

Accelerate Reinforcement Learning with Prohibitive Boundaries

Zongqiang Pang, Liping Bai *Member, IEEE*,

Abstract—Prohibitive Boundary is a common concept for athletic training. For instance, gymnast, divers, skaters would wear wires to prevent themselves from falling while acquiring new skills. The same is true for children, as parents caringly provide a helping hand instead of just let them try things out themselves. Reinforcement Learning is an extension of approximate dynamic programming, which reasons backwards and a lookup table is built in the mean time. With the newly added capacity brought about by deep learning, deep reinforcement learning is proven to be a formidable force when it comes to control. However, there is a clear difference between reinforcement learning and human learning. While out joints can have all kinds of combinatorial movement, we only use a few of those even in the most rigorous circumstances. Also, the way we learn to synchronize out joints is not an exhaustive trial and error process. Immitation learning seek to address this issue by provide a model trajectory, and meta-learning see to address this by provide agents with a sense of physics. In this paper, we mimic the prohibitive boundary method we see in athletic training to provide agents additional clues about the environment. There are two advantages in this proposed scheme. First, the prohibitive boundary prevent premature termination of an episode, which is of particular importance to environment where failure is costly. Second, the prohibitive boundary method accelerate data collection on states/action pair that matters. We implemented the proposed prohibitive boundary scheme on various OpenAI Gym environments, and for all the experiments carried out, prohibitive boundary has shown to accelerate training. All the code can be found at:

Index Terms—Reinforcement Learning, Assisted Reinforcement Learning, Reward Engineering

I. INTRODUCTION

REINFORCEMENT learning is the process of methodically extracting information from experiments to gradually bound the policy distribution, maximizing the expected reward along a path. Trajectory Optimization, on the other hand, tries to capture the dynamics of a system with differential or recursive equations, and reshape the phase portrait with actions at each step. When the first batch of reasonable RL results were introduced, they were met with coldness. The preface of Bertsekas's book *Neuro Dynamic Programming* provides a good sample on how reinforcement learning is perceived by the control community back in 1996: "...These methods (Reinforcement Learning) were aiming to provide effective suboptimal solutions to complex problems of planning and sequential decision making under uncertainty, that for a long time were thought to be intractable. Our first impression was that the new methods were ambitious, overly optimistic,

and lacked firm foundation....Three years later, after a lot of study, analysis, and experimentation, we believe that our initial impressions were largely correct. [1]

Things are certainly different today. Reinforcement Learning community routinely generates results that seem unattainable to the control community. What has changed is not the theoretical foundation of RL, which is as leaky as it was in 1996, but its computational infrastructure. DeepMind introduced DQN which enjoyed the success in Atari games, yet their success are not easy to reproduce. There are lot of tricks when it comes to training of the network etc.

Before 2010, the predominant tools for function approximation are kernel methods, where feature spaces are used to transform nonlinear functions into linear space such that regression can be performed. Today, the default function approximator are neural networks. Another major change in computation is that of GPU based acceleration. Before the advent of CUDA, GPU programming requires PhD in computer graphics. Today, anyone who is proficient in C/C++ can programme GPU to parallelize their computation. Software packages such as PyTorch and Tensorflow made this even easier.

jkjkjkjkjkjk

II. REINFORCEMENT LEARNING FOR CONTROL

Control communities use the notation system introduced by Lev Pontryagin. State is denoted \mathcal{X} , Action is denoted \mathcal{U} (the first letter of Russian for "Action"), the dynamics and stochasticity is captured by physical model constraints $x_{t+1} = f(x_t, u_t, e_t)$ where e denote the noise of a system. The objective is usually to minimize the cost function $\mathcal{J}(\cdot)$. Reinforcement Learning communities use the notation system introduced by Richard Bellman who studied dynamic programming. State is denoted \mathcal{S} , Action is denoted \mathcal{A} . The dynamics and stochasticity is captured via transition matrix \mathcal{P} of a Markov Decision Process. The objective of RL is the maximize the reward function $\mathcal{R}(\cdot)$. Yet, if we set the transition matrix to be identical to the noise, then it is clear that the underlying process behind these two notation system are exactly the same. The differences are nothing but style. Since the audience of our paper is the control community, we would cast reinforcement learning in the control notation system.

A. Where to Approximate

The constraints imposed in the Trajectory Optimization formulation manifest themselves directly in policy $\pi(x)$ and resulting in either a narrow band of trajectories or a single

optimal solution. However, adjustments in the reinforcement learning formulation is not the policy per se but its distribution. Eventually, what we hope to achieve via learning is a policy distribution, either through policy gradient method or cost to go method, which would maximize the expected reward of a trajectory.

Reinforcement Learning is not a new subject, control researchers probably know it by the name Approximate Dynamic Programming. Yet the major progress in recent years is the enhanced computation power which brought about the potential of neuronetworks as a to fruition. Most notably the series of wins reinforcement agents such as AlphaGo, AlphaStar forged against the best human players in the respective disciplines.

Broadly speaking, there are three revenues where neuronetwork based approximation can find its way into optimization as shown in 1. One is learning a dynamics model from samle; Second is policy gradient based learning; Third is approximation of cost to go functions such as value funtion and Q function. The details of the implementation would be specified in the subsequent sections.

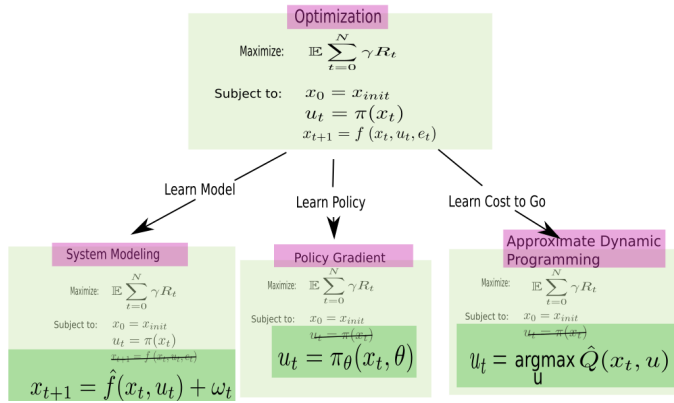


Fig. 1: From Optimization to Learning

Before we dive into the detailed researches in each category, we'd like to introduce additional mathematical tools and measurements that has been proven useful in merging reinforcement learning with optimization. One of the most important change of perspectives when control researchers ventures into the land of reinforcement learning is to formulate optimization as an inference problem [2]. Traditional optimization would translate constraints imposed on the struture into definitive trajectories, subject to disturbance and correction. Yet inference view of optimization is an ever narrowing band od distribution as information trickles in via experiments.

There are two things control researchers should be aware of. One, if you simply apply reinforcement learning as it is written in Richard Sutton's book, it probably won't work for you. Additional statistical learning techniques are required. Two, Statistical learning is booming with ideas at this point. Every newly invented measurement and methods could be pretzeled into a self-sustainly structure if you know how to make such arguments, that does not mean it would be a useful tool for your research.

Here we point out two concepts: Mutual Information and

Baysian NeuroNet, which have proven to be integral for converting optimization into an inference.

The go to text book for reinforcement learning community is that of Richard Sutton [3]. In the system constructed by Richard, the objective of an agent is to maximize discounted reward. This formulation is proven a sound goal in the context of video games. However, for optimizations with a physics underpinning, reward maximization is not adequate for noise and disturbances.

Exploration and Exploitation trade off is something studied exhaustively in the reinforcement learning community. A common strategy is ϵ greedy policy where the value of epsilon decays as the learning progresses. Yet, the decaying rate is a hyperparater that need to be tuned. Is there are more sysmtmatical way to balance the exploration and exploitation trade-off? This is particularly important for control related training since this application has considerable amount of disturbance and noise. If the policy converges too soon, any subsequent disturce can deviate the trajectory to the extend that it is no longer controllable.

The intuition behind maximum entropy reinforcenement learning is the heuristic that when we don't know all the circumstance, we should prioritize options that could give us more choices regardless of how the system dynamics turns out to be. Statitically speaking, this means we should maximize the entropy of a distribution, which measures how uncertain or how broad the distributionb is.

The mathematical measurement used is mutual information [4]. It measures how much information regarding the random variable X is contained in the distribution of random variable Y. A reasonable question to ask is that out of all those measurements which captures "distance" in measurement theory, why a divergence is chosen? Turns out, this choice is made because of its computational convenience, much like exponentials are chosen in integral transform because its nice features.

B. Intuition Behind Neural Network

By now, neural network is widely accepted that as the universal function approximator [5] and the grandfathers of this field such as Andrew Ng and Geoffrey Hinton are also its most proficient proselytizer. Yet not that many materials goes into the intuition behind neural network. Here we provide control researchers with minimal exposure to deep learning the graphical reasonings behind neural network. For people who are interested in tracing the deveopment of this subject, we list the following seminal papers: [?] [?]......

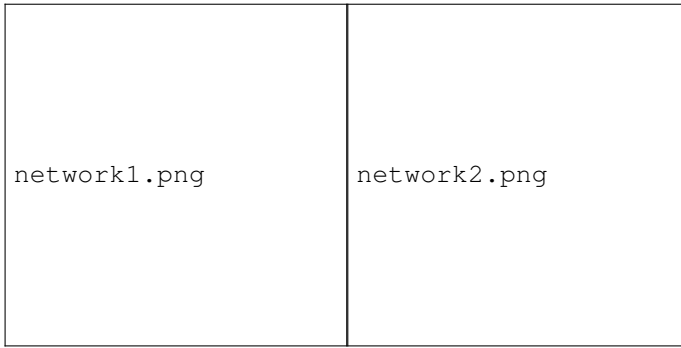


Fig. 2: Graphical Representation of Network Unit

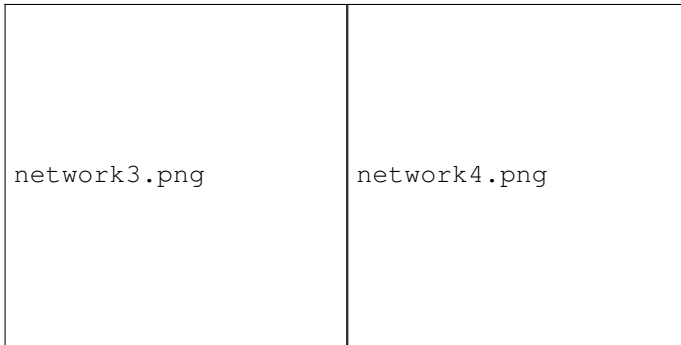


Fig. 3: jljlkljlklj

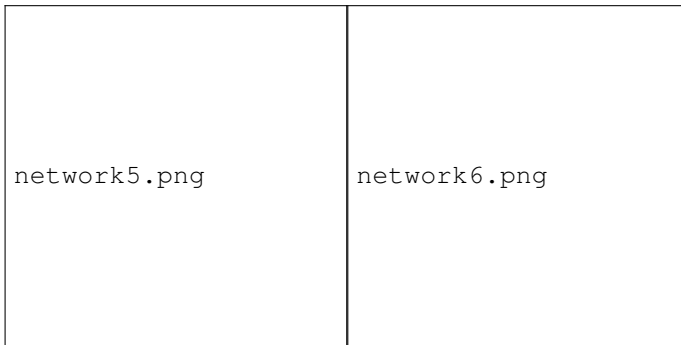


Fig. 4: lkjlkljlklj

A. Pole Angle Boundary at 10 and -10 degree

B. Cart Position at 0.9 and -0.9'

V. WHERE TO APPROXIMATE

VI. SYSTEM MODELLING

VII. POLICY GRADIENT

VIII. APPROXIMATE DYNAMIC PROGRAMMING

Control theorists are familiar with Approximate Dynamic Programming, it

A. Value Iteration and Policy Iteration

B. Maximum Entropy Reinforcement Learning

IX. OTHER WAYS TO UTILIZE INCREASED
COMPUTATIONAL POWER

A. Multiagent Formulation

B. Random Shoot

REFERENCES

- [1] D. Bertsekas and J. Tsitsiklis, "Neuro-dynamic programming," in *Encyclopedia of Machine Learning*, 1996.
- [2] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *ArXiv*, vol. abs/1805.00909, 2018.
- [3] R. Sutton and A. Barto, "Introduction to reinforcement learning," 1998.
- [4] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, pp. 79–86, 1951.
- [5] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

1) Basic Network:

2) Recurrent Network and Variational Inference: Pratially Observable Markov Decision Process (POMDP) is treated with recurrent neural network to contextualize the

III. PROHIBITIVE BOUNDARY

A. for CartPole

PAPRE2 and PAPER3 Should be combined

IV. CARTPOLE-V1 PROHIBITIVE BOUNDARY
IMPLEMENTATION

The CartPole environment observation space consist of four data points: Cart Position along the x axis, Cart Velocity, Pole Angle and Pole Angular Velocity. The terminal state is either cart position greater than 1, smaller than -1 or the pole angle greater than 12 degree or smaller than -12 degree. There are two ways to establish the prohibitive boundaries.