Многорукие бандиты

Рословец Влад

Что такое RL?

Раздел машинного обучения, изучающий поведение агента, максимизирующее некоторый выигрыш, в среде.

- Поиск оптимальной стратегии
- При обучении нет пар типа «Данные-ответ»

Постановка задачи

1. Множество состояний окружений S
2. Множество действий A
3. Множество вещественных выигрышей r
4. Стратегия в определенный момент времени $p_t(a)$

Является Марковским процессом принятия решений

Марковкий процесс принятия решений

Кратко о Марковских процессах:

Марковский процесс принятия решений задается кортежем из 4-х значений:

S – конечное множество состояний

A – конечное множество действий (часто представляется в виде множества A_s, доступных из состояния s)

P_a(s, s’) = Pr(s_t+1 = s’ | s_t = s, a_t = a) – вероятность, что действие a в состоянии s во время t приведет в состояние s’ ко времени t+1

R_a(s, s’) – вознаграждение, получаемое после перехода в состояние s’ из s с вероятностью перехода P_a(s, s’)

Игра агента со средой

1. Инициализация стратегии $p_1(a)$
2. Для всех $t = 1 \ldots \mathcal{T} \ldots$
    1. Агент выбирает действие $a \sim p_t(a)$;
    2. Среда генерирует премию r
    3. Агент корректирует стратегию $p_{t+1}(a)$

Exploration vs Exploitation

Проблема заключается в том, что агент должен не только максимизировать прибыль на каждом шаге, но и исследовать среду.

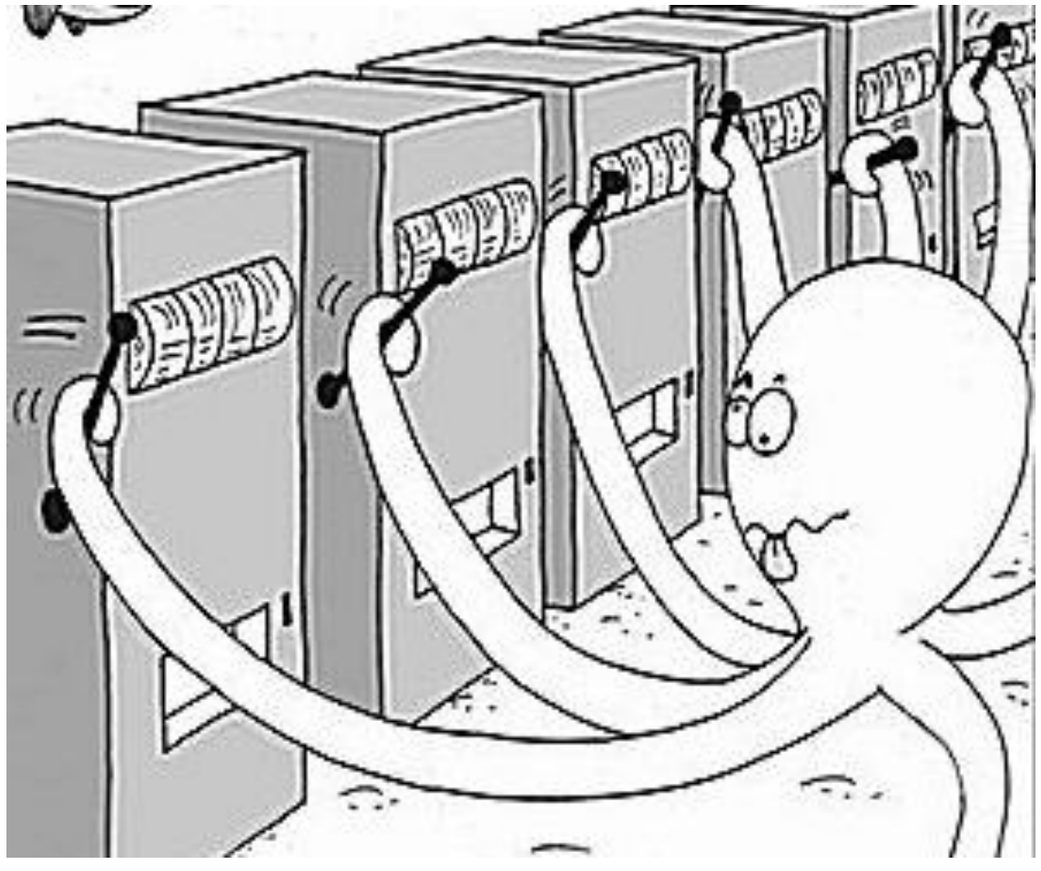
Оптимальная стратегия может отличаться от эксплуатируемой в текущем состоянии.

Применение RL

Примеры:

- Показ рекламы в интернете
- Игра на бирже
- Управление ценами в ретейле
- Логические игры(Go, шахматы, Dota)
- Тестирование

Многорукие бандиты



Многорукие бандиты

Модель: агент в комнате с несколькими игровыми автоматами. У каждого автомата своё ожидание выигрыша. Нужно за ограниченное количество попыток выбрать лучший автомат.

Супер жадный алгоритм

Всегда выбирать действие, максимизирующие прибыль, которую оцениваем как среднее вознаграждение:

$$a = argmax_{a \in A}(\frac{1}{k_a} \sum_{i=1}^{k_a} r_i)$$

Вопрос

Почему жадный алгоритм плох и как его можно улучшить?

ϵ - жадный алгоритм

- Вводим параметр $\epsilon \in (0,1)$
- С вероятностью $1 - \epsilon$ выбираем действие с максимальной оценкой математического ожидания, иначе выбираем случайное другое действие.
- Обычно $\epsilon = 0.1$

$\epsilon - deacreasing$

Аналогично предыдущему алгоритму. Однако теперь мы можем уменьшать с течением времени значение ϵ.
Тем самым увеличивая regret.

Минус жадных алгоритмов: алгоритмы не различают хорошую альтернативу и бесполезную.

Алгоритм UCB1

Upper Confidence Bounds

Рассчитываем приоритет каждого действия по формуле:

$$priority_i = \hat{\mu}_i + \sqrt{\frac{2lnt}{k_i}}$$

где $\hat{\mu}_i$ - средняя награда i-ого действия

t - кол-во сделанных ходов,

k_i - кол-во ходов с выбором i-ого действия

Выбираем действие с наивысшим приоритетом!

Алгоритм UCB1

Для нахождения оценки воспользуемся неравенством Чернова.

Пусть $\hat{\mu} = \frac{1}{k}\sum_i r_i$ и $\mu = \mathbb{E}(\hat{\mu}) = \frac{1}{k}\sum_i \mu_i$

$$P(\hat{\mu} + z < \mu) \leq e^{-2kz^2}$$

Алгоритм UCB1

$$P(\hat{\mu} + z < \mu) \leq e^{-2kz^2}$$

r_a награда за действие a

$\hat{\mu}$ средняя награда за действие a за все время

z односторонняя верхняя оценка

Верхняя оценка полученная после решения неравенста.

$$z = z(a,t) = \sqrt{\frac{2lnt}{k_a}}$$

Утверждается, что данного значения хватает, чтобы быть уверенным, что мы находимся в пределах истинного значения

Сэмплирование Томпсона

- Бета распределение является априорно сопряженным к распределению Бернулли

$$Beta\left(\alpha + y, \beta + 1 - y\right) = Beta\left(\theta \mid \alpha, \beta\right) \cdot Bernoulli\left(y \mid \theta\right)$$

- При $\alpha + \beta = 1$бета-распределение принимает форму равномерного распределения, т.е. именно такого, которое естественно использовать в ситуации полной неопределенности (например, в самом начале тестирования); чем более определенными становятся наши ожидания относительно прибыльности бандита, тем более скошенным становится распределение (влево — не прибыльный бандит, вправо — прибыльный);

- Модель интерпретируется, т.е. α - кол-во успешных испытаний, β - кол-во неудачных испытаний.

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

Сэмплирование Томпсона

Algorithm 1 Thompson Sampling for Bernoulli bandits

For each arm $i = 1, \ldots, N$ set $S_i = 0, F_i = 0$.

foreach $t = 1, 2, \ldots,$ **do**

  For each arm $i = 1, \ldots, N$, sample $\theta_i(t)$ from the Beta$(S_i + 1, F_i + 1)$ distribution.

  Play arm $i(t) := \arg\max_i \theta_i(t)$ and observe reward r_t.

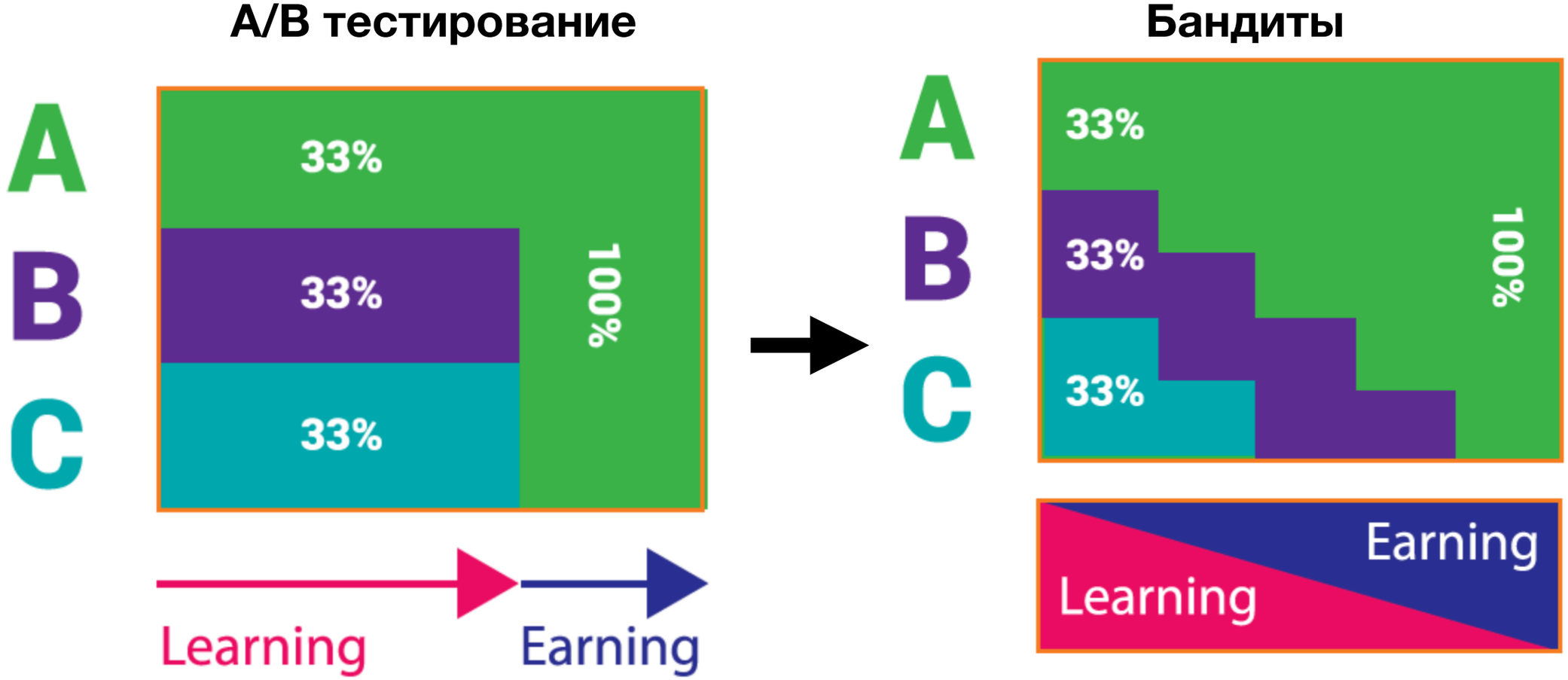
  If $r = 1$, then $S_{i(t)} = S_{i(t)} + 1$, else $F_{i(t)} = F_{i(t)} + 1$.

end

S_i, F_i - значение параметров бета распределения для i - ого действия

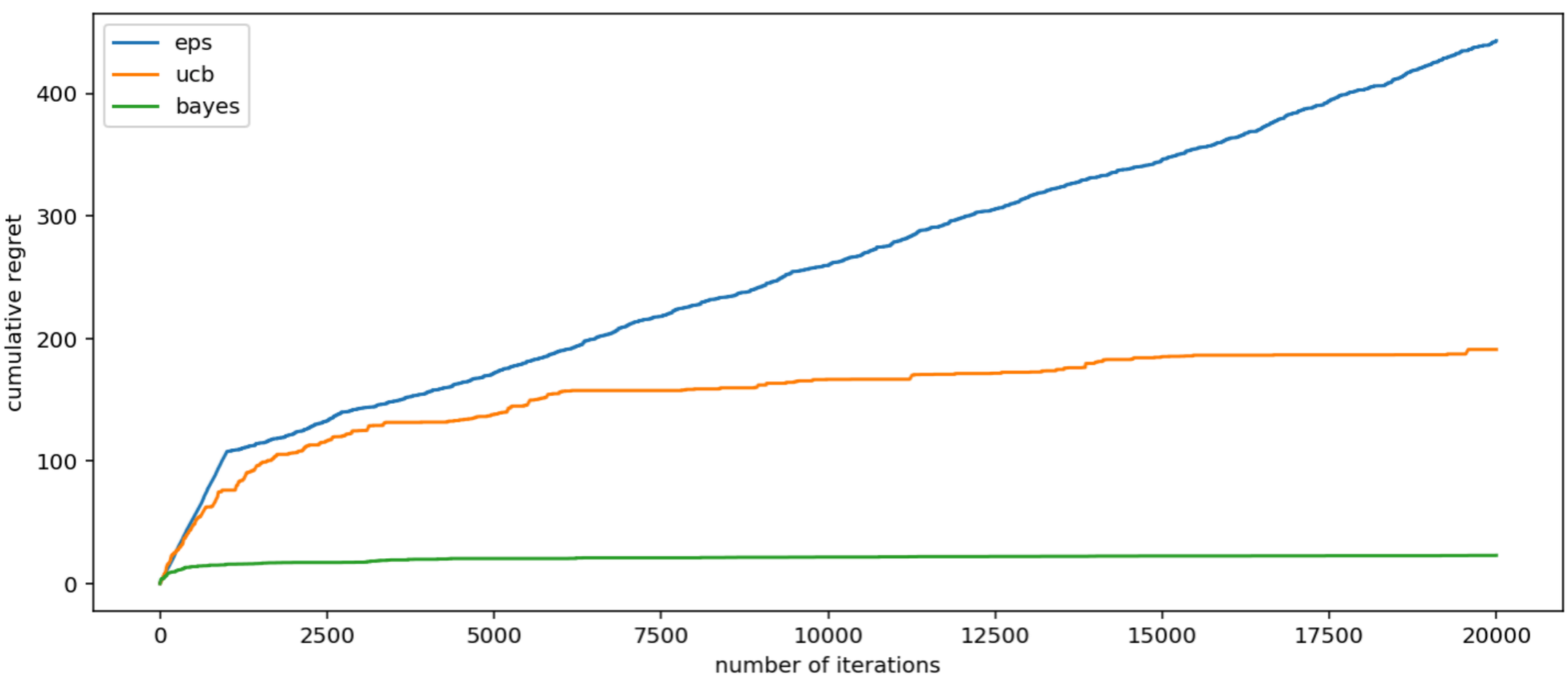
$\theta_i(t)$ - значение сгенерированное бета распределением i - ого действия

Зачем бандиты, если есть A/B тесты



Сравнение методов

Пример для 4 распределений Бернулли с параметрами p = 0.3, 0.5, 0.6, 0.7



Вопросы

1. В чем заключается проблема жадного алгоритма?
2. Записать формулу приоритета действия для алгоритма UCB.
3. Написать алгоритм семплирования Томпсона.

Источники

Статья по семплированию Томпсона

Статья по UCB1

Статья на wiki про сопряжённое априорное распределение

Байесовские многорукие бандиты с Habr-a

Видеоряд «обучение с подкреплением - К.В. Воронцов»