Bipedal Walking

EE597 RL Project Report

Vanshaj Agrawal
Department of Computer Science and Engineering
The Pennsylvania State University
vma5214@psu.edu

Abstract

Although remarkably common in the animal world, two-legged locomotion has been a particularly challenging open research problem. The ultimate goal of bipedal walking research is to develop a framework that enables an agent to learn to 'walk' in any unknown environment where this is permitted by the laws of physics. The objective of this project is to train an agent to maintain human gaits with maximum horizontal velocity in the Mujoco Humanoid-env. A first-principles approach to the implementation of the Soft Actor-Critic (SAC) stochastic policy optimization algorithm has been discussed and it's slow-convergence and potential implications analyzed.

1 Introduction

Background. The challenges of bipedal motion often beg the question of what is so special about the series of gaits that we humans so effortlessly adopt. The act of walking involves using one limb to propel the body forward using friction from the ground while using the second limb to balance the resulting torque, which helps balance the body and minimize net velocity in all but the horizontal directions over the course of a trajectory. Additionally, the motion of human joints is equivalent to the rotation of actuators in robots, and it is the torque produced by these joints that controls the gait and trajectory of the robot.

Relevance of RL techniques. To reiterate, the ultimate goal of bipedal walking research is to develop a framework that enables an agent to learn to 'walk' in any unknown environment where this is permitted by the laws of physics. This means that supervised learning techniques cannot lead to a complete solution, since out-of-domain environments can always be constructed such that the algorithm shall fail to make correct predictions, in which case the agent shall have to learn to train and generalize over these out-of-domain examples. This is exactly what reinforcement learning techniques seek to achieve. In particular, deep reinforcement learning techniques seek to obtain 'good' data or trajectories and to use this data to improve their policy in proportion to its 'goodness'. A metric is needed to quantify this goodness, and this is naturally encoded into the reinforcement learning framework in the form of rewards. Due to all these reasons, reinforcement learning techniques have become very promising approaches to this problem.

2 MDP Formulation

Goal. To formulate bipedal walking as an MDP, the quantities of <S, A, R, P, γ> also have to be defined. The choice of how to define these quantities in turn depends on the class of RL techniques used. Since it is often infeasible to have a real robot explore the world on account of cost and convenience, a baseline RL policy is often first obtained using simulations. Often, the model is first trained to operate in an idealized simulated environment, since this helps to easily test the plausibility of a class of techniques. Once this succeeds, it is often desirable to use a stochastic simulation that models the variability of the real world. Failure to effectively model this stochasticity leads to a

sim-to-real gap wherein an algorithm that shows great success in a simulation, fails to carry this success across to the real world especially while navigating new terrainn. The objective of this project is thus, to develop and test deep RL frameworks to successfully learn end-to-end policies for bipedal walking.

Simulation Environment. To establish a baseline, the Mujoco Humanoid-env [1]is being used to train the agent. The environment consists of a humanoid agent on a floor with an infinite horizon. The 376 dim state space consists of the various velocities that the humanoid agent can measure, and the continuous action space consists of the various torques the humanoid can apply to maximize the value function. Additionally, the reward here is obtained from the environment and contains terms that encourage the humanoid to adopt a bipedal gait while walking with the maximum horizontal velocity.

State Space. (shape = 44, number of variables = 376) Observations consist of positional values of different body parts of the Humanoid, followed by the velocities of those individual parts.

Action Space. (shape = 17): Each element of this array represents the values of the torques that may independently applied at the hinge joints.

Rewards. The rewards are computed by the simulated environment itself and are a function of both the previous state and the subsequent action. The reward function explicitly encourages bipedal walking with maximum horizontal velocity by including the following terms:

1. healthy_reward: Every timestep that the humanoid is alive
2. forward_reward: A reward of walking forward which is measured as forward_reward_weight * (average center of mass before action - average center of mass after action)/dt. dt is the time between actions and is dependent on the frame_skip parameter.
3. ctrl_cost: A negative reward for penalising the humanoid if it has too large of a control force. Such an action may damage the humanoid and is an unnecessary expenditure of energy.
4. contact_cost: A negative reward for penalising the humanoid if the external contact force is too large.

total reward returned = healthy_reward + forward_reward - ctrl_cost - contact_cost

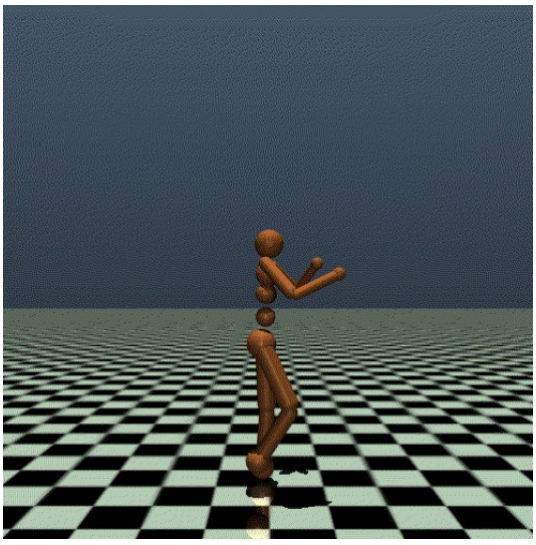

Figure 1: Mujoco Humanoid-env

3 Related Work

With the action space being continuous, this necessitates policy-based RL algorithms using neural networks as function approximators. Bao et al [2] provide a survey of the techniques recently used by the more successful approaches and this serves as a comprehensive guide to state-of-the-art bipedal walking research. Their survey emphasizes a crucial choice, that is whether to learn a hierarchical control scheme or an end-to-end model. A hierarchical control scheme involves a series of systems each with a different objective. Such an approach may involve a separate system for path planning and joint actuation. This increases the flexibility of optimizing each subsystem. On the other hand, an end-to-end model will directly output the actuator torques with the hope that the actuation will implicitly lead to a successful trajectory. Such an end-to-end model may result in a better globally optimum policy but it is much harder to successfully search for one in the much larger state space.

Singh et al [3] also made some interesting observations on the sim-to-real gap which they found was most impacted by degraded torque-tracking as a result of a mismatch in the desired torque determined by the policy in the simulation and the torque that is actually manifest in the real world. Their simulation incorporates this degraded torque-tracking and imperfect motor controller feedback and it would be interesting to compare a policy trained in their environment to other out-of-domain stochastic environments to see if the agent succeeds to walk out-of-the-box.

4 Approach

A first principles approach has been used to identify the baseline algorithms that are likely to learn good policies. Since the algorithm has to operate in continuous action spaces, a policy gradient-based approach becomes crucial. Additionally, due to the high (s, a) dimensionality, using a neural network as a functional approximator also becomes inevitable. These two factors make DDPG a natural choice. Moreover, DDPG can be run on-policy although an off-policy scheme may be more sample efficient since this would enable buffer replay. But DDPG also has a deterministic policy which could inhibit exploration. Therefore, the SAC algorithm, which is known to be improvement over DDPG, has been shortlisted to train a stochastic policy in an off-policy fashion. Bao et al, also substantiates this claim by mentioning that SAC's great success at converging to a good policy for bipedal walking with high sample efficiency.

Soft-Actor Critic Algorithm (SAC). The SAC algorithm [4] has three fully connected networks, one representing the actor or policy network $\pi(\theta)$ and the other two critics or Q networks $Q(\phi_1), Q(\phi_2)$. Here the Q networks are trained to predict the action-value of a particular state and so they map the 376 state variables to a continuous 17 dim action space. The policy network is also concurrently trained to predict the action that gives the max Q for a given state. But now the policy network outputs the mean and std_deviation of a normal distribution, from which samples are drawn to get the predicted action in continuous space. It thus maps the 376 state variables to two 17 dim parameters, i.e., the mean + std_deviation of this normal distribution.

Model Training. The two networks are jointly optimized such that sampling from normal distribution of the policy network gives the action with max Q which is calculated using the Q networks.
Here, the critics are trained by bootstrapping from the Bellman equation with the Mean Squared Bellman Error (MSBE) as loss. While bootstrapping, the Q network listens to the more pessimistic critic, which is the one outputting the lower Q value. Additionally, to explicitly encourage exploration, an entropy term is added to the Q back-up equations with the value of the term being inversely proportional to the probability of visiting that state under the target policy. Moreover, the Q networks are not updated in every iteration, and instead Polyak averaging is periodically applied after a certain number of timesteps to stabilize training. A replay_buffer is also used to store and sample trajectories in minibatches to increase sample efficiency.
The same model architecture is used for all networks, composed of 3 fully connected hidden layers, each followed by a ReLu activation as depicted in Figure 2.

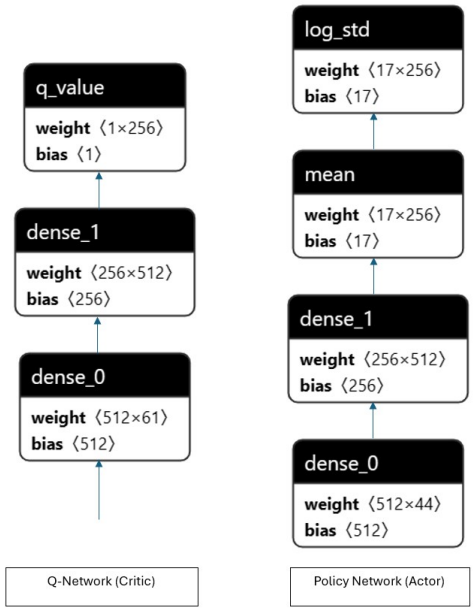

Figure 2: Model Architecture

5 Results

5.1 Hyperparameters

The SAC algorithm was implemented from scratch and it's graph computation over several iterations was closely analyzed. The model was run for 1500 iterations and all 3 networks were optimized using the Adam optimizer. Learning rate was decayed from 5e-4 and a buffer capacity of 1000000 states was maintained from which minibatches of size 64 were sampled. The average reward over all past timesteps was used to gauge the convergence of the policy and the instantaneous reward was tracked using Weights and Biases.

5.2 Results

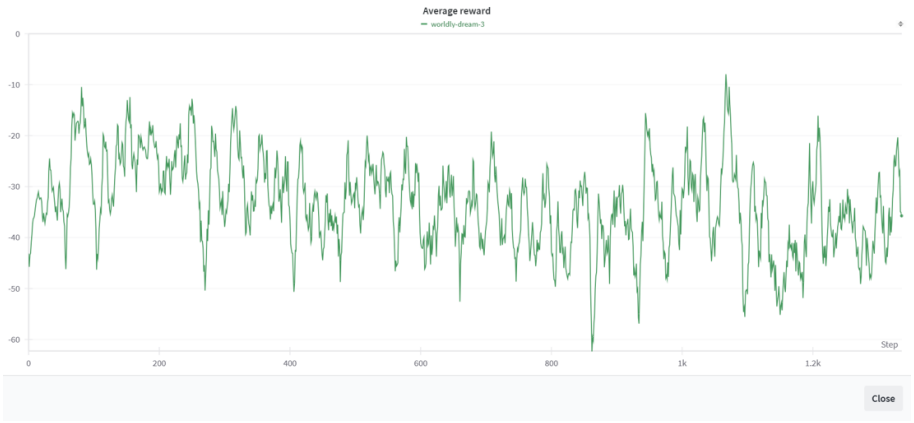

Figure 3: Model Architecture

As shown in the Figure 3, the model fails to get a positive reward in the first 1500 iterations. This is because everytime the humanoid attampting to take a step, it fell down thereby ending the episode.

These results are fairly in line with expectations since the SAC models often have to be run for several million iterations before they start to converge to a good policy.
Due to a lack of compute, training for longer durations was infeasible but since the initial results tally with the results reported in literature [2], it is expected that SAC will succeed in learning better policies if run for a greater number of iterations.

6 Analysis

To obtain better results, the key contributors to policy convergence were analyzed. The rewards here are chosen by the simulation environment and so cannot be directly optimized. Increasing network complexity could help, but it still be the bottleneck leading to negative rewards. Low entropy too was ruled out since entropy is always positive.
Thus it was hypothesized that the model is most likely choosing a wrong policy and getting stuck at a bad local optimum. Thus perhaps reducing step size or clipping the standard deviation further could help resolve this. Additionally reducing the high-dimensional state space from 376 dim using polynomial features or SVD could also be promising and should be investigated.

7 Conclusion

The Soft-Actor Critic algorithm was used in an attempt to train the Humanoid agent to learn bipedal walking on a Mujoco environment. The SAC algorithm was implemented from scratch and it's graph computation over several iterations was closely analyzed. Ultimately, it was found that SAC does not learn the optimal policy within the first 1500 iterations and that this may not be a limit of the algorithm itself. Other optimizations discussed thereafter may also help speed up policy optimization in future work.

8 Additional Materials

The source code has been made publicly available at https://github.com/120205690/WalkRL

9 Acknowledgements

The author is extremely grateful to the faculty and course staff of EE597 Reinforcement Learning Spring '24, for introducing these techniques and guiding the project to fruition

References

[1] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[2] Lingfan Bao, Josephine N. Humphreys, Tianhu Peng, and Chengxu Zhou. Deep reinforcement learning for bipedal locomotion: A brief survey. 2024.

[3] Rohan P. Singh, Mehdi Benallegue, Mitsuharu Morisawa, Rafael Cisneros, and Fumio Kanehiro. Learning bipedal walking on planned footsteps for humanoid robots. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 686–693, 2022.

[4] Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290, 2018.