Paper: Playing Atari Deep Reinforcement Learning

Summary

In this paper, the authors present a deep reinforcement learning model that uses a convolutional neural network (CNN) to learns control policies directly from high-dimensional sensory input. They use a variation of the traditional Q-learning. They apply their method to seven Atari 2600 games from the Arcade Learning Environment with no adjustment of the architecture or learning algorithms.

Applying reinforcement learning (RL) on sensory data presents several challenges from a deep learning perspective. Firstly, most successful deep learning applications require large amounts of hand-labeled training data whereas RL algorithms must be able to learn from a scalar reward signal that is frequently sparse, noisy, and delayed. Another issue is that most deep learning algorithms assume the data samples to be independent, while in RL one typically encounters sequences of highly correlated states. Furthermore, in RL the data distribution changes as the algorithm learn new behaviors, which can be problematic for deep learning methods that assume a fixed underlying distribution. In this paper, the authors demonstrate that CNN can overcome these challenges to learn successful control policies from raw video data in complex RL environments. They used an experience replay mechanism that randomly samples previous transitions and thereby smooths the training distribution over many past behaviors. This method helps in reducing the problem of correlated data and non-stationary distributions associated with RL as mentioned above.

The main goal that the authors are trying to achieve in this paper is to connect an RL algorithm to a deep neural network which operates directly on RGB images and efficiently process training data by using stochastic gradient updates. In contrast to TD-Gammon and similar online approaches, they utilize a technique called experience replay where they store the agent's experiences at each time-step in a dataset pooled over many episodes into a replay memory. They also utilize a €-greedy policy after performing experience replay. Since using histories of arbitrary length as inputs to a neural network can be difficult, their Q-function instead works on fixed-length representation of histories produced by a function Ø. They call this algorithm Deep Q-learning. Some of the advantages of this approach are data efficiency allowing to update many weights at each step, breaking correlation and reducing the variance of the updates by randomizing the samples.

During training, they made one change to the reward structure of the games. Since the scale of scores varies greatly from game to game, they fixed all positive rewards to be 1 and all negative rewards to be -1, leaving 0 rewards unchanged. This approach limits the scale of the error derivatives and makes it easier to use the same learning rate across multiple games. However, it could also affect the performance of the agent since it can not differentiate between rewards of different magnitude. They used the average total reward and action-value function metric to track the performance of their network. They found out that action-value function is a more stable metric among the two.

In comparison to other models that were fed with some information about the environment and the objects, the authors did not provide any such information to their DQN model and still, the model was able to generalize across various games. They showed that on all the games, except Invaders, their model performed better than other models. They also showed that their model achieves better performance than an expert player on games such as Breakout and Enduro. However, in the games such as Sea quest and Space Invaders, their model is far from the human level of performance. They explain that such games require the network to find a strategy that extends over long time scales which is more challenging.

Strengths

- The paper includes their algorithm for Deep Q-learning which will help the reader understand it better and even replicate it.
- They have a dedicated related work and background section which gives a good understanding of the previous models and how their model differs.

Weaknesses

- The figures they have included are most of their results and the game environment. If possible, they could have included some figures on their model's architecture. It will help understand the algorithm better.

Confusions

- I did not fully understand the concept of the action-value function.
- I was also a little confused about how they used the experience replay in their model.

Discussions

- What are the Markov Decision Process (MDP) and Bellman function?
- How is deep Q-learning different than online Q-learning?
- What is a function approximator? What is the difference between linear and non-linear function approximators?