Flying Controlled Trajectories using Linear Approximation Reinforcement Learning

Evana Gizzi[1] and David Zabner[1]

Tufts University, Medford MA 02155, USA

1 Introduction

The reinforcement learning literature focuses on building radically simple code controlling impressively complex systems. Throughout the semester, we have developed light weight systems in a pedagogical settings, but have not had the opportunity to implement these systems in a real life application. This was something that we wanted to explore with our class project, to see how the complexity issue takes form with real like applications. Thus, in this project, we explored the ability of a basic reinforcement learning (RL) agent to get an aircraft to fly in a stable configuration, as defined in our text (see section 2.4). Specifically, we aim to see how easy it is to use RL methods to get the aircraft to fly flat. Furthermore, we want to see how feature engineering effects the performance of the agent.

A variety of work has already been done showing that it is possible, through a variety of different RL methods, to safely pilot a plane through take-off, landing, cruise and navigational aspects [4] [3] [1] [2]. However, no-one we have found has given the RL agent control over all aspects of flight including trajectory management based on environmental information including data about other aircraft in the airfield. This is where our work will be novel and, hopefully, show results indicating that RL can be used to increase flight safety. It is also worth noting that our work will be novel in the sense that the majority of past work we found on RL and flight used apprenticeship learning and "expert instructors". Our lack of access to "experts" will present an interesting, important, and novel challenge in the RL-Flight literature. As such, we have the following project aims:

1. Can we control an aircraft with high level commands?
2. Can we use high level commands to fly the aircraft in a stable configuration?
3. Can we use feature engineering to increase our agents ability to successfully accomplish aim 1 and aim 2?

We decided to begin approaching this problem with the simplest algorithm that might solve it and therefore used Linear-Approximation and Q learning as the basis of our approach. This too, surprisingly, appears novel in the field of aviation RL. All of the previous work we found in the field used Neural Networks for their Q-approximation.

2 Problem Formulation

In the following, we describe our application domain and the tools available to us for this research effort.

2.1 Domain

In this research, we explore the general (commercial, private and military) aviation domain. Specifically, we use a scenario in which it is the goal of an intelligent system (built from our RL agent) to fly the aircraft in a simple stable configuration. Our work is broken up into two stages, where the first stage aims to show that we were able to control the aircraft using high level commands, and the second stage aims to show that we were able to use the control established in stage 1 to successfully fly the aircraft in a stable configuration using RL methods. In classic control systems stable flight is accomplished through the use of a carefully tuned PID, a system that uses Position, an Integral and a Derivative to dampen controls to reach a goal state. Writing and tuning PID control systems is, historically, where a large amount of the time is spent when designing vehicle control systems. Much work has been done in replacing

2.2 Scenario

For all of trials, the RL agent flew a Cirrus Vision SF50 aircraft, which is a light weight, 7 seat aircraft. The learning happened at the Palo Alto Airfield KPAO.

2.3 Tools

We will used the X-plane simulation environment, which has been used by many researchers in the past [2] [4] [3] [1] (version 11). The reason why we chose this environment is because it is a simulation that has been very widely used, and is very widely know. On the website for XPlane, it is noted that the environment is realistic and robust enough to be considered viable for flight hours to count toward pilot licenses (which certain contingencies). Although we chose this environment, we built our system in a way that is agnostic of the simulation environment, having consideration for future work. In our future work section, we describe an extension to a system which runs on a rapid proto-typing simulation environment located at NASA Langley Research Center in Hampton VA (VISTAS simulator).

2.4 Theoretical Problem Formulation

In order to implement our system, it was important that we develop a domain specific formulation of the RL problem, that maps to the fundamental RL structures presented in the Sutton and Barto textbook. As such, we present the following formulations:

State Space: We define our state space S as a set of states $S_1, S_2, \ldots S_n \in S$ as a collection of values corresponding to the following: $\{X, Y, Z, vX, vY, vZ, airspeed, theta, phi, psi, P, Q, R, groundSpeed\}$, all of which are described in the chart below.

field	field description
X	latitude
Y	longitude
Z	altitude
vX	velocity vector along x-axis
vY	velocity vector along y-axis
vZ	velocity vector along z-axis
airspeed	true airspeed of the aircraft
theta	pitch
phi	roll
psi	heading (or yaw)
P	the roll rotation rates (relative to the flight)
Q	the pitch rotation rates (relative to the flight)
R	the yaw rotation rates (relative to the flight)
groundSpeed	speed of aircraft relative to the ground

Action Space: We defined our action space as a set of actions in 4 dimensions based on the actual controls available to a pilot. Specifically, each action is some combination of two stick inputs {Latitude, Longitude}, a throttle input as a percentage of max throttle and a rudder input.

Reward Function: The agent will receive a small negative reward for each time-step and a large negative reward for "bad-orientations" and for going over a g-limit and a large positive reward for matching, to within some delta, the goal orientation and speed which will also end the episode. We don't know a lot about what a bad orientation really is so we will just assume that our plane should never exceed a 60° angle in pitch or roll.

Goals: The overall goal of the agent is to gain full high level control of the aircraft. Our goal for this initial work was to get the plane to fly flat. This goal has led us to craft *stability* criteria, which uses the values coming in from the cockpit data stream to check to see if the aircraft is flying flat. As such, we examine 14 data fields of input from the simulator, crafting the following stability criteria (which only employs 2 of the fields).

condition	reward	description
$\|pitch\| < 4 \wedge \|roll\| < 4$	4	positive reward if the aircraft pitches less than 4 degrees and rolls less than 4 degrees
$\|pitch\| < 8 \wedge \|roll\| < 8$	2	positive reward if the aircraft pitches less than 8 degrees and rolls less than 8 degrees
$\|pitch\| < 10 \wedge \|roll\| < 10$	1	positive reward if the aircraft pitches less than 10 degrees and rolls less than 10 degrees
$\|pitch\| > 10$	-1	negative reward if the aircraft pitches more than 10 degrees off the target pitch of 0 degrees
$\|roll\| > 10$	-1	negative reward if the aircraft rolls more than 10 degrees off the target roll of 0 degrees
$\|pitch\| > 15$	-1	negative reward if the aircraft pitches more than 15 degrees off the target pitch of 0 degrees
$\|roll\| > 15$	-1	negative reward if the aircraft rolls more than 15 degrees off the target roll of 0 degrees
$\|pitch\| > 30$	-1	negative reward if the aircraft pitches more than 30 degrees off the target pitch of 0 degrees
$\|roll\| > 30$	-1	negative reward if the aircraft rolls more than 30 degrees off the target roll of 0 degrees
$\|pitch\| > 40$	-2	negative reward if the aircraft pitches more than 40 degrees off the target pitch of 0 degrees
$\|roll\| > 40$	-2	negative reward if the aircraft rolls more than 40 degrees off the target roll of 0 degrees

2.5 Learning Method & Feature Engineering

We implemented a simple Linear Approximation Q-Learning agent with Eligibility Traces as described in "Reinforcement Learning: An Introduction" [5]. Additionally, all raw state information was normalized to be in the range of ± 1. Finally, in some tests we used extra "engineered" features arrived at by multiplying together state and action information (i.e. roll X latitudinal stick) in an attempt to allow for better control.

3 Research Questions

The main research question that we want to tackle is understanding how RL problems take form in real life applications, separate from the pedagogical applications that are often presented in text books and as a part of classwork. Specifically within our domain, we chose to evaluate this question through an

exploratory task which involves flying an aircraft in a flight simulation environment. Thus, we describe our research motivation in the context of our 3 project aims/research questions.

1. Can we control an aircraft with high level commands?

The first goal is to show that using reinforcement learning, we can develop an agent capable of carrying out specific high level commands in relation to changing the orientation and speed of the aircraft. These commands will consist of combinations of changes in the pitch, roll, and yaw of the aircraft as well as the airspeed. These commands will then serve as a basis for the next to sub-goals. All of the actions taken by this agent will keep the plane in a safe (mostly upright) orientation, hopefully, be g-limited for "customer comfort" although the g-limiting is very much a secondary goal. The input to this agent will consist of information about the orientation of the planes control surfaces, the current pitch, roll, yaw, airspeed, and a g-force vector along with the distance from the goal pitch, yaw, roll, and airspeed. The agents action space will be either to figure out if we can give the planes "directional/joystick" controls or if we need to independently control the throttle and each of the control surfaces.

2. Can we use high level commands to fly the aircraft in a stable configuration?

The second goal is to use the controls established in part 1 to fly the aircraft on a fly trajectory. The only values that we will be concerning ourselves with in engineering toward this goal is the pitch and roll of the aircraft. The heading will not be restricted to a certain direction. Likewise, we will not consider aircraft altitude or airspeed in this goal. It is our hope that in future work, we may fix altitude and observe how output would change with this added restriction.

3. How can simple feature engineering improve our agents ability to accomplish our previous goals?

The goal here is to gain an understanding of both the complexity and types of features required to get a simple, linear approximation agent, to exert correct control in a complex space. Related to this we will be asking whether multiple similarly engineered features will help overall performance or not.

4 Results and Discussion

We evaluate the performance of our system overall, along with performance bench marking on a more granular level. We first explored the ideal alpha to maximize the learning for our agent. The results indicated that the ideal value was 0.01 as can be seen below in figure 1. Similar work was done to choose ideal values for epsilon and lambda.

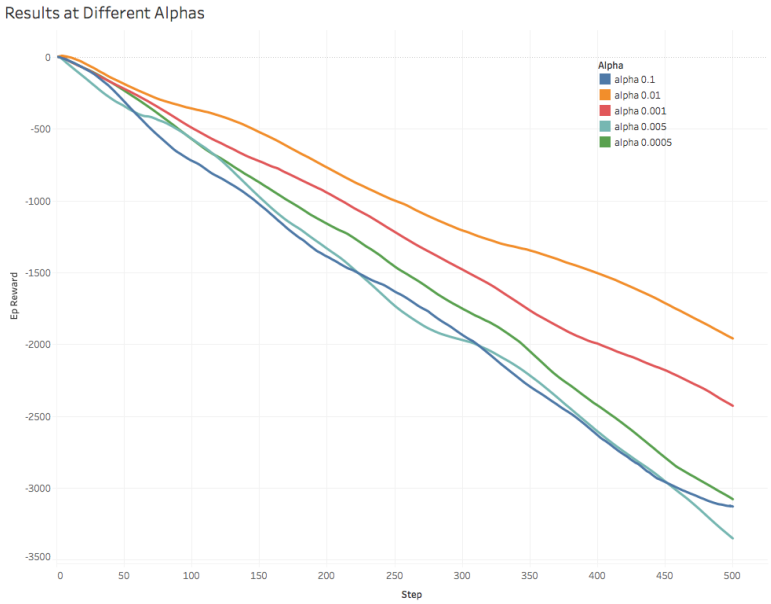

Fig. 1. This graph shows average reward curves over an episode at 5 different alphas. We found that 0.01 was best for training this agent.

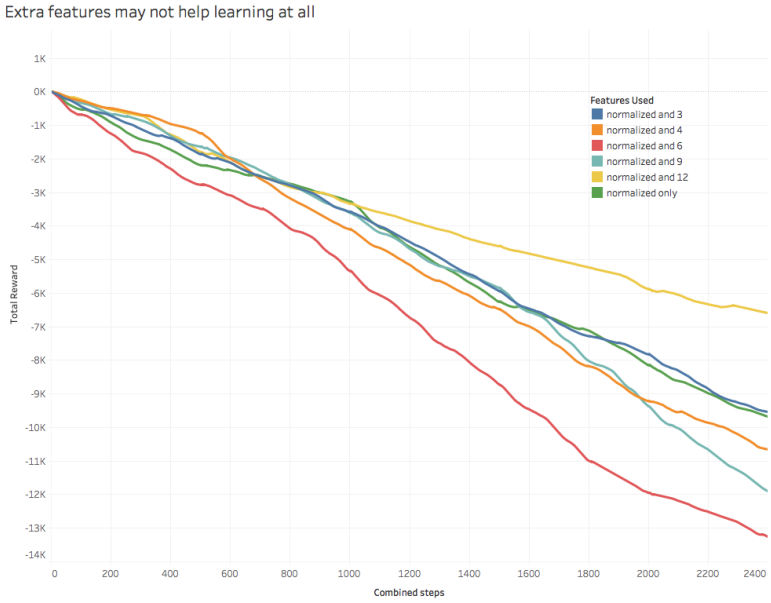

Fig. 2. The effect of additional "engineered" features on performance spanning multiple episodes

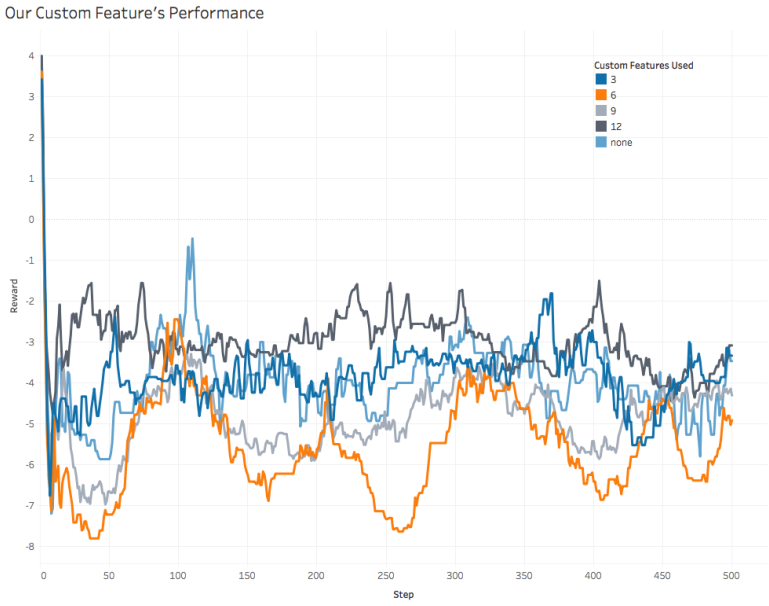

Fig. 3. The effect of custom features on per step return within episodes

The second question we explored was the effectiveness of our "engineered features". These features were created by combining information about the current pitch and roll with the relevant latitudinal and longitudinal actions to correct them to 0. In short, we found that they had very little effect if any at all as can be seen in figures 2 and 3. Fascinatingly, in point of fact, we were totally unable to find a feature set that was able to consistently generate positive rewards. Overall, what we found was that at best, the agent would learn to receive a relatively small negative reward by flying at a slight tilt, in circles, forever without crashing. Naturally, this was considerably better than crashing but ultimately rather unsatisfying.

5 Conclusion and Future Work

It is clear from our results so far that much more work can be done to study the effectiveness of this type of RL in the complex flight space. While it is definitely possible that linear approximation is insufficiently powerful to provide exciting results, it seems to us that this should not be the case. Therefore, our immediate future work should consist of continuing to explore different methods of engineering features and improving our linear q-learner.
In future work, we hope to investigate how the agent performs with additional consideration of other aircraft in the airfield, and how this effects overall safety of the airfield traffic. In order to do so, we will integrate data from an intelligent

traffic data management system to see how the input effects the intelligent agent with its overall success of flying a stable configuration, and if it optimizes overall safety while doing so. The Traffic Data Manager (TDM) system was developed under the advise of our coauthor, Vincent Houston, at NASA Langley Research Center, as a supervised learning system which is able to determine whether aircraft's in an airfield are relevant enough to a flyer to be considered dangerous to a real-time trajectory. The data used to train the learning algorithm was human labelled from subject matter experts, namely, commercial aviation pilots. TDM has been benchmarked for its performance against labelled datasets through comparison and validation studies, but has yet to be observed in an live, online learning setting. Thus, it has never been observed in a human-in-the-loop setting, let alone a fully autonomous environment. It is our goal to observe how TDM enhances autonomous performance, and also to observe how the safety measure of our RL system is effected by the integration of this data.

References

1. Haitham Baomar and Peter J. Bentley. An intelligent autopilot system that learns piloting skills from human pilots by imitation. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016. This paper covers the building of a fully autonomous autopilot system. It used a very short learning period combined with a very small amount of apprenticeship learning to get very impressive results.
2. Haitham Baomar and Peter J. Bentley. Autonomous navigation and landing of large jets using artificial neural networks and learning by imitation, 2017. This paper covers the building of a fully autonomous autopilot system for landing large aircraft. It used a very short learning period combined with a very small amount of apprenticeship learning to get very impressive results.
3. Oren Hazi. Final approach an automated landing system for the x-plane flight simulator. This unpublished paper explores the problem of landing a plane in the X-Plane flight simulator with RL and Imitation Learning.
4. Eduardo F. Morales and Claude Sammut. Learning to fly by combining reinforcement learning with behavioural cloning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 76–, New York, NY, USA, 2004. ACM. This paper uses "Relational Representations" of the state action space for approximating the state action space in flying an airplane.
5. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2017.