# Deep Q-learning

Соколов Роман

Национальный Исследовательский Университет Высшая Школа Экономики

30 ноября 2018 г.

# Q-learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\big)$$

# Q-learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\big)$$

- Используем $\epsilon$-greedy стратегию
-
-
-
-

# Q-learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\big)$$

- Используем $\epsilon$-greedy стратегию
- $Q(s, a) \approx Q(s, a, \theta)$ - Например нейросеть
-
-
-

# Q-learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\big)$$

- Используем $\epsilon$-greedy стратегию
- $Q(s, a) \approx Q(s, a, \theta)$ - Например нейросеть
- Нужна функция потерь:
  $L_i[\theta_i] = E_{s,a\sim p}\left[(y_i - Q(s, a; \theta_i))^2\right]$
  $y_i = E_{s'\sim\mathcal{E}}\left[r + \gamma max_{a'} Q(s', a', \theta_{i-1})|s, a\right]$
-
-

# Q-learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\big)$$

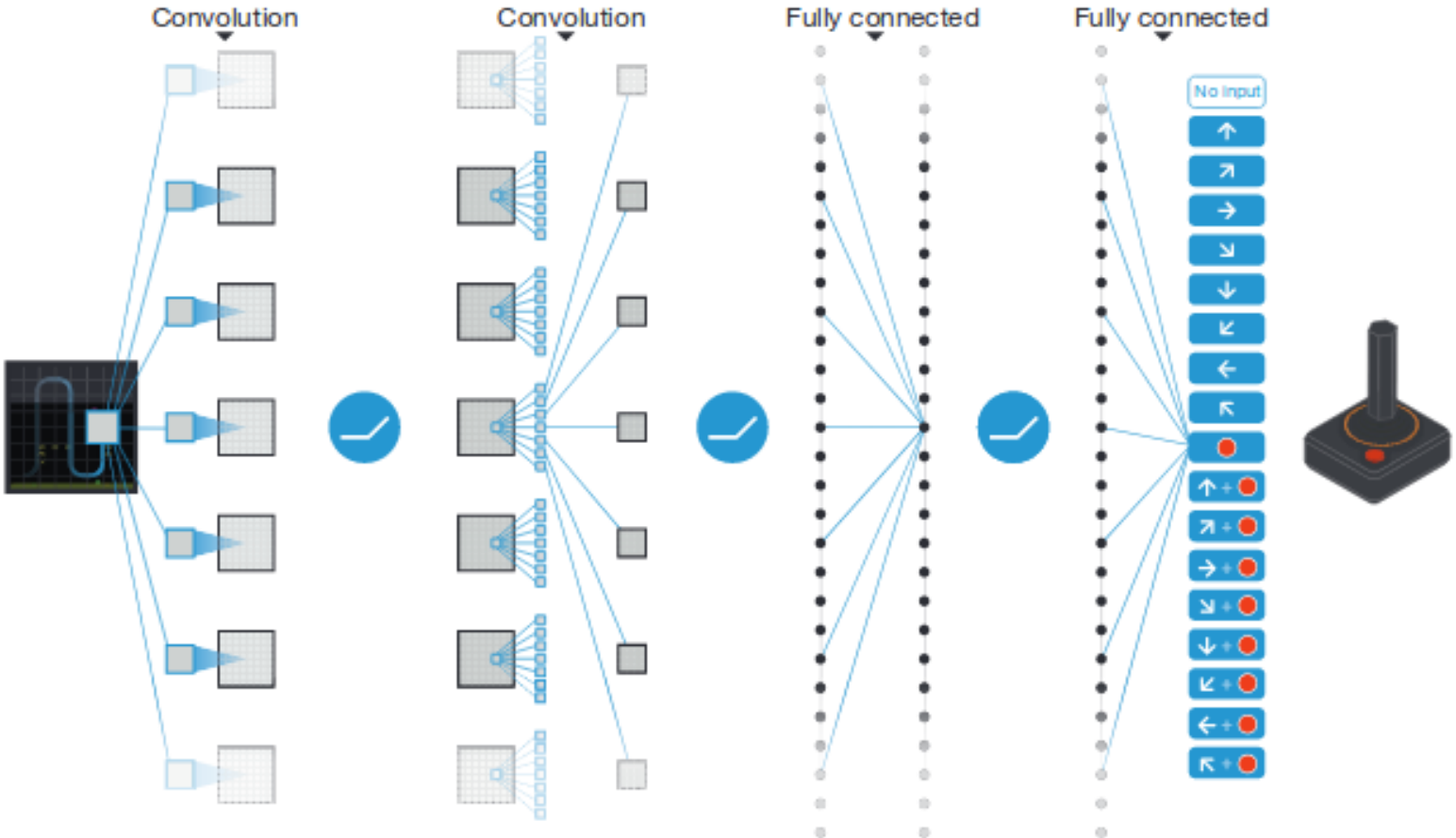
- Используем $\epsilon$-greedy стратегию
- $Q(s, a) \approx Q(s, a, \theta)$ - Например нейросеть
- Нужна функция потерь:
  $L_i[\theta_i] = E_{s,a\sim p}\left[(y_i - Q(s, a; \theta_i))^2\right]$
  $y_i = E_{s'\sim\mathcal{E}}\left[r + \gamma max_{a'} Q(s', a', \theta_{i-1})|s, a\right]$
- Состояние получаем из изображения
-

# Q-learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)\big)$$

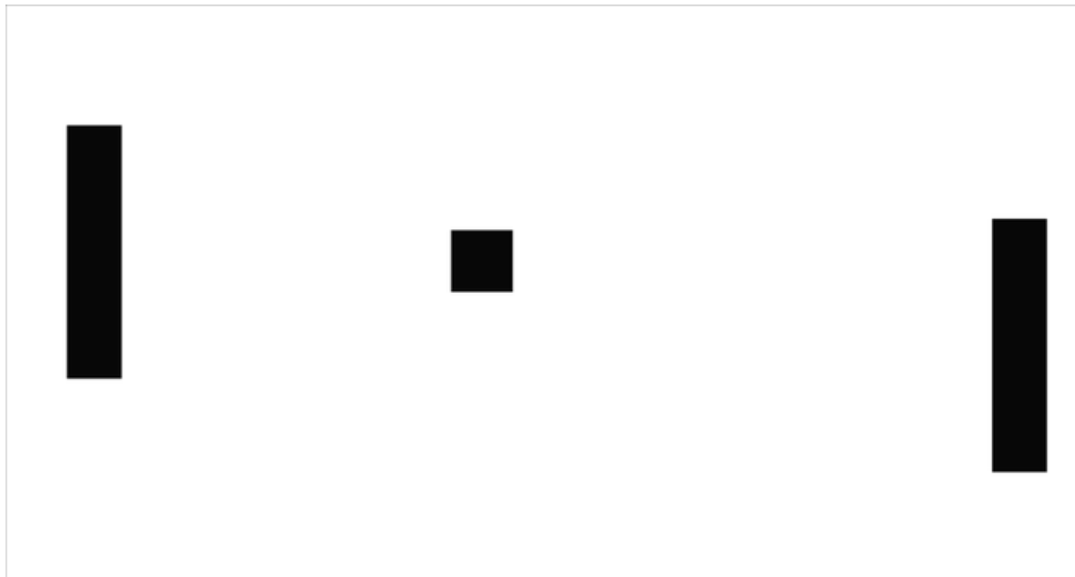
- Используем $\epsilon$-greedy стратегию
- $Q(s, a) \approx Q(s, a, \theta)$ - Например нейросеть
- Нужна функция потерь:
  $L_i[\theta_i] = E_{s,a\sim p}\left[(y_i - Q(s, a; \theta_i))^2\right]$
  $y_i = E_{s'\sim\mathcal{E}}\left[r + \gamma max_{a'} Q(s', a', \theta_{i-1})|s, a\right]$
- Состояние получаем из изображения
- Выход сети - вектор оценок для всех действий

# Архитектура

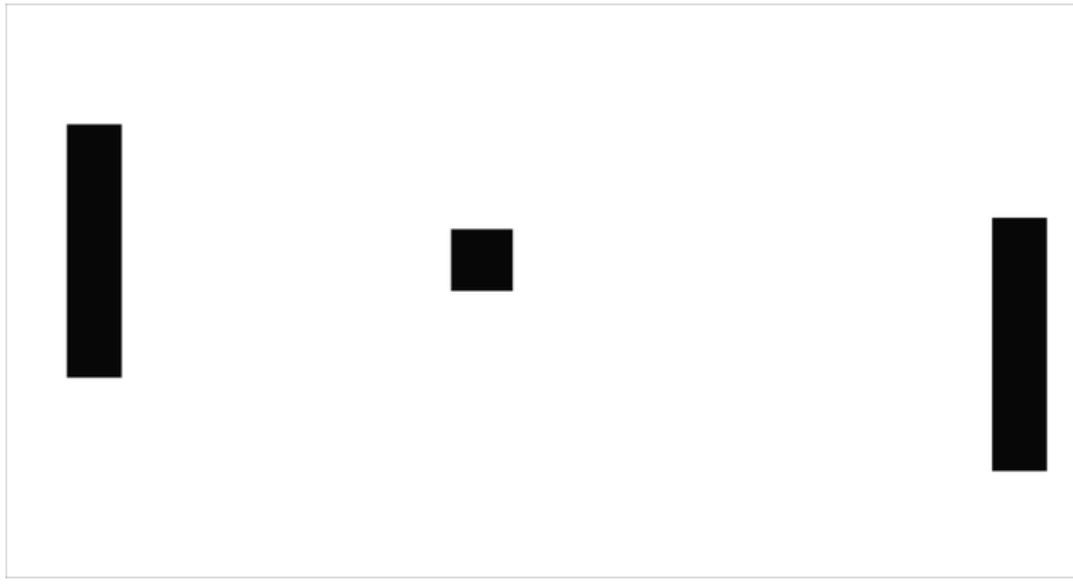


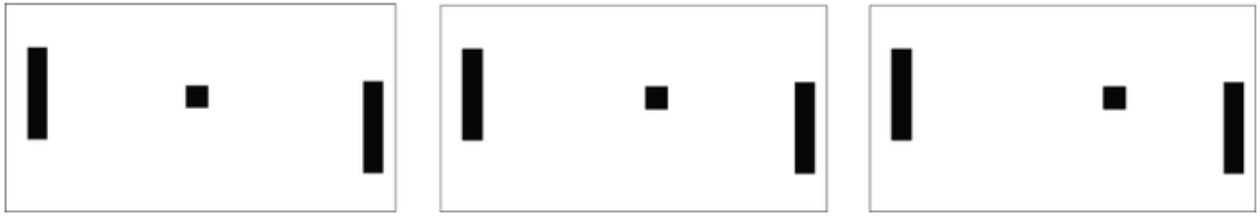

[1]

# Temporal limitation



# Temporal limitation





[2]

# Experience replay


# Experience replay



Запоминаем блоки $(s, a, r, s')$ и случайно семплируем батч для обучения из них.

[2]

# Prioritized experience replay

- Некоторые состояния важнее других
-
-
-
-

# Prioritized experience replay

- Некоторые состояния важнее других
- Утверждается что такие состояния можно найти по большому значению Temporal Difference Error:
  $TDE_i = |(r_{t+1} + \gamma \max_b Q(s_{t+1}, b)) - Q(s_t, a_t)|$,
  $P_i = TDE_i + \epsilon$
- 
- 
- 

# Prioritized experience replay

- Некоторые состояния важнее других
- Утверждается что такие состояния можно найти по большому значению Temporal Difference Error:
  $TDE_i = |(r_{t+1} + \gamma \max_b Q(s_{t+1}, b)) - Q(s_t, a_t)|$,
  $P_i = TDE_i + \epsilon$
- Можно сэмплировать с вероятностями $p_i = \frac{P_i^a}{\sum P_j^a}$, где $a \in [0, 1]$ регулирует баланс между равномерным и приоритетным сэмплированием.
-
-

# Prioritized experience replay

- Некоторые состояния важнее других
- Утверждается что такие состояния можно найти по большому значению Temporal Difference Error:
  $TDE_i = |(r_{t+1} + \gamma \max_b Q(s_{t+1}, b)) - Q(s_t, a_t)|$,
  $P_i = TDE_i + \epsilon$
- Можно сэмплировать с вероятностями $p_i = \frac{P_i^a}{\sum P_j^a}$, где $a \in [0, 1]$ регулирует баланс между равномерным и приоритетным сэмплированием.
- Можно заменить $P_i$ на $\frac{1}{rank(i)}$ где $rank(i)$ порядок блока в памяти отсортированной по $TDE$
-

# Prioritized experience replay

- Некоторые состояния важнее других
- Утверждается что такие состояния можно найти по большому значению Temporal Difference Error:
  $TDE_i = |(r_{t+1} + \gamma \max_b Q(s_{t+1}, b)) - Q(s_t, a_t)|$,
  $P_i = TDE_i + \epsilon$
- Можно сэмплировать с вероятностями $p_i = \frac{P_i^a}{\sum P_j^a}$, где $a \in [0, 1]$ регулирует баланс между равномерным и приоритетным сэмплированием.
- Можно заменить $P_i$ на $\frac{1}{rank(i)}$ где $rank(i)$ порядок блока в памяти отсортированной по $TDE$
- Поправим смещение на объектах выборки домножив на $(\frac{1}{n} \cdot \frac{1}{p_i})^b$

# Double DQN

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q^-(s_{t+1}, b) - Q(s_t, a_t)\big)$$

- Фиксируем целевую сеть
-
-

# Double DQN

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q^-(s_{t+1}, b) - Q(s_t, a_t)\big)$$

- Фиксируем целевую сеть
- Мы не уверены что максимальное значение функции $Q^-$ даст нам лучшее действие.
-

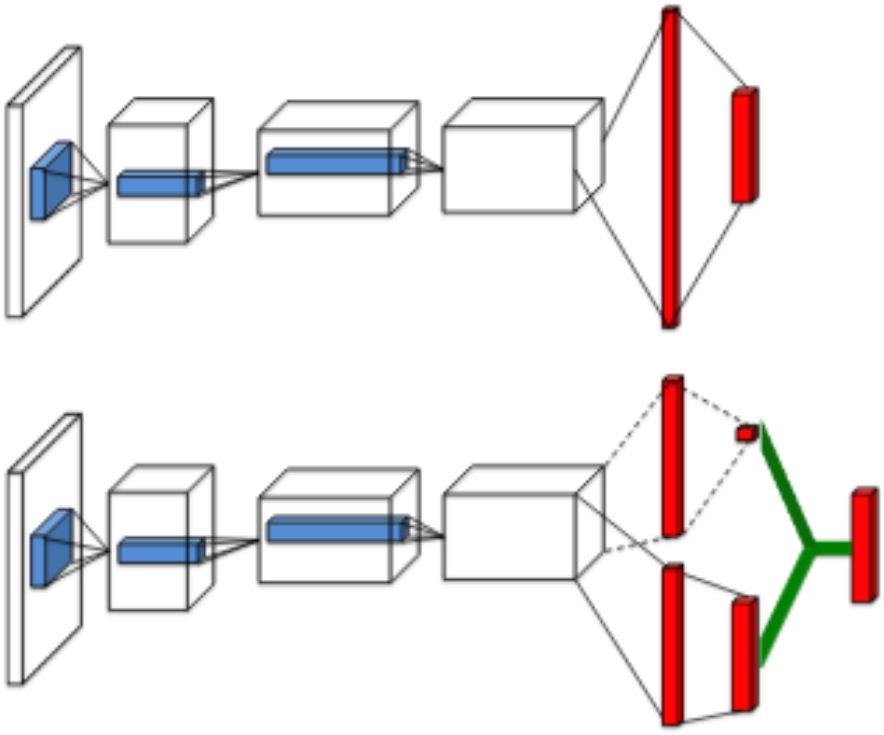
# Double DQN

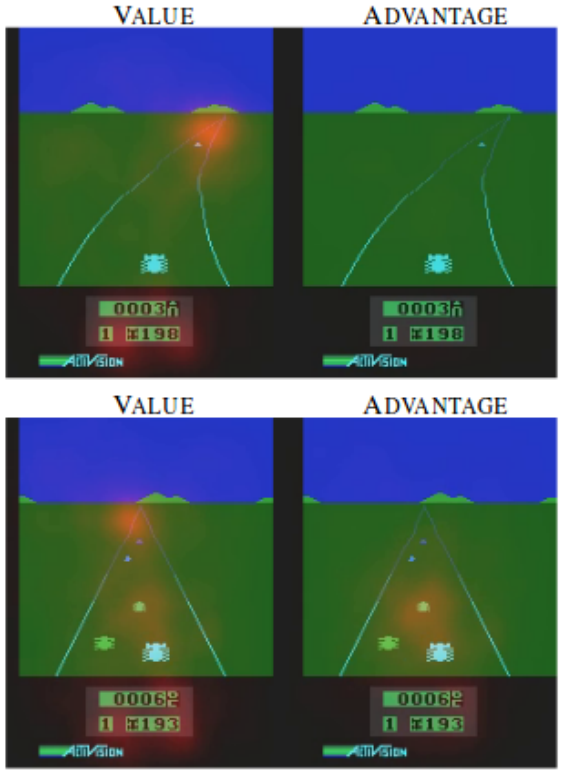
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \max_b Q^-(s_{t+1}, b) - Q(s_t, a_t)\big)$$

- Фиксируем целевую сеть
- Мы не уверены что максимальное значение функции $Q^-$ даст нам лучшее действие.
- Вместо $Q^-$ для выбора действия будем использовать $Q$ так как она обучалась дольше, но для подсчёта значения будем использовать $Q^-$ сеть чтобы сохранить хорошую сходимость.

$$\max_b Q^-(s_{t+1}, b) \to Q^-(s_{t+1}, argmax_b Q(s_{t+1}, b))$$

# Dueling DQN






[3]

# Dueling DQN

$$Q(s, a) = V(s) + A(s, a)$$

# Dueling DQN

$$\cancel{Q(s,a) = V(s) + A(s,a)}$$

$$Q(s,a) = V(s) + (A(s,a) - \max_{a'} A(s,a'))$$

$$Q(s,a) = V(s) + (A(s,a) - \frac{1}{A}\sum_{a'} A(s,a'))$$

# Actor Critic method

$$\Delta\theta = \alpha \cdot \nabla_\theta \cdot (log\pi(S_t, A_t, \theta)) \cdot \cancel{R(t)}$$

$$\Delta\theta = \alpha \cdot \nabla_\theta \cdot (log\pi(S_t, A_t, \theta)) \cdot Q_w(S_t, A_t)$$

$$\Delta w = \beta(R(s, a) + \gamma Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t))\nabla_w Q_w(s_t, a_t)$$

# Заключение

- Применение Q-learning с нейронными сетями
- Архитектура
- Temporal limitation
- Experience replay
- Double DQN
- Dueling DQN
- Actor Critic method

# Источники

[1] Human-level control through deep reinforcement learning
[2] An introduction to Deep Q-Learning: let's play Doom
[3] Dueling Network Architectures for Deep Reinforcement Learning
[4] An intro to Advantage Actor Critic methods: let's play Sonic the Hedgehog!
[5] Playing Atari with Deep Reinforcement Learning
[6] Prioritized experience replay