CS8803-O03 Reinforcement learning Project 2 report

Rohan D. Kekatpure
Email: rdk@gatech.edu

I. Introduction

The aim of this project is to train a reinforcement learning (RL) agent in a discrete-action continuous-state-space environment. The chosen agent is the lunar lander in the Box-2D environment. This project brings together many aspects of RL and the course together. The project is a miniature version of latest advances in the field of RL. As such, despite the pain and nearly 100+ hours spent on this project, it has forced synthesis of the class material.

II. Implementation methodology

Our implementation for this project is a simplified version of the methodology presented in the DQN paper[1]. While the RL part of the overall algorithm is identical to Algorithm 1 in the paper, our function approximation technique differs in from the DQN implementation in two important ways: (1) we use a simple, fully connected feed-forward neural network for function approximation (as opposed to convolutional net (CNN) in the DQN paper) and (2) we feed 8-component state augmented by a **one-hot-encoded** action vector.

In detail, we used a TD(0)-like RL algorithm with an ϵ-greedy exploration strategy. The parameter ϵ is initialized at 1.0 and is decayed by 1% at each step until it reaches a minimum of 0.1. We settled on **experience replay** as in the DQN paper after our failed experiments with batch-mode updates using states in a single episode. We noticed that training with states from a single episode makes the weights diverge. For our agent we fixed replay memory at 50000 the minibatch size at 32.

A. Motivation for neural network function approximator

RL success stories (TD-Gammon, Atari, Go) leave little doubt about the efficacy of neural nets as the function approximators. Yet, it is not obvious (at least to this author) why other supervised algorithms would not be able to outperform neural nets.

To test the strength of other supervised algorithms on RL tasks, we experimented with linear, SVM and decision tree learners (regressors). Table ?? summarizes the maximum average reward per 100 episodes that these learners were able to achieve.

Our empirical evidence, though not exhaustive, hints to superiority of neural nets over other learning algorithms for Q function generalization. One reason might be the non-linearity coupled with extreme non-convexity in the Q function landscape. As a result, linear regression may have insufficient flexibility (i.e. high bias) to approximate the Q function. Regression trees are provably **universal function approximators**. In theory they have equal expressive power as neural nets. Yet, in practice it may be difficult to search the hypothesis space efficiently in decision-tree representation. In addition, the hidden layers in neural nets have been interpreted to be representations of latent information in raw features (e.g. "

Conclusion

References

[1] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602 [cs.LG]* 2013.
[2] Richard S. Sutton, *Machine Learning*, **8** p. 9–44, 1988.
[3] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning," *MIT Press*, 1998.