Глава 3

Классическая теория

В данной главе будут доказаны основные теоретические результаты о MDP и получены важные «табличные» алгоритмы, работающие в случае конечных пространств состояний и действий; они лягут в основу всех дальнейших алгоритмов.

§3.1. Оценочные функции

3.1.1. Свойства траекторий

Как это всегда бывает, чем более общую задачу мы пытаемся решать, тем менее эффективный алгоритм мы можем придумать. В RL мы сильно замахиваемся: хотим построить алгоритм, способный обучаться решению «произвольной» задачи, заданной средой с описанной функцией награды. Однако в формализме MDP в постановке мы на самом деле внесли некоторые ограничения: марковость и стационарность. Эти предположения практически не ограничивают общность нашей задачи с точки зрения здравого смысла с одной стороны и при этом вносят в нашу задачу некоторую «структуру»; мы сможем придумать более эффективные алгоритмы решения за счёт эксплуатации этой структуры.

Что значит «структуру»? Представим, что мы решаем некоторую абстрактную задачу последовательного принятия решения, максимизируя некоторую кумулятивную награду. Вот мы находимся в некотором состоянии и должны выбрать некоторое действие. Интуитивно ясно, что на прошлое — ту награду, которую мы уже успели собрать — мы уже повлиять не можем, и нужно максимизировать награду в будущем. Более того, мы можем отбросить всю нашу предыдущую историю и задуматься лишь над тем, как максимизировать награду с учётом сложившейся ситуации — «текущего состояния».

Пример 45 — Парадокс обжоры: Обжора пришёл в ресторан и заказал кучу-кучу еды. В середине трапезы выяснилось, что оставшиеся десять блюд явно лишние и в него уже не помещаются. Обидно: они будут в счёте, да и не пропадать же еде, поэтому надо бы всё равно всё съесть. Однако, с точки зрения функции награды нужно делать противоположный вывод: блюда будут в счёте в любом случае, вне зависимости от того, будут ли они съедены — это награда за уже совершённое действие, «прошлое», — а вот за переедание может прилететь ещё отрицательной награды. Обжора понимает, что в прошлом совершил неоптимальное действие, и пытается «прооптимизировать» неизбежную награду за прошлое, в результате проигрывая ещё.

Давайте сформулируем эту интуицию формальнее. Как и в обычных Марковских цепях, в средах благодаря марковости действует закон «независимости прошлого и будущего при известном настоящем». Формулируется он так:

Утверждение 6 — Независимость прошлого и будущего при известном настоящем: Пусть $\mathcal{T}_{:t} \coloneqq \{s_0, a_0 \dots s_{t-1}, a_{t-1}\}$ — «прошлое», s_t — «настоящее», $\mathcal{T}_{t:} \coloneqq \{a_t, s_{t+1}, a_{t+1} \dots\}$ — «будущее». Тогда:

$$p\left(\mathcal{T}_{:t}, \mathcal{T}_{t:} \mid s_t\right) = p\left(\mathcal{T}_{:t} \mid s_t\right) p\left(\mathcal{T}_{t:} \mid s_t\right)$$

Доказательство. По правилу произведения:

$$p\left(\mathcal{T}_{:t}, \mathcal{T}_{t:} \mid s_t\right) = p\left(\mathcal{T}_{:t} \mid s_t\right) p\left(\mathcal{T}_{t:} \mid s_t, \mathcal{T}_{:t}\right)$$

Однако в силу марковости будущее зависит от настоящего и прошлого только через настоящее:

$$p\left(\mathcal{T}_{t:} \mid s_t, \mathcal{T}_{:t}\right) = p\left(\mathcal{T}_{t:} \mid s_t\right)$$

■

Для нас утверждение означает следующее: если мы сидим в момент времени t в состоянии s и хотим посчитать награду, которую получим в будущем (то есть величину, зависящую только от $\mathcal{T}_{t:}$), то нам совершенно не важна история попадания в s. Это следует из свойства мат. ожиданий по независимым переменным:

$$\mathbb{E}_{\mathcal{T}|s_t=s}\boldsymbol{R}(\mathcal{T}_{t:}) = \{\text{утв. 6}\} = \mathbb{E}_{\mathcal{T}_{:t}|s_t=s}\underbrace{\mathbb{E}_{\mathcal{T}_{t:}|s_t=s}\boldsymbol{R}(\mathcal{T}_{t:})}_{\text{не зависит от } \mathcal{T}_{:t}} = \mathbb{E}_{\mathcal{T}_{t:}|s_t=s}\boldsymbol{R}(\mathcal{T}_{t:})$$

Определение 31: Для траектории $\mathcal{T}$ величина

$$\boldsymbol{R_t} \coloneqq \boldsymbol{R}\left(\mathcal{T}_{t:}\right) = \sum_{\hat{t}\geq t}\gamma^{\hat{t}-t}r_{\hat{t}} \tag{3.1}$$

называется ***reward-to-go*** с момента времени t.

Благодаря второму сделанному предположению, о стационарности (в том числе стационарности стратегии агента), получается, что будущее также не зависит от текущего момента времени t: всё определяется исключительно текущим состоянием. Иначе говоря, агенту неважно не только, как он добрался до текущего состояния и сколько награды встретил до настоящего момента, но и сколько шагов в траектории уже прошло. Формально это означает следующее: распределение будущих траекторий имеет в точности тот же вид, что и распределение всей траектории при условии заданного начала.

Утверждение 7: Будущее определено текущим состоянием:

$$\boldsymbol{p}\left(\mathcal{T}_{t:} \mid s_t = s\right) \equiv \boldsymbol{p}\left(\mathcal{T} \mid s_0 = s\right)$$

Доказательство. По определению:

$$\boldsymbol{p}\left(\mathcal{T}_{t:} \mid s_t = s\right) = \prod_{\hat{t}\geq t}\boldsymbol{p}(s_{\hat{t}+1} \mid s_{\hat{t}}, a_{\hat{t}})\boldsymbol{\pi}(a_{\hat{t}} \mid s_{\hat{t}}) = (*)$$

Воспользуемся однородностью MDP и однородностью стратегии, а именно:

$$\begin{gathered}\boldsymbol{\pi}(a_{\hat{t}} \mid s_{\hat{t}} = s) = \boldsymbol{\pi}(a_0 \mid s_0 = s)\\ \boldsymbol{p}(s_{\hat{t}+1} \mid s_{\hat{t}} = s, a_{\hat{t}}) = \boldsymbol{p}(s_1 \mid s_0 = s, a_0)\\ \boldsymbol{\pi}(a_{\hat{t}+1} \mid s_{\hat{t}+1}) = \boldsymbol{\pi}(a_1 \mid s_1)\\ \boldsymbol{p}(s_{\hat{t}+2} \mid s_{\hat{t}+1}, a_{\hat{t}+1}) = \boldsymbol{p}(s_2 \mid s_1, a_1)\end{gathered}$$

и так далее, получим:

$$(*) = \prod_{t\geq 0}\boldsymbol{p}(s_{t+1} \mid s_t, a_t)\boldsymbol{\pi}(a_t \mid s_t) = \boldsymbol{p}\left(\mathcal{T} \mid s_0 = s\right) \qquad \blacksquare$$

Утверждение 8: Для любого t и любой функции f от траекторий:

$$\mathbb{E}_{\mathcal{T}|s_0=s}\boldsymbol{f}(\mathcal{T}) = \mathbb{E}_{\mathcal{T}|s_t=s}\boldsymbol{f}(\mathcal{T}_{t:})$$

Доказательство.

$$\mathbb{E}_{\mathcal{T}|s_0=s}\boldsymbol{f}(\mathcal{T}) = \{\text{утв. 7}\} = \mathbb{E}_{\mathcal{T}_{t:}|s_t=s}\boldsymbol{f}(\mathcal{T}_{t:}) = \{\text{утв. 6}\} = \mathbb{E}_{\mathcal{T}|s_t=s}\boldsymbol{f}(\mathcal{T}_{t:}) \qquad \blacksquare$$

Мы показали, что все свойства reward-to-go определяются исключительно стартовым состоянием.

3.1.2. V-функция

Итак, наша интуиция заключается в том, что, когда агент приходит в состояние s, прошлое не имеет значения, и оптимальный агент должен максимизировать в том числе и награду, которую он получит, стартуя из состояния s. Поэтому давайте «обобщим» наш оптимизируемый функционал, варьируя стартовое состояние:

Определение 32: Для данного MDP ***V-функцией*** (value function) или оценочной функцией состояний (state value function) для данной стратегии π называется величина

$$V^\pi(s) \coloneqq \mathbb{E}_{\mathcal{T}\sim\pi|s_0=s} R(\mathcal{T}) \tag{3.2}$$

По определению функция ценности состояния, или V-функция — это сколько набирает в среднем агент из состояния s. Причём в силу марковости и стационарности неважно, случился ли старт на нулевом шаге эпизода или на произвольном t-ом:

Утверждение 9: Для любого t верно:

$$V^\pi(s) = \mathbb{E}_{\mathcal{T}\sim\pi|s_t=s} R_t$$

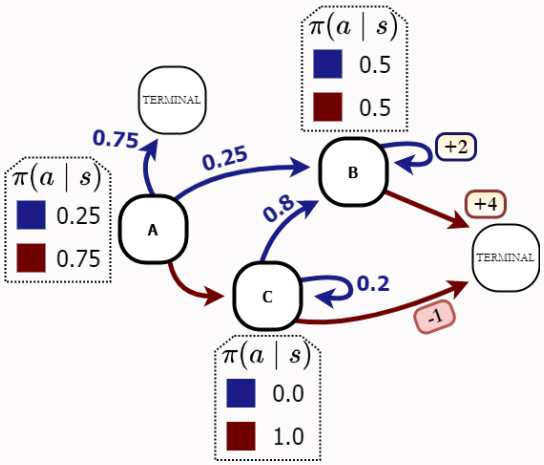
Пояснение. Применить утверждение 8 для $R(\mathcal{T})$. ■

Утверждение 10: $V^\pi(s)$ ограничено.

Утверждение 11: Для терминальных состояний $V^\pi(s) = 0$.

Заметим, что любая политика π индуцирует V^π. То есть для данного MDP и данной стратегии π функция V^π однозначно задана своим определением; совсем другой вопрос, можем ли мы вычислить эту функцию.

Пример 46: Посчитаем V-функцию для MDP и стратегии π с рисунка, $\gamma = 0.8$. Её часто удобно считать «с конца», начиная с состояний, близких к терминальным, и замечая связи между значениями функции для разных состояний.



Начнём с состояния C: там агент всегда выбирает действие ■, получает -1, и эпизод заканчивается: $V^\pi(s = C) = -1$.

Для состояния B с вероятностью 0.5 агент выбирает действие ■ и получает $+4$. Иначе он получает $+2$ и возвращается снова в состояние B. Вся дальнейшая награда будет дисконтирована на $\gamma = 0.8$ и тоже равна $V^\pi(s = B)$ по определению. Итого:

$$V^\pi(s = B) = \underbrace{0.5 \cdot 4}_{\blacksquare} + \underbrace{0.5 \cdot (2 + \gamma V^\pi(s = B))}_{\blacksquare}$$

Решая это уравнение относительно $V^\pi(s = B)$, получаем ответ 5.

Для состояния A достаточно аналогично рассмотреть все дальнейшие события:

$$V^\pi(s = A) = \underbrace{0.25}_{\blacksquare} \cdot \left(\underbrace{0.75 \cdot 0}_{\text{terminal}} + \underbrace{0.25\gamma V^\pi(s = B)}_{B} \right) + \underbrace{0.75}_{\blacksquare} \underbrace{\gamma V^\pi(s = C)}_{C}$$

Подставляя значения, получаем ответ $V^\pi(s = A) = -0.35$.

3.1.3. Уравнения Беллмана

Если s_0 — стартовое состояние, то $V^\pi(s_0)$ по определению и есть функционал (1.5), который мы хотим оптимизировать. Формально, это единственная величина, которая нас действительно волнует, так как она нам явно задана в самой постановке задачи, но мы понимаем, что для максимизации $V^\pi(s_0)$ нам нужно промаксимизировать и $V^\pi(s)$ (строго мы это пока не показали). Другими словами, у нас в задаче есть ***подзадачи эквивалентной структуры***: возможно, они, например, проще, и мы можем сначала их решить, а дальше как-то воспользоваться этими решениями для решения более сложной. Вот если граф MDP есть дерево, например, то очевидно, как считать V^π: посчитать значение в листьях (листья соответствуют терминальным состояниям — там ноль), затем в узлах перед листьями, ну и так далее индуктивно добраться до корня.

Мы заметили, что в примере 46 на значения V-функции начали появляться рекурсивные соотношения. В этом и есть смысл введения понятия оценочных функций — «дополнительных переменных»: в том, что эти значения связаны между собой ***уравнениями Беллмана*** (Bellman equations).

Теорема 8 — Уравнение Беллмана (Bellman expectation equation) для V^π:

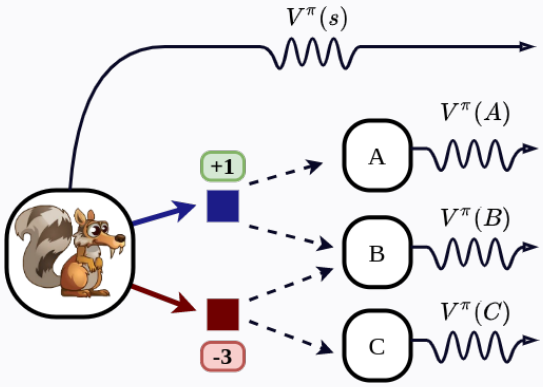
$$V^\pi(s) = \mathbb{E}_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V^\pi(s') \right] \tag{3.3}$$

Доказательство. Интуиция: награда за игру равна награде за следующий шаг плюс награда за оставшуюся игру; награда за хвост равна следующей награде плюс награда за хвост. Действительно, для всех траекторий $\mathcal{T}$ и для любых t верно:

$$R_t = r_t + \gamma R_{t+1}$$

Соответственно, для формального доказательства раскладываем сумму по времени как первое слагаемое плюс сумма по времени и пользуемся утверждением 9 о независимости V-функции от времени:

$$V^\pi(s) = \mathbb{E}_{\mathcal{T}|s_t=s} R_t = \mathbb{E}_{a_t} \left[r_t + \gamma \mathbb{E}_{s_{t+1}} \mathbb{E}_{\mathcal{T}\sim\pi|s_{t+1}} R_{t+1} \right] =$$
$$= \mathbb{E}_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V^\pi(s') \right] \quad \blacksquare$$



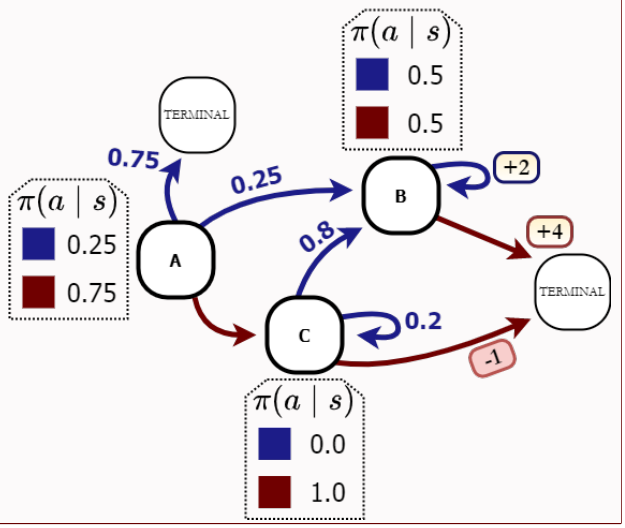
Пример 47: Выпишем уравнения Беллмана для MDP и стратегии π из примера 46. Число уравнений совпадает с числом состояний. Разберём подробно уравнение для состояния A:

$$V^\pi(A) = \underbrace{0.25}_{\blacksquare}(0 + \gamma 0.25 V^\pi(B)) + \underbrace{0.75}_{\blacksquare}(0 + \gamma V^\pi(C))$$

С вероятностью 0.25 будет выбрано действие ■, после чего случится дисконтирование на γ; с вероятностью 0.75 эпизод закончится и будет выдана нулевая награда, с вероятностью 0.25 агент перейдёт в состояние B. Второе слагаемое уравнения будет отвечать выбору действия ■; агент тогда перейдёт в состояние C и, начиная со следующего шага, получит в будущем $V^\pi(C)$. Аналогично расписываются два оставшихся уравнения.

$$V^\pi(A) = \frac{1}{16}\gamma V^\pi(B) + \frac{3}{4}\gamma V^\pi(C)$$
$$V^\pi(B) = 0.5\,(2 + \gamma V^\pi(B)) + 0.5 \cdot 4$$
$$V^\pi(C) = -1$$

Заметим, что мы получили систему из трёх линейных уравнений с тремя неизвестными.



Позже мы покажем, что V^π является единственной функцией $\mathcal{S} \to \mathbb{R}$, удовлетворяющей уравнениям Беллмана для данного MDP и данной стратегии π, и таким образом однозначно ими задаётся.

3.1.4. Оптимальная стратегия

У нас есть конкретный функционал $J(\pi) = V^{\pi}(s_0)$, который мы хотим оптимизировать. Казалось бы, понятие оптимальной политики очевидно как вводить:

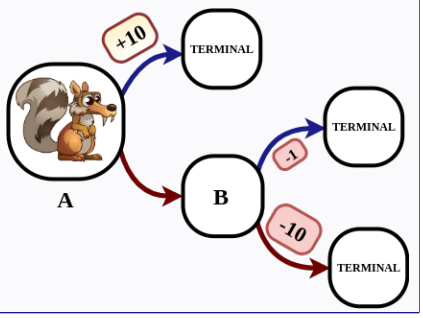
Определение 33: Политика π^* ***оптимальна***, если $\forall \pi\colon V^{\pi^*}(s_0) \geq V^{\pi}(s_0)$.

Введём альтернативное определение:

Определение 34: Политика π^* ***оптимальна***, если $\forall \pi, s\colon V^{\pi^*}(s) \geq V^{\pi}(s)$.

Теорема 9: Определения не эквивалентны.

Доказательство. Из первого не следует второе (из второго первое, конечно, следует). Контрпример приведён на рисунке. С точки зрения нашего функционала, оптимальной будет стратегия сразу выбрать ■ и закончить игру. Поскольку оптимальный агент выберет ■ с вероятностью 0, ему неважно, какое решение он будет принимать в состоянии B, в котором он никогда не окажется. Согласно первому определению, оптимальная политика может действовать в B как угодно. Однако, чтобы быть оптимальной согласно второму определению и в том числе максимизировать $V^{\pi}(s = B)$, стратегия обязана выбирать в B только действие ■. ■



Интуиция подсказывает, что различие между определениями проявляется только в состояниях, которые оптимальный агент будет избегать с вероятностью 1 (позже мы увидим, что так и есть). Задавая оптимальность вторым определением, мы чуть-чуть усложняем задачу, но упрощаем теоретический анализ: если бы мы оставили первое определение, у оптимальных политик могли бы быть разные V-функции (см. пример из последнего доказательства); согласно второму определению, V-функция всех оптимальных политик совпадает.

Определение 35: Оптимальные стратегии будем обозначать π^*, а соответствующую им ***оптимальную V-функцию*** — V^*:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{3.4}$$

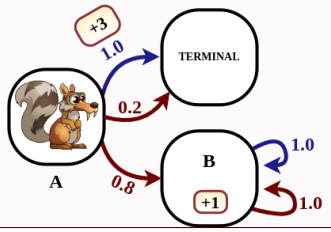
Пока нет никаких обоснований, что найдётся стратегия, которая максимизирует $V^{\pi}(s)$ сразу для всех состояний. Вдруг в одних s макисмум (3.4) достигается на одной стратегии, а в другом — на другой? Тогда оптимальных стратегий в сильном смысле вообще не существует, хотя формальная величина (3.4) существует. Пока заметим лишь, что для ситуации, когда MDP — дерево, существование оптимальной стратегии в смысле второго определения можно опять показать «от листьев к корню».

Пример 48: Рассмотрим MDP из примера 9; $\gamma = \frac{10}{11}$, множество стратегий параметризуется единственным числом $\theta := \pi(a = \blacksquare \mid s = A)$.

По определению оптимальная V-функция для состояния A равна

$$V^*(s = A) = \max_{\theta \in [0,1]} J(\pi) = \max_{\theta \in [0,1]} [3 + 5\theta] = 8.$$

Оценочные функции для состояния B для всех стратегий совпадают и равны $V^*(s = B) = 1 + \gamma + \gamma^2 + \ldots = 11$. Для терминальных состояний $V^*(s) = 0$.



3.1.5. Q-функция

V-функции нам не хватит. Если бы мы знали оптимальную value-функцию $V^*(s)$, мы не смогли бы восстановить хоть какую-то оптимальную политику из-за отсутствия в общем случае информации о динамике среды. Допустим, агент находится в некотором состоянии и знает его ценность $V^*(s)$, а также знает ценности всех других состояний; это не даёт понимания того, какие действия в какие состояния приведут — мы никак не дифференцируем действия между собой. Поэтому мы увеличим количество переменных: введём схожее определение для ценности не состояний, но пар состояние-действие.

Определение 36: Для данного MDP ***Q-функцией*** (state-action value function, action quality function) для данной стратегии π называется

$$Q^{\pi}(s, a) := \mathbb{E}_{\mathcal{T} \sim \pi \mid s_0 = s, a_0 = a} \sum_{t \geq 0} \gamma^t r_t$$

Теорема 10 — Связь оценочных функций: V-функции и Q-функции взаимозависимы, а именно:

$$Q^{\pi}(s, a) = r(s, a) + \gamma \mathbb{E}_{s'} V^{\pi}(s') \tag{3.5}$$

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(a|s)} Q^\pi(s,a) \tag{3.6}$$

Доказательство. Следует напрямую из определений. ■

Итак, если V-функция — это сколько получит агент из некоторого состояния, то Q-функция — это сколько получит агент после выполнения данного действия из данного состояния. Как и V-функция, Q-функция не зависит от времени, ограничена по модулю при рассматриваемых требованиях к MDP, и, аналогично, для неё существует уравнение Беллмана:

Теорема 11 — Уравнение Беллмана (Bellman expectation equation) для Q-функции:

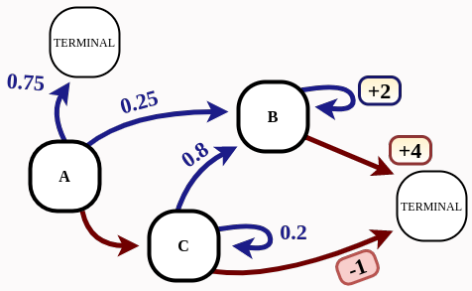
$$Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a'} Q^\pi(s',a') \tag{3.7}$$

Доказательство. Можно воспользоваться (3.5) + (3.6), можно расписать как награду на следующем шаге плюс хвостик. ■

Пример 49: Q-функция получает на вход пару состояние-действие и ничего не говорится о том, что это действие должно быть как-то связано с оцениваемой стратегией π.

Давайте в MDP с рисунка рассмотрим стратегию π, которая всегда детерминировано выбирает действие ■. Мы тем не менее можем посчитать $Q^\pi(s, ■)$ для любых состояний (например, для терминальных это значение формально равно нулю). Сделаем это при помощи QV уравнения:

$$Q^\pi(s=A, ■) = 0.25\gamma V^\pi(s=B)$$
$$Q^\pi(s=B, ■) = 2 + \gamma V^\pi(s=B)$$
$$Q^\pi(s=C, ■) = 0.8\gamma V^\pi(s=B) + 0.2\gamma V^\pi(s=C)$$



Внутри V^π сидит дальнейшее поведение при помощи стратегии π, то есть выбор исключительно действий ■: соответственно, $V^\pi(s=B) = 4$, $V^\pi(s=C) = -1$.

Мы получили все уравнения Беллмана для оценочных функций (с условными названиями VV, VQ, QV и, конечно же, QQ). Как видно, они следуют напрямую из определений; теперь посмотрим, что можно сказать об оценочных функциях оптимальных стратегий.

3.1.6. Принцип оптимальности Беллмана

Определение 37: Для данного MDP ***оптимальной Q-функцией*** (optimal Q-function) называется

$$Q^*(s,a) := \max_\pi Q^\pi(s,a) \tag{3.8}$$

Формально очень хочется сказать, что Q^* — оценочная функция для оптимальных стратегий, но мы пока никак не связали введённую величину с V^* и показать это пока не можем. Нам доступно только такое неравенство пока что:

Утверждение 12:

$$Q^*(s,a) \le r + \gamma \mathbb{E}_{s'} V^*(s')$$

Доказательство.

$$Q^*(s,a) = \{\text{определение } Q^* \text{ (3.8) }\} = \max_\pi Q^\pi(s,a) =$$
$$= \{\text{связь QV (3.5)}\} = \max_\pi [r + \gamma \mathbb{E}_{s'} V^\pi(s')] \le$$
$$\le \{ \text{ максимум среднего не превосходит среднее максимума } \} \le r + \gamma \mathbb{E}_{s'} \max_\pi V^\pi(s') =$$
$$= \{ \text{ определение } V^* \text{ (3.4) } \} = r + \gamma \mathbb{E}_{s'} V^*(s')$$ ■

Равенство в месте с неравенством случилось бы, если бы мы доказали следующий факт: что вообще существует такая стратегия π, которая максимизирует V^π сразу для всех состояний s одновременно (и которую мы определили как оптимальную). Другими словами, нужно показать, что максимизация $V^\pi(s)$ для одного состояния «помогает» максимизировать награду для других состояний. Для V-функции мы можем построить аналогичную оценку сверху:

Утверждение 13:

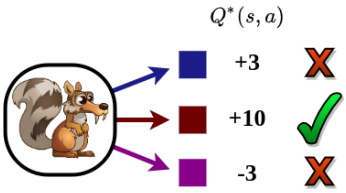
$$V^*(s) \le \max_a Q^*(s,a)$$

Доказательство.

$$\begin{aligned} V^*(s) = \{\text{определение } V^* \ (3.4)\ \} = \max_\pi V^\pi(s) = \\ = \{\text{связь VQ } (3.6)\} = \max_\pi \mathbb{E}_{a\sim\pi(a|s)} Q^\pi(s,a) \le \\ \le \{\ \text{по определению } Q^* \ (3.8)\ \} \le \max_\pi \mathbb{E}_{a\sim\pi(a|s)} Q^*(s,a) \le \\ \le \{\ \text{свойство } \mathbb{E}_x f(x) \le \max_x f(x)\ \} \le \max_a Q^*(s,a) \end{aligned}$$

■

Можно ли получить $\max_a Q^*(s,a)$, то есть достигнуть этой верхней оценки? Проведём нестрогое следующее рассуждение: представим, что мы сидим в состоянии s и знаем величины $Q^*(s,a)$, определённые как (3.8). Это значит, что если мы сейчас выберем действие a, то в дальнейшем сможем при помощи какой-то стратегии π, на которой достигается максимум[1] для конкретно данной пары s, a, получить $Q^*(s,a)$. Следовательно, мы, выбрав сейчас то действие a, на которых достигается максимум, в предположении «дальнейшей оптимальности своего поведения», надеемся получить из текущего состояния $\max_a Q^*(s,a)$.



Определение 38: Для данного приближения Q-функции стратегия $\pi(s) := \operatorname{argmax}_a Q(s,a)$ называется ***жадной*** (greedy with respect to Q-function).

Определение 39: ***Принцип оптимальности Беллмана***: жадный выбор действия в предположении оптимальности дальнейшего поведения оптимален.

Догадку несложно доказать для случая, когда MDP является деревом: принятие решения в текущем состоянии s никак не связано с выбором действий в «поддеревьях». Если в поддереве, соответствующему одному действию, можно получить больше, чем в другом поддереве, то понятно, что выбирать нужно его. В общем случае, однако, нужно показать, что жадный выбор в s «позволит» в будущем набрать то $Q^*(s,a)$, которое мы выбрали — вдруг для того, чтобы получить в будущем $Q^*(s,a)$, нужно будет при попадании в то же состояние s выбирать действие как-то по-другому? Если бы это было так, было бы оптимально искать стратегию в классе нестационарных стратегий.

3.1.7. Отказ от однородности

Утверждение, позволяющее, во-первых, получить вид оптимальной стратегии, а как следствие связать оптимальные оценочные функции, будет доказано двумя способами. В этой секции докажем через отказ от однородности («классическим» способом), а затем в секции 3.2 про Policy Improvement мы поймём, что все желаемые утверждения можно получить и через него.

Отказ от однородности заключается в том, что мы в доказательстве будем искать максимум $\max_\pi V^\pi(s)$ не только среди стационарных, но и нестационарных стратегий. Заодно мы убедимся, что достаточно искать стратегию в классе стационарных стратегий. Ранее стационарность означала, что вне зависимости от момента времени наша стратегия зависит только от текущего состояния. Теперь же, для каждого момента времени $t = 0, 1\dots$ мы запасёмся своей собственной стратегией $\pi_t(a \mid s)$. Естественно, что теорема 9 о независимости оценочной функции от времени тут перестаёт быть истинной, и, вообще говоря, оценочные функции теперь зависимы от текущего момента времени t.

Определение 40: Для данного MDP и нестационарной стратегии $\pi = \{\pi_t(a \mid s) \mid t \ge 0\}$ обозначим её ***оценочные функции*** как

$$V_t^\pi(s) := \mathbb{E}_{\pi_t(a_t|s_t=s)} \mathbb{E}_{p(s_{t+1}|s_t=s,a_t)} \mathbb{E}_{\pi_{t+1}(a_{t+1}|s_{t+1})} \dots R_t$$

$$Q_t^\pi(s,a) := \mathbb{E}_{p(s_{t+1}|s_t=s,a_t=a)} \mathbb{E}_{\pi_{t+1}(a_{t+1}|s_{t+1})} \dots R_t$$

Пример 50: Действительно, мы можем в состоянии s смотреть на часы, если $t = 0$ — кушать тортики, а если $t = 7$ — бросаться в лаву, т.е. $V^\pi_{t=0}(s) \ne V^\pi_{t=7}(s)$ для неоднородных π.

[1] мы знаем, что Q-функция ограничена, и поэтому точно существует супремум. Для полной корректности рассуждений надо говорить об ε-оптимальности, но для простоты мы это опустим.

Утверждение 14: Для нестационарных оценочных функций остаются справедливыми уравнения Беллмана:

$$V_t^\pi(s) = \mathbb{E}_{\pi_t(a|s)} Q_t^\pi(s,a) \tag{3.9}$$

$$Q_t^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} V_{t+1}^\pi(s') \tag{3.10}$$

Доказательство. Всё ещё следует из определений. ■

Определение 41: Для данного MDP ***оптимальными оценочными функциями среди нестационарных стратегий*** назовём

$$V_t^*(s) \coloneqq \max_{\substack{\pi_t \\ \pi_{t+1} \\ \cdots}} V_t^\pi(s) \tag{3.11}$$

$$Q_t^*(s,a) \coloneqq \max_{\substack{\pi_{t+1} \\ \pi_{t+2} \\ \cdots}} Q_t^\pi(s,a) \tag{3.12}$$

Заметим, что в определении Q-функции максимум берётся по стратегиям, начиная с π_{t+1}, поскольку по определению Q-функция не зависит от π_t (действие в момент времени t уже «дано» в качестве входа).

Утверждение 15: В стационарных MDP (а мы рассматриваем только их) оптимальные оценочные функции не зависят от времени, т.е. $\forall s, a, t_1, t_2$ верно:

$$V_{t_1}^*(s) = V_{t_2}^*(s) \qquad Q_{t_1}^*(s,a) = Q_{t_2}^*(s,a)$$

Доказательство. Вообще говоря, по построению, так как зависимость от времени заложена исключительно в стратегиях, по которым мы берём максимум (а его мы берём по одним и тем же симплексам вне зависимости от времени):

$$V_t^*(s) = \max_{\substack{\pi_t \\ \pi_{t+1} \\ \cdots}} \mathbb{E}_{a_t, s_{t+1} \cdots | s_t = s} R_t = \max_{\substack{\pi_0 \\ \pi_1 \\ \cdots}} \mathbb{E}_{a_0, s_1 \cdots | s_0 = s} R_0$$ ■

Последнее наблюдение само по себе нам ничего не даёт. Вдруг нам в условном MDP с одним состоянием выгодно по очереди выбирать каждое из трёх действий?

3.1.8. Вид оптимальной стратегии (доказательство через отказ от однородности)

Мотивация в отказе от однородности заключается в том, что наше MDP теперь стало деревом: эквивалентно было бы сказать, что мы добавили в описание состояний время t. Теперь мы не оказываемся в одном состоянии несколько раз за эпизод; максимизация $Q_t^*(s,a)$ требует оптимальных выборов «в поддереве», то есть настройки π_{t+1}, π_{t+2} и так далее, а для $\pi_t(a \mid s)$ будет выгодно выбрать действие жадно. Покажем это формально.

Теорема 12: Стратегия $\pi_t(s) \coloneqq \operatorname*{argmax}_a Q_t^*(s,a)$ оптимальна, то есть для всех состояний s верно $V_t^\pi(s) = V_t^*(s)$, и при этом справедливо:

$$V_t^*(s) = \max_a Q_t^*(s,a) \tag{3.13}$$

Доказательство. В силу VQ уравнения (3.9), максимизация $V_t^\pi(s)$ эквивалентна максимизации

$$V_t^*(s) = \max_{\substack{\pi_t \\ \pi_{t+1} \\ \cdots}} V_t^\pi(s) = \max_{\substack{\pi_t \\ \pi_{t+1} \\ \cdots}} \mathbb{E}_{\pi_t(a|s)} Q_t^\pi(s,a)$$

Мы уже замечали, что Q_t^π по определению зависит только от $\pi_{t+1}(a \mid s), \pi_{t+2}(a \mid s) \ldots$. Максимум $Q_t^\pi(s,a)$ по ним по определению (3.12) есть $Q_t^*(s,a)$. Значит,

$$V_t^*(s) \le \max_{\pi_t} \mathbb{E}_{\pi_t(a|s)} \max_{\substack{\pi_{t+1} \\ \pi_{t+2} \\ \cdots}} Q_t^\pi(s,a) = \max_{\pi_t} \mathbb{E}_{\pi_t(a|s)} Q_t^*(s,a)$$

Покажем, что эта верхняя оценка достигается. Сначала найдём π_t такую, что:

$$\begin{cases} \mathbb{E}_{\pi_t(a|s)} Q_t^*(s,a) \to \max\limits_{\pi_t} \\ \int\limits_{\mathcal{A}} \pi_t(a \mid s)\, \mathrm{d}a = 1; \end{cases} \qquad \forall a \in \mathcal{A}\colon \pi_t(a \mid s) \ge 0$$

Решением такой задачи, в частности*, будет детерминированная стратегия

$$\pi_t^*(s) \coloneqq \operatorname*{argmax}_a Q_t^*(s,a)$$

а сам максимум, соответственно, будет равняться $\max_a Q_t^*(s,a)$. Соответственно, $\max\limits_{\substack{\pi_t \\ \pi_{t+1} \\ \dots}} V_t^\pi(s)$ достигает верхней оценки при этой π_t^* и том наборе $\pi_{t+1}^*, \pi_{t+2}^* \dots$, на котором достигается значение $Q_t^*(s, \pi_t^*(s))$. ■

*здесь записана просто задача линейного программирования на симплексе (π_t обязано быть распределением); общим решением задачи, соответственно, будет любое распределение, которое размазывает вероятности между элементами множества $\operatorname*{Argmax}_a Q_t^*(s,a)$.

Утверждение 16: Для нестационарных оценочных функций верно:

$$Q_t^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} V_{t+1}^*(s') \tag{3.14}$$

Доказательство. Получим аналогично оценку сверху на $Q_t^*(s,a)$:

$$Q_t^*(s,a) = \max_{\substack{\pi_{t+1} \\ \pi_{t+2} \\ \dots}} Q_t^\pi(s,a) = \{\text{связь QV (3.10)}\} = \max_{\substack{\pi_{t+1} \\ \pi_{t+2} \\ \dots}} \left[r(s,a) + \gamma \mathbb{E}_{s'} V_{t+1}^\pi(s') \right] \leq$$

$$\leq \{\text{определение } V^* \text{ (3.11) }\} \leq r(s,a) + \gamma \mathbb{E}_{s'} V_{t+1}^*(s')$$

Эта верхняя оценка достигается на стратегии $\pi_t(s) = \operatorname*{argmax}_a Q_t^*(s,a)$, на которой, как мы доказали в предыдущей теореме 12, достигается максимум $V_t^\pi(s) = V^*(s)$ сразу для всех s одновременно. ■

Таким образом мы показали, что в нестационарном случае наши Q^* и V^* являются оценочными функциями оптимальных стратегий, максимизирующих награду из всех состояний. Осталось вернуться к стационарному случаю, то есть показать, что для стационарных стратегий выполняется то же утверждение.

Утверждение 17: Оптимальные оценочные функции для стационарных и нестационарных случаев совпадают, то есть, например, для V-функции:

$$\max_\pi V^\pi(s) = \max_{\substack{\pi_t \\ \pi_{t+1} \\ \dots}} V_t^\pi(s),$$

где в левой части максимум берётся по стационарным стратегиям, а в правой — по нестационарным.

Доказательство. По теореме 12 максимум справа достигается на детерминированной $\pi_t(s) = \operatorname*{argmax}_a Q_t^*(a,s)$. В силу утверждения 15, для всех моментов времени Q_t^* совпадают, следовательно такая π_t тоже совпадает для всех моментов времени и является стационарной стратегией. ■

Интуитивно: мы показали, что об MDP «с циклами в графе» можно думать как о дереве. Итак, в полученных результатах можно смело заменять все нестационарные оптимальные оценочные функции на стационарные.

3.1.9. Уравнения оптимальности Беллмана

Теорема 13 — Связь оптимальных оценочных функций:

$$V^*(s) = \max_a Q^*(s,a) \tag{3.15}$$

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} V^*(s') \tag{3.16}$$

Теперь V^* выражено через Q^* и наоборот. Значит, можно получить выражение для V^* через V^* и Q^* через Q^*:

Теорема 14 — *Уравнения оптимальности Беллмана* (Bellman optimality equation):

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q^*(s',a') \tag{3.17}$$

$$V^*(s) = \max_a [r(s,a) + \gamma \mathbb{E}_{s'} V^*(s')] \tag{3.18}$$

Доказательство. Подставили (3.15) в (3.16) и наоборот. ■

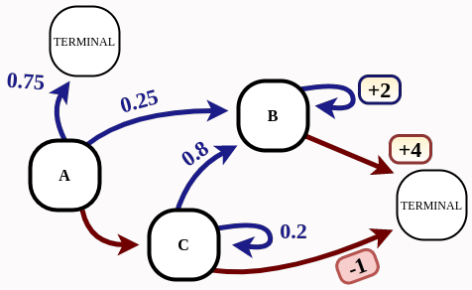
Хотя для строгого доказательства нам и пришлось поднапрячься и выписать относительно громоздкое рассуждение, уравнения оптимальности Беллмана крайне интуитивны. Для Q^*, например, можно рассудить так: что даст оптимальное поведение из состояния s после совершения действия a? Что с нами случится дальше: мы получим награду за этот выбор $r(s,a)$, на что уже повлиять не можем. Остальная награда будет дисконтирована. Затем среда переведёт нас в какое-то следующее состояние s' — нужно проматожидать по функции переходов. После этого мы, пользуясь принципом Беллмана, просто выберем то действие, которое позволит в будущем набрать наибольшую награду, и тогда сможем получить $\max\limits_{a'} Q^*(s',a')$.

Пример 51: Сформулируем для MDP с рисунка уравнения оптимальности Беллмана для V^*. Мы получим систему из трёх уравнений с трёмя неизвестными.

$$V^*(s=A) = \max\big(\underbrace{0.25\gamma V^*(s=B)}_{\blacksquare}, \underbrace{\gamma V^*(s=C)}_{\blacksquare}\big)$$

$$V^*(s=B) = \max\big(\underbrace{2+\gamma V^*(s=B)}_{\blacksquare}, \underbrace{4}_{\blacksquare}\big)$$

$$V^*(s=C) = \max\big(\underbrace{0.8\gamma V^*(s=B) + 0.2\gamma V^*(s=C)}_{\blacksquare}, \underbrace{-1}_{\blacksquare}\big)$$



Заметим, что в полученных уравнениях не присутствует мат.ожиданий по самим оптимальным стратегиям — предположение дальнейшей оптимальности поведения по сути «заменяет» их на взятие максимума по действиям. Более того, мы позже покажем, что оптимальные оценочные функции — единственные решения систем уравнений Беллмана. А значит, вместо поиска оптимальной стратегии можно искать оптимальные оценочные функции! Таким образом, мы свели задачу оптимизации нашего функционала к решению системы нелинейных уравнений особого вида. Беллман назвал данный подход «***динамическое программирование***» (dynamic programming).

3.1.10. Критерий оптимальности Беллмана

Давайте сформулируем критерий оптимальности стратегий в общей форме, описывающей вид всего множества оптимальных стратегий. Для доказательства нам понадобится факт, который мы технически докажем в рамках повествования чуть позже: для данного MDP Q^* — единственная функция $\mathcal{S}\times\mathcal{A}\to\mathbb{R}$, удовлетворяющая уравнениям оптимальности Беллмана.

Теорема 15 — Критерий оптимальности Беллмана: π оптимальна тогда и только тогда, когда $\forall s,a\colon \pi(a\mid s)>0$ верно:

$$a \in \operatorname*{Argmax}_a Q^\pi(s,a)$$

Необходимость. Пусть π — оптимальна. Тогда её оценочные функции совпадают с V^*, Q^*, для которых выполнено уравнение (3.15):

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

С другой стороны из связи VQ (3.6) верно $V^\pi(s) = \mathbb{E}_{\pi(a|s)} Q^\pi(s,a)$; получаем

$$\mathbb{E}_{\pi(a|s)} Q^\pi(s,a) = \max_a Q^\pi(s,a),$$

из чего вытекает доказываемое. ■

Достаточность. Пусть условие выполнено. Тогда для любой пары s,a:

$$Q^\pi(s,a) = \{\text{связь QQ (3.7)}\} = r(s,a) + \gamma\mathbb{E}_{s'}\mathbb{E}_{\pi(a'|s')} Q^\pi(s',a') = r(s,a) + \gamma\mathbb{E}_{s'}\max_{a'} Q^\pi(s',a')$$

Из единственности решения этого уравнения следует $Q^\pi(s,a) = Q^*(s,a)$, и, следовательно, π оптимальна. ■

Иначе говоря: теорема говорит, что оптимальны ровно те стратегии, которые пользуются принципом оптимальности Беллмана. Если в одном состоянии два действия позволят в будущем набрать максимальную награду, то между ними можно любым способом размазать вероятности выбора. Давайте при помощи этого критерия

окончательно ответим на вопросы о том, существует ли оптимальная стратегия и сколько их вообще может быть.

Утверждение 18: Если $|\mathcal{A}| < +\infty$, всегда существует оптимальная стратегия.

Доказательство. $\operatorname{Argmax}_a Q^*(s,a)$ для конечных множеств $\mathcal{A}$ всегда непуст, следовательно существует детерминированная оптимальная стратегия $\pi(s) \coloneqq \operatorname{argmax}_a Q^*(s,a)$. ■

Утверждение 19: Оптимальной стратегии может не существовать.

Контрпример. Одно состояние, $\mathcal{A} = [-1, 1]$, после первого выбора эпизод заканчивается; в качестве награды $r(a)$ можем рассмотреть любую не достигающую своего максимума функцию. Просто придумали ситуацию, когда $\operatorname{Argmax}_a Q^*(a)$ пуст. ■

Утверждение 20: Если существует хотя бы две различные оптимальные стратегии, то существует континуум оптимальных стратегий.

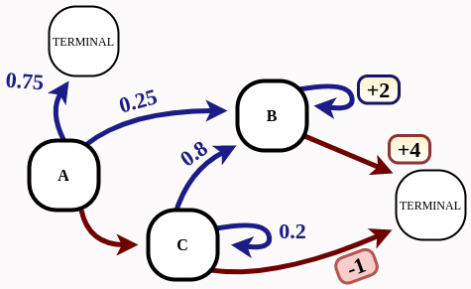
Доказательство. Существование двух различных оптимальных стратегий означает, что в каком-то состоянии s множество $\operatorname{Argmax}_a Q^*(s,a)$ содержит по крайней мере два элемента. Между ними можно размазать вероятности выбора любым способом и в любом случае получить максимальную награду. ■

Утверждение 21: Если существует хотя бы одна оптимальная стратегия, то существует детерминированная оптимальная стратегия.

Доказательство. Пусть π^* — оптимальна. Значит, $\operatorname{Argmax}_a Q^*(s,a)$ не пуст для всех s, и существует детерминированная оптимальная стратегия $\pi(s) \coloneqq \operatorname{argmax}_a Q^*(s,a)$. ■

Пример 52: Найдём все оптимальные стратегии в MDP из примера 51 для $\gamma = 0.5$.

Мы могли бы составить уравнения оптимальности Беллмана для Q^* и решать их, но сделаем чуть умнее и воспользуемся критерием оптимальности Беллмана (теорема 15). Например, в состоянии B оптимально или выбирать какое-то одно из двух действий с вероятностью 1, или действия эквивалентны, и тогда оптимально любое поведение. Допустим, мы будем выбирать всегда ■, тогда мы получим $\frac{2}{1-\gamma} = 4$; если же будем выбирать ■, то получим $+4$. Значит, действия эквивалентны, оптимально любое поведение, и $V^*(s = B) = 4$.



Проведём аналогичное рассуждение для состояния C. Если оптимально действие ■, то

$$Q^*(s = C, \blacksquare) = 0.2\gamma Q^*(s = C, \blacksquare) + 0.8\gamma V^*(s = B)$$

Решая это уравнение относительно неизвестного $Q^*(s = C, \blacksquare)$, получаем $\frac{16}{9} > Q^*(s = C, \blacksquare) = -1$. Значит, в C оптимальная стратегия обязана выбирать ■, и $V^*(C) = \frac{16}{9}$.

Для состояния A достаточно сравнить $Q^*(s = A, \blacksquare) = 0.25\gamma V^*(s = B) = \frac{1}{4}$ и $Q^*(s = A, \blacksquare) = \gamma V^*(s = C) = \frac{8}{9}$, определив, что оптимальная стратегия должна выбирать ■.

§3.2. Улучшение политики

3.2.1. Advantage-функция

Допустим, мы находились в некотором состоянии s, и засэмплировали $a \sim \pi(a \mid s)$ такое, что $Q^\pi(s,a) > V^\pi(s)$. Что можно сказать о таком действии? Мы знаем, что вообще в среднем политика π набирает из данного состояния $V^\pi(s)$, но какой-то выбор действий даст в итоге награду больше $V^\pi(s)$, а какой-то меньше. Если $Q^\pi(s,a) > V^\pi(s)$, то после того, как мы выбрали действие a, «приняли решение», наша средняя будущая награда вдруг увеличилась.

Мы ранее обсуждали в разделе 1.2.7 такую особую проблему обучения с подкреплением, как credit assingment, которая звучит примерно так: допустим, мы засэмплировали траекторию $s, a, s', a', \ldots$ до конца эпизода, и в конце в финальном состоянии через T шагов получили сигнал (награду) $+1$. Мы приняли T решений, но какое

из всех этих действий повлекло получение этого +1? «За что нас наградили?» Повлияло ли на получение +1 именно то действие $\boldsymbol{a}$, которое мы засэмплировали в стартовом $\boldsymbol{s}$? Вопрос нетривиальный, потому что в RL есть отложенный сигнал: возможно, именно действие $\boldsymbol{a}$ в состоянии $\boldsymbol{s}$ запустило какую-нибудь цепочку действий, которая дальше при любом выборе $\boldsymbol{a'}, \boldsymbol{a''}, \cdots$ приводит к награде +1. Возможно, конечно, что первое действие и не имело никакого отношения к этой награде, и это поощрение именно за последний выбор. А ещё может быть такое, что имело место везение, и просто среда в какой-то момент перекинула нас в удачное состояние.

Но мы понимаем, что если какое-то действие «затриггерило» получение награды через сто шагов, в промежуточных состояниях будет информация о том, сколько времени осталось до получения этой отложенной награды. Например, если мы выстрелили во вражеский инопланетный корабль, и через 100 шагов выстрел попадает во врага, давая агенту +1, мы будем видеть в состояниях расстояние от летящего выстрела до цели, и знать, что через такое-то время нас ждёт +1. Другими словами, вся необходимая информация лежит в идеальных оценочных функциях $\boldsymbol{Q^\pi}$ и $\boldsymbol{V^\pi}$.

Так, если в некотором состоянии $\boldsymbol{s}$ засэмплировалось такое $\boldsymbol{a}$, что $\boldsymbol{Q^\pi(s,a) = V^\pi(s)}$, то мы можем заключить, что выбор действия на этом шаге не привёл ни к какой «неожиданной» награде. Если же $\boldsymbol{Q^\pi(s,a) > > V^\pi(s)}$ — то мы приняли удачное решение, $\boldsymbol{Q^\pi(s,a) < V^\pi(s)}$ — менее удачное, чем обычно. Если, например, $\boldsymbol{r(s,a) + V^\pi(s') > Q^\pi(s,a)}$, то мы можем заключить, что имело место везение: среда засэмплировала такое $\boldsymbol{s'}$, что теперь мы получим больше награды, чем ожидали после выбора $\boldsymbol{a}$ в состоянии $\boldsymbol{s}$. И так далее: мы сможем отследить, в какой конкретно момент случилось то событие (сэмплирование действия или ответ среды), за счёт которого получена награда.

Таким образом, идеальный «кредит» влияния действия $\boldsymbol{a}$, выбранного в состоянии $\boldsymbol{s}$, на будущую награду равен

$$\boldsymbol{Q^\pi(s,a) - V^\pi(s)},$$

и именно эта величина на самом деле будет для нас ключевой. Поэтому из соображений удобства вводится ещё одно обозначение:

Определение 42: Для данного MDP ***Advantage-функцией*** политики $\boldsymbol{\pi}$ называется

$$\boldsymbol{A^\pi(s,a) \coloneqq Q^\pi(s,a) - V^\pi(s)} \tag{3.19}$$

Утверждение 22: Для любой политики $\boldsymbol{\pi}$ и любого состояния $\boldsymbol{s}$:

$$\boldsymbol{\mathbb{E}_{\pi(a|s)} A^\pi(s,a) = 0}$$

Доказательство.

$$\begin{aligned}\boldsymbol{\mathbb{E}_{\pi(a|s)} A^\pi(s,a) = \mathbb{E}_{\pi(a|s)} Q^\pi(s,a) - \mathbb{E}_{\pi(a|s)} V^\pi(s) =}\\ \boldsymbol{\{V^\pi \text{ не зависит от } a\} = \mathbb{E}_{\pi(a|s)} Q^\pi(s,a) - V^\pi(s) =}\\ \boldsymbol{\{\text{связь } V \text{ через } Q \text{ (3.6)}\} = V^\pi(s) - V^\pi(s) = 0}\end{aligned}$$

■

Утверждение 23: Для любой политики $\boldsymbol{\pi}$ и любого состояния $\boldsymbol{s}$:

$$\boldsymbol{\max_a A^\pi(s,a) \geq 0}$$

Advantage — это, если угодно, «центрированная» Q-функция. Если $\boldsymbol{A^\pi(s,a) > 0}$ — действие $\boldsymbol{a}$ «лучше среднего» для нашей текущей политики в состоянии $\boldsymbol{s}$, меньше нуля — хуже. И интуиция, что процесс обучения нужно строить на той простой идеи, что первые действия надо выбирать чаще, а вторые — реже, нас не обманывает.

Естественно, подвох в том, что на практике мы не будем знать точное значение оценочных функций, а значит, и истинное значение Advantage. Решая вопрос оценки значения Advantage для данной пары $\boldsymbol{s}, \boldsymbol{a}$, мы фактически будем проводить credit assingment — это одна и та же задача.

3.2.2. Relative Performance Identity (RPI)

Мы сейчас докажем одну очень интересную лемму, которая не так часто нам будет нужна в будущем, но которая прям открывает глаза на мир. Для этого вспомним формулу reward shaping-а (1.7) и заметим, что мы можем выбрать в качестве потенциала V-функцию произвольной стратегии $\boldsymbol{\pi_2}$:

$$\boldsymbol{\Phi(s) \coloneqq V^{\pi_2}(s)}$$

Действительно, требований к потенциалу два: ограниченность (для V-функций это выполняется в силу наших ограничений на рассматриваемые MDP) и равенство нулю в терминальных состояниях (для V-функций это

верно по определению). Подставив такой потенциал, мы получим связь между performance-ом $J(\pi) = V^{\pi}(s_0)$ двух разных стратегий. В общем виде лемма сравнивает V-функции двух стратегий в одном состоянии:

Теорема 16 — Relative Performance Identity: Для любых двух политик π_1, π_2:

$$V^{\pi_2}(s) - V^{\pi_1}(s) = \mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t A^{\pi_1}(s_t, a_t) \tag{3.20}$$

Доказательство.

$$\begin{aligned}
V^{\pi_2}(s) - V^{\pi_1}(s) &= \mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t r_t - V^{\pi_1}(s) = \\
&= \mathbb{E}_{\tau\sim\pi_2|s_0=s} \left[\sum_{t\geq 0} \gamma^t r_t - V^{\pi_1}(s_0)\right] = \\
\{\text{телескопирующая сумма (1.6)}\} &= \mathbb{E}_{\tau\sim\pi_2|s_0=s} \left[\sum_{t\geq 0} \gamma^t r_t + \sum_{t\geq 0} \left[\gamma^{t+1} V^{\pi_1}(s_{t+1}) - \gamma^t V^{\pi_1}(s_t)\right]\right] = \\
\{\text{перегруппируем слагаемые}\} &= \mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t \left(r_t + \gamma V^{\pi_1}(s_{t+1}) - V^{\pi_1}(s_t)\right) = \\
\{\text{фокус } \mathbb{E}_x f(x) = \mathbb{E}_x \mathbb{E}_x f(x)\} &= \mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t \left(r_t + \gamma \mathbb{E}_{s_{t+1}} V^{\pi_1}(s_{t+1}) - V^{\pi_1}(s_t)\right) = \\
\{\text{выделяем Q-функцию (3.5)}\} &= \mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t \left(Q^{\pi_1}(s_t, a_t) - V^{\pi_1}(s_t)\right) \\
\{\text{по определению (3.19)}\} &= \mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t A^{\pi_1}(s_t, a_t)
\end{aligned}$$

■

Мы смогли записать наш функционал как мат.ожидание по траекториям, сгенерированным одной политикой, по оценочной функции другой стратегии. Фактически, мы можем награду заменить Advantage-функцией произвольной другой стратегии, и это сдвинет оптимизируемый функционал на константу! Прикольно.

Конечно, это теоретическое утверждение, поскольку на практике узнать точно оценочную функцию какой-то другой стратегии достаточно сложно (хотя ничто не мешает в качестве потенциала использовать произвольную функцию, приближающую $V^{\pi_1}(s)$). Однако в этой «новой» награде замешаны сигналы из будущего, награды, которые будут получены через много шагов, и эта «новая» награда априори информативнее исходной $r(s, a)$.

Представим, что мы оптимизировали исходный функционал

$$\mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t r(s_t, a_t) \to \max_{\pi_2}$$

и сказали: слушайте, мы не знаем, как управлять марковской цепью, не очень понимаем, как выбор тех или иных действий в состоянии влияет на структуру траектории $p(\tau \mid \pi_2)$. А давайте мы притворимся, что у нас нет в задаче отложенного сигнала (что очень существенное упрощение), и будем просто во всех состояниях s оптимизировать $r(s, a)$: выбирать «хорошие» действия a, где функция награды высокая. То есть будем просто выбирать $\pi_2(s) = \operatorname{argmax}_a r(s, a)$. Смысла в этом будет мало.

Теперь же мы преобразовали функционал, сменив функцию награды:

$$\mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t A^{\pi_1}(s_t, a_t) \to \max_{\pi_2}$$

Что, если мы поступим также с новой наградой? Мы, например, знаем, что Advantage — не произвольная функция, и она обязана в среднем равняться нулю (утв. 22). Значит, если мы выберем

$$\pi_2(s) = \operatorname*{argmax}_a A^{\pi_1}(s, a),$$

то все встречаемые пары (s, a) в траекториях из π_2 будут обязательно с неотрицательными наградами за шаг $A^{\pi_1}(s, a) \geq 0$. Значит и вся сумма наград будет положительна для любого стартового состояния:

$$\mathbb{E}_{\tau\sim\pi_2|s_0=s} \sum_{t\geq 0} \gamma^t \underbrace{A^{\pi_1}(s_t, a_t)}_{\geq 0} \geq 0$$

И тогда из теоремы 16 об RPI мы можем заключить, что для любого s:

$$V^{\pi_2}(s) - V^{\pi_1}(s) \geq 0$$

Это наблюдение - ключ к оптимизации стратегии при известной оценочной функции другой стратегии.

3.2.3. Policy Improvement

Определение 43: Будем говорить, что стратегия π_2 «*не хуже*» π_1 (запись: $\pi_2 \succeq \pi_1$), если $\forall s$:

$$V^{\pi_2}(s) \geq V^{\pi_1}(s),$$

и *лучше* (запись: $\pi_2 \succ \pi_1$), если также найдётся s, для которого неравенство выполнено строго:

$$V^{\pi_2}(s) > V^{\pi_1}(s)$$

Мы ввели частичный порядок на множестве стратегий (понятно, что можно придумать две стратегии, которые будут «не сравнимы»: когда в одном состоянии одна будет набирать больше второй, в другом состоянии вторая будет набирать больше первой).

Зададимся следующим вопросом. Пусть для стратегии π_1 мы знаем оценочную функцию Q^{π_1}; тогда мы знаем и V^{π_1} из VQ уравнения (3.6) и A^{π_1} по определению (3.19). Давайте попробуем построить $\pi_2 \succ \pi_1$. Для этого покажем более «классическим» способом, что стратегии π_2 достаточно лишь в среднем выбирать действия, дающие неотрицательный Advantage стратегии π_1, чтобы быть не хуже.

Теорема 17 — Policy Improvement: Пусть стратегии π_1 и π_2 таковы, что для всех состояний s выполняется:

$$\mathbb{E}_{\pi_2(a|s)} Q^{\pi_1}(s,a) \geq V^{\pi_1}(s),$$

или, в эквивалентной форме:

$$\mathbb{E}_{\pi_2(a|s)} A^{\pi_1}(s,a) \geq 0.$$

Тогда $\pi_2 \succeq \pi_1$; если хотя бы для одного s неравенство выполнено строго, то $\pi_2 \succ \pi_1$.

Доказательство. Покажем, что $V^{\pi_2}(s) \geq V^{\pi_1}(s)$ для любого s:

$$\begin{aligned}
V^{\pi_1}(s) &= \{\text{связь VQ (3.6)}\} = \mathbb{E}_{\pi_1(a|s)} Q^{\pi_1}(s,a) \leq \\
&= \{\text{по построению } \pi_2\} = \mathbb{E}_{\pi_2(a|s)} Q^{\pi_1}(s,a) = \\
&= \{\text{связь QV (3.5)}\} = \mathbb{E}_{\pi_2(a|s)} \left[r + \gamma \mathbb{E}_{s'} V^{\pi_1}(s')\right] \leq \\
&\leq \{\text{применяем это же неравенство рекурсивно}\} = \mathbb{E}_{\pi_2(a|s)} \left[r + \mathbb{E}_{s'} \mathbb{E}_{\pi_2(a'|s')} \left[\gamma r' + \gamma^2 \mathbb{E}_{s''} V^{\pi_1}(s'')\right]\right] \leq \\
&\leq \{\text{раскручиваем цепочку далее}\} \leq \cdots \leq \mathbb{E}_{\mathcal{T} \sim \pi_2 | s_0 = s} \sum_{t \geq 0} \gamma^t r_t = \\
&= \{\text{по определению (3.2)}\} = V^{\pi_2}(s)
\end{aligned}$$

Если для какого-то s неравенство из условия теоремы было выполнено строго, то для него первое неравенство в этой цепочке рассуждений выполняется строго, и, значит, $V^{\pi_2}(s) > V^{\pi_1}(s)$. ■

Что означает эта теорема? Знание оценочной функции позволяет улучшить стратегию. Улучшать стратегию можно прямо в отдельных состояниях, например, выбрав некоторое состояние s и сказав: неважно, как это повлияет на частоты посещения состояний, но будем конкретно в этом состоянии s выбирать действия так, что значение

$$\mathbb{E}_{\pi_2(a|s)} Q^{\pi_1}(s,a) \tag{3.21}$$

как можно больше. Тогда, если в s действие выбирается «новой» стратегией π_2, а в будущем агент будет вести себя *не хуже*, чем π_1, то и наберёт он в будущем не меньше $Q^{\pi_1}(s,a)$. Доказательство теоремы 17 показывает, что выражение (3.21) является нижней оценкой на награду, которую соберёт «новый» агент со стратегией π_2.

Если эта нижняя оценка поднята выше $V^{\pi_1}(s)$, то стратегию удалось улучшить: и тогда какой бы ни была π_1, мы точно имеем гарантии $\pi_2 \succeq \pi_1$. Важно, что такой policy improvement работает всегда: и для «тупых» стратегий, близких к случайному поведению, и для уже умеющих что-то разумное делать.

В частности, мы можем попробовать нижнюю оценку (3.21) максимально поднять, то есть провести *жадный* (greedy) policy improvement. Для этого мы формально решаем такую задачу оптимизации:

$$\mathbb{E}_{\pi_2(a|s)} Q^{\pi_1}(s,a) \to \max_{\pi_2},$$

и понятно, что решение находится в детерминированной π_2:

$$\pi_2(s) = \operatorname*{argmax}_a Q^{\pi_1}(s,a) = \operatorname*{argmax}_a A^{\pi_1}(s,a)$$

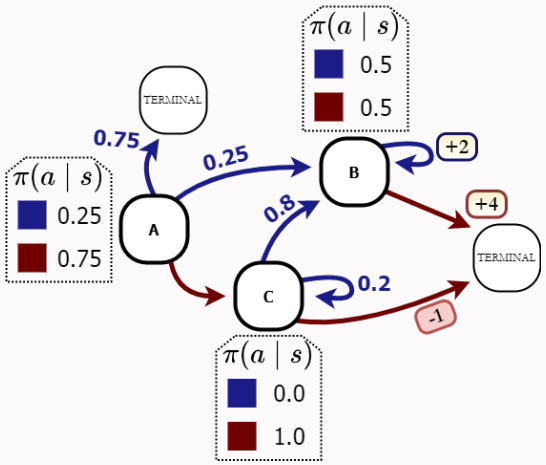
Конечно, мы так не получим «за один ход» сразу оптимальную стратегию, поскольку выбор $\pi_2(a \mid s)$ сколь угодно хитро может изменить распределение траекторий, но тем не менее.

Пример 53: Попробуем улучшить стратегию π из примера 46, $\gamma = 0.8$. Например, в состоянии C она выбирает ■ с вероятностью 1 и получает -1; попробуем посчитать $Q^\pi(s = C, ■)$:

$$Q^\pi(s = C, ■) = 0.2\gamma V^\pi(C) + 0.8\gamma V^\pi(B)$$

Подставляя ранее подсчитанные $V^\pi(C) = -1, V^\pi(B) = 5$, видим, что действие ■ принесло бы нашей стратегии π куда больше -1, а именно $Q^\pi(s = C, ■) = 3.04$. Давайте построим π_2, скопировав π в A и B, а в C будем с вероятностью 1 выбирать ■.

Что говорит нам теория? Важно, что она не даёт нам значение $V^{\pi_2}(C)$; в частности, нельзя утверждать, что $Q^{\pi_2}(s = C, ■) = 3.04$, и повторение вычислений подтвердит, что это не так. Однако у нас есть гарантии, что, во-первых, $Q^{\pi_2}(s = C, ■) \geq 3.04$, и, что важнее, из состояния C мы начали набирать больше награды: $V^{\pi_2}(C) > V^{\pi_1}(C)$ строго. Во-вторых, есть гарантии, что мы не «сломали» стратегию в других состояниях: во всех остальных состояниях гарантированно $V^{\pi_2}(s) \geq V^{\pi_1}(s)$. Для Q-функции, как можно показать, выполняются аналогичные неравенства.



3.2.4. Вид оптимальной стратегии (доказательство через PI)

Что, если для некоторой π_1 мы «не можем» провести Policy Improvement? Под этим будем понимать, что мы не можем выбрать π_2 так, что $\mathbb{E}_{\pi_2(a|s)} Q^{\pi_1}(s, a) > V^{\pi_1}(s)$ строго хотя бы для одного состояния s (ну, равенства в любом состоянии s мы добьёмся всегда, скопировав $\pi_1(\cdot \mid s)$). Такое может случиться, если и только если π_1 удовлетворяет следующему свойству:

$$\max_a Q^{\pi_1}(s, a) = V^{\pi_1}(s) \quad \Leftrightarrow \quad \max_a A^{\pi_1}(s, a) = 0$$

Но это в точности критерий оптимальности Беллмана, теорема 15! Причём мы можем, воспользовавшись теоремами RPI 16 и о Policy Improvement 17, теперь доказать этот критерий альтернативным способом, не прибегая к формализму оптимальных оценочных функций[2] и не требуя рассуждения про отказ от стационарности и обоснования единственности решения уравнений оптимальности Беллмана.

Теорема 18 — Критерий оптимальности (альт. доказательство): π оптимальна тогда и только тогда, когда $\forall s\colon \max_a A^\pi(s, a) = 0$.

Достаточность. Допустим, это не так, и существует $\pi_2, s\colon V^{\pi_2}(s) > V^\pi(s)$. Тогда по RPI (3.20)

$$\mathbb{E}_{\mathcal{T} \sim \pi_2 | s_0 = s} \sum_{t \geq 0} \gamma^t A^\pi(s_t, a_t) > 0,$$

однако все слагаемые в сумме неположительны. Противоречие. ■

Необходимость. Допустим, что π оптимальна, но для некоторого $\hat{s}$ условие не выполнено, и $\max_a A^\pi(\hat{s}, a) > 0$ (меньше нуля он, ещё раз, быть не может в силу утв. 23). Рассмотрим детерминированную π_2, которая в состоянии $\hat{s}$ выбирает какое-нибудь $\hat{a}$, такое что $A^\pi(\hat{s}, \hat{a}) > 0$ (это можно сделать по условию утверждения — сам максимум может вдруг оказаться недостижим для сложных пространств действий, но какое-то действие с положительным advantage-ем мы найдём), а в остальных состояниях выбирает какое-нибудь действие, т.ч. advantage-функция неотрицательна. Тогда

$$V^{\pi_2}(\hat{s}) - V^\pi(\hat{s}) = \mathbb{E}_{\mathcal{T} \sim \pi_2 | s_0 = \hat{s}} \sum_{t \geq 0} \gamma^t A^\pi(s_t, a_t) > 0$$

поскольку все слагаемые неотрицательны, и во всех траекториях с вероятностью* 1 верно $s_0 = \hat{s}, a_0 = \hat{a}$, то есть первое слагаемое равно $A(\hat{s}, \hat{a}) > 0$. ■

[2] в доказательствах RPI и Policy Improvement мы не использовали понятия Q^* и V^* и их свойства; тем не менее, из этих теорем все свойства оптимальных оценочных функций следуют: например, пусть A^*, Q^*, V^* — оценочные функции оптимальных стратегий, тогда в силу выводимого из RPI критерия оптимальности (теорема 18) $\forall s\colon \max_a A^*(s, a) = 0$, или, что тоже самое, $\max_a [Q^*(s, a) - V^*(s)] = 0$; отсюда $V^*(s) = \max_a Q^*(s, a)$. Аналогично достаточно просто можно получить все остальные утверждения об оптимальных оценочных функциях, не прибегая к рассуждению с отказом от стационарности.

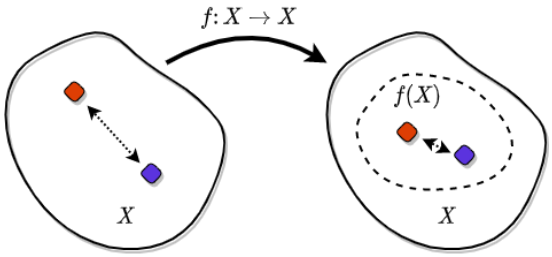
[*] мы специально стартовали из $\hat{s}$, чтобы пары $\hat{s}, \hat{a}$ «встретились» в траекториях, иначе могло бы быть такое, что агент в это состояние $\hat{s}$ «никогда не попадает», и отделиться от нуля не получилось бы.

Итак, мораль полученных результатов такая: зная Q^π, мы можем придумать стратегию лучше. Не можем — значит, наша текущая стратегия π уже оптимальная.

§3.3. Динамическое программирование

3.3.1. Метод простой итерации

Мы увидели, что знание оценочных функций открывает путь к улучшению стратегии. Напрямую по определению считать их затруднительно; попробуем научиться решать уравнения Беллмана. И хотя уравнения оптимальности Беллмана нелинейные, они, тем не менее, имеют весьма определённый вид и, как мы сейчас увидим, обладают очень приятными свойствами. Нам понадобится несколько понятий внезапно из функана о том, как решать системы нелинейных уравнений вида $x = f(x)$.



Определение 44: Оператор $f: X \to X$ называется ***сжимающим*** (contraction) с коэффициентом сжатия $\gamma < 1$ по некоторой метрике ρ, если $\forall x_1, x_2$:

$$\rho(f(x_1), f(x_2)) < \gamma \rho(x_1, x_2)$$

Определение 45: Точка $x \in X$ для оператора $f: X \to X$ называется ***неподвижной*** (fixed point), если

$$x = f(x)$$

Определение 46: Построение последовательности $x_{k+1} = f(x_k)$ для начального приближения $x_0 \in X$ называется методом ***простой итерации*** (point iteration) решения уравнения $x = f(x)$.

Теорема 19 — Теорема Банаха о неподвижной точке: В полном[*] метрическом пространстве X у сжимающего оператора $f: X \to X$ существует и обязательно единственна неподвижная точка x^*, причём метод простой итерации сходится к ней из любого начального приближения.

Сходимость метода простой итерации. Пусть x_0 — произвольное, $x_{k+1} = f(x_k)$. Тогда для любого $k > 0$:

$$\begin{aligned} \rho(x_k, x_{k+1}) &= \{\text{определение } x_k\} = \rho(f(x_{k-1}), f(x_k)) \le \\ &\le \{\text{свойство сжатия}\} \le \gamma \rho(x_{k-1}, x_k) \le \dots \le \\ &\le \{\text{аналогичным образом}\} \le \dots \le \gamma^k \rho(x_0, x_1) \end{aligned} \tag{3.22}$$

Теперь посмотрим, что произойдёт после применения оператора f n раз:

$$\begin{aligned} &\rho(x_k, x_{k+n}) \le \\ &\{\text{неравенство треугольника}\} \le \rho(x_k, x_{k+1}) + \rho(x_{k+1}, x_{k+2}) + \dots + \rho(x_{k+n-1}, x_{k+n}) \le \\ &\le \{(3.22)\} \le (\gamma^k + \gamma^{k+1} + \dots \gamma^{k+n-1}) \rho(x_0, x_1) \le \\ &\le \{\text{геом. прогрессия}\} \le \frac{\gamma^k}{1-\gamma} \rho(x_0, x_1) \xrightarrow{k \to \infty} 0 \end{aligned}$$

Итак, последовательность x_k — фундаментальная, и мы специально попросили такое метрическое пространство («полное»), в котором обязательно найдётся предел $x^* := \lim\limits_{k \to \infty} x_k$. ■

Существование неподвижной точки. Покажем, что x^* и есть неподвижная точка f, то есть покажем, что наш метод простой итерации конструктивно её построил. Заметим, что для любого $k > 0$:

$$\begin{aligned} &\rho(x^*, f(x^*)) \le \\ &\le \{\text{неравенство треугольника}\} \le \rho(x^*, x_k) + \rho(x_k, f(x^*)) = \\ &= \{\text{определение } x_k\} = \rho(x^*, x_k) + \rho(f(x_{k-1}), f(x^*)) \le \\ &\le \{\text{свойство сжатия}\} \le \rho(x^*, x_k) + \gamma \rho(x_{k-1}, x^*) \end{aligned}$$

Устремим $k \to \infty$; слева стоит константа, не зависящая от k. Тогда расстояние между x_k и x^* устремится к нулю, ровно как и между x_{k-1}, x^* поскольку x^* — предел x_k. Значит, константа равна нулю, $\rho(x^*, f(x^*)) = 0$, следовательно, $x^* = f(x^*)$. ■

Единственность. Пусть x_1, x_2 — две неподвижные точки оператора f. Ну тогда:

$$\rho(x_1, x_2) = \rho(f(x_1), f(x_2)) \leq \gamma\rho(x_1, x_2)$$

Получаем, что такое возможно только при $\rho(x_1, x_2) = 0$, то есть только если x_1 и x_2 совпадают. ■

*любая фундаментальная последовательность имеет предел

3.3.2. Policy Evaluation

Вернёмся к RL. Известно, что V^π для данного MDP и фиксированной политики π удовлетворяет уравнению Беллмана (3.3). Для нас это система уравнений относительно значений $V^\pi(s)$. $V^\pi(s)$ — объект (точка) в функциональном пространстве $\mathcal{S} \to \mathbb{R}$.

Будем решать её методом простой итерации[3]. Для этого определим оператор $\mathfrak{B}$, то есть преобразование из одной функции $\mathcal{S} \to \mathbb{R}$ в другую. На вход этот оператор принимает функцию $V : \mathcal{S} \subseteq \mathbb{R}^n$ и выдаёт некоторую другую функцию от состояний $\mathfrak{B}V$. Чтобы задать выход оператора, нужно задать значение выходной функции в каждом $s \in \mathcal{S}$; это значение мы будем обозначать $[\mathfrak{B}V](s)$ (квадратные скобки позволяют не путать применение оператора с вызовом самой функции) и определим его как правую часть решаемого уравнения (3.3). Итак:

Определение 47: Введём ***оператор Беллмана*** (Bellman operator) для заданного MDP и стратегии π как

$$[\mathfrak{B}V](s) \coloneqq \mathbb{E}_{a\sim\pi(a|s)}\left[r(s,a) + \gamma\mathbb{E}_{s'}V(s')\right] \tag{3.23}$$

Также нам нужна метрика на множестве функций $\mathcal{S} \to \mathbb{R}$; возьмём

$$d_\infty(V_1, V_2) \coloneqq \max_s |V_1(s) - V_2(s)|$$

Теорема 20: Если $\gamma < 1$, оператор $\mathfrak{B}$ — сжимающий с коэффициентом сжатия γ.

Доказательство.

$$\begin{aligned}
&d_\infty(\mathfrak{B}V_1, \mathfrak{B}V_2) = \max_s |[\mathfrak{B}V_1](s) - [\mathfrak{B}V_2](s)| = \\
&= \{\text{подставляем значение операторов, т.е. правые части решаемого уравнения}\} = \\
&= \max_s |\mathbb{E}_a\left[r(s,a) + \gamma\mathbb{E}_{s'}V_1(s')\right] - \mathbb{E}_a\left[r(s,a) + \gamma\mathbb{E}_{s'}V_2(s')\right]| = \\
&= \{\text{слагаемые } r(s,a) \text{ сокращаются}\} = \\
&= \gamma \max_s |\mathbb{E}_a\mathbb{E}_{s'}\left[V_1(s') - V_2(s')\right]| \leq \\
&\leq \{\text{используем свойство } \mathbb{E}_x f(x) \leq \max_x f(x)\} \leq \\
&\leq \gamma \max_s \max_{s'} |V_1(s') - V_2(s')| = \gamma d_\infty(V_1, V_2)
\end{aligned}$$

■

Итак, мы попали в теорему Банаха, и значит, метод простой итерации

$$V_{k+1} \coloneqq \mathfrak{B}V_k$$

гарантированно сойдётся к единственной неподвижной точке при любой стартовой инициализации V_0. По построению мы знаем, что $V^\pi(s)$ такова, что $\mathfrak{B}V^\pi = V^\pi$ (это и есть уравнение Беллмана), поэтому к ней и придём.

Важно помнить, что на каждой итерации такой процедуры текущее приближение не совпадает с истинной оценочной функцией: $V_k(s) \approx V^\pi(s)$, но точного равенства поставить нельзя. Распространено (но, к сожалению, не везде применяется) соглашение обозначать аппроксимации оценочных функций без верхнего индекса: просто V или Q. Однако, иногда, чтобы подчеркнуть, что алгоритм учит именно V^π, верхний индекс оставляют, что может приводить к путанице.

Обсудим, что случится в ситуации, когда $\gamma = 1$; напомним, что в таких ситуациях мы требовали эпизодичность сред, с гарантиями завершения всех эпизодов за $T^{\max}$ шагов. Оператор Беллмана формально сжатием являться уже не будет, и мы не подпадаем под теорему, поэтому этот случай придётся разобрать отдельно.

Теорема 21: В эпизодичных средах метод простой итерации сойдётся к единственному решению уравнений Беллмана не более чем за $T^{\max}$ шагов даже при $\gamma = 1$.

[3] вообще говоря, это система линейных уравнений относительно значений $V^\pi(s)$, которую в случае табличных MDP можно решать любым методом решения СЛАУ. Однако, дальнейшие рассуждения через метод простой итерации обобщаются, например, на случай непрерывных пространств состояний $\mathcal{S} \subseteq \mathbb{R}^n$.

Доказательство. Мы уже доказывали теорему 2, что граф таких сред является деревом. Будем говорить, что состояние s находится на ярусе T, если при старте из s у любой стратегии есть гарантии завершения за T шагов. Понятно, что для состояния s на ярусе T верно, что $\forall s', a$, для которых $p(s' \mid s, a) > 0$, ярус s' не превосходит $T - 1$.

Осталось увидеть, что на k-ой итерации метода простой итерации вычисляет точные значения $V_k(s) = V^\pi(s)$ для всех состояний на ярусах до k: действительно, покажем по индукции. Считаем, что терминальные состояния имеют нулевой ярус; а на k-ом шаге при обновлении $V_{k+1}(s) := [\mathfrak{B} V_k](s)$ для s на k-ом ярусе в правой части уравнения Беллмана будет стоять мат. ожидание по s' с ярусов до $k-1$-го, для которых значение по предположению индукции уже посчитано точно.

Соответственно, за $T^{\max}$ шагов точные значения распространятся на все состояния, и конструктивно значения определены однозначно. ■

Если $\gamma = 1$, а среда неэпизодична (такие MDP мы не допускали к рассмотрению), метод простой итерации может не сойтись, а уравнения Беллмана могут в том числе иметь бесконечно много решений. Пример подобного безобразия. Пусть в MDP без терминальных состояний с нулевой функцией награды (где, очевидно, $V^\pi(s) = 0$ для всех π, s) мы проинициализировали $V_0(s) = 100$ во всех состояниях s. Тогда при обновлении наша аппроксимация не будет меняться: мы уже в неподвижной точке уравнений Беллмана. В частности поэтому на практике практически никогда не имеет смысл выставлять $\gamma = 1$, особенно в сложных средах, где, может быть, даже и есть эпизодичность, но, тем не менее, есть «похожие состояния»: они начнут работать «как петли», когда мы перейдём к приближённым методам динамического программирования в дальнейшем.

Утверждение 24: Если некоторая функция $\tilde{V} : \mathcal{S} \to \mathbb{R}$ удовлетворяет уравнению Беллмана (3.3), то $\tilde{V} \equiv V^\pi$.

Мы научились решать задачу ***оценивания стратегии*** (Policy Evaluation): вычислять значения оценочной функции по данной стратегии π в ситуации, когда мы знаем динамику среды. На практике мы можем воспользоваться этим результатом только в ***«табличном» случае*** (tabular RL), когда пространство состояний и пространство действий конечны и достаточно малы, чтобы все пары состояние-действие было возможно хранить в памяти компьютера и перебирать за разумное время. В такой ситуации $V^\pi(s)$ — конечный векторочек, и мы умеем считать оператор Беллмана и делать обновления $V_{k+1} = \mathfrak{B} V_k$.

Алгоритм 6: Policy Evaluation

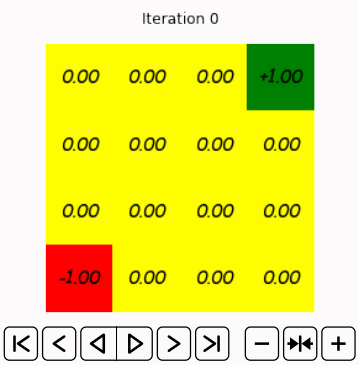
Вход: $\pi(a \mid s)$ — стратегия
Гиперпараметры: ε — критерий останова

Инициализируем $V_0(s)$ произвольно для всех $s \in \mathcal{S}$
На k-ом шаге:

1. $\forall s$: $V_{k+1}(s) := \mathbb{E}_a \left[r(s, a) + \gamma \mathbb{E}_{s'} V_k(s') \right]$
2. **критерий останова:** $\max_s |V_k(s) - V_{k+1}(s)| < \varepsilon$

Выход: $V_k(s)$

Пример 54: Проведём оценивание стратегии, случайно выбирающей, в какую сторону ей пойти, с $\gamma = 0.9$. Угловые клетки с ненулевой наградой терминальны; агент остаётся в той же клетке, если упирается в стенку. На каждой итерации отображается значение текущего приближения $V_k(s) \approx V^\pi(s)$.



Итак, мы научились считать V^π в предположении известной динамики среды. Полностью аналогичное рассуждение верно и для уравнений QQ (3.7); то есть, расширив набор переменных, в табличных MDP можно

методом простой итерации находить Q^π и «напрямую». Пока модель динамики среды считается известной, это не принципиально: мы можем посчитать и Q-функцию через V-функцию по формуле QV (3.5).

3.3.3. Value Iteration

Теорема Банаха позволяет аналогично Policy Evaluation (алг. 6) решать уравнения оптимальности Беллмана (3.17) через метод простой итерации. Действительно, проведём аналогичные рассуждения (мы сделаем это для Q^*, но совершенно аналогично можно было бы сделать это и для V^*):

Определение 48: Определим ***оператор оптимальности Беллмана*** (Bellman optimality operator, Bellman control operator) $\mathfrak{B}^*$:

$$[\mathfrak{B}^* Q](s,a) \coloneqq r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q(s',a')$$

В качестве метрики на множестве функций $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ аналогично возьмём

$$d_\infty(Q_1, Q_2) \coloneqq \max_{s,a} |Q_1(s,a) - Q_2(s,a)|$$

Нам понадобится следующий факт:

Утверждение 25:

$$|\max_x f(x) - \max_x g(x)| \le \max_x |f(x) - g(x)| \tag{3.24}$$

Доказательство. Рассмотрим случай $\max_x f(x) > \max_x g(x)$. Пусть x^* — точка максимума $f(x)$. Тогда:

$$\max_x f(x) - \max_x g(x) = f(x^*) - \max_x g(x) \le f(x^*) - g(x^*) \le \max_x |f(x) - g(x)|$$

Второй случай рассматривается симметрично. ■

Теорема 22: Если $\gamma < 1$, оператор $\mathfrak{B}^*$ — сжимающий.

Доказательство.

$$\begin{aligned}
&d_\infty(\mathfrak{B}^* Q_1, \mathfrak{B}^* Q_2) = \max_{s,a} |[\mathfrak{B}^* Q_1](s,a) - [\mathfrak{B}^* Q_2](s,a)| = \\
&= \{\text{подставляем значения операторов, т.е. правые части решаемой системы уравнений}\} = \\
&= \max_{s,a} \left| \left[r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q_1(s',a') \right] - \left[r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q_2(s',a') \right] \right| = \\
&= \{\text{слагаемые } r(s,a) \text{ сокращаются}\} = \\
&= \gamma \max_{s,a} \left| \mathbb{E}_{s'} \left[\max_{a'} Q_1(s',a') - \max_{a'} Q_2(s',a') \right] \right| \le \\
&\le \{\text{используем свойство } \mathbb{E}_x f(x) \le \max_x f(x)\} \le \\
&\le \gamma \max_{s,a} \max_{s'} \left| \max_{a'} Q_1(s',a') - \max_{a'} Q_2(s',a') \right| = \\
&\le \{\text{используем свойство максимумов (3.24)}\} \le \\
&\le \gamma \max_{s,a} \max_{s'} \max_{a'} |Q_1(s',a') - Q_2(s',a')| \le \\
&= \{\text{внутри стоит определение } d_\infty(Q_1,Q_2), \text{ а от внешнего максимума ничего не зависит}\} = \\
&= \gamma d_\infty(Q_1, Q_2)
\end{aligned}$$

■

Теорема 23: В эпизодичных средах метод простой итерации сойдётся к единственному решению уравнений оптимальности Беллмана не более чем за $T^{\max}$ шагов даже при $\gamma = 1$.

Доказательство. Полностью аналогично доказательству теоремы 21. ■

Утверждение 26: Если некоторая функция $\tilde{Q}\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ удовлетворяет уравнению оптимальности Беллмана (3.17), то $\tilde{Q} \equiv Q^*$.

Утверждение 27: Метод простой итерации сходится к Q^* из любого начального приближения.

Вообще, если известна динамика среды, то нам достаточно решить уравнения оптимальности для V^* — это потребует меньше переменных. Итак, в табличном случае мы можем напрямую методом простой итерации

решать уравнения оптимальности Беллмана и в пределе сойдёмся к оптимальной оценочной функции, которая тут же даёт нам оптимальную стратегию.

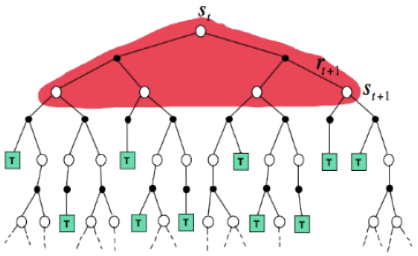
Алгоритм 7: Value Iteration

Вход: ε — критерий останова

Инициализируем $V_0(s)$ произвольно для всех $s \in \mathcal{S}$
На k-ом шаге:

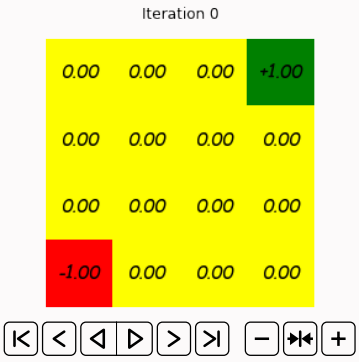
1. для всех s: $V_{k+1}(s) := \max\limits_a [r(s,a) + \gamma \mathbb{E}_{s'} V_k(s')]$
2. **критерий останова:** $\max\limits_s |V_{k+1}(s) - V_k(s)| < \varepsilon$

Выход: $\pi(s) := \operatorname{argmax}\limits_a [r(s,a) + \gamma \mathbb{E}_{s'} V(s')]$



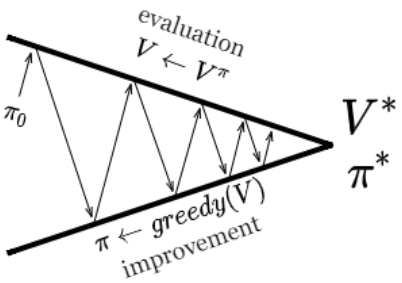
Итак, мы придумали наш первый табличный алгоритм планирования — алгоритм, решающий задачу RL в условиях известной модели среды. На каждом шаге мы обновляем («бэкапим») нашу текущую аппроксимацию V-функции на её ***одношаговое приближение*** (one-step approximation): смотрим на один шаг в будущее (a, r, s') и приближаем всё остальное будущее текущей же аппроксимацией. Такой «бэкап динамического программирования» (dynamic programming backup, DP-backup) — обновление «бесконечной ширины»: мы должны перебрать все возможные варианты следующего одного шага, рассмотреть все свои действия (по ним мы возьмём максимум) и перебрать всевозможные ответы среды — s' (по ним мы должны рассчитать мат.ожидание). Поэтому этот алгоритм в чистом виде напоминает то, что обычно и понимается под словами «динамическое программирование»: мы «раскрываем дерево игры» полностью на один шаг вперёд.

Пример 55: Решим задачу из примера 54, $\gamma = 0.9$; на каждой итерации отображается значение текущего приближения $V_k(s) \approx V^*(s)$. В конце концов в силу детермнированности среды станет понятно, что можно избежать попадания в терминальное -1 и кратчайшим путём добираться до терминального +1.



3.3.4. Policy Iteration

Мы сейчас в некотором смысле «обобщим» Value Iteration и придумаем более общую схему алгоритма планирования для табличного случая.



Для очередной стратегии π_k посчитаем её оценочную функцию Q^{π_k}, а затем воспользуемся теоремой Policy Improvement 17 и построим стратегию лучше; например, жадно:

$$\pi_{k+1}(s) := \operatorname*{argmax}_a Q^{\pi_k}(s,a)$$

Тогда у нас есть второй алгоритм планирования, который, причём, перебирает детерминированные стратегии, обладающие свойством монотонного возрастания качества: каждая следующая стратегия не хуже предыдущей. Он работает сразу в классе детерминированных стратегий, и состоит из двух этапов:

- **Policy Evaluation**: вычисление Q^π для текущей стратегии π;
- **Policy Improvement**: улучшение стратегии $\pi(s) \leftarrow \operatorname*{argmax}\limits_a Q^\pi(s,a)$;

При этом у нас есть гарантии, что когда алгоритм «останавливается» (не может провести Policy Improvement), то он находит оптимальную стратегию. Будем считать[4], что в такой момент остановки после проведения Policy Improvement наша стратегия не меняется: $\pi_{k+1} \equiv \pi_k$.

Теорема 24: В табличном сеттинге Policy Iteration завершает работу за конечное число итераций.

Доказательство. Алгоритм перебирает детерминированные стратегии, и, если остановка не происходит, каждая следующая лучше предыдущей:

$$\pi_k \succ \pi_{k-1} \succ \cdots \succ \pi_0$$

Это означает, что все стратегии в этом ряду различны. Поскольку в табличном сеттинге число состояний и число действий конечны, детерминированных стратегий конечное число; значит, процесс должен закончится. ■

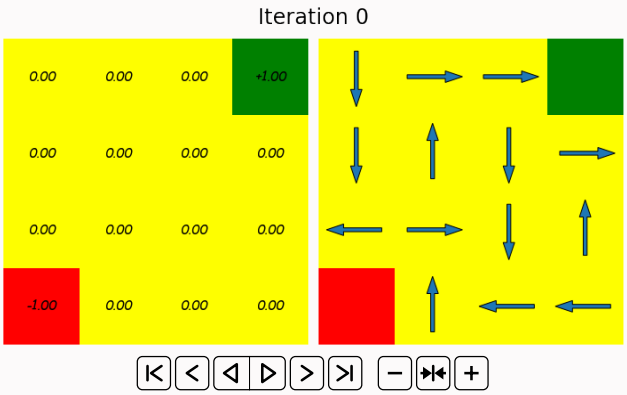
Алгоритм 8: Policy Iteration

Гиперпараметры: ε — критерий останова для процедуры **PolicyEvaluation**

Инициализируем $\pi_0(s)$ произвольно для всех $s \in \mathcal{S}$
На k-ом шаге:

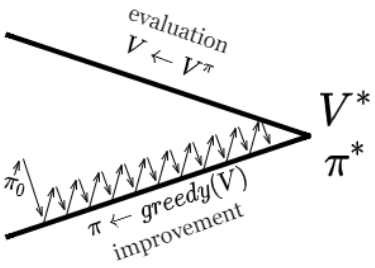
1. $V^{\pi_k} \coloneqq \textbf{PolicyEvaluation}(\pi_k, \varepsilon)$
2. $Q^{\pi_k}(s,a) \coloneqq r(s,a) + \gamma \mathbb{E}_{s'} V^{\pi_k}(s')$
3. $\pi_{k+1}(s) \coloneqq \operatorname{argmax}_a Q^{\pi_k}(s,a)$
4. **критерий останова:** $\pi_k \equiv \pi_{k+1}$

Пример 56: Решим задачу из примера 54, $\gamma = 0.9$; на каждой итерации слева отображается $V^{\pi_k}(s)$; справа улучшенная π_{k+1}. За 4 шага алгоритм сходится к оптимальной стратегии.



3.3.5. Generalized Policy Iteration

Policy Iteration — идеализированный алгоритм: на этапе оценивания в табличном сеттинге можно попробовать решить систему уравнений Беллмана V^π с достаточно высокой точностью за счёт линейности этой системы уравнений, или можно считать, что проводится достаточно большое количество итераций метода простой итерации. Тогда, вообще говоря, процедура предполагает бесконечное число шагов, и на практике нам нужно когда-то остановиться; теоретически мы считаем, что доводим вычисления до некоторого критерия останова, когда значения вектора не меняются более чем на некоторую погрешность $\varepsilon > 0$.



Но рассмотрим такую, пока что, эвристику: давайте останавливать Policy Evaluation после ровно N шагов, а после обновления стратегии не начинать оценивать π_{k+1} с нуля, а использовать последнее $V(s) \approx V^{\pi_k}$ в качестве инициализации. Тогда наш алгоритм примет следующий вид:

[4] считаем, что аргмакс берётся однозначно для любой Q-функции: в случае, если в **Argmax** содержится более одного элемента, множество действий как-то фиксированно упорядочено, и берётся действие с наибольшим приоритетом.

Алгоритм 9: Generalized Policy Iteration

Гиперпараметры: N — количество шагов

Инициализируем $\pi(s)$ произвольно для всех $s \in \mathcal{S}$
Инициализируем $V(s)$ произвольно для всех $s \in \mathcal{S}$
На k-ом шаге:

1. **Повторить N раз:**
   - $\forall s$: $V(s) \leftarrow \mathbb{E}_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V(s') \right]$
2. $Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s'} V(s')$
3. $\pi(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q(s,a)$

Мы формально теряем гарантии улучшения стратегии на этапе Policy Improvement, поэтому останавливать алгоритм после того, как стратегия не изменилась, уже нельзя: возможно, после следующих N шагов обновления оценочной функции, аргмакс поменяется, и стратегия всё-таки сменится. Но такая схема в некотором смысле является наиболее общей, и вот почему:

Утверждение 28: Generalized Policy Iteration (алг. 9) совпадает с Value Iteration (алг. 7) при $N = 1$ и с Policy Iteration (алг. 8) при $N = \infty$.

Доказательство. Второе очевидно; увидим первое. При $N = 1$ наше обновление V-функции имеет следующий вид:

$$V(s) \leftarrow \mathbb{E}_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V(s') \right]$$

Вспомним, по какому распределению берётся мат. ожидание $\mathbb{E}_a$: по π, которая имеет вид

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a) = \underset{a}{\operatorname{argmax}} \left[r(s,a) + \gamma \mathbb{E}_{s'} V(s') \right]$$

Внутри аргмакса как раз стоит содержимое нашего мат.ожидания в обновлении V, поэтому это обновление выродится в

$$V(s) \leftarrow \max_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V(s') \right]$$

Это в точности обновление из алгоритма Value Iteration. ■

Итак, Generalized Policy Iteration при $N = 1$ и при $N = \infty$ — это ранее разобранные алгоритмы, физический смысл которых нам ясен. В частности, теперь понятно, что в Value Iteration очередное приближение $Q_k(s,a) \approx Q^*(s,a)$ можно также рассматривать как приближение $Q_k(s,a) \approx Q^{\pi}(s,a)$ для $\pi(s) \coloneqq \underset{a}{\operatorname{argmax}}\, Q_k(s,a)$; то есть в алгоритме хоть и не потребовалось в явном виде хранить «текущую» стратегию, она всё равно неявно в нём присутствует.

Давайте попробуем понять, что происходит в Generalized Policy Iteration при промежуточных N. Заметим, что повторение N раз шага метода простой итерации для решения уравнения $\mathfrak{B} V^{\pi} = V^{\pi}$ эквивалентно одной итерации метода простой итерации для решения уравнения $\mathfrak{B}^N V^{\pi} = V^{\pi}$ (где запись $\mathfrak{B}^N$ означает повторное применение оператора $\mathfrak{B}$ N раз), для которого, очевидно, искомая V^{π} также будет неподвижной точкой. Что это за оператор $\mathfrak{B}^N$?

В уравнениях Беллмана мы «раскручивали» наше будущее на один шаг вперёд и дальше заменяли оставшийся «хвост» на определение V-функции. Понятно, что мы могли бы раскрутить не на один шаг, а на N шагов вперёд.

Теорема 25 — N-шаговое уравнение Беллмана: Для любого состояния s_0:

$$V^{\pi}(s_0) = \mathbb{E}_{\tau_{:N} \sim \pi | s_0} \left[\sum_{t=0}^{N-1} \gamma^t r_t + \gamma^N \mathbb{E}_{s_N} V^{\pi}(s_N) \right] \tag{3.25}$$

Доказательство по индукции. Для получения уравнения на N шагов берём $N-1$-шаговое и подставляем в правую часть раскрутку на один шаг из уравнения (3.3). Это в точности соответствует применению оператора Беллмана N раз. ■

Доказательство без индукции. Для любых траекторий $\boldsymbol{\mathcal{T}}$ верно, что

$$\boldsymbol{R(\mathcal{T}) = \sum_{t=0}^{N-1} \gamma^t r_t + \gamma^N R_N}$$

Возьмём мат.ожидание $\mathbb{E}_{\boldsymbol{\mathcal{T}\sim\pi|s_0}}$ слева и справа:

$$\boldsymbol{\mathbb{E}_{\mathcal{T}\sim\pi|s_0} R(\mathcal{T}) = \mathbb{E}_{\mathcal{T}\sim\pi|s_0}\left[\sum_{t=0}^{N-1} \gamma^t r_t + \gamma^N R_N\right]}$$

Слева видно определение V-функции. Справа достаточно разделить мат.ожидание на мат.ожидание по первым $\boldsymbol{N}$ шагам и хвост:

$$\boldsymbol{V^\pi(s_0) = \mathbb{E}_{\mathcal{T}_{:N}\sim\pi|s_0}\left[\sum_{t=0}^{N-1} \gamma^t r_t + \gamma^N \mathbb{E}_{s_N}\mathbb{E}_{\mathcal{T}_{N:}\sim\pi|s_N} R_N\right]}$$

Осталось выделить справа во втором слагаемом определение V-функции. ■

Утверждение 29: $\boldsymbol{\mathfrak{B}^N}$ — оператор с коэффициентом сжатия $\boldsymbol{\gamma^N}$.

Доказательство.

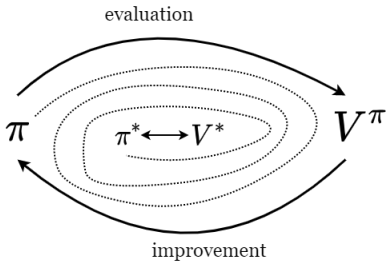
$$\boldsymbol{\rho(\mathfrak{B}^N V_1, \mathfrak{B}^N V_2) \le \gamma\rho(\mathfrak{B}^{N-1} V_1, \mathfrak{B}^{N-1} V_2) \le \dots \le \gamma^N \rho(V_1, V_2)}$$ ■

Означает ли это, что метод простой итерации решения $\boldsymbol{N}$-шаговых уравнений сойдётся быстрее? Мы по сути просто «за один шаг» делаем $\boldsymbol{N}$ итераций метода простой итерации для решения обычного одношагового уравнения; в этом смысле, мы ничего не выигрываем. В частности, если мы устремим $\boldsymbol{N}$ к бесконечности, то мы получим просто определение V-функции; формально, в правой части будет стоять выражение, вообще не зависящее от поданной на вход оператору $\boldsymbol{V(s)}$, коэффициент сжатия будет ноль, и метод простой итерации как бы сходится тут же за один шаг. Но для проведения этого шага нужно выинтегрировать все траектории — «раскрыть дерево полностью».

Но теперь у нас есть другой взгляд на Generalized Policy Iteration: мы чередуем одну итерацию решения $\boldsymbol{N}$-шагового уравнения Беллмана с Policy Improvement-ом.

Теорема 26: Алгоритм Generalized Policy Iteration 9 при любом $\boldsymbol{N}$ сходится к оптимальной стратегии и оптимальной оценочной функции.

Без доказательства. ■



Интуитивно, такой алгоритм «стабилизируется», если оценочная функция будет удовлетворять уравнению Беллмана для текущей $\boldsymbol{\pi}$ (иначе оператор $\boldsymbol{\mathfrak{B}^N}$ изменит значение функции), и если $\boldsymbol{\pi}$ выбирает $\boldsymbol{\operatorname{argmax}_a Q(s,a)}$ из неё; а если аппроксимация V-функции удовлетворяет уравнению Беллмана, то она совпадает с $\boldsymbol{V^\pi}$, и значит $\boldsymbol{Q(s,a) = Q^\pi(s,a)}$. То есть, при сходимости стратегия $\boldsymbol{\pi}$ будет выбирать действие жадно по отношению к своей же $\boldsymbol{Q^\pi}$, а мы помним, что это в точности критерий оптимальности.

Все алгоритмы, которые мы будем обсуждать далее, так или иначе подпадают под обобщённую парадигму «оценивание-улучшение». У нас будет два процесса оптимизации: обучение ***актёра*** (actor), политики $\boldsymbol{\pi}$, и ***критика*** (critic), оценочной функции (Q или V). Критик обучается оценивать текущую стратегию, текущего актёра: сдвигаться в сторону решения какого-нибудь уравнения, для которого единственной неподвижной точкой является $\boldsymbol{V^\pi}$ или $\boldsymbol{Q^\pi}$. Актёр же будет учиться при помощи policy improvement-а: вовсе не обязательно делать это жадно, возможно учиться выбирать те действия, где оценка критика «побольше», оптимизируя в каких-то состояниях (в каких - пока открытый вопрос) функционал (3.21):

$$\boldsymbol{\mathbb{E}_{\pi(a|s)} Q(s,a) \to \max_\pi}$$

Причём, возможно, в этом функционале нам не понадобится аппроксимация (модель) Q-функции в явном виде, и тогда мы можем обойтись лишь какими-то оценками $\boldsymbol{Q^\pi}$; в таких ситуациях нам достаточно будет на этапе оценивания политики обучать лишь модель $\boldsymbol{V^\pi}$ для текущей стратегии. А, например, в эволюционных методах мы обошлись вообще без обучения критика именно потому, что смогли обойтись лишь Монте-Карло оценками будущих наград. Этот самый простой способ решать задачу RL — погенерировать несколько случайных стратегий и выбрать среди них лучшую — тоже условно подпадает под эту парадигму: мы считаем Монте-Карло оценки значения $\boldsymbol{J(\pi)}$ для нескольких разных стратегий (evaluation) и выбираем наилучшую

стратегию (improvement). Поэтому Policy Improvement, как мы увидим, тоже может выступать в разных формах: например, возможно, как в Value Iteration, у нас будет приближение Q-функции, и мы будем просто всегда полагать, что policy improvement проводится жадно, и текущей стратегией неявно будет $\operatorname{argmax}_a Q(s,a)$.

Но главное, что эти два процесса, оценивание политики (обучение критика) и улучшение (обучение актёра) можно будет проводить стохастической оптимизацией. Достаточно, чтобы лишь *в среднем* модель оценочной функции сдвигалась в сторону V^π или Q^π, а актёр лишь *в среднем* двигался в сторону $\operatorname{argmax}_a Q(s,a)$. И такой «рецепт» алгоритма всегда будет работать: пока оба этих процесса проводятся корректно, итоговый алгоритм запустится на практике. Это в целом фундаментальная идея всего RL. В зависимости от выбора того, как конкретно проводить эти процессы, получатся разные по свойствам алгоритмы, и, в частности, отдельно интересными будут алгоритмы, «схлопывающие» схему Generalized Policy Iteration в её предельную форму, в Value Iteration.

Мы далее начнём строить model-free алгоритмы, взяв наши алгоритмы планирования — Policy Iteration и Value Iteration, — и попробовав превратить их в табличные алгоритмы решения задачи.

§3.4. Табличные алгоритмы

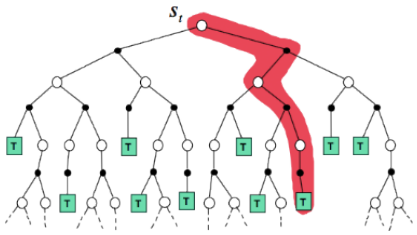
3.4.1. Монте-Карло алгоритм

Value iteration и Policy iteration имели два ограничения: 1) необходимо уметь хранить табличку размером $|\mathcal{S}|$ в памяти и перебирать все состояния и действия за разумное время 2) должна быть известна динамика среды $p(s' \mid s,a)$. Первое полечим нейронками, а сейчас будем лечить второе: в сложных средах проблема даже не столько в том, чтобы приблизить динамику среды, а в том, что интегралы $\mathbb{E}_{s'\sim p(s'|s,a)}$ мы не возьмём в силу огромного числа состояний и сможем только оценивать по Монте-Карло. Итак, мы хотим придумать табличный model-free RL-алгоритм: мы можем отправить в среду пособирать траектории какую-то стратегию, и дальше должны проводить итерации алгоритма, используя лишь эти сэмплы траекторий. Иначе говоря, для данного s мы можем выбрать a и получить ровно один сэмпл из очередного $p(s' \mid s,a)$, причём на следующем шаге нам придётся проводить сбор сэмплов именно из s'. Как в таких условиях «решать» уравнения Беллмана — неясно.

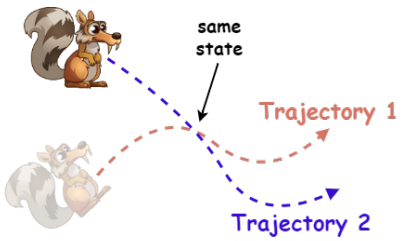
Рассмотрим самый простой способ превратить Policy Iteration в model-free метод. Давайте очередную стратегию π_k отправим в среду, сыграем несколько эпизодов, и будем оценивать Q^{π_k} по Монте-Карло:

$$Q^{\pi_k}(s,a) \approx \frac{1}{N}\sum_{i=0}^{N} R(\tau_i), \quad \tau_i \sim \pi_k \mid s_0 = s, a_0 = a$$

Теперь, доиграв эпизод до конца, мы для каждой встретившейся пары s,a в полученной траектории можем посчитать reward-to-go и использовать этот сэмпл для обновления нашей аппроксимации Q-функции — проведения ***Монте-Карло бэкапа*** (MC-backup). Такое обновление полностью противоположно по свойствам бэкапу динамического программирования: это «бэкап ширины один» бесконечной длины — мы использовали лишь один сэмпл будущего и при этом заглянули в него на бесконечное число шагов вперёд.



Формально, из-за петлей сэмплы являются скоррелированными. Если мы крутимся в петле, а потом в какой-то момент эпизода вышли и получили $+1$, то сэмплы будут выглядеть примерно так: $\gamma^5, \gamma^4, \gamma^3 \dots$. Причина в том, что мы взяли по несколько сэмплов для одной и той же Монте-Карло оценки (для одной и той же пары s,a) из одного и того же эпизода («***every-visit***»); для теоретической корректности следует гарантировать независимость сэмплов, например, взяв из каждого эпизода для каждой пары s,a сэмпл только для первого посещения («***first-visit***»).



Монте-Карло алгоритм, на первый взгляд, плох примерно всем. Нужно доигрывать игры до конца, то есть алгоритм неприменим в неэпизодичных средах; Монте-Карло оценки также обладают огромной дисперсией, поскольку мы заменили на сэмплы все мат.ожидания, стоящие в мат.ожидании по траекториям; наконец, мы практически перестали использовать структуру задачи. Если мы получили сэмпл $+100$ в качестве очередного сэмпла для $Q^\pi(s,a)$, то мы «забыли», какую часть из этой $+100$ мы получили сразу же после действия a, а какая была получена в далёком будущем — не использовали разложение награды за эпизод в сумму наград за шаг. Также мы посеяли «информацию о соединениях состояниях»: пусть у нас было две траектории (см. рисунок), имевших пересечение в общем состоянии. Тогда для начал этих траекторий мы всё равно считаем, что собрали лишь один сэмпл reward-to-go, хотя в силу марковости у нас есть намного больше информации.

Ещё одна проблема алгоритма: если для некоторых $s, a\colon \pi(a \mid s) = 0$, то мы ничего не узнали об $Q^\pi(s, a)$. А, как было видно в алгоритмах динамического программирования, мы существенно опираемся в том числе и на значения Q-функции для тех действий, которые π никогда не выбирает; только за счёт этого мы умеем проводить policy improvement для детерминированных стратегий.

А ещё в таком Монте-Карло алгоритме встаёт вопрос: когда заканчивать оценивание Q-функции и делать шаг Policy Improvement-а? Точное значение Q^{π_k} за конечное время мы Монте-Карло оценкой всё равно не получим, и в какой-то момент improvement проводить придётся, с потерей теоретических гарантий. Возникает вопрос: насколько разумно в таких условиях после очередного обновления стратегии начинать расчёт оценочной функции $Q^{\pi_{k+1}}$ «с нуля»? Может, имеет смысл проинициализировать Q-функцию для новой стратегии π_{k+1} как-то при помощи текущего приближения $Q(s, a) \approx Q^{\pi_k}(s, a)$? Да, хоть оно и считалось для «предыдущей» стратегии и формально содержит сэмплы не из того распределения, но всё-таки этих сэмплов там было аккумулировано много, да и стратегия потенциально поменялась не сильно; и всяко лучше какой-нибудь нулевой инициализации. Возникает желание усреднять сэмплы с приоритетом более свежих, приходящих из «правильной» стратегии; а «неправильные» сэмплы, из старой стратегии, всё-таки использовать, но с каким-то маленьким весом. Всё это хочется делать как-то онлайн, не храня всю историю Монте-Карло оценок.

3.4.2. Экспоненциальное сглаживание

Рассмотрим такую задачу. Нам приходят сэмплы $x_1, x_2 \dots x_n \sim p(x)$. Хотим по ходу получения сэмплов оценивать мат.ожидание случайной величины x. Давайте хранить Монте-Карло оценку, усредняя все имеющиеся сэмплы; для этого достаточно пользоваться следующим рекурсивным соотношением:

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

Обозначим за $\alpha_k := \frac{1}{k}$. Тогда формулу можно переписать так:

$$m_k := (1 - \alpha_k) m_{k-1} + \alpha_k x_k$$

Определение 49: ***Экспоненциальным сглаживанием*** (exponential smoothing) для последовательности $x_1, x_2, x_3 \dots$ будем называть следующую оценку:

$$m_k := (1 - \alpha_k) m_{k-1} + \alpha_k x_k,$$

где m_0 — некоторое начальное приближение, последовательность $\alpha_k \in [0, 1]$ — гиперпараметр, называемый ***learning rate***.

Можно ли оценивать мат.ожидание как-то по-другому? В принципе, любая выпуклая комбинация имеющихся сэмплов будет несмещённой оценкой. В частности, если $\alpha_k > \frac{1}{k}$, то мы «выдаём» более свежим сэмплам больший вес. Зададимся таким техническим вопросом: при каких других последовательностях α_k формула позволит оценивать среднее?

Определение 50: Будем говорить, что learning rate $\alpha_k \in [0, 1]$ удовлетворяет ***условиям Роббинса-Монро*** (Robbins-Monro conditions), если:

$$\sum_{k\geq 0}^{\infty} \alpha_k = +\infty, \quad \sum_{k\geq 0}^{\infty} \alpha_k^2 < +\infty \tag{3.26}$$

Теорема 27: Пусть $x_1, x_2 \dots$ — независимые случайные величины, $\mathbb{E}x_k = m, \mathbb{D}x_k \leq C < +\infty$, где C — некоторая конечная константа. Пусть m_0 — произвольно, а последовательность чисел $\alpha_k \in [0, 1]$ удовлетворяет условиям Роббинса-Монро (3.26). Тогда экспоненциальное сглаживание

$$m_k := (1 - \alpha_k) m_{k-1} + \alpha_k x_k \tag{3.27}$$

сходится к m с вероятностью 1.

Доказательство. Без ограничения общности будем доказывать утверждение для $m = 0$, поскольку для сведения к этому случаю достаточно вычесть m из правой и левой части (3.27) и перейти к обозначениям $\hat{m_k} := m_k - m, \hat{x_k} := x_k - m$.

Итак, пусть $\mathbb{E}x_k = 0$. Будем доказывать, что $v_k := \mathbb{E}m_k^2 \xrightarrow{k\to\infty} 0$. Для начала возведём обе стороны уравнения (3.27) в квадрат:

$$m_k^2 = (1 - \alpha_k)^2 m_{k-1}^2 + \alpha_k^2 x_k^2 + 2\alpha_k(1 - \alpha_k) x_k m_{k-1}$$

Возьмём справа и слева мат.ожидание:

$$\mathbb{E}m_k^2 = (1 - \alpha_k)^2 \mathbb{E}m_{k-1}^2 + \alpha_k^2 \mathbb{E}x_k^2 + 2\alpha_k(1 - \alpha_k)\mathbb{E}(x_k m_{k-1})$$

Последнее слагаемое зануляется, поскольку в силу независимости $\mathbb{E}(x_k m_{k-1}) = \mathbb{E}x_k \mathbb{E}m_{k-1}$, а $\mathbb{E}x_k$ равно нулю по условию. Используя введённое обозначение, получаем такой результат:

$$v_k = (1-\alpha_k)^2 v_{k-1} + \alpha_k^2 \mathbb{E}x_k^2 \tag{3.28}$$

Сейчас мы уже можем доказать, что $v_k \le C$. Действительно, сделаем это по индукции. База: $v_0 = 0 \le C$ по определению. Шаг: пусть $v_{k-1} \le C$, тогда

$$v_k = (1-\alpha_k)^2 v_{k-1} + \alpha_k^2 \mathbb{E}x_k^2 \le (1-\alpha_k)^2 C + \alpha_k^2 C \le C$$

где последнее неравенство верно при любых $\alpha_k \in [0, 1]$.

Особенность дальнейшего доказательства в том, что последовательность v_k вовсе не обязана быть монотонной. Поэтому применим пару фокусов в стиле матана. Сначала раскроем скобки в рекурсивном выражении (3.28):

$$v_k - v_{k-1} = -2\alpha_k v_{k-1} + \alpha_k^2 (v_{k-1} + \mathbb{E}x_k^2)$$

Мы получили счётное число равенств, проиндексированных k. Просуммируем первые n из них:

$$v_n - v_0 = -2 \sum_{k=0}^{n-1} \alpha_k v_k + \sum_{k=0}^{n-1} \alpha_k^2 (v_k + \mathbb{E}x_k^2) \tag{3.29}$$

Заметим, что $v_0 = 0$, а $v_n \ge 0$ по определению как дисперсия. Значит:

$$2 \sum_{k=0}^{n-1} \alpha_k v_k \le \sum_{k=0}^{n-1} \alpha_k^2 (v_k + \mathbb{E}x_k^2)$$

Применяем ограниченность v_k и $\mathbb{E}x_k^2$:

$$2 \sum_{k=0}^{n-1} \alpha_k v_k \le \sum_{k=0}^{n-1} \alpha_k^2 (C + C) = 2C \sum_{k=0}^{n-1} \alpha_k^2$$

Ряд справа сходится при $n \to +\infty$. Значит, сходится и ряд слева. Коли так, имеет предел правая часть (3.29). Значит, имеет предел и левая.

Было доказано, что последовательность v_k имеет предел. Понятно, что он неотрицателен. Допустим, он положителен и отделён от 0, равен некоторому $b > 0$. Возьмём какое-нибудь небольшое $\varepsilon > 0$, так что $b - \varepsilon > 0$. Тогда, начиная с некоторого номера i все элементы последовательности $v_k > b - \varepsilon$ при $k \ge i$. Получим:

$$\sum_{k=0}^{n-1} \alpha_k v_k \ge \sum_{k=i}^{n-1} \alpha_k v_k \ge (b - \varepsilon) \sum_{k=i}^{n-1} \alpha_k$$

Правая часть неравенства расходится, поскольку мы просили расходимость ряда из α_k; но ранее мы доказали сходимость левой части. Значит, предел равен нулю. ■

3.4.3. Стохастическая аппроксимация

Мы научились, можно считать, решать уравнения такого вида:

$$x = \mathbb{E}_\varepsilon f(\varepsilon),$$

где справа стоит мат.ожидание по неизвестному распределению $\varepsilon \sim p(\varepsilon)$ от какой-то функции f, которую для данного значения сэмпла ε мы умеем считать. Это просто стандартная задача оценки среднего, для которой мы даже доказали теоретические гарантии сходимости следующего итеративного алгоритма:

$$x_k := (1-\alpha_k) x_{k-1} + \alpha_k f(\varepsilon), \quad \varepsilon \sim p(\varepsilon)$$

Аналогично у нас есть итеративный алгоритм для решения систем нелинейных уравнений

$$x = f(x),$$

где справа стоит, если угодно, «хорошая» функция — сжатие. Формула обновления в методе простой итерации выглядела вот так:

$$x_k := f(x_{k-1})$$

Определение 51: ***Стохастическая аппроксимация*** (Stochastic approximation) — задача решения уравнения вида

$$\boldsymbol{x} = \mathbb{E}_{\boldsymbol{\varepsilon}\sim\boldsymbol{p}(\boldsymbol{\varepsilon})}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\varepsilon}), \tag{3.30}$$

где справа стоит мат.ожидание по неизвестному распределению $\boldsymbol{p}(\boldsymbol{\varepsilon})$, из которого доступны лишь сэмплы, от функции, которую при данном сэмпле $\boldsymbol{\varepsilon}$ и некотором значении неизвестной переменной $\boldsymbol{x}$ мы умеем считать.

Можно ли объединить идеи метода простой итерации и экспоненциального сглаживания («перевзвешанной» Монте-Карло оценки)? Давайте запустим аналогичный итеративный алгоритм: на $\boldsymbol{k}$-ой итерации подставим текущее приближение неизвестной переменной $\boldsymbol{x_k}$ в правую часть $\boldsymbol{f}(\boldsymbol{x_k},\boldsymbol{\varepsilon})$ для $\boldsymbol{\varepsilon}\sim\boldsymbol{p}(\boldsymbol{\varepsilon})$, но не будем «жёстко» заменять $\boldsymbol{x_{k+1}}$ на полученное значение, так как оно является лишь несмещённой оценкой правой части; вместо этого сгладим старое значение $\boldsymbol{x_k}$ и полученный новый «сэмпл»:

$$\boldsymbol{x_k} = (\boldsymbol{1}-\boldsymbol{\alpha_k})\boldsymbol{x_{k-1}} + \boldsymbol{\alpha_k}\boldsymbol{f}(\boldsymbol{x_{k-1}},\boldsymbol{\varepsilon}), \quad \boldsymbol{\varepsilon}\sim\boldsymbol{p}(\boldsymbol{\varepsilon}) \tag{3.31}$$

Есть хорошая надежда, что, если функция $\boldsymbol{f}$ «хорошая», распределение $\boldsymbol{p}(\boldsymbol{\varepsilon})$ не сильно страшное (например, имеет конечную дисперсию, как в теореме о сходимости экспоненциального сглаживания), а learning rate $\boldsymbol{\alpha_k}$ удовлетворяют условиям (3.26), то такая процедура будет в пределе сходиться.

Поймём, почему задача стохастической аппроксимации тесно связана со стохастической оптимизацией. Для этого перепишем формулу (3.31) в альтернативной очень интересной форме:

$$\boldsymbol{x_k} = \boldsymbol{x_{k-1}} + \boldsymbol{\alpha_k}\left(\boldsymbol{f}(\boldsymbol{x_{k-1}},\boldsymbol{\varepsilon}) - \boldsymbol{x_{k-1}}\right), \quad \boldsymbol{\varepsilon}\sim\boldsymbol{p}(\boldsymbol{\varepsilon}) \tag{3.32}$$

Да это же формула ***стохастического градиентного спуска*** (stochastic gradient descent, SGD)! Действительно, в нём мы, оптимизируя некоторую функцию $\boldsymbol{f}(\boldsymbol{x})$, прибавляем к очередному приближению $\boldsymbol{x_{k-1}}$ с некоторым learning rate $\boldsymbol{\alpha_k}$ несмещённую оценку градиента $\boldsymbol{\nabla f}(\boldsymbol{x_{k-1}})$, которую можно обозначить как $\boldsymbol{\nabla f}(\boldsymbol{x_{k-1}},\boldsymbol{\varepsilon})$:

$$\boldsymbol{x_k} = \boldsymbol{x_{k-1}} + \boldsymbol{\alpha_k}\boldsymbol{\nabla f}(\boldsymbol{x_{k-1}},\boldsymbol{\varepsilon}), \quad \boldsymbol{\varepsilon}\sim\boldsymbol{p}(\boldsymbol{\varepsilon})$$

О стохастической оптимизации можно думать не как об оптимизации, а как о поиске решения уравнения $\boldsymbol{\nabla f}(\boldsymbol{x}) = \boldsymbol{0}$. Действительно: SGD, как и любая локальная оптимизация, может идти как к локальному оптимуму, так и к седловым точкам, но в них $\mathbb{E}_{\boldsymbol{\varepsilon}}\boldsymbol{\nabla f}(\boldsymbol{x},\boldsymbol{\varepsilon}) = \boldsymbol{0}$.

И, глядя на формулу (3.32), мы понимаем, что, видимо, выражение $\boldsymbol{f}(\boldsymbol{x_{k-1}},\boldsymbol{\varepsilon}) - \boldsymbol{x_{k-1}}$ есть какой-то «стохастический градиент». Стохастический он потому, что это выражение случайно: мы сэмплируем $\boldsymbol{\varepsilon}\sim\boldsymbol{p}(\boldsymbol{\varepsilon})$; при этом это несмещённая оценка «градиента», поскольку она обладает следующим свойством: в точке решения, то есть в точке $\boldsymbol{x^*}$, являющейся решением уравнения (3.30), в среднем его значение равно нулю:

$$\mathbb{E}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{f}(\boldsymbol{x^*},\boldsymbol{\varepsilon}) - \boldsymbol{x^*}\right) = \boldsymbol{0}$$

И по аналогии можно предположить, что если стохастический градиентный спуск ищет решение $\mathbb{E}_{\boldsymbol{\varepsilon}}\boldsymbol{\nabla f}(\boldsymbol{x},\boldsymbol{\varepsilon}) = \boldsymbol{0}$, то процесс (3.32) ищет решение уравнения $\mathbb{E}_{\boldsymbol{\varepsilon}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\varepsilon}) - \boldsymbol{x} = \boldsymbol{0}$.

Это наблюдение будет иметь для нас ключевое значение. В model-free режиме как при оценивании стратегии, когда мы решаем уравнение Беллмана

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

так и когда мы пытаемся реализовать Value Iteration и напрямую решать уравнения оптимальности Беллмана

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

мы сталкиваемся в точности с задачей стохастической аппроксимации (3.30)! Справа стоит мат.ожидание по недоступному нам распределению, содержимое которого мы, тем не менее, при данном сэмпле $\boldsymbol{s'}\sim\boldsymbol{p}(\boldsymbol{s'}\mid\boldsymbol{s},\boldsymbol{a})$ и текущем приближении Q-функции, умеем считать. Возникает идея проводить стохастическую аппроксимацию: заменять правые части решаемых уравнений на несмещённые оценки и сглаживать текущее приближение с получаемыми сэмплами.

Отметим интересный факт: уравнение V*V* (3.18), имеющее вид «максимум от мат.ожидания по неизвестному распределению»

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

под данную форму не попадает. И с такими уравнениями мы ничего сделать не сможем. К счастью, нам и не понадобится.

3.4.4. Temporal Difference

Итак, попробуем применить идею стохастической аппроксимации для решения уравнений Беллмана. На очередном шаге после совершения действия a из состояния s мы получаем значение награды $r := r(s,a)$, сэмпл следующего состояния s' и генерируем $a' \sim \pi$, после чего сдвигаем с некоторым learning rate наше текущее приближение $Q(s,a)$ в сторону сэмпла

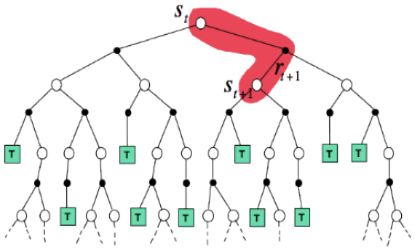
$$y := r + \gamma Q(s', a'),$$

который также будем называть ***таргетом***[5] (Bellman target). Получаем следующую формулу обновления:

$$Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \alpha_k \underbrace{\left(\overbrace{r + \gamma Q_k(s',a')}^{\text{таргет}} - Q_k(s,a) \right)}_{\text{временная разность}} \tag{3.33}$$

Выражение $r(s,a) + \gamma Q_k(s',a') - Q_k(s,a)$ называется ***временной разностью*** (temporal difference): это отличие сэмпла, который нам пришёл, от текущей оценки среднего.

Мы таким образом придумали ***TD-backup***: обновление, имеющее как ширину, так и длину один. Мы рассматриваем лишь одну версию будущего (один сэмпл) и заглядываем на один шаг вперёд, приближая всё дальнейшее будущее своей собственной текущей аппроксимацией. Этот ход позволяет нам учиться, не доигрывая эпизоды до конца: мы можем обновить одну ячейку нашей Q-функции сразу же после одного шага в среде, после сбора одного перехода (s, a, r, s', a').



Формула (3.33) отличается от Монте-Карло обновлений Q-функции лишь способом построения таргета: в Монте-Карло мы бы взяли в качестве y reward-to-go, когда здесь же используем одношаговое приближение. Поэтому смотреть на эту формулу также можно через интуицию ***бутстрапирования*** (bootstrapping)[6]. Мы хотим получить сэмплы для оценки нашей текущей стратегии π, поэтому делаем один шаг в среде и приближаем всю оставшуюся награду нашей текущей аппроксимацией среднего. Такой «псевдосэмпл» уже не будет являться корректным сэмплом для $Q^\pi(s,a)$, но в некотором смысле является «более хорошим», чем то, что у нас есть сейчас, за счёт раскрытия дерева на один шаг и получения информации об r, s'. Такое движение нашей аппроксимации в сторону чего-то хоть чуть-чуть более хорошего и позволяет нам чему-то учиться.

Пример 57: Вы сидите в кафе (s) и хотите вернуться домой. Вы прикидываете, что в среднем этой займёт $-Q(s,a) = 30$ минут. Вы тратите одну минуту ($-r$) на выход из кафе и обнаруживаете пробку (s'). За счёт этой новой информации вы можете дать более точную оценку времени до возвращения до дома: $-Q(s',a') = 40$ минут. Как можно в будущем откорректировать наши прогнозы? Можно засечь время, сколько в итоге заняла поездка — доиграть эпизод до конца и посчитать Монте-Карло оценку — а можно уже сделать вывод о том, что случилась некоторая «временная разность», ошибка за один шаг, равная $41 - 30 = 11$ минут. Заменять исходное приближение расстояния от кафе до дома $-Q(s,a)$ на 41 минуту, конечно же, слишком грубо: но временная разность говорит, что 30 минут было заниженной оценкой и её надо чуть-чуть увеличить.

Обсудим следующий важный технический момент. Какие есть ограничения на переходы (s, a, r, s', a'), которые мы можем использовать для обновлений по формуле (3.33)? Пока мы говорили, что мы хотим заниматься оцениванием стратегии π, и поэтому предлагается, видимо, ею и взаимодействовать со средой, собирая переходы. С точки же зрения стохастической аппроксимации, для корректности обновлений достаточно выполнения всего лишь двух требований:

- $s' \sim p(s' \mid s, a)$; если s' приходит из какой-то другой функции переходов, то мы учим Q-функцию для какого-то другого MDP.
- $a' \sim \pi(a' \mid s')$; если a' приходит из какой-то другой стратегии, то мы учим Q-функцию для вот этой другой стратегии.

Оба этих утверждения вытекают из того, что обновление (3.33) неявно ищет решение уравнения

$$Q(s,a) = \mathbb{E}_{s'} \mathbb{E}_{a'} y$$

[5] в англоязычной литературе встречается слово guess — «догадка»; этот таргет имеет смысл наших собственных предположений о том, какое будущее нас ждёт.

[6] бутстрэп — «ремешки на ботинках», происходит от выражения «потянуть самого себя за ремешки на ботинках и так перелезть через ограду». Русскоязычный аналог — «тащить самого себя за волосы из болота». Наши таргеты построены на принципе такого бутстрапирования, поскольку мы начинаем «из воздуха» делать себе сэмплы из уже имеющихся сэмплов.

как схема стохастической аппроксимации. Поэтому мы будем уделять много внимания тому, из правильных ли распределений приходят данные, которые мы «скармливаем» этой формуле обновления. Но и с другой стороны: схема не требует, например, чтобы после ячейки $Q(s, a)$ мы обязательно на следующем шаге обновили ячейку $Q(s', a')$, ровно как и не говорит ничего о требованиях на распределение, из которого приходят пары s, a. Как мы увидим позже, это наблюдение означает, что нам вовсе не обязательно обучаться с онлайн-опыта.

3.4.5. Q-learning

Поскольку довести $Q(s, a)$ до точного значения $Q^\pi(s, a)$ с гарантиями мы всё равно не сможем, однажды в алгоритме нам всё равно придётся сделать policy improvement. Что, если мы будем обновлять нашу стратегию $\pi_k(s) := \operatorname{argmax}_a Q_k(s, a)$ после каждого шага в среде и каждого обновления Q-функции? Наше приближение Policy Iteration схемы, аналогично ситуации в динамическом программировании, превратится в приближение Value Iteration схемы:

$$\begin{aligned} y &= r + \gamma Q_k(s', a') = \\ &= r + \gamma Q_k(s', \pi_k(s')) = \\ &= r + \gamma Q_k(s', \operatorname*{argmax}_{a'} Q_k(s', a')) = \\ &= r + \gamma \max_{a'} Q_k(s', a') \end{aligned}$$

Мы получили в точности таргет для решения методом стохастической аппроксимации уравнения оптимальности Беллмана; поэтому для такого случая будем обозначать нашу Q-функцию как аппроксимацию Q^*, и тогда наше обновление принимает следующий вид: для перехода s, a, r, s' обновляется только одна ячейка нашего приближения Q-функции:

$$Q_{k+1}(s, a) := Q_k(s, a) + \alpha_k \left(r + \gamma \max_{a'} Q_k(s', a') - Q_k(s, a) \right) \tag{3.34}$$

Теорема 28 — Сходимость Q-learning: Пусть пространства состояний и действий конечны, $Q_0(s, a)$ — начальное приближение, на k-ой итерации $Q_k(s, a)$ для всех пар s, a строится по правилу

$$Q_{k+1}(s, a) := Q_k(s, a) + \alpha_k(s, a) \left(r(s, a) + \gamma \max_{a'} Q_k(s'_k(s, a), a') - Q_k(s, a) \right)$$

где $s'_k(s, a) \sim p(s' \mid s, a)$, а $\alpha_k(s, a) \in [0, 1]$ — случайные величины, с вероятностью один удовлетворяющие для каждой пары s, a условиям Роббинса-Монро:

$$\sum_{k \geq 0}^{\infty} \alpha_k(s, a) = +\infty \qquad \sum_{k \geq 0}^{\infty} \alpha_k(s, a)^2 < +\infty \tag{3.35}$$

Тогда Q_k сходится к Q^* с вероятностью один.

Доказательство вынесено в приложение А.3. ■

В частности, если агент некоторым образом взаимодействует со средой и на k-ом шаге обновляет своё текущее приближение $Q \approx Q^*$ только для одной пары s, a, то можно считать, что $\alpha_k(s, a) \neq 0$ только для неё. При этом ограничения (3.35) всё ещё могут оказаться выполненными, поэтому эта интересующая нас ситуация есть просто частный случай сформулированной теоремы.

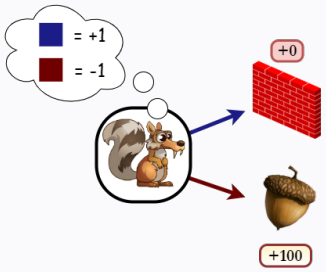
Заметим, что для выполнения условий сходимости необходимо для каждой пары s, a проводить бесконечное число обновлений $Q(s, a)$: в противном случае, в ряду $\sum_{k \geq 1} \alpha_k(s, a)$ будет конечное число ненулевых членов, и мы не попадём под теорему. Значит, наш сбор опыта должен удовлетворять условию ***infinite visitation*** — все пары s, a должны гарантированно встречаться бесконечно много раз. По сути теорема говорит, что это требование является достаточным условием на процесс порождения переходов s, a, r, s':

Утверждение 30: Пусть сбор опыта проводится так, что пары s, a встречаются бесконечное число раз. Пусть $n(s, a)$ — счётчик количества выполнений действия a в состоянии s во время взаимодействия агента со средой. Тогда можно положить $\alpha_k(s, a) := \frac{1}{n(s,a)}$, чтобы гарантировать сходимость алгоритма к Q^*.

3.4.6. Exploration-exploitation дилемма

Так какой же стратегией играть со средой, чтобы получать траектории? Коли мы учим Q^*, у нас на очередном шаге есть текущее приближение $\pi_k(s) \coloneqq \operatorname*{argmax}_a Q_k(s,a)$, и мы можем ***использовать*** (exploit) эти знания.

Теорема 29: Собирая траектории при помощи жадной стратегии, есть риск не сойтись к оптимальной.



Доказательство. Приведём простой пример. Есть два действия: получить приз (+100) и тупить в стену (+0), после выполнения любого игра заканчивается. Заинициализировали $Q_0(\text{приз}) = -1$, $Q_0(\text{стена}) = +1$. В первой игре, пользуясь текущей аппроксимацией $\pi_0(s) \coloneqq \operatorname*{argmax}_a Q_0(s,a)$, выбираем тупить в стену. Получая 0, сглаживаем 0 и текущее приближение +1, получая новое значение $Q_{k=1}(\text{стена}) \geq 0$. Оно превосходит $Q_{k=1}(\text{приз}) = -1$. Очевидно, и в дальнейших играх агент никогда не выберет приз, и оптимальная стратегия не будет найдена. ■

Действительно, детерминированные стратегии не позволяют получить свойство infinite visitation: многие пары s, a просто приниципиально не встречаются в порождаемых ими траекториях. В частности, из-за неё нельзя ограничиваться рассмотрением только класса детерминированных стратегий, хоть и была доказана теорема 21 о существовании в нём оптимальной стратегии: мы показали, что для сбора данных — взаимодействия со средой — детерминированные стратегии не подходят. Какие подходят?

Теорема 30: Любая стратегия, для которой $\forall s, a\colon \mu(a \mid s) > 0$, удовлетворяет условию infinite visitation, то есть с отличной от нуля вероятностью в траектории, порождённой π, встретится любая пара s, a.

Доказательство. Для любого s существует набор действий $a_0, a_1 \dots a_N$, которые позволяют с некоторой ненулевой вероятностью добраться до него из s_0: $p(s \mid s_0, a_0 \dots a_N) > 0$; если это не так, то ни одна стратегия никогда не попадёт в s, и мы без ограничения общности можем считать, что таких «параллельных вселенных» в нашей среде не существует. Значит, π с некоторой ненулевой вероятностью выберет эту цепочку действий $a_0 \dots a_N$, после чего с ненулевой вероятностью окажется в s и с ненулевой вероятностью выберет заданное a. ■

Если мы воспользуемся любой такой стохастической стратегией, мы попадём под действие теоремы о сходимости 28. Совершая случайные действия, мы рискуем творить ерунду, но занимаемся ***исследованием*** (exploration) — поиском новых возможностей в среде. Означает ли это, что нам достаточно взять полностью случайную стратегию, которая равновероятно выбирает из множества действий, отправить её в среду порождать переходики s, a, r, s', обучать на них Q-функцию по формуле (3.34), и на выходе в пределе мы получим Q^*? В пределе — да.

Пример 58: Представьте, что вы отправили случайную стратегию играть в Марио. Через экспоненциально много попыток она случайно пройдёт первый уровень и попадёт на второй; для состояний из второго уровня будет проведено первое обновление Q-функции. Через условно бесконечное число таких удач Q-функция для состояний из второго уровня действительно будет выучена...

Иначе говоря, исследование — корректный, но неэффективный способ генерации данных. Использование — куда более эффективный метод, позволяющий быстро добираться до «труднодоступных областей» в среде — такими областями обычно являются области с высокой наградой, иначе задача RL скорее всего не является особо интересной — и быстрее набирать сэмплы для обновлений пока ещё редко обновлённых ячеек $Q(s,a)$. Но это «некорректный» способ: детерминированная стратегия может застрять в локальном оптимуме и никогда не увидеть, что в каком-то месте другое действие даёт ещё большую награду.

Обсудим базовые варианты, как можно решать этот exploration-exploitation trade-off в контексте вычисления оптимальной Q-функции методом временных разностей. Нам нужно взять нашу стратегию использования $\pi(s) \coloneqq \operatorname*{argmax}_a Q(s,a)$ и что-нибудь с ней поделать так, чтобы она стала формально стохастичной.

Определение 52: ***ε-жадной*** (ε-greedy) называется стратегия

$$\mu(s) \coloneqq \begin{cases} a \sim \mathbf{Uniform}(\mathcal{A}) & \text{с вероятностью } \varepsilon; \\ \operatorname*{argmax}_a Q(s,a) & \text{иначе.} \end{cases} \tag{3.36}$$

Определение 53: ***Больцмановской*** с температурой τ называется стратегия

$$\mu(a \mid s) := \underset{a}{\mathbf{softmax}}\left(\frac{Q(s,a)}{\tau}\right) \tag{3.37}$$

Первый вариант никак не учитывает, насколько кажутся хорошими другие действия помимо жадного, и, если решает «исследовать», выбирает среди них случайно. Второй же вариант будет редко выбирать очень плохие действия (это может быть крайне полезным свойством), но чувствителен к масштабу приближения Q-функции, что обычно менее удобно и требует настройки температуры τ. Для Больцмановской стратегии мы увидим интересную интерпретацию в контексте обсуждения Maximum Entropy RL (раздел 6.2).

3.4.7. Реплей буфер

Итак, мы теперь можем собрать классический алгоритм табличного RL под названием Q-learning. Это метод временных разностей для вычисления оптимальной Q-функции с ε-жадной стратегией исследования.

Алгоритм 10: Q-learning

Гиперпараметры: α — параметр экспоненциального сглаживания, ε — параметр исследований

Инициализируем $Q(s,a)$ произвольно для всех $s \in \mathcal{S}, a \in \mathcal{A}$
Наблюдаем s_0
На k-ом шаге:

1. с вероятностью ε играем $a_k \sim \mathbf{Uniform}(\mathcal{A})$, иначе $a_k = \underset{a_k}{\mathbf{argmax}}\, Q(s_k, a_k)$
2. наблюдаем r_k, s_{k+1}
3. обновляем $Q(s_k,a_k) \leftarrow Q(s_k,a_k) + \alpha\left(r_k + \gamma \max\limits_{a_{k+1}} Q(s_{k+1},a_{k+1}) - Q(s_k,a_k)\right)$

Естественно, в эпизодичных средах нужно всегда следить за флагом **done** и учитывать, что для терминальных состояний оценочные функции равны нулю. Мнемонически можно запомнить, что этот флаг является частью дисконтирования: домножение на $\mathbf{1 - done}$ происходит всюду, где мы домножаем приближение будущей награды на γ.

Q-learning является типичным представителем off-policy алгоритмов: нам нигде не требовались сэмплы взаимодействия со средой конкретной стратегии. Наша стратегия сбора данных могла быть, вообще говоря, произвольной. Это крайне существенное свойство, потому что в таких алгоритмах возможно обучение с буфера: допустим, некоторый «эксперт» $\pi^{\mathbf{expert}}$ провёл много-много сессий взаимодействия со средой и собрал для нас кучу траекторий. Рассмотрим их как набор переходов. Тогда мы можем, вообще не взаимодействуя больше со средой, провести обучение Q^* с буфера: сэмплируем равномерно переход (s,a,r,s') и делаем обновление ячейки $Q(s,a)$ по формуле (3.34). Что мы тогда выучим?

Определение 54: Для данного буфера — набора переходов (s,a,r,s') — будем называть ***эмпирическим MDP*** (empirical MDP) MDP с тем же пространством состояний, действий и функций награды, где функция переходов задана следующим образом:

$$\hat{p}(s' \mid s,a) := \frac{N(s,a,s')}{N(s,a)},$$

где $N(s,a,s')$ — число троек s,a,s', входящих в буфер, $N(s,a)$ — число пар s,a, входящих в буфер.

Утверждение 31: При выполнении условий на learning rate, Q-learning, запущенный с фиксированного буфера, выучит Q^* для эмпирического MDP.

Доказательство. Именно из эмпирического распределения $\hat{p}(s' \mid s,a)$ приходит s' в формуле обновления (при равномерном сэмплировании переходов из буфера). Следовательно, для такого MDP мы и выучим оптимальную Q-функцию в силу теоремы 28. ■

Естественно, если буфер достаточно большой, то мы выучим очень близкую Q^* к настоящей. Ещё более интересно, что Q-learning как off-policy алгоритм может обучаться со своего собственного опыта — со своего же собственного буфера, составленного из порождённых очень разными стратегиями переходов.

Определение 55: ***Реплей буфер*** (replay buffer, experience replay) — это память со всеми собранными агентом переходами $(s, a, r, s', \text{done})$.

Если взаимодействие со средой продолжается, буфер расширяется, и распределение, из которого приходит s', становится всё больше похожим на настоящее $p(s' \mid s, a)$; на факт сходимости это не влияет.

Алгоритм 11: Q-learning with experience replay

Гиперпараметры: α — параметр экспоненциального сглаживания, ε — параметр исследований

Инициализируем $Q(s, a)$ произвольно для всех $s \in \mathcal{S}, a \in \mathcal{A}$
Наблюдаем s_0
На k-ом шаге:

1. с вероятностью ε играем $a_k \sim \text{Uniform}(\mathcal{A})$, иначе $a_k := \underset{a_k}{\text{argmax}}\, Q(s_k, a_k)$
2. наблюдаем r_k, s_{k+1}
3. кладём s_k, a_k, r_k, s_{k+1} в буфер
4. сэмплируем случайный переход s, a, r, s' из буфера
5. обновляем $Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max\limits_{a'} Q(s', a') - Q(s', a') \right)$

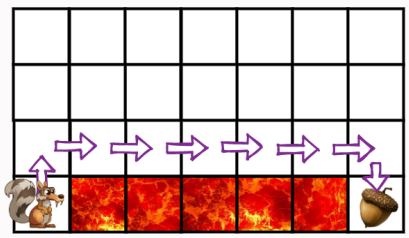
Реплей буфер — ключевое преимущество off-policy алгоритмов: мы сможем с каждого перехода потенциально обучаться бесконечное количество раз. Это развязывает нам руки в плане соотношения числа собираемых данных и количества итераций обновления нашей модели, поскольку здесь мы можем, в общем-то, сами решать, сколько переходов на один шаг обновления мы будем проводить. Проявляется это в том, что в Q-learning, как и в любом другом off-policy алгоритме, есть два независимых этапа: сбор данных (взаимодействие со средой при помощи ε-жадной стратегии и сохранение опыта в памяти — первые три шага), и непосредственно обучение (сэмплирование перехода из буфера и обновление ячейки — шаги 4-5). Можно проводить несколько шагов сбора данных на одно обновление, или наоборот: несколько обновлений на один шаг сбора данных, или даже проводить эти процессы независимо параллельно.

3.4.8. SARSA

Мы придумали off-policy алгоритм: мы умеем оценивать нашу текущую стратегию ($\underset{a}{\text{argmax}}\, Q(s, a)$, неявно сидящий внутри формулы обновления), используя сэмплы другой стратегии. Иными словами, у нас в алгоритме различаются понятия ***целевой политики*** (target policy) — стратегии, которую алгоритм выдаст по итогам обучения, она же оцениваемая политика, то есть та политика, для которой мы хотим посчитать оценочную функцию — и ***политики взаимодействия*** (behavior policy) — стратегии взаимодействия со средой, стратегии с подмешанным эксплорейшном. Это различие было для нас принципиально: оптимальны детерминированные стратегии, а взаимодействовать со средой мы готовы лишь стохастичными стратегиями. У этого «несовпадения» есть следующий эффект.

Пример 59 — Cliff World: Рассмотрим MDP с рисунка с детерминированной функцией переходов, действиями вверх-вниз-вправо-влево и $\gamma < 1$; за попадание в лаву начисляется огромный штраф, а эпизод прерывается. За попадание в целевое состояние агент получает $+1$, и эпизод также завершается; соответственно, задача агента — как можно быстрее добраться до цели, не угодив в лаву.

Q-learning, тем не менее, постепенно сойдётся к оптимальной стратегии: кратчайшим маршрутом агент может добраться до терминального состояния с положительной наградой. Однако даже после того, как оптимальная стратегия уже выучилась, Q-learning продолжает прыгать в лаву! Почему? Проходя прямо возле лавы, агент каждый шаг подбрасывает монетку и с вероятностью ε совершает случайное действие, которое при невезении может отправить его гореть! Если речь не идёт о симуляции, подобное поведение даже во время обучения может быть крайне нежелательно.



Возможны ситуации, когда небезопасное поведение во время обучения не является проблемой, но для, например, реальных роботов сбивать пешеходов из-за случайных действий — не самая лучшая идея. Что, если мы попробуем как-то «учесть» тот факт, что мы обязаны всегда заниматься исследованиями? То есть внутри

оценочной функции должно закладываться, что «оптимальное» поведение в будущем невозможно, а возможно только около-оптимальное поведение с подмешиванием исследования. Для примера будем рассматривать ε-жадную стратегию, пока что с константным ε.

Рассмотрим очень похожий на Q-learning алгоритм. Будем использовать пятёрки s, a, r, s', a' (hence the name) прямо из траекторий нашего взаимодействия со средой и апдейтить текущее приближение Q-функции по формуле

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r(s,a) + \gamma Q(s',a') - Q(s,a)\right) \tag{3.38}$$

Какую Q-функцию такой алгоритм будет учить? Поскольку $a' \sim \mu$, где μ — стратегия взаимодействия со средой, то мы стохастически аппроксимируем Q^μ для этой самой μ, сдвигаясь в сторону решения обычного уравнения Беллмана. Формула обновления не будет эквивалентна формуле Q-learning-а (3.34), где мы сдвигались в сторону Q^π, где π было жадной стратегией по отношению к этой же Q-функции: да, в большинстве случаев (с вероятностью $1-\varepsilon$) a' будет аргмаксимумом, и обновление будет совпадать с Q-learning, но иногда a' будет оказываться тем самым «случайным» действием, случившимся из-за исследований, и наша Q-функция будет сдвигаться в сторону ценности случайных действий. Так в оценочную функцию будет попадать знание о том, что в будущем мы на каждом шаге с вероятностью ε будем обязаны выбрать случайное действие, и подобное «дёрганье» будет мешать нам проходить по краю вдоль обрыва с лавой.

Алгоритм 12: SARSA

Гиперпараметры: α — параметр экспоненциального сглаживания, ε — параметр исследований

Инициализируем $Q(s,a)$ произвольно для всех $s \in \mathcal{S}, a \in \mathcal{A}$
Наблюдаем s_0, сэмплируем $a_0 \sim \mathbf{Uniform}(\mathcal{A})$
На k-ом шаге:

1. наблюдаем r_k, s_{k+1}
2. с вероятностью ε играем $a_{k+1} \sim \mathbf{Uniform}(\mathcal{A})$, иначе $a_{k+1} = \operatorname{argmax}\limits_{a_{k+1}} Q(s_{k+1}, a_{k+1})$
3. обновляем $Q(s_k,a_k) \leftarrow Q(s_k,a_k) + \alpha\left(r_k + \gamma Q(s_{k+1},a_{k+1}) - Q(s_k,a_k)\right)$

Попробуем понять формальнее, что происходит в таком алгоритме. Можно считать, что он чередует два шага: обновление (3.38), которое учит Q^π для текущей π, и, неявно, некий аналог policy improvement-а: замены π на ε**-greedy**(Q). Именно при помощи обновлённой стратегии мы будем взаимодействовать со средой на следующем шаге, то есть полагаем $\mu \equiv \pi$ («переходим в on-policy режим»). Является ли такое обновление policy improvement-ом (допустим, для идеально посчитанной Q^π)? Вообще говоря, нет, но наши стратегии π, которые мы рассматриваем — не произвольные. Они все ε-жадные. Введём на минутку такое определение.

Определение 56: Будем говорить, что стратегия π — ***ε-мягкая*** (ε-soft), если $\forall s, a\colon \pi(a \mid s) \geq \frac{\varepsilon}{|\mathcal{A}|}$.

Утверждение 32: Если π_1 — ε-мягкая, то $\pi_2 := \varepsilon$**-greedy**(Q^{π_1}) не хуже, чем π_1.

Доказательство. Проверим выполнение теоремы 17; выглядит немного страшновато, но суть этих выкладок довольно лобовая: если мы переложим всю вероятностную массу в самое «хорошее» с точки зрения $Q^{\pi_1}(s,a)$, оставив у остальных, не самых лучших, действий ровно $\frac{\varepsilon}{|\mathcal{A}|}$, то среднее значение увеличится.

$$\begin{aligned}
V^{\pi_1}(s) &= \{\text{уравнение VQ (3.6) }\} = \sum_a \pi_1(a \mid s) Q^{\pi_1}(s,a) = \\
&= \sum_a \left(\pi_1(a \mid s) - \frac{\varepsilon}{|\mathcal{A}|}\right) Q^{\pi_1}(s,a) + \frac{\varepsilon}{|\mathcal{A}|}\sum_a Q^{\pi_1}(s,a) = \\
&= (1-\varepsilon)\sum_a \frac{\pi_1(a \mid s) - \frac{\varepsilon}{|\mathcal{A}|}}{1-\varepsilon} Q^{\pi_1}(s,a) + \frac{\varepsilon}{|\mathcal{A}|}\sum_a Q^{\pi_1}(s,a) \leq \\
&\leq (1-\varepsilon)\max_a Q^{\pi_1}(s,a)\sum_a \frac{\pi_1(a \mid s) - \frac{\varepsilon}{|\mathcal{A}|}}{1-\varepsilon} + \frac{\varepsilon}{|\mathcal{A}|}\sum_a Q^{\pi_1}(s,a)
\end{aligned}$$

Заметим, что последний переход был возможен только потому, что $\pi_1(a \mid s) - \frac{\varepsilon}{|\mathcal{A}|} > 0$ по условию, так как

π_1 — ε-мягкая по условию. Осталось заметить, что

$$\sum_a \frac{\pi_1(a \mid s) - \frac{\varepsilon}{|\mathcal{A}|}}{1-\varepsilon} = \frac{\sum\limits_a \pi_1(a \mid s) - \varepsilon}{1-\varepsilon} = 1,$$

и мы показали, что $V^{\pi_1}(s) \leq \mathbb{E}_{\pi_2(a|s)} Q^{\pi_1}(s,a)$. ■

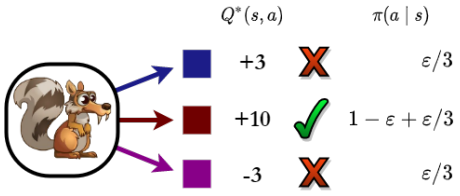
Давайте изменим постановку задачи: скажем, что мы запрещаем к рассмотрению стратегии, «похожие на детерминированные». Наложим ограничение в нашу задачу оптимизации: скажем, что стратегия обязательно должна быть ε-мягкой. В такой задаче будут свои оптимальные оценочные функции.

Определение 57: Для данного MDP оптимальными ε-мягкими оценочными функциями назовём:

$$V^*_{\varepsilon\text{-soft}}(s) \coloneqq \max_{\pi \in \varepsilon\text{-soft}} V^\pi(s)$$

$$Q^*_{\varepsilon\text{-soft}}(s,a) \coloneqq \max_{\pi \in \varepsilon\text{-soft}} Q^\pi(s,a)$$

Как тогда будет выглядеть принцип оптимальности? Раньше нужно было выбирать самое хорошее действие, но теперь так делать нельзя. Проводя аналогичные рассуждения, можно показать, что теперь оптимально выбирать самое хорошее действие с вероятностью $1-\varepsilon+\frac{\varepsilon}{|\mathcal{A}|}$, а всем остальным действиям выдавать минимально разрешённую вероятность $\frac{\varepsilon}{|\mathcal{A}|}$. Как это понять? Мы уже поняли, что «взятие ε-жадной» стратегии есть местный Policy Improvement. Если мы не можем его провести ни в одном состоянии, а то есть $\pi \equiv \varepsilon\text{-}\mathbf{greedy}(Q^\pi)$, то, видимо, придумать стратегию лучше в принципе невозможно:



Утверждение 33: Стратегия π оптимальна в классе ε-мягких стратегий тогда и только тогда, когда $\forall s, a$ таких, что $a \notin \mathbf{Argmax}\, Q^\pi(s,a)$ верно $\pi(a \mid s) = \frac{\varepsilon}{|\mathcal{A}|}$.

Скетч доказательства. Притворимся, что в будущем мы сможем выбирать действия как угодно ε-мягко, то есть для данного состояния s для каждого действия a сможем в будущем набирать $Q^*_{\varepsilon\text{-soft}}(s,a)$. Как нужно выбрать действия сейчас? Нужно решить такую задачу оптимизации:

$$\begin{cases} \mathbb{E}_{\pi(a|s)} Q^*_{\varepsilon\text{-soft}}(s,a) \to \max\limits_\pi \\ \int\limits_{\mathcal{A}} \pi(a \mid s)\, \mathrm{d}a = 1; \qquad \forall a \in \mathcal{A}\colon \pi(a \mid s) \geq \frac{\varepsilon}{|\mathcal{A}|} \end{cases}$$

Формально решая эту задачу условной оптимизации, получаем доказываемое. ■

Утверждение 34: Уравнения оптимальности, соответственно, теперь выглядят так:

$$Q^*_{\varepsilon\text{-soft}}(s,a) = r + \gamma \mathbb{E}_{s'} \left[(1-\varepsilon) \max_{a'} Q^*_{\varepsilon\text{-soft}}(s',a') + \frac{\varepsilon}{|\mathcal{A}|} \sum_{a'} Q^*_{\varepsilon\text{-soft}}(s',a') \right]$$

Доказательство. Их можно получить, например, взяв обычное уравнение QQ (3.7) и подставив вид оптимальной стратегии. ■

Мы теперь понимаем, что наша формула обновления (3.38) — просто метод решения такого уравнения оптимальности: сэмплируя a' из ε-жадной стратегии, мы просто стохастически аппроксимируем по мат.ожиданию из ε-жадной стратегии. Мы могли бы, вообще говоря, взять это мат.ожидание полностью явно, сбив таким образом дисперсию:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r(s,a) + \gamma \mathbb{E}_{a' \sim \pi} Q(s',a') - Q(s,a) \right), \tag{3.39}$$

где мат.ожидание по a' по определению π равно

$$\mathbb{E}_{a' \sim \pi} Q(s',a') = (1-\varepsilon) \max_{a'} Q(s',a') + \frac{\varepsilon}{|\mathcal{A}|} \sum_{a'} Q(s',a')$$

Такая схема называется Expected SARSA — «SARSA, в которой взяли мат.ожидание». Мы всё ещё учим Q-функцию текущей политики, но не используем сэмпл из текущей траектории, а вместо этого берём мат.ожидание по действиям из уравнения Беллмана (3.7) честно. Вообще говоря, в такой схеме мы работаем не с пятёрками s, a, r, s', a', а с четвёрками s, a, r, s' (несмотря на название).

Алгоритм 13: Expected-SARSA

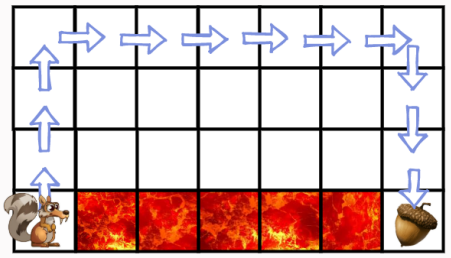
Гиперпараметры: α — параметр экспоненциального сглаживания, ε — параметр исследований

Инициализируем $Q(s, a)$ произвольно для всех $s \in \mathcal{S}, a \in \mathcal{A}$
Инициализируем π_0 произвольно
Наблюдаем s_0
На k-ом шаге:

1. играем $a_k \sim \pi_k(a_k \mid s_k)$
2. наблюдаем r_k, s_{k+1}
3. π_{k+1} есть с вероятностью ε выбрать $a_{k+1} \sim \mathbf{Uniform}(\mathcal{A})$, иначе $a_{k+1} := \underset{a_{k+1}}{\operatorname{argmax}} Q(s_{k+1}, a_{k+1})$
4. обновляем $Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha \left(r_k + \gamma \mathbb{E}_{a_{k+1} \sim \pi_{k+1}} Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k)\right)$

Соответственно, и SARSA, и Expected SARSA будут сходиться уже не к обычной оптимальной оценочной функции, а к $Q^*_{\varepsilon\text{-soft}}(s, a)$ — оптимальной оценочной функции в семействе ε-мягких стратегий.

Пример 60 — Cliff World: Попробуем запустить в MDP из примера 59 SARSA. Мы сойдёмся вовсе не к оптимальной стратегии — а «к безопасной» оптимальной стратегии. Внутри нашей оценочной функции сидит вероятность сорваться в лаву при движении вдоль неё, и поэтому кратчайший маршрут перестанет давать нам наибольшую награду просто потому, что стратегии, для которых такой маршрут был оптимален, мы перестали допускать к рассмотрению.



Принципиальное отличие схемы SARSA от Q-learning в том, что мы теперь учимся ровно на тех же сэмплах действий a', которые отправляем в среду. Наши behavior и target policy теперь совпадают: для очередного шага алгоритма нужно сделать шаг в среде при помощи текущей π, и поэтому мы должны учиться онлайн, «в on-policy режиме».

Чтобы понять, к чему это приводит, рассмотрим, что случится, если взять буфер некоторого эксперта π_{expert}, генерировать пятёрки s, a, r, s', a' из него и проводить обновления (3.38) по ним. Что мы выучим? Применяем наш стандартный ход рассуждений: $a' \sim \pi_{\text{expert}}(a' \mid s')$, и, значит, мы учим $Q^{\pi_{\text{expert}}}$! Если же мы попробуем запустить SARSA с experience replay, то есть обучаться на собственной же истории, то мы вообще творим полную ахинею: на каждом шаге мы движемся в сторону Q-функции для той стратегии π, которая породила засэмплированный переход (например, если переход был засэмплирован откуда-то из начала обучения — скорее всего случайной или хуже). Такой алгоритм не просто будет расходиться, но и не будет иметь никакого смысла. Поэтому SARSA нельзя (в таком виде, по крайней мере) запустить с реплей буфера.

§3.5. Bias-Variance Trade-Off

3.5.1. Дилемма смещения-разброса

Мы обсудили два вида бэкапов, доступных в model-free обучении: Монте-Карло бэкап и Temporal-Difference бэкап. На самом деле, они очень похожи, поскольку делают обновление вида

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(y_Q - Q(s, a)\right),$$

и отличаются лишь выбором y_Q: Монте-Карло берёт reward-to-go, а TD-backup — одношаговую бутстрапированную оценку с использованием уже имеющейся аппроксимации Q-функции:

$$y_Q := r(s, a) + \gamma Q(s', a')$$

Какой из этих двух вариантов лучше? Мы уже обсуждали недостатки Монте-Карло оценок: высокая дисперсия, необходимость играть до конца эпизодов, игнорирование структуры получаемой награды и потеря информации о соединениях состояний. Но не то, чтобы одношаговые оценки сильно лучше: на самом деле, они обладают полностью противоположными свойствами и проблемами.

Да, одношаговые оценки аппроксимируют решение одношаговых уравнений Беллмана и приближают алгоритм динамического программирования: поэтому они не теряют информации о том, на каком шаге какой сигнал от среды был получен, и сохраняют информацию о сэмплах s' из функции переходов; в том числе, как

мы видели, одношаговые алгоритмы могут использовать реплей буфер, по сути и хранящий собранную выборку таких сэмплов. Взамен в одношаговых алгоритмах возникает ***проблема распространения сигнала***.

Пример 61: Представьте, что за 100 шагов вы можете добраться до сыра (+1). Пока вы не добьётесь успеха, сигнала нет, и ваша аппроксимация Q-функции остаётся всюду нулём. Допустим, вы учитесь с одношаговых оценок с онлайн опыта. После первого успеха +1 распространится лишь в пару s, a, непосредственно предшествующей получению сыра; после второго успеха +1 распространится из пары s, a в предыдущую и так далее. Итого, чтобы распространить сигнал на 100 шагов, понадобится сделать 100 обновлений. С другой стороны, если бы использовалась Монте-Карло оценка, после первого же успеха +1 распространился бы во все пары s, a из успешной траектории.

Вместо высокой дисперсии Монте-Карло оценок в одношаговых оценках нас ждёт большое ***смещение*** (bias): если y_Q оценено через нашу же текущую аппроксимацию через бутстрапирование, то оно не является несмещённой оценкой искомой $Q^\pi(s, a)$ и может быть сколь угодно «неправильным». Как мы увидели, гарантии сходимости остаются, но естественно, что методы стохастической аппроксимации из-за смещения будут сходиться сильно дольше экспоненциального сглаживания, которому на вход поступают несмещённые оценки искомой величины. Но дисперсия y_Q в temporal difference обновлениях, например, в алгоритме Q-learning 10, конечно, сильно меньше дисперсии Монте-Карло оценок: внутри нашей аппроксимации Q-функции уже усреднены все будущие награды, то есть «взяты» все интегралы, относящиеся к будущим после первого шага наградам. Итого одношаговая оценка y_Q — случайная величина только от s', а не от всего хвоста траектории $s', a', s'' \dots$. Выбор между оценками с высокой дисперсией и отсутствием смещения (Монте-Карло) и оценками с низкой дисперсией и большим смещением (одношаговые оценки) — особая задача в обучении с подкреплением, называемая ***bias-variance trade-off***.

3.5.2. N-step Temporal Difference

Какие есть промежуточные варианты между одношаговыми оценками и Монте-Карло оценками? Давайте заглядывать в будущее не на один шаг и не до самого конца, а на N шагов. Итак, пусть у нас есть целый фрагмент траектории:

Определение 58: Фрагмент траектории $s, a, r, s', a', r', s'', a'', r'', \dots s^{(N)}, a^{(N)}$, где $s^{(N)}, a^{(N)}$ — состояние и действие, которое агент встретил через N шагов, будем называть ***роллаутом*** (rollout) длины N.

Определение 59: ***N-шаговой оценкой*** (N-step estimation) для $Q^\pi(s, a)$ назовём следующий таргет:

$$y_Q \coloneqq r + \gamma r' + \gamma^2 r'' + \dots + \gamma^{N-1} r^{(N-1)} + \gamma^N Q(s^{(N)}, a^{(N)}),$$

Такой таргет является стохастической аппроксимацией правой части N-шагового уравнения Беллмана (3.25), и формула (3.40) с таким таргетом позволяет эту систему уравнений решать в model-free режиме. По каким переменным мы заменили интегралы на Монте-Карло приближения в такой оценке? По переменным $s', a', s'', a'' \dots s^{(N)}, a^{(N)}$, которые, пусть и не все присутствуют явно в формуле, но неявно задают то уравнение, которое мы решаем. Соответственно, чтобы выучить $Q^\pi(s, a)$, нужно, чтобы состояния приходили из функции переходов, а действия — из оцениваемой стратегии π. Другими словами, роллаут, использованный для построения таргета, должен быть порождён оцениваемой стратегией.

Почему гиперпараметр N отвечает за bias-variance trade-off? Понятно, что при $N \to \infty$ оценка переходит в Монте-Карло оценку. С увеличением N всё больше интегралов заменяется на Монте-Карло оценки, и растёт дисперсия; наше же смещённое приближение будущих наград $Q^\pi(s^{(N)}, a^{(N)})$, которое может быть сколь угодно неверным, домножается на γ^N, и во столько же раз сбивается потенциальное смещение; с ростом N оно уходит в ноль. Замешивая в оценку слагаемое $r + \gamma r' + \gamma^2 r'' + \dots + \gamma^{N-1} r^{(N-1)}$ мы теряем информацию о том, что из этого в какой момент было получено, но и начинаем распространять сигнал в N раз «быстрее» одношаговой оценки.

Сразу заметим, что при $N > 1$ нам необходимо иметь в N-шаговой оценке сэмплы $a' \sim \pi(a \mid s), s'' \sim \sim p(s'' \mid s', a')$. Это означает, что мы не можем получить такую оценку с буфера: действительно, в буфере для данной четвёрки s, a, r, s' не лежит сэмпл $a' \sim \pi(a \mid s)$, ведь в произвольном буфере a' генерируется стратегией сбора данных (старой версией стратегией или «экспертом»). Само по себе это, вообще говоря, не беда: мы могли бы взять из буфера четвёрку, прогнать стратегию π, которую хотим оценить, на s' и сгенерировать себе сэмпл a'; но для него мы не сможем получить сэмпл s'' из функции переходов! Поэтому обучаться на многошаговые оценки с буфера не выйдет; по крайней мере, без какой-либо коррекции.

Также отметим, что все рассуждения одинаковы применимы как для обучения Q^π, так и V^π. Для V-функции общая формула обновления выглядит так:

$$V(s) \leftarrow V(s) + \alpha \left(y_V - V(s) \right), \tag{3.40}$$

где y_V — reward-to-go при использовании Монте-Карло оценки, и $y_V \coloneqq r(s, a) + \gamma V(s')$ для одношагового метода временных разностей, где $a \sim \pi(a \mid s)$ (сэмплирован из текущей оцениваемой стратегии), $s' \sim p(s' \mid$

$| s, a)$. Соответственно, для V-функции обучаться с буфера (без каких-либо коррекций) невозможно даже при $N = 1$, поскольку лежащий в буфере a сэмплирован из стратегии сбора данных, а не оцениваемой π.

Для простоты и наглядности будем обсуждать обучение V^π. Также введём следующие обозначения:

Определение 60: Введём такое обозначение ***N-шаговой временной разности*** (N-step temporal difference) для пары s, a:

$$\Psi_{(N)}(s,a) := \sum_{t=0}^{N-1} \gamma^t r^{(t)} + \gamma^N V(s^{(N)}) - V(s) \tag{3.41}$$

где V — текущая аппроксимация V-функции, $s, a, \dots s^{(N)}$ — роллаут, порождённый π.

Ранее мы обсуждали temporal difference, в котором мы сдвигали нашу аппроксимацию на $\Psi_{(1)}(s,a)$:

$$V(s) \leftarrow V(s) + \alpha\left(r + \gamma V(s') - V(s)\right) = \\ = V(s) + \alpha\Psi_{(1)}(s,a)$$

Теперь же мы можем обобщить наш метод, заменив оценку V-функции на многошаговую оценку:

$$V(s) \leftarrow V(s) + \alpha\Psi_{(N)}(s,a)$$

Но какое N выбрать?

3.5.3. Интерпретация через Credit Assingment

Вопрос, обучаться ли со смещённых оценок, или с тех, которые имеют большую дисперсию, имеет прямое отношение к одной из центральных проблем RL — credit assingment. На самом деле, это ровно та же самая проблема.

Рассмотрим проблему credit assingment-а: за какие будущие награды «в ответе» то действие, которое было выполнено в некотором состоянии s? Как мы обсуждали в разделе 3.2.1, «идеальное» решение задачи — значение $A^\pi(s,a)$, но на практике у нас нет точных значений оценочных функций, а есть лишь аппроксимация, допустим, $V \approx V^\pi$. Положим, мы знаем сэмпл траектории $a, s', r, s'', a'', \dots$ до конца эпизода.

С одной стороны, мы можем выдавать кредит, полностью опираясь на аппроксимацию:

$$Q^\pi(s,a) = r(s,a) + \gamma\mathbb{E}_{s'}V^\pi(s') \approx r(s,a) + \gamma V(s')$$

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \approx r(s,a) + \gamma V(s') - V(s) \tag{3.42}$$

Проблема в том, что наша аппроксимация может быть сколь угодно неверна, и выдавать полную ерунду. Более «безопасный» с этой точки зрения способ — в приближении Q-функции не опираться на аппроксимацию и использовать reward-to-go:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \approx \sum_{t\geq 0} \gamma^t r^{(t)} - V(s) \tag{3.43}$$

У этого второго способа выдавать кредит есть важное свойство, которого нет у первого: в среднем такой кредит будет больше у тех действий, которые действительно приводят к более высокой награде. То есть, если a_1, a_2 таковы, что $Q^\pi(s,a_1) > Q^\pi(s,a_2)$, то при любой аппроксимации $V(s)$ среднее значение кредита для s, a_1 будет больше s, a_2. Это и есть свойство несмещённости Монте-Карло оценок в контексте проблемы выдачи кредита.

Беда Монте-Карло в том, что в этот кредит закладывается награда не только за выбор a в s, но и за будущие решения. То есть: представьте, что через десять шагов агент выбрал действие, которое привело к +100. Эта награда +100 попадёт и в кредит всех предыдущих действий, хотя те не имели к нему отношения.

Пример 62: Допустим, мы ведём машину, и в состоянии s, где аппроксимация V-функции прогнозирует будущую награду ноль, решили выехать на встречку. Среда не сообщает нам никакого сигнала (пока не произошло никакой аварии), но аппроксимация V-функции резко упала; если мы проводим credit assingment одношаговым способом (3.42), то мы получаем сильно отрицательный кредит, который сообщает, что это действие было явно плохим.

Дальше агент следующими действиями исправил ситуацию, вернулся на правильную полосу и после решил поехать в магазин тортиков, где оказался юбилейным покупателем и получил бесплатный тортик +10. Если бы мы проводили credit assingment методом Монте-Карло (3.43), кредит получился бы сильно положительным: $+10 - 0 = +10$: мы положили, что выезд на встречку привёл к тортику.

Видно, что эти два крайних способа выдачи кредита есть в точности «градиент» $y_V - V(s)$, по которому учится V-функция в формуле (3.40). Фактически, занимаясь обучением V-функции и используя формулу

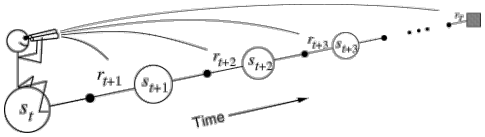
$$V(s) \leftarrow V(s) + \alpha\Psi(s,a),$$

мы выбором функции $\Psi(s, a)$ по-разному проводим credit assingment.

Таким образом видна новая интерпретации bias-variance trade-off-a: и «дисперсия» с этой точки зрения имеет смысл возложения на действие ответственности за те будущие награды, к которым это действие отношения на самом деле не имеет. Одношаговая оценка $\Psi_{(1)}$ говорит: действие влияет только на награду, которую агент получит тут же, и весь остальной сигнал будет учтён в аппроксимации V-функции. Монте-Карло оценка $\Psi_{(\infty)}$ говорит, что действие влияет на все будущие награды. А N-шаговая оценка $\Psi_{(N)}$ говорит странную вещь: действие влияет на события, который происходят с агентом в течение следующих N шагов.

Как и в любом trade-off, истина лежит где-то по середине. Однако подбирать на практике «хорошее» N, чтобы получить оценки с промежуточным смещением и дисперсией, затруднительно. Но что ещё важнее, все N-шаговые оценки на самом деле неудачные. Во-первых, они плохи тем, что не используют всю доступную информацию: если мы знаем будущее на M шагов вперёд, то странно не использовать награды за шаг за все эти M шагов, и странно не учесть прогноз нашей аппроксимации оценочной функции $V(s^{(t)})$ для всех доступных t. Во-вторых, странно, что в стационарной задаче, где всё инвариантно относительно времени, у нас появляется гиперпараметр, имеющий смысл количества шагов — времени. Поэтому нас будет интересовать далее альтернативный способ «интерполировать» между Монте-Карло и одношаговыми оценками. Мы всё равно оттолкнёмся именно от N-шаговых оценок, поскольку понятно, что эти оценки «корректны»: они направляют нас к решению уравнений Беллмана, для которых искомая V^{π} является единственной неподвижной точкой.

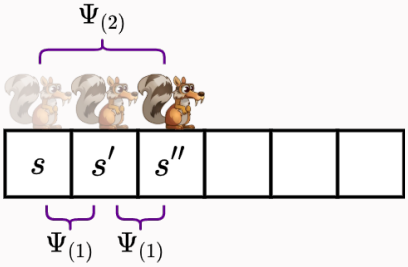
Мы придумали эти N-шаговые оценки, посмотрев на задачу под следующим углом: мы знаем сэмпл будущего (хвост траектории) и хотим выдать кредит «настоящему»: самому первому действию. Такой взгляд на оценки называется «***forward view***»: мы после выполнения a из s знаем «вперёд» своё будущее и можем обновить оценочную функцию для этой пары.



3.5.4. Backward View

Оказывается, мы можем посмотреть на задачу немного по-другому: можно, используя настоящее, обновлять кредит для прошлого. Рассмотрим эту идею, развив пример с кафе 57.

Пример 63: Ещё раз сядем в кафе (s) и захотим вернуться домой. Текущая аппроксимация даёт $-V(s) = 30$ минут. Делаем один шаг в среде: тратим одну минуту ($-r$) и обнаруживаем пробку (s'). Новая оценка времени возвращения даёт: $-V(s') = 40$ минут, соответственно с нами случилась одношаговая временная разность $\Psi_{(1)}(s, a) = 41 - 30 = 11$ минут, которая позволяет нам корректировать $V(s)$.



Давайте сделаем ещё один шаг в среде: тратим одну минуту $-r'$, видим пожар s'' и получаем новую оценку $-V(s'') = 60$ минут. Тогда мы можем посчитать как одношаговую временную разность для пары s', a', равную $\Psi_{(1)}(s', a') = 61 - 40 = 21$ минуте, и уточнить свою аппроксимацию V-функции для состояния с пробкой; а можем посчитать и двухшаговую временную разность для кафе: мы потратили две минуты $r + r'$ на два шага и наша нынешнее приближение равно 60 минутам. Итого двухшаговая временная разность равна $\Psi_{(2)}(s, a) = 62 - 30 = 32$ минуты. Forward view говорит следующее: если мы хотим учиться на двухшаговые оценки вместо одношаговых, то нам не следовало на первом шаге использовать 11 минут для обновления $V(s)$, а нужно было дождаться второго шага, узнать двухшаговую ошибку в 32 минуты и воспользоваться ей.

Но понятно, что двухшаговая ошибка это сумма двух одношаговых ошибок! Наши 32 минуты ошибки — это 11 минут ошибки после выхода из кафе в пробку плюс 21 минута ошибки от выхода из пробки в пожар. Давайте после первого шага используем уже известную часть ошибки в 11 минут, а на втором шаге, если мы готовы обучаться с двухшаговых ошибок, возьмём и добавим недостающие 21 минуту.

Формализуем эту идею. Пусть мы взаимодействуем в среде при помощи стратегии π, которую хотим оценить; также будем считать learning rate α константным. После совершения первого шага «из кафе» мы можем, зная s, a, r, s' сразу же обновить нашу V-функцию:

$$V(s) \leftarrow V(s) + \alpha\Psi_{(1)}(s, a) \tag{3.44}$$

Затем мы делаем ещё один шаг в среде, узнавая a', r', s'' и временную разность за этот случившийся шаг $\Psi_{(1)}(s', a')$. Тогда мы можем просто ещё в нашу оценку V-функции добавить слагаемое:

$$V(s) \leftarrow V(s) + \alpha\gamma\Psi_{(1)}(s', a') \tag{3.45}$$

Непосредственной проверкой легко убедиться, что суммарное обновление V-функции получится эквивалентным двухшаговому обновлению:

Утверждение 35: Последовательное применение обновлений (3.44) и (3.45) эквивалентно двухшаговому обновлению

$$V(s) \leftarrow V(s) + \alpha \Psi_{(2)}(s, a)$$

Доказательство.

$$\begin{aligned} V(s) &\leftarrow V(s) + \alpha \left(\Psi_{(1)}(s, a) + \gamma \Psi_{(1)}(s', a')\right) = \\ &= V(s) + \alpha \left(r + \gamma V(s') - V(s) + \gamma r' + \gamma^2 V(s'') - \gamma V(s')\right) = \\ &= V(s) + \alpha \left(r + \gamma r' + \gamma^2 V(s'') - V(s)\right) = \\ &= V(s) + \alpha \Psi_{(2)}(s, a) \end{aligned}$$

■

Понятно, что можно обобщить эту идею с двухшаговых ошибок на N-шаговые: действительно, ошибка за N шагов равна сумме одношаговых ошибок за эти шаги.

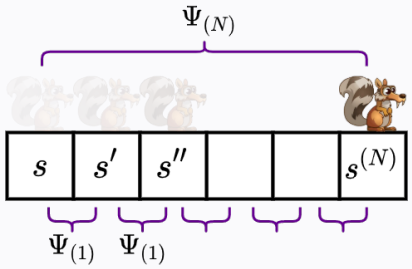
Теорема 31:

$$\Psi_{(N)}(s, a) = \sum_{t=0}^{N-1} \gamma^t \Psi_{(1)}(s^{(t)}, a^{(t)}) \tag{3.46}$$

Доказательство. Докажем по индукции. Для $N = 1$ справа стоит только одно слагаемое, $\Psi_{(1)}(s, a)$, то есть одношаговая оценка.

Пусть утверждение верно для N, докажем для $N + 1$. В правой части при увеличении N на единицу появляется одно слагаемое, то есть для доказательства достаточно показать, что

$$\Psi_{(N+1)}(s, a) = \Psi_{(N)}(s, a) + \gamma^N \Psi_{(1)}(s^{(N)}, a^{(N)}) \tag{3.47}$$



Убедимся в этом, подставив определения:

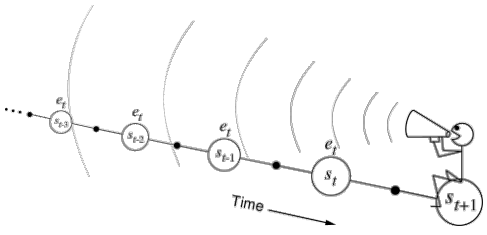
$$\begin{aligned} \Psi_{(N+1)}(s, a) &= \sum_{t=0}^{N} \gamma^t r^{(t)} + \gamma^{N+1} V(s^{(N+1)}) - V(s) = \\ &= \sum_{t=0}^{N-1} \gamma^t r^{(t)} + \gamma^N V(s^{(N)}) - V(s) + \gamma^N \left(r^{(N)} + \gamma V(s^{(N+1)}) - V(s^{(N)})\right) = \\ &= \Psi_{(N)}(s, a) + \gamma^N \Psi_{(1)}(s^{(N)}, a^{(N)}) \end{aligned}$$

■

Это наблюдение открывает, что все наши формулы обновления выражаются через одношаговые ошибки — $\Psi_{(1)}(s, a)$. Это интересный факт, поскольку одношаговая временная разность

$$\Psi_{(1)}(s, a) = r + \gamma V(s') - V(s)$$

очень похожа на reward shaping (1.7), где в качестве потенциала выбрана наша текущая аппроксимация $V(s)$. Поэтому эти ***дельты***, как их ещё иногда называют, можно интерпретировать как некие «новые награды», центрированные — которые в среднем должны быть равны нулю, если аппроксимация V-функции точная.



Итого, оказывается, мы можем на каждом шаге добавлять к оценкам V-функции ранее встречавшихся в эпизоде пар s, a только что случившуюся одношаговую ошибку $\Psi_{(1)}(s, a)$ и таким образом получать N-шаговые обновления: достаточно пару s, a, посещённую K шагов назад, обновить с весом γ^K. То есть мы начинаем действовать по-другому: зная, что было в прошлом, мы правильным образом обновляем оценочную функцию посещённых состояний из прошлого, используя временную разность за один последний шаг («настоящее»). Такой подход к обновлению оценочной функции называется, соответственно, «***backward view***», и он позволяет взглянуть на обучение оценочных функций под другим углом.

3.5.5. Eligibility Trace

Рассмотрим случай $N = +\infty$, то есть допустим, что мы готовы обновлять V-функцию с reward-to-go. Наше рассуждение, можно сказать, позволяет теперь это делать, не доигрывая эпизоды до конца: мы сразу же в ходе эпизода можем уже «начинать» сдвигать V-функцию в правильном направлении. Это позволяет запустить «как бы Монте-Карло» алгоритм даже в неэпизодичных средах. Однако, на каждом шаге нам нужно перебирать все встретившиеся ранее в эпизоде состояния, чтобы обновить их, и, если эпизоды длинные (или среда неэпизодична), это хранение истории на самом деле становится избыточным.

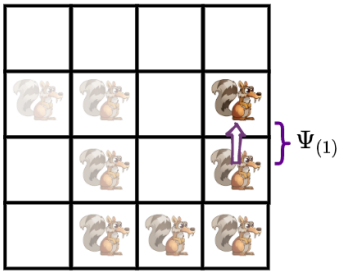
Допустим, мы сделали шаг в среде и получили на этом одном шаге какую-то одношаговую ошибку $\Psi_{(1)}$. Рассмотрим какое-нибудь состояние s. С каким весом, помимо learning rate, нужно добавить эту ошибку к нашей текущей аппроксимации? Это состояние в течение прошлого эпизода было, возможно, посещено несколько раз, и за каждое посещение вес увеличивается на γ^K, где K — число шагов с момента посещения. Такой счётчик можно сохранить в памяти: заведём вектор $e(s)$ размером с число состояний, проинициализируем его нулём и далее на t-ом шаге будем обновлять следующим образом:

$$e_t(s) := \begin{cases} \gamma e_{t-1}(s) + 1 & \text{если } s = s_t \\ \gamma e_{t-1}(s) & \text{иначе} \end{cases} \tag{3.48}$$

После этого на каждом шаге мы будем добавлять текущую одношаговую ошибку, временную разность $\Psi_{(1)}(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t)$, ко *всем* состояниям с коэффициентом $e_t(s)$.

Определение 61: Будем называть $e_t(s)$ ***следом*** (eligibility trace) для состояния s в момент времени t эпизода коэффициент, с которым алгоритм обновляет оценочную функцию $V(s)$ на t-ом шаге при помощи текущей одношаговой ошибки $\Psi_{(1)}(s_t, a_t)$:

$$V(s) \leftarrow V(s) + \alpha e_t(s) \Psi_{(1)}(s_t, a_t) \tag{3.49}$$



Допустим, эпизод доигран до конца, и мы в алгоритме используем формулу (3.49) со следом (3.48). Одношаговое обновление будет превращено в двухшаговое, двухшаговое — в трёхшаговое и так далее до N-шагового, где N — количество шагов до конца эпизода. Таким образом (если в ходе таких обновлений learning rate и оцениваемая стратегия не меняется) наши обновления в точности соответствуют Монте-Карло.

Важно, что eligibility trace имеет физический смысл «кредита», который выдан решениям, принятым в состоянии s: это та степень ответственности, с которой выбранное в этом состоянии действия влияют на события настоящего, на получаемые сейчас награды (временные разности). Действительно, обновление (3.49) говорит следующее: прогноз будущей награды в состоянии s нужно увеличить с некоторым learning rate-ом на получаемую награду («$\Psi_{(1)}(s_t, a_t)$») , домноженную на степень ответственности решений в s за события настоящего («$e_t(s)$»). И пока мы используем Монте-Карло обновления, кредит ведёт себя так: как только мы принимаем в s решение, он увеличивается на единичку и дальше не затухает. Домножение на γ можно не интерпретировать как затухание, поскольку это вызвано дисконтированием награды в нашей задаче, «затуханием» самой награды со временем. То есть мы рисуем мелом на стене «я здесь был», и выдаём постоянный кредит этому состоянию.

А что, если мы не хотим использовать бесконечношаговые (Монте-Карло) обновления? Мы помним, что это одна крайность, в которой обновления имеют большую дисперсию. Можно броситься в другую крайность: если мы хотим ограничиться лишь, допустим, одношаговыми, то мы можем использовать просто «другое» определение следа:

$$e_t(s) := \begin{cases} 1 & \text{если } s = s_t \\ 0 & \text{иначе} \end{cases}$$

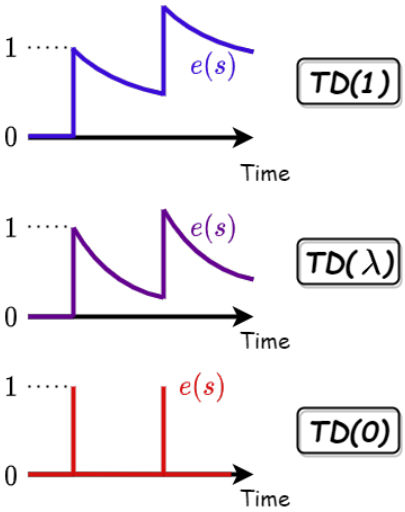
То есть для одношаговых оценок нам нужно домножать след не на γ, а на ноль. Тогда вектор $e(s)$ на каждой итерации будет нулевым за исключением одного лишь только что встреченного состояния s_t, для которого он будет равен единице, и обновление 3.49 будет эквивалентно обычному одношаговому temporal difference. В таком кредите решение, принятое в s, влияет только на то, что произойдёт на непосредственно том же шаге; «мел на стене испаряется мгновенно».

3.5.6. TD(λ)

Как можно интерполировать между Монте-Карло обновлениями и одношаговыми с точки зрения backward view? Раз eligibility trace может затухать в γ раз, а может в 0, то, вероятно, можно тушить его и любым другим промежуточным способом. Так мы теперь можем придумать другой вид «промежуточных оценок» между Монте-Карло и одношаговыми. Пусть на очередном моменте времени след для состояния s затухает сильнее, чем в γ раз, но больше, чем в ноль; что тогда произойдёт?

После момента посещения состояния s след для него вырастет на единичку и оценочная функция обновится следующим образом:

$$V(s) \leftarrow V(s) + \alpha\Psi_{(1)}(s,a)$$



Допустим, на следующем шаге мы выбрали $\lambda_1 \in [0,1]$ — «коэффициент затухания» — и потушили след не с коэффициентом γ, а с коэффициентом $\gamma\lambda_1$. Тогда дальше мы добавим новую текущую временную разность $\Psi_{(1)}(s',a')$ с коэффициентом $\lambda\gamma$, получая суммарно следующее обновление:

$$V(s) \leftarrow V(s) + \alpha\left(\Psi_{(1)}(s,a) + \lambda_1\gamma\Psi_{(1)}(s',a')\right),$$

которое в силу (3.46) для $N=2$ преобразуется в:

$$V(s) \leftarrow V(s) + \alpha\left((1-\lambda_1)\Psi_{(1)}(s,a) + \lambda_1\Psi_{(2)}(s,a)\right)$$

Таким образом, мы ***заансамблировали*** одношаговое и двухшаговое обновление. Из этого видно, что такая процедура корректна: V^π является неподвижной точкой как одношагового, так и двухшагового уравнения Беллмана, и значит неподвижной точкой любой их выпуклой комбинации.

С точки зрения кредита, мы сказали, что решение, принятое в s, влияет на то, что случится через 2 шага, но не так сильно, как на то, что случится через 1 шаг: степень ответственности за один шаг затухла в λ_1 раз. Мы пользуемся здесь следующим прайором из реальной жизни: решения более вероятно влияют на ближайшее будущее, нежели чем на далёкое.

Для понятности проведём ещё один шаг рассуждений. Допустим, мы сделали ещё один шаг в среде и увидели $\Psi_{(1)}(s'',a'')$; потушили eligibility trace $e(s)$, на этот раз, в $\gamma\lambda_2$ раз, где $\lambda_2 \in [0,1]$. Тогда след стал равен $e(s) = \gamma^2\lambda_1\lambda_2$, и мы получим следующее обновление:

$$V(s) \leftarrow V(s) + \alpha\left((1-\lambda_1)\Psi_{(1)}(s,a) + \lambda_1\Psi_{(2)}(s,a) + \gamma^2\lambda_1\lambda_2\Psi_{(1)}(s'',a'')\right)$$

Вспоминая формулу (3.47), последние два слагаемых преобразуются:

$$\Psi_{(2)}(s,a) + \gamma^2\lambda_2\Psi_{(1)}(s'',a'') = (1-\lambda_2)\Psi_{(2)}(s,a) + \lambda_2\Psi_{(3)}(s,a)$$

То есть долю λ_2 двухшаговой ошибки мы превращаем в трёхшаговое, а долю $1-\lambda_2$ — нет. Суммарное обновление становится таким ансамблем:

$$V(s) \leftarrow V(s) + \alpha\left((1-\lambda_1)\Psi_{(1)}(s,a) + \lambda_1(1-\lambda_2)\Psi_{(2)}(s,a) + \lambda_1\lambda_2\Psi_{(3)}(s,a)\right)$$

И так далее. Интерпретировать это можно так. Если мы тушим след в γ раз, то на очередном шаге обновления мы «превращаем» N-шаговое обновление в $N+1$-шаговое. Если мы тушим след полностью, зануляя его, то мы «отказываемся превращать» N-шаговое обновление в $N+1$-шаговое. Оба варианта корректны, и поэтому мы, выбирая $\lambda_t \in [0,1]$, решаем заансамблировать их: мы возьмём и долю λ_t N-шагового обновления «превратим» в $N+1$-шаговое, а долю $1-\lambda_t$ трогать не будем и оставим без изменения. Поэтому λ_t ещё называют «коэффициентом смешивания».

Мы уже обсуждали, что странно проводить credit assingment нестационарно, то есть чтобы в процедуре была какая-то зависимость от времени, прошедшего с момента посещения состояния, поэтому коэффициент затухания $\lambda \in [0,1]$ обычно полагают не зависящем от момента времени, каким-то константным гиперпараметром, отвечающим за bias-variance trade-off. Естественно, $\lambda = 1$ соответствует Монте-Карло обновлениям, $\lambda = 0$ — одношаговым.

Определение 62: Будем называть ***temporal difference обновлением*** с параметром $\lambda \in [0,1]$ («обновление TD(λ)») обновление (3.49) со следом $e_t(s)$, проинициализированным нулём и далее определённым следующим образом:

$$e_t(s) := \begin{cases} \gamma\lambda e_{t-1}(s) + 1 & \text{если } s = s_t \\ \gamma\lambda e_{t-1}(s) & \text{иначе} \end{cases}$$

Формула следа задаёт алгоритм в парадигме backward view. Естественно, что любая оценка, придуманная в терминах backward view, то есть записанная в терминах следа (в явном виде хранящего «кредит» ответственности решений для каждого состояния), переделывается в парадигме forward view (как и наоборот), когда мы, используя сэмплы будущего, строим некоторую оценку Advantage $\Psi(s,a) \approx A^\pi(s,a)$ и просто сдвигаем по нему значение V-функции:

$$V(s) \leftarrow V(s) + \alpha\Psi(s,a) \tag{3.50}$$

Какому обновлению в парадигме forward view соответствует обновление TD(λ)?

Теорема 32 — Эквивалентные формы TD(λ): Обновление TD($\boldsymbol{\lambda}$) эквивалентно (3.50) с оценкой

$$\boldsymbol{\Psi(s,a) \coloneqq \sum_{t\geq 0} \gamma^t \lambda^t \Psi_{(1)}(s^{(t)}, a^{(t)}) = (1-\lambda) \sum_{t>0} \lambda^{t-1} \Psi_{(t)}(s,a)} \tag{3.51}$$

Доказательство. Составим такую табличку: какое суммарное обновление у нас получается в TD($\boldsymbol{\lambda}$) после $\boldsymbol{t}$ шагов в среде. Справа запишем, с какими весами входят разные $\boldsymbol{N}$-шаговые оценки в получающийся ансамбль.

Step	**Update**	$\boldsymbol{\Psi_{(1)}(s,a)}$	$\boldsymbol{\Psi_{(2)}(s,a)}$	$\boldsymbol{\Psi_{(3)}(s,a)}$	$\boldsymbol{\dots}$	$\boldsymbol{\Psi_{(N)}(s,a)}$
1	$\boldsymbol{\Psi_{(1)}(s,a)}$	1	0	0		0
2	$\boldsymbol{\Psi_{(1)}(s,a) + \gamma\lambda\Psi_{(1)}(s',a')}$	$\boldsymbol{1-\lambda}$	$\boldsymbol{\lambda}$	0		
3	$\boldsymbol{\Psi_{(1)}(s,a) + \gamma\lambda\Psi_{(1)}(s',a') + (\gamma\lambda)^2\Psi_{(1)}(s'',a'')}$	$\boldsymbol{1-\lambda}$	$\boldsymbol{(1-\lambda)\lambda}$	$\boldsymbol{\lambda^2}$		0
$\vdots$					$\ddots$	
$\boldsymbol{N}$	$\boldsymbol{\sum_{t\geq 0}^{N} (\gamma\lambda)^t \Psi_{(1)}(s^{(t)}, a^{(t)})}$	$\boldsymbol{1-\lambda}$	$\boldsymbol{(1-\lambda)\lambda}$	$\boldsymbol{(1-\lambda)\lambda^2}$		$\boldsymbol{\lambda^N}$

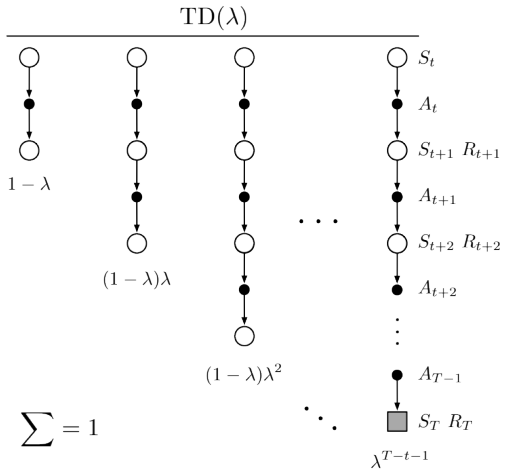
Продолжая строить такую табличку, можно по индукции увидеть, что после окончания эпизода суммарное обновление V-функции получится следующим:

$$\boldsymbol{V(s) \leftarrow V(s) + \alpha \sum_{t\geq 0} (\gamma\lambda)^t \Psi_{(1)}(s^{(t)}, a^{(t)})}$$

Это и есть суммарное обновление V-функции по завершении эпизода. ■

Итак, полученная формула обновления имеет две интерпретации. Получается, что подобный ансамбль многошаговых оценок эквивалентен дисконтированной сумме будущих одношаговых ошибок («модифицированных наград» с потенциалом $\boldsymbol{V(s)}$), где коэффициент дисконтирования равен $\boldsymbol{\gamma\lambda}$; исходный коэффициент $\boldsymbol{\gamma}$ отвечает за затухание ценности наград со временем из исходной постановки задачи, а $\boldsymbol{\lambda}$ соответствует затуханию кредита ответственности действия за полученные в будущем награды.

А с другой стороны можно интерпретировать TD($\boldsymbol{\lambda}$) как ансамбль многошаговых оценок разной длины. Мы взяли одношаговую оценку с весом $\boldsymbol{\lambda}$, двухшаговую с весом $\boldsymbol{\lambda^2}$, $\boldsymbol{N}$-шаговую с весом $\boldsymbol{\lambda^N}$ и так далее. Сумма весов в ансамбле, как водится, должна равняться единице, отсюда в формуле домножение на $\boldsymbol{1-\lambda}$.



Если через $\boldsymbol{N}$ шагов после оцениваемого состояния $\boldsymbol{s}$ эпизод закончился, то все многошаговые оценки длины больше $\boldsymbol{N}$, понятно, совпадают с $\boldsymbol{N}$-шаговой и равны reward-to-go. Следовательно, в такой ситуации reward-to-go по определению замешивается в оценку с весом $\boldsymbol{(1-\lambda)(\lambda^{N-1} + \lambda^N + \dots) = \lambda^{N-1}}$.

Мы привыкли, что любая формула обновления для нас — стохастическая аппроксимация решения какого-то уравнения. TD($\boldsymbol{\lambda}$) не исключение. Если $\boldsymbol{N}$-шаговая оценка направляет нас в сторону решения $\boldsymbol{N}$-шагового уравнения Беллмана $\boldsymbol{V^\pi = \mathfrak{B}^N V^\pi}$, то ансамбль из оценок направляет нас в сторону решения ансамбля $\boldsymbol{N}$-шаговых уравнений Беллмана:

$$\boldsymbol{V^\pi = (1-\lambda)(\mathfrak{B}V^\pi + \lambda\mathfrak{B}^2 V^\pi + \lambda^2\mathfrak{B}^3 V^\pi + \dots) = (1-\lambda) \sum_{N>0} \lambda^{N-1} \mathfrak{B}^N V^\pi} \tag{3.52}$$

Поскольку $\boldsymbol{V^\pi}$ является неподвижной точкой для операторов $\boldsymbol{\mathfrak{B}^N}$ для всех $\boldsymbol{N}$, то и для их выпуклой комбинации, «ансамбля», он тоже будет неподвижной точкой. Итого TD($\boldsymbol{\lambda}$) дало нам прикольную идею: мы не могли выбрать одну из многошаговых оценок, и поэтому взяли их все сразу.

Итак, мы получили алгоритм TD($\boldsymbol{\lambda}$) оценивания стратегии, или temporal difference с параметром $\boldsymbol{\lambda}$, который при $\boldsymbol{\lambda = 1}$ эквивалентен Монте-Карло алгоритму (с постоянным обновлением V-функции «по ходу эпизода»), а при $\boldsymbol{\lambda = 0}$ эквивалентен одношаговому temporal difference методу, который мы, в частности, применяли в Q-learning и SARSA для оценивания стратегии.

Алгоритм 14: TD(λ)

Вход: π — стратегия
Гиперпараметры: α — параметр экспоненциального сглаживания, $\lambda \in [0, 1]$ — степень затухания следа

Инициализируем $V(s)$ произвольно для всех $s \in \mathcal{S}$
Инициализируем $e(s)$ нулём для всех $s \in \mathcal{S}$
Наблюдаем s_0
На k-ом шаге:

1. выбираем $a_k \sim \pi(a_k \mid s_k)$
2. играем a_k и наблюдаем r_k, s_{k+1}
3. обновляем след $e(s_k) \leftarrow e(s_k) + 1$
4. считаем одношаговую ошибку $\Psi_{(1)} := r_k + \gamma V(s_{k+1}) - V(s_k)$
5. для всех s обновляем $V(s) \leftarrow V(s) + \alpha e(s) \Psi_{(1)}$
6. для всех s обновляем $e(s) \leftarrow \gamma \lambda e(s)$

Выход: $V(s)$

Мы обсуждали и выписали этот алгоритм для V-функции для задачи именно оценивания стратегии; естественно, мы могли бы сделать это для Q-функции или добавить policy improvement после, например, каждого шага в среде, получив табличный алгоритм обучения стратегии. Позже в разделе 3.5.7 мы рассмотрим формулировку теоремы о сходимости таких алгоритмов для ещё более общей ситуации.

Очевидно, TD(λ) обновление не эквивалентно никаким N-шаговым temporal difference формулам: в нём замешана как Монте-Карло оценка, то есть замешана вся дальнейшая награда (весь будущий сигнал), так и приближения V-функции во всех промежуточных состояний (при любом $\lambda \in (0, 1)$). Гиперпараметр λ также не имеет смысла времени, и поэтому на практике его легче подбирать.

Полезность TD(λ) в том, что λ непрерывно и позволяет более гладкую настройку «длины следа». На практике алгоритмы будут чувствительны к выбору λ в намного меньшей степени, чем к выбору N. При этом даже если $\lambda < 1$, в оценку «поступает» информация о далёкой награде, и использование TD(λ) позволит бороться с проблемой распространения сигнала.

Всюду далее в ситуациях, когда нам понадобится разрешать bias-variance trade-off, мы будем обращаться к формуле forward view TD(λ) обновления (3.51). Однако как отмечалось ранее, для работы с многошаговыми оценками, а следовательно и с обновлением (3.51) TD(λ), необходимо работать в on-policy режиме, то есть иметь сэмплы взаимодействия со средой именно оцениваемой стратегии $\pi(a \mid s)$, и поэтому возможность разрешать bias-variance trade-off у нас будет только в on-policy алгоритмах.

3.5.7. Retrace(λ)

Что же тогда делать в off-policy режиме? Итак, пусть дан роллаут $s, a, r, s', a', r', s'', \ldots$, где действия сэмплированы из стратегии μ. Мы хотим при этом провести credit assingment для стратегии π, то есть понять, как обучать $V \approx V^\pi$ или $Q \approx Q^\pi$.

Проблема в том, что на самом деле это не всегда даже в принципе возможно. Представим, что a таково, что $\pi(a \mid s) = 0$. Тогда в роллауте хранится информация о событиях, которые произойдут с агентом, использующем стратегию π, с вероятностью 0. Понятно, что никакой полезной информации о том, как менять аппроксимацию V-функции, мы из таких данных не получим. Поэтому для детерминированных стратегий задача обучения с многошаговых оценок в off-policy режиме может запросто оказаться бессмысленной.

Пример 64: Допустим, стратегия μ с вероятностью один в первом же состоянии прыгает в лаву. Мы же хотим посчитать $V^\pi(s)$, будущую награду, для стратегии π, которая с вероятностью один не прыгает в лаву, а кушает тортики. Поскольку в задаче RL функция переходов $p(s' \mid s, a)$ для разных a может быть произвольно разной, мы ничего не можем сказать о ценности одного действия по информации о другом действии.

Возможность в принципе обучаться off-policy в Q-learning обеспечивалась тем, что, когда мы учим Q-функцию с одношаговых оценок, нам для любых s, a, которые можно взять из буфера, достаточно лишь сэмпла s'. При этом сэмпл $a' \sim \pi(a' \mid s')$ мы всегда сможем сгенерировать «онлайн», прогнав на взятом из буфера s' оцениваемую стратегию. Обычно в off-policy режиме учат именно Q-функцию, что важно в том числе тем, что по крайней мере одношаговая оценка будет доступна всегда. Если же a' из буфера такого, что $\pi(a' \mid s') = 0$,

то любые наши коррекции схлопнутся в одношаговую оценку (или же будут теоретически некорректны), но по крайней мере хоть что-то мы сможем сделать.

Итак, попробуем разрешить bias-variance trade-off-а для Q-функции. Для этого снова вернёмся к идее следа. Допустим, для взятых из буфера s, a, r, s' мы составили одношаговое обновление:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s', a'_\pi) - Q(s,a)),$$

где $a'_\pi \sim \pi(\cdot \mid s')$, положив неявно значение следа $e(s,a) = 1$ (след при обучении Q-функции зависит от пары s, a). Также лучше, если есть такая возможность, использовать не сэмпл a'_π, а усреднить по нему (аналогично тому, как было проделано в формуле (3.39) Expected SARSA):

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \mathbb{E}_{a'_\pi \sim \pi} Q(s', a'_\pi) - Q(s,a)) \tag{3.53}$$

Затем возьмём из буфера a', r', s''. Какое значение следа мы можем выбрать? Можно $e(s,a) = 0$, оставив одношаговое обновление и «не превращая» его в двухшаговое, это одна крайность. А как превратить обновление в двухшаговое целиком? Хотелось бы прибавить

$$\gamma(r' + \gamma \mathbb{E}_{a''_\pi \sim \pi} Q(s'', a''_\pi) - \mathbb{E}_{a'_\pi \sim \pi} Q(s', a'_\pi)),$$

где $s'' \sim p(s'' \mid s', a'_\pi)$. Однако в буфере у нас нет такого сэмпла, а вместо него есть сэмпл $s'' \sim p(s'' \mid s', a')$. Технически, ошибка за второй шаг является случайной величиной от a', который должен приходить из π, когда у нас есть сэмпл лишь из μ. Поэтому здесь необходимо применить importance sampling коррекцию, после которой ошибка за второй шаг принимает следующий вид[7]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\gamma \frac{\pi(a' \mid s')}{\mu(a' \mid s')} \left(r' + \gamma \mathbb{E}_{a''_\pi \sim \pi} Q(s'', a''_\pi) - Q(s', a')\right), \tag{3.54}$$

где a', r', s'' — взяты из буфера.

Утверждение 36: Последовательное применение обновлений (3.53) и (3.54) является корректным двухшаговым обновлением, то есть эквивалентно

$$Q(s,a) \leftarrow Q(s,a) + \alpha(y(Q) - Q(s,a)),$$

где среднее значение $y(Q)$ по всей заложенной в ней стохастике является правой частью двухшагового уравнения Беллмана для Q-функции:

$$\mathbb{E} y(Q) = \mathbb{E}_{s'} \mathbb{E}_{a' \sim \pi} \mathbb{E}_{s''} \mathbb{E}_{a'' \sim \pi} \left[r(s,a) + \gamma r(s', a') + \gamma^2 Q(s'', a'')\right]$$

Доказательство. После двух рассматриваемых обновлений получается, что «целевая переменная», таргет, равен:

$$y(Q) = r + \gamma \mathbb{E}_{a'_\pi \sim \pi} Q(s', a'_\pi) + \gamma \frac{\pi(a' \mid s')}{\mu(a' \mid s')} \left(r' + \gamma \mathbb{E}_{a''_\pi \sim \pi} Q(s'', a''_\pi) - Q(s', a')\right),$$

где случайными величинами являются s', a', s'', и $a' \sim \mu(a' \mid s')$. Возьмём среднее по этим величинам:

$$\mathbb{E} y(Q) = \mathbb{E}_{s'} \left[r + \gamma \mathbb{E}_{a'_\pi \sim \pi} Q(s', a'_\pi) + \gamma \mathbb{E}_{a' \sim \mu} \frac{\pi(a' \mid s')}{\mu(a' \mid s')} \left(r' + \gamma \mathbb{E}_{s''} \mathbb{E}_{a''_\pi \sim \pi} Q(s'', a''_\pi) - Q(s', a')\right)\right]$$

Раскроем importance sampling коррекцию, воспользовавшись

$$\mathbb{E}_{a' \sim \mu} \frac{\pi(a' \mid s')}{\mu(a' \mid s')} f(a') = \mathbb{E}_{a' \sim \pi} f(a'),$$

[7] видно, что в отношении последнего слагаемого возможны вариации, например, ошибку за второй шаг можно оценить как

$$\gamma \frac{\pi(a' \mid s')}{\mu(a' \mid s')} \left(r' + \gamma \mathbb{E}_{a''_\pi \sim \pi} Q(s'', a''_\pi) - \mathbb{E}_{a'_\pi \sim \pi} Q(s', a'_\pi)\right),$$

или даже не корректировать последнее слагаемое importance sampling коррекцией, так как в нём не требуется брать a' обязательно из буфера:

$$\gamma \frac{\pi(a' \mid s')}{\mu(a' \mid s')} \left(r' + \gamma \mathbb{E}_{a''_\pi \sim \pi} Q(s'', a''_\pi)\right) - \gamma \mathbb{E}_{a'_\pi \sim \pi} Q(s', a'_\pi).$$

Далее в формулах предполагается вариант, рассматриваемый в статье про Retrace(λ).

получим:

$$\mathbb{E}\boldsymbol{y}(\boldsymbol{Q}) = \mathbb{E}_{\boldsymbol{s}'}\mathbb{E}_{\boldsymbol{a}'\sim\pi}\mathbb{E}_{\boldsymbol{s}''}\mathbb{E}_{\boldsymbol{a}''_{\pi}\sim\pi}\left[\boldsymbol{r} + \boldsymbol{\gamma}\boldsymbol{Q}(\boldsymbol{s}', \boldsymbol{a}') + \boldsymbol{\gamma}\left(\boldsymbol{r}' + \boldsymbol{\gamma}\boldsymbol{Q}(\boldsymbol{s}'', \boldsymbol{a}''_{\pi}) - \boldsymbol{Q}(\boldsymbol{s}', \boldsymbol{a}')\right)\right]$$

Осталось заметить, что слагаемое $\boldsymbol{\gamma}\boldsymbol{Q}(\boldsymbol{s}', \boldsymbol{a}')$ по аналогии с on-policy режимом сокращается. ■

Таким образом, одношаговую ошибку за второй шаг для получения полного превращения одношагового обновления в двухшаговое необходимо добавить не с весом $\boldsymbol{\gamma}$, как в on-policy режиме, а с весом $\boldsymbol{\gamma}\frac{\boldsymbol{\pi}(\boldsymbol{a}'|\boldsymbol{s}')}{\boldsymbol{\mu}(\boldsymbol{a}'|\boldsymbol{s}')}$.

Продолжая рассуждение дальше, можно получить, что одношаговая ошибка через $\boldsymbol{t}$ шагов после выбора оцениваемого действия $\boldsymbol{a}$ в состоянии $\boldsymbol{s}$ в off-policy режиме зависит от случайных величин $\boldsymbol{s}', \boldsymbol{a}', \boldsymbol{s}'', \ldots \boldsymbol{s}^{(t+1)}$, и поэтому importance sampling коррекция для неё будет равна:

$$\prod_{i=1}^{\hat{t}=t} \frac{\boldsymbol{\pi}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})\boldsymbol{p}(\boldsymbol{s}^{(i+1)} \mid \boldsymbol{s}^{(i)}, \boldsymbol{a}^{(i)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})\boldsymbol{p}(\boldsymbol{s}^{(i+1)} \mid \boldsymbol{s}^{(i)}, \boldsymbol{a}^{(i)})} = \prod_{i=1}^{i=t} \frac{\boldsymbol{\pi}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}$$

Здесь и далее считается, что при $\boldsymbol{t} = \boldsymbol{0}$ подобные произведения равны единице.

Получается, что для того, чтобы строить оценку максимальной длины, нужно на $\boldsymbol{t}$-ом шаге домножать след на $\boldsymbol{\gamma}\frac{\boldsymbol{\pi}(\boldsymbol{a}^{(t)}|\boldsymbol{s}^{(t)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(t)}|\boldsymbol{s}^{(t)})}$. Итоговую формулу обновления часто записывают в следующем виде:

$$\boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}) \leftarrow \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}) + \boldsymbol{\alpha}\sum_{\boldsymbol{t}\geq\boldsymbol{0}} \boldsymbol{\gamma}^{\boldsymbol{t}} \left(\prod_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{i}=\boldsymbol{t}} \frac{\boldsymbol{\pi}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}\right) \boldsymbol{\Psi}_{(1)}(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}), \tag{3.55}$$

где

$$\boldsymbol{\Psi}_{(1)}(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}) = \boldsymbol{r}^{(t)} + \boldsymbol{\gamma}\mathbb{E}_{\boldsymbol{a}_{\pi}^{(t+1)}\sim\boldsymbol{\pi}}\boldsymbol{Q}(\boldsymbol{s}^{(t+1)}, \boldsymbol{a}_{\pi}^{(t+1)}) - \boldsymbol{Q}(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}) \tag{3.56}$$

Мы получили «off-policy» Монте-Карло оценку в терминах следа. Теперь по аналогии с TD($\boldsymbol{\lambda}$) проинтерполируем между одношаговыми (где след зануляется после первого шага) и бесконечношаговыми обновлениями (где след есть importance sampling дробь): оказывается, оценка будет корректна при любом промежуточном значении следа. Чтобы записать это формально, перепишем формулу (3.55) в следующем виде:

$$\boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}) \leftarrow \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}) + \boldsymbol{\alpha}\sum_{\boldsymbol{t}\geq\boldsymbol{0}} \boldsymbol{\gamma}^{\boldsymbol{t}} \left(\prod_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{i}=\boldsymbol{t}} \boldsymbol{c}_{\boldsymbol{i}}\right) \boldsymbol{\Psi}_{(1)}(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}), \tag{3.57}$$

где $\boldsymbol{c}_{\boldsymbol{i}}$ — коэффициенты затухания следа: в on-policy они могли быть в диапазоне $[\boldsymbol{0}, \boldsymbol{1}]$ (и мы выбирали его равным гиперпараметру $\boldsymbol{\lambda}$), а здесь, в off-policy режиме, он может быть в диапазоне

$$\boldsymbol{c}_{\boldsymbol{i}} \in \left[\boldsymbol{0}, \frac{\boldsymbol{\pi}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}\right]$$

То, что при любом выборе способа затухания следа табличные алгоритмы, использующие оценку (3.57), будут сходиться — один из ключевых и самых общих результатов табличного RL. Приведём несколько нестрогую формулировку этой теоремы:

Теорема 33 — Retrace: Пусть число состояний и действий конечно, таблица $\boldsymbol{Q}_{\boldsymbol{0}}(\boldsymbol{s}, \boldsymbol{a})$ проинициализирована произвольно. Пусть на $\boldsymbol{k}$-ом шаге алгоритма для каждой пары $\boldsymbol{s}, \boldsymbol{a}$ ячейка таблицы обновляется по формуле

$$\boldsymbol{Q}_{\boldsymbol{k}+\boldsymbol{1}}(\boldsymbol{s}, \boldsymbol{a}) \leftarrow \boldsymbol{Q}_{\boldsymbol{k}}(\boldsymbol{s}, \boldsymbol{a}) + \boldsymbol{\alpha}_{\boldsymbol{k}}(\boldsymbol{s}, \boldsymbol{a})\sum_{\boldsymbol{t}\geq\boldsymbol{0}} \boldsymbol{\gamma}^{\boldsymbol{t}} \left(\prod_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{i}=\boldsymbol{t}} \boldsymbol{c}_{\boldsymbol{i}}\right) \boldsymbol{\Psi}_{(1)}(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}),$$

$$\boldsymbol{\Psi}_{(1)}(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}) = \boldsymbol{r}^{(t)} + \boldsymbol{\gamma}\mathbb{E}_{\boldsymbol{a}_{\pi}^{(t+1)}\sim\boldsymbol{\pi}_{\boldsymbol{k}}}\boldsymbol{Q}_{\boldsymbol{k}}(\boldsymbol{s}^{(t+1)}, \boldsymbol{a}_{\pi}^{(t+1)}) - \boldsymbol{Q}(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)})$$

где $\boldsymbol{\mathcal{T}} \sim \boldsymbol{\mu}_{\boldsymbol{k}} \mid \boldsymbol{s}_{\boldsymbol{0}} = \boldsymbol{s}, \boldsymbol{a}_{\boldsymbol{0}} = \boldsymbol{a}$ сгенерирована произвольной стратегией сбора данных $\boldsymbol{\mu}_{\boldsymbol{k}}$ (причём не обязательно стационарной), learning rate $\boldsymbol{\alpha}_{\boldsymbol{k}}(\boldsymbol{s}, \boldsymbol{a})$ — случайно, $\boldsymbol{\pi}_{\boldsymbol{k}}$ произвольно, и коэффициенты следа — любые в диапазоне

$$\boldsymbol{c}_{\boldsymbol{i}} \in \left[\boldsymbol{0}, \frac{\boldsymbol{\pi}_{\boldsymbol{k}}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}{\boldsymbol{\mu}_{\boldsymbol{k}}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}\right].$$

Тогда, если с вероятностью 1 learning rate удовлетворяет условиям Роббинса-Монро (3.26), а стратегия $\boldsymbol{\pi}_{\boldsymbol{k}}$ с вероятностью 1 становится жадной по отношению к $\boldsymbol{Q}_{\boldsymbol{k}}$ в пределе $\boldsymbol{k} \to \boldsymbol{\infty}$, то при некоторых технических ограничениях с вероятностью 1 $\boldsymbol{Q}_{\boldsymbol{k}}$ сходится к оптимальной Q-функции $\boldsymbol{Q}^*$, а $\boldsymbol{\pi}_{\boldsymbol{k}}$, соответственно, к оптимальной стратегии.

Без доказательства; интересующиеся могут обратиться к оригинальной статье по Retrace. ■

Пользуясь этой теоремой, мы можем в полной аналогии с TD($\boldsymbol{\lambda}$) выбрать гиперпараметр $\boldsymbol{\lambda} \in [\mathbf{0}, \mathbf{1}]$, и считать след по формуле

$$\boldsymbol{c_i} = \boldsymbol{\lambda} \frac{\boldsymbol{\pi}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}$$

В частности, в on-policy режиме $\boldsymbol{\pi} \equiv \boldsymbol{\mu}$ мы получим коэффициент затухания следа, равный просто $\boldsymbol{\lambda}$. Это очень удобно, но на практике неприменимо.

Такая оценка страдает сразу от двух проблем. Первая проблема — типичная: ***затухающий след*** (vanishing trace), когда $\boldsymbol{\mu}(\boldsymbol{a}^{(t)} \mid \boldsymbol{s}^{(t)}) \gg \boldsymbol{\pi}(\boldsymbol{a}^{(t)} \mid \boldsymbol{s}^{(t)})$ для какого-то $\boldsymbol{t}$, и соответствующий множитель близок к нулю. Такое случится, если, например, $\boldsymbol{\mu}$ выбирает какие-то действия, которые $\boldsymbol{\pi}$ выбирает редко, что типично. В предельном случае для детерминированных стратегий может быть такое, что числитель коэффициента равен нулю ($\boldsymbol{\pi}$ не выбирает такого действия никогда), и коррекция скажет, что вся информация, начиная с этого шага, полностью неактуальна. Эта проблема неизбежна.

Вторая проблема — ***взрывающийся след*** (exploding trace): $\boldsymbol{\mu}(\boldsymbol{a}^{(t)} \mid \boldsymbol{s}^{(t)}) \ll \boldsymbol{\pi}(\boldsymbol{a}^{(t)} \mid \boldsymbol{s}^{(t)})$. Такое может случиться, если в роллауте попалось редкое для $\boldsymbol{\mu}$ действие, которое тем не менее часто выполняется оцениваемой стратегией $\boldsymbol{\pi}$. С одной стороны, кажется, что как раз такие роллауты наиболее ценны для обучения $\boldsymbol{Q^{\pi}}$, ведь в них описывается шаг взаимодействия со средой, который для $\boldsymbol{\pi}$ как раз достаточно вероятен. Но на практике взрывающийся importance sampling коэффициент — источник большой дисперсии рассматриваемой оценки.

Название	**Коффициенты $\boldsymbol{c_i}$**	**Проблема**
TD($\boldsymbol{\lambda}$)	$\boldsymbol{\lambda}$	только on-policy режим
Одношаговые	0	сильное смещение
Importance Sampling	$\boldsymbol{\lambda} \frac{\boldsymbol{\pi}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}$	легко взрываются

Идея борьбы со взрывающимся коэффициентом, предложенной в оценке Retrace($\boldsymbol{\lambda}$), заключается в следующем: если importance sampling коррекция для $\boldsymbol{t}$-го шага взорвалась, давайте воспользуемся тем, что мы теоретически обоснованно можем выбрать любой коэффициент меньше («быстрее» потушить след), и выберем единицу:

$$\boldsymbol{c_i} \coloneqq \boldsymbol{\lambda} \min\left(\mathbf{1}, \frac{\boldsymbol{\pi}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}\right) \tag{3.58}$$

Коррекция с такими коэффициентами корректна, стабильна, но, конечно, никак не помогает с затуханием следа. Важно помнить, что если $\boldsymbol{\pi}$ и $\boldsymbol{\mu}$ сильно отличаются, то велика вероятность, что коэффициенты затухания будут очень близки к нулю, и мы получим что-то, парктически всегда похожее на одношаговое обновление. С этим мы фундаментально не можем ничего сделать, хотя из-за этого итоговая формула обновления получается сильно смещённой. По этой причине смысла выбирать $\boldsymbol{\lambda} < \mathbf{1}$ в off-policy режиме обычно мало, и его почти всегда полагают равным единицей.

Также обсудим, что говорит формула Retrace в ситуации, когда оцениваемая политика $\boldsymbol{\pi}$ детерминирована. Если на шаге $\boldsymbol{t}$ политика сбора данных $\boldsymbol{\mu}$ выбрала ровно то действие, которое выбирает политика $\boldsymbol{\pi}$, то коэффициент $\boldsymbol{c_i} = \boldsymbol{\lambda}$, то есть ситуация совпадает с on-policy режимом (действительно, в дискретных пространствах действий в числителе единица, а в знаменателе что-то меньшее единицы, когда в непрерывных пространствах действий числитель технически равен бесконечности, поэтому мы обрежем дробь до единицы). В противном же случае дробь $\frac{\boldsymbol{\pi}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}{\boldsymbol{\mu}(\boldsymbol{a}^{(i)} \mid \boldsymbol{s}^{(i)})}$ имеет ноль в числителе, и след занулится. Таким образом, пока в буфере в цепочке $\boldsymbol{s'}, \boldsymbol{a'}, \boldsymbol{s''}, \boldsymbol{a''}, \cdots$ записанные действия совпадают с теми, которые выбирает детерминированная $\boldsymbol{\pi}$, мы «пользуемся TD($\boldsymbol{\lambda}$)», но вынуждены оборвать след, как только очередное действие разойдётся. Есть некоторая польза в том, чтобы в алгоритме $\boldsymbol{\pi}$ и $\boldsymbol{\mu}$ были стохастичны: тогда след по крайней мере полностью никогда не затухнет.

Мы дальше в off-policy будем обсуждать в основном одношаговые оценки, в том числе для простоты. Из одношаговости будут вытекать все ключевые недостатки таких алгоритмов, связанные со смещённостью подобных оценок и невозможностью полноценно разрешать bias-variance trade-off. Во всех этих алгоритмах с этой проблемой можно будет частично побороться при помощи идей Retrace($\boldsymbol{\lambda}$).