Многорукие бандиты

Сергей Николенко

Академия MADE — Mail.Ru
30 сентября 2020 г.

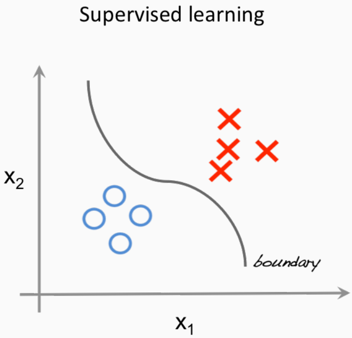
Random facts:

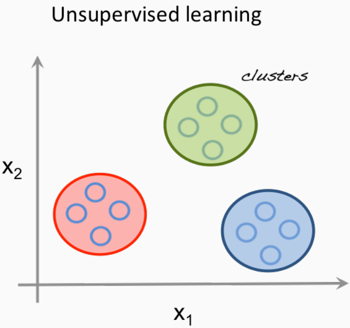
- 30 сентября 1452 г. в Майнце Иоганн Гутенберг напечатал свою первую Библию
- 30 сентября 1882 г. в городе Эпплтон (штат Висконсин) на реке Фокс заработала первая в мире гидроэлектростанция по системе Эдисона мощностью 12.5кВт
- 30 сентября 1929 г. прошла первая телетрансляция BBC, а 30 сентября 1939 г. NBC впервые провела телетрансляции матча по американскому футболу
- 30 сентября 1520 г. стал султаном Сулейман I Великолепный
- 30 сентября 2005 г. в датской газете «Jyllands-Posten» были опубликованы двенадцать карикатур на пророка Мухаммеда

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).



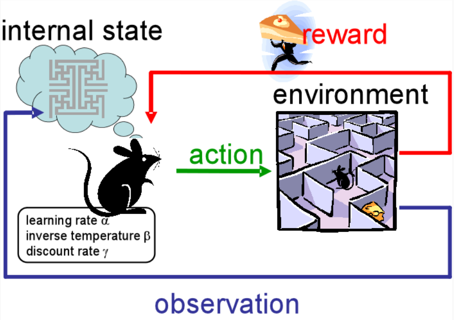


Постановка задачи

- 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.

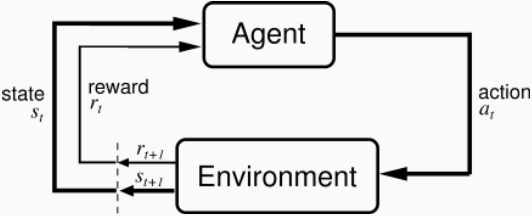
Постановка задачи

- 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...

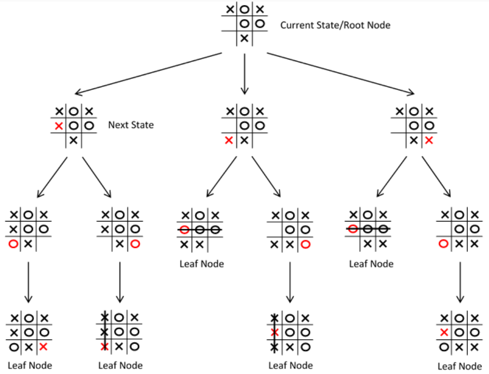
- В этом примере агент успел вернуться в состояние 1 и исследовать ранее не пробовавшуюся опцию 4 (получив за это существенную награду).

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.

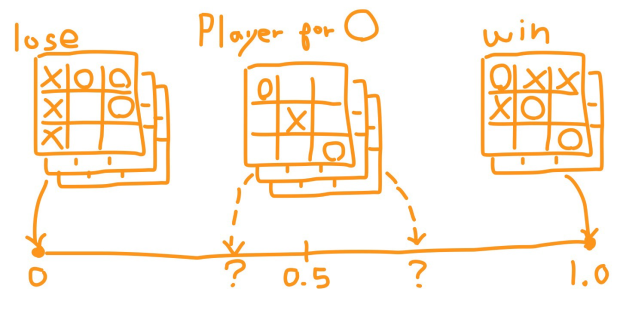
Пример

- 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

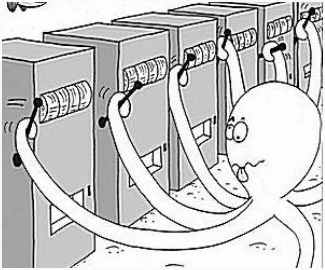
$$V(s) := V(s) + \alpha \left[V(s') - V(s) \right].$$

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

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

- Что не так с таким алгоритмом?

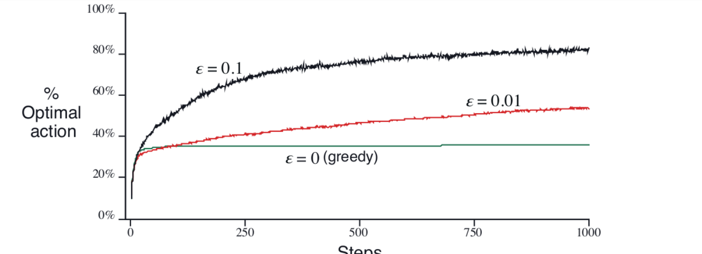
Жадный алгоритм

- 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.



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, $s=\sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$.
- Отличный метод, если вероятностные предположения соответствуют действительности (часто непонятно).

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:

$$\text{НоваяОценка} := \text{СтараяОценка} + \text{Шаг}\left[\text{Цель} - \text{СтараяОценка}\right].$$

- 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 $\mathcal{S} = \{n_1, w_1, \ldots, n_k, w_k\}$, 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 $t+1$:

$$
\begin{aligned}
V^*(n_1, w_1, \dots, n_k, w_k) = \\
&= \max_i \left(\rho_i(1 + V^*(\dots, n_i + 1, w_i + 1, \dots)) + \right. \\
&\qquad \left. (1-\rho_i)V^*(\dots, n_i + 1, w_i, \dots)\right),
\end{aligned}
$$

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 $\mathbf{s}_t = (s_{1t}, \ldots, s_{Kt})$ 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:

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

- 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

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

- Теорема Гиттинса (1979): задачу поиска оптимальной стратегии

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

  можно факторизовать и свести к

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

- $\gamma(s_{it})$ – *индекс Гиттинса*; но его подсчитать тоже обычно трудно; есть приближения.

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

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

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)

Оценки на regret

- Алгоритм Exp3 (Exponential-weight algorithm for Exploration and Exploitation):
  - для данного $\gamma \in [0,1]$, инициализируем $w_i(1) = 1$, $i = 1, \dots, K$;
  - на каждом раунде t:
    - считаем вероятности для каждого i:
    $$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^K w_j(t)} + \frac{\gamma}{K};$$
    - берём следующее действие i_t случайно с вероятностью $p_i(t)$, получаем награду $x_{i_t}(t)$;
    - обновляем вес этого действия (остальные не меняются):
    $$w_{i_t}(t+1) = w_{i_t}(t)e^{\frac{\gamma}{K}\frac{x_{i_t}(t)}{p_i(t)}}.$$
- Для такого алгоритма можно доказать adversarial оценку $2.63\sqrt{gK\log K}$ для $\gamma = \min\left(1, \frac{K\log K}{(e-1)g}\right)$, где g – верхняя оценка на максимальный доход ручки $\max_j \sum_{t=1}^T x_j(t)$.

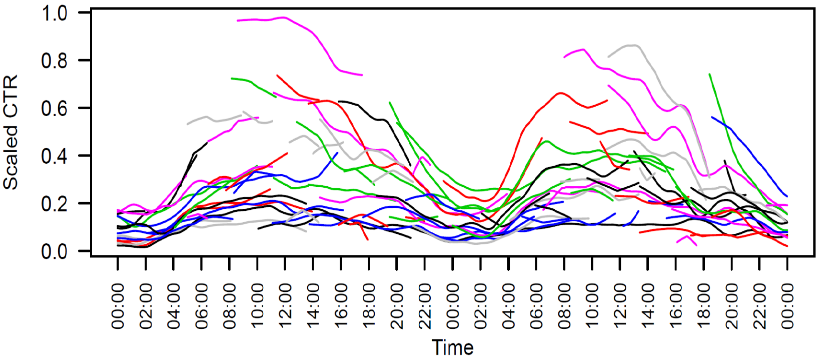
Сэмплирование по Томпсону

- 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 ручки из апостериорного распределения* и выбирать максимальную.

Клики на странице новостей

Пример: клики на странице новостей

- Пример:



Пример: клики на странице новостей

- Мы в момент t должны распределить доли показов $(x_1, x_2, \ldots, x_K)$ так, чтобы оптимизировать CTR.
- Совсем простая ситуация: два момента времени, $t = 0$ и $t = 1$, выбор из двух объектов:
  - объект P имеет CTR p_0 в момент $t = 0$ и p_1 в момент $t = 1$, мы на этот счёт не уверены, есть распределение какое-то;
  - про объект Q всё знаем точно, q_0 и q_1.
- Надо найти x – долю показов P в момент $t = 0$; у нас есть N_0 показов для распределения в $t = 0$ и N_1 в $t = 1$.

Пример: клики на странице новостей

- Пусть мы получили c кликов после того как выбрали x; c – случайная величина.
- Мы пронаблюдаем c, и на втором этапе оптимальное решение будет понятно: дать все N_1 кликов P iff

$$\hat{p}_1(x, c) = \mathrm{E}\left[p_1 \mid x, c\right] > q_1.$$

- Т.е. мы должны оптимизировать x с точки зрения общего ожидаемого числа кликов, учитывая, что что-то новое мы узнаем про p_1 к моменту $t = 1$.

Пример: клики на странице новостей

- Ожидаемое число кликов:

$$N_0 x \hat{p}_0 + N_0(1-x)q_0 + N_1 \mathsf{E}_c\left[\max\{\hat{p}_1(x,c), q_1\}\right] =$$
$$= N_0 q_0 + N_1 q_1 + N_0 x(\hat{p}_0 - q_0) + N_1 \mathsf{E}_c\left[\max\{\hat{p}_1(x,c) - q_1, 0\}\right].$$

- Второе слагаемое – это и есть выгода от исследования P:

$$\mathrm{Gain}(x, q_0, q_1) = N_0 x(\hat{p}_0 - q_0) + N_1 \mathsf{E}_c\left[\max\{\hat{p}_1(x,c) - q_1, 0\}\right],$$

его мы и оптимизируем по x.

Пример: клики на странице новостей

- Если приблизить $\hat{p}_1(x, c)$ нормальным распределением (разумно по ЦПТ):

$$\begin{aligned}
\mathrm{Gain}(x, q_0, q_1) &= N_0 x(\hat{p}_0 - q_0) + N_1 \left[\sigma_1(x)\Phi\left(\frac{q_1 - \hat{p}_1}{\sigma_1(x)}\right) + \right. \\
&\qquad \left. + \left(1 - \Phi\left(\frac{q_1 - \hat{p}_1}{\sigma_1(x)}\right)\right)(\hat{p}_1 - q_1)\right], \\
p_1 &\sim \mathrm{Beta}(a, b) \text{ (априорное распределение)}, \\
\hat{p}_1 &= \mathsf{E}_c\left[\hat{p}_1(x, c)\right] = \frac{a}{a + b}, \\
\sigma_1^2(x) &= \mathrm{Var}\left[\hat{p}_1(x, c)\right] = \frac{xN_0}{a + b + xN_0} \frac{ab}{(a + b)^2(1 + a + b)}.
\end{aligned}$$

Пример: клики на странице новостей

- Для нескольких вариантов ($K > 2$) задача становится гораздо сложнее; её можно несколько ослабить и свести к K независимым задачам с двумя вариантами.
- А что для нескольких временных слотов ($T > 1$)?

Контекстуальные бандиты

Контекстуальные бандиты

- А что если жизнь устроена сложнее? Пусть мы каждую ручку дёргаем в некотором *контексте*; например, даём рекомендации пользователям, и при этом о пользователях что-то знаем.
- Т.е. на каждом шаге наблюдаем контекст $x_t \in C$, потом выбираем $I_t \in 1, \ldots, K$, получаем $r_t \sim p(r_t \mid I_t, x_t)$.
- Тогда мы уже должны не одну ручку выбрать, а обучить стратегию $\pi : C \to 1, \ldots, K$.

Контекстуальные бандиты

- Наивный подход: давайте запустим Exp3 для каждого контекста по отдельности.
- Это не так уж плохо, если контекстов мало.
- Но получается дополнительный сомножитель $\sqrt{|C|}$ в оценке на regret, а это нехорошо, вдруг их очень много.

Контекстуальные бандиты

- Другая идея – давайте вместо этого запускать Exp3 *по стратегиям* (пусть их мало); на каждом раунде t:
  - получаем совет $\xi_{k,t}$ от каждой стратегии $k \in \Pi$ в виде распределения вероятностей на ручках;
  - берём следующее действие I_t случайно с распределением $p_t = \mathbb{E}_{k \sim q_t}\left[\xi_{k,t}\right]$, получаем награду $x_{I_t,t}$;
  - вычисляем ожидаемую награду для каждого i $\tilde{x}_{i,t}$ и ожидаемую награду для каждой стратегии

$$\tilde{y}_{k,t} = \mathbb{E}_{i \sim \xi_{k,t}}\left[\tilde{x}\right]_{i,t} = \sum_{i=1}^{K} \xi_{k,t}(i)\tilde{x}_{i,t}.$$

  - обновляем распределение на стратегиях

$$q_{j,t+1} \propto e^{-\eta \sum_{s=1}^{t} \tilde{y}_{k,s}}.$$

- Это алгоритм Exp4 (Exponential-weight algorithm for Exploration and Exploitation with Experts)
- Всё понятно?..

Контекстуальные бандиты

- ...но откуда возьмётся $\tilde{x}_{i,t}$?
- Очень важный трюк в reinforcement learning: off-policy estimation.
- Пусть мы хотим оценить качество стратегии π, но играли мы по другой стратегии p. Что делать?
- Нам нужно оценить

$$R(x, \pi(x)) = \frac{1}{n} \sum_{s=1}^{n} \mathbb{E}_{r \sim p(r|x_s, \pi(x_s))}[r].$$

- Это было бы легко, если бы мы могли оценить для каждой ручки a

$$R(x, a) = \mathbb{E}_{r \sim p(r|a,x)}[r].$$

Контекстуальные бандиты

- Оказывается, это можно сделать по данным другой стратегии. Рассмотрим

$$\hat{R}(x_s, a) = r_s \frac{[\![a_s = a]\!]}{p(a_x \mid x_s)}.$$

- Тут надо только предполагать, что $p(a_x \mid x_s)$ положительно, т.е. стратегия p покрывает стратегию π.
- Тогда

$$\mathbb{E}\left[\hat{R}(x_s, a) \mid x_s, a\right] = \mathbb{E}_{r_s}\left[\mathbb{E}_{a_s \sim p(x_s)}\left[r_s \frac{[\![a_s = a]\!]}{p(a_x \mid x_s)}\right]\right] =$$
$$= \mathbb{E}_{r_s}\left[r_s \frac{p(\pi(x_s) = a_s)}{p(a_s \mid x_s)}\right] = R(x_s, a),$$

у которого $a_s \sim p$.
- Т.е. надо просто перевзвесить вознаграждения.

Контекстуальные бандиты

- Итого:

Algorithm 2 Exp4 Algorithm (Exponential weights algorithm for Exploration and Exploitation with Experts)

Input: Set of K arms, set of experts Π.
Parameter: real number η
Let q_1 be the uniform distribution over the experts (policies), $\{1, \cdots, |\Pi|\}$.
For each round $t = 1, \cdots, T$:

- **Receive** expert advice $\xi_{k,t}$ for each expert $k \in \Pi$, where each $\xi_{k,t}$ is a probability distribution over arms.
- **Draw** an arm I_t from the probability distribution $p_t = (p_{1,t}, \cdots, p_{K,t})$, where $p_t = \mathbb{E}_{k \sim q_t} \xi_{k,t}$.
- **Observe** loss $\ell_{I_t,t}$. For each arm i, compute $\tilde{\ell}_{i,t}$, using the Inverse Propensity Score trick in Theorem 1 to obtain an unbiased estimator for the loss of arm i:

$$\tilde{\ell}_{i,t} = \frac{\ell_{i,t}}{p_{i,t}} \mathbb{1}_{I_t = i} \qquad i = 1, \cdots, K$$

- **Compute** the estimated loss for each expert, by taking the expected loss over the expert's predictions.

$$\tilde{y}_{k,t} = \mathbb{E}_{i \sim \xi_{k,t}} \tilde{\ell}_{i,t} = \sum_{i=1}^{K} \xi_{k,t}(i) \tilde{\ell}_{i,t} \qquad k = 1, \cdots, |\Pi|$$

- **Compute** the new probability distribution over the experts $q_{t+1} = (q_{1,t+1}, \cdots, q_{N,t+1})$, where

$$q_{j,t+1} = \frac{\exp(-\eta \sum_{s=1}^{t} \tilde{y}_{k,s})}{\sum_{k=1}^{|\Pi|} \exp(-\eta \sum_{s=1}^{t} \tilde{y}_{k,s})}$$

Контекстуальные бандиты

- А ещё, конечно, в контекстуальном бандите можно сделать модель какую-нибудь. Например, LinUCB предполагает, что ожидаемая награда – это линейная функция:

$$r_{a,t} = \mathrm{x}_{a,t}^{\top}\boldsymbol{\theta}_a^* + \epsilon_t,$$

  где $\boldsymbol{\theta}_a^*$ — вектор коэффициентов, который нужно оценивать, а $\mathrm{x}_{a,t}$ — вектор признаков контекста.
- И тогда по LinUCB надо выбирать $I_t = \arg\max_a u_{a,t}$, где

$$u_{a,t} = \max_{\boldsymbol{\theta}_a \in C_{a,t-1}} \mathrm{x}_{a,t}^{\top}\boldsymbol{\theta}_a,$$

  а $C_{a,t-1}$ — это доверительное множество, аналог доверительного интервала, которое мы в обычном UCB оценивали.

Контекстуальные бандиты

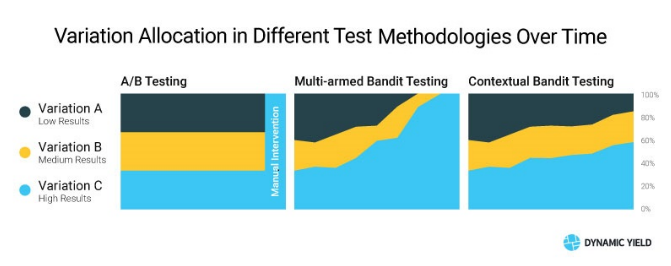
- Итого:

Algorithm 3 LinUCB with Contextual Bandits

Input: $R \in \mathbb{R}^+$, regularization parameter λ
for $t = 1, 2, \ldots, T$ **do**
  Observe feature vectors of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{a,t} \in \mathbb{R}^d$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \lambda \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\theta}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
    $C_{a,t} \leftarrow \left\{\theta_a^* \in \mathbb{R}^d : \left|\left|\hat{\theta}_{a,t} - \theta_a^*\right|\right|_{\mathbf{A}_a} \leq R\sqrt{2\log\left(\frac{\det(\mathbf{A}_a)^{1/2}\det(\lambda I)^{-1/2}}{\delta}\right)} + \lambda^{1/2}S\right\}$
    $p_{a,t} \leftarrow \arg\max_{\hat{\theta}_a \in C_{a,t}} \mathbf{x}_{a,t}^T\hat{\theta}_a$
  end for
  Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{a,t}$ with ties broken arbitrarily, and observe payoff r_t
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{a_t,t}\mathbf{x}_{a_t,t}^T$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{a_t,t}$
end for

Зачем нужны многорукие бандиты

- Многорукие бандиты используются, например, для A/B тестирования.



- Можно и для оптимизации гиперпараметров в тех же нейронных сетях (или где угодно).
- Контекстуальные – для рекомендаций, для выбора из нескольких вариантов и т.д.
- Но для нас сейчас это первый шаг, упрощённая постановка...

Спасибо!

Спасибо за внимание!