reinforcement learning: process control through experience

Gavin Taylor\* and Thomas A. Adams II\*\*

\*United States Naval Academy, Annapolis, MD 21401

\*\*McMaster University, Hamilton, ON, Canada L8S 4L7

Abstract

Significant recent advances in Machine Learning have resulted in an explosion of interest in the field and its possible applications. Reinforcement Learning is a field of ML which attempts to address the control question, including in the control of difficult-to-model, dynamic systems. Reinforcement Learning has undergone particular growth, with several recent high-profile successes. In this paper, I introduce the field of Reinforcement Learning, and describe its strengths and weaknesses to help the reader understand if the approach is appropriate for their control problem. Additionally, I describe ongoing work to develop new chemical process control strategies that incorporate Reinforcement Learning.

Keywords

Reinforcement Learning, Control Theory, Machine Learning, Artificial Intelligence.

Introduction

Optimal control in a production setting is an important part of a profitable and effective system. Once the production line is designed, running it efficiently is essential. Control theory has therefore been a major area of research. Typically, control theory involves carefully modeling the behavior of the system, then solving for the control input which provides the behavior desired. However, minor errors in these models can result in chaotic outcomes, and the numerical solving of the models can be extremely computationally expensive. Additionally, despite this work, older approaches tend to be used in practice, due to simple implementation and the trust that follows decades of testing. While these approaches are safe, they are rarely optimal.

In recent years, Machine Learning has become vastly more effective at an incredible rate. One area of Machine Learning, Reinforcement Learning (RL), is the Machine Learning approach to control. RL is not a current common approach to real-life control, due to the historical difficulty of the problem. However, the recent advances mean RL is newly an option for real-world control problems, with an entirely different set of advantages and disadvantages from traditional control theory. In this paper, we introduce RL and describe these strengths and weaknesses. Additionally, we describe ongoing work in which we apply RL to the control of a simple toy chemical factory problem.

Problem Statement

Imagine a child learning to ride a bicycle. From their senses, the child is confronted with a continued stream of useful information that describes the current state of their bicycle-riding adventure. For example, this data includes the angle of their handlebars, the amount of weight being shifted from one side of the bike to the other, the speed at which the bicycle is being ridden, an observation of the type of surface being ridden on, and any pain in their knee there may be from previous falls and scrapes. After many experiences and experiments, the child has sufficient data to learn to ride, maximizing the positive parts of the experience (euphoria from speed and independence, parental congratulations), while minimizing the negative aspects (falls, scrapes, embarrassment). Most likely, unlike traditional control theory, the child does this with no real understanding of gravity or physics.

Mathematically, we attempt to describe this problem with a Markov Decision Process (MDP). An MDP is a tuple , where is the set of all states the problem might be in (in the case of a bicycle, this could include angle from upright, the angle of the handlebars, the current speed, the road surface, etc.); is the set of all actions the agent can take (turn the handlebars, pedal at some speed, etc.); is a transition kernel describing the probabilities of transitioning between two states when performing an action (, where ); is a reward function (for example, positive numbers for states in which the rider has reached a goal, or negative numbers for states in which the rider’s knee hurts); and is a discount factor, which is used to describe how much we prefer rewards in the short term to rewards in the long term.

The goal of the agent moving through the states of this MDP is to learn behavior (mathematically, a policy ) which maximizes the rewards received, as expressed by the Bellman equation, . The Bellman equation is best thought of as an expectation over the sum of future rewards given the policy, discounted so that short-term rewards are more valuable than long-term rewards. This policy is learned from a number of samples of the form , meaning the agent was in a state, took an action, received some reward, and ended up in another state. A wide variety of approaches of learning effective policies exist, and have been active areas of research for decades.

Progress and Successes

The Bellman equation was introduced in the late 1950s. Progress in the decades since and other reinforcement learning fundamentals are best reviewed in Sutton and Barto (2018). Common experiments included some very basic control problems, including the Inverted Pendulum, in which a rod on a hinge must be kept balanced upright by moving its base left and right, and the Mountain Car, when an underpowered car in a valley must gain momentum by accelerating and reversing.

Very recently, however, neural network theory and GPU hardware have combined to produce some extremely effective agents on much more difficult problems. In 2015, Mnih et al. published a paper in which deep neural networks approximated the Bellman equation accurately enough to produce policies capable of playing many Atari games at a super-human level. In the few years since, deep neural networks have produced agents capable of performing a wide range of difficult planning and control problems, including robotic control, humanoid mobility, and the world’s best players of the games of Go and chess (for example, Shulman et al., 2015 and 2017; Mnih et al., 2016; and Silver et al., 2017 and 2018), all purely from experience and without any human modeling or knowledge.

This research has resulted in a number of easily-deployed algorithms that can learn useful policies on a wide variety of tasks. This opens the door for the application of RL to real-world control problems.

Advantages

RL has several advantages to traditional control theory in many types of problems. The biggest advantage to the RL approach to control is the fact that accurate modeling of the system is unnecessary. For example, in the bicycle-riding task described earlier in this paper, at no point does the scientist have to model any of the forces acting upon the bicycle. This allows for control of difficult-to-understand systems (this is illustrated in related successes in language comprehension and computer vision). Additionally, there is no danger of an insufficiently accurate or detailed model resulting in a poor controller. Finally, if an RL agent can learn an effective controller without the effort of modeling, then human effort can be conserved.

A second advantage of RL is that of providing unforeseen solutions. This is perhaps best illustrated by the newest successes of RL in games such as Go and chess (Silver et al., 2018). These RL agents are trained with no human input, instead allowing an increasingly-adept agent to play against, and learn from, itself. The reward functions are non-zero only for victory states. In this way, the agent learns with no bias from a teacher or conventional wisdom how best to play the game. Winning moves have frequently surprised commentators, as the RL agent was able to find superior, unorthodox solutions to problems that had been analyzed by humans (and human-tuned computer programs) for centuries.

A third advantage of RL is in lifelong learning. As a system runs, the dynamics of the system may change. However, a traditional controller will not seek out new optimal behaviors as the system changes, making its behavior less and less optimal. An RL agent, on the other hand, can continually take in new information, tracking and following the optimal policy as the system changes. In chemical plants, this commonly occurs in the form of process drift.

Disadvantages

Despite its advantages, RL is frequently inappropriate for many kinds of problems. The primary drawback is a need for a very large amount of data. The modern approaches referenced in this paper use hundreds of thousands of training samples generated by simulator. While there are approaches in the field of transfer learning for modifying an agent trained on one domain with plentiful data for another without such data, some large amount of useful data needs to be generated or recorded. Additionally, this data requires a great deal of computing power to process during the learning stage, in the form of many GPUs or special-purpose computing chips.

Second, if the system is easy to model and is well understood, traditional control is likely a superior approach. In RL, the problem is derived assuming that the system is a black-box, and it is difficult to include previous knowledge in making the system easier to learn.

Third, the neural networks achieving the greatest practical results are extremely effective, but are not currently well understood. Theoretical understanding of neural networks is advancing quickly, but lags behind progress in experimental results. If it is important to understand why a control agent is making the choices it is, other approaches provide this understanding easier.

This lack of understanding also introduces possible cybersecurity flaws. Adversarial attacks (for example, Goodfellow et al. 2014, and Kurakin et al. 2016) and poisoning attacks (Shafahi et al. 2018) are being explored in other applications of neural networks, and it is likely neural networks for RL are similarly vulnerable. Research needs to advance before it is clear how big of a problem these attacks are.

Ongoing Work

We are interested in demonstrating the capabilities of RL while acknowledging realities of industry. In this section, we describe ongoing work, and the modifications to typical RL we are making to fit the constraints of an applied field.

We have chosen a common chemical engineering 'toy process' to demonstrate the concept. The process consists of a liquid phase CSTR with a single feed stream, a single product stream, and a steam jacket. A hypothetical endothermic A→B reaction takes place in the presence of a diluent C in the reactor, where A and B are both dilute. Two independent state variables are tracked: the temperature of the reaction mixture and the concentration of A in the reactor (with the usual well-mixed assumption). The system has two inputs which are manipulated by the controller, the heat duty of the steam jacket and concentration of A in the feed to the reactor. Assumed constants include reaction rate law parameters, volumetric flow rates of the feed and product streams, heat capacity, density, and other factors. The resulting model of this system consists of two nonlinear ODEs, two state variables, and two inputs. The model equations and parameters are almost identical to those used in other studies such as Mahmood and Mhaskar (2008) and Homer and Mhaskar (2017), except that in the cited studies, the heat duty of the steam is allowed to be negative (i.e. supporting instantaneous switching between heating and cooling), whereas in the present work the duty is restricted to heating only (non-negative values). This toy problem was chosen because it has sufficiently interesting non-linearity and complexity for control purposes such that there are many possible control solutions for any given problem, but is sufficiently small such that it can be studied rapidly. The toy process has been very well studied such that it has known stability and null-controllable regions (Mahmood and Mhaskar, 2008; Homer and Mhaskar, 2017).

Traditional RL requires the agent begin with no knowledge of the system, and a random, extremely ineffective policy, in order to learn its way to optimal behavior. However, this is infeasible when working out of simulation, learning on a real factory environment. This provides us with two tracks to explore.

In the first, we assume we do not have a simulator, but only a real-life system, and so the penalty for poor agent performance is unacceptably large. In this case, we begin by using a PID controller or linear-MPC for control of the system, and train an RL network as a supervised learning problem to output a policy mimicking the “safe” controller for any state within the stability region to the desired setpoint; this is easily done. The RL agent can then be swapped in for the more traditional controller without significant change. From there, the RL agent can be trained by collecting samples by changing the policy in small, safe ways, in order to iteratively improve without endangering the system. This can be done with learning algorithms such as Trust Region Policy Optimization (Shulman, et al., 2015) and Proximal Policy Optimization (Shulman, et al., 2017), which are highly effective and designed to never explore outside a trust region around policies that are known to be effective. We hope this will result in continual improvement from existing control algorithms without risking the system.

In the second track, we assume we do have a simulator, and will therefore need to confront a process model mismatch. This is being addressed in RL with approaches from transfer learning, where policies learned on one problem with ample data are used as starting points for similar problems with less data (for example, Taylor & Stone, 2009; Zhang et al., 2018). In this scenario, we can train an RL agent from scratch, allowing many, varied failures. Once the policy is developed, it can be applied to the physical factory, and further trained to recover from the differences between the simulator and the physical system.

The differences between the two tracks will lend themselves to some interesting comparisons. As with our first track, Google’s first RL-based Go champion was first trained to mimic expert players, and then improves itself through self-play. This approach then beat the world human champion to prove itself the best player in the world. The second iteration, however, learned from scratch, without first imitating human orthodoxy; this player was far superior to the first iteration. Similarly, we expect that an RL agent trained from scratch on a simulator will be superior to one that first mimics another control algorithm.

The second interesting comparison will be in handling the process model mismatch. There are very few theoretical results in generalizing RL agents from one domain to another; it may be that an RL agent will be more effective on the training environment than another control algorithm, but less effective when transferred to the physical system (until it can receive sufficient additional training on the physical system). However, this is unknown; modern neural networks often surprise with their generalizability.

We intend to compare our RL-based controllers with the existing state-of-the-art, such as nonlinear MPC, along a number of axes. First, we hope the new controller is better at the task itself. Second, we hope the controller is faster to compute a control action. Third, we hope this controller will be simpler to implement and require less specialized human training.

Conclusions

Reinforcement Learning has reached a tipping point, and it is now feasible to apply it to real-world problems with the requisite data. With such quickly-advancing capabilities, undoubtedly surprises will be uncovered when real-world constraints are considered. These surprises, and the possibility of superior control, merit research and attention.

Acknowledgments

Gavin Taylor is supported by the Office of Naval Research and the DOD High Performance Computing Modernization Program.

References

Goodfellow, I. et al. “Explaining and Harnessing Adversarial Examples.” *arXiv preprint arXiv:1412.6572*. (2014).

Homer, T., & Mhaskar, P. (2017). “Constrained control Lyapunov function-based control of nonlinear systems.” Systems & Control Letters, 110, 55-61.

Kurakin, A., et al. “Adversarial examples in the physical world.” *arXiv preprint arXiv:1607.02533*. (2016).

Mahmood, M., & Mhaskar, P. (2008). “Enhanced stability regions for model predictive control of nonlinear process systems.” AIChE journal, 54(6), 1487-1498.

Mnih, V., et al. “Human-level control through deep reinforcement learning.” *Nature* 518.7540 (2015): 529.

Mnih, V., et al. “Asynchronous methods for deep reinforcement learning.” *International conference on machine learning*. 2016.

Shafahi, A. et al. “Poison frogs! Targeted clean-label poisoning attacks on neural networks.” *Neural Information Processing Systems*. 2018.

Shulman, J. et al. “Trust region policy optimization.” *International conference on machine learning*. 2015.

Shulman, J. et al. “Proximal policy optimization algorithms.” *arXiv preprint arXiv:1707.06347* (2017).

Silver, D., et al. “Mastering the game of Go without human knowledge.” *Nature* 550.7676 (2017): 354.

Silver, D., et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.” *Science* 362.6419 (2018): 1140-1144.

Sutton, R. S., Barto, A. G. (2018). Reinforcement learning, An introduction. *MIT Press*.

Taylor, M. E., & Stone, P. (2009). “Transfer learning for reinforcement learning domains: A survey.” Journal of Machine Learning Research, 10(Jul), 1633-1685.

Zhang, A., Satija, H., & Pineau, J. (2018). “Decoupling dynamics and reward for transfer learning.” *arXiv preprint arXiv:1804.10689*.