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$ from states to actions; $\pi(s)$ denotes the action choice in state s.

Definition 3: The cumulative discounted sum of rewards R_∞ is as follows:

$$R_\infty \stackrel{\text{def}}{=} \sum_{t=0}^{\infty} \gamma^t \, R_{a_t}(s_t, s_{t+1}),$$
$$\text{where } a_t = \pi(s_t)$$

Definition 4: The optimal policy π^* is as follows:

$$\pi^* \stackrel{\text{def}}{=} \arg\max_{\pi} \mathbf{E}\left[R_\infty\right]$$



1. $\gamma = 0.1$
   - On turning left, $R_\infty = -20 + 0.1(0) + 0.1^2(100) = -19$.
   - On turning right, $R_\infty = 0.1^2(20) = 0.2$.
2. $\gamma = 0.9$
   - On turning left, $R_\infty = -20 + 0.9(0) + 0.9^2(100) = 61$.
   - On turning right, $R_\infty = 0.9^2(20) = 16.2$.

Definition 5: A state-action value function is a mapping $Q : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$.

$$Q^{\pi}(s,a) \stackrel{\text{def}}{=} \sum_{s'} P(s'|s,a)\Big(R_a(s,s') + \gamma Q^{\pi}(s',\pi(s'))\Big)$$

Q-Learning

Bellman Operator B

$$BQ(s,a) \stackrel{\text{def}}{=} \sum_{s'} P(s'|s,a)\Big(R_a(s,s') + \gamma \max_{a'} Q(s',a')\Big)$$

Under optimal Policy π^*

$$Q^{\pi^*}(s',\pi^*(s')) = \max_{a'} Q(s',a')$$

$$\Leftrightarrow BQ^* = Q^*$$

Q-LEARNING ALGORITHM

Updates

$$Q(s,a) \leftarrow \sum_{s'} P(s'|s,a)\Big(R_a(s,s') + \gamma \max_{a'} Q(s',a')\Big)$$

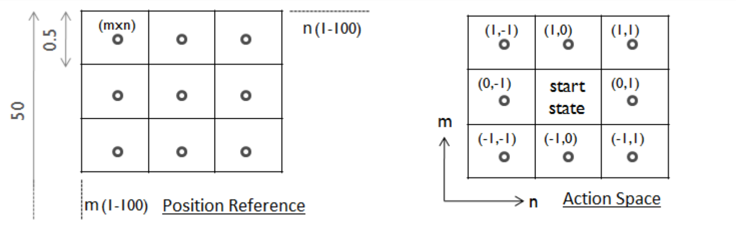
$$\pi(s) \leftarrow \arg\max_{a} Q(s,a)$$

Algorithm 1: Q Learning

Input: Bellman operator B for the MDP, and allowable error ϵ
Initialize $Q(s,a) \leftarrow 0$
Initialize $converged = false$
repeat
  $Q_{new} = BQ_{old}$
  if $\max(|Q_{new} - Q_{old}|) < \epsilon$ **then**
    $converged = true$
  end if
  $Q_{new} = Q_{old}$
until $converged = true$
return Q_{new}

ROBOT NAVIGATION TASKS

State and Action Spaces

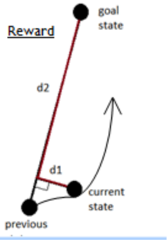


$$a = (m_{change}, n_{change})$$

$$\mathcal{A}_s = \{(1,1)\ \ (1,0)\ \ (1,-1)\ \ (0,1)\ \ (0,-1)\ \ (-1,1)\ \ (-1,0)\ \ (-1,-1)\}$$

MDP SETUP

Reward Function



$$d1 = \frac{|(m_g - m_p)(n_c - n_p) - (n_g - n_p)(m_c - m_p)|}{\sqrt{(m_g - m_p)^2 + (n_g - n_p)^2}}$$

$$d2 = 1 - \frac{|(m_g - m_p)(m_c - m_p) + (n_g - n_p)(n_c - n_p)|}{\sqrt{(m_g - m_p)^2 + (n_g - n_p)^2}}$$

Point 1. (m_p, n_p) = Previous position reference

Point 2. (m_c, n_c) = Current position reference

Point 3. (m_g, n_g) = Goal position reference

$$R = -(1 + d1)(1 + d2)$$

Training - MATLAB Virtual Robot

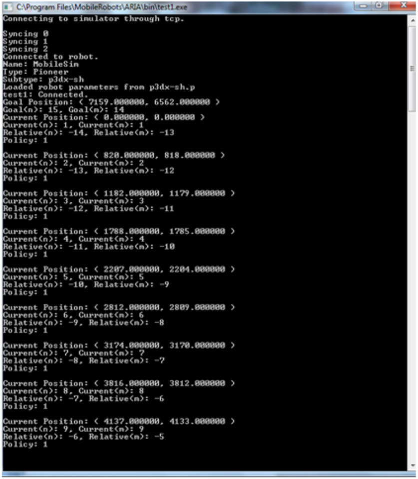
- $s = (m_{rel}, n_{rel})$
- $\mathcal{A}_s = \{(1,1)\ (1,0)\ (1,-1)\ (0,1)\ (0,-1)\ (-1,1)\ (-1,0)\ (-1,-1)\}$
- $R = -(1 + d1)(1 + d2)$
- Transition probabilities
    1. $P(s'|s,a) = 1$ for $s' = (m_{rel} - m_{change}, n_{rel} - n_{change})$
    2. $P(s'|s,a) = 0$ for all other s'.
- $\gamma = 0.95$
- $\text{size}(Q) = \Big((2m_{max} - 1) \times (2n_{max} - 1) \times \text{size}(\mathcal{A})\Big)$

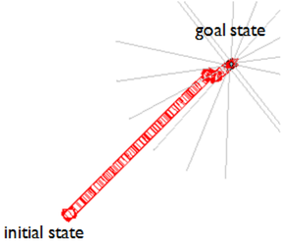
```
while( (m_rel ~= 0 || n_rel ~= 0) && (step <= step_max) )

    [max_cur_act, max_cur_index] = max(state_q_values(:,(m_rel + m_max), (n_rel + n_max)));
    m_rel_next = m_rel + next_rel(max_cur_index, 1);
    n_rel_next = n_rel + next_rel(max_cur_index, 2);

    if((abs(m_rel_next) > (m_max-1)) || (abs(n_rel_next) > (n_max-1)))
        state_q_values(max_cur_index,(m_rel + m_max), (n_rel + n_max)) = -Inf;
    else
        max_next_act = max(state_q_values(:,(m_rel_next + m_max), (n_rel_next + n_max)));
        reward = -((m_rel_next)^2 + (n_rel_next)^2)^0.5;
        state_q_values( max_cur_index,(m_rel + m_max), (n_rel + n_max)) = reward + gamma * max_next_act;
        m_rel = m_rel_next;
        n_rel = n_rel_next;
        step = step + 1;
    end
end
```

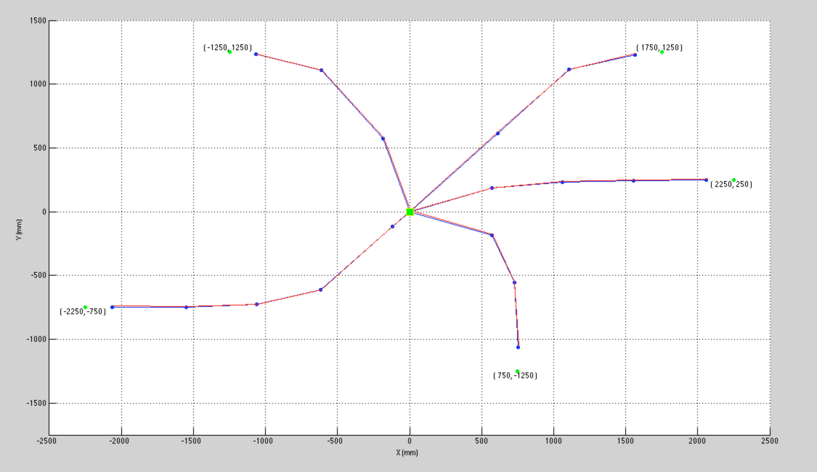
Testing (ARIA) - MobileSim Simulation





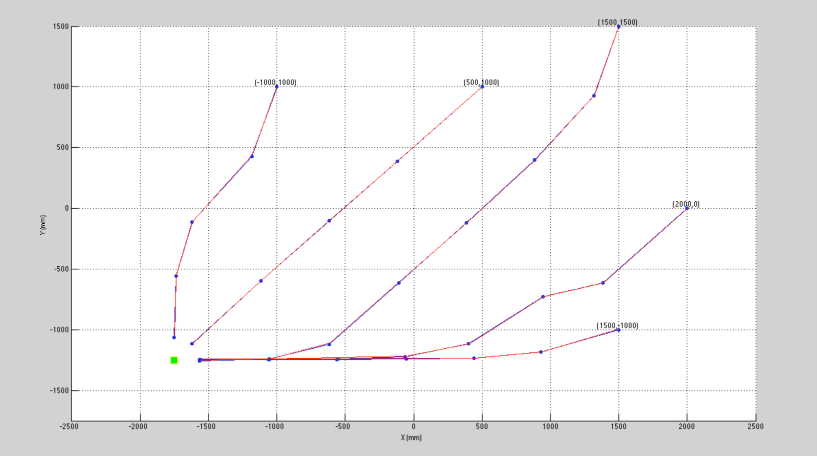
Testing - P3-DX Physical Robot

Fixed start reference, different goal references



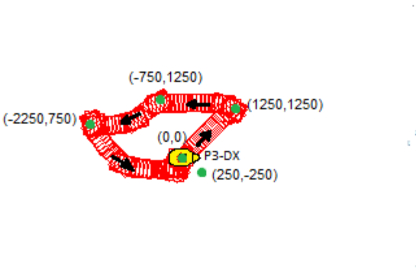
Testing - P3-DX Physical Robot

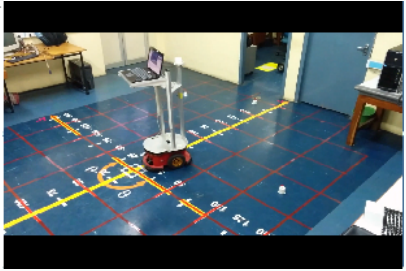
Fixed goal reference, different start references



Testing - P3-DX Physical Robot

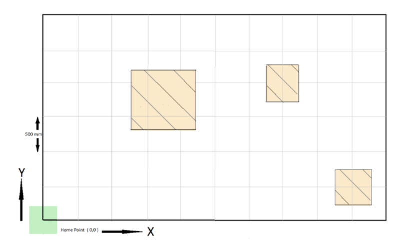
Multiple goals



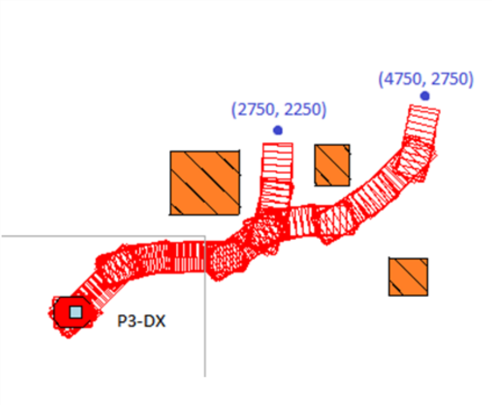


Training - MATLAB Virtual Robot

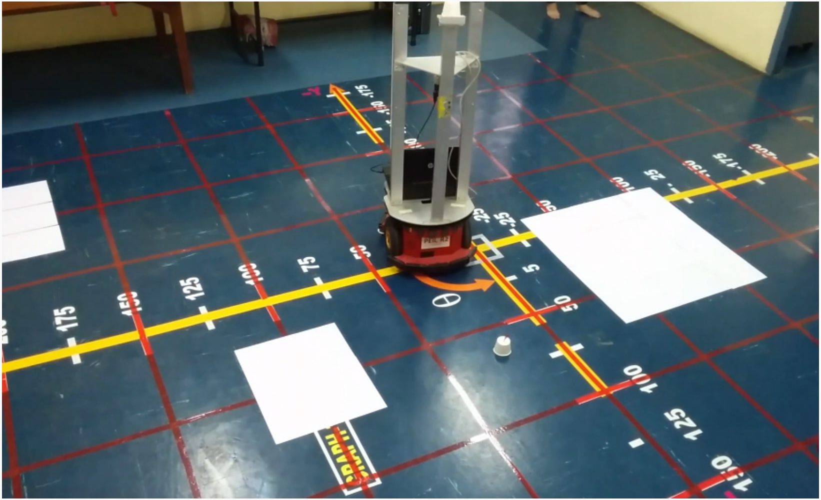
- $s = (m_c, n_c)$
- $\mathcal{A}_s = \{(1,1)\ (1,0)\ (1,-1)\ (0,1)\ (0,-1)\ (-1,1)\ (-1,0)\ (-1,-1)\}$
- $R = -(1 + d1)(1 + d2)$
- Transition probabilities
  1. $P(s'|s,a) = 1$ for $s' = (m_c - m_{change}, n_c - n_{change})$
  2. $P(s'|s,a) = 0$ for all other s'.
- $\gamma = 0.95$
- $\text{size}(Q) = \left(m_{max} \times n_{max} \times \text{size}(\mathcal{A})\right)$



Testing (ARIA) - MobileSim Simulation



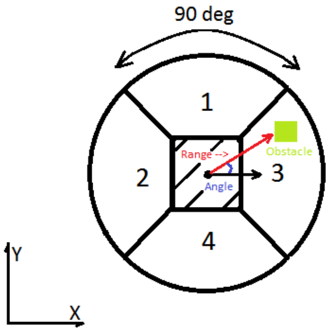
Testing - P3-DX Physical Robot

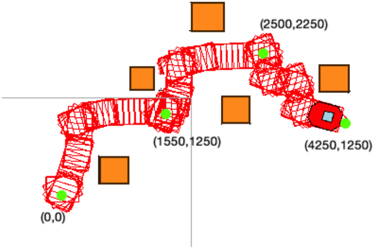


Training - MATLAB Virtual Robot

- $s \;=\; (\; m_{rel},\; n_{rel},\; O_N,\; O_W,\; O_E,\; O_S \;)$
- $\mathcal{A}_s = \{\; (1,0)\; (0,1)\; (0,-1)\; (-1,0)\}$
- $R = \;\; -(1\; +\; d1)(1\; +\; d2)$
- Transition probabilities
  1. $P(s'|s,a) = 1$ for
     $s' = (m_{rel} - m_{change}, n_{rel} - n_{change},\; O'_N,\; O'_W,\; O'_E,\; O'_S\;)$
  2. $P(s'|s,a) = 0$ for all other s'.
- $\gamma \;=\; 0.95$
- $\text{size}(Q) = \Big((2m_{max} - 1) \times (2n_{max} - 1) \times 2 \times 2 \times 2 \times 2 \times \text{size}(\mathcal{A})\Big)$

Testing (ARIA) - MobileSim Simulation





Testing - P3-DX Physical Robot

CONCLUSION

SUMMARY OF PROJECT

In a problem with discrete state- and action-spaces, by solving three independent tasks of mounting complexity, I successfully trained the robot to go from **any** initial position to **any** goal position, in an unknown environment with **obstacles**, by engaging its SONAR range device. This algorithm may also be applied in dynamic obstacle navigation.

Each task was performed in three stages:

1. Training on a virtual robot using MATLAB.
2. Testing in simulation, using MobileSim, a simulator for MobileRobots. The arena is set up using ARIA, the higher-level software interface for MobileRobots platform.
3. Testing on the physical robot Pioneer 3-DX. [1]

---

[1] Video for testing can be found here - `https://www.dropbox.com/sh/fd2b2p1xi6wsjnu/AACxvR9695aX-UGdan80Ta3ba?dl=0`

FUTURE WORK IN THIS AREA

1. Continuous State and Action Spaces
2. Options - An example set of options that may be learnt are:
   2.1 Avoid obstacle
   2.2 Go to goal
   2.3 Follow the wall

REFERENCES

[1] Andrew G Barto. Reinforcement Learning: An Introduction. MIT press, 1998.

[2] Andrey V Gavrilov and Artem Lenskiy. Mobile robot navigation using reinforcement learning based on neural network with short term memory. In Advanced Intelligent Computing, pages 210–217. Springer, 2012.

[3] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. Journal of artificial intelligence research, pages 237–285, 1996.

[4] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In Abstraction, Reformulation, and Approximation, pages 212–223. Springer, 2002.

[5] Mu-Chun Su, De-Yuan Huang, Chien-Hsing Chou, and Chen-Chiung Hsieh. A reinforcement-learning approach to robot navigation. In Networking, Sensing and Control, 2004 IEEE International Conference on, volume 1, pages 665–669. IEEE, 2004.

Questions?