Лекция 12

Обучение с подкреплением

Никита Юдин, iudin.ne@phystech.edu

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

8 мая 2024

ФПМИ

План лекции

Поговорили о бандитах

Multi-Armed Bandit Problem

$$R_T := TV^\star - \sum_{k=0}^{T-1} Q(a_k) \to \min_{\{a_k\}_{k=0}^{T-1}}.$$

1) ε-жадный бандит;
2) Нестационарный бандит;
3) UCB-бандит;
4) Сэмплирование Томпсона;
5) Bernoulli-бандиты.

Поставили задачу планирования

Определение

Для данного состояния s_0 набор действий $a_0, a_1, a_2, \ldots$ называется *планом*.

Постановка задачи планирования

Пусть известны функция переходов $p(s'|s,a)$ и функция награды, тогда задача планирования есть:

$$\underset{a_0,a_1,a_2,\ldots}{\operatorname{argmax}} \mathbb{E}_{\mathcal{T}|s_0,a_0,a_1,a_2,\ldots} R(\mathcal{T}).$$

World Models

Модель прямой динамики

Модель функции переходов $p_\theta(s' | s, a)$ называется *моделью прямой динамики*.

Сновидениями называется обучение агента на опыте, собранном при помощи приближения динамики среды $p_\theta(s', r | s, a)$.

Условия задачи

Пусть у нас имеется идеальный симулятор среды для MDP с дискретным пространством действий.
Построим алгоритм планирования для задачи планирования: $\underset{a_0, a_1, a_2, \dots}{\operatorname{argmax}} \mathbb{E}_{\mathcal{T} | s_0, a_0, a_1, a_2, \dots} R(\mathcal{T})$.

Самое наивное решение

В случае, когда среда является детерминированной, с помощью известной нам функции переходов $s' = f(s, a)$ мы можем построить полное дерево среды и гарантированно получать оптимальные планы.

Проблемы такого решения

1) В сложных средах дерево будет расти экспоненциально быстро.
2) Наши среды чаще всего стохастичны.

Шаги в сторону MCTS

Определение

Деревом MDP с корнем в состоянии s_0 будем называть дерево, где каждой дуге соответствует действие a; узел на t-ом уровне в дереве соответствуют плану $a_0, a_1, \dots, a_{t-1}$, соответствующему пути от корня.

Будем строить это дерево эффективным способом, сохраняя дополнительную информацию об узле (N) и ребре из него (N, a) — счетчик прохождения по данной дуге $n(N, a)$ и ценность дуги $Q(N, a)$.

Шаги в сторону MCTS

Построение дерева

Один шаг процедуры **Monte-Carlo Tree Search** (MCTS) заключается в проведении четырёх этапов: *Selection*, *Expansion*, *Simulation* и *Update*.

MCTS: Selection

Selection

1) Стартуем в корне, которому соответствует текущее состояние в реальной игре s_0.
2) Далее в цикле, пока не попадём в *лист дерева* на уровне t.
   2.1) При помощи стратегии **TreePolicy**, которая использует данные на дугах, выбираем действие a_i.
   2.2) Спускаемся по дереву на уровень глубже по дуге, соответствующей действию a_i, сэмплируем s_{i+1}, r_{i+1} из известного распределения $p(s_{i+1}, r_{i+1} | s_i, a_i)$.

Получим

Мы запоминаем (знаем) всю траекторию от корня до листа, то есть фактически выбираем таким образом начало некоторого плана $a_0, a_1, \dots, a_{t-1}$.

MCTS: Selection

TreePolicy

Задачей **TreePolicy** является выбор того плана, для которого мы будем дальше строить дерево. В каждом узле N дерева, не являющимся листом, **TreePolicy** знает $|A|$ оценок ценности дуг $Q(N, a)$ из этого узла N и количество посещений этой ветки $n(N, a)$.То есть ему нужно найти самый перспективный путь в дереве, используя эти статистики — эквивалент проблемы многоруких бандитов.

UCB

В качестве **TreePolicy** отлично подойдет UCB-бандит!

MCTS: Expansion

Expansion

На шаге **Expansion** в выбранном на предыдущем этапе листе создаём для каждого действия $a_t \in A$ по новому листу, соответствующему выбору этого действия: таким образом, мы расширяем дерево вдоль выбранной ветки «на один шаг вперёд».

Замечание

Если $|A|$ очень велико, то имеет смысл определять поднабор действий случайным образом.

MCTS: Simulation (Evaluation)

Simulation (Evaluation)

Simulation (Evaluation) заключается в построении некоторой эвристичной оценки *reward-to-go* для каждого нового построенного на предыдущем шаге листа.

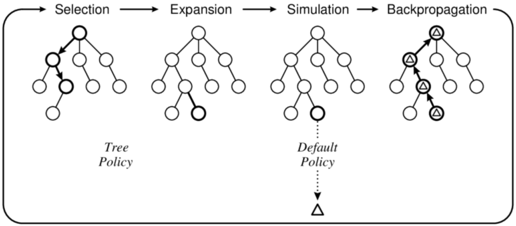
Пример

Для каждого a_t просимулировать игру (или несколько) из s_t, a_t до конца эпизода, выбирая действия при помощи некоторой другой стратегии, которую назовём **Default Policy** (часто это просто случайная стратегия), и которая уже не может использовать никакой информации в узлах дерева, поскольку эти узлы мы еще не строили.

MCTS: Update (Backpropagation)

Update (Backpropagation)

Обновление счётчиков и оценок во всех дугах дерева, по которым мы проходили на данном шаге при помощи полученных на шаге симуляции новых Монте-Карло оценок.



Применение MCTS

Хорошие практики

1) Предобучение дерева не проводится. Дерево постоянно обновляется в процессе реального использования в среде.
2) Перед каждым выбором действия для реальной среды делаем условно 1000 шагов MCTS-процедуры.
3) После очередного реального шага, спускаясь на один узел вниз в дереве, мы можем оставлять только поддерево того узла, в который пришли, для оптимизации.
4) Для выбора действия для реальной среды не обязательно использовать **TreePolicy** (поскольку в нём предусмотрен *exploration*, который не нужен на инференсе). Часто используют выбор с большей вероятностью наиболее часто выбираемых для исследования процедурой действий.

Итоговый алгоритм MCTS

Алгоритм, обозначения

a) **Вход:**
   - s_0 — текущее состояние реальной среды;
   - $p(s', r|s, a)$ — симулятор;
   - C — гиперпараметр UCB-бандита;
   - N_0 — корень текущего дерева в памяти с хранением счетчиков $n(N, a)$ и оценок $Q(N, a)$ в дугах;
   - K — число шагов, T — температура для алгоритма отжига.

b) **Выход:**
   - Стратегия $\pi(a_0|s_0) \propto n(N_0, a_0)^T$.

Итоговый алгоритм MCTS

На k-ом шаге из K:

1. Садимся в корень $N := N_0, s := s_0$.
2. Инициализируем траекторию симуляции: $\mathcal{T} := (s_0)$.
3. Пока N — не лист:
   - выбираем ветку, куда пойти: $n(N) = \sum_a n(N, a)$, $a := \mathrm{argmax}_a(Q(N, a) + C\sqrt{\log n(N)/n(N, a)})$;
   - генерируем $s', r \sim p(s', r|s, a)$;
   - сохраняем a, r, s' в $\mathcal{T}$;
   - спускаемся по дереву: $N \leftarrow \mathrm{child}(N, a), s \leftarrow s'$.
4. Для каждого $a \in A$:
   - создаём узел $\hat{N}$ — ребенка N с дугой для действия a;
   - симулируем $\mathcal{T}_a \sim \pi^{\mathrm{random}}|s, a$;
   - инициализируем $n(\hat{N}, a) := 1, Q(\hat{N}, a) := R(\mathcal{T}_a)$.

Итоговый алгоритм MCTS

5. Для каждой посещённой дуги N, a:
    - считаем $\hat{V}$ — суммарный reward-to-go в траектории $\mathcal{T}$, полученный после посещения данной дуги, где награда после посещения листа оценена как $\frac{1}{|A|}\sum_a R(\mathcal{T}_a)$;
    - $Q(N,a) \leftarrow Q(N,a) + \frac{1}{n(N,a)}(\hat{V} - Q(N,a))$;
    - $n(N,a) \leftarrow n(N,a) + 1$.

Итого

Мы получили очень дорогостоящую по времени, разумную по памяти, но работающую процедуру поиска хороших действий в игре.

Постановка задачи

Условия

Пусть модель динамики среды и награды известны нам как дифференцируемые и детерминированные функции. Пусть также эпизод состоит из T шагов и $\gamma = 1$. При этом пространство действий непрерывно $A \equiv \mathbb{R}^d$. Функции награды и динамики среды дополнительно зависят от дискретной переменной времени $t \in \{1, \ldots, T\}$.

Замечание

В силу детерминированности функции переходов наш выбор действий полностью определяет траекторию. Таким образом будем искать **оптимальное управление**, то есть набор действий $a_1, a_2, \ldots, a_T$, которые приведут к наилучшей траектории.

Постановка задачи

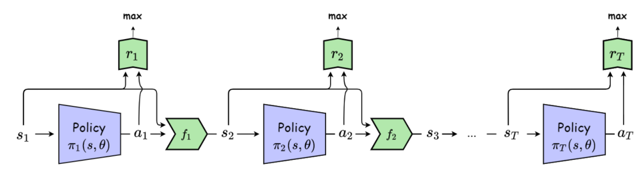
Математическая формулировка

$$\begin{cases} \sum\limits_{t=1}^{T} r_t(s_t, a_t) \to \max\limits_{a_1, \ldots, a_T}; \\ s_t = f_{t-1}(s_{t-1}, a_{t-1}), \; t = \overline{2, T}. \end{cases}$$

Варианты решений

Нейросетью «в лоб»

Можно построить нечто подобное RNN:



Проблема

Так же как и в RNN в такой сети при большом T градиенты будут затухать.

LQR: Мотивация

Слегка упростим задачу

Оказывается, если мы заменим функции наград на квадратичное приближение, а ограничение — функцию динамики — на линейное приближение, то мы сможем за счёт структуры задачи просто методом динамического программирования аналитически решить задачу.

Обозначения

$$f_t(s, a) = F_t \begin{bmatrix} s \\ a \end{bmatrix} + f_t;$$

$$r_t(s, a) = \frac{1}{2} \begin{bmatrix} s \\ a \end{bmatrix}^\top R_t \begin{bmatrix} s \\ a \end{bmatrix} + \begin{bmatrix} s \\ a \end{bmatrix}^\top r_t.$$

LQR

Обозначения

Матрицы F_t, R_t и векторы r_t, f_t считаем известными. Будем считать, что блоки матрицы R_t и блоки вектора r_t выглядят следующим образом:

$$R_t := \begin{bmatrix} R_{t,s,s} & R_{t,s,a} \\ R_{t,a,s} & R_{t,a,a} \end{bmatrix}; r_t := \begin{bmatrix} r_{t,s} \\ r_{t,a} \end{bmatrix}.$$

Для упрощения выкладок также будем полагать, что $R_{t,s,a} = R_{t,a,s}^\top$.

LQR

Теорема

Оптимальное действие в последний момент времени a_T^* — линейная форма от состояния s_T.

Доказательство

Рассмотрим последний момент времени T и распишем оптимальную Q-функцию:

$$Q_T^\star(s_T, a_T) = r_T(s_T, a_T) = \frac{1}{2}\begin{bmatrix} s_T \\ a_T \end{bmatrix}^\top R_T \begin{bmatrix} s_T \\ a_T \end{bmatrix} + \begin{bmatrix} s_T \\ a_T \end{bmatrix}^\top r_T.$$

Слагаемых так мало, потому что действие последнее в эпизоде.

LQR

Доказательство

Мы легко можем найти оптимальное действие, если на последнем шаге оказались в состоянии s_T, промаксимизировав $Q^\star_T$ по действию a_T:

$$a^\star_T = \operatorname*{argmax}_{a_T} Q^\star_T(s_T, a_T).$$

Ищем оптимум квадратичной формы, приравняв градиент $Q^\star_T$ по действиям к нулю:

$$\nabla_{a_T} Q^\star_T(s_T, a_T) = R_{T,a,a} a_T + R_{T,a,s} s_T + r_{T,a} = 0;$$
$$a^\star_T = -R^{-1}_{T,a,a}(R_{T,a,s} s_T + r_{T,a}).$$

Видно, что оптимальное действие — линейная форма от последнего состояния.

LQR

Обозначим

$$K_T := -R^{-1}_{T,a,a} R_{T,a,s}, \;\; k_T := -R^{-1}_{T,a,a} r_{T,a};$$
$$a^\star_T = K_T s_T + k_T.$$

Теорема

Ценность последнего состояния $V^\star(s_T)$ — квадратичная форма от состояния s_T.

LQR

Доказательство

По определению, в силу детерминированности среды, $V^\star(s_T) = Q^\star(s_T, a_T^\star)$; осталось заметить, что подстановка линейной формы в квадратичную даст квадратичную:

$$V^\star(s_T) = Q^\star(s_T, a_T^\star) = \frac{1}{2}\begin{bmatrix} s_T \\ a_T^\star \end{bmatrix}^\top R_T \begin{bmatrix} s_T \\ a_T^\star \end{bmatrix} + \begin{bmatrix} s_T \\ a_T^\star \end{bmatrix}^\top r_T$$

$$= \frac{1}{2}\begin{bmatrix} s_T \\ K_T s_T + k_T \end{bmatrix}^\top R_t \begin{bmatrix} s_T \\ K_T s_T + k_T \end{bmatrix} + \begin{bmatrix} s_T \\ K_T s_T + k_T \end{bmatrix}^\top r_T = (*),$$

введя обозначения: $V_T := R_{T,s,s} + K^\top R_{T,a,a} K_T + K_T^\top R_{T,a,s} + R_{T,s,a} K_T$, $v_T := R_{T,s,a} k_T + r_T^\top K_T^\top r_T$, получим:

LQR

Доказательство

$$(*) = \frac{1}{2} s_T^\top V_T s_T + v_T^\top s_T.$$

Осталось понять, что мы можем раскручивать это к началу времён, не выходя из квадратичных форм. Так и есть.

Теорема

Оптимальные оценочные функции $Q_t^\star$ — квадратичные формы от $\begin{bmatrix} s_t \\ a_t \end{bmatrix}$, оптимальные действия $a_t^\star$ — линейные формы от s_t, а оптимальные оценочные функции $V_t^\star$ — квадратичные формы от s_t.

LQR

Доказательство

Начнём с $T-1$. Из определений и детерминированности:

$$Q^\star_{T-1}(s_{T-1}, a_{T-1}) = r_{T-1}(s_{T-1}, a_{T-1}) + V^\star_T(s_T) =$$
$$r_{T-1}(s_{T-1}, a_{T-1}) + V^\star\left(F_T\begin{bmatrix} s_{T-1} \\ a_{T-1}\end{bmatrix} + f_T\right) = (*).$$

В последнее слагаемое подставили линейную форму динамики среды. При этом подстановка линейной формы в квадратичную оставит её квадратичной.

$$(*) = r_{T-1}(s_{T-1}, a_{T-1}) +$$
$$+\frac{1}{2}\left[F_T\begin{bmatrix} s_{T-1} \\ a_{T-1}\end{bmatrix} + f_T\right]^\top V_T\left[F_T\begin{bmatrix} s_{T-1} \\ a_{T-1}\end{bmatrix} + f_T\right] + v_T^\top\left[F_T\begin{bmatrix} s_{T-1} \\ a_{T-1}\end{bmatrix} + f_T\right] =$$

LQR

Доказательство

$$= r_{T-1}(s_{T-1}, a_{T-1}) + \frac{1}{2}\begin{bmatrix} s_{T-1} \\ a_{T-1} \end{bmatrix}^\top F_T^\top V_T F_T \begin{bmatrix} s_{T-1} \\ a_{T-1} \end{bmatrix} + \begin{bmatrix} s_{T-1} \\ a_{T-1} \end{bmatrix}^\top \cdot$$

$$\cdot (F_T^\top V_T f_T + F_T^\top v_T) + \mathrm{const}(a_{T-1}).$$

Переписываем в виде квадратичной формы:

$$Q^\star_{T-1}(s_{T-1}, a_{T-1}) = \frac{1}{2}\begin{bmatrix} s_{T-1} \\ a_{T-1} \end{bmatrix}^\top Q_{T-1} \begin{bmatrix} s_{T-1} \\ a_{T-1} \end{bmatrix} +$$

$$+ \begin{bmatrix} s_{T-1} \\ a_{T-1} \end{bmatrix}^\top q_{T-1} + \mathrm{const}(a_{T-1}).$$

LQR

Доказательство

$$Q_{T-1} := R_{T-1} + F_T^\top V_T F_T;$$
$$q_{T-1} := r_{T-1} + F_T^\top V_T f_T + F_T^\top v_T.$$

Коли это снова квадратичная формула, мы можем посчитать оптимальное действие, которое будет линейной формой:

$$a_{T-1} = K_{T-1} s_{T-1} + k_{T-1};$$
$$K_{T-1} := -Q_{T-1,a,a}^{-1} Q_{T-1,a,s}, \; k_{T-1} := -Q_{T-1,a,a}^{-1} q_{T-1,a}.$$

Подставляем её в $V_{T-1}^\star$ и получаем квадратичную и так далее с $T-2$ до 1.

LQR: обратный проход

LQR: обратный проход, обозначения

a) **Вход**:
- F_t, f_t — функция динамики, $t = \overline{1, T}$;
- R_t, r_t — функция награды, $t = \overline{1, T}$.

b) **Выход**:
- $\pi_t(s) := K_t s + k_t$, $t = \overline{1, T}$.

LQR: обратный проход

LQR: обратный проход

Инициализировать $V_{T+1} = 0_{|\mathcal{S}|\times|\mathcal{S}|}$, $v_{T+1} = 0_{|\mathcal{S}|}$.

for t от T до 1:

1. Считаем Q-функцию:

$$Q_t := R_t + F_{t+1}^\top V_{t+1} F_{t+1};$$
$$q_t := r_t + F_{t+1}^\top V_{t+1} f_{t+1} + F_{t+1}^\top v_{t+1}.$$

2. Считаем оптимальную стратегию:

$$K_t := -Q_{t,a,a}^{-1} Q_{t,a,s};$$
$$k_t := -Q_{t,a,a}^{-1} q_{t,a}.$$

LQR: обратный проход

LQR: обратный проход

3. Считаем V-функцию:

$$V_t := Q_{t,s,s} + K_t^\top Q_{t,a,a} K_t + K_t^\top Q_{t,a,s} + Q_{t,s,a} K_t;$$
$$v_t := Q_{t,s,a} k_t + q_t^\top K_t^\top q_t.$$

Поскольку в рамках сделанных предположений мы на самом деле детерминированно знаем, в каких состояниях окажемся (считаем стартовое состояние s_1 также известным), мы можем просто вывести оптимальное управление:

LQR: прямой проход

LQR: прмяой проход, обозначения

a) **Вход**:
   - K_t, k_t — стратегия с обратного прохода, $t = \overline{1, T}$;
   - f_t — функция динамики, $t = \overline{1, T}$.

b) **Выход**:
   - $a_1, a_2, \ldots, a_T$ — план.

LQR: прямой проход

LQR: прмяой проход

for t от 1 до T:

1. $a_t = K_t s_t + k_t$;
2. $s_{t+1} = f(s_t, a_t)$.

Постановка задачи

Условие

LQR обобщается на случай недетермнированных сред, у которых динамика — нормальное распределение с линейной функцией от предыдущих состояний-действий и какой-то фиксированной (зависящей только от момента времени t, но не состояний-действий) матрицей ковариации:

$$p(s_t | s_{t-1}, a_{t-1}) := \mathcal{N}(f_t(s_{t-1}, a_{t-1}), \Sigma_t). \tag{1}$$

Что поменяется?

Теорема

В предположении 1 схема LQR остаётся неизменной.

Доказательство

Единственное, что поменялось — это зависимость V-функции от Q-функции:

$$Q^*_{t-1}(s_{t-1}, a_{t-1}) = r_{t-1}(s_{t-1}, a_{t-1}) + \mathbb{E}_{s_t} V^\star_t(s_t).$$

Покажем, что формулы для матрицы Q_{t-1} и векторы q_{t-1} не поменялись, а изменилась только константа (которая на вывод оптимальной стратегии не влияет).

Что поменяется?

Доказательство

$$\mathbb{E}_{s_t} V_t^\star(s_t) = \mathbb{E}_{s_t}\left[\frac{1}{2} s_t^\top V_t s_t + v_t^\top s_t + \mathrm{const}(s_t)\right] = (*);$$

из matrix cookbook ур. 318 :

$$\mathbb{E}_{s_t \sim \mathcal{N}(\mu_t, \Sigma_t)} s_t^\top V_t s_t = \mathrm{Tr}(V_t \Sigma_t) + \mu_t^\top V_t \mu_t,$$

где $\mu_t = f_t(s_{t-1}, a_{t-1})$ — среднее гауссианы. Итого получим:

$$(*) = \frac{1}{2}\mathrm{Tr}(V_t \Sigma_t) + \frac{1}{2}\mu_t^\top V_t \mu_t + v_t^\top \mu_t + \mathrm{const}(s_t) = V_t^\star(\mu_t) + \mathrm{const}(s_t),$$

то есть поменялась исключительно константа, которая на оптимальное управление не влияет.

Iterative LQR (iLQR)

Мотивация

Вернёмся к детерминированному случаю. Что делать, если функции f, r не являются линейными и квадратичными соответственно? Мы хотели воспользоваться локальным приближением этих функций в окрестности некоторого текущего плана, и в качестве приближений будем использовать разложение в ряд Тейлора до первого и до второго члена соответственно.

Iterative LQR (iLQR)

Реализация

Итак, возьмём какой-нибудь план и просчитаем честным прямым проходом точные значения состояний, в которых мы окажемся. Для полученной траектории $\hat{s}_1, \hat{a}_1, \hat{s}_2, \ldots, \hat{s}_T, \hat{a}_T$ разложим функции переходов и награды в Тейлора следующим образом:

$$f_t(s_{t-1}, a_{t-1}) \approx f_t(\hat{s}_{t-1}, \hat{a}_{t-1}) + \nabla_{s,a} f(\hat{s}_{t-1}, \hat{a}_{t-1})^\top \begin{bmatrix} s_{t-1} - \hat{s}_{t-1} \\ a_{t-1} - \hat{a}_{t-1} \end{bmatrix};$$

$$r_t(s_t, a_t) \approx \frac{1}{2} \begin{bmatrix} s_t - \hat{s}_t \\ a_t - \hat{a}_t \end{bmatrix}^\top \nabla^2_{s,a} r_t(\hat{s}_t, \hat{a}_t) \begin{bmatrix} s_t - \hat{s}_t \\ a_t - \hat{a}_t \end{bmatrix} + \nabla_{s,a} r(\hat{s}_t, \hat{a}_t)^\top \begin{bmatrix} s_t - \hat{s}_t \\ a_t - \hat{a}_t \end{bmatrix}.$$

Iterative LQR (iLQR)

Реализация

Вводим обозначения аналогично LQR:

$$F_t := \nabla_{s,a} f_t(\hat{s}_{t-1}, \hat{a}_{t-1}), \quad f_t := f_t(\hat{s}_{t-1}, \hat{a}_{t-1});$$
$$R_t := \nabla^2_{s,a} r_t(\hat{s}_t, \hat{a}_t), \quad r_t := \nabla_{s,a} r_t(\hat{s}_t, \hat{a}_t).$$

Чтобы применить алгоритм как мы делали это раньше технически нужно только учесть, что в этой рассматриваемой задаче мы «перецентрировали» пространства состояний и действий: $s_t - \hat{s}_t$, $a_t - \hat{a}_t$. Чтобы это учесть в forward pass нужно: $a_t = K_t(s_t - \hat{s}_t) + k_t + \hat{a}_t$, что соответствует простому учёту добавленных слагаемых в свободном векторе: $k_t \leftarrow k_t - K_t\hat{s}_t + \hat{a}_t$.

Iterative LQR (iLQR)

Алгоритм, обозначения

a) **Вход**:
   - r — функция награды;
   - f — функция динамики.

b) **Выход**:
   - $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_T$ — план.

Iterative LQR (iLQR)

Алгоритм

Проинициализировать траекторию $\hat{s}_1, \hat{a}_1, \hat{s}_2, \ldots, \hat{s}_T, \hat{a}_T$ случайно.

На каждом шаге от 1 **до** N**:**

1) Получить F_t, f_t, R_t, r_t по формулам, $t = \overline{1, T}$;
2) Получить матрицы K_t, k_t при помощи алгоритма LQR с матрицами динамики F_t, f_t и награды R_t, r_t, $t = \overline{1, T}$;
3) Учесть коррекцию $k_t \leftarrow k_t - K_t \hat{s}_t + \hat{a}_t$, $t = \overline{1, T}$;
4) При помощи стратегии $\pi(s_t) = K_t s_t + k_t$ заспавнить траекторию $\hat{s}_1, \hat{a}_1, \hat{s}_2, \ldots, \hat{s}_T, \hat{a}_T$, используя честный прямой проход с использованием точных функций f, r, $t = \overline{1, T}$.