REINFORCEMENT LEARNING FOR CONTROL AND MULTIPLAYER GAMES

IIB Project Investigating the application of Google Deepmind's AlphaZero Algorithm to Classical Control Problems

UNIVERSITY OF CAMBRIDGE
DEPARTMENT OF ENGINEERING

Author Alex Darch

Supervisor Dr. Glenn Vinnecombe

Assessor Dr. Ioannis Lestas

St John's College

Cambridge

May 8, 2019

0 Contents

1 Introduction

1.1 Controlling Dynamical Systems

Dynamical systems have long been of great interest to engineers and scientists due to their ability to handily describe real world phenomena. They describe how a system changes through geometrical space with time and, in principle, their trajectories can be predicted solely from their initial state. In recent years this has become relatively easy, even for non-linear systems, by using numerical methods coupled with increases in computing power. However, often a precise solution is of less import than a prediction of the long term qualitative behaviour of the system. Much effort has been made to find methods to describe this long-term behaviour such as those made by Lyapunov in stability theory.

A corollary of the importance of dynamical systems is that influencing their behaviour is particularly useful. Highly non-linear models are difficult to control optimally and robustly, and the few mathematical techniques developed to deal with these can only handle very small subcategories of problems. The study of more general techniques to solve complex control problems has recently come to the forefront of the field with the advent of machine learning techniques such as reinforcement and deep learning. Many of these are not aimed specifically at control problems and are often designed to play games. This project looks at adapting one such algorithm - AlphaZero - from playing board games to solving general control problems such as the control of an aircraft in flight or the control of under-actuated robotics by starting with a simple inverted pendulum.

1.2 Control Theory

The first attempts at controlling dynamical systems came from classical control theory, which consisted of a series of 'cut-and-try' techniques based largely on adjusting the gains of PID controllers and lead/lag compensators until satisfactory closed loop dynamics were achieved [1]

It wasn't until the unification of the calculus of variations, classical control, random process theory and linear/non-linear programming by bellman in the 1950's [1] that truly optimal control was discovered. Optimal control consists of finding the optimal control law for a specified optimality criterion and can be thought of as a non-linear feedback law, $u(t) = -K(t)x(t)$. An optimal control law can be found by solving the Hamilton-Jacobi-Bellman equation in continuous time, or via dynamic programming (DP) in discrete time. DP is a systematic procedure for transforming an optimisation over a sequence of h inputs into h minimisations over 1 inputs (but all states).

$$V(x_k, k) = \underset{u_{k:h-1}}{min}\left(\sum_{i=k}^{h-1} c(x_i, u_i) + J_h(x_h)\right) \tag{1.1}$$

$$= \underset{u_k}{min}\Big(c(x_k, u_k) + V(x_{k+1}, k+1)\Big) \tag{1.2}$$

$$u_k^* = \underset{u_k}{argmin}\Big(c(x_k, u_k) + V(x_{k+1}, k+1)\Big) \tag{1.3}$$

This enables a backwards recursion to find a sequence of value functions. This can be solved for the linear case with quadratic costs (LQR) analytically or, if the system is non-linear, via gradient descent. However, over a finite horizon this is essentially open-loop control. Optimal control of this form has been used to control complex dynamical systems such as spaceflight and aileron folding on aircraft [2, 3]. A closed loop extension to this is Model Predictive Control (MPC), which employs a receding horizon rather than a finite or infinite horizon. MPC can therefore easily deal with plant disturbances and uncertainties, constraints, indefinite horizons and can also be extended to get a control law for non-linear systems. MPC has recently been shown to work in trajectory control for interplanetary rovers [4]. Optimal control is limited in requiring sophisticated models of the environment/plant and generally struggle with highly non-linear models - state of the art is currently linearisation about the predicted trajectory. Furthermore it is only feasible to 'grid' up to 5/6 dimensions in discrete cases.

1.3 Reinforcement Learning

Two further generalisations to dynamical systems and optimal control as defined in equation 1.1 are stochastic dynamics and indefinite horizons (i.e. episodic tasks). This discrete time stochastic control process known as a Markov Decision Process (MPD). In MDPs the cost function is often written as a reward function and, due to the indefinite nature of the process, the value function for the next step is discounted:

$$V(x_k) = \underset{u_k}{max}\left(\sum_{i=k}^{\infty} \lambda^{i-k} r(x_i, u_i) \right) \tag{1.4}$$

$$= \underset{u_k}{max}\, \mathbb{E}\Big[r(x_k, u_k) + \lambda V(x_{k+1}) \Big] \tag{1.5}$$

$$u_k^* = \underset{u_k}{argmax}\, \mathbb{E}\Big[r(x_k, u_k) + \lambda V(x_{k+1}) \Big] \tag{1.6}$$

Reinforcement learning (RL) aims to learn an optimal policy, $\pi^*(x_k)$ ($= u^*(x_k)$ in control) of an MDP. This differs from optimal control in its inherent stochastic nature and therefore can lead to intractable search spaces. A solution to this is to learn form sample trajectories. Algorithms such as Q-Learning, SARSA and DYNA have recently had great success in control applications, for example, their use in controlling mobile robots [6]. Furthermore, the advent of neural networks has led to the extension of these to functional approximations from tabula-rasa methods, making the control highly non-linear dynamical systems possible. Notably, Deepminds' recent success with training a robot to gently manipulate objects [7], would not be possible to reproduce using classical or modern control techniques.

1.4 AlphaZero

References are: [8] [9] [10]

AlphaGo Zero is a revolutionary Reinforcement Learning algorithm that achieved superhuman performance in the game of go, winning 100–0 against the previously published, champion-defeating AlphaGo. It's successor, AlphaZero, is a generalised version that can achieve superhuman performance in many games. There are three key sub-algorithms that form the basis of their success: self-play, a convolutional neural network (CNN) and a Monte-Carlo tree search (MCTS).

1.4.1 Neural Network

1.4.2 Monte-Carlo Tree Search

1.4.3 Self-Play

The application of AlphaZero to dynamical systems has not been tried yet, but it is promising because optimal disturbance modelling, etc. This project investigates to application of the general reinforcement learning algorithm AlphaZero to the control of dynamical systems.

2 Theory & Methodology

Explain the assumptions behind the theoretical development you are using and the application of the theory to your particular problem. Any heavy algebra or details of computing work should go into an appendix.

2.1 The Inverted Pendulum

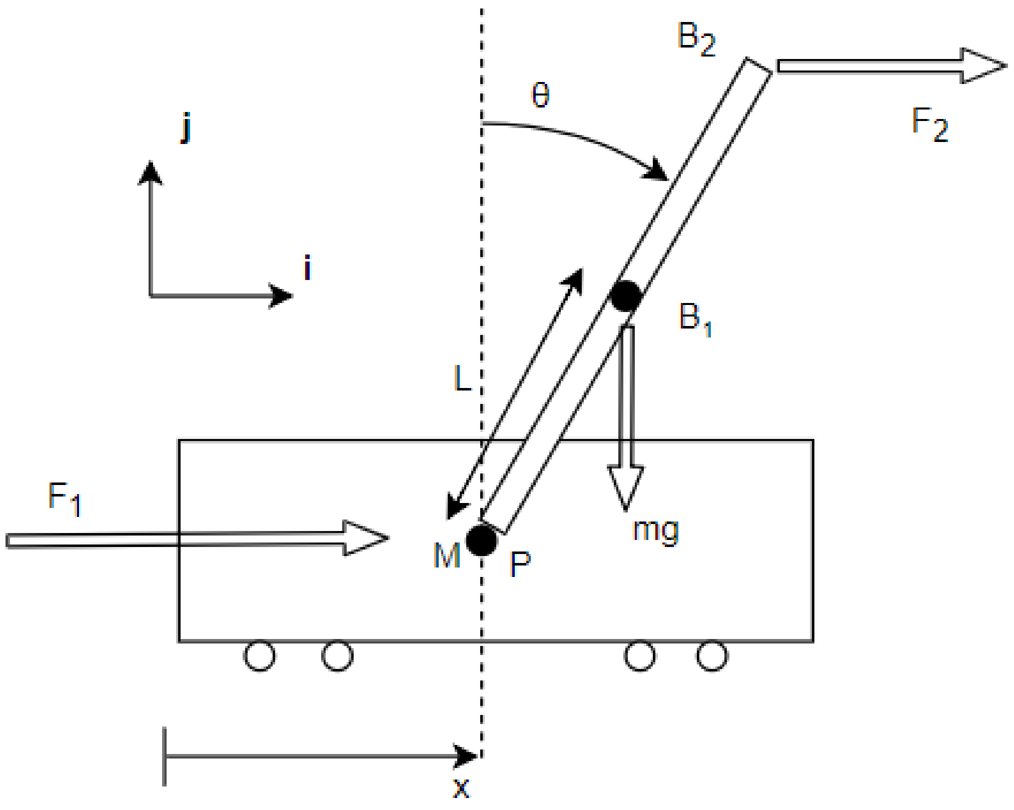

Figure 2.1: A free-body diagram of the cartpole system. For the OpenAI cartpole the system is in discrete time with a time-step of $\tau = 0.02s$. The other constants are $l = 0.5m$, $m = 0.1kg$, $M = 1kg$, $F = \pm 10N$, $x_{max} = \pm 2.4m$, $\theta_{max} = \pm 12^o$.

We can find the state space equations for the Cartpole using d'Alembert forces. Firstly we define the distance and velocity vectors to the important points:

$$\begin{aligned}
&\boldsymbol{r}_P = x\boldsymbol{i} \\
&\boldsymbol{r}_{B_1/P} = Lsin\theta\boldsymbol{i} + Lcos\theta\boldsymbol{j} \\
&\boldsymbol{r}_{B_1} = (x + Lcos\theta)\boldsymbol{i} + L\dot{\theta}sin\theta\boldsymbol{j} \\
&\dot{\boldsymbol{r}}_{B_1} = (\dot{x} + L\dot{\theta}cos\theta)\boldsymbol{i} - L\dot{\theta}sin\theta\boldsymbol{j}
\end{aligned}$$

Linear Momentum, $\boldsymbol{\rho} = \sum_i m_i\dot{\boldsymbol{r}}_{i/o} = m\dot{\boldsymbol{r}}_{B_1} + M\dot{\boldsymbol{r}}_P$:

$$\boldsymbol{\rho} = \begin{bmatrix} (M+m)\dot{x} + ml\dot{\theta}cos\theta \\ -ml\dot{\theta}sin\theta \\ 0 \end{bmatrix}$$

Moment of momentum about P, $\boldsymbol{h}_P = \boldsymbol{r}_{B_1/P} \times m\dot{\boldsymbol{r}}_{B_1}$:

$$\begin{aligned}
\boldsymbol{h}_P &= -mL(L\dot{\theta} + \dot{x}cos\theta)\boldsymbol{k} \\
\therefore \dot{\boldsymbol{h}}_P &= -mL(L\ddot{\theta} + \ddot{x}cos\theta - \dot{x}\dot{\theta}sin\theta)\boldsymbol{k}
\end{aligned}$$

We can balance moments using $\dot{\boldsymbol{h}}_P + \dot{\boldsymbol{r}}_P \times \boldsymbol{\rho} = \boldsymbol{Q}_e$ and $\boldsymbol{Q}_e = \boldsymbol{r}_{B_1/P} \times -mg\boldsymbol{j} + \boldsymbol{r}_{B_2/P} \times F_2\boldsymbol{i}$:

$$\dot{\boldsymbol{h}}_P + \dot{\boldsymbol{r}}_P \times \boldsymbol{\rho} = \begin{bmatrix} 0 \\ 0 \\ -mL(\ddot{x}cos\theta + L\ddot{\theta}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -L(mgsin\theta + 2F_2cos\theta) \end{bmatrix} = \boldsymbol{Q}_e$$

And also balance linear momentum using $\boldsymbol{F}_e = \dot{\boldsymbol{\rho}}$:

$$\dot{\boldsymbol{\rho}} = \begin{bmatrix} (m+M)\ddot{x} + mL(\ddot{\theta}cos\theta - \dot{\theta}^2sin\theta) \\ -mL(\ddot{\theta}sin\theta + \dot{\theta}^2cos\theta) \\ 0 \end{bmatrix} = \begin{bmatrix} F_1 + F_2 \\ R - mg \\ 0 \end{bmatrix} = \boldsymbol{F}_e$$

Finally we can write the system dynamics in terms of $\ddot{\theta}$ and $\ddot{x}$:

$$\ddot{\theta}\big(M + msin^2\theta\big)L = \left(\frac{2M-m}{m}F_2 - F_1\right)cos\theta + g(M+m)sin\theta - mL\dot{\theta}^2sin\theta cos\theta \quad (2.1)$$

$$\ddot{x}(M + msin^2\theta) = F_1 + F_2cos(2\theta) + msin\theta(L\dot{\theta}^2 - gcos\theta) \quad (2.2)$$

Simplifying this for our problem by substituting in constants, we can write the full state space equation:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{\left(\frac{2M-m}{m}F_2 - F_1\right)cos\theta + g(M+m)sin\theta - mL\dot{\theta}^2sin\theta cos\theta}{(M+msin^2\theta)} \\ \dot{\theta} \\ \frac{F_1 + F_2cos(2\theta) + msin\theta(L\dot{\theta}^2 - gcos\theta)}{L(M+msin^2\theta)} \end{bmatrix} \quad (2.3)$$

2.1.1 Linearisation about $\boldsymbol{x}_e$ and Conrollability

Ignoring second order terms and linearising about $\boldsymbol{x}_e = [x_e, \dot{x}_e, \theta_e, \dot{\theta}_e]^T = [0, 0, 0, 0]^T$:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{f+m\theta g}{M} \\ \dot{\theta} \\ \frac{-f-(M+m)g\theta}{lM} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & \frac{mg}{M} & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & -\frac{(M+m)g}{lM} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{lM} \end{bmatrix} f \tag{2.4}$$

It can be proved that the cartpole system is controllable by showing:

$$rank[\boldsymbol{B}\ \boldsymbol{AB}\ \boldsymbol{A^2B}\ \boldsymbol{A^3B}] = 4 \tag{2.5}$$

Therefore for any initial condition we can reach $\boldsymbol{x}_e$ in finite time under these linear assumptions. However, for $\theta \approx 0$ we need a more sophistocated model.

2.1.2 Swing-Up Control

One way to get the cart to swing the pendulum up to the linear-range is to find a homoclinic orbit (a trajectory that passes though an unstable fixed point). I.e. we must find a controller that that drives the pendulum to the unstable equilibrium. This can be done using energy shaping, and in the context of the cartpole, this constitutes applying force to maximise the potential energy and minimise kinetic. Once in the linear region we then switch to an LQR controller to complete the task.

2.2 Neural Network

2.2.1 Architectures

2.2.2 Loss Functions and Pareto

2.3 Scoring and Comparison

3 Experimental Techniques

This section should describe the running of the experiment or experiments and what equipment was used, but should not be a blow by blow account of your work. Experimental accuracy could be discussed here.

3.1 blah blah

4 References

[1] Arthur E. Bryson Jr, *Optimal Control - 1950 to 1985.* IEEE Control Systems, 0272-1708/95 pg.26-33, 1996.

[2] I. Michael Ross, Ronald J. Proulx, and Mark Karpenko, *Unscented Optimal Control for Space Flight.* ISSFD S12-5, 2014.

[3] Zheng Jie Wang, Shijun Guo, Wei Li, *Modeling,Simulation and Optimal Control for an Aircraft of Aileron-less Folding Wing* WSEAS TRANSACTIONS on SYSTEMS and CONTROL, ISSN: 1991-8763, 10:3, 2008

[4] Giovanni Binet, Rainer Krenn and Alberto Bemporad, *Model Predictive Control Applications for Planetary Rovers.* imtlucca, 2012.

[5] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction (2nd Edition).* The MIT Press, Cambridge, Massachusetts, London, England

[6] I. Carlucho, M. De Paula, S. Villar, G. Acosta . *Incremental Q-learning strategy for adaptive PID control of mobile robots.* Expert Systems with Applications. 80. 10.1016, 2017

[7] Sandy H. Huang, Martina Zambelli, Jackie Kay, Murilo F. Martins, Yuval Tassa, Patrick M. Pilarski, Raia Hadsell, *Learning Gentle Object Manipulation with Curiosity-Driven Deep Reinforcement Learning.* arXiv 2019.

[8] David Silver, Julian Schrittwieser, Karen Simonyan et al, *Mastering the game of Go without human knowledge.* Nature, vol. 550, pg.354–359, 2017.

[9] David Silver, Thomas Hubert, Julian Schrittwieser et al, *A general reinforcement learning algorithm that masters chess, shogi and Go through self-pla.* Science 362:6419, pg.1140-1144, 2018.

[10] Yuxi Li, *Deep Reinforcement Learning: An Overview*. CoRR, abs/1810.06339, 2018.