Bandycki Streaming

Szybki skrót tego co mamy do zrobienia (więcej dopowie prowadzący ;]).

Sprawy do zrobienia można podzielić na dwie sekcje. Pierwsza dotyczy implementacji poszczególnych metod. Druga dotyczy modyfikacji samego eksperymentu (i przygotowania eksperymentów dodatkowych). Zadania z obu sekcji można sobie dowolnie mieszać (ich kolejność nie jest sztywna!).

Implementacja metod

W ramach laboratorium będziemy zajmowali się testowaniem w praktyce różnych podejść do rozwiązywania problemu k-rękich bandytów. Dla uproszczenia skupimy się na jego bardzo prostym wariancie: przewidywaniu, który z utworów może stać się przyszłym hitem (a więc warto puszczać go kolejnym użytkownikom). Nagroda 1 oznacza słuchacza, który przesłuchał go w całości, nagroda 0 oznacza, że utwór został pominięty. Do zaimplementowania mamy następujące algorytmy.

Explore-Then-Commit

Zależny od parametru `m` - czasu przeznaczonego na eksplorację.

1: **Input** m.
2: In round t choose action

$$A_t = \begin{cases} (t \bmod k) + 1\,, & \text{if } t \leq mk\,; \\ \operatorname{argmax}_i \hat{\mu}_i(mk)\,, & t > mk\,. \end{cases}$$

(ties in the argmax are broken arbitrarily)

Algorithm 1: Explore-then-commit.

Zachłanny

Omówiony w części wykładowej - warto rozważyć wariant czysto zachłanny, wariant eksplorujący (z różnym *experiment rate*) oraz wariant z optymistycznymi początkowymi estymatami.

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Loop forever:
$A \leftarrow \begin{cases} \operatorname{argmax}_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]$

UCB1

Omówiony na wykładzie. Warto przyjrzeć się działaniu parametru `c` wpływającego na siłę eksploracji.

Wariant gradientowy

Omówiony na wykładzie. Warto zastanowić się nad odpowiednią realizacją modelu decyzyjnego (czy potrzebujemy tutaj pełnej sieci neuronowej?) oraz skorzystać z ulubionego frameworku. Istotny będzie też wykorzystany *learning rate*.

Próbkowanie Thompsona

Na zamknięcie - algorytm z nieco innej rodziny, na wykładzie dopiero się pojawi. W uogólnionej wersji bywa niewygodny w praktyce (bez silnego polegania na aproksymowaniu rozkładów)...

Algorithm 4 Thompson$(\mathcal{X}, p, q, r)$

for $t = 1, 2, \ldots$ **do**
  #sample model:
  Sample $\hat{\theta} \sim p$

  #select and apply action:
  $x_t \leftarrow \text{argmax}_{x \in \mathcal{X}} \mathbb{E}_{q_{\hat{\theta}}}[r(y_t)|x_t = x]$
  Apply x_t and observe y_t

  #update distribution:
  $p \leftarrow \mathbb{P}_{p,q}(\theta \in \cdot | x_t, y_t)$
end for

...ale dla naszego problemu (tzw. bandyci Bernoulliego) i modelowania rozkładami beta znacząco się upraszcza!

Algorithm 2 BernTS(K, α, β)

for $t = 1, 2, \ldots$ **do**
  #sample model:
  for $k = 1, \ldots, K$ **do**
    Sample $\hat{\theta}_k \sim \text{beta}(\alpha_k, \beta_k)$
  end for

  #select and apply action:
  $x_t \leftarrow \text{argmax}_k \hat{\theta}_k$
  Apply x_t and observe r_t

  #update distribution:
  $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$
end for

Realizacja eksperymentów

Podstawowy eksperyment to uruchomienie istniejącego już szkieletu kodu i badanie, jak będą kumulowały się w czasie nagrody otrzymane przez poszczególne algorytmy. Dla ciekawszych: kilka dodatkowych rzeczy do zrobienia.

Studium parametryczne

Jeżeli algorytm jest zależny od pewnych meta-parametrów, to warto ocenić go w pełnym przekroju ich wartości (patrz odpowiedni slajd na wykładzie). Dla zaimplementowanych przez nas rozwiązań warto przygotować analogiczne studium.

Losowe wartości oczekiwane prawdopodobieństw (oraz tzw. *regret*)

Badanie algorytmów dla tylko jednej instancji problemu utrudnia ich bardziej obiektywną ocenę. Rozszerz protokół testowy o generowanie za każdym razem nowego problemu (z osobnym układem prawdopodobieństw). Jak agregować wyniki uzyskane dla różnych problemów (w przypadku niektórych znacznie łatwiej o otrzymywanie nagród)? Opierając się o obserwację tego, na ile zgromadzone nagrody odbiegają od ich wartości oczekiwanej w sytuacji podejmowania za każdym razem optymalnej decyzji - jest to tzw. *regret*.

$$\text{let } \mu^*(\nu) = \max_{a \in \mathcal{A}} \mu_a(\nu) \text{ be the largest mean of all the arms.}$$

The regret of policy π on bandit instance ν is

$$R_n(\pi, \nu) = n\mu^*(\nu) - \mathbb{E}\left[\sum_{t=1}^{n} X_t\right],$$

where the expectation is taken with respect to the probability measure on outcomes induced by the interaction of π and ν.

Bandyci niestacjonarni

Zmodyfikuj problem tak, by symulował stopniowy dryf preferencji odbiorców muzyki - niech z każdym krokiem czasowym prawdopodobieństwa przesłuchania utworu do końca zmieniają się o pewną niewielką wartość (np. losowaną z użyciem rozkładu normalnego) - w praktyce jest to pewne błądzenie losowe. Jak taka zmiana wpłynęła na wzajemną skuteczność badanych algorytmów? PS. W tym przypadku warto zmodyfikować algorytm zachłanny tak, by korzystał z pewnego *learning rate* - czyli zrealizować średnią wykładniczo ważoną aktualnością.

Trudniejszy dodatkowy problem [+1 punkt bonusowy]

Kilka utrudniających założeń:

- utworów jest nieco więcej (+- kilkadziesiąt);
- inny algorytm, na który nie mamy wpływu, wygenerował zawierające je playlisty (playlist jest więcej niż utworów, +- kilkaset);
- ten sam utwór może być elementem wielu playlist (w różnych pozycjach na playliście);
- słuchaczom polecamy całe playlisty, nie pojedyńcze utwory;
- jeżeli słuchaczowi nie spodoba się utwór, to porzuca słuchanie całej playlisty;
- naszym celem jest, by słuchał playlistę jak najdłużej (najlepiej do końca).

Zaproponuj konkretne ramy eksperymentu i zmodyfikuj dwa z wymienionych wcześniej algorytmów tak, by radziły sobie z powyższą sytuacją. Sprawdź, który wypadnie lepiej.

Bibliografia

- Lattimore, Tor, and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.
- Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Russo, Daniel J., et al. "A tutorial on thompson sampling." *Foundations and Trends® in Machine Learning* 11.1 (2018): 1-96.
- https://cse442-17f.github.io/LinUCB/