Машинное обучение
Обучение с подкреплением

Содержание лекции

- Постановка задачи обучения с подкреплением
- Дисконтирование
- Уравнение Беллмана
- Q-learning
- Exploration/exploitation
- DQN
- Проблемы сходимости
- Reward reshaping
- Стохастические награды

Постановка задачи

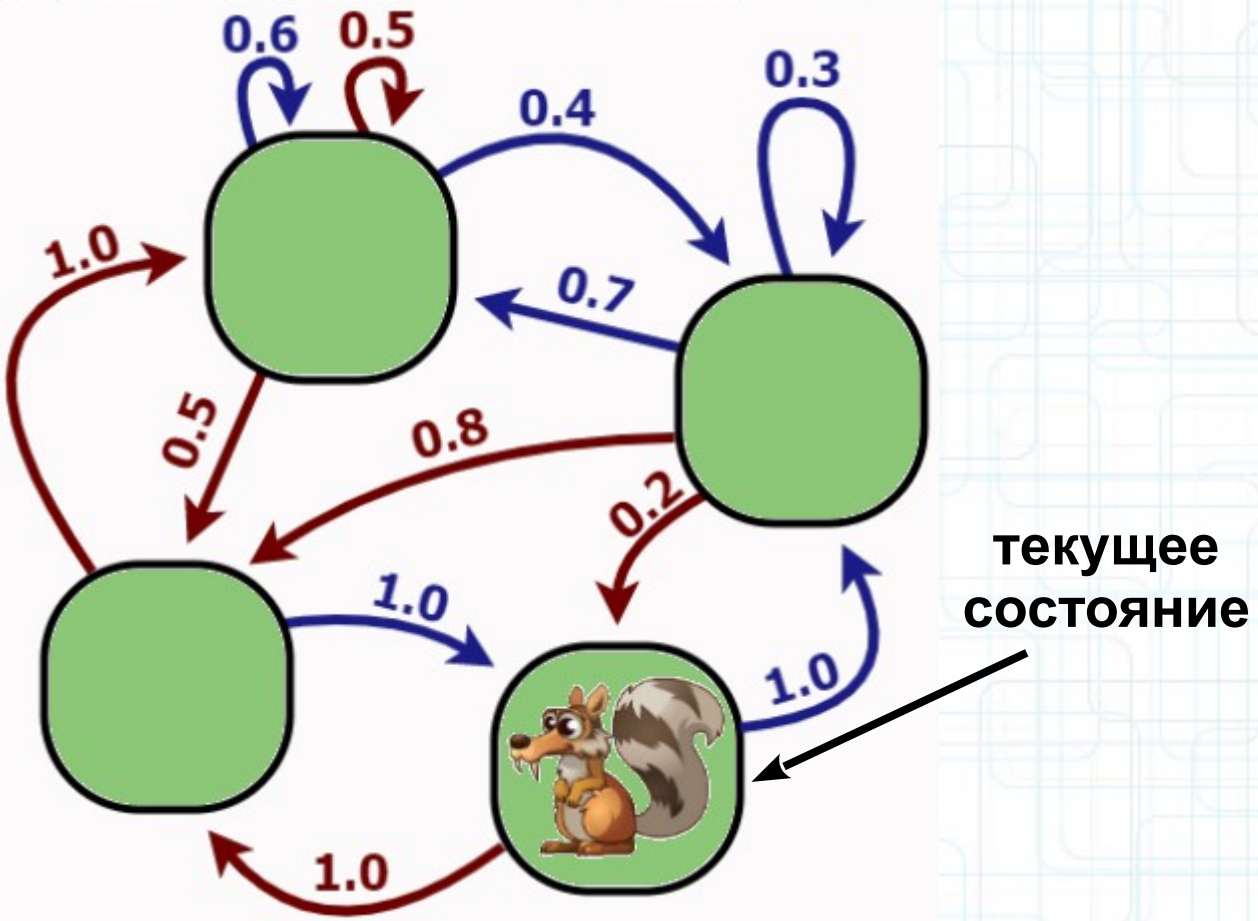
- S — пространство состояний (state space), множество состояний, в которых в каждый момент времени может находится среда
- A — пространство действий (action space), множество вариантов, из которых нужно производить выбор на каждом шаге своего взаимодействия со средой
- P — функция переходов (transition function), которая задаёт изменение среды после того, как в состоянии s было выбрано действие a. В общем случае функция переходов может быть стохастична, и тогда такая функция переходов моделируется распределением p(s'|s,a)
- r(s,a) — функция награды (reward function), выдающая скалярную величину за выбор действия a в состоянии s. Это наш «обучающий сигнал»

Терминология

- Агент (agent) - субъект, взаимодействующий со средой и влияющий на неё
- Стратегия (policy) – правило $\pi(a|s)$, как выбирать действия в зависимости от текущего состояния среды
- Эпизод - набор взаимодействий со средой от начального состояния до попадания в терминальное состояние
- Марковский процесс принятия решений (MDP) – четверка (S, A, P, r)
- Марковость – следующее состояние зависит только от предыдущего и действия (нет зависимости от истории действий и нет изменения законов мира с течением времени)
- Частично наблюдаемый MDP – агенту доступны лишь какие-то частичные наблюдения состояния

Пример MDP

Состояния – зеленые; переходы в результате действия A – красные; в результате действия B – синие. Числа – вероятности p(s'|s,a)



Эквивалентные определения

- r(s') – можно считать, что награда дается не за выполнение действия a в прошлом состоянии s, а за переход в новое состояние s'. Тогда r(s') фактически будет наградой r(s) на следующем шаге
- r(s',a,s) – можно свести к r(s,a), если сделать состоянием пару (s,s')
- Случайную награду можно свести к детерминированной, если размножить состояние s' по числу вероятных наград

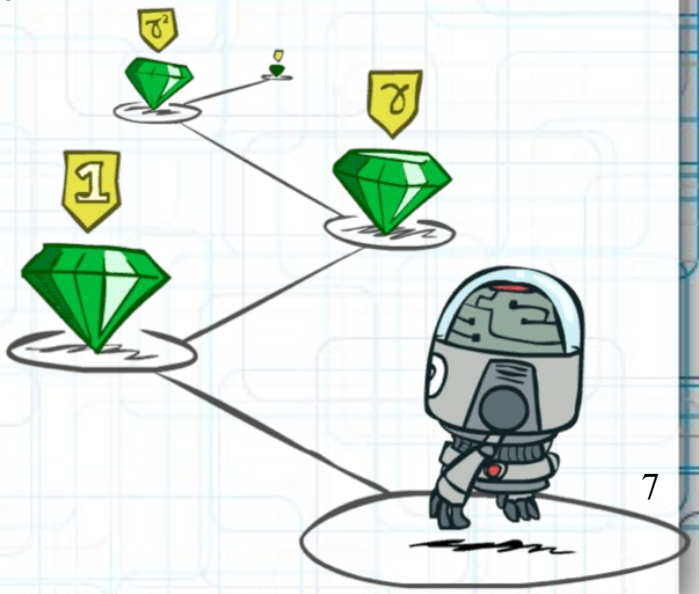
Дисконтирование награды

- Мы хотим научиться выбирать действия так, чтобы собирать в среднем как можно больше награды, т.е. найти оптимальную траекторию T (последовательность действия и состояний)

$$\mathbb{E}_{\mathcal{T}\sim\pi}\sum_{t\geq 0} r_t \to \max_{\pi}$$

- Но если игра может продолжаться бесконечно долго, то оптимальная награда равна бесконечности
- Выход: умножать награду на следующем шаге на число < 1

$$\mathbb{E}_{\mathcal{T}\sim\pi}\sum_{t\geq 0} \gamma^t r_t \to \max_{\pi}$$



Решение

- Допустим, что мы смогли хотя бы приближенно найти оценочную функцию Q*(s,a) - то, сколько максимально награды можно (в среднем) набрать после выбора действия a из состояния s

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

- Принцип оптимальности Беллмана: оптимальной является жадная стратегия выбора максимального действия Q*

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

- Как найти Q* ?

Уравнение оптимальности Беллмана для Q-функции

- Q* выражается через саму себя: в следующем будущем состоянии s' мы, в предположении оптимальности поведения, выберем то действие a', на котором достигается максимум Q(s',a')

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

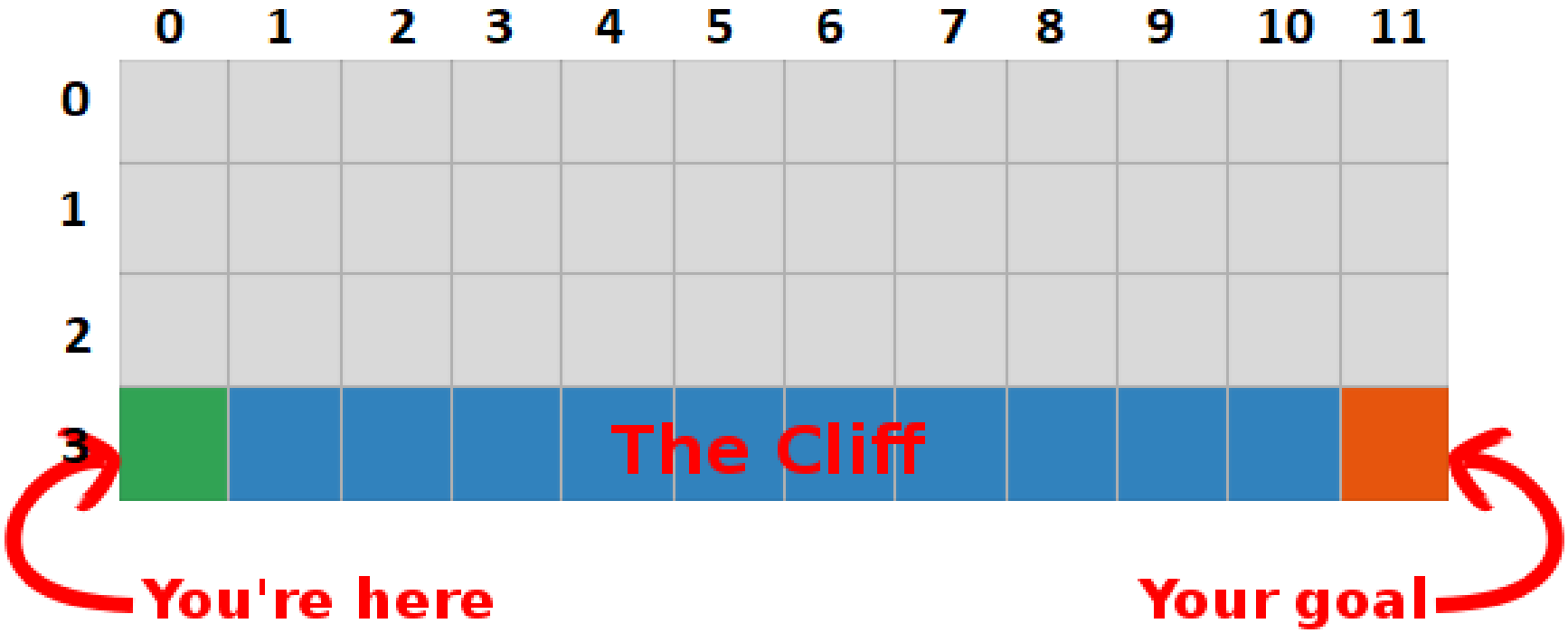
- Это уравнение предлагает нам метод простой итерации для поиска Q*

Метод простой итерации (Q-learning)

- В соответствии с некоторой начальной стратегией (например, случайной) можно генерировать переходы (s,a,r,s').
- Математическое ожидание в уравнении Беллмана можно заменить экспоненциальным скользящим средним с параметром 0< α < 1
- В итоге получится формула пересчета Q*:

$$Q^*_{k+1}(s,a) \leftarrow (1-\alpha)Q^*_k(s,a) + \alpha(r + \gamma \max_{a'} Q^*_k(s',a'))$$

Пример: Cliff walking



Фрагмент Q-матрицы

States	up	down	right	left
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0

Пример: Cliff walking

Дилемма Exploration-exploitation

- Если во время обучения использовать жадную оптимальную на данном этапе обучения стратегию, то многие пары (s,a) никогда не будут встречаться просто потому, что мы никогда не выбираем действие a в состоянии s
- Пример застревания алгоритма: мы не умираем в игре, если стоим на месте и никода не ходим
- Что делать?

Дилемма Exploration-exploitation

- Всегда во время обучения применять случайную стратегию выбора действия (плюсы/минусы - ?)
- С вероятностью ε применяем случайную стратегию, в остальных случаях – жадную (ε-жадная стратегия)
- Генерировать действие случайно в соответствии с вероятностным распределением π(a|s), которое "пропорционально" значеням Q-функции

Большая размерность пространства состояний

- Аппроксимируем Q*(s,a) моделью машинного обучения, зависящей от параметров θ
- Нейросети использовать выгодней всего т.к. метод стохастического градиента тренировки нейросети практически совпадает с методом простой итерации для уравнения Белмана

$$Q^*_{k+1}(s,a) \leftarrow (1-\alpha)Q^*_k(s,a) + \alpha(r + \gamma \max_{a'} Q^*_k(s',a')) =$$

$$= Q^*_k(s,a) + \alpha(r + \gamma \max_{a'} Q^*_k(s',a') - Q^*_k(s,a))$$

Связь с методом СГ

- Обозначим через "y" Белмановский таргет и "забудем" про его зависимость от θ:

$$y = r + \gamma \max_{a'} Q^*(s', a', \theta)$$

- тогда метод простои итерации практически совпадет с методом СГ:

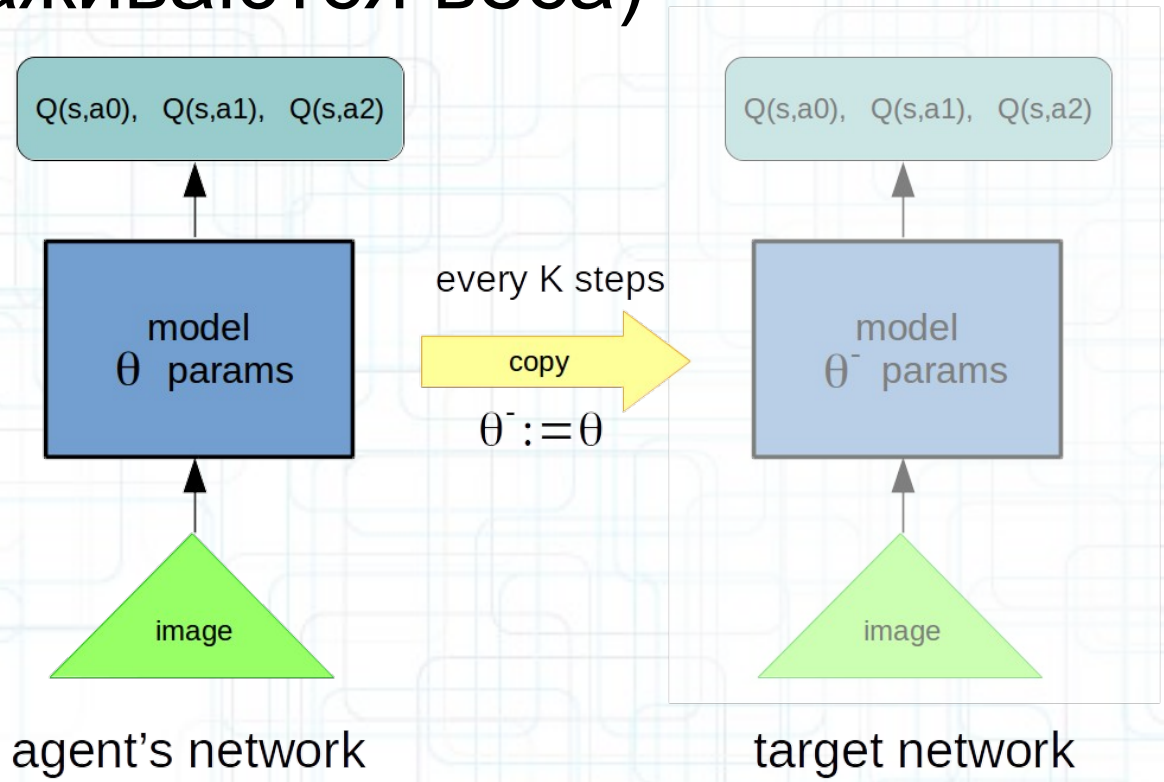
$$\begin{aligned}\theta_{k+1} \leftarrow & \theta_k - \alpha \nabla_\theta (y - Q^*(s, a, \theta))^2 = \\ = & \theta_k + 2\alpha (y - Q^*(s, a, \theta)) \nabla_\theta Q^*(s, a, \theta) = \\ = & \theta_k + 2\alpha (r + \gamma \max_{a'} Q^*(s', a', \theta) - Q^*(s, a, \theta)) \nabla_\theta Q^*(s, a, \theta)\end{aligned}$$

Общие замечания

- Целевая переменная "y" (Белмановский таргет) зависит от нашей же собственной модели. Во время обучения приходится использовать приближенную модель. Такое переиспользование объектов называется бутстрапирование
- На практике процесс обучения очень неустойчив: неадекватные значения начальной Q влекут неадекватное обновление весов нейросети, аппроксимирующей Q, и начинается цепная реакция

Стабилизация процесса обучения

- Использовать две нейросети: сеть агента и сеть для аппроксимации Белмановского таргета (раз в K шагов копируется сеть агента и замораживаются веса)



Experience Replay

- Для эффективной тренировки нейросетей нужен батч из переходов (s,a,r,s')

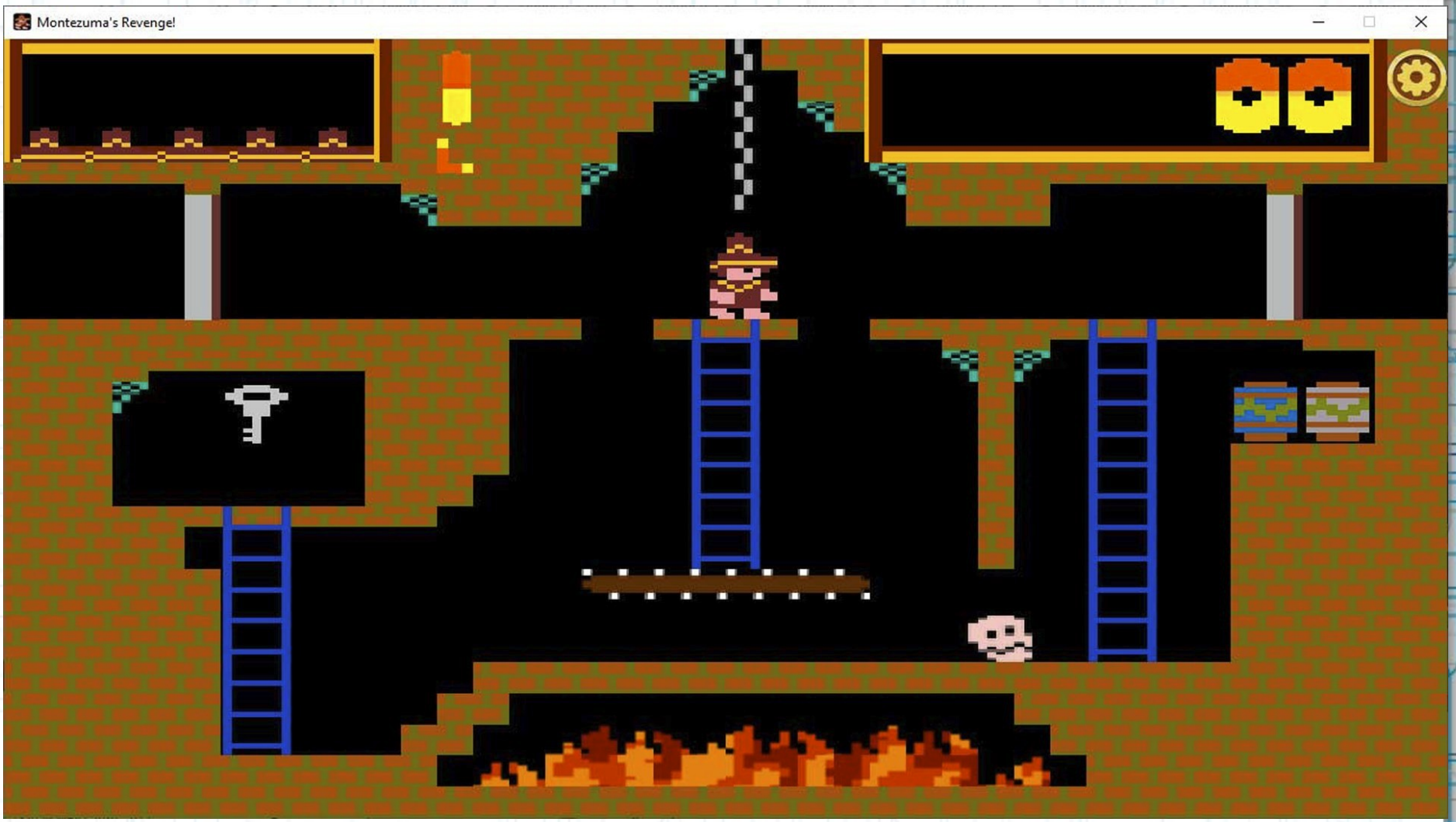
- Можно, не обновляя стратегию, нагенерировать переходов и сохранить их в буфер

- Target/agent нейросети + Replay Buffer =
  = Deep Q-learning

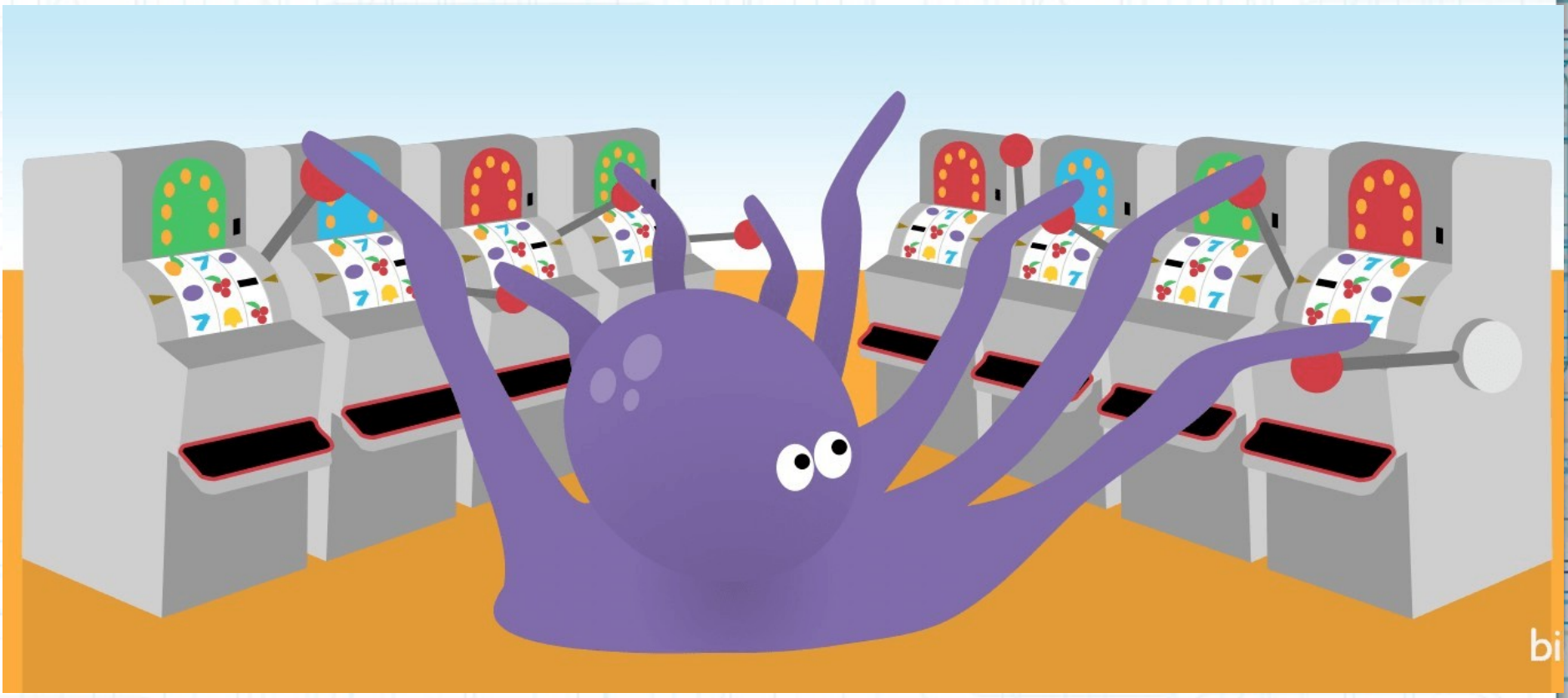
Проблемы сходимости

- Во многих ситуациях награждение дается только за особую последовательность действий, например, победа в игре/прохождение уровня (**Sparse Rewards**)
- Иногда стоит задача минимизации числа шагов процесса обучения
- Когда награда является **стохастической**, найти наилучшую стратегию тоже очень трудно
- Выполняя случайные действия начальной стратегии, агент может никогда не получать награду в случае Sparse Rewards
- Если агент все время действовал правильно, но в конце ошибся, то это дискредитирует все его предыдущие действия

Пример Sparse Rewards - Montezuma’s revenge



Пример стохастических наград: Multi-armed bandit



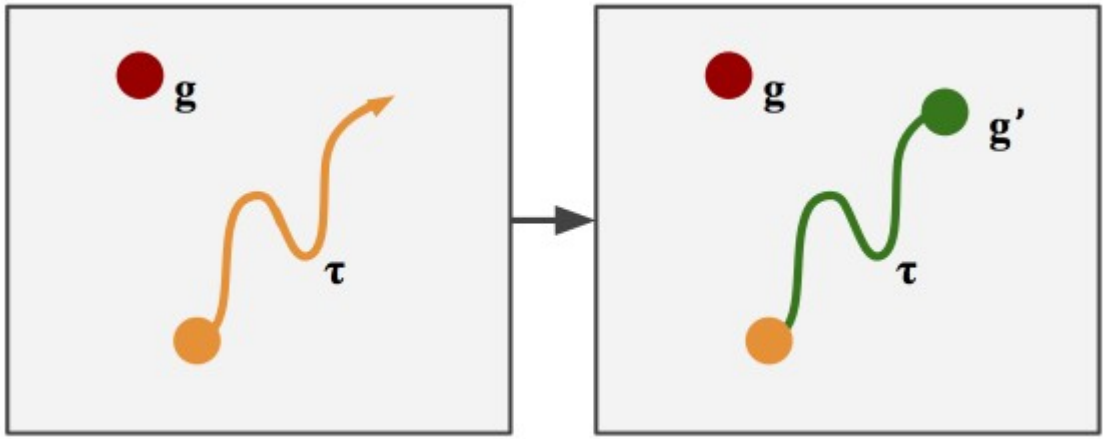
Применение в медицине: лечение пациентов одним из N лекарств. Какое лучше? Пациенты выздоравливают с некоторой вероятностью (разной для каждого лекарства).
Чем меньше пациентов мы "потратим" на эксперименты, тем лучше.

Эвристика: Reward reshaping

- **Reward Chipping** (дробление вознаграждения) – выдача небольших наград за успешные частичные последовательности действий
- **Auxiliary Tasks** (дополнительные задачи) – комбинирование максимизации редкой награды, например, с "пониманием" последствий наших действий или/и с предсказанием получения награды
- **Hindsight Experience Replay** – учиться достигать произвольную цель, а не только заданную. После обучения задать нужную цель и добраться до нее

Hindsight Experience Replay

- Для задач с бинарной наградой и "целью", которая может быть достигнута (r=1), а может – нет (r=0)
- Когда, случайно двигаясь, достичь цель невозможно, то агент не получает положительной награды, и метод не учится
- Выход: "разбавлять" обучающую выборку положительными наградами с неверной целью, достигнутой в конечном состоянии



Проблема в Exploration

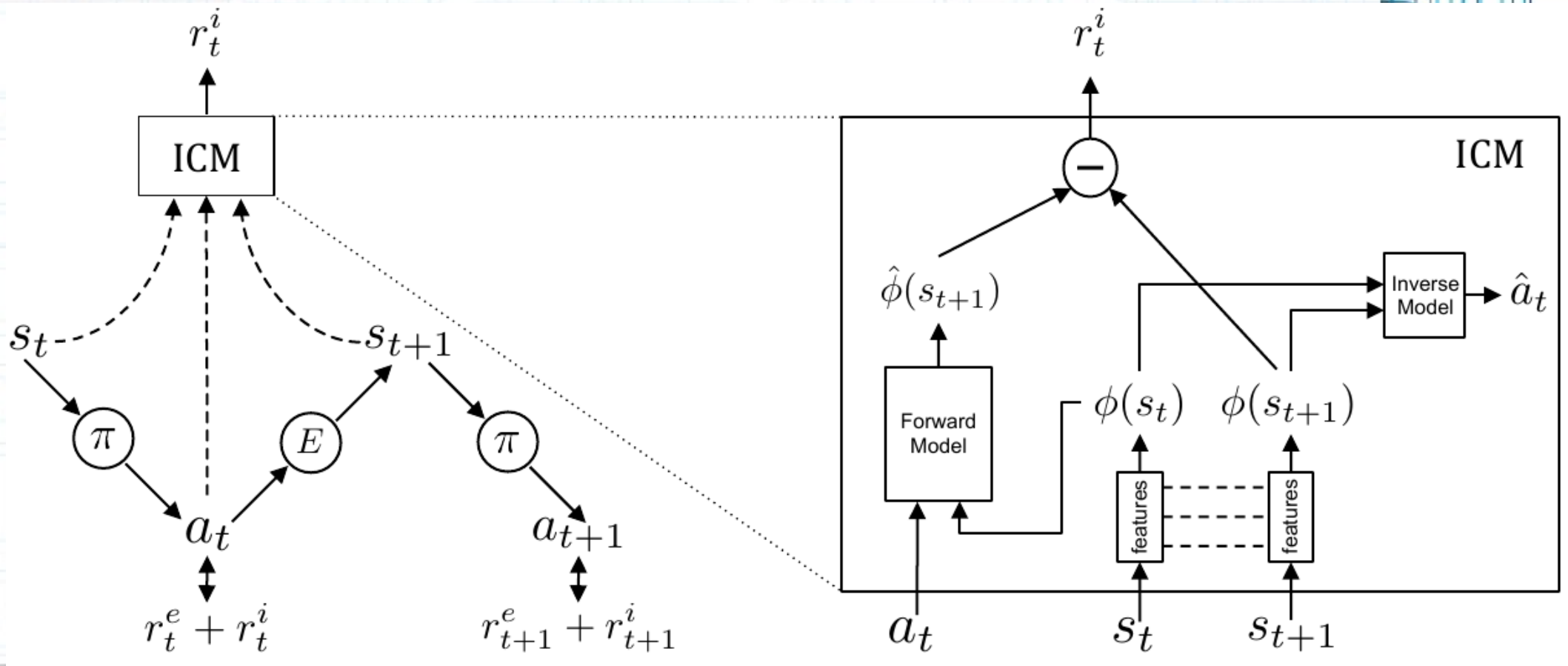
- Степень исследования пространства состояний (exploration) правильно определять как разность*(-1) между средними награждениями в соответствии с оптимальной стратегией и в соответствии с приближенной
- Редкие награждения приводят к очень низкому exploration даже после большого количества итераций
- Т.о. низкое exploration = медленная сходимость

Эвристики для повышения Exploration

- Раньше мы с вероятностью ε применяли случайную стратегию, но этот подход требует большого ε и не возвращается со временем к старой награде (т.е. найденная оптимальная стратегия не будет оптимальным решением исходной задачи)
- Подсчет **частот** посещенных состояний (для дискретного случая). Меняем награду так, чтобы посетить все состояния r += 1/sqrt(N(s))
- **Curiosity Driven Exploration** – агенту любопытно исследовать те состояния, о которых ему наперед ничего не известно

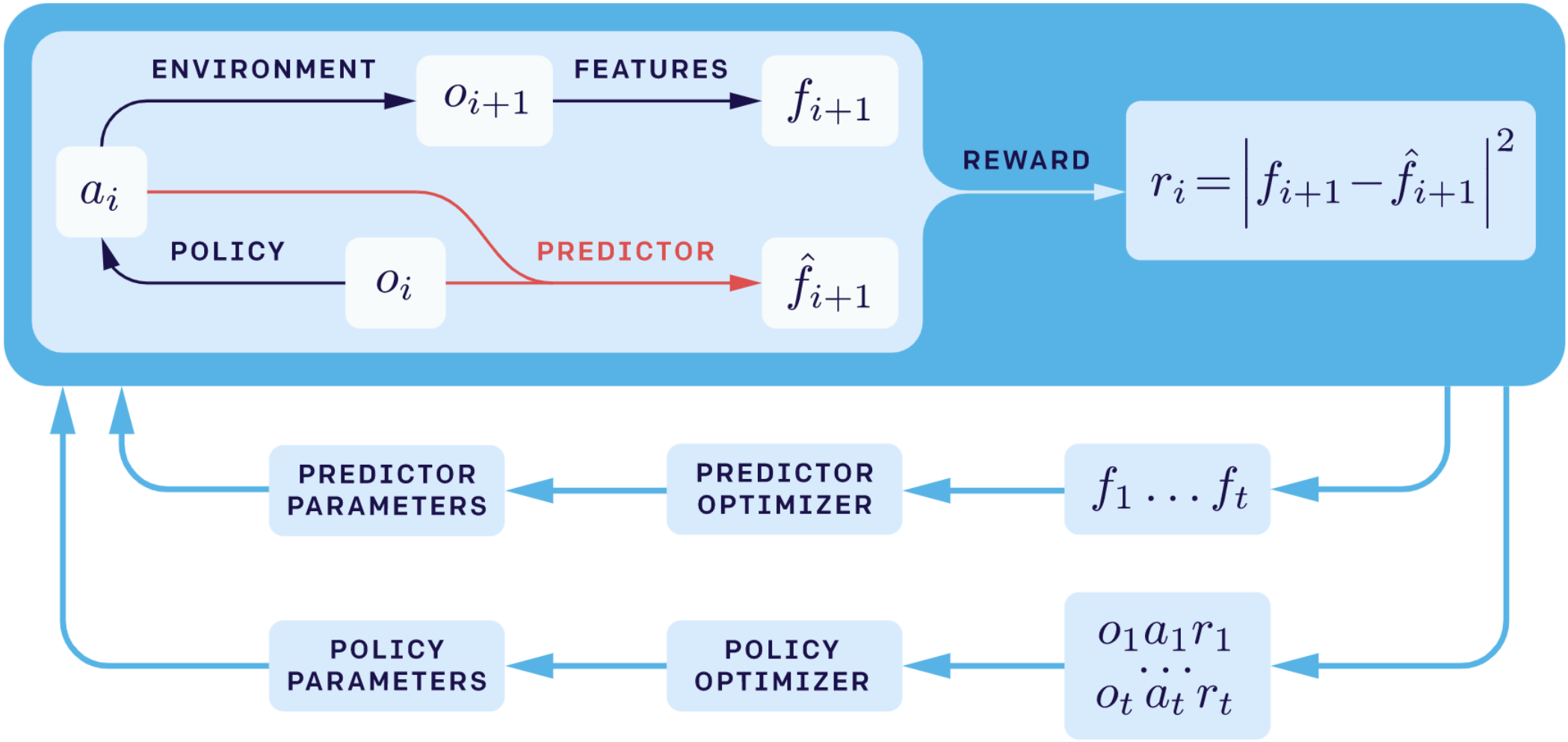
Curiosity Driven Exploration

- Forward – предсказывает следующее состояние и оценивает "любопытство"
- Вложение ϕ и Inverse удаляют из состояния лишнюю информацию, на которую мы не можем повлиять



Random Network Distillation

- Inverse model в ICM не работает для стохастических переходов
- Выход - использовать 2 нейронных сети: фиксированную случайную и ее аппроксимацию



Случай стохастической награды

- А что если функция награды тоже случайна? Exploration состояний тут уже не обойтись
- В простейшем примере многорукого бандита, где всего одно состояние, чем больше экспериментов мы проводим, тем точнее знаем вероятности выигрышей.
- Какое вероятностное распределение имеет величина вероятности выигрыша?

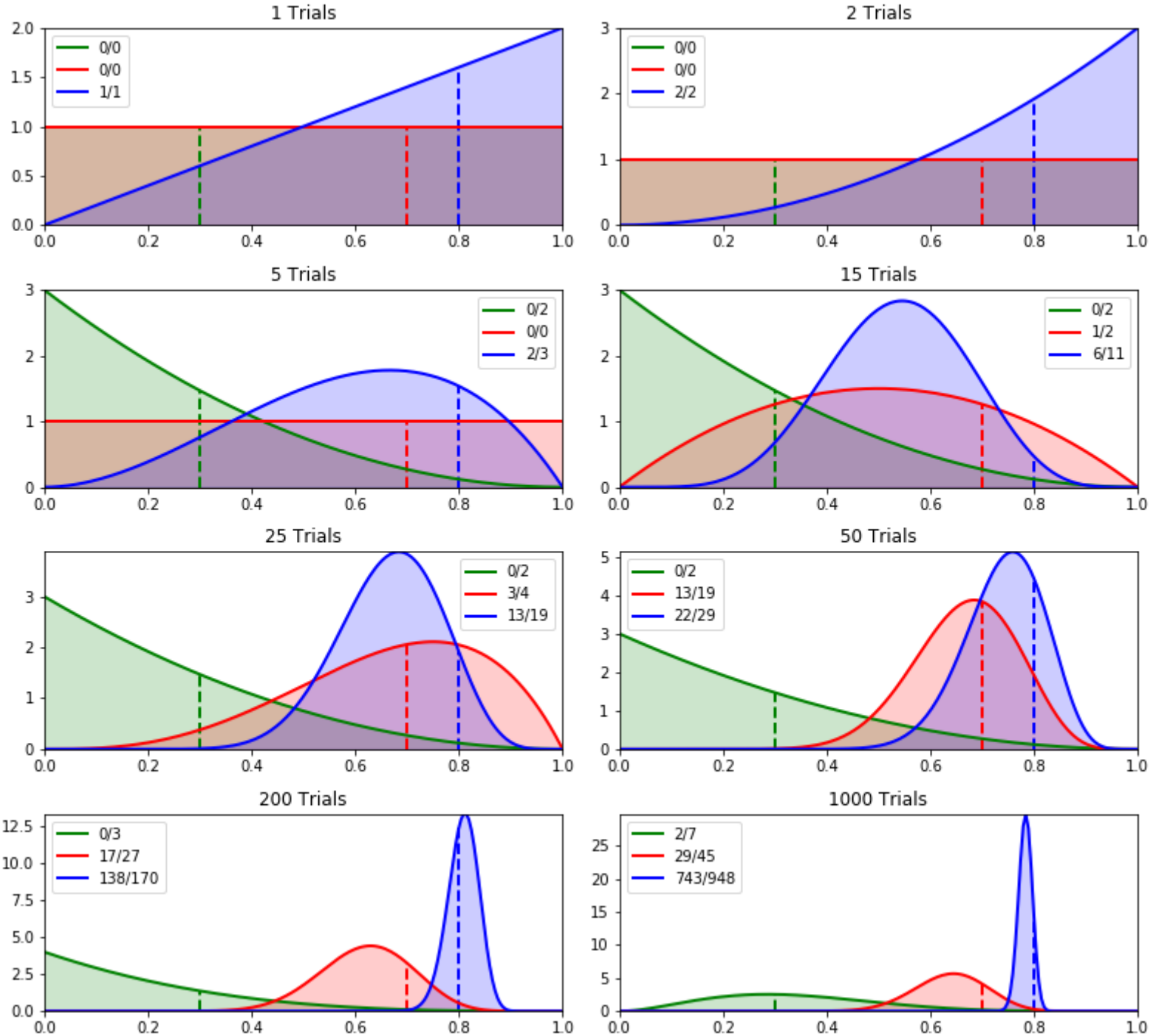
Случай стохастической награды

- Ответ: бета-распределение с параметрами α – число выигрышей, β – число неудач

$$f_X(x) = \frac{1}{\mathrm{B}(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

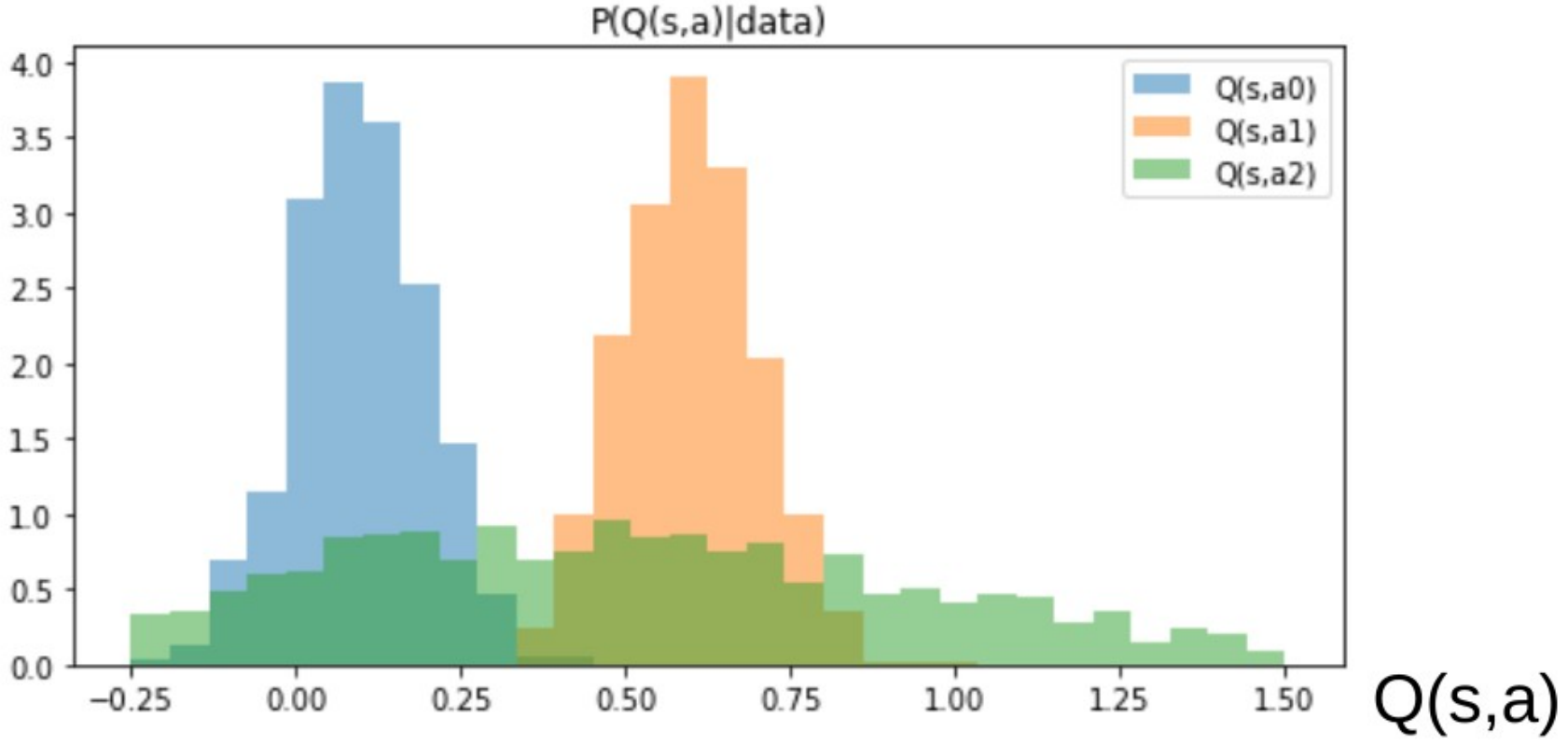
- Стратегия: генерируем числа из бета-распределений бандитов и выбираем наибольшее
- Bayesian Inference: обновляем распределения в процессе игры

Бета-распределение



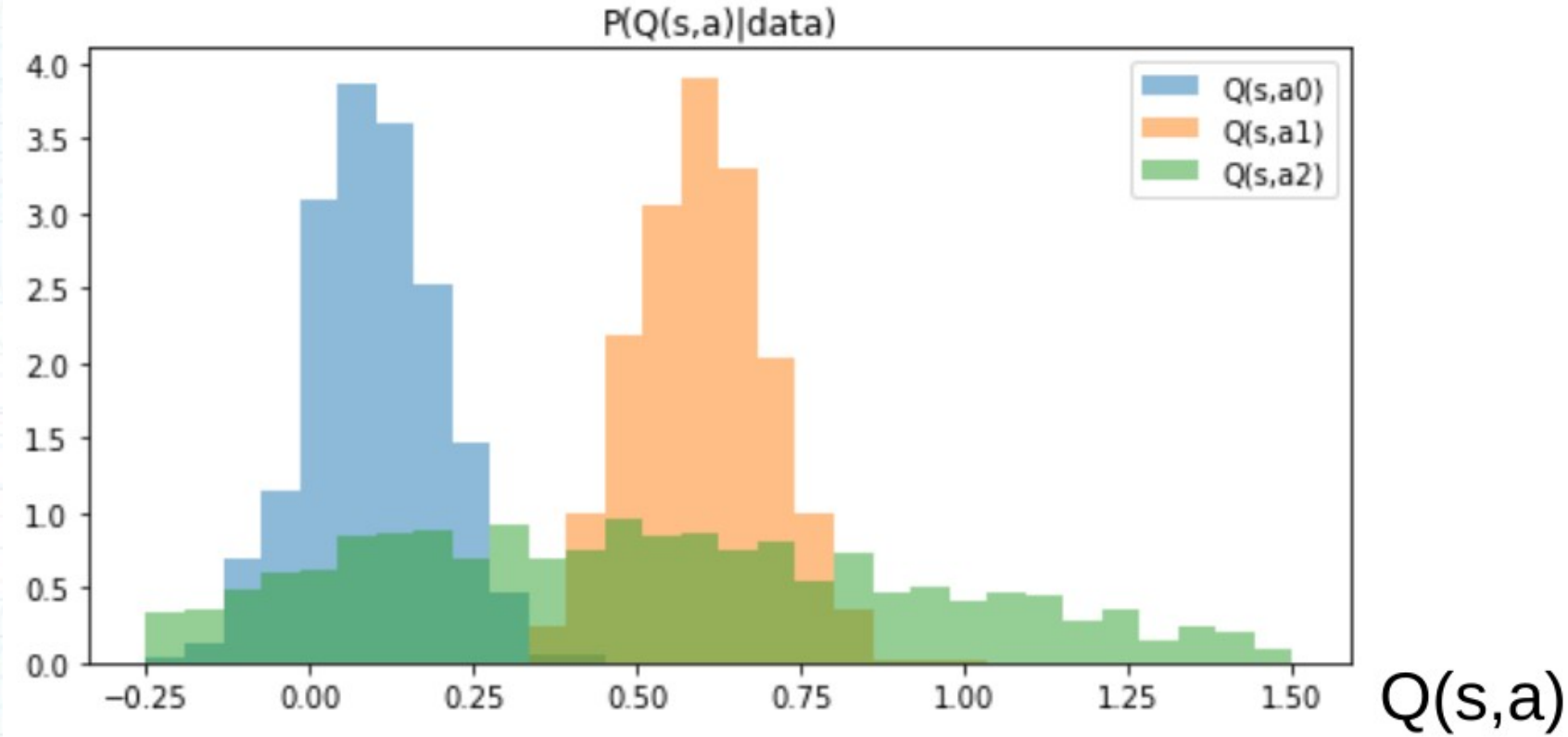
Случай стохастической награды

- Какое действие выбрать в случае произвольных распределений? Использовать сопряженное априорное распределение?



Случай стохастической награды

- Какое действие выбрать в случае произвольных распределений? Использовать сопряженное априорное распределение?



- Стратегия: вычислим 95% квантиль каждого распределения и выберем наибольший

Частотный анализ состояний

Стратегия **UCB** (Upper Confidence Bound):
Пусть $n_{s,a}$ – количество раз, которое мы попробовали действие a в состоянии s,
Q(s,a) – средний выигрыш от действия a в состоянии s,
N_s – число раз посещения состояния s

Будем придерживаться жадной стратегии, но для измененной Q-функции:

$$\tilde{Q}(s,a) = Q(s,a) + \alpha \cdot \sqrt{\frac{2 \log N_s}{n_{s,a}}}$$