МНОГОРУКИЕ БАНДИТЫ

Сергей Николенко

Академия MADE — Mail.Ru

15 сентября 2021 г.

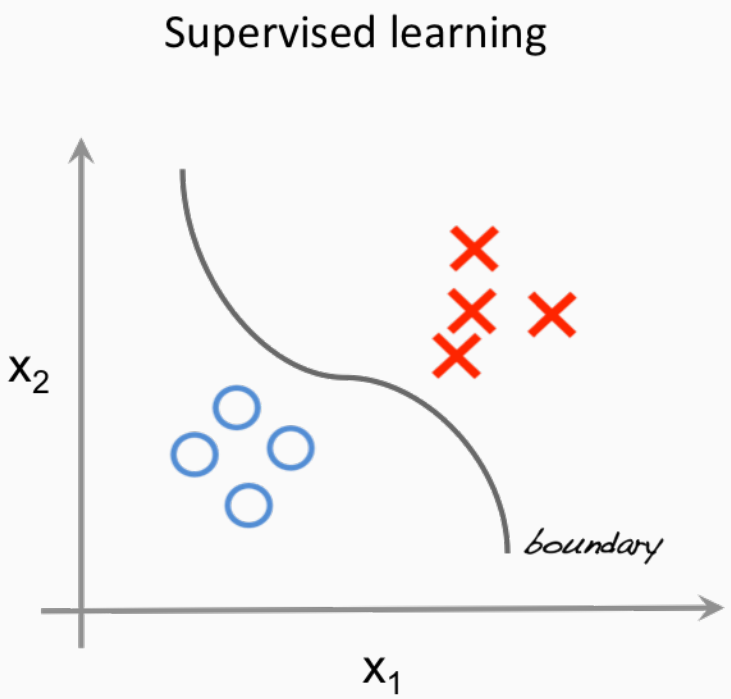
Random facts:

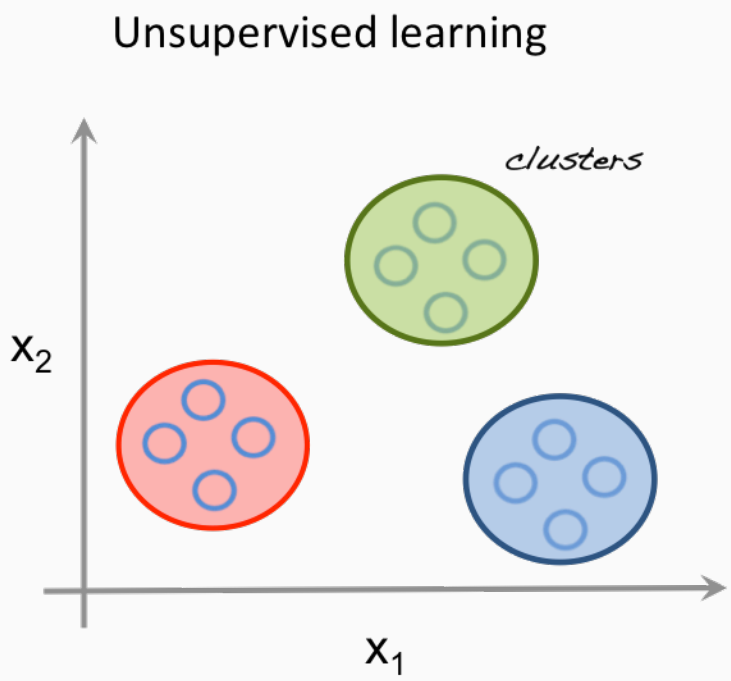
- 15 сентября 1776 г. английские войска оккупировали Нью-Йорк и почти захватили Джорджа Вашингтона; Вашингтон пытался остановить бегущих, но в итоге швырнул на землю свою шляпу и воскликнул: «И вот с этими людьми я должен защищать Америку!»
- 15 сентября 1910 г. первые парламентские выборы в Южной Африке выиграли белые националисты; далеко не в последний раз, конечно
- 15 сентября 1915 г. государь император распустил Государственную думу на каникулы без указания даты возобновления заседаний
- 15 сентября 1933 г. Генрих Ягода сообщил Сталину о раскрытии «общества педерастов» в Ленинграде; было арестовано более 150 человек
- 15 сентября 1935 г. по Нюрнбергским законам немецкие евреи были лишены гражданства, а государственным символом Германии стала свастика
- В результате 15 сентября стало Международным днём демократии, провозглашённым Генеральной Ассамблеей ООН в 2007 году; тема 2021 года звучит как «Strengthening democratic resilience in the face of future crises»; так что, видимо, все на выборы!..

Обучение с подкреплением

Постановка задачи

- В машинном обучении задача обычно ставится так:
  - или есть набор «правильных ответов», и нужно его продолжить на всё пространство (supervised learning),
  - или есть набор тестовых примеров без дополнительной информации, и нужно понять его структуру (unsupervised learning).



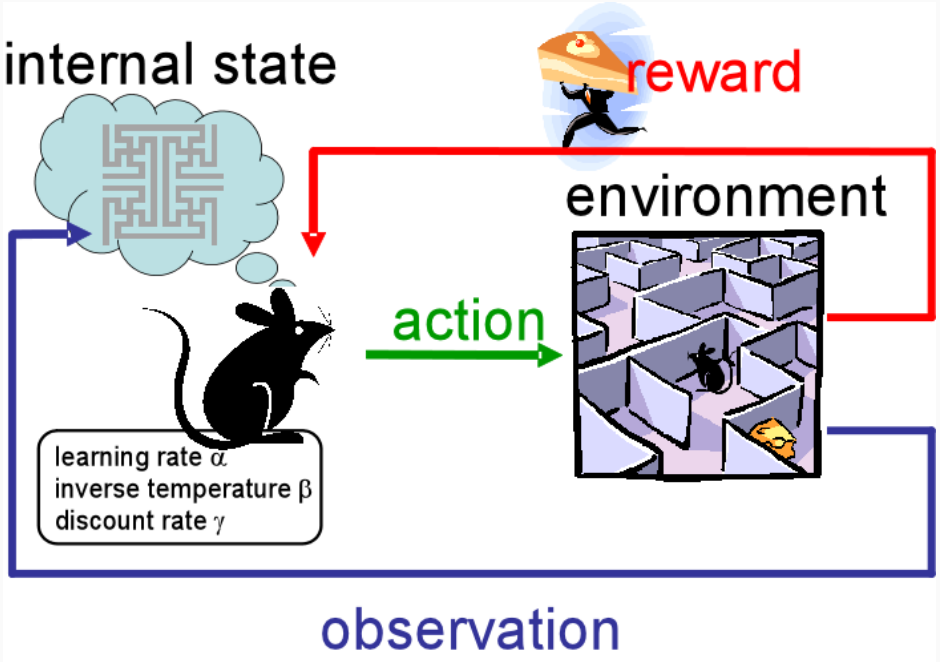


Постановка задачи

- Но как работает обучение в реальной жизни?
- Как ребёнок учится ходить?
- Мы далеко не всегда знаем набор правильных ответов, мы просто делаем то или иное действие и получаем результат.

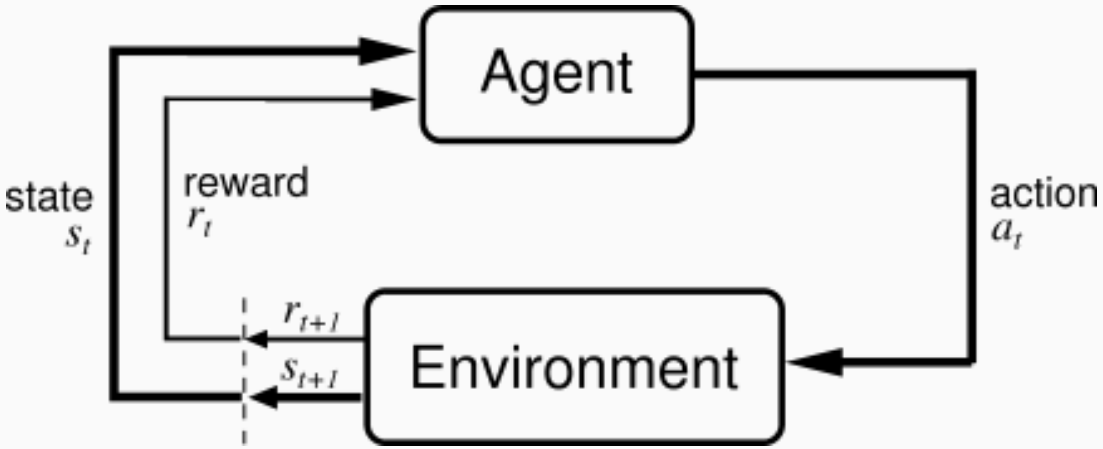
Постановка задачи

- Отсюда и обучение с подкреплением (reinforcement learning).
- Агент взаимодействует с окружающей средой, предпринимая действия; окружающая среда его поощряет за эти действия, а агент продолжает их предпринимать.



Постановка задачи -- формально

- На каждом шаге агент может находиться в состоянии $s \in S$.
- На каждом шаге агент выбирает из имеющегося набора действий некоторое действие $a \in A$.
- Окружающая среда сообщает агенту, какую награду r он за это получил и в каком состоянии s' после этого оказался.



- (Sutton, Barto, 1998)

Пример

- Диалог:

Среда: Агент, ты в состоянии 1; есть 5 возможных действий.
Агент: Делаю действие 2.
Среда: Даю тебе 2 единицы за это. Попал в состояние 5, есть 2 возможных действия.
Агент: Делаю действие 1.
Среда: Даю тебе за это -5 единиц. Попал в состояние 1, есть 5 возможных действий.
Агент: Делаю действие 4.
Среда: Даю тебе 14 единиц за это. Попал в состояние 3, есть 3 возможных действия...

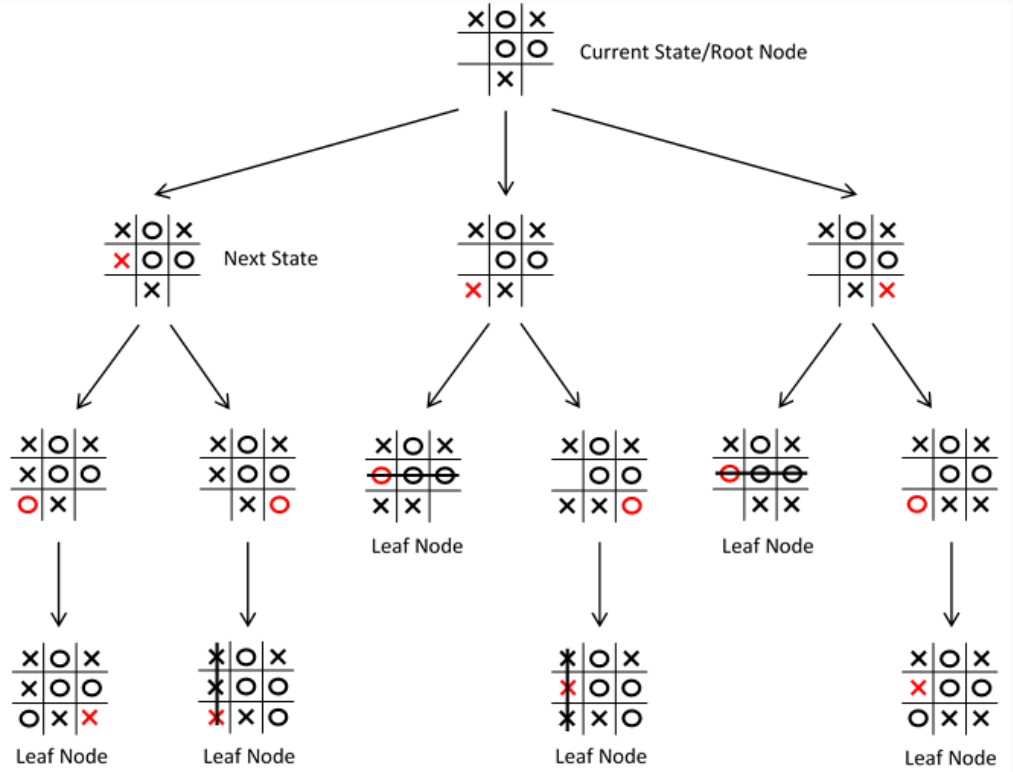
- В этом примере агент успел вернуться в состояние 1 и исследовать ранее не пробовавшуюся опцию 4 (получив за это существенную награду).

Exploitation vs. exploration

- Каждый алгоритм должен и изучать окружающую среду, и пользоваться своими знаниями, чтобы максимизировать прибыль.
- Вопрос — как достичь оптимального соотношения? Та или иная стратегия может быть хороша, но вдруг она не оптимальная?
- Этот вопрос всегда присутствует в обучении с подкреплением.

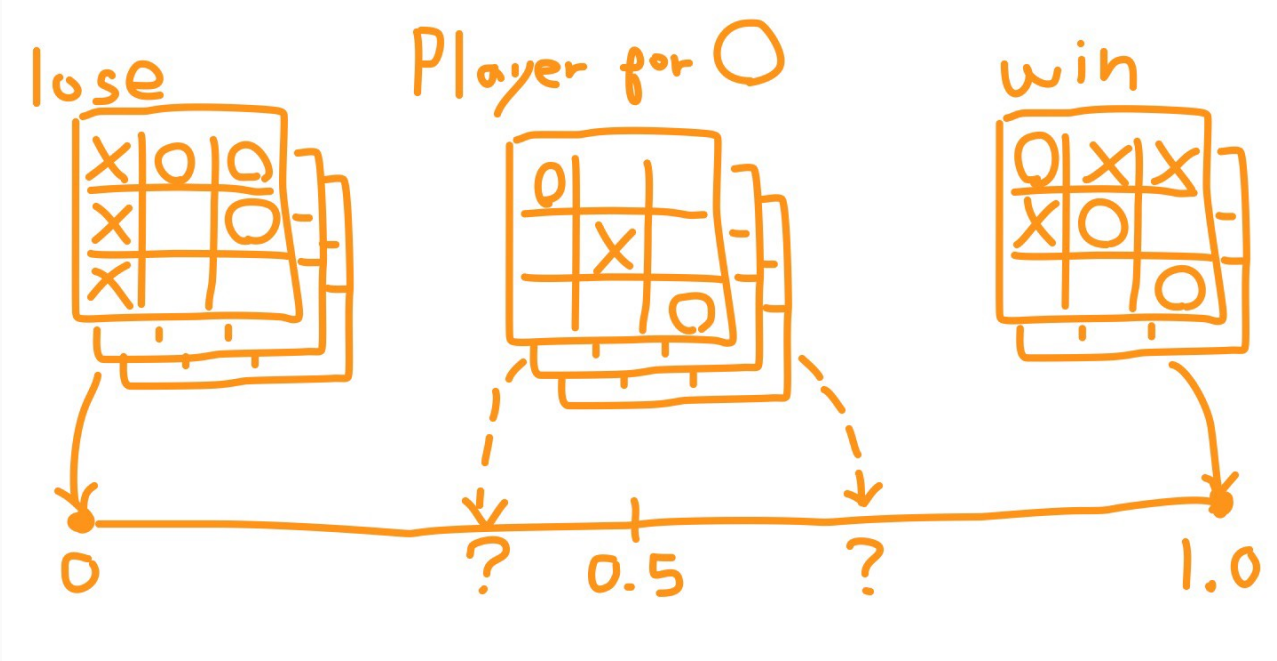
Пример

- Пример: крестики-нолики. Как научить машину играть и *выигрывать* в крестики-нолики?
- Можно, конечно, дерево построить, но это не масштабируется.



Пример

- Состояния – позиции на доске.
- Для каждого состояния введём функцию $V(s)$ (value function).
- Подкрепление приходит только в самом конце, когда мы выиграли или проиграли; как его распространить на промежуточные позиции?



Пример

- Небольшой трейлер того, что будет дальше: можно пропагировать оценку позиции обратно.
- Если мы попадали из s в s', апдейтим

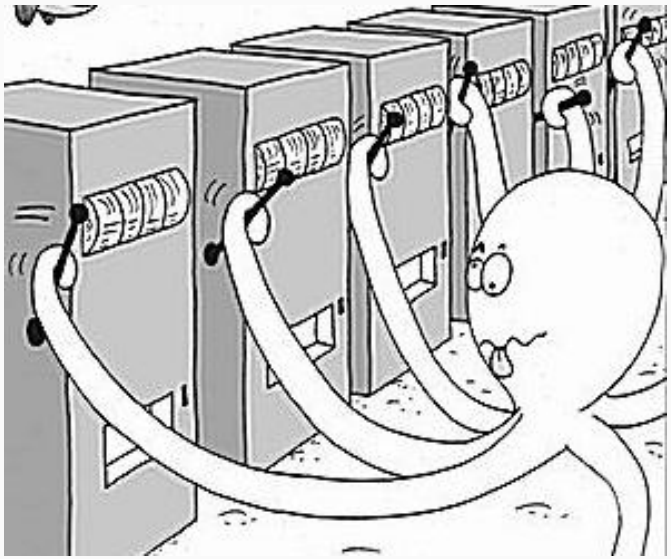
$$V(s) := V(s) + \alpha \left[V(s') - V(s) \right].$$

- Это называется TD-обучение (temporal difference learning), оно очень хорошо работает на практике и лежит в основе и AlphaGo, и много чего ещё.
- Но это будет ещё не сегодня..

Многорукие бандиты

Агенты с одним состоянием

- Формально всё то же самое, но $|S| = 1$, т.е. состояние агента не меняется. У него фиксированный набор действий A и возможность выбора из этого набора действий.
- Модель: агент в комнате с несколькими игровыми автоматами. У каждого автомата своё ожидание выигрыша.
- Нужно заработать побольше: exploration vs. exploitation.



Жадный алгоритм

- *Жадный алгоритм*: всегда выбирать стратегию, максимизирующую прибыль; прибыль можно оценить как среднее вознаграждение, полученное от этого действия:

$$Q_t(a) = \frac{r_1 + r_2 + ... + r_{k_a}}{k_a}.$$

- Что не так с таким алгоритмом?

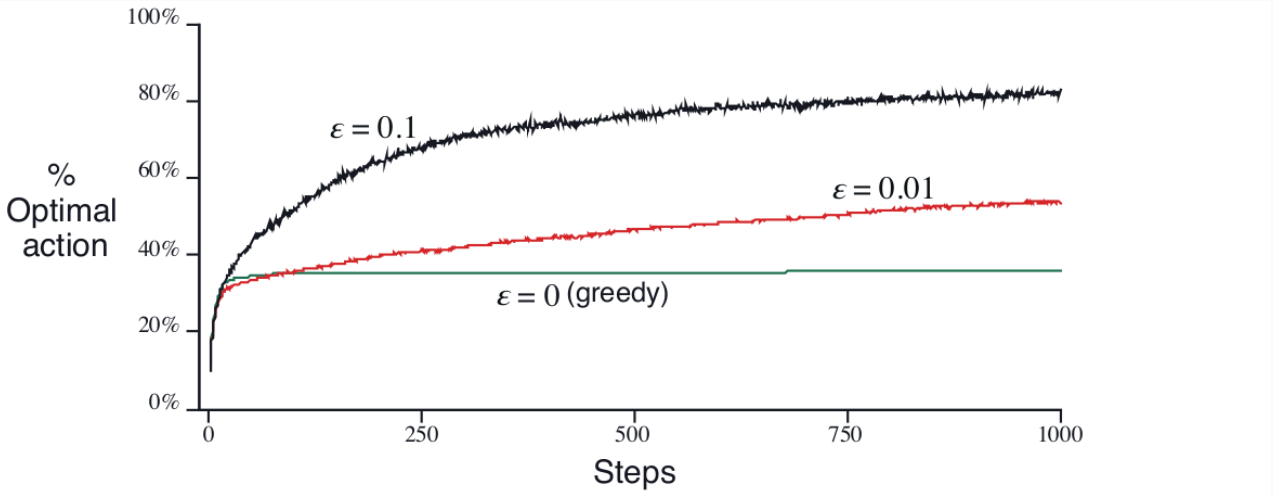
Жадный алгоритм

- Оптимум легко проглядеть, если на начальной выборке не повезёт (что вполне возможно).
- Поэтому полезная эвристика — *оптимизм при неопределённости*.

- То есть выбирать жадно, но при этом прибыль оценивать оптимистично, и нужны серьёзные свидетельства, чтобы отклонить стратегию.

Случайные стратегии

- *ϵ-жадная стратегия* (ϵ-greedy): выбрать действие с наилучшей ожидаемой прибылью с вероятностью $1 - \epsilon$, а с вероятностью ϵ выбрать случайное действие.



- Обычно начинают с больших ϵ, затем уменьшают.
- Алгоритм не отличает хорошую альтернативу от бесполезной, но всё равно разумный, про него много чего доказать можно.

Интервальные оценки

- Естественный способ применить оптимистично–жадный метод – доверительные интервалы.
- Для каждого действия мы храним статистику n и w, а потом вычисляем доверительный интервал для вероятности успеха (с границей $1 - \alpha$) и для выбора стратегии используем *верхнюю границу* этого интервала.
- Например, для испытаний Бернулли (монетка) с вероятностью $.95$ среднее лежит в интервале

$$\left(\bar{x} - 1.96\frac{s}{\sqrt{n}}, \bar{x} + 1.96\frac{s}{\sqrt{n}}\right),$$

где 1.96 берётся из распределения Стьюдента, n — количество испытаний, $s = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$.

Правило инкрементального обновления

- Теперь – о среднем и $Q_t(a)$.
- Как пересчитывать $Q_t(a) = \frac{r_1 + \ldots + r_{k_a}}{k_a}$ при поступлении новой информации?
- Довольно просто:

$$Q_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} r_i = \frac{1}{k+1} \left[r_{k+1} + \sum_{i=1}^{k} r_i \right] =$$
$$= \frac{1}{k+1} \left(r_{k+1} + kQ_k \right) = Q_k + \frac{1}{k+1} \left(r_{k+1} - Q_k \right).$$

Правило инкрементального обновления

- Это частный случай общего правила – сдвигаем оценку так, чтобы уменьшалась ошибка:

  НоваяОценка := СтараяОценка + Шаг [Цель − СтараяОценка] .

- Заметим, что шаг у среднего непостоянный, $\alpha_k(a) = \frac{1}{k_a}$:

$$Q_{k+1} = Q_k + \frac{1}{k+1}\left(r_{k+1} - Q_k\right).$$

- Изменяя последовательность шагов, можно добиться других эффектов.

Нестационарная задача

- Часто бывает, что выплаты из разных бандитов на самом деле нестационарны, т.е. меняются со временем.
- В такой ситуации имеет смысл давать большие веса недавней информации и маленькие веса – давней.
- Пример: у правила апдейта

$$Q_{k+1} = Q_k + \alpha \left[r_{k+1} - Q_k \right]$$

с постоянным α фактически веса затухают экспоненциально:

$$Q_k = Q_{k-1} + \alpha \left[r_k - Q_{k-1} \right] = \alpha r_k + (1-\alpha) Q_{k-1} =$$

$$= \alpha r_k + (1-\alpha)\alpha r_{k-1} + (1-\alpha)^2 Q_{k-2} = (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha (1-\alpha)^{k-i} r_i.$$

Нестационарная задача

- Такое правило апдейта не обязательно сходится, но это и хорошо – мы хотим следовать за целью.
- Общий результат – правило апдейта сходится, если последовательность весов удовлетворяет

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty \quad \text{и} \quad \sum_{k=1}^{\infty} \alpha_k^2(a) < \infty.$$

- Например, для $\alpha_k(a) = \frac{1}{k_a}$ явно сходится.

Стратегии, минимизирующие regret

Динамическое программирование

- Предположим, что агент действует на протяжении h шагов.
- Используем байесовский подход для определения оптимальной стратегии.
- Начинаем со случайных параметров $\{p_i\}$, например, равномерно распределённых, и вычисляем отображение из *belief states* (состояния после нескольких раундов обучения) в действия.
- Состояние выражается как $\mathcal{S} = \{n_1, w_1, \dots, n_k, w_k\}$, где каждого бандита i запустили n_i раз и получили w_i единичек (считаем, что результат бинарный).

Динамическое программирование

- $V^*(\mathcal{S})$ — ожидаемый оставшийся выигрыш.
- Рекурсивно: если $\sum_{i=1}^{k} n_i = h$, то больше нечего делать, и $V^*(\mathcal{S}) = 0$.
- Если знаем V^* для всех состояний, когда осталось t запусков, сможем пересчитать и для $t+1$:

$$V^*(n_1, w_1, \dots, n_k, w_k) = \\ = \max_i \left(\rho_i(1 + V^*(\dots, n_i + 1, w_i + 1, \dots)) + \right. \\ \left. (1 - \rho_i)V^*(\dots, n_i + 1, w_i, \dots)\right),$$

где ρ_i — апостериорные вероятности того, что действие i оправдается (если изначально p_i равномерно распределены, то $\rho_i = \frac{w_i+1}{n_i+2}$).

Байесовский подход к многоруким бандитам

- А теперь давайте посмотрим на многоруких бандитов в общем вероятностном виде.
- Для простоты – бинарный случай, выплата либо 1, либо 0.

Байесовский подход к многоруким бандитам

- Пусть во время t у нас состояние $\mathbf{s}_t = (s_{1t}, \dots, s_{Kt})$ для K ручек, и мы хотим дёрнуть такую ручку, чтобы максимизировать общее ожидаемое число успехов.
- Есть функция вознаграждения $R_i(\mathbf{s}_t, \mathbf{s}_{t+1})$ – награда за дёргание ручки i (a_i), которое переводит состояние $\mathbf{s}_t$ в $\mathbf{s}_{t+1}$.
- Есть вероятность перехода $p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_i\right)$.
- И мы хотим обучить стратегию $\pi(\mathbf{s}_t)$, которая возвращает, какую ручку дёргать.

Байесовский подход к многоруким бандитам

- Тогда value function в самом общем виде до горизонта T:

$$V_T(\pi, \mathbf{s}_0) = \mathrm{E}\left[R_{\pi(\mathbf{s}_0)}(\mathbf{s}_0, \mathbf{s}_1) + V_{T-1}(\pi, \mathbf{s}_1)\right] =$$
$$= \int p(\mathbf{s}_1 \mid \mathbf{s}_0, \pi(\mathbf{s}_0)) \left[R_{\pi(\mathbf{s}_0)}(\mathbf{s}_0, \mathbf{s}_1) + V_{T-1}(\pi, \mathbf{s}_1)\right] d\mathbf{s}_1.$$

- Если всё известно, и T невелико, то можно, опять же, динамическим программированием.
- Но даже подсчитать отдачу от фиксированной стратегии может быть очень дорого, не говоря уж об оптимизации.

Байесовский подход к многоруким бандитам

- Если T большой/неограниченный, логично рассмотреть

$$R = R(0) + \gamma R(1) + \gamma^2 R(2) + \dots, \quad 0 < \gamma < 1.$$

- Теорема Гиттинса (1979): задачу поиска оптимальной стратегии

$$\pi(\mathbf{s}_t) = \arg\max_\pi V(\pi, \mathbf{s}_t = (s_{1t}, \dots, s_{Kt}))$$

можно факторизовать и свести к

$$\pi(\mathbf{s}_t) = \arg\max_i \gamma(s_{it}).$$

- $\gamma(s_{it})$ – *индекс Гиттинса.*

Байесовский подход к многоруким бандитам

- Это очень крутой результат, который очень сильно сокращает задачу:

Professor P. Whittle (Statistical Laboratory, Cambridge): We should recognize the magnitude of Dr Gittins' achievement. He has taken a classic and difficult problem, that of the multi-armed bandit, and essentially solved it by reducing it to the case of comparison of a single arm with a standard arm. In this paper he brings a number of further insights. Giving words their everyday rather than their technical usage, I would say that my admiration for this piece of work is unbounded, meaning, of course, very great.

Despite the fact that Dr Gittins proved his basic results some seven years ago, the magnitude of his advance has not been generally recognized and I hope that one result of tonight's meeting will be that the strength of his contribution, its nature and its significance will be apparent to all.

As I said, the problem is a classic one; it was formulated during the war, and efforts to solve it so sapped the energies and minds of Allied analysts that the suggestion was made that the problem be dropped over Germany, as the ultimate instrument of intellectual sabotage. In the event, it seems to have landed on Cardiff Arms Park. And there is justice now, for if a Welsh Rugby pack scrumming down is not a multi-armed bandit, then what is?

- Но, к сожалению, индекс Гиттинса подсчитать тоже очень вычислительно трудно; поэтому не будем в это углубляться.

Оценки на regret

- Другой вариант – давайте рассчитаем приоритет каждой ручке i так, чтобы непосредственно regret ограничить.
- [Auer et al., 2002]: стратегия UCB1. Учитывает неопределённость, «оставшуюся» в той или иной ручке, старается ограничить regret. Если мы из t экспериментов n_j раз дёрнули за j-ю ручку и получили среднюю награду $\bar{x}_i$, алгоритм UCB1 присваивает ей приоритет

$$\text{Priority}_i = \bar{x}_i + a(j,t) = \bar{x}_i + c\sqrt{\frac{\log t}{n_j}}.$$

Дёргать дальше надо за ручку с наивысшим приоритетом.

Оценки на regret

- При таком подходе для $c = \sqrt{2}$ можно доказать, что субоптимальные ручки будут дёргать $O(\log T)$ раз, и regret будет $O(\sqrt{KT \log T})$.
- Если хватит времени, давайте оценку докажем...
- Но можно и ещё лучше (но доказательства будут ещё сложнее):
  - UCB2 — дёргаем за ручки по несколько раз, более сложная форма добавки;
  - UCB-Tuned — заменим $\sqrt{\frac{2 \log t}{n_j}}$ на

$$\sqrt{\frac{\log t}{n_j} \min\left(\frac{1}{4}, V_j(n_j)\right)}, \quad \text{где}$$

$$V_j(n_j) = \left(\frac{1}{n_j} \sum_{\tau=1}^{n_j} r_\tau^2\right) - \left(\frac{1}{n_j} \sum_{\tau=1}^{n_j} r_\tau\right)^2 + \sqrt{\frac{2 \log t}{n_j}}.$$

Сэмплирование по Томпсону

- И ещё один вариант: сэмплирование по Томпсону (Thompson sampling).
- Как добавить байесовской мудрости в наших многоруких бандитов?
- Идея: давайте не присваивать какой-то приоритет, а *сэмплировать ожидания наград из их апостериорных распределений* и выбирать максимальную.
- Любопытно, что тут есть два термина:
  - Thompson sampling — сэмплируем ожидания из апостериорных распределений и берём максимум;
  - probability matching — выбираем действие с вероятностями, с которыми это действие будет оптимальным согласно текущим апостериорным распределениям; это встречается где-то самостоятельно, но в целом эквивалентно.

Спасибо!

Спасибо за внимание!