Playing Jedi Academy with Deep Curiosity

Dmitri Tkatch

Abstract

A curious artificial intelligence learns to explore free-for-all maps and defend itself with a saber against built-in bots in duel mode in a few hours of learning on a modest 8GB GPU. It is based on six deep learning neural networks – a future predictor, an action predictor, a state embedder, an image embedder, a snapshot embedder and an pseudo-A3C actor. The AI can use either raw game data, raw pixels or both as input. It uses a novel form of curiosity where unpredictable futures and predictable actions are rewarded. It defeats Cultist 3-1, Jedi Trainer 1-0 while starting at a disadvantage, and the final boss of the game Desann 20-11. We note the importance of curiosity for maintaining a variety of play and our novel form of curiosity for maintaining smooth actions. To our knowledge, this is the first time curiosity has been applied to a realtime 3D game that sees competitive play.

Videos – https://www.youtube.com/@eternalyze0.

Program – https://github.com/Eternalyze0/jediknight.

Correspondence – eternalyze0@gmail.com

1 Introduction

Star Wars Jedi Knight Jedi Academy is a 3D competitive saber-combat game where players may choose from 3 sabers (one saber, dual sabers, staff), 3 saber styles (fast, medium, strong), 7 saber swings (left-to-right, right-to-left, bottom-left-to-top-right, bottom-right-to-top-left, top-to-bottom, top-right-to-bottom-left, top-left-to-bottom-right in each saber style), and 3 katas (one in each saber style). Players are able to move freely in 3D with the WASD keys and look freely with the mouse, even during a saber-swing. Players can also jump with space and switch saber styles with L. Swings are chosen with the direction keys. For instance a bottom-right-to-top-left swing corresponds to pressing the AS keys. Although published in 2003 Jedi Knight Jedi Academy still sees competitive play thanks to its deep melee system. An experienced player may be victorious 10-0 against a novice player. A professional player may be victorious 10-0 against an experienced player. We name our artificial intelligence Sapphire in reference to the dragon Sapphira from Eragon by Christopher Paolini, the Phi neural module from Pathak et al's paper, and the gemstone skin of the blue shadowtrooper from the Jedi Knight game.

2 Reward-Based Learning

An actor A interacts with an environment E over discrete time steps. At each time step t the actor receives a view v_t of the environment and then makes an action a_t. The view v_t includes a reward r_t. Usually the actor's goal is to maximize the a future sum of rewards referred to as the return, R_t,

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + ...$$

$$\gamma \to^- 1$$

As our notation implies, we suggest starting off with a small γ and letting it grow, although we save this for future work.

3 Artificial Intelligence

Actors usually learn to improve their reward-maximization performance by, in one way or another, guessing the future sum of rewards R_t and then checking this guess at $t+1$. The discrepancy between the guess and the check is what drives learning. In neural network-based deep learning, this discrepancy is minimized by nudging parameters in the neural networks which originally produced the guess. If the actor knows how to make perfect guesses as to which actions correspond to which future sums of rewards, then the actor knows how to play perfectly for that given γ. An artificial intelligence is model-based if has a future prediction model.

4 Curiosity

An artificial intellgience is curious if part of its rewards correspond to errors in the future prediction model. Although there are many papers for 2D or toy 3D environments, there are relatively few papers for realistic (3D, realtime, competitive) games. Part of the reason may be because of the exploration challenge introduced by an additional dimension. In particular, while artificial intelligences without curiosity tend to stagnate and get stuck in a mode of behavior, a curious artificial intelligence maintains a variety of play.

5 Program

Our inspiration and implementation is based on Deepak Pathak et al's Curiosity-driven Exploration by Self-supervised Prediction [1], Tim Pearce et al's Counter-Strike Deathmatch with Large-Scale Behavioural Cloning [2], and Seungeun Rho's implementation [3] of A3C [4]. In particular, we use the curiosity module from [1], the interfacing and mouse movement bucketing ingenuity of [2] and the A3C program of [3]. Note however that we do not use the asynchronous

properties of A3C. That is, we only use one actor-learner thread. Note there are no terminal states in our environment, see Algorithm 1.

Algorithm 1 Pseudo-Asynchronous Advantage Actor-Critic

while True **do**
  $d\theta_\pi = 0$ and $d\theta_v = 0$
  $t_{start} = t$
  Get state s_t
  while $t - t_{start} < t_{max}$ **do**
    Perform action a_t according to $\pi(a_t|s_t;\theta_\pi)$
    Receive reward r_t and new state s_{t+1}
    $t = t + 1$
  end while
  $R = V(s_t, \theta_v)$
  $i = t - 1$
  while $i \leq t_{start}$ **do**
    $R = r_i + \gamma R$
    $d\theta_\pi = d\theta_\pi + \nabla_{\theta_\pi} \log \pi(a_i|s_i;\theta)(R - V(s_i;\theta_v))$
    $d\theta_v = d\theta_v + \partial(R - V(s_i;\theta_v))^2/\partial\theta_v$
    $i = i + 1$
  end while
  Perform synchronous update of θ_π using $d\theta_\pi$ and of θ_v using $d\theta_v$
end while

We are also inspired by the couch-potato effect [5] to employ many embedding networks. We perform experiments without curiosity, with the curiosity module and with a modified smooth version of the curiosity module. Without curiosity the artificial intelligence easily gets stuck in some corner of the map and in a repetitive pattern of behavior. With curiosity the artificial intelligence maintains a variety of play and exploratory behavior, preventing it from ever getting stuck. With smooth curiosity the artificial intelligence also never gets stuck but also behaves in more human-like, less jittery way. We utilize a dual input scheme of both raw pixels and raw game client data. We use OCR to scrape the momentum from the on-screen game HUD momentum indicator, though we eventually turn this feature off to maintain 10 actor time steps per second on a modest computer. The game is run normally and no modification is performed to the actor between the three featured videos. That is, the same reward structure is utilized for both exploration and combat. The partial raw game client data consists of roughly 10000 entities or numbers. We cast all the entities to floats and treat them all the same with exception of those utilized in the reward structure as discussed in the next section. In the program the raw game client data is referred to as a snapshot. The raw pixels are referred to as the image. The snapshot is passed to a snapshot embedder and the image is passed to the image embedder. The resulting embeddings are then stacked and passed to the state embedder. The state embedding is passed to the actor model which then produces actions. The state embedding and action are passed to the future pre-

dictor which then produces the next time step predicted state embedding. The mean squared error between the actual next time step state embedding and the predicted next time step state embedding is what is referred to as the curiosity. The actual state embeddings at the current and next time steps are passed to the action predictor, which produces the predicted action. The mean squared error between the predicted action and actual action is referred to as the action prediction error. Together, the state embedder, action embedder and future predictor form a curiosity module as described in [1]. There are 48 actions total – 7 keys, 3 mouse buttons, and 38 mouse movement buckets. All actions are treated with binary classification. That is, an action is either taken or not and actions are discrete.

6 Motivation

Innate motivation is dedicated to unsupervised A3C learning. That is, instead of rewards coming from the environment the artificial intelligence is free to choose which goals it would like to undertake. The curiosity module not only rewards error in the future prediction model (which predicts the future state given the present state and action) but also employs an action predictor model (which predicts actions given a state transition). Moreover neither the future nor action predictor models operate on states directly but rather utilize a neural state compression module. The action predictor model backpropagates errors to the neural state compression module. We modify the reward structure of [1] to reward predictable actions thereby making AI behavior more human-like,

$$r_c = 100(E_f - E_a)$$

$$r_m = M$$

$$r_d = H + 0.5D$$

$$r_s = 100(V - 0.5L)$$

$$r_b = 1000$$

$$r = r_c + r_m + r_d + r_s + r_b$$

where r_c is the curiosity reward, E_f the future prediction error, and E_a the action prediction error, r_m the momentum reward, M the momentum (usually ranges from (0-1000), r_d the health reward, H the player health, D the damage inflicted, r_s is the score reward, V is the number of victories, L is the number of losses, r_b is the baseline reward, and r is the combined reward which is received by the actor neural module. We use a high baseline reward because it has been shown that optimistic initial values are beneficial for exploration [6]. We use dropout instead of ϵ-greedy strategies.

7 Results

See videos. Sapphire runs on one 8GB graphics processing unit, explores half of Tattooine (roughly 10 areas or "rooms") in 36 minutes, adequately defends itself against built-in bots, and learns to do all this in a few hours. Professional human players say the AI is better and more interesting to duel against than built-in bots. The AI explores quickly even if $r = r_c$ and the input is only raw pixels. If r_c is set too small then the AI stagnates into one mode of behavior. Rewarding predictable actions results in smoother behavior. Finally, we remark that there are unusually few papers in the literature which apply deep learning to realtime, competitive 3D games.

8 References

We thank the authors of these references for their dedication as well as friends and coworkers who helped with interfacing with the game and editing the paper,

[1] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction. In ICML 2017.

[2] Tim Pearce, Jun Zhu. Counter-Strike Deathmatch with Large-Scale Behavioural Cloning, In proc. of IEEE Conference on Games (CoG), Beijing, China, 2022.

[3] Seungeun Rho. A3C. 2019.

[4] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. arXiv, 2016.

[5] Yuri Burda, Harrison Edwards, Amos Storkey, Oleg Klimov. Exploration by Random Network Distillation. arXiv, 2018.

[6] Richard Sutton, Andrew Barto. Reinforcement Learning: An Introduction. 2018.