Markovian Decision Process in Bipedal Locomotion

Bratin Mandal[1], Harsh Sharma[1], Prasanna Paithankar[1], Sanchit Yewale[1],
Somya Kumar[1], Swarnabh Mandal[1]

[1]Department of Computer Science and Engineering

1. Introduction

In this note, we will demonstrate how Markov processes can be used to model environmental feedback toward an agent's action. The agent here can be a legged robot or even a toddler learning to walk. Understanding how humans learn to make near-optimal decisions in voluntary actions has been an important research domain in neuroscience and psychology. On the other hand, engineers are taking inspiration from the field to help their automation(controllers and robots) take better decisions by learning from the outcome of the previous ones.

2. Framework

We define the agent as a means to instantiate actions that change a set of environmental parameters. The transition function can be represented by a transition probability matrix(tpm). We set a reward function as a parameter to be maximized (in the long run). The environment would provide a reward to the agent which is the function of the current state and action taken. The agent is supposed to learn to perform actions that lead to higher cumulative rewards in the long run of the operation. The goal of the agent is to finally maximize the value function. We mathematically state the reward, value function, state, and action value function respectively. The modelling follows Markov process depending upon the current state and action.

$$R(s,a) = \mathbb{E}[R_{t+1}|s_t = s, A_t = a] = \sum_{r\in R} r \sum_{s'\in S} P(s', r|s, a)$$

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$\mathbb{V}_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

$$\mathbb{Q}_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

3. Modelling the Interaction & Learning

The agent is rewarded based upon the orientation and goal meeting behaviour. The reward is higher for on foot mobile orientation and highly negative for bot being sideways or upside down position. The reward has another parameter based upon reaching the destination.

The action space is infinite and is dependent on the degree of freedom the robot has (the number of actuators). The state space consists of the coordinates completely characterizing the robot spatially. The following are the Bellman Equations to solve for the optimal policy and action space.

$$\begin{aligned}
\mathbb{V}_*(s) &= \max_{a\in A} \mathbb{Q}_*(s,a) \\
&= \max_{a\in A} \left(R(s,a) + \gamma \sum_{s'\in S} P^a_{aa'} \mathbb{V}_*(s')\right) \\
\mathbb{Q}_*(s,a) &= R(s,a) + \gamma \sum_{s'\in S} P^a_{aa'} \mathbb{V}_*(s') \\
&= R(s,a) + \gamma \sum_{s'\in S} P^a_{aa'} \max_{a'\in A} \mathbb{Q}_*(s',a')
\end{aligned}$$

4. Solving MDP's

We can obtain exact as well as approximate solutions for MDP's. Dynamic programming and strictly converging iterative methods are used to solve exactly. Where the problem setup is more complex getting the exact solution is near to impractical, hence Monte Carlo and Temporal Difference learning algorithms play a role.

The iterative algorithm presented below strictly converges to the optimal value function is it is learnable.

$$\begin{aligned}\mathbb{V}_{t+1}(s) &= \mathbb{E}[r + \gamma\mathbb{V}(s')|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} P(s', r|s, a)(r + \gamma\mathbb{V}_t(s'))\end{aligned}$$

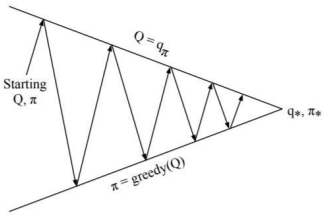

Figure 1: Iterative Convergence

5. Monte Carlo Methods for Unknown MDP's

Monte Carlo simulation is performed by sampling various actions during a state with their respective probabilities. The outcome is analyzed as the current estimated value function given a state indicator. Alternatively Temporal Difference learning methods are utilized in SARSA (on policy learning) and Q learning (off policy learning).

$$\mathbb{V}(s) = \frac{\sum_{t=1}^{T} \mathbb{1}[S_t = s]G_t}{\sum_{t=1}^{T} \mathbb{1}[S_t = s]}$$

6. Putting it all Together

The training and characterization of reinforcement learning based development is industrially performed in python. The algorithms iterate or use stochastic algorithms as outlined above. Typical training periods take several hours on high end hardware clusters. These systems continually learn during their operational span similar to that during training. Overall macro-training results in highly functional machines like Spot by Boston Dynamics and Cassie by Oregon State's Dynamic Robotics and AI Lab.

7. Conclusion

Making robots walk on natural terrains is difficult. Programming them explicitly with classical control algorithms works only in limited well-defined environments and could not be scaled. Reinforcement learning provides the freedom to take decisions through continual learning similar to that of living beings. Not only RL provides better control agents but also helps us to gain insight on how humans learn and take decisions.

References

[1] Richard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA.

[2] Xie, Zhaoming & Berseth, Glen & Clary, Patrick & Hurst, Jonathan & Panne, Michiel. (2018). Feedback Control For Cassie With Deep Reinforcement Learning.

[3] Fakoor, Mahdi & Kosari, Amirreza & Jafarzadeh, Mohsen. (2016). Humanoid robot path planning with fuzzy Markov decision processes. Journal of Applied Research and Technology. 14. 300–310. 10.1016/j.jart.2016.06.006.

[4] Shi Y, Li S, Guo M, Yang Y, Xia D, Luo X. Structural Design, Simulation and Experiment of Quadruped Robot. Applied Sciences. 2021; 11(22):10705. https://doi.org/10.3390/app112210705

Supplementary

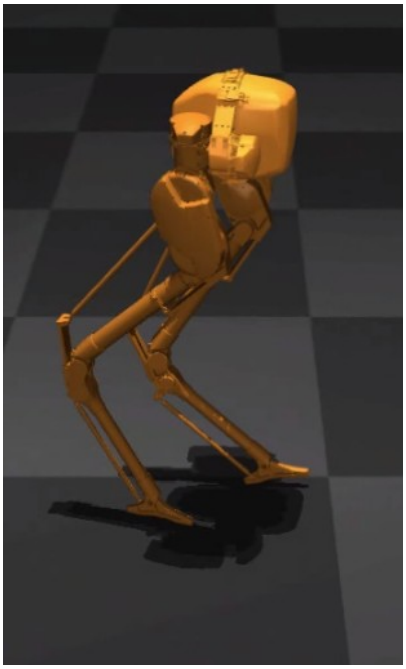

Figure 2: Cassie Bipedal Robot developed by Oregon State's Dynamic Robotics and AI Lab

Figure 3: Spot Quadraped developed by Boston Dynamics