A Control Rearchers' Guide to the Reinforcement Learning Galaxy: A Survey

Zongqiang Pang,Liping Bai *Member, IEEE,*

***Abstract*—This paper is aimed at providing researchers in the field of control a bridge for incorporating deep reinforcement learning into their work. We are not talking about adding visual recognition capacity into end-to-end control loops, but technical details on how to take advantage of the progresses brought about by deep learning and its computational infrastructure. Reinforcement Learning is an inference based "Blackbox" approach while traditional optimization is the "Whitebox" approach where phase portrait of a dynamic system can be manipulated. Clearly, neither is perfect. Some of the techniques developed in the reinforcement community can be useful to control researchers in tackling their problem, yet there are four barriers between these two silos: different notation systems; lack of theoritical proof; obfuscating acronyms; changing "best practices". In this paper, we present the taxotamy of reinforcement learning from the perspective of control and optimization. We tease out the most important concepts and hope to provide control researchers ideas and clues as of how to further their research by riding the wave of reinforcement learning.**

***Index Terms*—Reinforcement Learning, Control Theory, Optimal Control, Optimization**

I. Introduction

REINFORCEMENT learning is the process of methotically extracting information from experiments to gradually bound the policy distribution, maximizing the expected reward along a path. Trajectory Optimization, on the other hand, utilizes myriade of computational techniques to incoperate scripted constraints, which capture the dynamics of a system with differential equations or recurssive equations, to shape the phase portrait. The theoretical foundation of reinforcement learning are statistical learning theory and dynamic programming. When the first batch of reasonable RL results were introduced, they were met with coldness. The preface of Bertseka's book Neuro Dynamic Programming provides a good sample on reinforcement learning is perceived by the control community back then: ...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. [**?**]

Things are certainly different today. Reinforment Learning community routinely generates results that seem untainable to the control community. What has changed is not the theoretical foundation of RL but its computational infrastructure. Before 2010, the predominent 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 compuation. Software packages such as PyTorch and Tensorflow made this even easier. This capacity sometimes brought about unexpected result. For instance, the asymchronous actor critic method(A3C) is not exactly rigirous mathematically speaking, yet with fast parallelized computation, it works emperically.

Reinforcement Leraning face the problem of data efficiency. Their community has been trying to make it more broadly applicable and more efficient: Imitation learning and reverse reinforcemnt learning [1] aims at derive constraints from observed best solution; Sergey Levine utilizes unsupervised learning to build an agent with intuitions of the physical world [2]; Chelsea Finn introduced Meta-Learning [3] to extract overaching invariant structures from similar tasks with different setting. This line of research require firm grasp of statistical learning theory, information theory to design efficient schemes for exploration and minimize sampling collection. The "White Box" approach is based on classical control theories. It has a rich history and enjoyed broad success, yet despite effort from the best control theoriests, some complicated dynamics still elude mathematical scription.

Model of a dynamic system and its control strategies can be learned statistically or they can be written mathematically, and there is no reason that these two perspectives can't merge together. Unfortunately, these two subjects are studied by two communities of researchers who use different notations to discribe the same processes and they publish their research only in journals of their own disciplines. In this paper, we hope to provide a bridge for researchers in the field of control theories who would like to incorporate reinforcement learning into their works. We are not the first ones to take up this challenge. Recently, there are interdiscipline conferences held specifially for the purpose of briding the gaps between these groups of people, for instance the L4DC (Learning for Dynamic Control) conference and Intersections between Control, Learning and Optimization Workshop.

In this paper, we build on the work of Benjamin Recht [4] to formulate reinforcelent learning in the language of optimization. The first barrier facing control researchers is

Nanjing Unversity of Posts and Telecommunications, College of Automation & College of Artificial Intelligence, Nanjing, Jiangsu,210000 China email:zqpang@njupt.edu.cn

consolidating the notation between control and reinforcement learning communities. Control communities use the notation system introduced by Lev Pontryagin. State is denoted $\mathcal{X}$, Action is denoted $\mathcal{U}$, which is the first letter of Russian for "Action", the dynamics and stocasticity 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 funtion $\mathcal{J}(.)$. 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 stochaticity is captured via transition matrix $\mathcal{P}$ of a Markov Decision Process.The objective of RL is the maximize the reward function $\mathcal{R}(.)$. 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.

The second barrier facing the control researchings is the sense of standing on shifting sand instead of firm ground. For instance, while the convergence of dynamic programming method is proven, no guarantee can be found in the approximated version of the same algorithm; The asynchronous actor critic method obviously updates the parameters in a haphazard manner yet it works well empirically; the DQN method replace system function with the a specific form of Q function without theoretical proof, etc. For classically trained control researchers who is used to rigirous mathematical formalism, and they may have a broader set of mathematical tools available to them, which might turns out to be a hinderance. For instance, a control researcher might ask why KL divergence is used instead of all the other measurements the guages how different two fucntions are. Turns out, it is emperically easier to implement, yet you need to go to a senimar to find the reasoning behind it. they may find the lack of theoretical support hard to palade. The idea of reinforcement learning is rather straight forward, what is hard is its implementation details.

The third barrier facing control researchers is the jagons and speed of change. If you try to pick up RL from text book today, most likely you would be overwholed by acronyms such XXXXXXXXX. Some of them might already be proven ineffective implementation of the underlying ideas

This wave of innovation brought about by the reinforcement learning community is not just the implementation techniques, but also the computational infrastructure, notabally the GPU emabled compuating and the PyTorch, Tensorflow software packages. There are classical control algorithms that such as Differential Dynamic Programming can now be powered by those packages. We would also like to list the papers who adapt well known control algorithms for the newly available compuational platforms.

We first provide a taxotamy of reinforcement learning, then we elaborate on the enumerated path in section II and list all the important papers in this area of research. Finally, in section VII, we introduce how the compuational capacity unleashed for reinforcement learning algorithms can be better merged with well-known control algorithms.

II. 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 reinforecement learning formuation is not the policy per se but its distribution. Eventually, what we hope to acheive 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 compuation power which brought about the potential of neuronetworks as a universal approximator [5] 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.

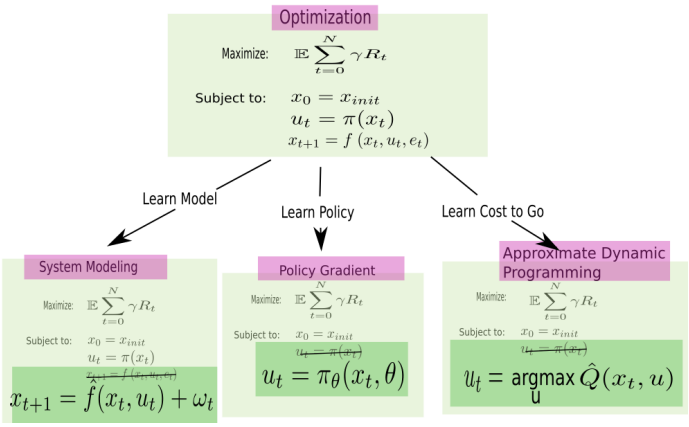


Fig. 1: From Optimization to Learning

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 [6]. Tranditional 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 [7]. In the system contstructed 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 underpining, reward maximization is not adequate since noise and disturbances is common place in control.

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 [8]. 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 meansurement 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.

III. System Modelling

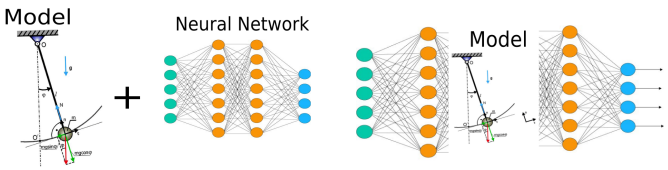
There are two ways for collecting data of the system. One is offline. Just collecting as much trajectories as possible about this system and then build the model later. Second is online data collection where a sample is collected and then incorporated into the model building process right away.

Strictly speaking, offline data collection based model building belongs to the domain of System Identification rather than reinforcement learning. But since it is tangentially related to our paper, we still list it here. After the data is collected and a system model is trained from offline data, an appropriate controllers can be computed based on that model. There are numerous applications in this line of research, supposedly the first successfully implemented non-linear controller based on this method is that of Caltech's Neural Lander. [9]

Model building in the reinforcement context really refers to the only data collection and model refinement process. This model training process doesn't have to start from scratch, althought that is certainly an option. Most likely there already exist a model that partially describes the system. It would be more efficient if we can somehow combine that partial model with a neural network, which would capture the unmodelled dynamics via online training.

ONLINE MIRROR DESCENT. BRYAN BOON'S WORK ON PATH PLANNING AND REINFORCEMENT LERANING

A existing model can be additively combined [10] with a neural network, or it can be embedded into a neural network [11] as shown by Fig. 2. Optimization solver can also be embedded in the neural network as a laywer to encourage faster convergence. [12] [13] [14].



(a) Additive (b) Embedding

Fig. 2: Combine Model and Neural Net

While in theory the aforementioned modelling method should work, in practice it is proven to be far from optimal. [15]. One possible explanation is that any overfitting of the model along the way would be difficult to overcome by subsequent training, resulting in suboptimal performance. One solution is to measure uncertainly of the model with Bayesian Neural Network.

The venena formulation of neuronetwork is one where the weight of each neuron w_i is a single number, which is adjusted based on the backward propogation process. Baysian Neural Network(BNN) is a network where the weigh is subject to a parameterized distribution as shown in Fig. 3

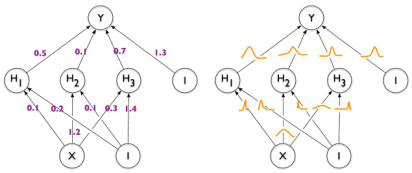


Fig. 3: Neural Network and BNN

BNN is used to measure the uncertainly of the model and better choices can be made when uncertainly of the model is part of the decision making process. [16]

THIS IS THE PART ABOUT BAYSIAN REINFORCEMENT LEARNING

The collected data can be used to train a model which informs which actions should be chosen. It can also be used to train the policy distribution either directly or through policy gradient. The direct policy training method is a theoretical possibility, but due since the number of actions along a trajectory is usually rather large, the valishing gradient problem facing this

methods makes it had to implement. In this paper,we focus our discussion on the Policy Gradient Method.

When control researchers are first exposed to the proof of policy gradient method provided in Richard Sutton's book, they would find it difficult to swallow. The notations are tidious and the logic non-rigirous. A better proof was only recently presented. Please refer to this paper for policy gradient derivation. [17]

IV. Policy Gradient

V. Approximate Dynamic Programming

1. MAX entropy 2. the formulation of cost to go without a system model

VI. Other Ways to Utilize Increased Computational Power

A. Multiagent Formulation

B. Monte Carlo Tree Search

C. Random Shoot

One of the corrilary of the development of reinforcement leraning is improvement in computational infrastructure, which is something the control community can take advantage of. Before this wave of hype in machine learning, GPU enabled computation was a specialty knowledge that is only accessable to large corporations. But now, with CUDA and related software, such compuational power is as easy

Random shoot is the idea if rolling out trajectory at random and wht the fuck is random shoot?

References

[1] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NIPS*, 2016.

[2] C. Finn, I. J. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," *ArXiv*, vol. abs/1605.07157, 2016.

[3] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *ArXiv*, vol. abs/1703.03400, 2017.

[4] B. Recht, "A tour of reinforcement learning: The view from continuous control," *ArXiv*, vol. abs/1806.09460, 2018.

[5] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[6] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *ArXiv*, vol. abs/1805.00909, 2018.

[7] R. Sutton and A. Barto, "Introduction to reinforcement learning," 1998.

[8] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, pp. 79–86, 1951.

[9] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9784–9790, 2019.

[10] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," 2020.

[11] A. Mohan, N. Lubbers, D. Livescu, and M. Chertkov, "Embedding hard physical constraints in neural network coarse-graining of 3d turbulence." *arXiv: Computational Physics*, 2020.

[12] F. de Avila Belbute-Peres, K. Smith, K. R. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," in *NeurIPS*, 2018.

[13] F. de Avila Belbute-Peres, T. D. Economon, and J. Z. Kolter, "Combining differentiable pde solvers and graph neural networks for fluid flow prediction," *ArXiv*, vol. abs/2007.04439, 2020.

[14] A. Agrawal, B. Amos, S. Barratt, S. P. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *ArXiv*, vol. abs/1910.12430, 2019.

[15] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7559–7566.

[16] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *ArXiv*, vol. abs/1505.05424, 2015.

[17] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015.