A Survey of Model-Free and Model-Based Reinforcement Learning for the Acrobot

Shubhom Bhattacharya

Department of Electrical & Computer Engineering, Cornell University

I. Introduction

The fields of reinforcement learning and control theory naturally share many concepts. The problem of making optimal decisions about a given environment for an agent with no predefined optimal policy is complex and dependent on the agent's access to information about the environment. Indeed, in many applications, defining optimality itself is nontrivial. In a nondeterministic environment with possibly incomplete information, including about other agents, it is difficult to make direct relations between observed states, actions, and rewards and therefore to find optimal or near-optimal policies for a given task. In the most fundamental sense, algorithms in both reinforcement learning and control theory attempt to characterize a policy that is almost optimal for a given objective.

The reinforcement learning community at large has made significant strides in recent years. Several landmark results that have received popular media coverage include the victory of DeepMind's AlphaGO agent against world Go champion Lee Sedol0 (DeepMind, n.d.)and the success of multi-agent reinforcement learning for the real-time strategy game DotA by OpenAI (OpenAI Five, n.d.). Though the field has been extant for quite some time, these public showcases of reinforcement learning along with evolving techniques in deep learning and modern computational power have reinvigorated research and support for reinforcement learning.

There are two primary families of algorithms in reinforcement learning: model-free and model-based (Sutton & Barto, 1998). Model-free algorithms seek to learn exclusively from experience without any characterization of the underlying system, while model-based algorithms learn from planning upon an environment. It should be noted that the two paradigms are not mutually exclusive and can be integrated in many popularly-utilized algorithms. Also, the term "model" in reinforcement learning literature refers to a model of the system dynamics as opposed to a model of a given policy or value function.

One of the classic continuous control environments used in reinforcement learning is the double-inverted pendulum, commonly known as the Acrobot. A controller for the Acrobot attempts to supply a torque on the pivot connecting the upper and lower pendulum such that some portion of the system is raised above a "ground" line.

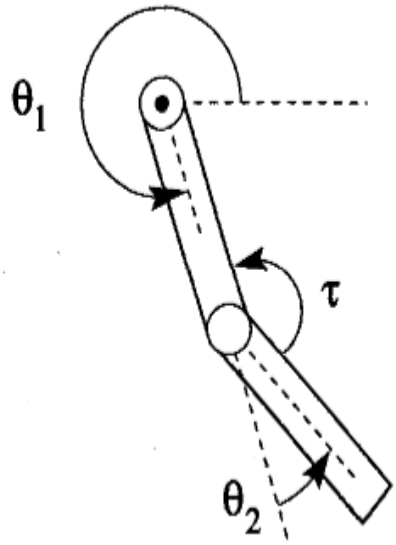


Figure 1 The acrobot swings freely at joint 1 and is actuated on the pivot at joint 2 (Boone, 1997)

The details of the state-dynamics and decision problem encoding of this system is left for Section II. It suffices to say that the Acrobot is an interesting problem in continuous control due to its properties as a deterministic system with a continuous state-space and discrete-action space with well-defined state-transition functions that are a result of the mechanical equations underlying its motion. As a result, Acrobot continues to be a way to quickly test and debug reinforcement learning algorithms that strive to solve continuous-space problems.

This survey selects several model-free and model-based techniques and evaluate their relative strengths and weaknesses on the Acrobot task. While this is not indicative of these algorithms' performance on continuous state-space control problems in general or exhaustive of the techniques available, the ultimate goal is to produce a finding about the importance of learning a model.

II. Models and Algorithms

Acrobot Environment

To have a nuanced understanding of the continuous control problem embodying the Acrobot, a rigorous formulation of the environment and the controller's decision process is necessary.

In this environment, the state observation is encoded as a vector of six scalar numbers as follows:

$$s_t = [\cos(\theta_1), \sin(\theta_1), \cos(\theta_2), \sin(\theta_2), \omega_1, \omega_2]^T$$

, where ω_1, ω_2 are the angular velocities of the first and second joints and θ_1, θ_2 are the angles between the first joint and the ground line and the second joint and the first joint respectively (see Figure 1). The controller's input $\tau_t \in [-1, 0, 1]$ is a scalar number that indicates magnitude and

direction of torque on the second joint. Intuitively, it follows that the input torque τ_t and state s_t as defined can be used to define a state transition function based the mechanics of the system. Some assumptions from mechanics are kept: that the standard gravitational acceleration constant $g = 9.8\, m/s^2$ is used and that all masses can be treated as point masses. This equation is:

$$M(\theta)\frac{d^2\theta}{dt^2} + C\left(\theta, \frac{d\theta}{dt}\right)\frac{d\theta}{dt} + G(\theta) = \begin{bmatrix} 0 \\ \tau \end{bmatrix}$$

Where M is the positive definite inertial tensor, C is a centrifugal/Coriolis force, and G accounts for the gravitational force and $\theta = (\theta_1, \theta_2)$. It is possible to show that M, C, and G are fully characterizable with a given θ, the gravitational acceleration constant g, the masses of the pivot points, and the lengths of the legs (Murray & Hauser, 1991). Crucially, these tensors are dependent on sinusoidal functions of $\theta = (\theta_1, \theta_2)$ and the system mechanics are thus nonlinear in terms of θ.

The reward signal to the controller from the environment is defined as follows: a scalar reward of 0 is observed for a terminal step, meaning some portion of the Acrobot is above the ground line, and a scalar reward of -1 is observed for all other steps. The following inequality holds for a terminal step and thus generates a 0 reward signal if it is true at step t:

$$-cos(\theta_1) - cos(\theta_2 + \theta_1) > 1$$

Policy Evaluation Metrics

If a reinforcement learning agent aims to maximize a reward signal objective, then this reward signal is at most 0 for a given timestep, such that a near-optimal policy qualitatively will achieve the largest aggregated reward over a horizon of T timesteps. Note that in this section, an episodic reward refers to the sum of the rewards from the initial observation to the terminal observation, a series of timesteps that correspond to one episode of training. The magnitude of this episodic reward signal is therefore one important metric for evaluating the algorithms used to find a nearly optimal policy to the Acrobot control problem.

$$episodic\ reward = \sum_{i=0}^{T} r_i$$

If using a closed-loop feedback controller or approximate dynamic programming techniques, another metric to consider is the rate of learning over time: over successive episodes, a desirable policy may show higher aggregate rewards with fewer episodes run while not necessarily attaining as much episodic reward while running more episodes. If some convergence in episodic reward is expected or empirically found, then at a high-level, it may be desirable for a controller to converge to such a value in fewer epochs of training. Hence, we can assess another metric: the aggregate episodic reward over the number of episodes used to train, say N.

$$aggregated\ episodic\ reward = \sum_{k=1}^{N}\sum_{i=0}^{T_k} r_i$$

Model Selection

It is nearly impossible to perform an exhaustive survey of all algorithms used to approach similar reinforcement learning problems. The goal of this project is instead to implement several commonly used reinforcement learning and control theory algorithms for the model-free and model-based policies. Then, the goal is to compare these chosen techniques with other published work for the Acrobot problem, and realize any trends in the metrics listed in the previous section, between the set of model-free and the set of model-based algorithms as well as between model-free algorithms and model-based algorithms. To do justice to previous work and current research trends, both gradient-free and gradient-based algorithms are considered.

Model-Free Algorithms

Experimental Setup

To simulate experiments, the OpenAI Gym package (Brockman, et al., 2016) was used to render the Acrobot and perform training, in a Python 3.6 Anaconda virtual environment. In the OpenAI Gym simulation environment, the link masses (kg) and moments of inertia ($kg \cdot m^2$) are both set to 1.0, while the link lengths are set to 1.0 (m).

For the implementations discussed later in this section, the model-free algorithms tested were built using the Tensorforce library, a Python wrapper for the popular Tensorflow library with modular reinforcement learning agent implementation. Due to time and computational resource constraints, each model was tested with a learning rate of 0.001 and 0.005, with hidden layer sizes of [128,256] and [256,512], and the rectified linear-unit activation. For a full reference of tested parameters, please see the Appendix for code and refer to Tensorforce documentation for additional details. To assess learning, aggregate episodic reward was measured, such that using fewer iterations in an episode corresponded to smaller cost, as defined in the Acrobot decision problem formulation. Therefore, an increasing aggregate episodic reward indicated learning was occurring, because the Acrobot was being pivoted to the terminal state in fewer timesteps. Each algorithm was trained for either 50,000 training iterations or 100 episodes, whichever came first. The reason for limiting the algorithm to 100 episodes was that this was the evaluation criterion for the OpenAI Gym leaderboard.

The algorithms used in implementation were the vanilla policy gradient and proximal policy optimization. The first represented a fundamental policy gradient method that did not rely on any value function approximation, while the latter represented a commonly used "actor-critic" reinforcement learning algorithm. Thus, two large families of reinforcement learning algorithms were represented.

Vanilla Policy-Gradient

A policy gradient algorithm in general approximates the optimal policy by updating a differentiable parametrized policy with parameter θ, learning rate α, and timestep t through gradient ascent on an objective performance measure we seek to maximize.

$$\theta_{t+1} = \theta_t + \alpha \nabla \tilde{J}(\theta_t)$$

The function $\tilde{J}(\theta_t)$ for the generic policy gradient algorithm is a stochastic estimate of the performance metric $J(\theta_t)$ in expectation, i.e. $E[J(\theta_t)] = \tilde{J}(\theta_t)$. The vanilla policy gradient

algorithm (VPG) is also known as the REINFORCE (Williams, 1992) method. In this case, $J^{\sim}(\theta_t)$ is approximated in some sense using Monte-Carlo sampling and observed return (Sutton & Barto, 1998). More specifically,

$$\nabla J(\theta) \approx E_\pi[G_t \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}]$$

where G_t is the observed reward at timestep t, and $\pi(A_t|S_t,\theta)$ is the policy function with the considered action A_t given state S_t and current parameters θ. A qualitative understanding of this objective gradient can be formed by evaluating an arbitrary action A_t with observed reward G_t. Say the current policy π_t is parametrized by θ_t. Then, the objective gradient will be greater if $|G_t|$ is large or $\pi(A_t|S_t,\theta)$ is small. On the other hand, a small gradient results from small $|G_t|$ and large $\pi_t(A_t|S_t,\theta)$. Then, $\frac{\nabla_\theta \pi_t(A_t|S_t,\theta)}{\pi_t(A_t|S_t,\theta)}$ can be thought of as a normalized gradient with respect to the policy so that the largest gradient updates happen for unexplored actions with greatest rewards, and the smallest gradient updates happen for explored actions with lesser rewards, which is an intuitive way to make policy updates.

Experimentally, the VPG agent showed several interesting results. Some episodes were particularly long (over 1000 iterations) and caused the Acrobot to repeatedly move in a periodic swinging motion below the ground line that swept a very small θ_1. This could be interpreted as a policy convergence towards a set of parameters that favored a repeated state transition between two small sets of states, neither of which contained a terminal state, meaning this policy large number of iterations in the episode and therefore a lower reward. For the full implementation results, see the Appendix.

Proximal Policy Optimization

Proximal Policy Optimization (PPO) uses a surrogate objective that is trained on batches of experience (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) to bound the divergence between θ_t and θ_{t+1}, in the sense that the expected difference in the parameters themselves is clipped. This objective is defined as:

$$L^{CLIP}(\theta) = \hat{E}_t[\min((r_t)\widehat{A_t}, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A_t})]$$

where $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\ old}(a_t|s_t)}\widehat{A_t}$ denoting a probability ratio times the estimated advantage $\widehat{A_t}$ at timestep t, and ε is a hyperparameter or clipping the probability ratio so that it is not too large or small. Intuitively, the clipped loss is the minimum of an unclipped ratio (i.e. the parameter update is within the clipping bounds) and the clipped ratio, so that an upper bound on the change in the policy parameter is made with respect to the objective.

The surrogative objective function is added with two terms from earlier work that encourage exploration through an entropy bonus and the value-function objective in an actor-critic framework, which is defined according to the desired actor-critic framework. In other words,, the PPO algorithm adds the clipped loss objective onto the value-function based objectives and entropy bonus used in other actor-critic frameworks.

$$L^{CLIP+VF+S}(\theta) = \hat{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF} + c_2 S[\pi(\theta)|s_t]]$$

The above equation is the PPO algorithm actor-critic objective that is maximized by gradient ascent.

The best achieved performances of both VPG and PPO in the Acrobot problem are not directly available. However, one solution on the OpenAI Gym leaderboard for the Acrobot problem uses PPO with selected hyperparameters achieving a (best case) -72.2 episodic reward, meaning approximately 72 iterations are needed to reach the terminal timestep on average using the PPO-trained policy. Slightly worse best case aggregate episodic reward was seen in the PPO implementation used in this paper.

The PPO implementation on the Acrobot OpenAI Gym environment was done using the same experimental setup as the VPG implementation. For a full list of results see the Appendix.

Non-Gradient Based Methods

The two implemented methods were both gradient-based model-free reinforcement learning algorithms. However, gradient-free methods also exist to approximate a policy for the Acrobot problem. One popular algorithm is tabular Q-learning. In the general formulation of Q-learning, a discrete state space is sampled repeatedly, and the observed rewards are used for a recursive Bellman update of a so-called Q-function, which is the state-action value for all states and actions at all timesteps. It is evident that this requires discretization of the state and action spaces if they are continuous, and that such methods quickly become infeasible as sampling all states for most problems nearly impossible. However, Boone shows that discretized tabular Q-learning can achieve the terminal step after only 126 iterations are used after using 401,593 training steps and that the related SARSA algorithm[1] can achieve the terminal step in 76 iterations. (Boone, 1997). While these tabular algorithms use significantly more training iterations, they display similar rewards after sampling the full space. In other words, infinite sampling of a continuous state space would yield the optimal policy through these tabular methods.

Model-Based Algorithms

Model-based algorithms rely on some prior knowledge of the model dynamics when updating a policy. This knowledge is not necessarily known to the agent before training- instead, it may be that states and actions encountered during training at a given timestep are used to estimate model dynamics to be used in future training steps. The selected model-based algorithms consist of Bayesian Deep Reinforcement Learning, a gradient-based method, and the Linear Quadratic Regulator, a gradient-free method.

Bayesian Deep Reinforcement Learning

Bayesian Deep Reinforcement Learning models parameters as random variables. The resulting Bayesian Neural Networks (BNNs) provide some favorable qualities: uncertainty estimates, robustness to stochastic functions, and reduced overfitting. (Ghosh, Yao, & Doshi-Velez, 2018).

[1] A modification of Q-learning in which a policy uses the Q-value of the (state,action) pair

Variational inference algorithms[2] exist to construct these Bayesian priors for each parameter and update these during training (Blei, n.d.). These priors form probabilistic assumptions on the model dynamics, such that Bayesian reinforcement learning algorithms can be model-based. Hong et. al. showed that Kalman Temporal Difference (KTD) learning coupled with feedback yield learning policies on the Acrobot environment but do not specify the rewards attained (Hong, Jongmin, Kim, Ortega, & Lee, 2018). Model-based Bayesian optimization (MBOA) shows promising results, outperforming Deep Q-learning in a variety of continuous state problems including Acrobot. MBOA empirically converges to greater reward within 50 episodes compared to non-model-based Bayesian optimization (BOA) and DQN (Wilson, Fern, & Tadepalli, 2018)[3].

Linear Quadratic Regulators

Another applicable solution scheme from control theory is the linear quadratic regulator (LQR). In this linear time-invariant (LTI) system, the state transition function is represented as:

$$x_{t+1} = Ax_t + Bu_t + w_t$$

for state x_t, input u_t, and a Gaussian, disturbance w_t at timestep t[4]. Note that the disturbance is assumed to be Gaussian with 0 mean and finite variance, though the original Acrobot system does not add a stochastic perturbation in its state transition so w_t can be assumed to be 0 for the problem. Furthermore, there is no observation noise i.e. the observed x_{t+1} is the actual $x_{t+1}=y_{t+1}$.
Finally, the LQR system minimizes an objective that is defined in terms of the state x_t and input u_t:

$$\int_0^\infty (x^T Q x + u^T R u) dt$$

It is clear that there are 4 parameters that must be known a priori or calculated from the model: A, B, Q, and R[5]. Qualitatively, A and B serve to weight the state and input, respectively, for the linear state transition while Q and R weight the state and input, respectively, for the quadratic cost. Spong shows that while the Acrobot state dynamics are known to be nonlinear, the use of nonlinear feedback with an LQR controller can yield an optimal policy. For example, if the vertical inverted position $\theta_1 = \frac{\pi}{2}, \theta_2 = 0$ is defined as the equilibrium a priori, the LQR control policy successfully swings up the Acrobot and balances it at this equilibrium point with the following A,B, Q, and R:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 12.49 & -12.54 & 0 & 0 \\ -14.49 & 29.36 & 0 & 0 \end{bmatrix}$$

[2] Variational inference algorithms are statistical methods of optimizing an approximation of the posterior using the Kulback-Leibler divergence that serves as an alternative to Monte Carlo sampling methods that estimate a posterior (Blei, n.d.)

[4] The LQR formulation is the almost identical for the discrete-time and continuous time cases except for discrete summation as opposed to continuous integration

[5] Several assumptions are made about the definite-ness and invertibility of these matrices. Furthermore, A and B affect the controllability and observability of the LTI system. Discussion on these topics is excluded here but important in considering the practicality of these methods on a general continuous-control problem.

$$B = [0 \quad 0 \quad -2.98 \quad 5.98]^T$$

$$Q = I^{4x4}$$

$$R = 1$$

Then, the state feedback controller is$u = -Kx$ where

$$K = [-242.52 \quad -96.33 \quad -104.59 \quad -49.05]$$

(Spong, 1994). Using the same equilibrium point for the Acrobot problem, Brown and Passino test another LQR parametrization on different initial positions the matrices and report success in maintaining the inverted equilibrium with different A and B, but report that this is only possible if the starting positions of the robot are not too far from the equilibrium point.

Here, we compared the LQR control task to the Acrobot in a slightly different setting, where reward was based on some prior equilibrium position. Yet, the model-based parameters deciding initial conditions and equilibrium positions and the calculated values of A and B, show that the basic LQR controller may be capable of maintaining a predefined "equilibrium" point at $\theta_1 = \frac{\pi}{2}, \theta_2 = 0$, which was defined as a terminal state in previously considered studies. It is important to note that this terminal state is non-unique: for example, $\theta_1 = \frac{3\pi}{2}, \theta_2 = 0$ is also an equilibrium point. In other words, the "solution" to the problem consists of a set of terminal states that are solutions to the equation $-cos(\theta_1) - cos(\theta_2 + \theta_1) > 1$ as opposed to the one equilibrium point.

III. Results and Discussion

To begin with, the model-free policy gradient methods showed several interesting traits. In general, it seemed that the average reward metric fell into local minima that caused learning to slow down. The VPG with learning rate of 0.001 and hidden layer sizes of 128 and 256 nodes offered the most consistent increase in reward over time. This suggests that the more conservative network architecture and slower learning rate allow for less overfitting of the policy network parameters based upon on a small number of terminal experiences. Further evidence of this trend is visible in a comparison of the policy performances of the [256,512] hidden layer implementations vs. the [128,256] ones, which showed the latter set was less prone to being "stuck" in a nonterminal policy rollout, thereby decreasing average aggregated reward in both the VPG and PPO algorithms. Another interesting point was that the 0.005 learning rate PPO showed smaller terminal episodes than the other models, implying that the variance in the episodic aggregate reward for this PPO model was higher. One caveat of these insights is that these implementations were run only once per model. As a result, repeating the training several times would allow for a better approximated curve of each model's learning, and would be more informative about their relative strengths and weaknesses. The tabular methods give an optimal result given full information about the discretized system dynamics, but yield different results for the Q-learning and SARSA algorithms that are competitive with the observed rewards from PPO and VPG. One drawback to these methods is that they require more training iterations to reach these results, even when PPO and

VPG as implemented are not necessarily optimized by hyperparameters- the reported Q-learning and SARSA results used over 8 times and over 2 times as many training iterations compared to the implemented VPG/PPO.

The model-based policies in general showed a greater tendency to find better solutions to the Acrobot problem. While information about the Acrobot mechanics were used to formulate these models, they displayed promising results. In particular, the Bayesian gradient-based methods empirically showed learning on the Acrobot problem, with rewards that reportedly outperformed the Deep Q-Network, another popular gradient-based deep reinforcement learning algorithm. The Linear Quadratic Regulator was optimal for its task, but required full knowledge of the system a priori as well as a nonlinear feedback that allowed for the LQR state transition to be linear. While the motivation for a nonlinear feedback is clear (i.e. to apply the LQR control), the construction of this nonlinear feedback is vague. As a result, it is not obvious in what instances a nonlinear feedback can be assumed and whether such a feedback can be approximated.

For the Acrobot problem, model-based algorithms based upon the fully-known environment dynamics seem to be more effective, with the reported Bayesian methods outperforming the aggregate rewards of the model-free methods. However, it seems that more complex environment dynamics (possibly with the introduction of perturbations in the state transition function) would render these techniques far less feasible. Model-free algorithms are not directly hindered by this and thus may be a more favorable solution if the environment dynamics are not as well-defined a priori.

IV. Conclusions

Learning Experiences

The first learning experience from this project was increased familiarity with the algorithms surveyed. Thorough examination of published textbooks and papers allowed an appreciation for the intuitions underlying the mathematical formulations used in the algorithms. Between model-free and model-based methods, there is a tradeoff between resources used for tuning and testing hyperparameters and those used in forming/updating a model a priori . Atkeson and Santamaria claimed that modelling the environment is more data efficient, more rewarding, and more robust to changing objectives, but advances since then in gradient-based methods such as those covered above indicate that this is not true and that model-free methods can form nearly-optimal policy parametrizations. On the other hand, convergence is not guaranteed in deep reinforcement learning, and extensive hyperparameter tuning is required in almost all cases.

One of the personal goals of this project for the author was to find some unification of the fields of (deep) reinforcement learning and control theory. While deep reinforcement learning as a field has experienced several landmark results in recent years, it is erroneous to apply gradient-based policy approximation to all problems. The LQR controller's success at maintaining an inverted equilibrium is evidence for this. Also, the success of Bayesian Deep Reinforcement Learning in some continuous tasks presents satisfying bounds on the uncertainty of BNN parameters that are more informative than the so-called "blackbox" optimization of vanilla neural network parametrizations.

Future Work

This survey was not exhaustive and the gradient-based model-free implementations are known to be suboptimal as algorithms with better aggregated rewards exist. Given more time and computational resources, more thorough optimization of the gradient-based methods would be necessary before arriving to a conclusion about their performance, so that the (sub)optimal set of hyperparameters would be used to compare algorithms' performance. For example, training for more timesteps might allow better exploration of the state space. Considering more algorithms and controllers would be ideal as well. A litany of gradient-based model-free reinforcement learning algorithms exist that were not considered that may show much better performance for the Acrobot problem or for continuous-control problems in general. It would be ideal to implement and evaluate a larger sample of these algorithms, as a small number of algorithms cannot be used to characterize the performance of an entire class of algorithms. Similarly, the LQR is one of the most fundamental controller formulations in control theory, and assessing the performance of other paradigms unexplored by this paper for optimal control (such as "fuzzy" controllers) would be a more nuanced treatment of the subject. Finally, another line of inquiry would be to apply and compare the performance of similar algorithms on more complex dynamical systems, both in their optimality (or lack thereof) and their convergence properties. Implementing a Bayesian framework for the Acrobot would also be a useful learning experience and would provide some certificate of reproducibility for the Bayesian methods discussed. One possible case study is the humanoid locomotion problem, which could compare mechanics-based models with model-free algorithms that can serve as controllers in real-world applications such as haptic feedback or prosthetics.

V. Bibliography

Atkeson, C. G., & Santamaria, J. C. (1997). A Comparison of Direct and Model-Based Reinforcement Learning. *IEEE Conference on Robotics and Automation*.

Bertsekas, D. P. (2012). *Dynamic Programming and Optimal Control: Approximate Dynamic Programming, Vol.II.* Athena Scientific. Retrieved 12 12, 2018, from http://www.athenasc.com/dpbook.html

Blei, D. (n.d.). *Variational Inference*. Retrieved 12 12, 2018, from https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf

Boone, G. (1997). *Efficient reinforcement learning: model-based Acrobot control*. Retrieved 12 9, 2018, from http://ieeexplore.ieee.org/document/620043

Brockman, G., Cheung, V., P. L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. Z. (2016). OpenAI Gym. *arXiv*.

Brown, S. C., & Passino, K. M. (1997). Intelligent Control for an Acrobot. *Journal of Intelligent and Robotic Systems*.

DeepMind. (n.d.). *AlphaGO*. Retrieved from deepmind.com/research/alphago

Ghosh, S., Yao, J., & Doshi-Velez, F. (2018). Structured Variational Learning of Bayesian Neural Networks with Horseshoe Priors.

Google achieves AI 'breakthrough' by beating Go champion. (n.d.). Retrieved 12 9, 2018, from https://www.bbc.com/news/technology-35420579

Greg Brockman, V. C. (2016). OpenAI Gym. *arXiv*.

Hong, T., Jongmin, L., Kim, K.-E., Ortega, P. A., & Lee, D. (2018). Bayesian Reinforcement Learning with Behavioral Feedback. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-1*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. A. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv: Learning*. Retrieved 12 9, 2018, from https://arxiv.org/abs/1312.5602

Murray, R. M., & Hauser, J. (1991). A Case Study in Approximate Linearization: The Acrobot Example.

OpenAI Five. (n.d.). Retrieved 12 9, 2018, from https://blog.openai.com/openai-five

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv: Learning*. Retrieved 12 11, 2018, from https://arxiv.org/pdf/1707.06347

Spong, M. W. (1994). *Swing up control of the Acrobot*. Retrieved 12 9, 2018, from http://dblp.uni-trier.de/db/conf/icra/icra1994-3.html

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press. Retrieved 12 9, 2018

Takashima, S. (1991). Control of a Gymnast on a High Bar. *IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS)*.

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*.

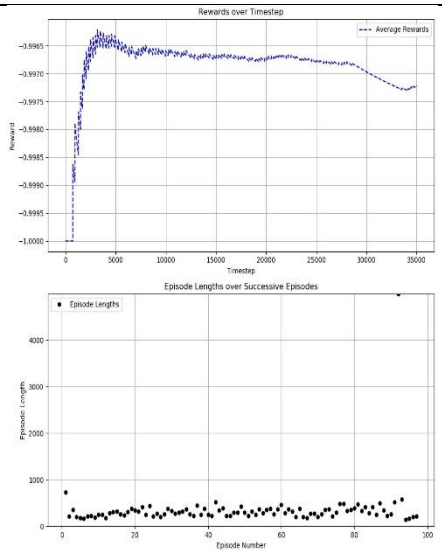
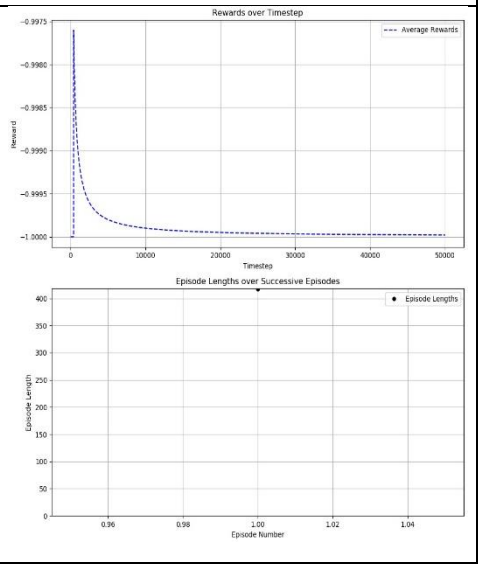
Wilson, A., Fern, A., & Tadepalli, P. (2018). Incorporating Domain Models into Bayesian Optimization for RL.

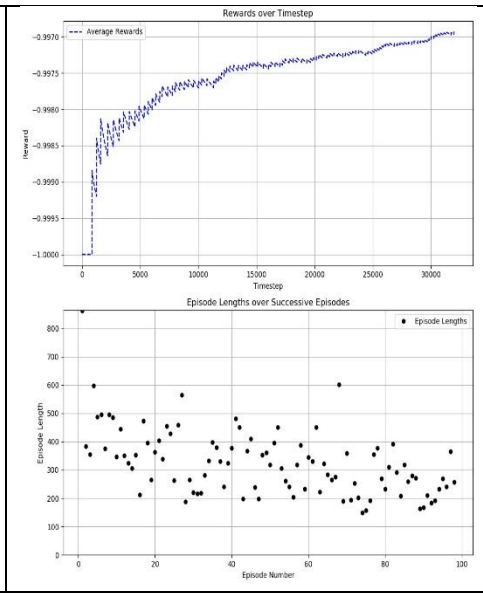
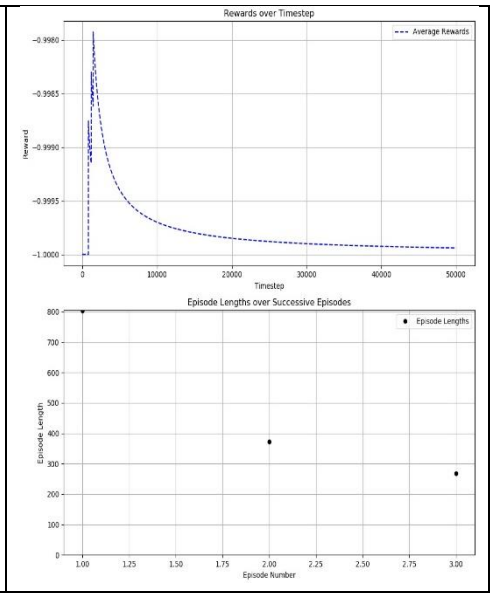
VI. Appendix

OpenAI Gym Acrobot-v1 environment

Model-Free Implementations

VPG

Learning rate α\[hidden1,hidden2]	[128,256]	[256,512]
0.005		

0.001		

For all code that was written for this project, visit https://github.com/shubhomb/acrobot_control_paper.