Playing Atari with Deep Reinforcement Learning

November 30, 2023

1 Summary

0. The paper implements a successful algorithm in Reinforcement Learning that uses then recent developments in deep learning. It does this by outperforming all existing algorithms in 6 out of 7 Atari 2600 games. Challenges relative to supervised learning include: not having labeled data, significant time gap between actions and rewards in the process of learning, data that is correlated, and changing data distribution unlike supervised learning.

1. **Background:** The agent interacts with an emulator $\mathcal{E}$. At each time step the agent selects action $a_t \in \mathcal{A}$. The action is passed to the emulator which then changes game score and its internal state. Agent only observes image $x_t \in \mathbb{R}$ and change in game score r_t. The agent decides on action based on the sequence $s_t = x_1, a_1, \cdots, a_{t-1}, x_t$. This is treated as a Markov Decision Process with state s_t. Future discounted return at t is $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$.

   Optimal action-value function $Q^*(s, a)$ is the maximum expected reward attainable after observing (s, a) from any strategy π(maps sequences to actions),

   $$Q^*(s, a) = \max_{\pi} \mathbb{E}\{R_t | s_t = s, a_t = a, \pi\}.$$

   Using recursive nature of the problem we can write:

   $$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}}\big[r + \max_{a'} Q^*(s', a') | s, a\big].$$

   The basic idea behind reinforcement learning is to estimate Q^* function using the Bellman equation as an iterative update $Q_{i+1}(s, a) = \mathbb{E}_{s' \sim \mathcal{E}}\big[r + \max_{a'} Q_i(s', a') | s, a\big]$. However, doing this for every sequence is not practical, reinforcement learning uses function approximator $Q(s, a, \theta) \approx Q^*(s, a)$. Typically a linear function approximator is used but this paper relies on a neural network called Q-network.

   The Q-network is trained by minimizing loss function

   $$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)}\big[(y_i - Q(s, a; \theta_i))^2\big],$$

where $y_i = \mathbb{E}_{s' \sim \mathcal{E}}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}|s, a)]$ and $\rho(s, a)$ is the probability distribution over s, a called *behavior distribution*. Target y_i keeps changing in contrast to supervised learning where it is fixed. The parameters θ_{i-1} is fixed when optimizing $L_i(\theta_i)$. The Q-learning algorithm implements this by replacing full expectations by single sample at a time using ρ. It is an off-policy approach from the 90s.

2. In contrast to above algorithm, the past few samples are stored in memory and sampled in the algorithm. This helps in avoiding correlations in consecutive states and hence greater learning efficiency.

3. To improve algorithm only state is used in the input space of Q-network as opposed state-action pairs used in earlier algorithms. They use the CNN architecture by AlexNet paper to process the images.

2 Analysis

Motivated by the AlexNet paper, this paper applied the framework in Reinforcement Learning context and obtains great improvement over earlier algorithms. The modifications over earlier Q-networks is only minor. However, this paper demonstrates that success in one area of Machine Learning can be beneficial in another.