Глубокое обучение с подкреплением играх Atari

Александр Панов

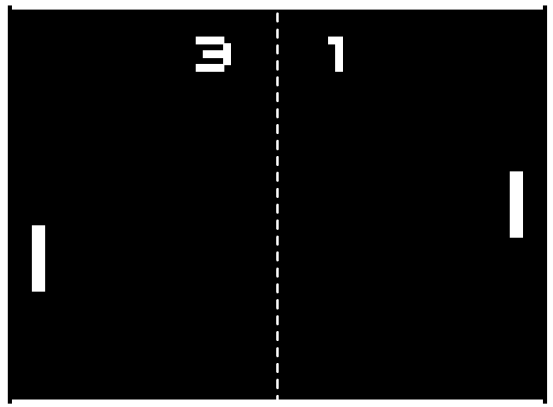
ИСА РАН

30 июня 2016 г.

Постановка задачи

- Внешняя среда (стохастическая) $\mathcal{E}$
- Дискретное время $t \in \mathbb{N}$
- Действия $a_t \in \mathcal{A} = \{1, .., K\}$
- Наблюдаемое изображение $x_t \in \mathbb{R}^d$
- Вознаграждение (изменение счета) r_t
- Описание текущего состояние через последовательность действий и наблюдений:

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



- Цель агента - взаимодействовать со средой, выбирая действия таким образом, чтобы максимизировать будущее вознаграждение.

Mnih, Volodymyr et al. "Playing Atari with Deep Reinforcement Learning". In: *arXiv preprint arXiv: ...* (2013), pp. 1–9. arXiv: 1312.5602.

Q-обучение

Будущее вознаграждение спадает со временем

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'},$$

T - время окончания игры.

Оптимальное значение функции оценки действий

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

π - стратегия выбора действия.

Уравнение Беллмана

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

откуда следует итеративный процесс вычисления оптимального значения $Q_i \to Q^*$:

$$Q_{i+1}(s,a) = \mathbb{E}\Big[r + \gamma \max_{a'} Q_i(s',a') \Big| s, a\Big].$$

Q-обучение

Для придания обобщающей силы используют аппроксимации

$$Q(s, a; \theta) \approx Q^*(s, a).$$

При использовании в качестве аппроксиматора нейронной сети получаем Q-сеть с функцией потерь

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

где $\rho(s, a)$ - распределение вероятности по последовательностям s и действиям a, а

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

Соответствующий градиент для SGD

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

Q-обучение, переигровка (experience reply)

Model-free, off-policy: $a = \max_a Q(s, a; \theta)$.

Каждый момент сохраняется опыт (прецедент) $e_t = (s_t, a_t, r_t, s_{t+1})$ в память переигровок $\mathcal{D} = e_1, \ldots, e_N$.

Архитектура сети:

- Входное цветное изображение 210x160 ужимается до серых тонов 84x84 (функция ϕ).
- Функция ϕ применяется к последним 4 кадрам (4 изображения).
- Первый скрытый слой - 16 сверточных фильтров 8x8 с шагом 4.
- Второй скрытый слой - 32 сверточных фильтра 4x4 с шагом 2.
- Последний скрытый слой - полносвязный (256 rectifier units).
- Выходной слой - полносвязный линейный на выход для каждого возможного действия (4–18).

DQN с переигровкой

1: Initialize replay memory $\mathcal{D}$ to capacity N
2: Initialize action-value function Q with random weights
3: **for all** episode $= 1, M$ **do**
4: Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
5: **for all** $t = 1, T$ **do**
6: With probability ϵ select a random action a_t
7: otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
8: Execute action a_t in emulator and observe reward r_t and image x_{t+1}
9: Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
10: Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
11: Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
12: Set

$$y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$$

13: Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$
14: **end for**
15: **end for**