A Control Rearcher's 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 and optimization a bridge for incorporating deep reinforcement learning into their work. We are not talking about the low hanging fruits such as adding visual recognition capacity into the control loops, but technical details on how to take advantage of the progresses brought about by deep learning and its computational infrastructure. Off the shelf implementations of various reinforcement learning agents written in either PyTorch or Tensorflow can be easily found. Yet, when control researchers venture into this subject, they would find out that there is still quite a lot caveats and pitfalls when it comes to applying neural network to optimization problems. They can't just formulate their optimization problem into reinforcement learning setup and let those off the shelf agents have a go at it. In this paper, we present the taxotamy of reinforcement learning from the perspective of optimization and we tease out the most important concepts while screening away ideas that might be relevent to other field yet less so for control. We hope to provide control researchers roadmap 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 observations to gradually bound the policy distribution, either directly through policy gradient methods or via scaffolding measurements, maximizing the expected reward along a trajectory. This objective is quite similar to that of Trajectory Optimization which utilizes myriade of computational techniques to incorporating scripted constraints, which capture the dynamics of a system with exquisit differential equations, to carve out the optimal trajectories.

Respectively, they are the "Black Box" and "White Box" approach to the problem of control, neither is perfect. The "Black Box" approach requires copious amount of data. Efforts has been made to make it more broadly applicable and more efficient. For instance, imitation learning and reverse reinforcemnt learning [1] aims at automatically 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 structures embedded in similar tasks of different setting. This is a very active area of research, mostly done by computer scientists who are traiend in statistical learning theory. Meanwhile, despite efforts from the best mathematical minds, some complicated dynamics still elude the "White Box" approach.

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

It is obvious that biological entities approaches control from both angles. We have some inkling of how a system might work and how our actions would affect that system and we can learn to control it via trial and error. Yet, unfortunately, these two subjects are studied by two communities of researchers who use different notations to discribe the same processes and they publish their researches only in journals of their own disciplines. In this paper, we hope to provide a bridge for researchers in the field of control 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 task facing control researchers is to consolidate 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 to 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.

We would first introduce where to bring approximation into optimization. We then elaborate on the enumerated path in section II and list all the key papers in this area of research. In section VI, we introduce other methods such as distributed and multiagent formulation of optimization problems to better take advantage of the computational infrastructure for control problems.

II. Where to Approximate

Reinforcement Learning is not a new subject, control researchers probably know it by the name Approximate Dynamic Programming. The renewed interest in this area is the result of progressed made in Deep Learning, which brought the potential of neural network as a universal approximator [5] to fruition. When reinforcement learning is armed with enhanced approximating capacity, we finally get a glimpse into what this line of research can do. AlphaGo, AlphaStar, just name a few.

Broadly speaking, there are three avenues where neural network based approximation can find its way into optimization as shown in 1. One is learning a dynamics model from sample; 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.

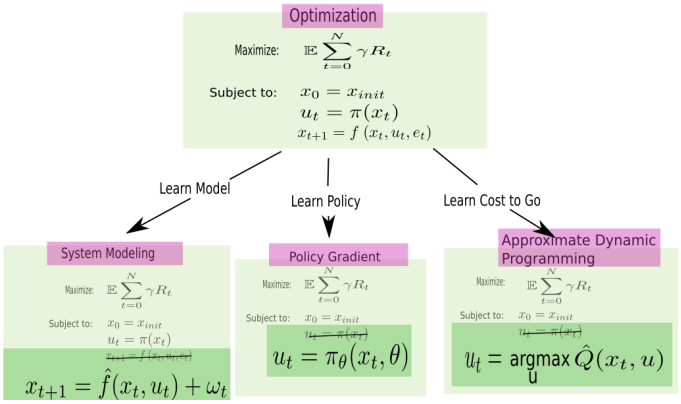


Fig. 1: From Optimization to Learning

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.

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 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 [7]. 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.

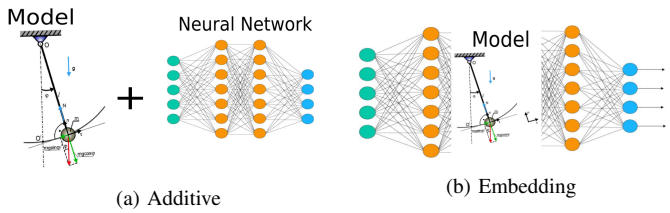
III. System Modelling

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. [8]

The online data collection and training method goes under the name iteration learning based model predictive control (ILMPC).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.

A existing model can be additively combined [9] with a neural network, or it can be embedded into a neural network [10] as shown by Fig. 2. Optimization solver can also be embedded in the neural network as a laywer to encourage faster convergence. [11] [12] [13].



(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. [14]. 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

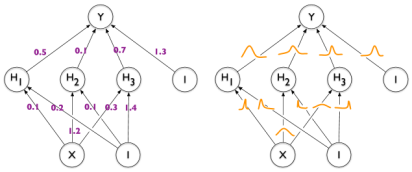


Fig. 3: Neural Network and BNN

BNN is used to measure the uncertainly of the model and avoid overfitting. For exact method, please refer to these papers. [15] [16]

IV. Policy Gradient

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.

The policy is modelled with parameterized distribution, samples collected based on that policy. Based on how well the trajectories pan out, we adjust policy parameters to make the better choices more likely. The classical Policy Gradient based method is called REINFORCE and can be found in this paper: Simple statistical gradient-following algorithms for connectionist reinforcement learning: introduces REINFORCE algorithm. [17]. 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 more rigirous proof was only recently presented, please see this paper for rigirous proof of policy gradient method. [18]

Policy Gradient is one of the core component of reinforcement learning, yet as would refrain from list the important papers for now since most of modern implementation of policy gradient method is used in tanden with methods we are going to introduce in the next section to construct some sort of bootstrap structure for variance reduction.

V. Approximate Dynamic Programming

A. Value Iteration and Policy Iteration

B. Maximum Entropy Reinforcement Learning

The go to text book for reinforcement learning community is that of Richard Sutton [19]. 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.

VI. Other Ways to Utilize Increased Computational Power

A. Multiagent Formulation

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

[8] 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.

[9] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," 2020.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *ArXiv*, vol. abs/1505.05424, 2015.
[16] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *NIPS*, 2017.
[17] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.
[18] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015.
[19] R. Sutton and A. Barto, "Introduction to reinforcement learning," 1998.