Playing atari with deep reinforcement learning

Introduction

• Create a single neural network agent that is able to successfully learn to play as many of the games as possible

• The network has outperformed all previous RL algorithms on six of the seven games we have attempted and surpassed an expert human player on three of them
Introduction
Background

Optimal action-value function

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q^*(s', a') \mid s, a \right]$$

Loss function

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

$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid s, a \right]$$

Loss function (gradient)

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[ \left( r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right) \nabla_{\theta_i} Q(s, a; \theta_i) \right]$$
Deep Reinforcement Learning

- TD-Gammon
  - A computer backgammon (西洋雙陸棋戲) program
- On-policy
- Experience replay
  - Store the agent’s experiences at each time-step, \( e_t = (s_t, a_t, r_t, s_{t+1}) \) in a data-set \( D = e_1, ..., e_N \)
- Off-policy (Q-learning)
Deep Reinforcement Learning

Algorithm 1 Deep Q-learning with Experience Replay

1. Initialize replay memory $\mathcal{D}$ to capacity $N$.
2. Initialize action-value function $Q$ with random weights.

for episode = 1, $M$ do

1. Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$.

for $t = 1, T$ do

1. With probability $\epsilon$ select a random action $a_t$.
2. Otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$.
3. Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$.
4. Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$.
5. Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$.
6. Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$.
7. Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$.
8. Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3.

end for

end for
Preprocessing

- Raw: 210 × 160 pixel images with a 128 color palette
- Convert into gray scale, down-sampling to 110x84
- Crop an 84x84 region (Conv2D they selected expects square inputs)
- Stack the last 4 frames
Model Architecture

• The input to the neural network is an $84 \times 84 \times 4$ image

• First layer: convolves 16 $8 \times 8$ filters, stride 4, rectifier nonlinearity

• Second layer: convolves 32 $4 \times 4$ filters, stride 2, rectifier nonlinearity

• Final layer: fully-connected and consists of 256 rectifier units

• The output layer is a fully-connected linear layer with a single output for each valid action.

• The number of valid actions varied between 4 and 18 on the games we considered
Experiments

- Set positive rewards to 1, negative rewards to -1, and unchanged rewards to 0

- Minibatch size = 32

- $\epsilon$-greedy with $\epsilon$ annealed linearly from 1 to 0.1 over the first million frames, and fixed at 0.1 thereafter

- Frame skipping technique: the agent sees and selects actions on every $k$th frame instead of every frame, and its last action is repeated on skipped frames ($k=3$ or 4)
Experiments
Experiments
Experiments

<table>
<thead>
<tr>
<th></th>
<th>B. Rider</th>
<th>Breakout</th>
<th>Enduro</th>
<th>Pong</th>
<th>Q*bert</th>
<th>Seaquest</th>
<th>S. Invaders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>354</td>
<td>1.2</td>
<td>0</td>
<td>−20.4</td>
<td>157</td>
<td>110</td>
<td>179</td>
</tr>
<tr>
<td>Contingency [4]</td>
<td>1743</td>
<td>6</td>
<td>159</td>
<td>−17</td>
<td>960</td>
<td>723</td>
<td>268</td>
</tr>
<tr>
<td>DQN</td>
<td>4092</td>
<td><strong>168</strong></td>
<td><strong>470</strong></td>
<td><strong>20</strong></td>
<td><strong>1952</strong></td>
<td>1705</td>
<td>581</td>
</tr>
<tr>
<td>Human</td>
<td>7456</td>
<td>31</td>
<td>368</td>
<td>−3</td>
<td>18900</td>
<td>28010</td>
<td>3690</td>
</tr>
<tr>
<td>HNeat Best [8]</td>
<td>3616</td>
<td>52</td>
<td>106</td>
<td>19</td>
<td>1800</td>
<td>920</td>
<td><strong>1720</strong></td>
</tr>
<tr>
<td>HNeat Pixel [8]</td>
<td>1332</td>
<td>4</td>
<td>91</td>
<td>−16</td>
<td>1325</td>
<td>800</td>
<td>1145</td>
</tr>
<tr>
<td>DQN Best</td>
<td><strong>5184</strong></td>
<td><strong>225</strong></td>
<td><strong>661</strong></td>
<td><strong>21</strong></td>
<td><strong>4500</strong></td>
<td><strong>1740</strong></td>
<td><strong>1075</strong></td>
</tr>
</tbody>
</table>