ПРИБЛИЖЕННЫЕ МЕТОДЫ В RL

Сергей Николенко

СПбГУ — Санкт-Петербург

13 мая 2022 г.

Random facts:

- Пятница 13-е стала несчастливым днём в 1307 году: в пятницу, 13 октября Франции Филипп Красивый отдал приказ схватить всех тамплиеров, многие из которых были позже сожжены
- 13 мая 1829 г. Александр Пушкин через Фёдора Толстого-Американца попросил руки Натальи Гончаровой; ответ получил неопределённый и в тот же день отправился на Кавказ, в действующую армию
- 13 мая 1857 г. многие парижане в панике покинули город из-за стремительно распространившегося слуха о том, что в городе вот-вот упадут высокие дома
- 13 мая 1888 г. в Бразилии вышел «Золотой закон» («Золотая булла», Lei Áurea), который окончательно отменил рабство в Бразилии; с тех пор 13 мая, или «День мулата» (Dia do Mulato), — памятный день бразильского календаря
- 13 мая 1895 г. в США сгорела лаборатория Николы Теслы; погибли лабораторные журналы с записями результатов исследований в области радио, рентгеновского излучения, беспроволочной передачи звука, передачи энергии и создания самодвижущихся экипажей, которые Тесла не успел запатентовать

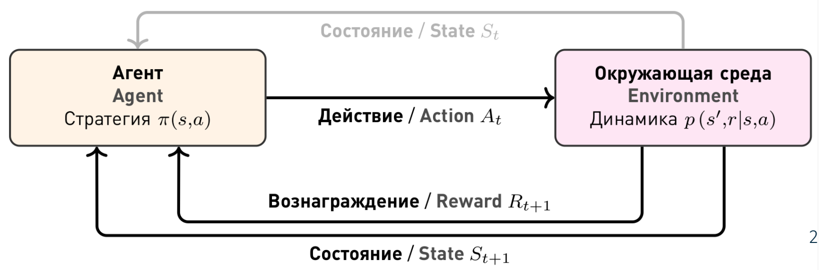
Обучение с подкреплением

- В прошлый раз мы ввели основные понятия динамики марковских процессов принятия решений:
  - собственно динамику процесса:

$$p\left(s',r \mid s,a\right) = p\left(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a \,|;\right)$$

  - награды за каждый эпизод, начиная со времени t:

$$G_t = R_{t+1} + \gamma G_{t+1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1};$$



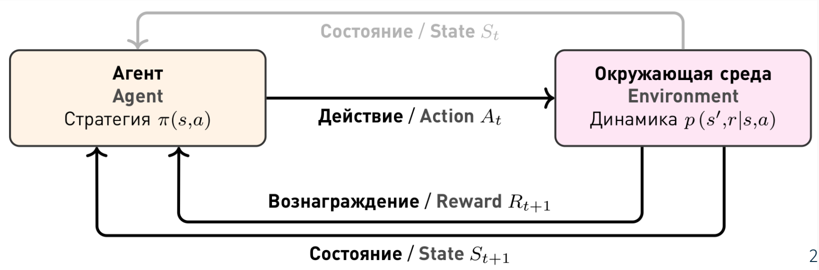
Обучение с подкреплением

- Определили функции значений V и Q

$$V_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right],$$

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

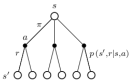
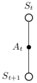
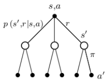
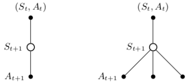
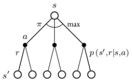
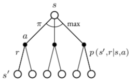
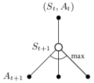
- Выписали уравнения Беллмана и научились их решать.



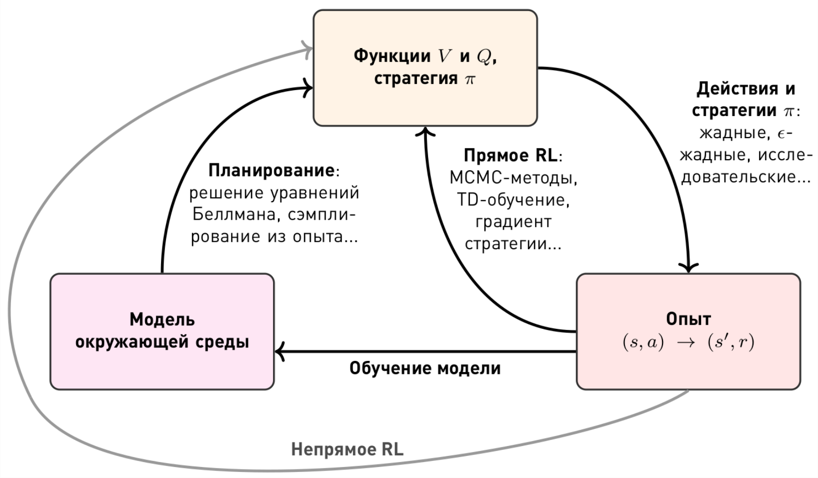
Основные задачи

- Теоретически всё готово, но у нас много проблем:
    - уравнения знаем, но пока не знаем, как их решать, то есть как найти V^π для данного π?
    - разных стратегий очень, очень много — как найти оптимальную стратегию поведения агента в данной модели и соответствующие V^*?
    - но уравнений тоже не знаем — в реальности обычно P и R не даны, их тоже нужно обучить; как?
    - более того, их обычно даже записать не получится, слишком уж много состояний в любой реальной задаче... что делать?

- Давайте есть слона по частям...

	Функция	Обновление в ожидании	Обновление по выборке
Оценка стратегии π	$V_\pi(S_t) :=$	**Уравнение Беллмана** $\sum_{a\in\mathcal{A}} \pi(a\vert S_t) \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\vert s,a)\,(r+\gamma V_\pi(s'))$ 	**TD**(0) $V(S_t)+\alpha\,(R_{t+1}+\gamma V(S_{t+1})-V(S_t))$ 
	$Q_\pi(S_t,A_t) :=$	**Уравнение Беллмана** $\sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\vert S_t,A_t)\,(r+\gamma \sum_{a'\in\mathcal{A}} \pi(a'\vert s')\,Q_\pi(s',a'))$ 	**Sarsa, Sarsa с ожиданием** $Q(S_t,A_t)+\alpha\,(R_{t+1}+\gamma Q(S_{t+1},A_{t+1})-Q(S_t,A_t))$ $Q(S_t,A_t)+\alpha\left(R_{t+1}+\gamma \sum_a \pi(a\vert S_{t+1})\,Q(S_{t+1},a)-Q(S_t,A_t)\right)$ 
Управление, поиск π_*	$V_*(S_t) :=$	**Уравнение Беллмана** $\max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\vert S_t,a)\,(r+\gamma V_*(s'))$ 	
	$Q_*(S_t,A_t) :=$	**Уравнение Беллмана** $\sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\vert S_t,A_t)\left(r+\gamma \max_{a'} Q_*(s',a')\right)$ 	**Q-обучение** $Q(S_t,A_t)+\alpha\,(R_{t+1}+\gamma \max_a Q(S_{t+1},a)-Q(S_t,A_t))$ 

Обучение с подкреплением



Приближённые методы в RL

Основные задачи

- Вспомним ключевые проблемы:
  - уравнения знаем, но пока не знаем, как их решать...
  - разных стратегий очень много — как найти оптимальную...
  - но уравнений тоже не знаем — обычно P и R не даны...
  - более того, их обычно даже записать не получится, слишком уж много состояний в любой реальной задаче... что делать?

- Вроде всё решили, кроме последней проблемы. То есть мы можем посетить небольшую часть (s, a), и надо как-то обобщить эту информацию, построить функцию, которая продолжала бы имеющуюся информацию о V или Q на другие пары (s, a).
- Как же это сделать?..

- …ну конечно, всё машинное обучение этому посвящено!
- Давайте заведём какую-нибудь модель $\hat{V}(s, \mathbf{w})$, которая будет приближать $V(s)$.
- Качество приближения оценим, например, как

$$\widetilde{VE}(\mathbf{w}) = \sum_{s \in S} \mu(s) \left(V(s) - \hat{V}(s, \mathbf{w})\right)^2,$$

где $\mu(s)$ – веса (например равномерные, или отражающие долю времени, проведённого нами в том или ином состоянии).

- Теперь можно по этой модели строить, например, SGD:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left(V(S_t) - \hat{V}(S_t, \mathbf{w}_t) \right) \nabla_{\mathbf{w}} \hat{V}(S_t, \mathbf{w}_t).$$

- Мы, конечно, не знаем $V(S_t)$, так что вместо него подставим
  - либо текущий сэмпл G_t, получая Gradient MC:

$$\mathbf{w} := \mathbf{w} + \alpha \left(G_t - \hat{V}(S_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{V}(S_t, \mathbf{w});$$

  - либо TD-оценку, получая Semi-gradient TD(0):

$$\mathbf{w} := \mathbf{w} + \alpha \left(R_{t+1} + \gamma \hat{V}(S_{t+1}, \mathbf{w}) - \hat{V}(S_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{V}(S_t, \mathbf{w}).$$

- То же самое можно сделать и с $Q(s, a)$.

- И управление тоже можно сделать:
  - episodic semi-gradient Sarsa:

$$\mathbf{w} := \mathbf{w} + \alpha \left(R_{t+1} + \gamma \hat{Q}(S_{t+1}, A_{t+1}, \mathbf{w}) - \hat{Q}(S_t, A_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(S_t, A_t, \mathbf{w});$$

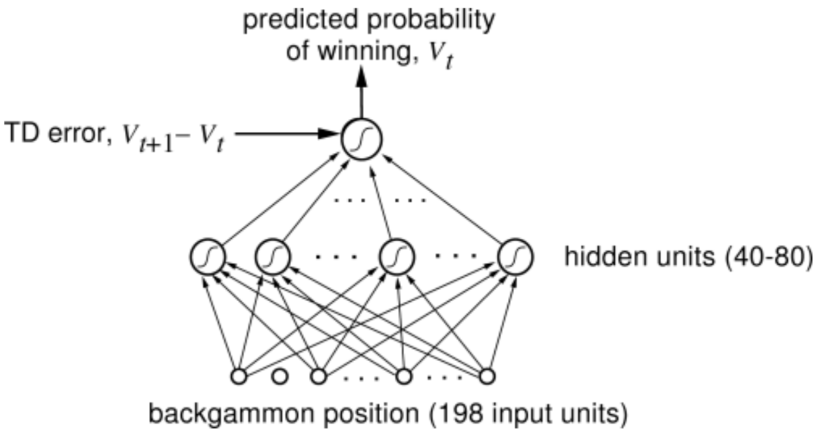
  - semi-gradient off-policy TD(0):

$$\mathbf{w} := \mathbf{w} + \alpha \frac{\pi\left(A_t | S_t\right)}{b(A_t \mid S_t)} \left(R_{t+1} + \gamma \hat{V}(S_{t+1}, \mathbf{w}) - \hat{V}(S_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{V}(S_t, \mathbf{w}).$$

- Весь Deep RL по сути является одним из этих методов, просто в качестве $\hat{Q}$ и $\hat{V}$ мы берём глубокие нейронные сети.

Градиентные методы в RL

- Первый супер-успешный пример – TD-Gammon, хотя это ещё Shallow RL :)

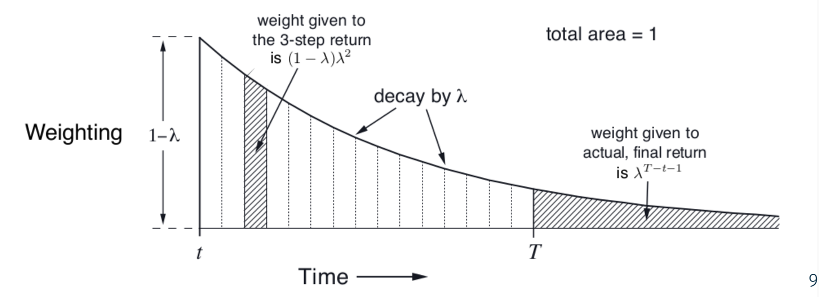


- К Deep RL мы и перейдём.

- И ещё одно расширение, которое тоже заполняет пространство между методами Монте-Карло и TD-обучением: мы называли базовый алгоритм TD(0), но что там может быть вместо нуля?
- Eligibility trace:
  - давайте введём вектор $\mathbf{z}_t \in \mathbb{R}^d$, который соответствует тому, насколько недавно в результате участвовали компоненты $\mathbf{w}_t \in \mathbb{R}^d$;
  - тогда мы будем обучать этот компонент $\mathbf{w}_t$, если z ещё не обнулился, когда мы получили ненулевую TD-ошибку.
- Параметр λ в $TD(\lambda)$ отвечает как раз за затухание этого следа.

- Если формально:
  - мы говорили только что о $G_{t:t+n}$; но можно и усреднить несколько, например $\frac{1}{2}G_{t:t+1} + \frac{1}{4}G_{t:t+2} + \frac{1}{4}G_{t:t+3}$;
  - давайте определим экспоненциально затухающее среднее $G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}$;
  - тогда можно все алгоритмы так же определить, но с целью G_t^λ:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\left(G_t^\lambda - \hat{V}(S_t, \mathbf{w}_t)\right)\nabla_{\mathbf{w}}\hat{V}(S_t, \mathbf{w}_t).$$

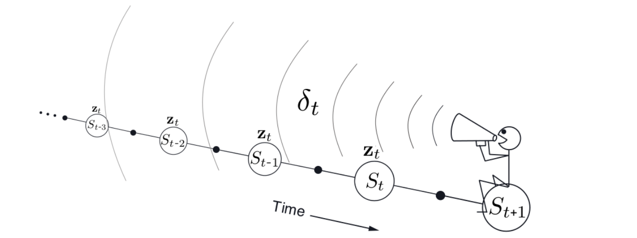


- Например, TD-обучение для оценки V_π.

Алгоритм Semi-gradient $TD(\lambda)$:

- инициализировать $\hat{V}(s, \mathbf{w})$, веса $\mathbf{w}$;
- повторять по эпизодам:
  - инициализировать S, $\mathbf{z} := 0$;
  - для каждого шага в эпизоде $t = 0, \dots, T$:
    - выбрать A по стратегии π, сделать действие A, получить R, S';
    - $\mathbf{z} := \gamma\lambda\mathbf{z} + \nabla_{\mathbf{w}}\hat{V}(S, \mathbf{w})$;
    - $\mathbf{w} := \mathbf{w} + \alpha\left(R + \gamma\hat{V}(S', \mathbf{w}) - \hat{V}(S, \mathbf{w})\right)\mathbf{z}$; и переходим $S := S'$.

- Обновляем по eligibility traces в прошлом:



Eligibility traces

- TD(λ) обобщает то, что было раньше:
  - для $\lambda = 0$ получим обычный value gradient $\mathbf{z} = \nabla_{\mathbf{w}} \hat{V}(S, \mathbf{w})$;
  - для $\lambda = 1$ более ранние состояния умножаются только на γ, и получается в точности метод Монте-Карло;
  - кстати, TD(1) – это более удобный и правильный метод реализовывать Монте-Карло, чем те, что было раньше.
- Есть ещё разные варианты, например dutch traces, но про них давайте не будем, см. (Sutton, Barto)...

Policy gradient

Policy gradient

- Мы до сих пор говорили про action-value методы: берём марковский процесс принятия решений и ищем V и/или Q.
- Но это не единственный подход!
- Что если мы будем оптимизировать стратегии напрямую?

$$\pi\left(a|s,\theta\right) = p\left(A_t = a \mid S_t = s, \theta_t = \theta\right)$$

- Может быть, есть ещё и $\hat{V}(s, \mathbf{w})$, это отдельно.

- Тогда можно попробовать сформулировать цель $J(\theta)$ и просто двигаться как

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta).$$

- Такие методы называются policy gradient методами.
- А если ещё и $\hat{V}(s, \mathbf{w})$, то actor-critic методами: критик – это $\hat{V}$, которая «критикует» стратегию π.

- Если действий не слишком много, можно искать $h(s, a, \theta)$ (хоть линейно, хоть нейросетью) и строить

$$\pi\left(a|s, \theta\right) = \mathrm{softmax}\left(h\right).$$

- Преимущества:
  - можно сходиться к детерминированной стратегии без явной стратегии уменьшения ϵ;
  - а можно строить стохастическую стратегию, которую через $Q(s, a)$ вообще непонятно как найти;
  - оптимизировать π может быть проще, чем Q.
- Но как же это сделать?

- Policy gradient theorem:

$$\nabla V_\pi(s) = \nabla \left(\sum_a \pi\left(a|s\right) Q_\pi(s,a) \right) =$$

$$= \ldots =$$

$$= \sum_{x \in S} \sum_{k=0}^{\infty} p(\text{перейти из } s \text{ в } x \text{ за } k \text{ шагов по } \pi) \sum_a \nabla \pi\left(a|x\right) Q_\pi(x,a),$$

то есть

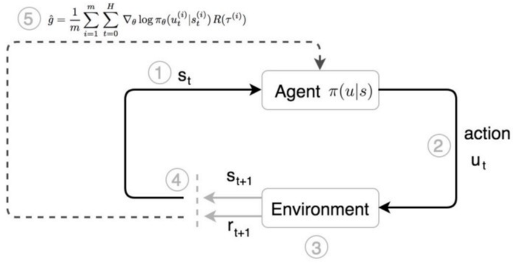
$$\nabla J(\theta) \propto \nabla V_\pi(s) = \sum_s \mu(s) \sum_a \nabla \pi\left(a|s\right) Q_\pi(s,a),$$

где $\mu(s)$ – ожидаемая доля времени, проведённого в состоянии s.

- Иначе говоря,

$$\nabla J(\theta) \propto \mathbb{E}_\pi \left[\sum_a \nabla \pi \left(a | s \right) Q_\pi(s, a) \right],$$

и теперь можно и алгоритмы построить.



- Можно all-actions алгоритм:

$$\theta_{t+1} = \theta_t + \alpha \sum_a \hat{Q}(S_t, a, \mathbf{w}) \nabla_\theta \pi \left(a | S_t, \theta \right).$$

- Но лучше уж сразу REINFORCE (Williams, 1992):

$$\nabla J(\theta) \propto \mathbb{E}_\pi \left[\sum_a \nabla \pi \left(A_t | S_t, \theta \right) Q_\pi(S_t, a) \right] =$$
$$= \mathbb{E}_\pi \left[Q_\pi(S_t, A_t) \frac{\nabla \pi \left(A_t | S_t, \theta \right)}{\pi \left(A_t | S_t, \theta \right)} \right] = \mathbb{E}_\pi \left[G_t \frac{\nabla \pi \left(A_t | S_t, \theta \right)}{\pi \left(A_t | S_t, \theta \right)} \right],$$

  потому что $Q_\pi(S_t, A_t) = \mathbb{E}_\pi \left[G_t \mid S_t, A_t \right]$.
- И алгоритм такой:

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla \log \pi \left(A_t | S_t, \theta \right).$$

- Т.е. двигаем туда, где максимальный return, всё логично.

- G_t можно оценить по методу Монте-Карло:

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

- Но это ещё не всё. Ещё можно добавить baseline:

$$\nabla J(\theta) \propto \nabla V_\pi(s) = \sum_s \mu(s) \sum_a \left(Q_\pi(s,a) - b(s)\right) \nabla\pi\left(a|s,\theta\right),$$

  и это не повлияет ни на что, мы вычитаем $b(s)\nabla\sum_a \pi\left(a|s,\theta\right) = b(s)\nabla 1 = 0$.
- Это не меняет ожидание, но может сильно изменить дисперсию! В том числе в хорошую сторону.
- Естественный baseline – это оценка состояния $\hat{V}(S_t, \mathbf{w})$.

- Его нужно будет оценить, как мы обычно оцениваем $\hat{V}$, то есть теперь два разных градиента получается:

REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\quad (G_t)$
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

- А чтобы получился полноценный actor-critic, нужно через $\hat{V}$ оценить и результаты действия тоже:

$$\theta_{t+1} = \theta_t + \alpha\left(G_{t:t+1} - \hat{V}(S_t, \mathbf{w})\right)\nabla \log \pi\left(A_t | S_t, \theta\right) =$$
$$= \theta_t + \alpha\left(R_{t+1} + \gamma\hat{V}(S_{t+1}, \mathbf{w}) - \hat{V}(S_t, \mathbf{w})\right)\nabla \log \pi\left(A_t | S_t, \theta\right) =$$
$$= \theta_t + \alpha\delta_t \frac{\nabla\pi\left(A_t | S_t, \theta\right)}{\pi\left(A_t | S_t, \theta\right)}.$$

- Алгоритм получается такой:

One-step Actor–Critic (episodic), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize S (first state of episode)
    $I \leftarrow 1$
    Loop while S is not terminal (for each time step):
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla\hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I\delta\nabla\ln\pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

- Можно с eligibility traces, как мы раньше обсуждали:

Actor–Critic with Eligibility Traces (episodic), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize S (first state of episode)
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
    $I \leftarrow 1$
    Loop while S is not terminal (for each time step):
        $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{v}(S,\mathbf{w})$
        $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla\ln\pi(A|S,\boldsymbol{\theta})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

- А можно попробовать сделать off-policy вариант; для другой стратегии β:

$$\nabla J(\theta) \propto \nabla_\theta \sum_s \mu_\pi(s) \sum_a \pi\left(a|s\right) Q_\pi(s,a) =$$
$$= \sum_s \mu_\pi(s) \sum_a \left[\nabla_\theta \pi\left(a|s\right) Q_\pi(s,a) + \pi\left(a|s\right) \nabla_\theta Q_\pi(s,a)\right],$$

- Теперь первое слагаемое можно оценить как обычно:

$$\sum_s \mu_\pi(s)[...] = \mathbb{E}_\pi\left[...\right] = \mathbb{E}_\beta\left[\frac{\pi\left(a|s\right)}{\beta(a \mid s)}...\right], \text{ то есть}$$

$$\nabla J(\theta) \propto \mathbb{E}_\beta\left[\frac{\pi\left(a|s\right)}{\beta(a \mid s)} Q_\pi(s,a) \nabla_\theta \ln \pi\left(a|s\right)\right].$$

- На первый взгляд кажется, что всё равно всё пропало, потому что $\nabla_\theta Q_\pi(s,a)$ мы никак не оценим.
- Но оказывается (Degris et al., 2012), что его можно просто выбросить, и полученная аппроксимация всё равно неплохая и минимум у неё там же.
- Так что в алгоритме просто добавятся importance weights:

Algorithm 1 The Off-PAC algorithm

Initialize the vectors $\mathbf{e}_v$, $\mathbf{e}_u$, and $\mathbf{w}$ to zero
Initialize the vectors $\mathbf{v}$ and $\mathbf{u}$ arbitrarily
Initialize the state s
For each step:
  Choose an action, a, according to $b(\cdot|s)$
  Observe resultant reward, r, and next state, s'
  $\delta \leftarrow r + \gamma(s')\mathbf{v}^\mathsf{T}\mathbf{x}_{s'} - \mathbf{v}^\mathsf{T}\mathbf{x}_s$
  $\rho \leftarrow \pi_\mathbf{u}(a|s)/b(a|s)$
  Update the critic (GTD(λ) algorithm):
    $\mathbf{e}_v \leftarrow \rho\left(\mathbf{x}_s + \gamma(s)\lambda\mathbf{e}_v\right)$
    $\mathbf{v} \leftarrow \mathbf{v} + \alpha_v\left[\delta\mathbf{e}_v - \gamma(s')(1-\lambda)(\mathbf{w}^\mathsf{T}\mathbf{e}_v)\mathbf{x}_s\right]$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha_w\left[\delta\mathbf{e}_v - (\mathbf{w}^\mathsf{T}\mathbf{x}_s)\mathbf{x}_s\right]$
  Update the actor:
    $\mathbf{e}_u \leftarrow \rho\left[\frac{\nabla_\mathbf{u}\pi_\mathbf{u}(a|s)}{\pi_\mathbf{u}(a|s)} + \gamma(s)\lambda\mathbf{e}_u\right]$
    $\mathbf{u} \leftarrow \mathbf{u} + \alpha_u\delta\mathbf{e}_u$
  $s \leftarrow s'$

Policy gradient

- В policy gradient можно рассмотреть даже, например, непрерывные действия.
- Например, параметризуем действие $a \in \mathbb{R}$ через гауссиан:

$$\pi\left(a|s,\theta\right) = \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} e^{-\frac{1}{2\sigma(s,\theta)}\left(a-\mu(s,\theta)\right)^2}.$$

- Если, например,

$$\mu(s,\theta) = \theta_\mu^\top \mathbf{x}_\mu(s), \qquad \sigma(s,\theta) = e^{\theta_\sigma^\top \mathbf{x}_\sigma(s)},$$

то можно подсчитать градиенты:

$$\nabla \ln \pi\left(a|s,\theta_\mu\right) = \frac{1}{\sigma(s,\theta)^2}\left(a-\mu(s,\theta)\right)\mathbf{x}_\mu(s),$$
$$\nabla \ln \pi\left(a|s,\theta_\sigma\right) = \left(\frac{1}{\sigma(s,\theta)^2}\left(a-\sigma(s,\theta)\right)^2 - 1\right)\mathbf{x}_\sigma(s).$$

Спасибо!

Спасибо за внимание!