Machine Learning Engineer Nanodegree : Capstone Proposal

Aarshee Mishra

Domain Background

I have chosen the domain of reinforcement learning for my capstone project. Reinforcement Learning has been around for a long time, with the work on Markov Decision Processes (MDP's) and Dynamic Programming done by Richard Bellman as early as in 1950's. He is also credited with the famous Bellman Equation which is the cornerstone of modern day reinforcement learning. In recent years we have seen a renewed interest in the same after Deepmind combined reinforcement learning and convolutional neural networks to play Atari games. They beat the state of the art methods on six games out of seven, outperforming human experts on three of them (Playing Atari with Deep RL). Later on, they designed AlphaGo, a computer program trained using reinforcement learning which went on to beat Mr. Lee Sedol, considered as greatest player of the last decade in the game of Go (AlphaGo). Go is an ancient Chinese board game which in spite of its simple rules is much more complex than other board games such as chess.

Problem Statement

In this project I want to solve the problem of solving a 3 * 3 rubik's cube. Rubik's cube is a puzzle game which consists of 6 faces, each having a different color. Since there are six faces with each face consisting of 9 blocks, number of positions on the cube are 9 * 6 = 54. Now each position can have 6 colors hence we can represent each colored position by a one-hot vector of size 6. Thus we can then represent one state of cube by vector of size 54 * 6 = 324. This is also measurable as we can check how many times we are able to solve a scrambled cube and reproducible as it will generate similar results over a sufficiently large dataset of scrambled cubes of simlar difficulty.

Dataset and Inputs

Dataset for training and testing can be generated by ourselves. We can have a class for generating rubik's cube. We can have a rotate method for rotating each of the six sides of the cubes both clockwise and anticlockwise. Now using these methods we can randomly scramble the cube to various degrees of complexity by rotating its various faces different number of times. One more method can be used to orient the cube in different orientations, for example red side facing front or blue side facing front.

Solution Statement

We will use deep Q-learning networks(DQN) for for solving the problem as described in the Deepmind atari paper. It is essentially just the Q-learning with Action value function approximated using a deep neural network. We will have a separate neural network for calculating the target value i.e.

$$Target = \max_{action} Q(next\ state,\ action).$$

We will also use experience replay for approximating the action value function. Experience replay is storing a bunch of states, actions, rewards and next states tuples in a buffer which are used for batch update every iteration. We will also use epsilon greedy approach for explore exploit dilemma where we will slowly decrease epsilon leading us to make more greedy decisions and reduce the exploration. One approach that we can take is to see if the model converges to solution more quickly if we slowly increase the complexity of the scrambled cube similar to curriculum learning.

Benchmark Models

Lot of work has been done on solving rubik's cube using machine learning. People have tried to solve the problem using both reinforcement learning as well as supervised learning. It has been already proven that the God's number i.e. the maximum number of moves required to solve the rubik's cube is 20 (God's Number Is 20)

Using supervised learning people have attained good accuracy upto 5 scrambles, which declines sharply coming to 10 moves (Exploring Boosted Neural Nets for Rubik's Cube Solving). Szymon Matejczyk has done some work on trying to solve n * n cube using reinforcement learning, but his results are also not very encouraging (Learning to solve Rubik's cube). Unfortunately, I'm not able to find exactly what accuracy he was able to obtain on solving 3*3 cube (Rubiks-Cube-Neural-Network)

Evaluation Metrics

The evaluation of the algorithm can be done by seeing how it behaves at various complexities of the cube. We can generate sufficiently large number of cube scrambled states at various complexities and see what fraction of them we are able to solve.

Project Design

First of I will have a rubik's cube class. This will have methods for rotating a face of the cube in particular direction, for scrambling the cube, for changing cube orientation, getting the state of the cube as 324 size vector, checking if the cube is solved or not, resetting the cube. For the Q function approximation I will use Keras library for creating deep neural network and for optimization function. Input size will be 324 and output to be 12, as we can rotate each face in both clockwise and counterclockwise direction. Since our loss function is somewhat involved, I will write custom loss function.

$$Loss = reward + \gamma \cdot \max_{action} Q(next\ state,\ action) - Q(state,\ current\ action)$$

There will be a method which plays one episode of the game while updating the experience replay buffer. Each action will be sampled from the action value model using epsilon greedy approach and after each action is taken we will train the model. Since I want to use experience replay and there won't be any experience at the starting of the game, we can initially play some episodes taking random action each time. For the experience replay buffer we will store four tuples of state, action, reward and next state. Reward will be 100 for solved cube and -1 for the rest of the states. There would also be an upper limit on the number of actions we will take per episode as we want to solve the cube in a reasonable number of moves.

References

God's Number Is 20 http://www.cube20.org/.

Exploring Boosted Neural Nets for Rubik's Cube Solving https://www.alexirpan.com/public/research/nips_2016.pdf

Rubiks-Cube-Neural-Network https://github.com/germuth/Rubiks-Cube-Neural-Network

Playing Atari with Deep RL https://deepmind.com/research/publications/playing-atari-deep-reinforcement-learning/

AlphaGo https://deepmind.com/research/alphago/

Deepmind atari Paper http://arxiv.org/pdf/1312.5602v1.pdf

Learning to solve Rubik's Cube https://youtu.be/5Mm6NQXVLAg?t=23m24s