B.Tech

Project Report

Semester (VI)

Navigation of quadruped on uneven terrain using reinforcement learning

Student:
Sudhir PratapYadav
(B14EE033)

Mentor:
Dr. Suril V. Shah
Assistant Professor
Department of Mechanical Engineering

May 22, 2021

Abstract

In this project, reinforcement learning would be used to make a quadruped learn to navigate on uneven terrain. The planner would take terrain as input and would generate a sequence footsteps. These foot positions could be generated based on goal or walking direction. Specifically temporal difference (TD) methods would be used to train an RL agent. Initially, the robot would be trained on computer simulation and then learned parameters would be transferred to the actual robot controller.

1 Introduction

Legged locomotion of robots is very important. It allows the robot to navigate the unknown and uneven terrain. Robots with the capability to navigate in a cluttered environment can be used in many situations like disaster management, exploring unknown areas, completing a task in hazardous areas etc. State of the art techniques depend upon modeling of robot and environment and then devising sophisticated control laws to make robot navigate in an environment. But this is dependent on robot dynamics and how accurately it is modeled. This lead to very inefficient walking methods. Recent developments in the area of reinforcement learning can be exploited to make a robot learn to walk like a toddler. Once a learning is complete, it can navigate on learned environment as well as it can generalize to new environments. This is a very general method, therefore the same program can be used on different robots and they all can learn simultaneously and from each other.

2 Project Objective

This project is about locomotion of quadruped on uneven terrain but before that major goal is to make quadruped move on the plane surface. This would involve following steps

- Creating a simulation of quadruped navigation in C++.
- Creating a Reinforcement Learning (RL) agent in C++. This agent would learn by exploring different action sequences (i.e. where to place leg of the robot) and reviewing stability rewards given by environment.
- Temporal Difference (TD) methods would be used for training RL agent. TD(0) method is good enough for making quadruped walk on uneven terrain.

3 Methodology

3.1 Preliminaries

3.1.1 Quadrupedal Walking

The pattern of movement of the limbs of animals is called gait. Variety of gaits are used by animals. Current methods of quadruped walking depend upon careful modeling and control of quadruped based on stability of the robot. Stability Margin

S_m, is defined as a minimum distance of the vertical projection of the COM to the boundaries of the support polygon in the horizontal plane. The main body of the quadruped is modeled as rectangle and four legs are attached at corners.

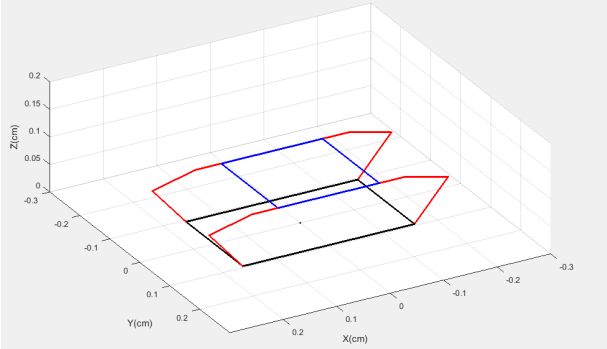

Figure 1: Quadruped Model

3.1.2 Reinforcement Learning

In reinforcement learning, we model a system to be controlled as having a set of possible states S, and a set of possible actions A. Further, we are given a reward function R, so that $R(s,a)$ is a reward for taking action a in state s. A policy is any function π mapping from states to actions. We take action in given state based on policy, $a = \pi(s)$. The value function $V^{\pi}(s_0)$ for policy π is defined as expected reward agent is going to get starting from this state till the end while following policy π. The optimal value function can be found using Bellman equation of optimality. Temporal Difference (TD) learning is a method of learning optimal value function from many observations (states, actions, rewards etc.) that agent collects while exploring the environment.

3.2 Reinforcement Learning for Quadruped locomotion

In particular problem of quadruped walking RL agent is learning from episodic sequences (i.e. Agent start a new sequence of actions as soon as stability margin is less than zero). States is defined as the linear combination of weights and feature vector. Feature vector consists of error in direction of walking, configuration of robot and terrain features. Action space is defined as the grid of coordinates where leg needs to be placed. Leg sequence needed to be lifted is predefined. Greedy policy (choose an action that takes to state with max value function) is used selecting the best action for given state. Positive reward is given if the robot is stable at given state. Negative rewards are given for error in direction of motion and a large negative reward is given if robot falls down.
RL agent starts exploring this environment and collects rewards and take actions. Based on received rewards agent keeps updating its value function. After much of exploration, value function converges to optimal value function and robot starts navigating the environment.

3.2.1 State Space

State Value function $V(s)$ is approximated as linear combination of features of state s.

$$\hat{\boldsymbol{V}}(s) = W^T \phi(s), \tag{1}$$

where $\phi(s)$ is vector of features of the state s, and W is a parameter vector that would be learned. $\phi(s)$ ($\mathbb{R}^{35\times 1}$) is state feature vector. It consists of following features

- 3 body orientation features expressed as, $1-\cos(\phi), 1-\cos(\theta), 1-\cos(\psi)$
- 2 features for error in direction of motion, e_1 and e_2. e_1 is perpendicular distance of COM from actual trajectory of motion. e_2 is angular error from direction of motion of COM.
- 12 coordinate features, 3 for each leg $\{x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z3, x_4, y_4, z_4\}$. Coordinates are defined w.r.t. center of body of quadruped (same as COM).
- 12 joint angles
- area of support polygon
- height of COM
- 4 binary features for leg to be lifted $\{l_1, l_2, l_3, l_4\}$, where $l_i \in \{1, 0\}$. 1 if that leg is to be lifted next.

3.2.2 Action Space

Action space is defined as grid of locations where foot can be placed. Sequence of lifting leg is predefined i.e. 1 4 2 3. Action $a \in A, where A = \{0, 1, 2, \ldots, 68, 69\}$ is set of indices of grid defined as shown in figure 2. Therefore, if starting position of foot i, p_i^s is $\{x_i^s, y_i^s, z_i^s\}$ then after taking action a final position of foot i, p_i^f would be $\{x_i^s + a_x, y_i^s + a_y, z_i^s\}$, where

$$a_x = (a \bmod 5) - 2, \tag{2}$$
$$a_y = \lfloor a \div 5 \rfloor + 1. \tag{3}$$

3.2.3 Reward Function

This section explains reward function and dynamics of environment. Reward $r \in R$, where $R = \{r_s, r_e\}$. r_s is reward for stability. It is huge negative if robot is unstable otherwise it is equal to static stability margin S_m. r_e is negative reward for error in motion of quadruped. It is negative of absolute sum of motion errors i.e. $r_e = -|e_1 + e_2|$, where e_1 and e_2 are motion error as defined in state space.

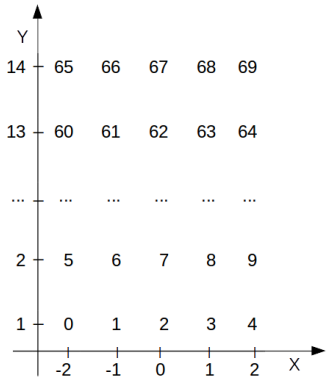

Figure 2: Action Space indices allocation

3.2.4 Evolution of state of Quadruped

Dynamics of environment are very simple. RL agent selects the best action a^* based upon State Value Function $V_\pi(s)$ using greedy policy π. Thereafter, State of quadruped is evolved based on action. This requires placing foot of leg to lifted (l_i) to its next location based on action a^*.

COM is shifted to appropriate position and next leg to be lifted is changed. Finally other state parameters are updated like stability margins, errors in motion, area of support polygon etc. Reward is calculated same as discussed in previous section.

3.2.5 TD(0) algorithm

Algorithm 1 illustrates implementation of TD(0).

Algorithm 1 TD(0) algorithm for quadruped navigation

$W = \{0\}$ // initializing weights to zero, these are to be learned
$\alpha = 0.05$ // learning rate
$n = 1$
$numTr = 1000$
for $n \leq numTr$ **do**
  $a = \pi(\phi(s), W)$
  $s' = \{0\}$ // initializing next state to zero
  s = initializeState() // initializing current state
  S_m = getStability(s)
  while $S_m \geq 0$ **do**
    $a = \pi(\phi(s), W)$
    $[r, s^{'}]$ = evolveState($\phi(s)$, a)
    $\delta W = \alpha(r + V_\pi(s^{'}, W) - V_\pi(s, W))$
    $W_i = W_i + \delta W, \forall\, i \in \{1, 2, \ldots, 35\}$
    S_m = getStability($s^{'}$)
    $s \leftarrow s^{'}$
  end while
end for

4 Results and Discussion

RL agent is able to make quadruped walk up to 4-6 steps. Based on shifting of COM maximum steps taken was 9 when COM is shifted by 1cm in y-direction on each step. Joint angles in state features are of no use and therefore could be removed. Main bottleneck in increasing number of steps in a walk is shifting COM properly and defining state space and action space appropriately. Some ideas are presented in next section.

COM shifting strategy	Max. steps
Shift COM to center of support polygon	6
0.5cm shift in Y	2
1cm shift in Y	9
2cm shift in Y	5

Table 1: Maximum number of steps taken by Quadruped after learning with different strategies (Learning is done for 100 trajectories).

5 Ideas and Further work to be done

Some ideas for improving number of steps are listed below.

- Motion of COM should be either be based on motion or should be learned.
- Remove Joint angles from state space.
- Use Non-Linear state approximation (Deep NN).
- Use Q-learning instead of TD algorithm.

References

[1] Lee, Honglak, et al. "Quadruped robot obstacle negotiation via reinforcement learning." *Robotics and Automation*, 2006. *ICRA* 2006. *Proceedings* 2006 *IEEE International Conference on. IEEE*, 2006.

[2] Kohl, Nate, and Peter Stone. "Policy gradient reinforcement learning for fast quadrupedal locomotion." *Robotics and Automation*, 2004. *Proceedings. ICRA'04*. 2004 *IEEE International Conference on. Vol. 3. IEEE*, 2004.

[3] Kajita, Shuuji, and Bernard Espiau. "Legged robots." Springer handbook of robotics, *Springer Berlin Heidelberg*, 361-389, 2008.

[4] Woering, R. Simulating the first steps of a walking hexapod robot, *University of Technology Eindhoven* , 2011.

[5] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature 518.7540*, 529-533, 2015.

[6] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. *Vol. 1. No. 1. Cambridge: MIT press*, 1998.

Student:
Sudhir PratapYADAV

Mentor:
Dr. Suril V. SHAH