Building Locomotion Policies Using Online Reinforcement Learning

Jared Buchanan, Steven Tran, Donald Dean, and Michael Yip, Ph.D.

University of California, San Diego Department of Bioengineering

Introduction

Locomotion robotics currently perform a wide range of tasks such as surgery or bomb disposal and the number of applications will surely increase as the field advances. The software embedded on these systems that control how the hardware operates to create locomotion is just as important to successful robotic operation as the hardware itself. Therefore, there is a need to create efficient locomotion policies with a low cost/effort development process.We seek to implement a simple reinforcement learning algorithm that allows the snake-like robot to self-learn an optimal locomotion policy online. *Reinforcement learning* (RL), a subset of machine learning, is focused on goal-directed learning by mapping situations, or *states*, to *actions*, with the end goal being the maximization of a *reward* signal (Sutton).

Reinforcement Learning

Reinforcement Learning provides a framework that allows an agent to self-learn an optimal locomotion policy through trial and error interactions with the environment. An optimal policy is one that maximizes cumulative reward for every state-action mapping.

State (s): variables describing the current system's orientation, location, speed, etc.
Action (a): something the agent does to move to the next state.
Policy ($\pi(s) = a$): a mapping of states to actions to be taken.
State-Action Space: all possible combinations of state and actions.
Reward Function: a user-defined function that provides a scalar feedback called **reward (R)** regarding how an action performed for a given task.
Markov Property: a necessary property which is satisfied if the state signal encodes all the relevant information regarding the actual state.
Q* function: returns the expected cumulative reward if an agent starts in state (s), follows action (a), and then follows the optimal policy thereafter.
Curse of Dimensionality: adding new dimensions combinatorially expands the state-action space, thus making training combinatorially more time consuming.

Design Goals

- Low dependency and user input.
- High quality locomotion policy output.
- Low convergence and runtime to find the policy.
- Low software development cost.

Final Algorithm

Based on our decision matrix, we implemented a **model-free Q-learning w/ IFDD** algorithm for creating a generalized locomotion policy (Geramifard).

Q-Learning tabular algorithm

The Q-function below is used to update the value of the current state based on the next chosen action.

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha_t * (r_{t+1} + \gamma * (MAX_a(Q(s_{t+1}, a)) - Q(s_t, a_t)))$$

IFDD linear function approximation technique

1. Used to approximate the Q-function.
2. Start with sparse feature representation of state.
3. Measure magnitude of temporal difference error and add to stored sum for every possible feature conjunction.
4. If the stored error sum exceeds a user-defined threshold, add feature conjunction to feature set as new feature

State-Action-Space Representation

- **Number of joint DOF per joint:** 1
- **Joint angle limits:** -108° to 108°
- **Joint discretization:** 36°
- **Orientation Global DOF:** 3
- **Orientation Vector angle limits:** 0° to 360°
- **Orientation Vector Discretization:** 36°
- **Action Choices per Angle:** -36°,+0°,+36°

- **State-Action-Space Size (2 joints)**: 441,000
- **State-Action-Space Size (3 joints)**: 9,261,000

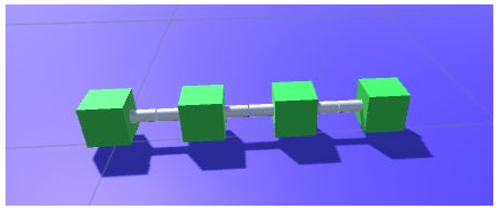

Image of the model used for testing algorithms in a physics simulator.

Results

The distance traveled for a given number of iterations is shown. The means and standard deviations of the distances are shown in the below graphs.

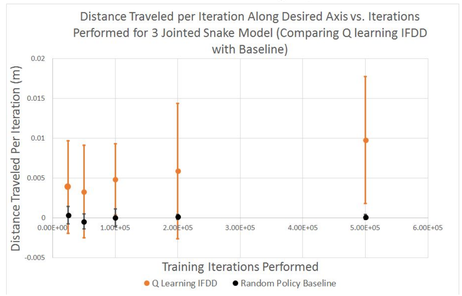

Figure 1

Q-learning w/ IFDD performs better than random baseline policy (Figure 1). There is improvement with more iterations (i.e more training data).

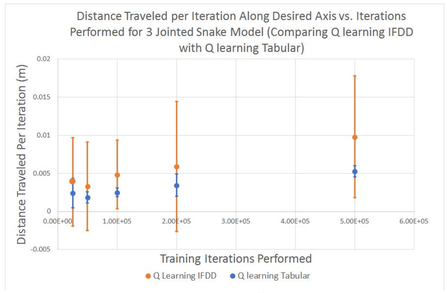

Figure 2

Q-learning w/ IFDD also performs better than Tabular Q-learning (i.e. Q learning w/o IFDD), thus justifying the inclusion of IFDD into our algorithm (Figure 2). Theoretically, IIFD enhanced Q learning performance by generalizing information from observed data.

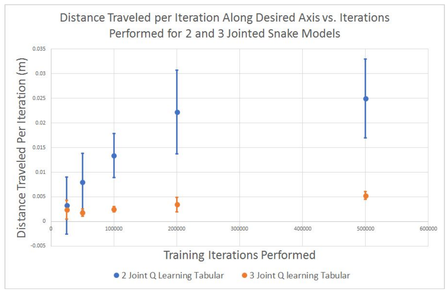

Figure 3

Lastly, we use the same Q-learning algorithm with a 2 and 3 jointed snake. The 2-jointed snake performs better, thus confirming the theoretical prediction that smaller state-action spaces result in faster learning (Figure 3).

Conclusion

The results show that the Q learning with IFDD algorithm does in fact produce a policy that learns from experience in a simulated physics environment. This physics environment and the associated algorithms can be applied to any jointed robotic model in order to assist potential researchers with developing locomotion policies. This testing could ultimately serve as a feasibility test for the algorithm and the associated state-action space representation before putting encoding it on a physical robot. Alternatively, it could serve as a method to pretrain the snake's locomotion policy before putting it on the robot.

However, there are certain limitations associated with this software platform. First, the curse of dimensionality severely limits the tractability of high dimensional robotic models. We were only able to approach at maximum a 3 jointed snake. In the same vein the curse of dimensionality severely limits the complexity of the environment and the complexity of the task that can be successfully trained. Nonetheless, successful implementation of the IFDD function approximation technique with Q learning should be highlighted as one of the major accomplishments of this design that will hopefully assist researchers in designing successful locomotion policies.

Future Directions

We would like to implement more robust function approximation methods with Q learning such as neural networks or even DeepQ Learning. This could significantly improve performance and time to convergence of the locomotion policy. Additionally, we would like to do further testing into hyperparameters such as learning rate, discount rate, and relevance threshold and how they affect performance. Lastly, we would like to test this algorithm on different jointed models in order to obtain a wider range of experimental results.

References

Geramifard, Alborz, Christoph Dann, and Jonathan P. How. "Off-policy learning combined with automatic feature expansion for solving large MDPs." *Proc. 1st Multidisciplinary Conf. on Reinforcement Learning and Decision Making*. 2013.

Sutton, Richard S., and Andrew G. Barto. *Introduction to reinforcement learning*. Vol. 135. Cambridge: MIT Press, 1998.

Acknowledgements: We would like to thank our mentor Michael Yip, PhD. for providing this project. We'd also like to thank Bruce Wheeler, PhD. and the bioengineering department at UCSD for providing additional support throughout this course.