Human-level control through deep reinforcement learning

Mnih et al.

Presentation by Maximilian Baronig
Agenda

• Introduction
• Task & Challenges
• Input
• Architecture
• Output
• Training & Evaluation
• Results
• Learning Algorithm
• Methods & Tricks
Task

Play games on Atari 2600

- 49 games
- Find general algorithm
- Played in an emulator
- Score compared with human player

26.11.2019
Maximilian Baronig
Challenge

All games differ in
• features
• objectives
• strategies

But the authors wanted to use the same
• algorithm
• hyperparameters
Input

Raw Image
210 x 160 RGB image

Preprocessing
- Extract luminance channel
- Scale & crop to 84 x 84
- Stack 4 frames in a row

Preprocessed Input
84 x 84 x 4

No feature extraction
→ Input considered as raw
Architecture

• DQN (Deep Q-Network)
• Approximate $Q^*(s, a) \rightarrow$ optimal action-value function
• Deep Convolutional Network
DQN

3 convolutional layers
• 32 filters of 8 x 8 stride 4 rectifier
• 64 filters of 4 x 4 stride 2 rectifier
• 64 filters of 3 x 3 stride 1 rectifier

1 fully connected layer
• 512 rectifier units

1 output layer
• 4 to 18 outputs, fully connected
Output

• Estimated Q-value for each action
• $\epsilon$-greedy policy

$\text{perform action} \begin{cases} \text{with probability } 1/\epsilon \text{ randomly} \\ \text{with highest value } Q(s, a) \text{ otherwise} \end{cases}$

$Q(s, a)$ action value function of state $s$ and action $a$
Training & Evaluation

• Training
  • DQN
    played each game for 50 mio. frames (~ 38 days of experience)
  • Human
    2h practice

• Evaluation
  • DQN
    Each game 30 times for 5 minutes
  • Human
    Each game 20 times for max. 5 minutes

• No audio
• Average score of all episodes
Results

• Scale
  0% → random agent
  100% → human agent

• DQN performed above 75% in 29 of 49 games!
• Outperformed best previous reinforcement learner on 43 games
Best Results

100% = human level
0% = random agent
Medium Results

At human-level or above

Below human-level

100% = human level
0% = random agent

- Space Invaders: 121%
- Beam Rider: 119%
- Tutankham: 112%
- Kung-Fu Master: 102%
- Freeway: 102%
- Time Pilot: 100%
- Enduro: 97%
- Fishing Derby: 93%
- Up and Down: 92%
- Ice Hockey: 79%
- Q*bert: 78%
- H.E.R.O.: 76%
- Asterix: 69%
- Battle Zone: 67%
- Wizard of Wor: 67%
- Chopper Command: 64%
- Centipede: 62%
- Bank Heist: 57%
- River Raid: 57%
- Zaxxon: 54%
- Amidar: 43%
- Alien: 42%
Poor Results

- Alien: 42%
- Venture: 32%
- Seaquest: 25%
- Double Dunk: 17%
- Bowling: 14%
- Ms. Pac-Man: 13%
- Asteroids: 7%
- Frostbite: 6%
- Gravitar: 5%
- Private Eye: 2%
- Montezuma's Revenge: 0%

100% = human level
0% = random agent

Maximilian Baronig
Ms. Pacman

• Very poor performance of 13 %
• Problem?
• Planning
• Randomness
• Observability
Montezuma’s Revenge

• 0% → just like random agent
• Again planning
• Artifacts from real life
• Reward only after getting key
Recap: TD-learning

• TD(0) – learning update

\[ V(S_t) \leftarrow V(S_t) + \alpha \left[ R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right] \]

• TD(0) for control → SARSA

\[ Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right] \]

Source: Sutton & Barto 2018
Optimal Action Value Function

\[ Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi] \]

The optimal action-value function is the expected discounted sum of the maximum achievable reward.

Source: Mnih et al. 2015
Q-Learning

From Bellman equation we know that

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

Source: Mnih et al. 2015
Q-Learning update step

\[ Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \]

Off-policy

Even if we follow A, B, C
\[ \rightarrow \text{A is updated with } r_x + Q(B, z) \]
Q-Learning with gradient descent

Substitute optimal target

\[ r + \gamma \max_{a'} Q^*(s',a') \]

with approximation from DQN

\[ r + \gamma \max_{a'} Q(s',a'; \theta^-_i) \]

and define Loss function for iteration \( i \)

\[
L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[ \left( r + \gamma \max_{a'} Q(s',a'; \theta_i^-) - Q(s,a; \theta_i) \right)^2 \right]
\]
Q-Learning with gradient descent

Loss function

\[ L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[ \left( r + \gamma \max_{a'} Q(s',a'; \theta_i) - Q(s,a; \theta_i) \right)^2 \right] \]

Gradient w.r.t. \( \Theta \)

\[ \nabla_{\theta_i} L(\theta_i) = \mathbb{E}_{s,a,r,s'} \left[ \left( r + \gamma \max_{a'} Q(s',a'; \theta_i) - Q(s,a; \theta_i) \right) \nabla_{\theta_i} Q(s,a; \theta_i) \right] \]
Q-Learning with gradient descent

• For $s$ we use sequences of

\[ s_t = x_1, a_1, x_2, \ldots, a_{t-1}, x_t \]

\[ x_n \quad \text{state at timestep } n \]
\[ a_n \quad \text{action performed at timestep } n \]

• Single images are ambiguous
Q-Learning with gradient descent

Gradient descent step for timestep $t$

$$w_{t+1} = w_t + \alpha \left[ R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, w_t) - \hat{q}(S_t, A_t, w_t) \right] \nabla \hat{q}(S_t, A_t, w_t)$$
Methods & Tricks

• Separate Network for generating targets

\[ \nabla_{\theta_i} L(\theta_i) = \mathbb{E}_{s,a,r,s'} \left[ \left( r + \gamma \max_{a'} Q(s',a'; \theta_i^-) - Q(s,a; \theta_i) \right) \nabla_{\theta_i} Q(s,a; \theta_i) \right] \]

• Train only network A → leave B as it is
• Each C iterations copy network A to B
→ Semi-gradient method!

Maximilian Baronig
Methods & Tricks – Semi-gradient

• Problem with bootstrapping
• Oscillations or divergence
• Add delay between update and effect
• Improve stability
Collect many experiences at different time steps in data pool

\[ e_t = (s_t, a_t, r_t, s_{t+1}) \]

- Stochastic Gradient Descent
  - Choose \( e \) from pool randomly
- Remove correlations
- Avoid loops
- Improve stability

Requires off-policy algorithm!
Methods & Tricks – Reward Clipping

• Different score scales for different games
• Clip each increase of score to +1
• Each decrease to -1
• 0 if no change
Methods & Tricks – Error Clipping

Error value of

\[ r + \gamma \max_{a'} Q(s', a'; \theta_i) - Q(s, a; \theta_i) \]

clicked to interval [-1,1]

Absolute value loss function for error outside interval

→ Further stability improvement
Conclusion

• Considered a breakthrough in reinforcement learning
• One algorithm can solve different tasks
• Learn feature extraction
Thank you!
Questions?
Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory \( D \) to capacity \( N \)
Initialize action-value function \( Q \) with random weights \( \theta \)
Initialize target action-value function \( \hat{Q} \) with weights \( \theta^- = \theta \)

For episode = 1, \( M \) do
  Initialize sequence \( s_1 = \{x_1\} \) and preprocessed sequence \( \phi_1 = \phi(s_1) \)
  For \( t = 1, T \) do
    With probability \( \epsilon \) select a random action \( a_t \),
    otherwise select \( a_t = \text{argmax}_a Q(\phi(s_t), a; \theta) \)
    Execute action \( a_t \) in emulator and observe reward \( r_t \) and image \( x_{t+1} \)
    Set \( s_{t+1} = s_t, a_t, x_{t+1} \) and preprocessor \( \phi_{t+1} = \phi(s_{t+1}) \)
    Store transition \( (\phi_t, a_t, r_t, \phi_{t+1}) \) in \( D \)
    Sample random minibatch of transitions \( (\phi_j, a_j, r_j, \phi_{j+1}) \) from \( D \)
    Set \( y_j = \begin{cases} 
      r_j & \text{if episode terminates at step } j+1 \\
      r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise}
    \end{cases} \)
    Perform a gradient descent step on \( (y_j - Q(\phi_j, a_j; \theta))^2 \) with respect to the network parameters \( \theta \)
    Every \( C \) steps reset \( \hat{Q} = Q \)
  End For
End For
## Parameters

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minibatch size</td>
<td>32</td>
<td>Number of training cases over which each stochastic gradient descent (SGD) update is computed.</td>
</tr>
<tr>
<td>replay memory size</td>
<td>1000000</td>
<td>SGD updates are sampled from this number of most recent frames.</td>
</tr>
<tr>
<td>agent history length</td>
<td>4</td>
<td>The number of most recent frames experienced by the agent that are given as input to the Q network.</td>
</tr>
<tr>
<td>target network update frequency</td>
<td>10000</td>
<td>The frequency (measured in the number of parameter updates) with which the target network is updated (this corresponds to the parameter $C$ from Algorithm 1).</td>
</tr>
<tr>
<td>discount factor</td>
<td>0.99</td>
<td>Discount factor gamma used in the Q-learning update.</td>
</tr>
<tr>
<td>action repeat</td>
<td>4</td>
<td>Repeat each action selected by the agent this many times. Using a value of 4 results in the agent seeing only every 4th input frame.</td>
</tr>
<tr>
<td>update frequency</td>
<td>4</td>
<td>The number of actions selected by the agent between successive SGD updates. Using a value of 4 results in the agent selecting 4 actions between each pair of successive updates.</td>
</tr>
<tr>
<td>learning rate</td>
<td>0.00025</td>
<td>The learning rate used by RMSProp.</td>
</tr>
<tr>
<td>gradient momentum</td>
<td>0.95</td>
<td>Gradient momentum used by RMSProp.</td>
</tr>
<tr>
<td>squared gradient momentum</td>
<td>0.95</td>
<td>Squared gradient (denominator) momentum used by RMSProp.</td>
</tr>
<tr>
<td>min squared gradient</td>
<td>0.01</td>
<td>Constant added to the squared gradient in the denominator of the RMSProp update.</td>
</tr>
<tr>
<td>initial exploration</td>
<td>1</td>
<td>Initial value of $\epsilon$ in $\epsilon$-greedy exploration.</td>
</tr>
<tr>
<td>final exploration</td>
<td>0.1</td>
<td>Final value of $\epsilon$ in $\epsilon$-greedy exploration.</td>
</tr>
<tr>
<td>final exploration frame</td>
<td>1000000</td>
<td>The number of frames over which the initial value of $\epsilon$ is linearly annealed to its final value.</td>
</tr>
<tr>
<td>replay start size</td>
<td>50000</td>
<td>A uniform random policy is run for this number of frames before learning starts and the resulting experience is used to populate the replay memory.</td>
</tr>
<tr>
<td>no-op max</td>
<td>30</td>
<td>Maximum number of “do nothing” actions to be performed by the agent at the start of an episode.</td>
</tr>
</tbody>
</table>
## Comparison Replay, Target Q

<table>
<thead>
<tr>
<th>Game</th>
<th>With replay, with target Q</th>
<th>With replay, without target Q</th>
<th>Without replay, with target Q</th>
<th>Without replay, without target Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakout</td>
<td>316.8</td>
<td>240.7</td>
<td>10.2</td>
<td>3.2</td>
</tr>
<tr>
<td>Enduro</td>
<td>1006.3</td>
<td>831.4</td>
<td>141.9</td>
<td>29.1</td>
</tr>
<tr>
<td>River Raid</td>
<td>7446.6</td>
<td>4102.8</td>
<td>2867.7</td>
<td>1453.0</td>
</tr>
<tr>
<td>Seaquest</td>
<td>2894.4</td>
<td>822.6</td>
<td>1003.0</td>
<td>275.8</td>
</tr>
<tr>
<td>Space Invaders</td>
<td>1088.9</td>
<td>826.3</td>
<td>373.2</td>
<td>302.0</td>
</tr>
</tbody>
</table>