ML Advanced

Policy Gradient

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в учебной группе

Задаем вопрос
в чат

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Тема вебинара

ML Advanced
Policy Gradient

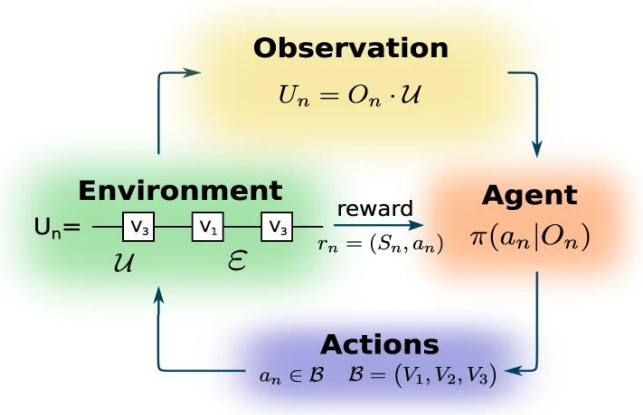
Андрей Канашов

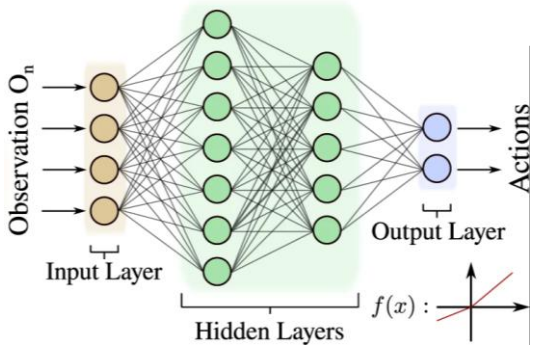
Team Lead Data Scientist в ПИК

- Ценообразование и тарификация
- Рекомендательные системы
- Прогнозирование ключевых метрик
- Анализ клиентского поведения

Маршрут вебинара

- Знакомство
- Постановка задачи
- Policy Gradient theorem
- Алгоритм REINFORCE
- Улучшения алгоритма
- Группа алгоритмов PG





Цели вебинара

К концу занятия вы сможете

1. Применять RL к задачам с непрерывным пространством действий
2. Policy Gradient theorem
3. Понять алгоритм REINFORCE
4. Понять пути улучшения REINFORCE

Смысл

Зачем вам это уметь

1. Программировать задачи обучения с подкреплением для окружений большой размерности и непрерывного пространства действий
2. Иметь базу для перехода к Actor-Critic

Постановка задачи

Markov Property

$$\mathbb{P}[S_{t+1}|S_tA_t] = \mathbb{P}[S_{t+1}|S_1A_1, S_2A_2, \dots, S_tA_t]$$

$$\mathbb{P}[R_t|S_tA_t] = \mathbb{P}[R_t|S_1A_1, S_2A_2, \dots, S_tA_t] = 1$$

Markov Decision Process $\langle \mathcal{S}, \mathcal{S}_F, \mathcal{A}, \mathcal{P}, \mathcal{P}_0, \mathcal{R}, \gamma \rangle$

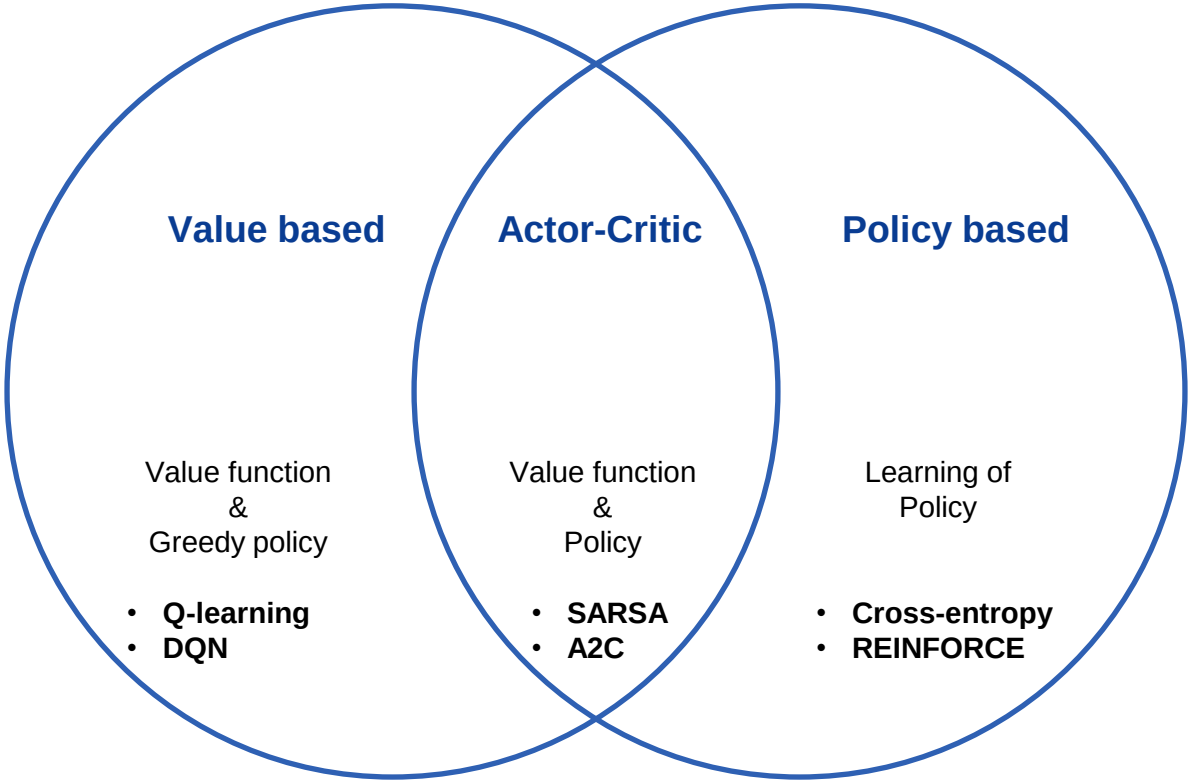
- $\mathcal{S}$ – is an infinite state space
- $\mathcal{S}_F$ – is a set of final states
- $\mathcal{A}$ – is an infinite ($|\mathcal{A}| = m$) action space
- $\mathcal{P}$ – is an unknown transition probability function

  $$\mathcal{P}(s'|s,a) = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$$
- $\mathcal{P}_0$ – is an unknown initial state probability function
- $\mathcal{R}$ – is an unknown reward function

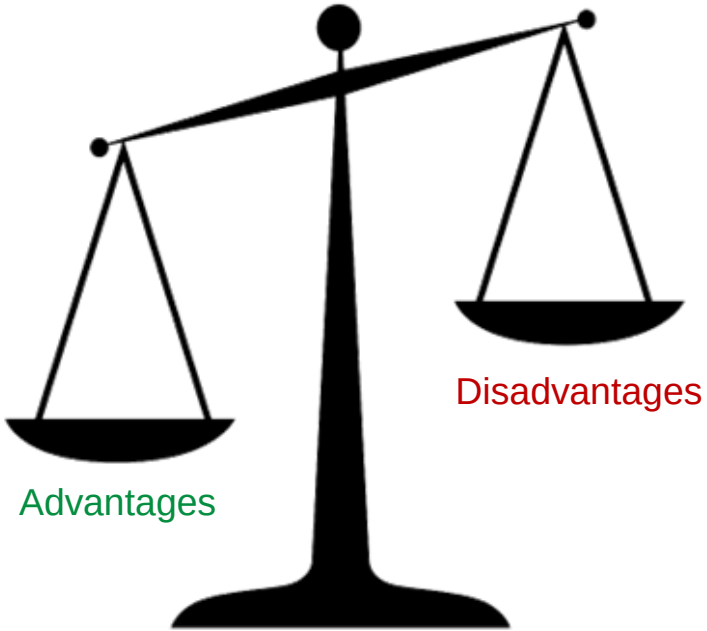
  $$\mathcal{R}(s,a) = R_t, \Leftrightarrow \mathbb{P}[R_t|S_t = s, A_t = a] = 1$$
- $\gamma \in [0, 1]$ – is a discount coefficient

Policy based methods

Value based and Policy based



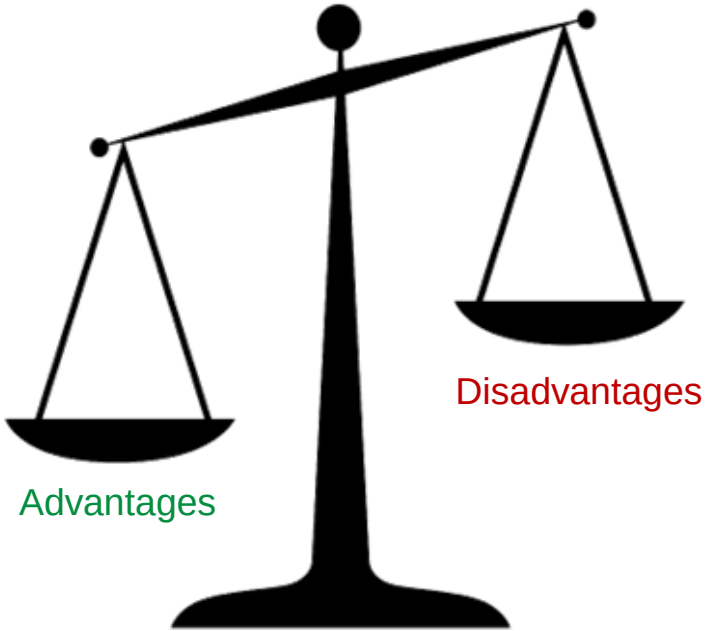
достоинства и недостатки



DQN – достоинства и недостатки

Off-policy

Использование
Experience replay



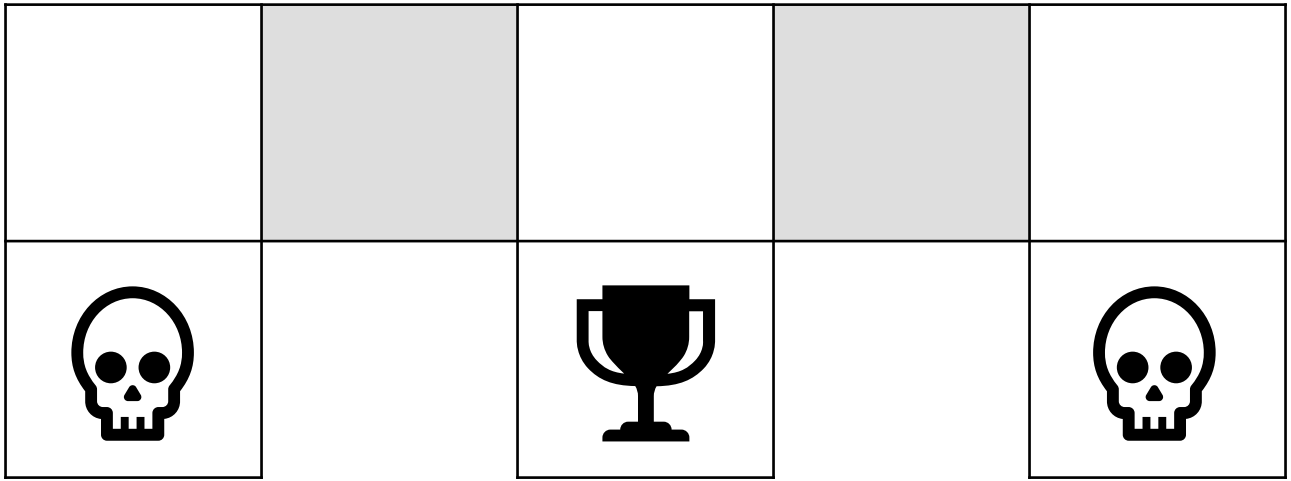
Медленная сходимость, большая дисперсия, плохая устойчивость

Дискретное пространство действий, плохо реагирует на «резкие» переходы

Мы всегда получаем детерминированную стратегию, в силу $\arg\max Q$ в формуле обновления весов

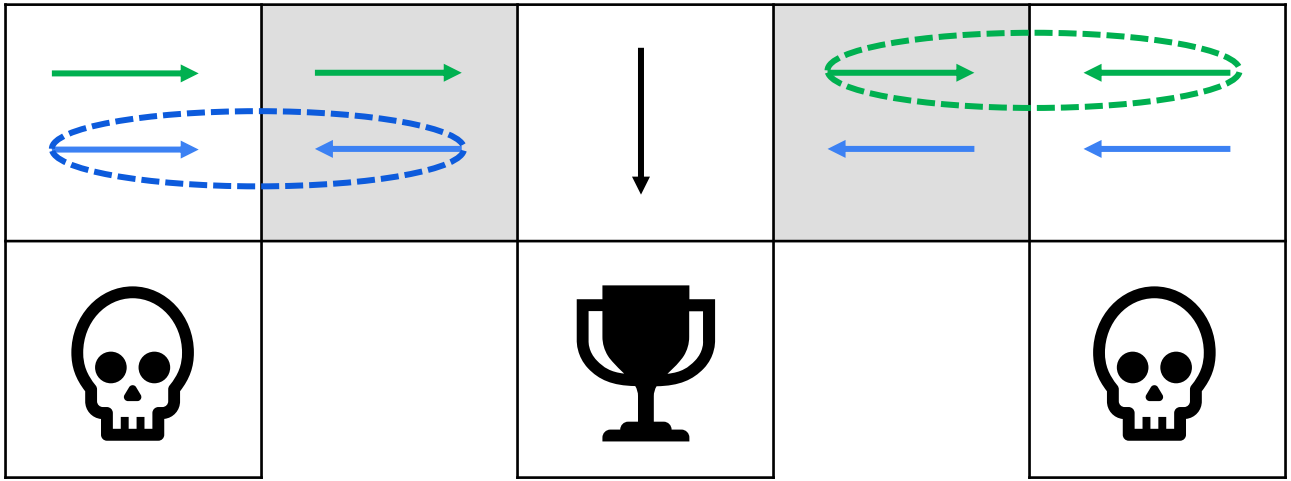
Клеточный мир

Агент не различает серые клетки – q-learning не работает



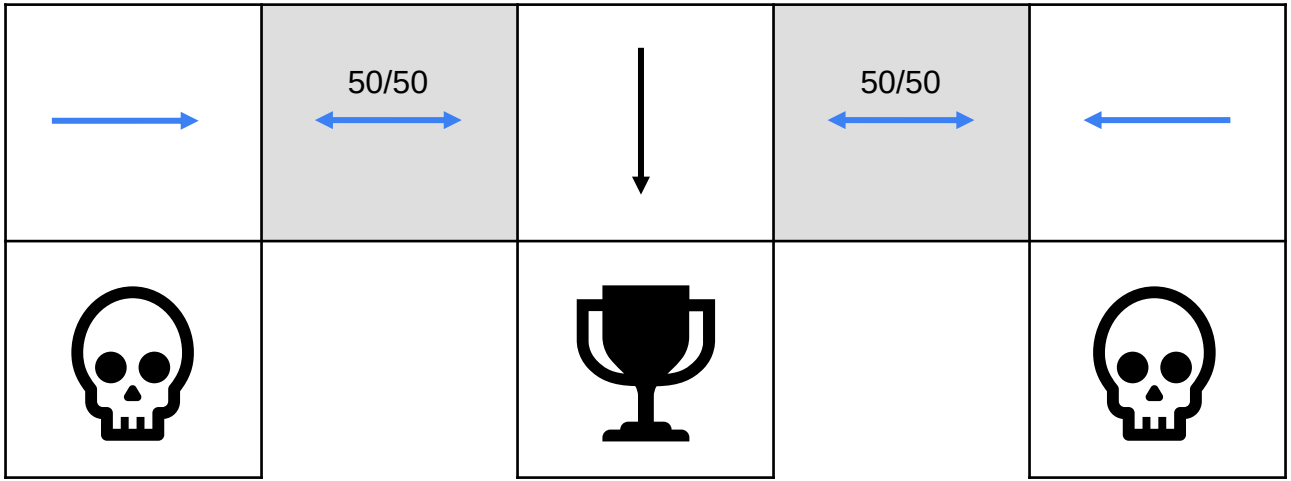
Клеточный мир

Агент не различает серые клетки – q-learning не работает. Агент «застревает» в углу.



Клеточный мир

Оптимальна стохастическая стратегия.



Алгоритм REINFORCE

REINFORCE

Задача – максимизировать суммарную награду агента, т.е максимизировать мат ожидание награды при текущей политике:

$$\mathbb{E}_{\pi}[G]$$

Алгоритм Q-learning (DQN) – искали приближение оптимальной Q-функции.

Вместо приближения Q функции – будем сразу искать оптимальную политику.

Deep Cross-entropy

Оценка политики $\mathbb{E}_{\pi}[G]$:

Получаем траекторию:

$$\mathbb{E}_{\pi_n}[G] = \frac{1}{K}\sum_{k=1}^{k} G(\tau_k), \qquad \pi_n(s) = \left[F^{\theta_n}(s) + noise(\varepsilon)\right]_{\mathcal{A}}$$

Улучшение политики:

Выбираем «элитные» траектории и усредняем действие на них для каждого состояния и обновляем веса модели:

$$Loss(\theta_n) = \frac{1}{|\tau_n|}\sum_{(a|s)\in\tau_n} \left\|F^{\theta_n}(s) - a\right\|^2, \qquad \theta_{n+1} = \theta_n - \eta\nabla_\theta Loss(\theta_n)$$

Не эффективно!

Как улучшать модель на каждом шаге,
а не по множеству траекторий?

REINFORCE

Задача – максимизировать суммарную награду агента, т.е максимизировать мат ожидание награды при какой-то политике π^η (задача конечномерной оптимизации):

$$J(\eta) = \mathbb{E}_{\pi^\eta}[G] \to \max_\eta$$

Поскольку политика есть отображение пространства действий на пространство состояний то можем представить эту награду в виде суммы всех наград по текущей политике, а в случае непрерывного пространства состояний – интегралом:

$$J(\eta) = \frac{1}{N}\sum_k R(a_k), \quad a_k \sim \pi^\eta(\cdot)$$

$$J(\eta) = \mathbb{E}_{\pi^\eta}[G] = \int_{a \in \mathcal{A}} \pi^\eta(a) R(a) \mathrm{d}a$$

Как вычислить этот интеграл?

REINFORCE

$$J(\eta) = \mathbb{E}_{\pi^\eta}[G] = \int_{a \in \mathcal{A}} \pi^\eta(a) R(a) \mathrm{d}a$$

Приблизим политику нейросетью. Обновляем веса: $\eta \leftarrow \eta - \alpha \nabla_\eta J(\eta)$

$$\nabla_\eta J(\eta) = \nabla_\eta \int_{a \in \mathcal{A}} \pi^\eta(a) R(a) \mathrm{d}a = \int_{a \in \mathcal{A}} \nabla_\eta \pi^\eta(a) R(a) \mathrm{d}a$$

Как вычислить этот интеграл?

Внимание! $\nabla_\eta \pi^\eta(a)$ не является функцией плотности распределения вероятностей!

⇒ Не можем воспользоваться методами семплирования для стохастического интеграла.

$$\nabla_\eta J(\eta) \approx \frac{J(\eta + \delta) - J(\eta)}{\delta}$$

Метод конечных приращений. В явном виде - просто, быстро, не эффективно, почти не работает!

Policy gradient theorem

Градиент политики можно приблизить выражением:

$$\nabla_\eta J(\eta) = \int\limits_{a\in\mathcal{A}} \nabla_\eta \pi^\eta(a) R(a)\mathrm{d}a = \int\limits_{a\in\mathcal{A}} \pi^\eta(a)\frac{\nabla_\eta \pi^\eta(a)}{\pi^\eta(a)} R(a)\mathrm{d}a = \int\limits_{a\in\mathcal{A}} \pi^\eta(a)\nabla_\eta \ln \pi^\eta(a)\, R(a)\mathrm{d}a =$$

$$\mathbb{E}_{\substack{s\sim\rho_{\pi^\eta} \\ a\sim\pi^\eta}}\left[\nabla_\eta \ln \pi^\eta(a|s) q_{\pi^\eta}(s,a)\right] \approx \boxed{\nabla_\eta \ln \pi^\eta(a|s) q_{\pi^\eta}(s,a) = \nabla_\eta J(\eta)}$$

где $\rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s|\pi)$.

Оценка методом Монте-Карло:

$$q_{\pi^\eta}(s,a) \approx \sum_{i=t}^{T} \gamma^{i-t} R_i = G_t$$

REINFORCE - алгоритм

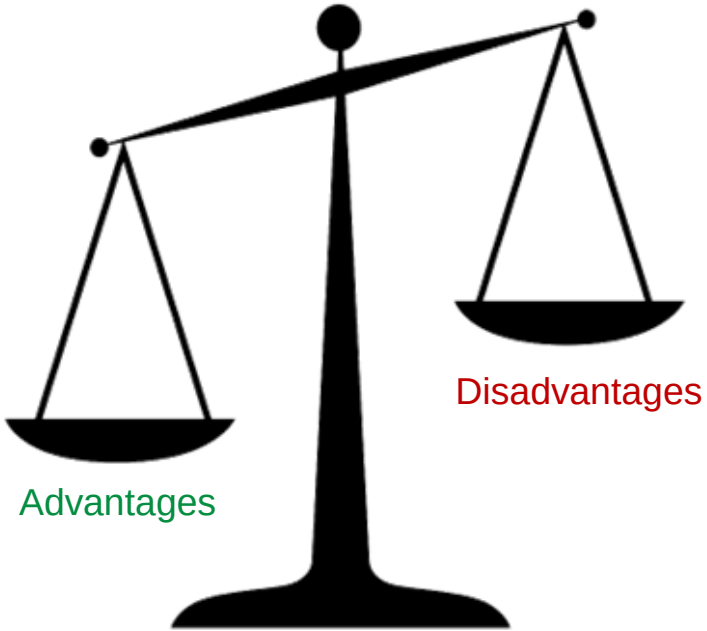
- Задаем начальное приближение политики $\pi^{\eta}(a|s)$
- Запускаем обучение. Для каждого эпизода:
  - Действуя по текущей политике π^{η} получаем траекторию $\tau = (S_0, S_1, \dots, S_T)$, награды $r = (R_0, R_1, \dots, R_{T-1})$ и определяем $g = (G_0, G_1, \dots, G_{T-1})$:

$$\mathrm{G}_t = \sum_{i=t}^{T} \gamma^{i-t} R_i$$

  - Следуя по траектории обновляем веса модели на каждом шаге по правилу:

$$\eta \leftarrow \eta \ - \alpha \nabla_{\eta} \ln \pi^{\eta}(A_t|S_t) G_t$$

достоинства и недостатки



REINFORCE – достоинства и недостатки

Непрерывное пространство действий

Обучаем сразу политику

On-policy

Не можем использовать Experience replay

Не устойчив; Сходится к локальному минимуму

Advantage REINFORCE

В силу стохастичности стратегии в одном и том же состоянии в разных эпизодах могут выбираться разные действия. Это может запутать обучение, потому что один пример требует увеличить вероятность выбора некоторого действия, а другой – уменьшить ее.

Функция преимущества (advantage function):

$$F_t = G_t - V(s_t)$$

$$= Q(S_t, A_t) - V(S_t) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Тогда обновление весов модели производится по формуле:

$$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) F_t$$

Введение преимущества позволяет откалибровать вознаграждения относительно среднего действия в данном состоянии.

Дальнейшие улучшения

Обновление весов модели:

$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) G_t$ - REINFORCE

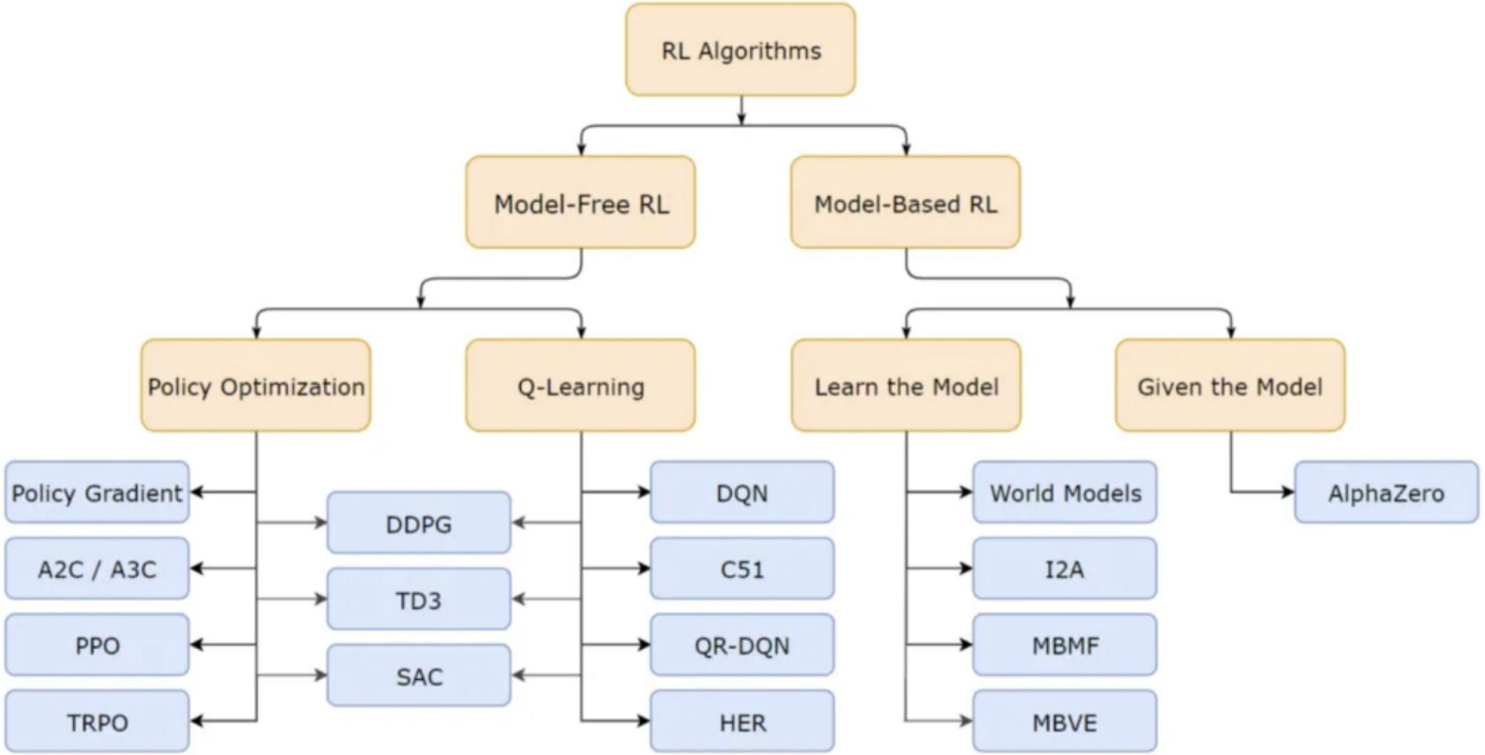
$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) F_t$ - Advantage REINFORCE

$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) Q(S_t, A_t)$ - Actor-Critic

$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$ - Advantage Actor-Critic

$\eta \leftarrow \eta - \alpha \nabla_\eta \left(\frac{1}{N} \sum_{i=1}^{N} Q^\theta(s_i, \pi^\eta(s_i))\right)$ - DDPG (Deep deterministic Policy Gradient)

Algorithms


RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

Практика

Практика

1. Реализовать алгоритм REINFORCE
2. Реализовать алгоритм Advantage REINFORCE

Список материалов для изучения

1. PPO algorithm
2. PPO from scratch with PyTorch
3. Reinforcement Learning frameworks
4. Список ресурсов
5. Лю Ю. (Х.) Обучение с подкреплением на PyTorch: сборник рецептов / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 282 с.
6. Недостатки/неудачи обучения с подкреплением
7. Обзор алгоритмов и их недостатков
8. Особенности основных алгоритмов
9. Эволюционные стратегии в RL

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Цели вебинара

1. Задачи с непрерывным пространством действий
2. Алгоритм REINFORCE
3. Пути улучшения REINFORCE

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Следующие вебинары

26.11.25 – Actor-Critic алгоритм