Deep Q-Learning with Recurrent Neural Networks

Clare Chen
cchen9@stanford.edu

Vincent Ying
vincenthying@stanford.edu

Dillon Laird
dalaird@cs.stanford.edu

Abstract

For our project, we plan to explore a newer subfield of reinforcement learning called deep Q learning (DQN). In DQN we train a neural network model with a variant of Q-learning, where our input is raw pixels and our output is a value function estimating future rewards [1].

1 Introduction

In DQN we use a neural network to predict our action value, the Q function. The algorithm is similar to Q learning except for two distinctions. The first is called experience replay which randomly selects a past state to use in its gradient. This "break up" the learning from using consecutive correlated sequences and helps reduce variance in the updates. The second distinction is to make a copy of the Q network to generate targets to train on. This helps reduce the chance of oscillations or divergence. [0]

We think there are several ways of improving DQN. Current deep RL algorithms use slow gradient-based updates of policy or value functions. As a result, these algorithms cannot rapidly assimilate winning game strategies. We wish to experiment with approaches that can rapidly update experience replay. Experience replay can require significant memory overhead and naively samples over the set randomly to help with training. We want to investigate ways to store less experiences and also select experiences that might provide a better gradient update than a completely random experience. One way to attack this problem is to prioritize the experiences by some metric that would lead to better updates.

A key component of the DQN algorithm is the preprocessing function ϕ which maps observations and action sequences (experiences) to our Q function. In [0] a convolutional neural network is used to map the game images. This can be improved upon by feeding the output of the convolutional neural network into an recurrent neural network (RNN) as in [4] to provide a representation over multiple observations. We can take this idea even further by augmenting the RNN with an attention-like mechanism or a memory component so it can obtain a better representation of states even further in the past than what a typical RNN could do.

There are several metrics we can use. One is the score achieved on each game. This can be found in extended data table 2 [0]. Another metric is average score per episode or average action value over training epochs, found in figure 2 [0]. We will be focusing on games that DQN struggles with such as Ms. Pac-Man and Frostbite. This can be used to compare the speed at which different algorithms learn. To run the simulations we will be using OpenAI's gym toolkit for simulating Atari games and we will use Tensorflow to construct and execute the algorithms.

2 Experiments

Experiments.

References

0. "Human-level control through deep reinforcement learning" Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis

1. "Playing Atari with Deep Reinforcement Learning" Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller

2. "Model-Free Episodic Control" Charlse Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Runderman, Joel Z Leibo, Jack Rae, Daan Wierstra, Demis Hassabis

3. "Neural Machine Translation By Jointly Learning To Align and Translate" Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio

4. "Deep Recurrent Q-Learning for Partially Observable MDPs" Matthew Hausknecht and Peter Stone