

Beating Atari Breakout with Asynchronous Advantage Actor-critic

Donghun Lee

Udacity, Machine Learning Nanodegree

DHFROMKOREA@GMAIL.COM

Abstract

I review and evaluate an Asynchronous Advantage Actor-critic (A3C) model Chow and Liu (1968) that successfully learns to play Atari Breakout with visual input. A3C is a reinforcement learning algorithm that runs multiple agents in parallel to train their deep neural network controllers. I show that the training time for A3C can be significantly lower than Deep Q Network (DQN) requires and that the training can be stabilized without experience replay. A3C's performance has been evaluated against DQN with prioritized experience replay and cross entropy method (CEM). Pros and cons discussed in the discuss.

Keywords: A3C, Deep Reinforcement Learning, Atari, Q Learning

1. Introduction

The field of Reinforcement Learning has recently seen breakthrough results by approximating Q action-value functions with Deep Neural Networks (DQN). DQN has been shown to beat expert human players on a number of the ATARI games by training only with raw visual inputs of the games and without any domain knowledge specific to the games given [4, 7]. Despite DQN's promising results, it is known to be unstable in its vanilla form. There are three main causes to this. First, the samples an agent experiences are highly correlated (non-iidness). Second, a small update to the Q action value function can lead to a large change to the data the agent experiences (non-stationarity). Third, the action values (Q) and the target (reward + $\gamma \max Q$) are correlated, just like a cat chasing its own tail (non-stationarity). To remedy the instability, Experience Replay (EP) and the trick of using a separate target were developed, and achieved a stabilizing effect.

However, EP has several drawbacks. First, it requires more memory and computation because the agent must keep track of the storage of experiences and in the case of Prioritized EP, the task of updating the priorities of the samples can be computationally heavy. Second, EP forces us to do off-policy learning where on-policy learning algorithms like SARSA, actor-critic methods is impossible.

Asynchronous Advantage Actor Critic (A3C) is an algorithm proposed to solve the issues mentioned above. A3C is deemed outperforming DQN, in terms of the simplicity of implementation, the training time, and performance. Unlike DQN with EP that uses a single agent interacting with the environment, A3C runs multiple agents where each agent interacts, in parallel (asynchronously), with a different instance of the environment. This scheme has the effect of reducing the correlations between experience samples, and making the learning more stationary because the variety of experiences will increase. Practically, the parallel scheme allows the training time to be faster than in DQN, and the training can be performed on a single computer with a standard multi-core CPU.

The problem to solve is building a reinforcement learning agent to play autonomously Breakout providing minimal game-specific knowledge to the agent: the input data are visual images and the number of actions available. The goal of Breakout is that the agent must move a paddle so the ball bounces off it to break bricks. The more bricks it breaks at given time, the better. Concretely, I aim to build an agent that learns to acquire over an arbitrary amount of 500 points.

2. Related Work

Probabilistic inference has become a core technology in AI, largely due to developments in graph-theoretic methods for the representation and manipulation of complex probability distributions (Pearl, 1988). Whether in their guise as directed graphs (Bayesian networks) or as undirected graphs (Markov random fields), *probabilistic graphical models* have a number of virtues as representations of uncertainty and as inference engines. Graphical models allow a separation between qualitative, structural aspects of uncertain knowledge and the quantitative, parametric aspects of uncertainty...

3. System Description: A3C

1. Environment * openAI's gym: <https://gym.openai.com/envs/Breakout-v0> * state: continuous. Game Memory (RAM). * action: discrete. ["LEFT", "RIGHT", "NOOP"] * observation: visual input: 210 x 160 x 3. * ALE environment also has a peculiar hidden feature called action repeat probability where every time an agent takes an action, the environment ignores the action with a 25% probability. * OpenAI environment seems to enforce the frame skipping their own way where K becomes a variable on 1, 2, 3 to add stochasticity. Once an agent takes an action, it may be repeated once, twice or three times. There is no component-wise maximum value frame encoding. I conjecture OpenAI fixed the flickering issue when they created the environment.

2. Input preprocessing: * raw input: pixel-based image: 210 (width) x 160 (height) x 3 (RGB, 128 color palette). We assume this much of resolution is not necessary to learn an optimal policy and can cause a computation bottleneck in real-time play. Therefore, we preprocess the raw sensory input to a lower dimension. * frame encoding: [4] encoded a frame by taking component-wise maximum value over the current raw input frame and the previous one on each color channel. For example, if the current frame has (R:34, G:50, B:102) and the previous (R:4, G:100, B:202), the encoded current frame is (R:34, G:100, B:202). This was necessary to remove flickering (pixels not showing when they should) that is present in some ATARI games. * downsampling: downsample the RGB frame to 84 x 84 image. In [4], linear scaling (bilinear interpolation) was done to downsample without keeping the original image aspect ratio. In [7], the image was downsampled to 110 x 84 with the same aspect ratio and an additional step of cropping to 84 x 84 was taken. * greyscale conversion: grey scale conversion was done through the built-in ALE (Arcade Learning Environment) greyscale conversion method. * frame skipping: to reduce the computational burden and address partial observability, the agent outputs its decision every kth frame. The k in our experiment will be 4, the same as [4, 7]. During the k number of frames, the agent repeats

the same action decided k times. * final input: K most recent consecutive frames will be stacked to form a final input of 4 x 84 x 84. The agent makes a decision by looking k steps backwards that likely include motion-related information. Note the consecutive final inputs are overlapping by (k-1) frames.

3. Output: * action ["LEFT", "RIGHT", "NOOP"]

3.1 Evaluation Metrics

1. training time to achieve an appropriate score level (e.g. 100 points) 2. total average points per episode (moving average over 100 episodes) acquired by a successfully trained model.

3.2 Baseline methods

1. DQN with Experience Replay This will be trained on my local machine. Since training DQN successfully may take days, I will try to use Google Cloud or EC2 for training it. If possible, I may try to evaluate double DQN(D-DQN), D-DQN with prioritized experience replay, or dueling D-DQN.

2. CEM (Cross Entropy Method) This is an evolutionary algorithm that is considered a well-performing baseline. The implementation will share the implementations of [5] and [6].

4. Experiments

The solution is an A3C algorithm. The A3C model will be trained on my local machine: Dell XPS 9560 with quad-core i7 CPU and NVIDIA GTX 1050 (4GB memory). The trained model will be evaluated on my machine whose result will be posted publicly on the OpenAI's scoreboard. 500 points is on par with some of the highest scores posted there.

4.1 visualization

visualization

4.2 results table

future table

5. Conclusion

Probabilistic inference has become a core technology in AI, largely due to developments in graph-theoretic methods for the representation and manipulation of complex probability distributions (Pearl, 1988). Whether in their guise as directed graphs (Bayesian networks) or as undirected graphs (Markov random fields), *probabilistic graphical models* have a number

5.1 Further work

of virtues as representations of uncertainty and as inference engines. Graphical models allow a separation between qualitative, structural aspects of uncertain knowledge and the

quantitative, parametric aspects of uncertainty...

Remainder omitted in this sample. See <http://www.jmlr.org/papers/> for full paper.

Acknowledgments

We would like to acknowledge support for this project from the National Science Foundation (NSF grant IIS-9988642) and the Multidisciplinary Research Program of the Department of Defense (MURI N00014-00-1-0637).

Appendix A.

In this appendix we prove the following theorem from Section 6.2:

Theorem *Let u, v, w be discrete variables such that v, w do not co-occur with u (i.e., $u \neq 0 \Rightarrow v = w = 0$ in a given dataset \mathcal{D}). Let N_{v0}, N_{w0} be the number of data points for which $v = 0, w = 0$ respectively, and let I_{uv}, I_{uw} be the respective empirical mutual information values based on the sample \mathcal{D} . Then*

$$N_{v0} > N_{w0} \Rightarrow I_{uv} \leq I_{uw}$$

with equality only if u is identically 0. ■

Proof. We use the notation:

$$P_v(i) = \frac{N_v^i}{N}, \quad i \neq 0; \quad P_{v0} \equiv P_v(0) = 1 - \sum_{i \neq 0} P_v(i).$$

These values represent the (empirical) probabilities of v taking value $i \neq 0$ and 0 respectively. Entropies will be denoted by H . We aim to show that $\frac{\partial I_{uv}}{\partial P_{v0}} < 0 \dots$

Remainder omitted in this sample. See <http://www.jmlr.org/papers/> for full paper.

References

- C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, IT-14(3):462–467, 1968.
- Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman Publishers, San Mateo, CA, 1988.