Q-learning

Васильевых Павел, БПМИ213

План

1. Условие задачи.
2. Алгоритм.
3. Улучшения алгоритма.

Какую задачу решает?

s — некоторое состояние

a — некоторое действие

$\mathrm{r}(s, a)$ — награда за заданное действие в заданном состоянии

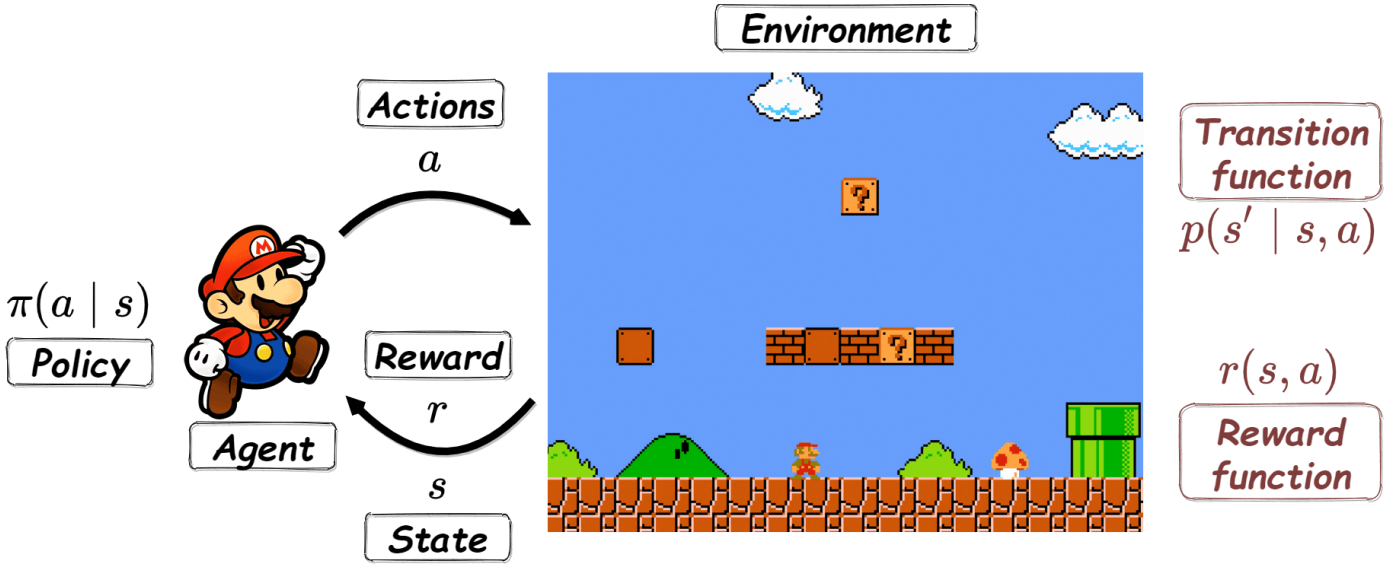
$\pi(a|s)$ — распределение действий агента в заданном состоянии (политика)

$\mathcal{T}$ — траектория (последовательность из реализованных состояний, действий и наград)

γ — коэффициент убывания награды

$$\mathrm{Q}(s, a) = \max_{\pi} \mathop{\mathbb{E}}_{\mathcal{T} \sim \pi | s_0 = s, a_0 = a} \sum_{t \geq 0} \gamma^t \mathrm{r}(s_t, a_t)$$

Q-learning подбирает $\mathrm{Q}(s, a)$, по которому можно определить $\pi(a|s)$.



Как упростить задачу?

Во-первых, можно вывести уравнение оптимальности Беллмана:

$$\mathrm{Q}(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathrm{p}(s'|s,a)} \max_{a'} \mathrm{Q}(s',a')$$

Во-вторых, можно простым образом выразить оптимальную политику через Q:

$$\pi(s) = \operatorname*{argmax}_{a} \mathrm{Q}(s,a)$$

Как сделать задачу более практической?

1. Как-то инициализировать $\mathrm{Q}(s, a)$.

2. Заметить, что можно забыть про $\mathrm{p}(s'|s, a)$ и моделировать переходы экспериментально, совершая действия $\underset{a}{\operatorname{argmax}}\,\mathrm{Q}(s, a)$. В результате можно получить переходы вида (s, a, r, s').

3. Вывести из уравнения Беллмана функцию перехода:

$$\mathrm{Q}(s, a) \leftarrow r(s, a) + \gamma \max_{a'} \mathrm{Q}(s', a')$$

4. Сгладить функцию перехода:

$$\mathrm{Q}(s, a) \leftarrow (1 - \alpha) \cdot \mathrm{Q}(s, a) + \alpha \cdot \left(r(s, a) + \gamma \max_{a'} \mathrm{Q}(s', a')\right)$$

Сходимость

Пусть $\mathrm{Q}(s,a)$ и $r(s,a)$ инициализированы, $\gamma \in [0,1)$. Доказать, что процесс переходов $\mathrm{Q}(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathrm{p}(s'|s,a)} \max_{a'} \mathrm{Q}(s',a')$ по всему множеству состояний и действий сходится.

Обозначим функционал перехода за R. Рассмотрим неизвестную истинную функцию $Q^*(s,a)$, для неё верно $R \circ Q^* = Q^*$. Заметим, что $\|R \circ Q - Q^*\|_\infty = \|R \circ Q - R \circ Q^*\|_\infty = \gamma \cdot \left\| \mathop{\mathbb{E}}_{s' \sim \mathrm{p}(s'|s,a)} \left[\max_{a'} \mathrm{Q}(s',a') - \max_{a'} Q^*(s',a') \right] \right\|_\infty \leq \gamma \cdot \left\| \mathop{\mathbb{E}}_{s' \sim \mathrm{p}(s'|s,a)} [\mathrm{Q}(s',a(s')) - Q^*(s',a(s'))] \right\|_\infty \leq \gamma \cdot \|Q - Q^*\|_\infty$, где $\mathrm{a}(s') = \operatorname{argmax}_a \mathrm{Q}(s',a)$. Значит, $0 \leq \lim_{n\to\infty} \|R^n \circ Q - Q^*\|_\infty \leq \lim_{n\to\infty} (\gamma^n \cdot \|Q - Q^*\|_\infty) = 0.$

Дилемма exploration-exploitation

Из-за детерминированности $\underset{a}{\operatorname{argmax}}\,\mathrm{Q}(s,a)$ вероятность попасть в некоторые состояния может быть равна 0, нужно добавить случайность, чтобы агент исследовал среду. Но она должна быть незначительной, чтобы агент использовал накопленный опыт.

$$\pi(s) = \begin{cases} \text{случайное действие, вероятность } \varepsilon \\ \underset{a}{\operatorname{argmax}}\,\mathrm{Q}(s,a)\text{, вероятность } 1-\varepsilon \end{cases}$$

Что получилось?

Алгоритм, подходящий для $\mathrm{Q}(s, a)$, которые можно задать таблицей:

1. Инициализировать $\mathrm{Q}(s, a)$.
2. Получить из среды начальное состояние s_0.
3. Повторить для $k = 0 \ldots n$:

    1) выбрать случайное действие a_k с вероятностью ε, иначе $a_k = \underset{a}{\operatorname{argmax}} \mathrm{Q}(s_k, a)$;

    2) получить награду r_k и следующее состояние s_{k+1};

    3) обновить $\mathrm{Q}(s_k, a_k) \leftarrow (1 - \alpha) \cdot \mathrm{Q}(s_k, a_k) + \alpha \cdot \left(r_k + \gamma \max_{a'} \mathrm{Q}(s_{k+1}, a')\right)$.

Таргет-сеть

В случае, когда состояний много, можно задать $\mathrm{Q}(s, a)$ нейросетью с обновляемыми таргетами $r_k + \gamma \max\limits_{a'} \mathrm{Q}(s_{k+1}, a')$. Но в такой модификации у алгоритма возникает ещё один недостаток — нестабильность сети из-за зависимости таргетов от предыдущих выходов.

Чтобы сделать сеть стабильной, нужна таргет-сеть. Создадим копию имеющейся сети и будем использовать её для обновления таргетов, заменяя её веса параметрами основной сети раз в m шагов обучения.

Experience replay

Для качественного обучения нейросети обычно нужно усреднять градиент по батчу. Если мы будем следить за последовательными действиями одного агента, сеть будет забывать предыдущий опыт.

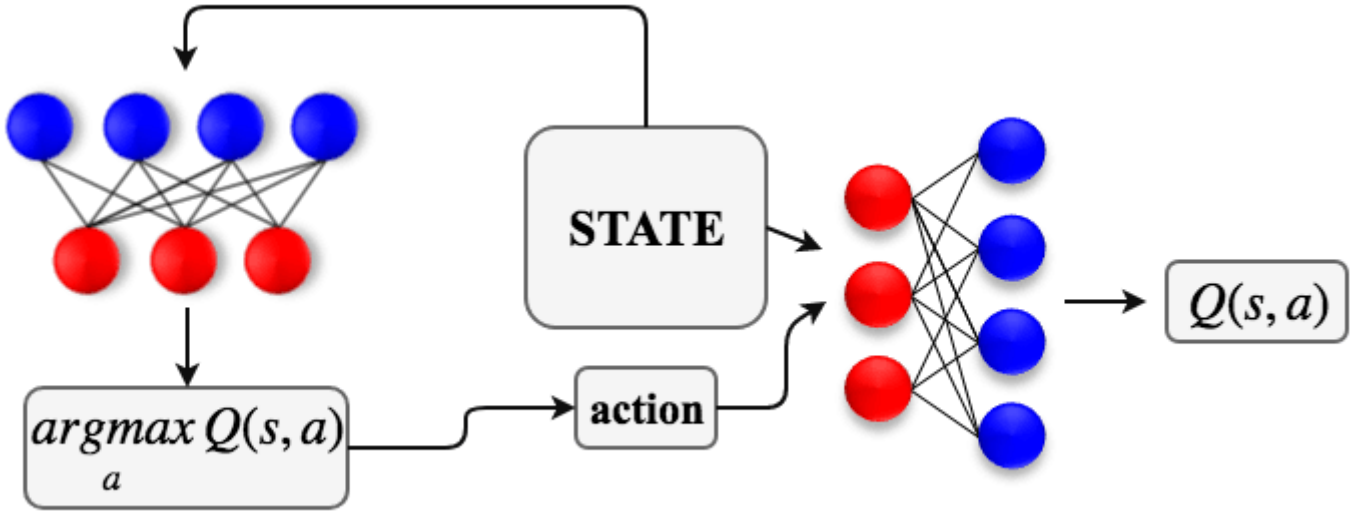
Решения:

1) Несколько агентов одновременно;

2) experience replay, то есть запись всех переходов в буфер и семплирование случайных батчей из него.

А если действий много?

Сделаем ещё одну нейросеть $\pi(s)$ для приближения $\underset{a}{\operatorname{argmax}}\, \mathrm{Q}(s, a)$. Её можно обучать параллельно с основной: на каждом шаге алгоритма будем делать шаг градиентного подъёма для батча функций $\mathrm{Q}(s, \pi(s))$, где параметры сети $\mathrm{Q}(s, a)$ зафиксированы.



Deep Q-learning (Actor-Critic)

1. Одинаково инициализировать $\mathrm{Q}(s, a)$ и таргет-сеть.
2. Инициализировать $\pi(s)$.
3. Получить из среды начальное состояние s_0.
4. Повторить для $k = 0 \ldots n$:
    1) выбрать случайное действие a_k с вероятностью ε, иначе $a_k = \underset{a}{\operatorname{argmax}}\, \mathrm{Q}(s_k, a)$;
    2) получить награду r_k и следующее состояние s_{k+1};
    3) добавить переход (s_k, a_k, r_k, s_{k+1}) в буфер;
    4) сэмплировать батч из буфера;
    5) сделать шаг градиентного подъёма для $\mathrm{Q}(s, \pi(s))$ по параметрам $\pi(s)$;
    6) посчитать таргеты $\mathrm{r} + \gamma\, \mathrm{Q}(s', \pi(s'))$, используя таргет-сеть;
    7) сделать шаг градиентного спуска для $\mathrm{Q}(s, a)$;
    8) если $\mathrm{k} : m$, обновить веса таргет-сети.

Модификации

DDPG (Deep determistic policy gradient): для Q и π есть по две нейросети (одна меняется на каждом шаге, другая постепенно двигает веса к весам первой), действия генерируются из нормального распределения с центром в значении меняющейся π, а таргеты для обучения Q считаются с помощью двух стабильных сетей. Есть проблема переоценки, решается добавлением второй Q.

SAC (Soft actor-critic): в этом алгоритме случайные (исследовательские) шаги не создаются отдельно или поверх выхода π. Случайность π включается в задачу оптимизации через прибавление энтропии текущей π с константным весом к каждой награде r.

Выводы

Мы рассмотрели алгоритм, для обучения которого можно использовать весь накопленный опыт (off-policy). Очевидно, есть большой простор для улучшений и оптимизаций под конкретные задачи.

Однако мы могли бы подойти к задаче RL и со стороны функции политики. Тогда бы мы попали в область on-policy алгоритмов, которые могут оказаться эффективнее, но это тема следующего семинара.

Ссылки

Статья про Q-learning: https://education.yandex.ru/handbook/ml/article/obuchenie-s-podkrepleniem.

Лекция про DDPG и SAC: https://dzen.ru/video/watch/6521821790b03d45f18118ce.

Применение Deep Q-network: https://deepmind.com/blog/article/Agent57-Outperforming-the-human-Atari-benchmark.