Reinforcement Learning with X-Plane

MIDN 1/C Matthew Yates and Asst Prof Gavin Taylor

Abstract

Reinforcement learning has become a victim of its own success. Problems that used to be considered difficult and interesting have become no longer challenging enough for modern techniques. With some of these domains, it is no longer easy to distinguish whether one RL approach is better than another. To remedy this, new domains to evaluate machine learning algorithms need to be created. Our X-Plane Machine Learning plug-in offers a solution with an easily approached interface to a challenging domain with realistic physics and noise.

Introduction

Reinforcement Learning is one way to pursue autonomy, by leveraging past observed experiences by the agent to learn a useful policy that accomplishes the given task. Due to the increased interest in the use of Machine Learning and Data Mining in general and autonomy in particular, RL is currently an area of extensive research. It is important that this research have challenging domains to explore, with realistic physics and noise models, so that the work can be easily applied to real physical systems. We seek to provide such a domain.

In Reinforcement Learning, the domain is modeled as a Markov Decision Process (MDP). We define an MDP M as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the measurable, possibly infinite set of states, and $\mathcal{A}$ is the finite set of actions. $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is the transition function, where $P(s'|s, a)$ represents the probability of transition to state s' from state s, given action a. The function $R : \mathcal{S} \mapsto \Re$ is the reward function, and γ is the discount factor.

We are particularly interested in assisting work finding a value function V that maps each state $s \in \mathcal{S}$ to the expected total γ-discounted reward for the process. Value functions can be useful in creating or analyzing a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ such that for all $s \in \mathcal{S}$, $\sum_{a \in \mathcal{A}} \pi(s, a) = 1$. The value function of the optimal policy (notated V^*) is defined by the Bellman equation:

$$V^*(s) = R(s) + max_a \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^*(s').$$

In the creation of policies, it can also be helpful to solve for a Q-function of state-action pairs:

$$Q^*(s, a) = R(s) + max_{a'} \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) Q^*(s', a').$$

In large or continuous domains, V^* and Q^* can only be approximated. Value function approximation is therefore a robust field of study, needing domains to try new approaches on. We offer an easily-implemented domain which is challenging and realistic.

Problem

All learning algorithms need some method of being evaluated, to determine whether they

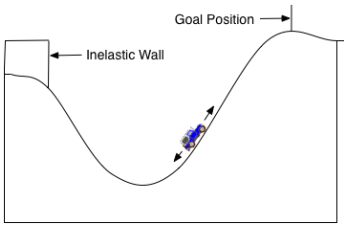

Figure 1: Mountain Car

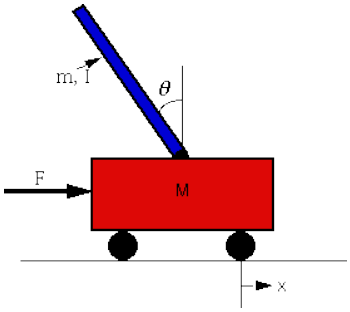

Figure 2: Inverted Pendulum

are better than others at solving problems. Learning algorithms that produce classifiers can evaluated by using the produced classifiers to label a prelabeled data set. Then the results form the learned classifier can be compared to a correct set of labels. The evaluation of a reinforcement learning trained agent requires more than a mere set of labeled data. Traditionally a reinforcement learning algorithm is evaluated in how the produced policy performs in solving a task that requires a series of effective actions be taken to reach the goal.

These domains need to be problems that require that the correct decision be made at several points in order to reach the goal state. The domain also needs to allow for recording of visited states. Examples of these domains are the mountain car problem (Sutton and Barto 1998), the inverted pendulum problem (Wang, Tanaka, and Griffin 1996), and the bicycle problem (Randløv and Alstrøm 1998).

- *Mountain Car* The mountain car problem is about moving a car up a hill when it is not powerful enough to climb up on its own. The car starts in a valley in between two hills. The car has the ability to add a small acceleration in either direction. In order to get over the hill, the car has to rock back and forth between the two hills. Successful agents learn to move in whatever direction the car is already moving. This problem is illustrated in Figure 1.
- *Inverted Pendulum* The inverted pendulum problem is about balancing a pendulum upright that is attached to a cart. This problem is illustrated in Figure 2 (Luntz 1997). For every step the cart can be moved back and forth. A successful agent learns to move the car so that the pendulum remains upright.
- *Bicycle* The bicycle problem is about moving a bike that start at a position A and is trying to reach a goal position B. When the bicycle starts at a right angle to the most direct path to the goal. At every step the agent can move one of five things: lean left, lean right, turn handlebars left, turn handlebars right, or do nothing. The agent learns the proper order of these actions to reach the goal state

The problem with these approaches to evaluation is that they have not become more difficult as machine learning algorithms improved. The fundamental difficulty trying to solve the inverted pendulum has not changed in over 10 years. These problems are not solvable by all approaches, so they offer a means of separating the poor approaches form the good. They however do not offer an ability to separate the truly great and innovative learning algorithms from those that are merely good because older RL algorithms did too well against these problems. Newer and better approaches to RL do not have enough room to improve upon the success of the previous generation of algorithms to show their merits.

In the next section, we propose an alternative domain that addresses this problem.

X-Plane Domain

As a new domain we offer the X-Plane Flight Simulator. X-Plane offers a C++ plugin

API. To use X-Plane as a domain we recreated a simpler API designed for Reinforcement Learning that has built in functions for reading weights, controlling movements, and sampling data. The simulator has a free to use demo version.

X-Plane very closely models the real world in their simulation of flight mechanics and weather. It uses real airports, runways, and aircraft in simulation. 3813 "datarefs" can be read and set while X-Plane is running. With these datarefs every single control surface's position and and every value that the plane has a sensor for can be read. The X-Plane domain can offer many new interesting and meaningful problems for researchers to solve. Piloting aircraft involve many complicated tasks that would be a good test for any algorithm. Further, whenever a RL algorithm can solve one problem in X-Plane a researcher can always change the plane the simulator is using.

Using the datarefs through our X-Plane plugin one is able to add or remove noise from the simulator as weather and wind are settable conditions. In addition to weather, the plugin can simulate unresponsive controls, violent updrafts, mechanical failures, and control the flight of the airplane by setting the deflection of the control surfaces.

This means that researchers will be faced with robust noise, the ability to define descriptive states, and a large potential set of actions. It also means that if in the course of improving an algorithm a researcher stumbles upon a very useful optimization tuned to specifically to the X-Plane domain, then that researcher has likely made an discovery with real world application. This could mean that the work that researchers in RL can very quickly lead real benefits for mankind. Although the X-Plane domain is still an abstraction from the real world, it is an important step toward moving autonomy from research to application.

Proof of Concept

To demonstrate the merits of using X-Plane as a domain for evaluation of Reinforcement Learning we created an agent trained with Least-Squares Policy Iteration (LSPI) (Lagoudakis and Parr 2003) to take off a 747. Actions were made every second. We define $\mathcal{S}$ as the following:

- Elapsed time since the beginning of the start of the flight
- Altitude of the plane
- Speed over ground
- Wind speed
- Wind direction
- Position of the elevators

We define $\mathcal{A}$ as the following:

- Elevators down completely
- Elevators down halfway
- Elevators at a neutral position
- Elevators up halfway
- Elevators up completely

For each sample, the current state was noted, a random action was selected, and the state after one second was noted. Starting from a stopped plane on the runway, this was performed until the plane crashed. A set of samples consisted of a number of these trials.

To gather samples of state transitions X-Plane was put into an infinite loop of start, make random decisions about the elevator flaps, and then crash. X-Plane automatically restarts the flight and sends a message to the plugin process that can be read whenever the plane crashes. Each of those occurrences are logged and the plane starts again gathering more data. The following is a link to the gathering of X-Plane data through random sampling (http://www.youtube.com/watch?v=fxVJLdErrKk)

After the training phase was complete there were 32 runs each with around 100 samples. Then using LSPI with a discount of .9 a series of weights were created. Using these weights a the plugin was able to takeoff. The following is a link to the trained Autopilot taking off(http://www.youtube.com/watch?v=lyQq6OUwmgE).

Conclusion

Reinforcement learning needed more difficult domains to challenge modern techniques. We

created a new more challenging domain using a modified X-Plane flight simulator that can be used to properly evaluate RL algorithms. To prepare X-Plane for use as a domain for testing Machine learning approaches we create a plugin that collected data, set up flights, and input commands without a person in the loop.

After creating the plugin, in order to demonstrate the effectiveness of using the X-Plane flight simulator, we created an agent that was trained to takeoff a 747 on its own. The LSPI trained auto pilot was able to take off. The weights from LSPI make the plane move the elevators to the half down position at the beginning of the runway. Then halfway through the runway the elevators move to the neutral position. As it approaches the end the of the runway the elevators are set at half up. The plane is able to gracefully take off at the end of the runway and slowly climbs into the upper atmosphere.

X-Plane is a usable domain for reinforcement learning and will be in the future useful to researchers to give their algorithms environments with extensive noise and large action spaces. This domain also moves RL research closer to the development of real world autonomy.

Future Work

Work needs to be done on solving multiple aspects of controlling the plane as it takes off. Future work will involve learning how to train one actuator at a time in some form of iterative learning. The difficulty in training to learn these codependent controls is that poor decisions made by the controller for pitch will make smart decisions by the a roll agent appear to lead to failure and vice versa.

More work will be put into making the plugins easier to adjust, by reading parameters from an XML file as opposed to making changes in code that needs to be recompiled. In trying to make the plugin easier to use, work will be done in trying to suppress to graphics and make the simulator run faster so more data can be gathered in the same time.

References

[Lagoudakis and Parr 2003] Lagoudakis, M. G., and Parr, R. 2003. Least-Squares Policy Iteration. *The Journal of Machine Learning Research*.

[Luntz 1997] Luntz, J. 1997. Modeling an inverted pendulum.

[Randløv and Alstrøm 1998] Randløv, J., and Alstrøm, P. 1998. Learning to Drive a Bicycle using Reinforcement Learning and Shaping. In *Proceedings of the 15th International Conference on Machine Learning*, 463–471.

[Sutton and Barto 1998] Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: an Introduction*. MIT Press.

[Wang, Tanaka, and Griffin 1996] Wang, H. O.; Tanaka, K.; and Griffin, M. F. 1996. An Approach to Fuzzy Control of Nonlinear Systems: Stability and Design Issues. *IEEE Transactions on Fuzzy Systems* 4(1):14–23.