Лекция 3

Обучение с подкреплением

Никита Юдин, iudin.ne@phystech.edu

Московский физико-технический институт
Физтех-школа прикладной математики и информатики

21 февраля 2024

ФПМИ

Что делали?

- Ввели понятие $Q-$ и $V-$ функции, с помощью которых ввели частичный порядок на политиках.
- V-функция — средняя награда по политике π, если агент начинает действовать в момент времени t из состояния s.
- Q-функция то же, что V функция, только теперь из состояния $s_t = s$ обязательно сначала совершается действие $a_t = a$.

Что делали?

- Поняли, что знать Q^π-функцию очень важно, ведь мы можем изменить нашу политику так, что политика не ухудшится, а в лучшем случае станет лучше:

$$\hat{\pi}(s) = \arg\max_a Q^\pi(s, a) \implies \hat{\pi} \geqslant \pi.$$

- Поняли, что, если такое улучшение для всех s политику уже не изменяет, то эта политика оптимальна:

$$\pi^*(s) = \arg\max_a Q^*(s, a).$$

Что делали?

- Найти $Q-$ и $V-$ функции помогают выведенные рекуррентные формулы на эти функции и на их оптимальные аналоги — уравнения Беллмана:

$$V^{\pi}(s) = \mathbb{E}_{\pi(a|s)}\left[Q^{\pi}(s,a)\right];$$
$$Q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)}\left[V^{\pi}(s')\right];$$

$$V^{*}(s) = \max_{a \in A}\left\{Q^{*}(s,a)\right\};$$
$$Q^{*}(s,a) = r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)}\left[V^{*}(s')\right].$$

Что делали?

- Если о **MDP** задаче нам известно $p(s'|s,a)$, $r(s,a)$, а также S и A конечны, то решение уравнения Беллмана для V-функции:
  - Есть СЛАУ, $V^\pi = U + \gamma P V^\pi =: F(V^\pi)$.
  - Решается методом простой итерации, так как оператор Беллмана $F(V^\pi)$ — сжимающий с коэффициентом γ, Такой метод решения называется *Policy Evaluation*.
- Если о **MDP** задаче нам известно $p(s'|s,a)$, $r(s,a)$, а также S и A конечны, то решение уравнения Беллмана для оптимальной V^*-функции:
  - Сводится к задаче линейного программирования, откуда будет следовать существование оптимальной *детерминированной политики*.

Policy Evaluation / Policy Iteration / Value Iteration

- *Policy Evaluation* — оценивает V-функцию для текущей политики. Останавливается, когда V-функция меняется мало.
- *Policy Iteration* — оценивает V-функцию для фиксированной политики (с помощью Policy Evaluation), а затем улучшает политику. Останавливается, когда политика перестает меняться.
- *Value Iteration* — частный случай Policy Iteration: шаг оценки V-функции и изменения политики «схлопнут» в один без непосредственного вычисления политики. Получение самой политики — отдельный шаг.

Недостатки

Для применения этих алгоритмов нужно знать о среде слишком много! При этом сама среда должна быть не очень большой!

Различные подходы

Постановка задачи

Как уже было изложено выше — на практике очень часто мы не знаем как устроена среда, то есть нам неизвестны $p(s'|s,a)$ и $r(s,a)$, поэтому от этого предположения мы отказываемся, но оставляем $|S| \ll \infty$ и $|A| \ll \infty$.

Идея 1: Model-based RL

Давайте сведем задачу к решенной: будем аппроксимировать модель и применять уже известные методы.

Идея 2: Model-free RL

Давайте сразу учить политику без обучения среды.

Model-based RL

Плюсы

- Задача аппроксимация среды может быть сведена к классическому *supervised learning*: отправляем в среду агента и обучаемся на данных полученных от него.

Минусы

- Переход к supervised learning почти никогда не работает, поскольку задача получается слишком сложная и нужно подбирать агента, который будет ее исследовать.

Model-free RL

Плюсы

- Учить среду не нужно.
- Оказывается, чтобы свести предыдущие алгоритмы к новой задаче, не обязательно учить среду.

Минусы

- Нельзя свести к обучению с учителем, поскольку для этого нет набора данных.

Model-free RL

Модификации алгоритмов

В *Policy Iteration* и *Value Iteration* обычно учат V-функцию, поскольку она занимает меньше памяти, однако ничего не мешает сразу начать учить Q-функцию, а потом менять по ней политику.

Замечание

Однако, чтобы выучить Q-функцию в алгоритмах выше, всё равно необходимо знать среду:

$$Q^{\pi_k}(s,a) = r(s,a) + \gamma \underline{\mathbb{E}_{s'}} \mathbb{E}_{a'} Q^{\pi_k}(s',a').$$

Чтобы решить данную проблему, придется считать Q-функцию методом Монте-Карло по определению.

Метод Монте-Карло

Оценка Q-функции

Насэмплируем достаточное количество траекторий по политике и усредним по ним значения Q-функции:

$$Q^{\pi}(s, a) \approx \frac{1}{M} \sum_{i=1}^{M} R(\tau_i),$$

где τ_i — одна из траекторий по политике π.

Замечание

Такой алгоритм будет несмещенной оценкой Q-функции, однако будет иметь очень большую дисперсию: в случае, если и среда и политика не детерминированы, то оценка есть сумма большого числа случайных величин и чем длиннее траектория, тем больше дисперсия.

Алгоритм Монте-Карло

Схема алгоритма

1) Инициализируем произвольную политику π.
2) В цикле по k:
   2.1) сэмплируем несколько траекторий при фиксированной политике π.
   2.2) Оцениваем $Q^{\pi}(s, a)$, используя Монте-Карло.
   2.3) Обновляем политику $\pi(s) \leftarrow \arg\max_a Q^{\pi}(s, a)$.

Недостатки алгоритма Монте-Карло

Недостатки

- Обновить Q-функцию нельзя, пока эпизод не сыгран до конца (не подходит для бесконечных игр).
- Не пользуемся MDP (потому что отказались от уравнения Беллмана).
- За конечное время точное значение Q-функции теперь не посчитать.
- Высокая дисперсия алгоритма.
- В данном виде алгоритм каждую новую итерацию цикла заново пересчитывает оценки Q-функции, выбрасывая старую оценку.

Модификация алгоритма Монте-Карло

Онлайн алгоритм

$$m_k := \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = [\alpha_k := \frac{1}{k}] =$$

$$= \underbrace{(1-\alpha_k)m_{k-1} + \alpha_k x_k}_{\text{exp. smoothing}} = \underbrace{m_{k-1} + \alpha_k(x_k - m_{k-1})}_{\approx \text{ stoch. grad. desc.}}$$

Замечание

Последнее равенство есть оптимизация *MSE Loss'a*. В оптимизации часто шаг обучения берут не только равным $\frac{1}{k}$.

Модификация алгоритма Монте-Карло

Теорема

$m_k \xrightarrow[\text{w. p. } 1]{k\to\infty} \mathbb{E}x$, если $\mathbb{E}x^2 < +\infty$ и шаг обучения удовлетворяет условиям Робинса-Монро (*Robbins-Monro conditions*):

$$\alpha_k \in [0,1],\ k \in \mathbb{N},\ \sum_{k=1}^{+\infty} \alpha_k = +\infty,\ \sum_{k=1}^{+\infty} \alpha_k^2 < +\infty.$$

Замечание

В применении в *RL* для сходимости метода Монте-Карло в условие теоремы необходимо добавить условие на политику:
$\forall s, a \implies \pi(a|s) > 0$, которое обеспечивает гарантировнное посещение агентом всех состояний среды (об этом подробнее в конце лекции).

Temporal Difference

Идея

$$Q^\pi(s,a) = \mathbb{E}_{s'}\underbrace{[r(s,a) + \gamma \mathbb{E}_{a'} Q^\pi(s',a')]}_{f(s',x)} =$$

$$= \mathbb{E}_{s'}\mathbb{E}_{a'}\underbrace{[r(s,a) + \gamma Q^\pi(s',a')]}_{f(s',a',x)}.$$

Теперь, чтобы обновить Q-функцию для (s,a) нам нужен только переход (s,a,r,s',a'):

$$y := r + \gamma Q^\pi(s',a');$$
$$Q^\pi(s,a) \leftarrow (1-\alpha_k)Q^\pi(s,a) + \alpha_k y.$$

Temporal Difference

Перепишем полученное

Для перехода (s, a, r, s', a') :

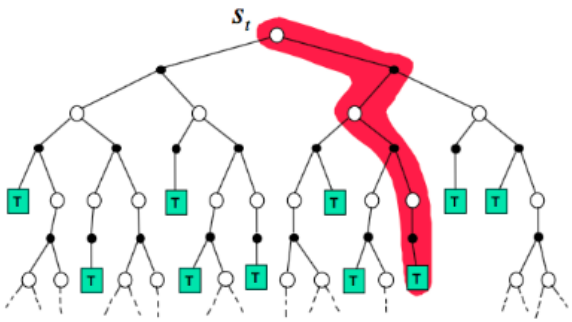
$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \underbrace{\left(\overbrace{r + \gamma Q^{\pi}_{k}(s',a')}^{\text{Bellman target}} - Q^{\pi}_{k}(s,a) \right)}_{\text{temporal difference}},$$

где s, a — фиксированы, $s' \sim p(s'|s,a)$ — случайная величина от взаимодействия со средой, $a' \sim \pi(a'|s')$ — случайная величина из нашего алгоритма.

Сравнение работы алгоритмов

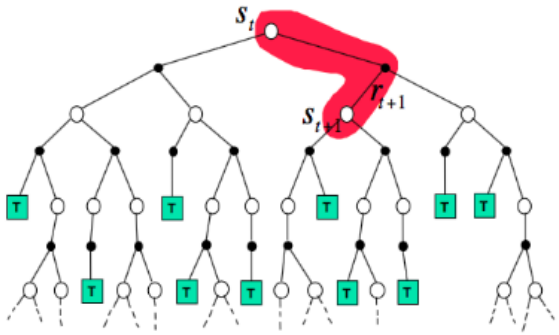
Monte-Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$



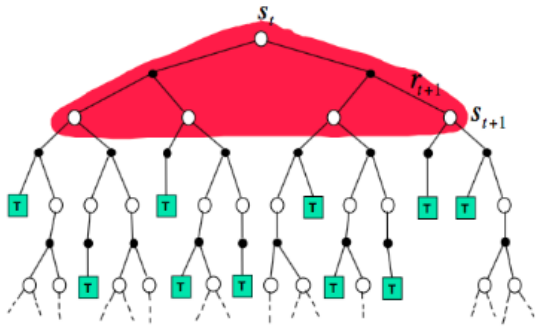
Temporal-Difference

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



Dynamic Programming

$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$



Сравнение работы алгоритмов

$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k(y(s,a) - Q^{\pi}_{k}(s,a))$$

Temporal Difference

- $y(s,a) := r + \gamma Q^{\pi}_{k}(s',a')$
- Обновление происходит после каждого шага.
- Медленное распространение награды (посещает по паре состояний).

Monte Carlo

- $y(s,a) := r + \gamma r' + \gamma r'' + \ldots$
- Обновление происходит в конце эпизода.
- Быстрое распространение награды (посещает несколько состояний в эпизоде разом).

Сравнение работы алгоритмов

Temporal Difference

- Маленькая дисперсия.
- Смещенная оценка.

Monte Carlo

- Большая дисперсия.
- Несмещенная оценка.

Добавляем улучшение политики

Идея

Идея такая же как в Value Iteration — давайте улучшать нашу оценку Q-функции с помощью TD и тут же улучшать нашу политику:

$$
\begin{aligned}
&Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \left(r + \gamma Q^{\pi}_{k}(s',a') - Q^{\pi}_{k}(s,a)\right) = \\
&= Q^{\pi}_{k}(s,a) + \alpha_k (r + \gamma Q^{\pi}_{k}(s', \pi_k(s')) - Q^{\pi}_{k}(s,a)) = \\
&= Q^{\pi}_{k}(s,a) + \alpha_k (r + \gamma Q^{\pi}_{k}(s', \arg\max_a Q^{\pi}_{k}(s,a)) - Q^{\pi}_{k}(s,a)) = \\
&= Q^{\pi}_{k}(s,a) + \alpha_k (r + \gamma \max_{a'} Q^{\pi}_{k}(s',a') - Q^{\pi}_{k}(s,a)).
\end{aligned}
$$

Сходимость Q-обучения

Теорема

Пусть в конечном MDP, где $Q_0^*(s,a)$ — произвольно, дан алгоритм Q-обучения вида:

$$Q_{k+1}^*(s,a) \leftarrow Q_k^*(s,a) + \alpha_k(s,a)(r(s,a) + \\ + \gamma \max_{a'} Q_k^*(s'_{k,s,a}, a') - Q_k^*(s,a)).$$

Тогда, если $s'_{k,s,a} \sim p(s'|s,a)$ и с вер-ю 1 $\forall s,a\ \alpha_k(s,a) \in (0,1]$ удовлетворяет условиям Роббинса-Монро, то

$$Q_k^*(s,a) \xrightarrow[k\to\infty]{\text{w.p. } 1} Q^*(s,a).$$

Сходимость Q-обучения

Замечание

Эта теорема точно справедлива, если мы на каждой итерации обновляем Q-функцию для всех состояний и действий. Но на практике это не выполняется — мы обновляем Q-функию только в тех состояниях, которые видим. Однако оказывается, что, если каждое состояние мы можем посетить с положительной вероятностью, то эта теорема будет работать, что приводит нас к *Exploration/Exploitation trade off*.

Сходимость Q-обучения

Проблема

Жадная по Q-функции политика с большой вероятностью не посетит все возможные состояния среды (вероятность принять жадное действие равна 1, а остальные 0), поэтому сходимость весьма условна!

Решение

Если $\forall s, a \implies \pi(a|s) > 0$, то гарантированно рано или поздно агент посетит все состояния. Этого можно достичь, например, тривиальной равномерной политикой, и теорема о сходимости будет работать.

Итого

Свойство $\forall s, a \implies \pi(a|s) > 0$ — отвечает за исследование среды, жадность — за решение задачи. Нужно найти баланс между двумя этими политиками.

ε-жадное исследование среды

Идея метода

На каждом шаге агента с вероятностью $1 - \varepsilon$ выбирается жадное действие, а с вероятностью ε — совершенно случайное действие. Для такой политики будет верно, что $\forall s, a \implies \pi(a|s) > 0$, поэтому будет наблюдаться сходимость.

ε-жадное исследование среды

Достоинство

Очень простой метод в реализации, применяется в RL повсеместно.

Недостаток

Очень плохо работает, идея очень наивная.

Схема алгоритма Q-обучения

Q-learning

1) Инициализируем $Q^*(s, a)$ произвольно.
2) Наблюдаем s_0.
3) В цикле по $k = 0, 1, 2, \ldots$:
   3.1) Выбираем действие $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$.
   3.2) Наблюдаем r_k, s_{k+1}.
   3.3) Обновляем Q-функцию:

$$y = r_k + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1});$$

$$Q^*(s_k, a_k) \leftarrow (1 - \alpha_k) Q^*(s_k, a_k) + \alpha_k y.$$

Замечание

Такой алгоритм гарантированно сходится.

Буфер в алгоритме Q-обучения

Идея

На самом деле Q-обучение (в отличие от *Monte Carlo* алгоритма) является *off-policy* алгоритмом (об этом позже), это означает в частности, что алгоритм может обучаться не только на своих дейсвиях в данный момент времени, но и на действиях экспертов или на своих же старых действиях, поэтому очень удобно добавить в алгоритм буфер, в который мы будем класть действия агента и их результаты, а потом во время дальнейшего обучения, будем случайно доучиваться на сэмплах из буфера.