

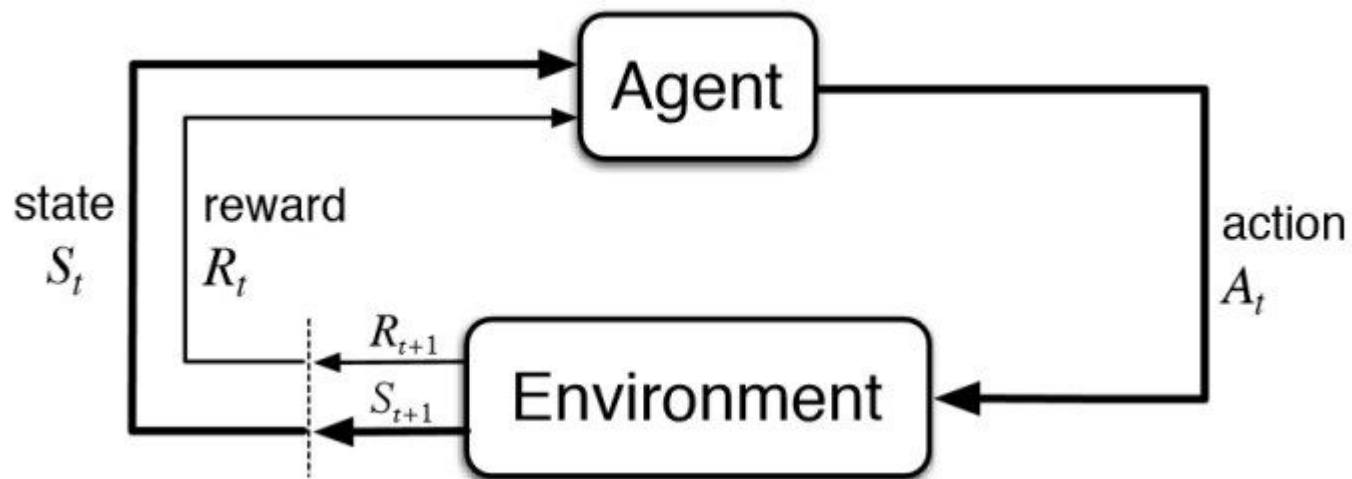
RL 2

возвращение RL'я

Deep q-learning

Артем Мельников

ЧТО ТОЧНО СТОИТ ПОНИМАТЬ



в предыдущей серии:

$$G_t = \sum_{t'=t}^T \gamma^{(t'-t)} r_{t'}$$

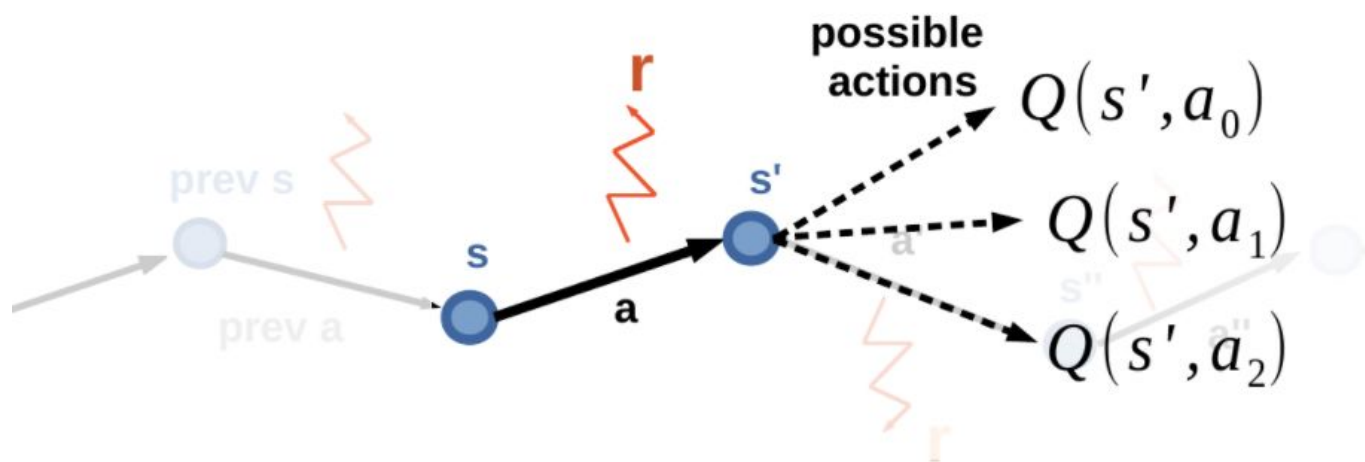
$$Q^{\pi}(s, a) = E_{\pi}[G_t | s_t = s, a_t = a]$$

$$V^{\pi}(s) = E_{\pi}[G_t | s_t = s]$$

$Q^*(s, a)$ — q функция с оптимальной политикой

$V^*(s)$ — v функция с оптимальной политикой

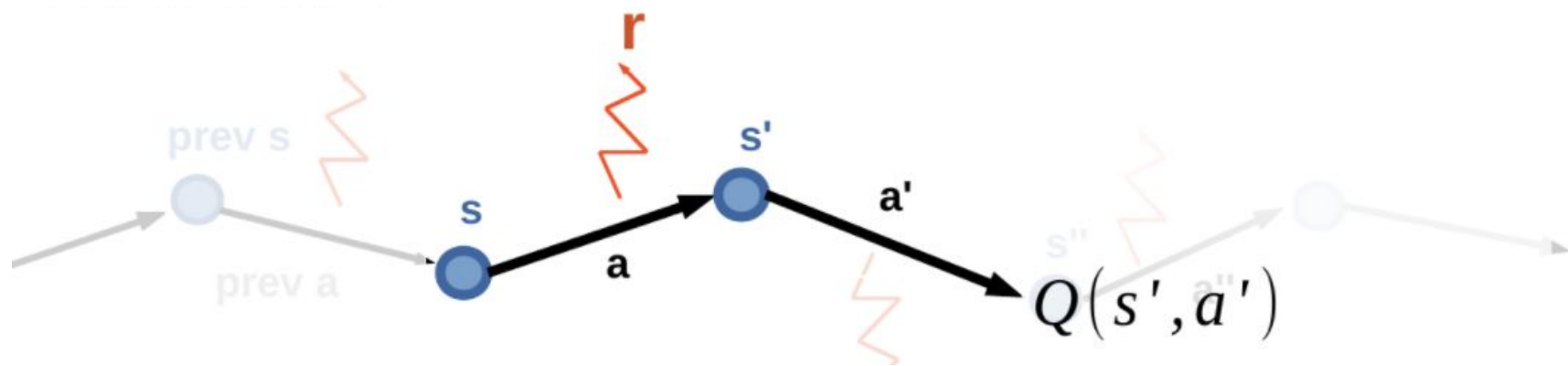
q-learning



$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \underbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}_{\text{new value (temporal difference target)}}$$

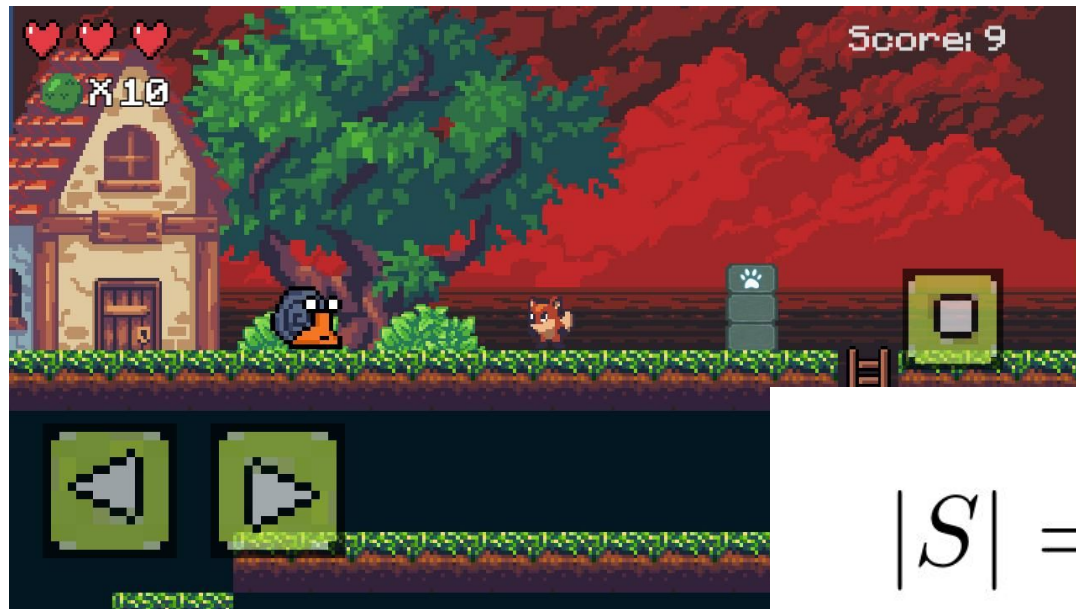
temporal difference

sarsa



- Sample $\langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}', \mathbf{a}' \rangle$ from env
- Compute $\hat{Q}(s, a) = r(s, a) + \gamma Q(s', a')$
- Update $Q(s, a) \leftarrow \alpha \cdot \hat{Q}(s, a) + (1 - \alpha) Q(s, a)$

Однако беды с тем, что делать если среда слишком сложная

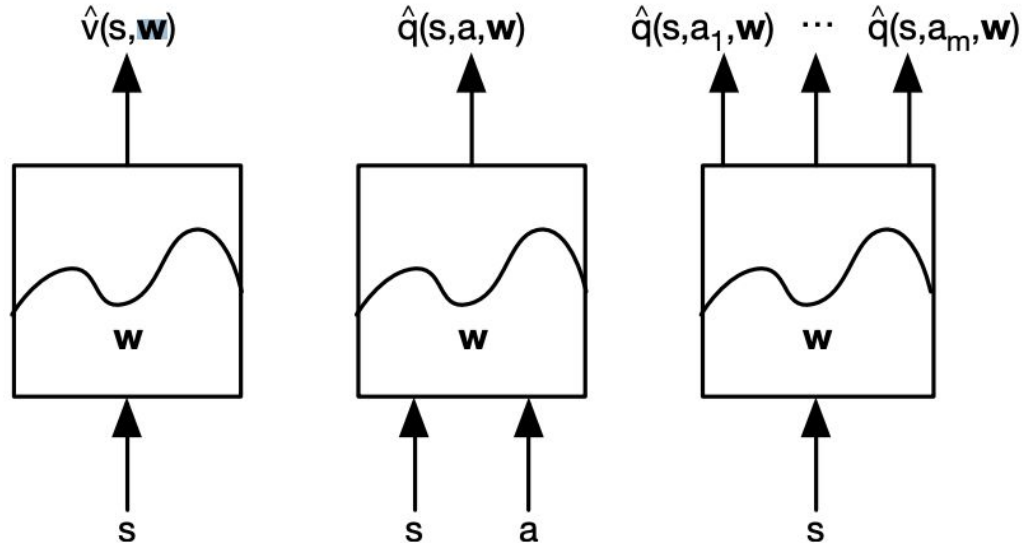


$$|S| = 2^{1280 \cdot 720 \cdot 3 \cdot 255}$$

хочется оценить как-то среду

$$\operatorname{argmin}_{w,b} (Q(s_t, a_t) - [r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a')])^2$$

- По факту - задача регрессии
- Нам подойдет все что угодно, главное - хотим оценить Q



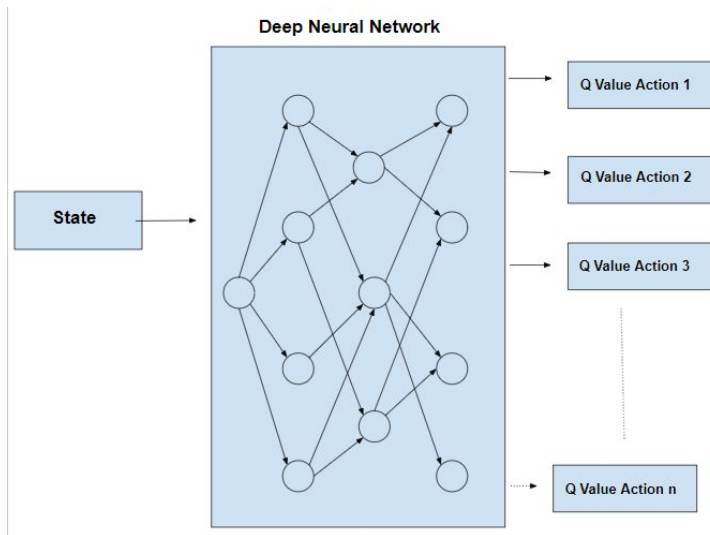
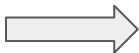
Можем использовать тучу различных методов

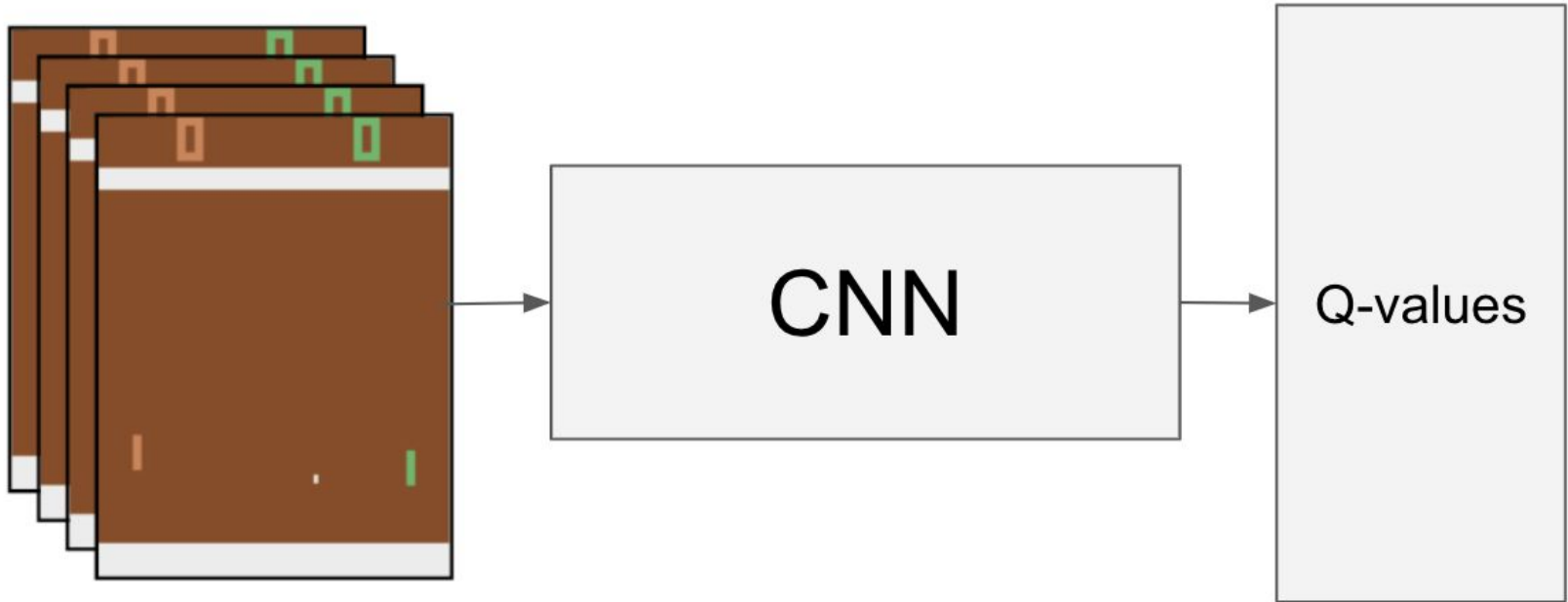
- Линейная модель
- Нейроночки
- Решающие деревья
- Ближайшие соседи
- *вставьте ваш любимый алгос для решения задачи регрессии*

беда - по одному кадру ничего не понять

$$s_t \neq o(s_t)$$

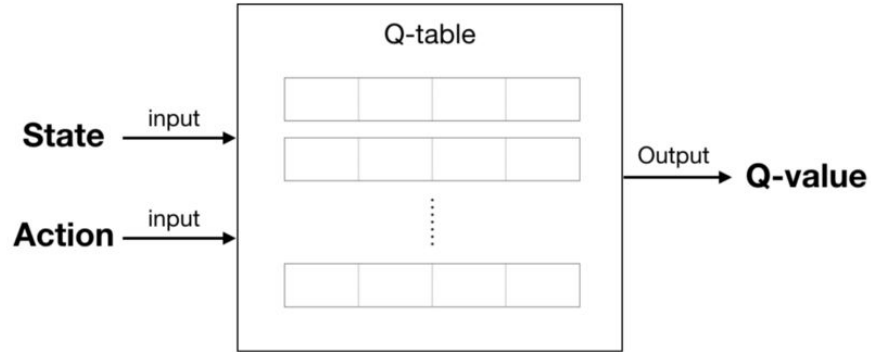
$$s_t \approx (o(s_{t-n}), a_{t-n}, \dots, o(s_{t-1}), a_{t-1}, o(s_t))$$





4 last frames as input

Q-Learning



Q-values:

$$\hat{Q}(s_t, a_t) = r + \gamma \cdot \max_{a'} Q(s_{t+1}, a')$$

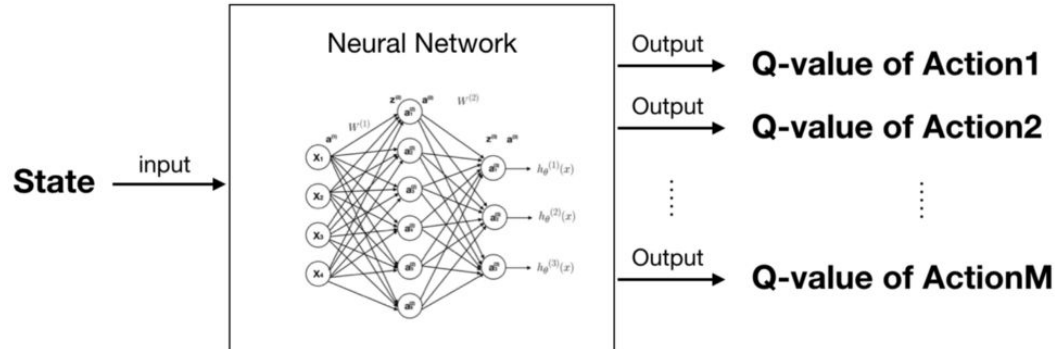
Objective:

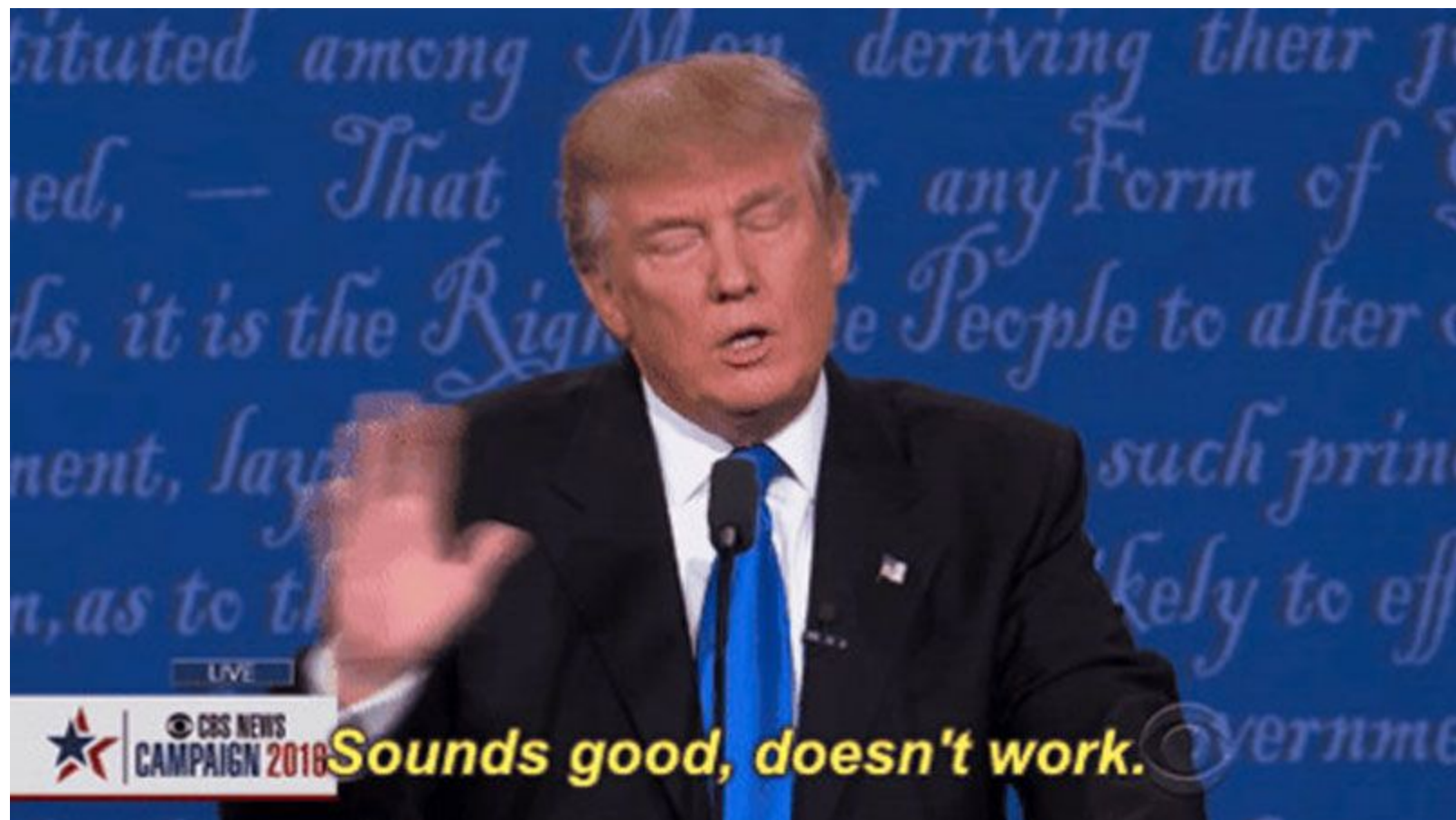
$$L = (Q(s_t, a_t) - [r + \gamma \cdot \max_{a'} \underbrace{Q(s_{t+1}, a')}_{\text{Const}}])^2$$

Gradient step:

$$w_{t+1} = w_t - \alpha \cdot \frac{\delta L}{\delta w}$$

Deep Q-Learning





LIVE



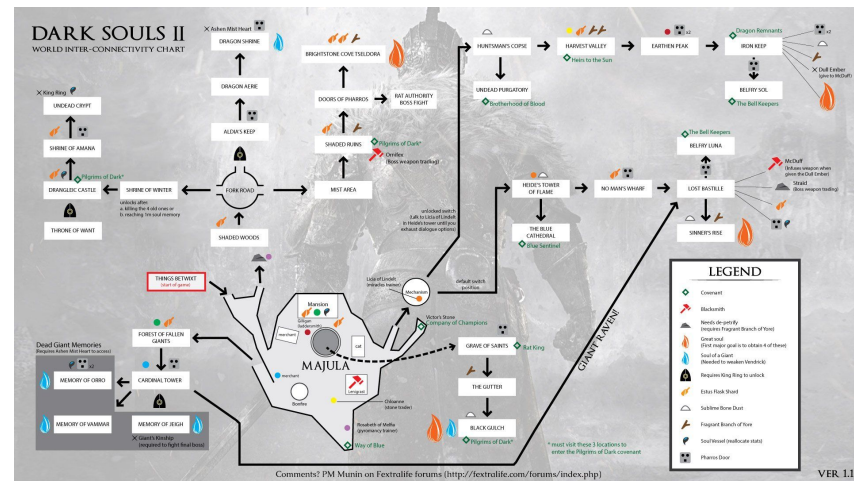
CBS NEWS
CAMPAIGN 2018

Sounds good, doesn't work.

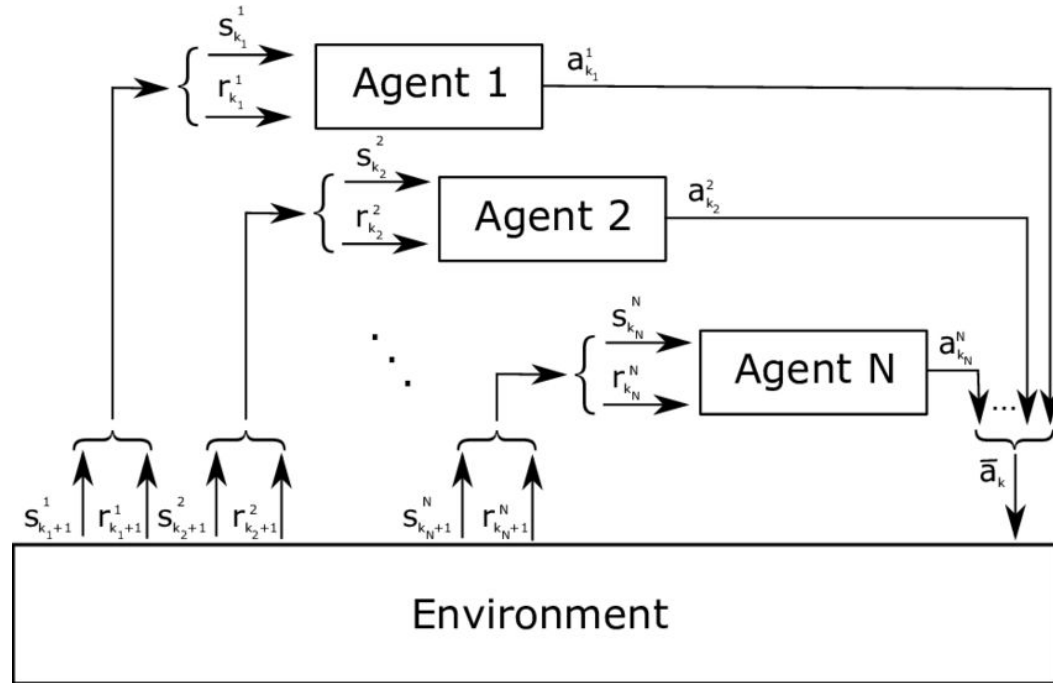


verme

беда - данные всегда упорядочены

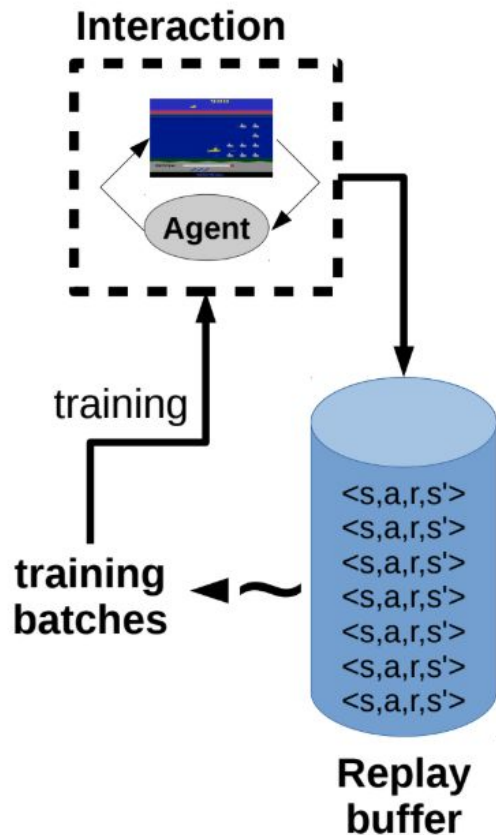
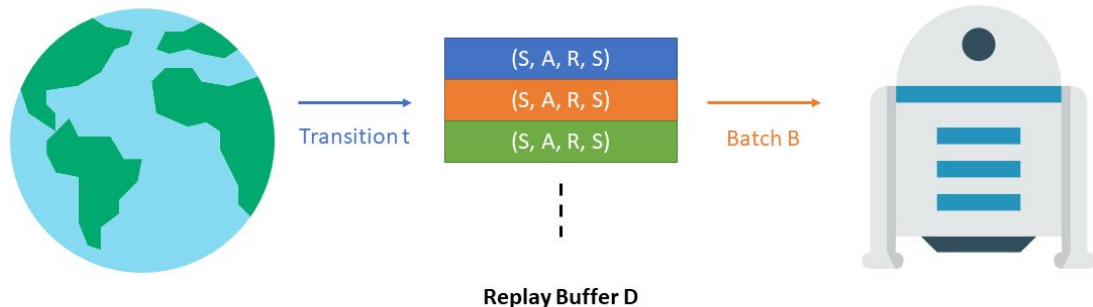


решение 1: запускаем много агентов

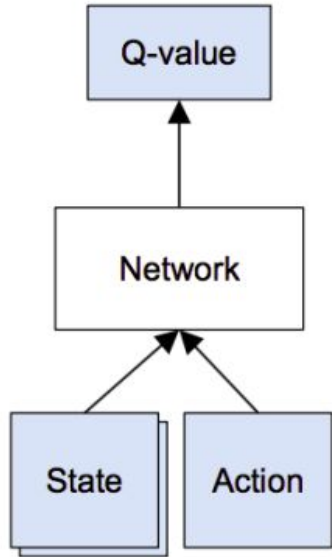


experience replay основная идея

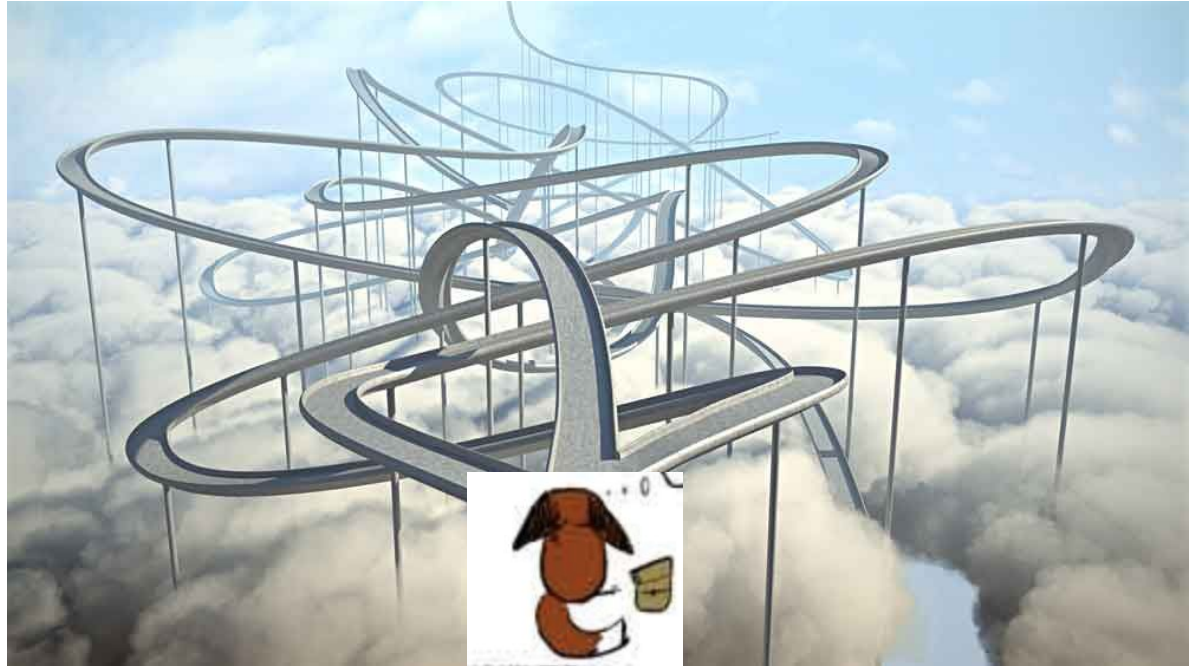
Отделяем процесс обучения от среды, сохраняем n пар состояние/действие/награда и учимся на этом буфере



Еще один способ, про который надо сказать



Given (s,a)
Predict $Q(s,a)$



Список источников

Лекции ШАДа: https://github.com/yandexdataschool/Practical_RL

Цикл статей RL и его модификации:

<https://lilianweng.github.io/lil-log/tag/reinforcement-learning>

Лекции Stanford:

<https://www.youtube.com/playlist?list=PLoROMvodv4rOSOPzutgyCTapiGIY2Nd8u>

Разбор примера OpenAI:

<https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/>

Лекция на тему оценки функций препода из DeepMind:

<https://www.davidsilver.uk/wp-content/uploads/2020/03/FA.pdf>

Deep q-learning статья (про нейронки):

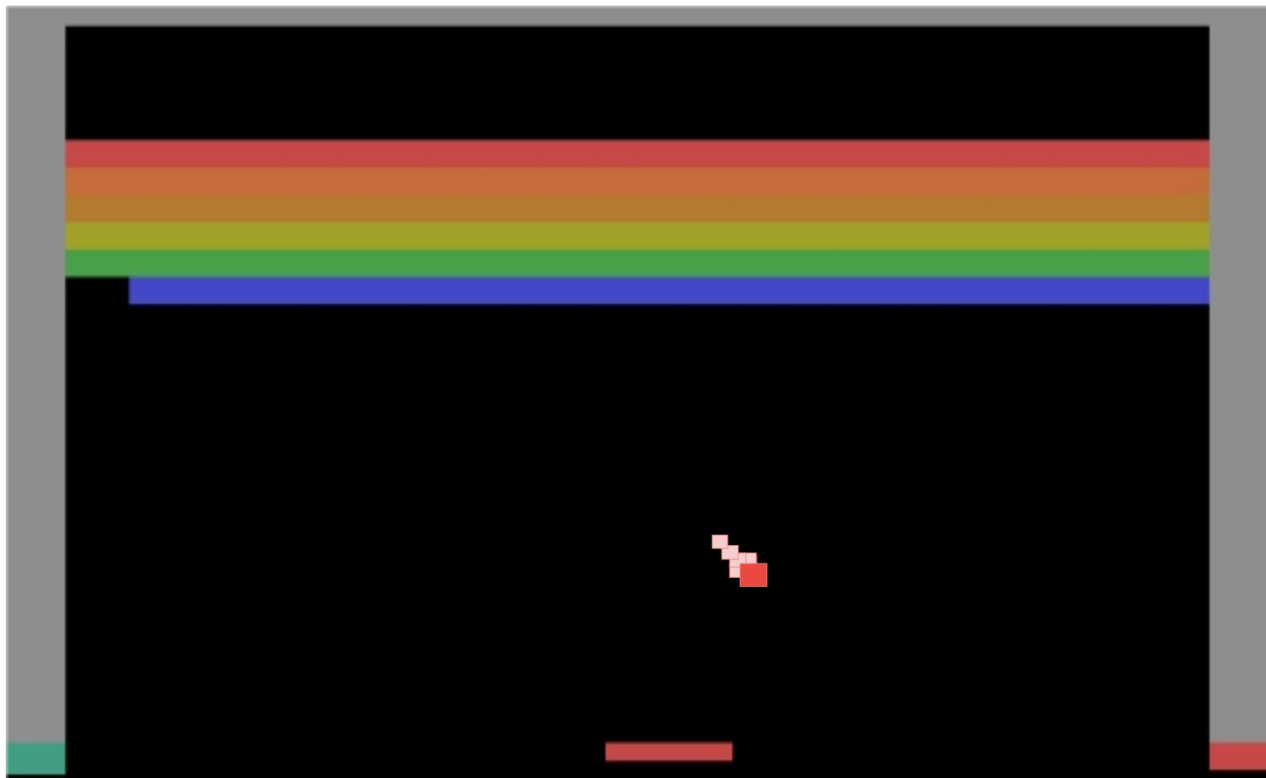
<https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python>

DQN pyTorch: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

Policy gradient

Поляков Дмитрий

Влево или вправо?

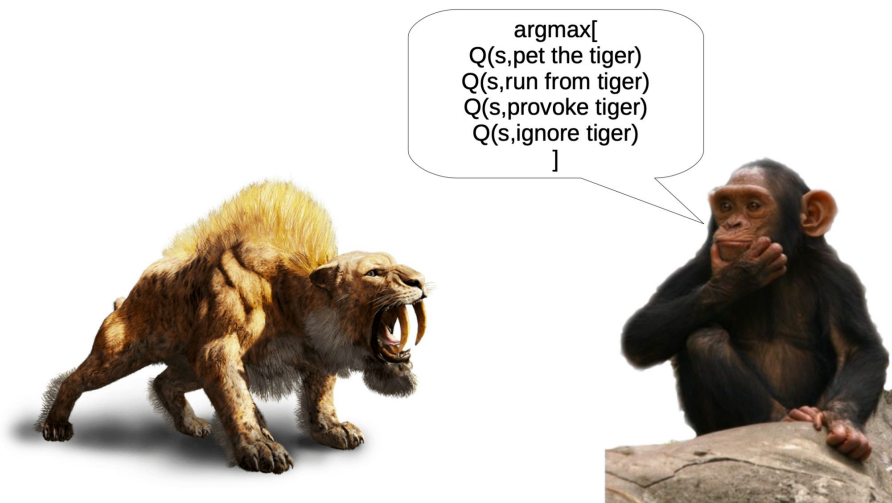


$Q(s, a) - ?$

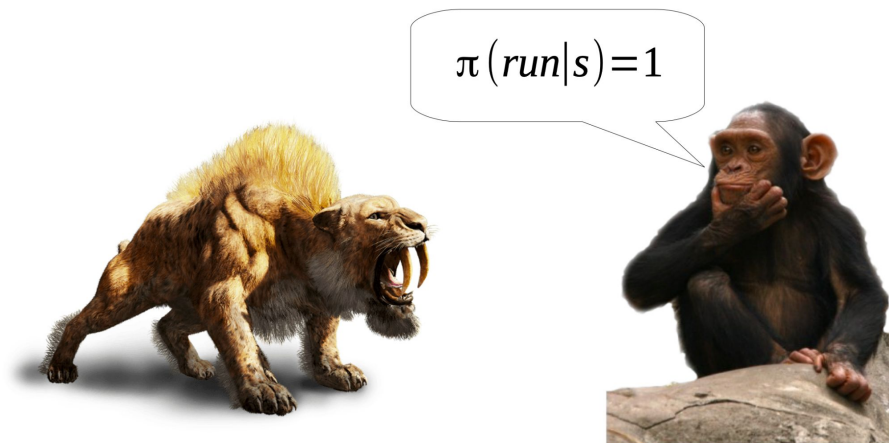


Поучительный мем

NOT how humans survived



how humans survived



Проблема DQN

DQN is trained to minimize

$$L \approx E[Q(s_t, a_t) - (r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a'))]^2$$

Simple 2-state world

	True	(A)	(B)
Q(s0,a0)	1	1	2
Q(s0,a1)	2	2	1
Q(s1,a0)	3	3	3
Q(s1,a1)	100	50	100

Q-learning will prefer worse policy (B)!

better
policy

less
MSE

Выводы

- Подсчет q -функции часто может быть сложнее выбора действия
- Мы можем не обременять себя подсчетом q -функции напрямую обучая политику агента

Главная идея Policy gradient

Давайте будем максимизировать
награду напрямую!

Что хотим максимизировать

Ожидаемую награду

$$J = E_{\substack{s \sim p(s) \\ a \sim \pi_{\theta}(a|s) \\ \dots}} R(s, a, s', a', \dots)$$

Ожидаемую дисконтированную награду

$$J = E_{\substack{s \sim p(s) \\ a \sim \pi_{\theta}(a|s)}} G(s, a)$$

Пример на бандите

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_{\theta}(a|s)}}{E} R(s, a) = \int_s p(s) \int_a \pi_{\theta}(a|s) R(s, a) da ds$$



**Reward for 1-step
session**

Как оценивать

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_{\theta}(a|s)}}{E} R(s, a) = \int_s p(s) \int_a \pi_{\theta}(a|s) R(s, a) da ds$$

$$J \approx \frac{1}{N} \sum_{i=0}^N R(s, a)$$



sample N sessions
under current policy

Как оптимизировать?

$$J = E_{\substack{s \sim p(s) \\ a \sim \pi_{\theta}(a|s)}} R(s, a) = \int_s p(s) \int_a \pi_{\theta}(a|s) R(s, a) da ds$$

↑
Параметры здесь

$$dJ/d\theta - ?$$

Что хотим от оптимизации

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_{\theta}(a|s)}}{E} R(s, a) = \int_s p(s) \int_a \pi_{\theta}(a|s) R(s, a) da ds$$

Хотим:

- Аналитический градиент
- Легкие/устойчивые приближения

Что хотим от оптимизации

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) R(s, a) da ds$$

Такое считать не удобно

Логарифм спешит на помощь


Немного математики

$$\nabla \log \pi(z) = \frac{1}{\pi(z)} \cdot \nabla \pi(z)$$

$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$

Логарифм спешит на помощь

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) R(s, a) da ds$$

$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$


$$\nabla J = \int_s p(s) \int_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) R(s, a) da ds$$

that's expectation :)

REINFORCE (bandit)

- Инициализируем веса нейросети $\theta_0 \leftarrow random$
- Повторяем:
 - Сэмплируем N эпизодов по текущей политике
 - Вычисляем градиент $\nabla J \approx \frac{1}{N} \sum_{i=0}^N \nabla \log \pi_{\theta}(a|s) \cdot R(s, a)$
 - Обновляем веса

$$\theta_{i+1} \leftarrow \theta_i + \alpha \cdot \nabla J$$

Немного черной магии

$$\nabla J = \int_s p(s) \int_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) Q(s, a) da ds$$

Заменяем R на Q

$$\nabla J = E_{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}} \nabla \log \pi_\theta(a|s) \cdot Q(s, a)$$

Градиент

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s, a \in \mathcal{Z}_i} \nabla \log \pi_\theta(a|s) \cdot Q(s, a)$$

Приближенное вычисление

REINFORCE (discounted)

- Инициализируем веса нейросети $\theta_0 \leftarrow random$

- Повторяем:

- Сэмплируем N эпизодов по текущей политике

- Вычисляем градиент
$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} \nabla \log \pi_{\theta}(a|s) \cdot Q(s,a)$$

- Обновляем веса

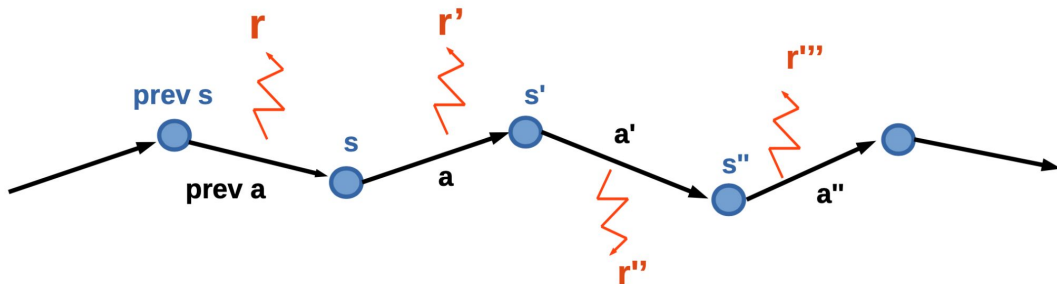
$$\theta_{i+1} \leftarrow \theta_i + \alpha \cdot \nabla J$$

Откуда взять $Q(s, a)$

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s, a \in \mathcal{Z}_i} \nabla \log \pi_{\theta}(a|s) \cdot \underline{Q(s, a)}$$

$$Q_{\pi}(s_t, a_t) = E_{s'} G(s_t, a_t)$$

$$G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \dots$$



Чего б еще пожелать

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s, a \in z_i} \nabla \log \pi_{\theta}(a|s) \cdot Q(s, a)$$

Для разных состояний s значения $Q(s, a)$ могут значительно отличаться. Хочется, чтобы обучение шло более “равномерно”.

Идея baseline

$$\nabla J = \mathop{E}_{\substack{s \sim p(s) \\ a \sim \pi_\theta(a|s)}} \nabla \log \pi_\theta(a|s) (Q(s, a) - b(s)) = \dots$$

$$\dots = \mathop{E}_{\substack{s \sim p(s) \\ a \sim \pi_\theta(a|s)}} \nabla \log \pi_\theta(a|s) Q(s, a) - \mathop{E}_{\substack{s \sim p(s) \\ a \sim \pi_\theta(a|s)}} \nabla \log \pi_\theta(a|s) b(s) = \dots$$

$$\mathop{E}_{\substack{s \sim p(s) \\ a \sim \pi_\theta(a|s)}} \nabla \log \pi_\theta(a|s) b(s) = b(s) \cdot \mathop{E}_{\substack{s \sim p(s) \\ a \sim \pi_\theta(a|s)}} \nabla \log \pi_\theta(a|s) = 0$$

Чем хорош baseline

- Сохраняет градиент
- Не меняет приоритетность действий
- Может снижать дисперсию

$$\nabla J = E_{\substack{s \sim p(s) \\ a \sim \pi_{\theta}(a|s)}} \nabla \log \pi_{\theta}(a|s) (Q(s, a) - b(s)) = \dots$$

$$\dots = E_{\substack{s \sim p(s) \\ a \sim \pi_{\theta}(a|s)}} \nabla \log \pi_{\theta}(a|s) Q(s, a)$$

$$\text{Var}[Q(s, a) - b(s)] = \text{Var}[Q(s, a)] - 2 \cdot \text{Cov}[Q(s, a), b(s)] + \text{Var}[b(s)]$$

Примеры хороших baseline-ОВ

- $b(s) = \text{среднее } Q(s, a) \text{ по } a$
- $b(s) = V(s)$

Список источников

Лекции ШАДа: https://github.com/yandexdataschool/Practical_RL

Книга про RL: <http://incompleteideas.net/book/bookdraft2017nov5.pdf>

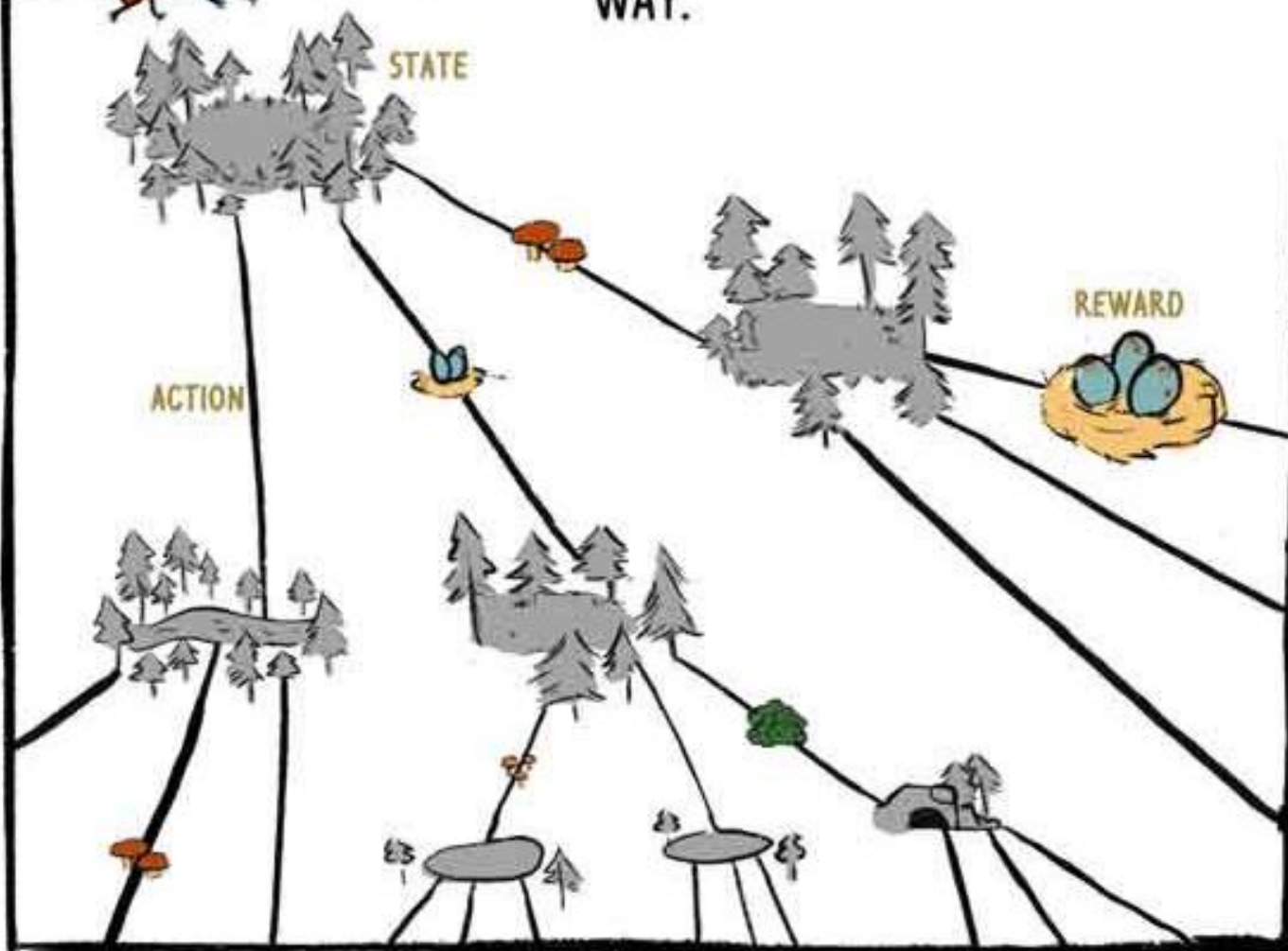
Actor critic

Сапожников Денис

Преимущества и недостатки policy gradient по сравнению с Q-learning

- Преимущества:
 - Легко обобщается на задачи с большим множеством действий, в том числе на задачи с непрерывным множеством действий;
 - По большей части избегает конфликта между эксплуатацией (exploitation) и исследованием (exploration), так как оптимизирует непосредственную стохастическую стратегию $\pi_{\theta}(a|s)$
 - Имеет более сильные гарантии сходимости
- Недостатки:
 - Очень-очень долгий
 - В случае конечных МППР Q-learning сходится к global minimum

IN REINFORCEMENT LEARNING, AN AGENT MOVES THROUGH STATES IN AN ENVIRONMENT BY TAKING ACTIONS, TRYING TO MAXIMIZE REWARDS ALONG THE WAY.



A2CS TAKE IN A STATE—SENSORY INPUTS IN CRANBERRY'S CASE—AND GENERATE **TWO OUTPUTS**:

1) AN ESTIMATE OF HOW MANY REWARDS THEY EXPECT TO GET FROM THAT POINT ONWARDS, THE STATE VALUE.

2) A RECOMMENDATION OF WHAT ACTION TO TAKE, THE POLICY

THE "CRITIC"

WOW, WHAT A WONDERFUL GLEN! THIS WILL BE A FRUITFUL DAY OF FORAGING. I BET I'LL GATHER 20 REWARD POINTS BEFORE SUNSET TODAY.

$$V(\hat{S}) = 20$$

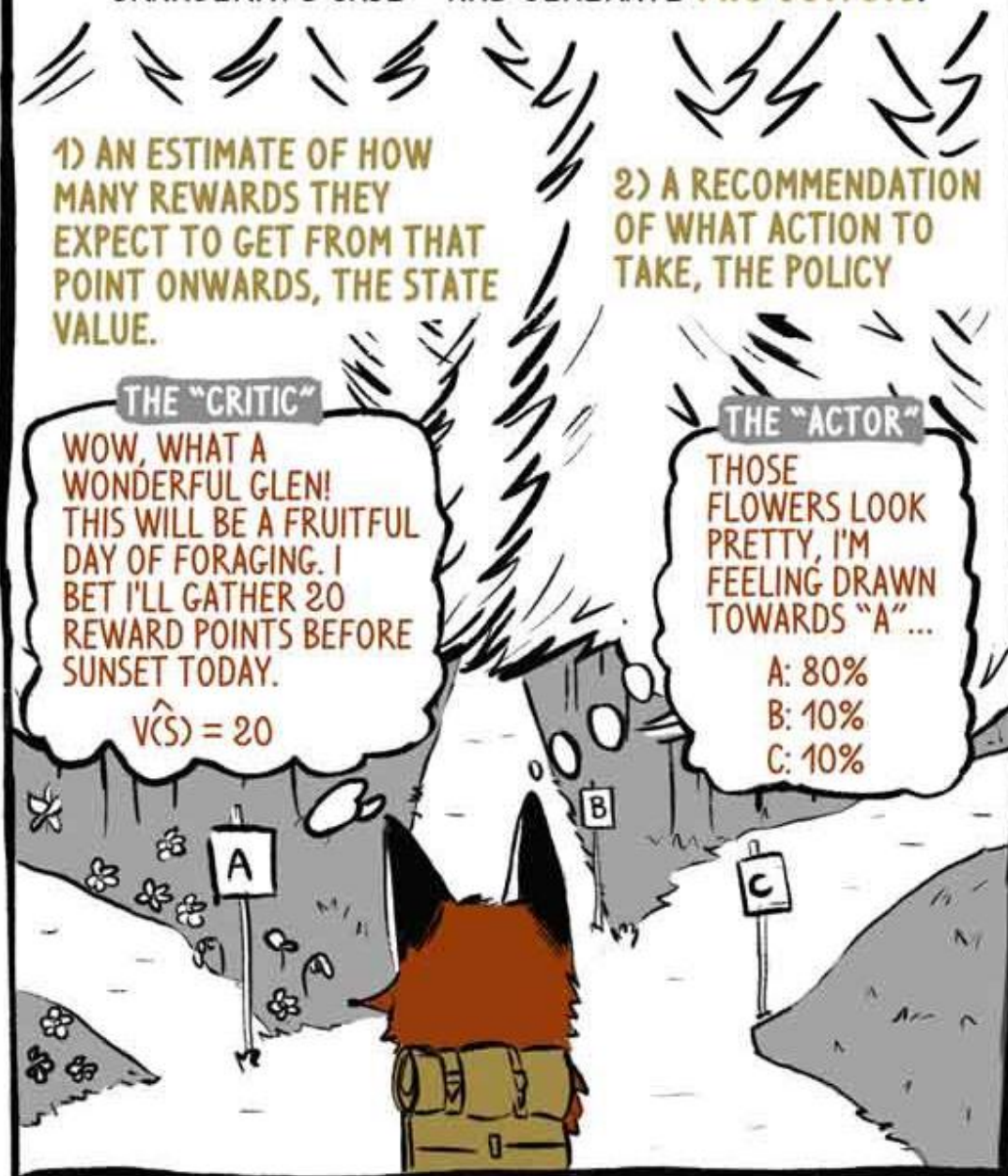
THE "ACTOR"

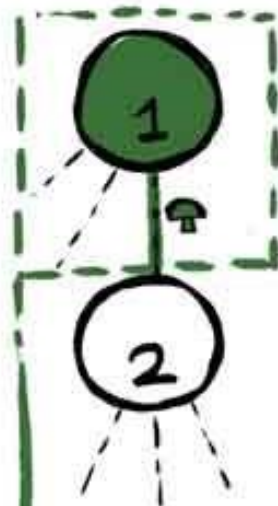
THOSE FLOWERS LOOK PRETTY, I'M FEELING DRAWN TOWARDS "A"...

A: 80%

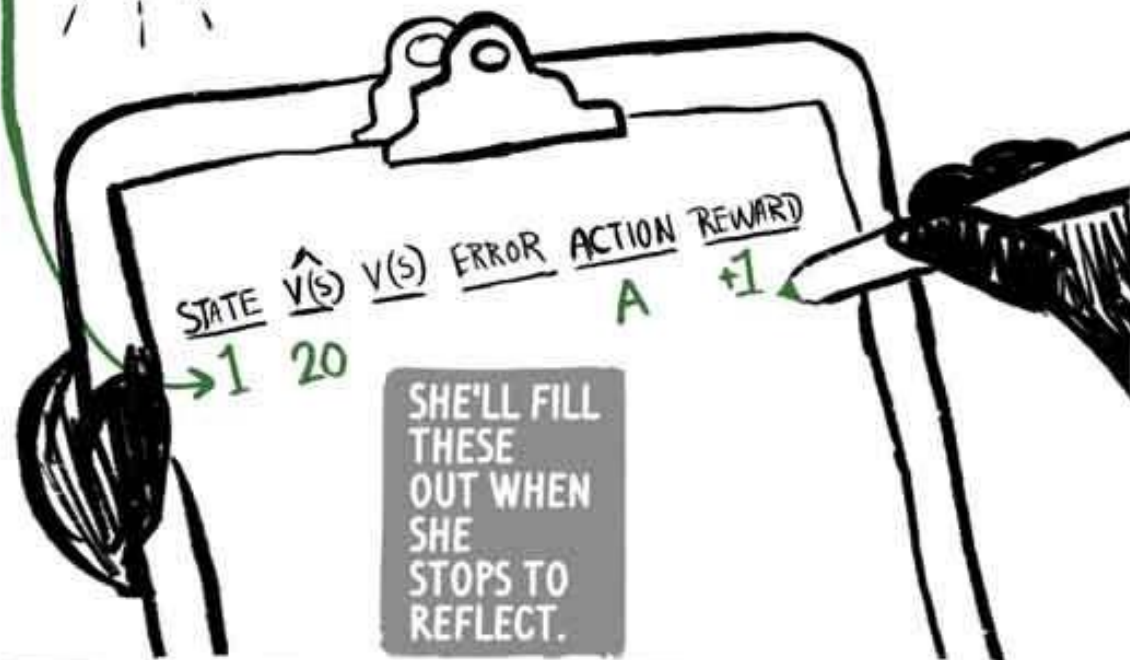
B: 10%

C: 10%





THIS SET OF STATE-ACTION-REWARD
MAKES UP A SINGLE OBSERVATION.
SHE'LL RECORD THIS ROW OF DATA IN
HER JOURNAL BUT SHE'S NOT GOING TO
REFLECT ON IT JUST YET.

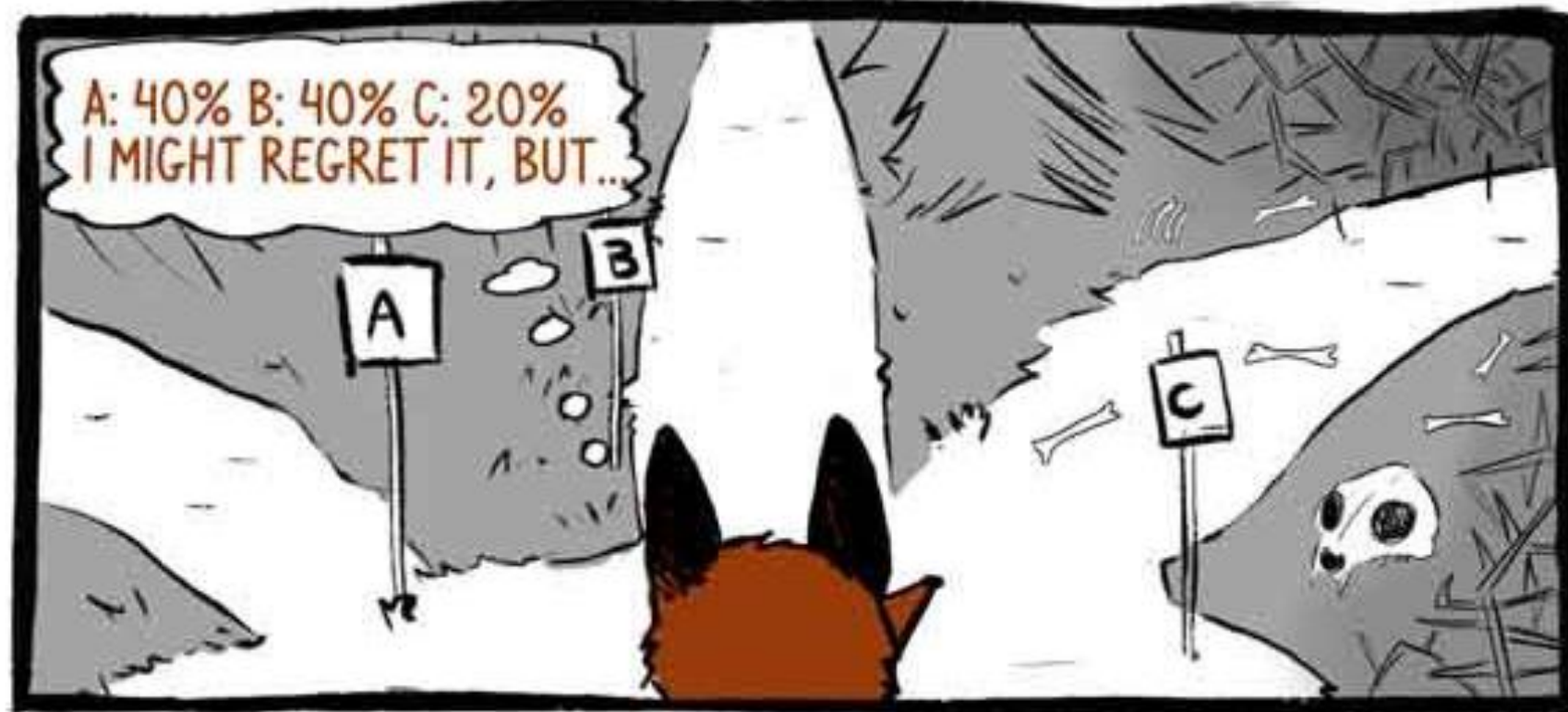


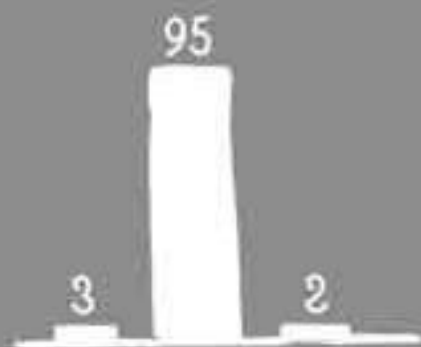
SOME AUTHORS ASSOCIATE REWARD 1 WITH TIMESTEP 1, OTHERS ASSOCIATE IT WITH TIMESTEP 2. ALL ARE REFERRING TO THE SAME CONCEPT: A REWARD IS ASSOCIATED WITH THE STATE AND ACTION DIRECTLY PRECEDING IT.

SHE REPEATS THE PROCESS AGAIN



STATE	$\hat{V}(s)$	$V(s)$	ERROR	ACTION	REWARD
1	20			A	+1
2	19			C	+20
3	22			B	+2





LOW
ENTROPY
DISCOURAGE

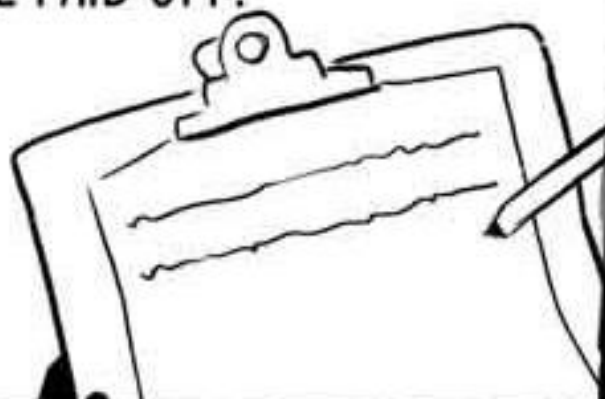


HIGH
ENTROPY
ENCOURAGE

TO FURTHER ENCOURAGE EXPLORATION, A VALUE CALLED ENTROPY IS SUBTRACTED FROM THE LOSS FUNCTION. ENTROPY REFERS TO THE "SPREAD" OF THE ACTION DISTRIBUTION.

+5!

LOOKS LIKE THE GAMBLE PAID OFF!



A SIMPLE **POLICY GRADIENT FOX** WOULD LOOK AT THE ACTUAL RETURNS FOLLOWING AN ACTION AND TUNE HER POLICY TO MAKE GOOD RETURNS MORE LIKELY.



LOOKS LIKE MY POLICY FROM THAT STATE LED TO LOSING 20 POINTS, I GUESS I BETTER MAKE "C" LESS LIKELY IN THE FUTURE...

$$\hat{V}(S) = -100$$
$$V(S) = -20$$

ADVANTAGE=80

-20!



BUT WAIT!

IT'S NOT FAIR TO PLACE THE BLAME ON ACTION C. THAT STATE HAD AN ESTIMATED VALUE OF -100, SO TAKING "C" AND ENDING UP WITH -20 WAS ACTUALLY A **RELATIVE IMPROVEMENT** OF 80! I SHOULD MAKE "C" **MORE** LIKELY IN THE FUTURE.

INSTEAD OF TUNING HER POLICY IN RESPONSE TO THE **TOTAL RETURNS** SHE GOT BY TAKING ACTION C, SHE TUNES HER ACTIONS TO THE **RELATIVE RETURNS** OF TAKING ACTION C. THIS IS CALLED THE "ADVANTAGE".

WHAT WE CALLED THE **ADVANTAGE** IS JUST THE **ERROR**. AS THE **ADVANTAGE**, CRANBERRY USES IT TO MAKE ACTIONS THAT WERE SURPRISINGLY GOOD MORE LIKELY. AS THE **ERROR**, SHE USES THE SAME QUANTITY TO NUDGE HER INNER CRITIC TO MAKE BETTER ESTIMATIONS OF STATE VALUES.

ACTOR USES ADVANTAGE



WOW, THAT WORKED OUT BETTER THAN I THOUGHT. ACTION C MUST HAVE BEEN A GOOD IDEA.

CRITIC USES ERROR



BUT WHY WAS I SURPRISED IN THE FIRST PLACE? I PROBABLY SHOULDN'T HAVE ESTIMATED THAT STATE AS NEGATIVELY AS I DID.



NOW WE CAN SHOW HOW TOTAL LOSS IS COMPUTED—THIS IS THE FUNCTION WE MINIMIZE TO IMPROVE OUR MODEL.

TOTAL LOSS = ACTION LOSS + VALUE LOSS - ENTROPY.

NOTICE WE'RE SHOVING GRADIENTS OF THREE QUALITATIVELY DIFFERENT TYPES BACK THROUGH A SINGLE NN. THIS IS EFFICIENT BUT IT CAN MAKE CONVERGENCE MORE DIFFICULT.

Formal problem

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) Q_{\tau_i, t}$$

то что было в policy gradient

$$Q^{\pi}(s_t, a_t) = \sum_{t'=t}^T E_{\pi_{\theta}}[r(s_{t'}, a_{t'}) | s_t, a_t],$$
$$V^{\pi}(s_t) = E_{a_t \sim \pi_{\theta}(a_t | s_t)}[Q^{\pi}(s_t, a_t)] = \sum_{t'=t}^T E_{\pi_{\theta}}[r(s_{t'}, a_{t'}) | s_t]$$

чуть-чуть улучшим,
заменяя семплы на матожидание

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) A^{\pi}(s_t^i, a_t^i)$$

заменяем Q на A
потому что почему бы и нет?

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + E_{s_{t+1} \sim p(s_{t+1} | s_t, a_t)}[V^{\pi}(s_{t+1})] \approx r(s_t, a_t) + V^{\pi}(s_{t+1}),$$
$$A^{\pi}(s_t^i, a_t^i) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1}) - V^{\pi}(s_t),$$

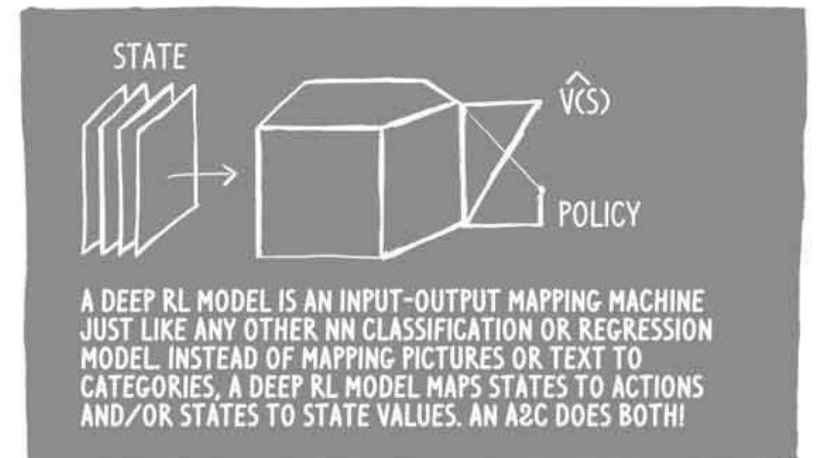
сведём вычисления A к V

$$V^{\pi}(s_t) = r(s_t, a_t) + V^{\pi}(s_{t+1})$$
$$V^{\pi}(s_t) \leftarrow (1 - \beta)V^{\pi}(s_t) + \beta(r(s_t, a_t) + V^{\pi}(s_{t+1}))$$

трюк из SARSA

Advantage Actor-Critic (A2C)

1. производим действие $a \sim \pi_{\theta}(a|s)$, переходим в состояние s' и получаем вознаграждение r ;
2. $V^{\pi}(s) \leftarrow (1 - \beta)V^{\pi}(s) + \beta(r + V^{\pi}(s'))$;
3. $A_{\pi}(s, a) \leftarrow r + V^{\pi}(s') - V^{\pi}(s)$;
4. $\nabla_{\theta} J(\theta) \leftarrow \nabla_{\theta} \log \pi_{\theta}(a|s) A^{\pi}(s, a)$;
5. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$;
6. Если не сошлись к экстремуму, повторить с пункта 1.



Пруфы будут?

Заметим, что если b - константа относительно τ , то

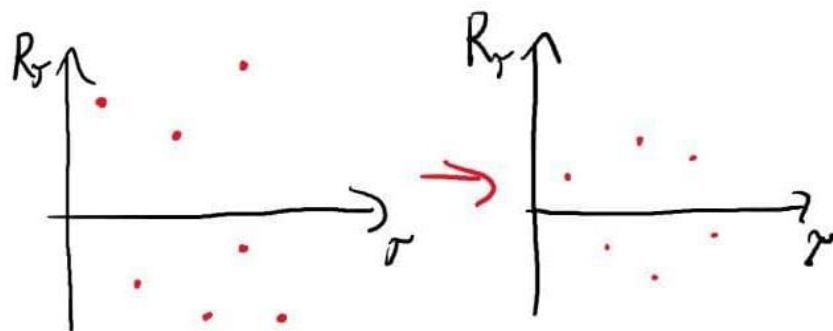
$$E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau)(R_{\tau} - b)] = E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau)R_{\tau}],$$

так как

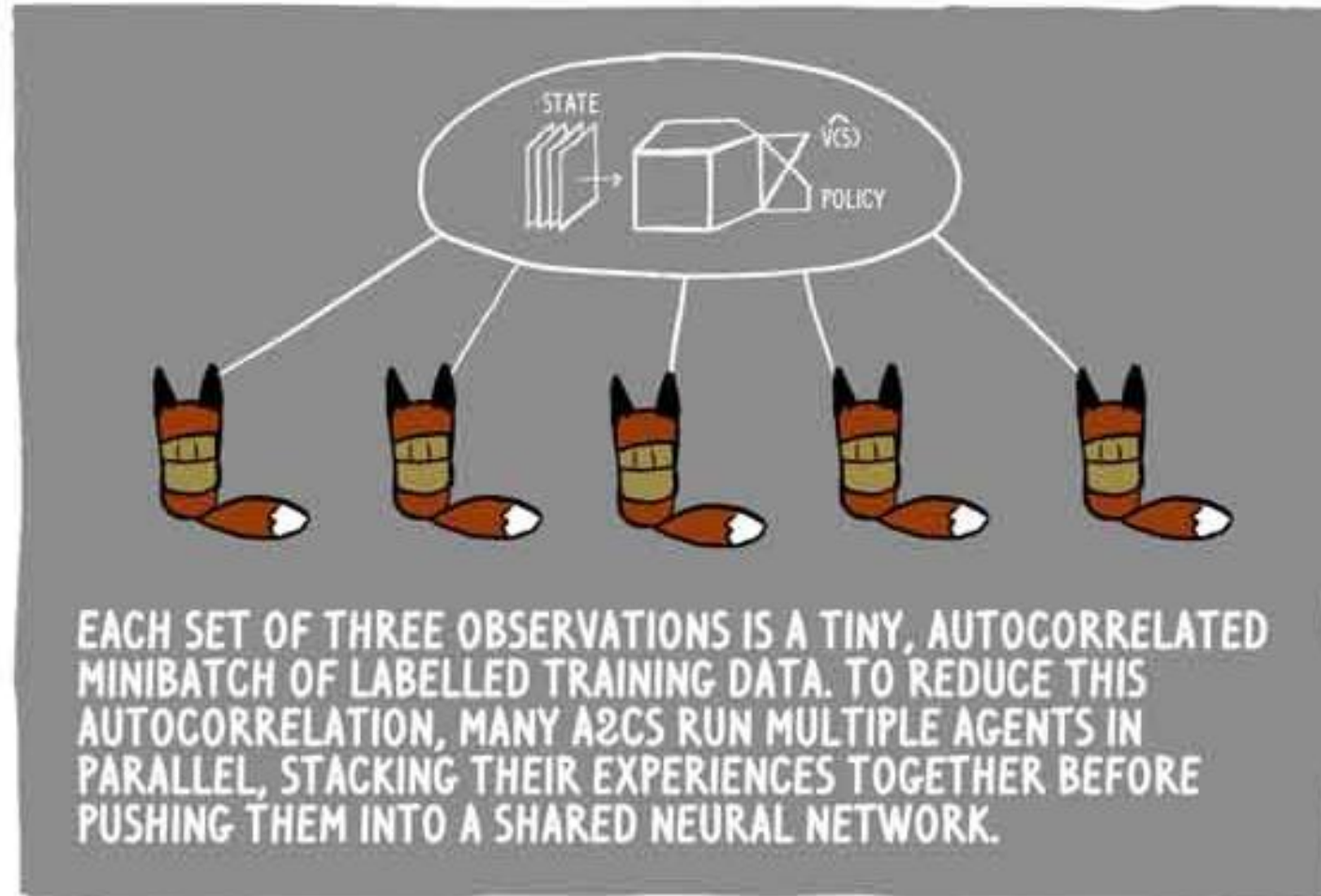
$$E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau)b] = \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) b d\tau = \int \nabla_{\theta} p_{\theta}(\tau) b d\tau = b \nabla_{\theta} \int p_{\theta}(\tau) d\tau = b \nabla_{\theta} 1 = 0,$$

Таким образом, изменение R_{τ} на константу не меняет оценку $\nabla_{\theta} J(\theta)$. Однако дисперсия $Var_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau)(R_{\tau} - b)]$ зависит от b :

$$Var_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau)(R_{\tau} - b)] = \underbrace{E_{\tau \sim p_{\theta}(\tau)} [(\nabla_{\theta} \log p_{\theta}(\tau)(R_{\tau} - b))^2]}_{\text{depends on } b} - \underbrace{E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau)(R_{\tau} - b)]^2}_{=E[\nabla_{\theta} \log p_{\theta}(\tau)R_{\tau}]^2},$$



Asynchronous Advantage Actor-Critic (A3C)



СПИСОК ИСТОЧНИКОВ

- <https://habr.com/ru/post/442522/>
- <https://hackernoon.com/intuitive-rl-intro-to-advantage-actor-critic-a2c-4ff545978752>
- https://neerc.ifmo.ru/wiki/index.php?title=Методы_policy_gradient_и_алгоритм_асинхронного_актера-критика
- <https://www.machinelearningmastery.ru/the-idea-behind-actor-critics-and-how-a2c-and-a3c-improve-them-6dd7dfd0acb8/>
- https://github.com/yandexdataschool/Practical_RL