Deepmind paper-Atari

Complete Summary of Article:

Playing Atari with Deep Reinforcement Learning

The main purpose of the study is to apply the deep learning model with the reinforcement learning to all seven Atari 2600 games. The deep learning model has convolutional neural network with the raw pixels input layer, and function produces future rewards as an output. Used reinforcement learning learn control policies from high-dimensional raw input i.e video data.

Firstly, it is quite **challenging** to apply reinforcement learning with the deep learning because deep learning applications required a huge amount labelled training data set, on the other hand, RL algorithms based on scalar rewards signals, and those signals are too noisy, sparse (this metrics contains mostly zero values), and delay between actions and rewards. The most common issue is that in most of the deep learning algorithms data samples that are selected randomly so that its observations do not depend on the value's other observations, but in RL the data is typically highly correlated with the each other. Thus, in this paper they have applied convolutional neural network to overcome these challenges and successfully apply control policies in complex RL environments.

To **train the network** they have applied different Q-learning algorithm (such as value based, policy based, and off-policy), with the stochastic gradient decent (SGD) optimizer to update the weights. Also, they have used Experience Replay Mechanism technique to store the agent experiences at each time-step to reduce the problem of highly co-related data, and noisy data.

They have considered an agent that directly interact with an **Atari emulator environment** (represents as E), actions that taken by the agent, observations, and rewards. The Atari emulator is responsible to taken care of its internal states and change the score of the game each timestamp. So, we can say that here the E (which is environment) can be stochastic. Since the emulator internal states observes the images from the raw pixel values of the each current screen, so it is almost impossible to decide the agent actions from only the current screen scenario. Thus, they have decide to consider the sequence of actions and observations, $s_t = x_1 + a_1 + x_2 + a_2,\ldots a_{t-1} + x_t$. Here the a= agent's action x= image observation and S= sequence of actions and observations. Since, sequences in the emulator has finite number of time-steps, so control this they have used standard reinforcement learning method, which is **MDP (Marlov decision process)**. So, the main goal is to The goal is to maximize the future rewards that are discounted by a factor n.

The optimal value function is defined as $Q^*(s,a) = \max\pi E [Rt| st = s; at = a; \pi]$ that which is basic idea of **Bellman Equation**. The authors propose to use a non-linear function approximator such as a neural network to estimate the optimal Q value to find the **optimal value function**. The Q network can be trained by minimizing the loss functions $L_i(\Theta_i)$ that changes at every iteration i. The key thing is that here the Q- learning algorithm is model-free, which means it doesn't need to learn the main rules of the game (no teacher), like a model-based approach. It is also off-policy. In fact, they also used multiple approaches, instead use just e-greedy algorithm each and every time.

The most closely related work in this paper is **Neural fitted Q-learning** (NFQ), the main task of NFQ is to **optimizes** the loss function using the known as **RPROP algorithm** to update the parameters of the Q-learning. **But**, again the problem they have faced is that they uses stochastic gradient descent that has low constant cost per iteration and can scale to large datasets. NFQ used deep autoencoders to learn a low dimensional representation of the task. However, according to their approach reinforcement learning end-to-end directly collect the data from raw visual inputs thus learning features that are directly relevant to discriminating action values.

As the paper aims to **connect a reinforcement learning algorithm to a deep neural network** that directly takes in RGB images as input and processes it using SGD (stochastic gradient decent). This paper uses a method called Experience Replay. In which, the agent's experience at each step is stored as $e_t = (s_t, a_t, r_t, s_t + 1)$ in a dataset pooled over multiple episodes and is called replay memory or experience reply. The Q-learning updates are applied to a random batch of samples from the pool, so during the training data samples more random and uncorrelated, thus making it more 'stationary' to the neural network as each new batch is filled with random strategy experiences. After performs this computation the agents select and executes an action according to an E-greedy policy.

Thus, Deep Q-learning algorithm with **Experience Replay has the following advantages:**

- Each step of the experience is potentially used in many weight updates, which allows much efficiency in for large data.
- By using experience replay the behavior distribution is averaged over many of its previous states, smoothing out learning and avoiding divergence in the parameters.

The **input image in RL model is 210 x 160**, which is large and computationally very expensive. So, to overcome this problem, the raw image is pre-processed by **reducing it to greyscale** with dimension 110 x 84. And accepts only square inputs.

Instead, giving a **both state**, action to the Q using neural network, in this architecture, only the state representation is given as input and separate output units are present for each possible action for the given state. The main advantage of this type of architecture is the ability to compute Q-values for all possible actions in a given state with only a single forward pass through the network.

Thus, the **final input to the neural network** consists of an 84 x 84 x 4 image produced by theta. The first layer has 168 x 8 filters with stride 4, the ReLu activation 32 4x4 filters with stride 2, again followed by ReLu activation a **fully connected layer**, 256 rectified units which is non-linier.

In summary, one particular success of this paper is in its ability to train a large neural network with reinforcement learning and SGD without encountering any divergence issues. DeepMind was able to achieve better performance than an expert human player on their games which are Breakout, Enduro and Pong and achieve close to human performance on Beam Rider. The DQN method proposed in this paper can successfully solve problems with high-dimensional observation spaces but cannot handle high-dimensional action spaces. Many physical control tasks have continuous high dimensional action spaces. This has paved the way for further research in the applicability of DQN.