Reinforcement Learning Approaches for Atari Breakout

Vincent Pierre Berges, Reid Pryzant, Priyanka Rao
CS 221 Final Project

Motivation
How do different RL approaches compare in a custom implementation of Atari Breakout?

Model

- **Discrete Feature Set**
  - Game state
  - Ball and paddle location
  - Ball angle
  - Brick indicators

- **Multiple Continuous Feature Sets**
  - Ball and paddle locations and velocities
  - Distance of ball relative to walls, bricks, and paddles
  - Interaction features

- **Pixel Intensities**
  - 3D vector representation of pixel RGB values
  - Used with a 5-layer neural network

Algorithms

<table>
<thead>
<tr>
<th>Baseline</th>
<th>SARSA(λ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follows a random policy</td>
<td>Maps (s, a) to Q values of current policy</td>
</tr>
<tr>
<td></td>
<td>Combine past rewards, more recent = more important</td>
</tr>
<tr>
<td></td>
<td>Maintain an eligibility trace to assign blame to parameters</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Q-learning with Replay Memory</td>
<td>Maps (s, a) to Q-values of optimal policy</td>
</tr>
<tr>
<td></td>
<td>Estimate Q(s,a) with linear and neural network function approximators</td>
</tr>
<tr>
<td></td>
<td>Bootstrap estimate of future value by sampling from experience</td>
</tr>
<tr>
<td></td>
<td>Cache parameters of target function for stable updates</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy Gradients</td>
<td>Directly learn parameters ( \theta ) of a policy ( \pi_{\theta} ) (vs ε-greedy)</td>
</tr>
<tr>
<td></td>
<td>Use a neural network for the policy, but wait to fill in gradients until eventual reward is received</td>
</tr>
</tbody>
</table>

Analysis

- Almost all algorithms outperformed baseline by 2x
- Q learning w/ out function approximation struggled because large state space was inadequately explored
- Replay memory added no benefit – for Breakout, correlations between adjacent game states sometimes help agent performance. Delays between actions (ex. returning a ball) correlate to delays in rewards (ex. breaking a brick).
- Agents leveraging nonlinear policy & value networks generally underperformed.
- Neural network did not help despite hyperparameter tuning and different network structures – too little info captured in feature sets
- Future work: investigate each model (especially the more opaque ones) to understand their performance in Breakout vs other games

Results

Cumulative points over multiple runs of 250 test games (ε = 0) after 2000 training games (ε = 0.5). 3 points are awarded per broken brick and 1000 points for winning. Experiments were conducted with \( \gamma = 0.993, \eta = (i / x), |\text{memory}| = 5000, \text{update cycle} = 750, \text{eligibility trace threshold} = 0.1, \text{trace decay} = 0.98. \) The SARSA(λ) agent won 46 games.

Challenges

- Featurizing a huge state space
- Delayed rewards
- Exploration vs exploitation
- Determining relevance of hyperparameters
- Learning from losing vs from hitting bricks
- Highly correlated states
- Experience replay vs SARSA(λ)
- Training neural network

Contact Information
rpryzant@stanford.edu, vberges@stanford.edu, prao96@stanford.edu

References