پروژه عملی یادگیری تقویتی

گراف‌های تصادفی به همراه اطلاعات جانبی

امیر محمد اوحدی محمد اسمعیلی یاشار طالبی راد

۵ اسفند ۱۳۹۸

چکیده

مسئله‌ی مورد نظر، «راه‌زن بر روی گراف به همراه اطلاعات شباهتی» است، که ایده‌ی یادگیری با استفاده از بازخورد گرفتن از دوست‌ها در یک شبکه‌ی اجتماعی را مطرح می‌کند. هدف اول بررسی و پیاده سازی راه‌کارهای موجود برای گراف‌های اردوش–رنی (Erdos-Renyi) است. سپس رسیدن به الگوریتم‌های جدید برای مدل بارابسی–آلبرت (Barabsi-Albert) ، که گراف‌های تصادفی واقعی‌تری برای مدل کردن شبکه‌های اجتماعی را شامل می‌شود.

فهرست مطالب

۱ مقدمه

۲ مسئله راهزن چند دست و کنترل پشیمانی

۱.۲ تعاریف اولیه

مدل ابتدایی راهزن چند دست به شکل زیر است:
K گزینه و T واحد زمانی داریم. الگوریتم مورد نظر در هر واجد زمانی یک گزینه را انتخاب کرده و سپس پاداش r_t را دریافت می کند. در حالت ساده می توان فرض کرد که پاداش هر گزینه مستقل از بقیه گزینه ها از توزیع ثابتی می‌آید و در حالت کلی پاداش یک فرایند تصادفی است. الگوریتم باید تعادلی بین اکتشاف و بهره بری از محیط برقرار نماید، پشیمانی معیاری برای نشان دادن اینکه الگوریتم چقدر خوب عمل کرده با در نظر گرفتن برخط بودن تصمیمات است و به صورت زیر تعریف می شود:

$$R(T) = \mu^*.T - \sum_{t=۱}^{T} \mu(a_t)$$

که a_t گزینه انتخاب شده در مرحله t ام، $\mu(a_t)$ میانگین توزیع آن و μ^* گزینه ی با بیشترین میانگین است. در واقع پیشمانی بیان می‌کند نسبت به حالتی که به صورت غیر برخط از ابتدا تصمیم می گرفتیم و بهترین گزینه را انتخاب می کردیم، چقدر بد عمل کرده ایم.
در این بخش به بیان الگوریتم‌هایی برای کمینه کردن پشیمانی خواهیم پرداخت.

۲.۲ الگوریتم‌های اکتشافی یکنواخت

اولین الگوریتم ایده ساده‌ای دارد و از دو فاز مستقل از هم اکتشاف و بهره‌برداری تشکیل شده است :

Explore First with Parameter N
فاز اکتشاف : هر گزینه را N بار انجام بده. فاز بهره بری: گزینه با میانگین بیشترین پاداش را تا انتها انجام بده.

الگوریتم فوق کران بالای

$$E[R(T)] \leq T^{\frac{۲}{۳}} \times O(K \log T)^{\frac{۱}{۳}}$$

را برای پشیمانی می دهد.
می‌توان اکتشاف را در طول زمان یکنواخت کرد تا در ابتدای کار یک فاز طولانی اکتشاف نداشته باشیم و در هر مرحله نسبت به بهترین گزینه تا آن زمان حریصانه عمل کرد که به الگوریتم زیر می‌رسیم:

Epsilon-Greedy with exploration probability ϵ_t
در هر مرحله با احتمال ϵ_t اکتشاف (با انتخاب یکنواخت یک گزینه) و با احتمال $۱ - \epsilon_t$ بهره بری از گزینه با بیشترین میانگین پاداش تا اینجا انجام بده.

قضیه الگوریتم ϵ ـ حریصانه با پارامتر $\epsilon_t = t^{\frac{۱}{۳}} \times (K \log t)^{\frac{۱}{۳}}$ دارای کران پشیمانی $E[R(t)] \leq t^{\frac{۲}{۳}} \times O(K \log t)^{\frac{۱}{۳}}$ است. (برای هر $t \leq T$)

۳.۲ الگوریتم‌های اکتشافی تطبیق‌پذیر

یکی از مشکلات هر دو الگوریتم ارائه شده در قسمت قبل این است که در فاز اکتشاف از تاریخچه پاداش‌های به دست آمده هیچ استفاده ای نمی‌کنند. در ادامه دو الگوریتم ارائه خواهیم داد که خود را با این اطلاعات تطبیق می‌دهند.
فرض کنیم در دور t ام هستیم و $n_t(a)$ تعداد انتخاب گزینه a تا لحظه t باشد و $\bar{\mu}_t(a)$ متوسط پاداش دریافتی از این گزینه، با استفاده از

نابرابری هوفدینگ داریم:

$$Pr[|\bar{\mu}_t(a) - \mu_a| \leq r_t(a)] \geq ۱ - \frac{۲}{T^۴}, \quad r_t(a) = \sqrt{\frac{۲ \log T}{n_t(a)}}$$

$r_t(a)$ را *شعاع اطمینان* می نامیم. گیریم:

$$\mathbf{UCB}_t(a) = \bar{\mu}_t(a) + r_t(a)$$
$$\mathbf{LCB}_t(a) = \bar{\mu}_t(a) - r_t(a)$$

در این صورت $[\mathbf{LCB}_t(a), \mathbf{UCB}_t(a)]$ را بازه اطمینان می نامیم. پس با احتمال ارائه شده در نابرابری بالا پاداش دریافی از گزینه a در بازه اطمینان متناظر قرار دارد. حال این ایده را استفاده می‌کنیم که اگر برای دو گزینه مثل a, a' داشته باشیم $\mathbf{UCB}_t(a') < \mathbf{LCB}_t(a)$ احتمالا a گزینه بهتری است. در این حالت می‌نویسیم $a > a'$.

Successive Elimination Algorithm
تمامی گزینه ها را در حالت فعال قرار بده. در هر فاز (شامل چند مرحله می تواند باشد): تمامی گزینه های فعال را یک بار انجام بده. اگر گزینه‌ای وجود داشت که با رابطه ترتیب بالا گزینه بهتر از آن وجود داشت آن را حذف کن. (در هر مرحله بازه های اطمینان به روز رسانی می شود.) مراحل بالا را تا پایان زمان تکرار کن.

قضیه Successive Elimination داری کران پشیمانی

$$E[R(t)] = O(\sqrt{Kt \log T}) \quad \forall t \leq T$$

است.

UCB1
هر گزینه را یک بار تست کن. در هر مرحله گزینه با بیشترین $\mathbf{UCB}_t(a)$ را انجام بده.

داشتیم :

$$\mathbf{UCB}_t(a) = \bar{\mu}_t(a) + r_t(a)$$
$$r_t(a) = \sqrt{\frac{۲ \log T}{n_t(a)}}$$

پس یک گزینه به دو دلیل می تواند کران بالای بیشتری داشته باشد. یا دارای میانگین پاداش بالا است یا به اندازه کافی برای اکتشاف تست نشده است.

قضیه UCB1 داری کران پشیمانی

$$E[R(t)] = O(\sqrt{Kt \log T}) \quad \forall t \leq T$$

است.

۳ راهزن با اطلاعات شباهتی

در این حالت ممکن است تعداد زیادی بازو داشته باشیم (حتی نامتناهی)و مسئله به نظر غیر قابل حمله می‌رسد مگر اینکه به نحوی بین انتخاب‌ها شباهت تعریف کنیم. یک راه استفاده از یک متر روی فضای انتخاب ها است که امید پاداش نسبت به آن لیپشیتز باشد و کمک می‌کند نوعی شباهت موضعی بین پاداش های یک گزینه وجود داشته باشد.

۱.۳ Fixed Descretization

نیاز به توازن برای کم یا زیاد بودن مجموعه گسسته سازی شده با توجه به فرمول زیر.

$$E[R(T)] = R_S(T) + T.\mathbf{DE}(S), \quad \mathbf{DE}(S) = \mu^*(X) - \mu^*(S)$$

۲.۳ Zooming

۴ راهزن وابسته به قراین

در راهزن وابسته به قراین [۱] پاداش هر مرحله وابسته به یک قرینه [۲] است که الگوریتم در هر مرحله و پیش از تصمیم گیری آن را مشاهده می کند. به طور مثال در نمایش تبلیغات به کاربر، اینکه کدام کاربر آماده مشاهده تبلیغ است یک قرینه محسوب می شود. اگر گزینه a_t پس از مشاهده قرینه x_t انتخاب شود، پاداش r_t دریافت می‌کنیم که از توزیعی وابسه به زوج (x_t, a_t) می‌آید که در طول زمان ثابت و مستقل از بقیه توزیع‌ها است. میانگین آن را با $\mu(a|x)$ نمایش می‌دهیم. همان طور که اشاره شد اینکه هر کاربر علایق و ویژگی های منحصر به فرد خود را دارد از اصلی ترین انگیزه مطالعه راهزن در این شرایط است، زیرا احتمالا هر کاربر به هر گزینه واکنش متفاوتی نشان خواهد داد و پاداش متفاوتی خواهیم گرفت. قراین می‌توانند ویژگی‌های محیط مانند روز هفته، همزمانی با رویداد ها و ... هم باشند و همچنین می‌توانیم فرض کنیم برای هر قرینه هم زیر مجموعه‌ای از گزینه ها قابل انتخاب است، به طور مثال محدوده سنی کاربر مانع از نمایش برخی تبلیغات می شود. به طور مشابه پیشمانی را به صورت زیر تعریف می کنیم:

$$R(T) = rew(\pi^*) - rew(alg)$$

که در آن:

$$\pi^*(x) = max_a \mu(a|x)$$

$$E[rew(alg)] = \sum_{t=1}^{T} \mu(a_t|x_t)$$

۱.۴ الگوریتم ها

الگوریتم LinUCB متداول ترین روش مدرن برای راهزن وابسته به قراین است که در آن گزینه‌ها بر اساس کران بالای اطمینان [۳] محاسبه شده روی پاداش تخمینی با فرض داشتن بردارهای قراین انتخاب می‌شوند. این الگوریتم فرض می‌کند که پارامترهای راهزن ها از هم مستقلند. همچنین این الگوریتم فقط از ویژگی‌های دیده‌شده استفاده می‌کند و با ویژگی‌های پنهان کاری ندارد.

در مقابل الگوریتم CoLin یک الگوریتم دیگر برای راهزن وابسته به قراین است که وابستگی بین راهزن ها را با یک گراف وزن دار با ماتریس ورنش مدل می‌کند. در این گراف هر گره متناظر با یک راهزن برای یک نفر و وزن هر یال متناظر با میزان تاثیر پذیری بین دو نفر مجاور است. طبق این ساختار انتظار می‌رود پاداش دیده‌شده از یک نفر تقریبا توسط همسایه‌هایش در گراف معلوم شود. همچنین قراین و پاداش های گرفته شده توسط یک نفر در کل گراف به صورت آنلاین پخش می‌شود.

۵ گراف‌های تصادفی

در این پروژه از دو مدل مختلف گراف تصادفی استفاده می‌شود. در این بخش سعی می‌کنیم این دو مدل را بررسی کنیم.

مدل اردوش–رنیی [۴]

در مدل اردوش–رنیی بعد از مشخص‌کردن تعداد یال‌ها(M) و تعداد رئوس(n) ، یک گراف را بین همه‌ی گراف‌های با M یال و n رأس را با احتمال یکسان به صورت تصادفی انتخاب می‌کند. در یک روش دیگر که آن هم به اردوش–رنیی موسوم است، هر یال گراف کامل با احتمال p انتخاب می‌شوند که در گراف تصادفی حضور داشته باشد. این گراف‌ها بیشتر برای اثبات وجود گرافی با خاصیت‌های مشخص

[۱] Contextual Bandit
[۲] Context
[۳] Upper Confidence Bound
[۴] Erdős–Rényi model

به کمک روش احتمالاتی استفاده می‌شود. در کتابخانه‌ی NetworkX در پایتون، ساختن یک گراف اردوش رنیی نوع اول به کمک دستور nx.gnm_random_graph(n, m) و نوع دوم آن با دستور Erdos_renyi_graph(n, p) انجام می‌شود.

مدل باراباسی ـ آلبرت[۵]

بسیاری از شبکه‌های مورد بررسی موجود، شبکه‌های مستقل از مقیاس[۶] هستند، یعنی تعداد رئوسی که درجه‌ای برابر k دارند متناسب با k^{γ}است که γ پارامتری است که معمولا بین ۲ و ۳ است. گراف‌های تولیدشده توسط مدل اردوش ـ رنیی، این خاصیت را نداشته و بنابراین مدل خوبی برای شبیه‌سازی شبکه‌های واقعی نیستند. مدل باراباسی ـ آلبرت یکی از مدل‌های پرکاربرد گراف‌های تصادفی است که به خاطر مستقل از مقیاس بودن، مدل خوبی برای شبیه‌سازی بسیاری از شبکه‌های انسانی مانند اینترنت یا شبکه‌های اجتماعی به شمار می‌رود.

این گراف‌ها همچنین شامل گره‌هایی به نام هاب هستند که درجهٔ بالای غیر عادی در مقایسه با سایر گره‌های شبکه دارند. این به این علت است که اصولا در شبکه‌های این چنینی، گره‌هایی که درجه بالاتری دارند با احتمال بیشتری گره‌های جدید شبکه را جذب می‌کنند. این مشاهده، خود روش ساختی برای گراف‌های باراباسی ـ آلبرت به ما می‌دهد.به طور دقیق تر، ابتدا از m گره متصل شروع کرده و در هر مرحله، گره جدیدی اضافه می‌شود که با احتمالp_i به رأس i وصل می‌شود که $p_i = \frac{d_i}{\sum_j d_j}$ و k_i درجه گره i است. در پایتون، ساختن چنین گراف‌هایی با استفاده از دستور nx.barabasi_albert_graph(n, m) انجام می‌شود.

۶ راهزن بر روی گراف با اطلاعات شباهتی

رئوس گراف را کاربران در نظر گرفته و روابط دوستی یا شباهت بین آن ها را یال در نظر می گیریم و هدف تعریف تابعی هموار به عنوان متر روی گراف هست که با گذار روی همسایه ها خیلی تغیرات ویژگی نداشته باشیم. از مقادیر ویژه لاپلاسین کمک گرفته می شود و به راهزن طیفی [۷] مشهور است. تا به بتوانیم از راهزن لیپشیتز وابسته به قراین کمک بگیریم.

۷ روش امیر ـ اسی ـ راد

مسئله‌ی advertising management را در نظر بگیرید(در باقی کاربرد ها مسئله مشابه این مسئله است) فرض کنید شرکت YouTube به دنبال بیشینه سازی سود خود در قسمت تبلیغات است ، او برای هر فرد مناسب‌ترین تبلیغات را انتخاب کرده و اگر آن شخص به روی آن کلیک کند مبلغی به حساب شرکت واریز می‌شود.

حال سه حالت وجود دارد اگر تبلیغ به شخص نشان داده شود و وی کلیک کند عدد ۱+ نظیر می‌شود ،اگر کلیک نکند عدد ۱ ـ نظیر می‌شود .و اگر تبلیغ نشان داده نشود عدد ۰ نظیر می‌شود. و این شرکت برای هر مشتری به اندازه‌ی k تا سابقه‌ی تبلیغ برای هر مشتری نگه می‌دارد،یعنی یک ارایه‌ی k تایی از ۱+،۰، ۱ ـ که هر ردیفشان متانظر با یک تبلیغ خاص است .

حال به مدل Erdos-renyi میرویم این یک گراف تصادفی است یعنی یک یال به یک احتمال خاصی وصل است(و ما از قبل دینامیک دوستی افراد را نداریم) حال بر اساس این ارایه‌ی k تایی باید تخمین بزنیم که به چه احتمالی افراد با هم دوست هستند ، این گونه که دو بردار k تایی را در هم ضرب داخلی کرده (بدیهی است که اگر سایز مساله زیاد باشد نمی‌توان به ازای هر دو عنصر حساب کرد بلکه باید از ایده‌ی Batch learning استفاده کرد)(که ممکن است عدد مذکور منفی شود) سپس این اعداد را تبدیل به احتمال کرده (یعنی با k جمع کرده سپس تقسیم بر ۲k می‌کنیم) حال میخواهیم پیش بینی کرده که این شخص در برابر یک تبلیغ جدید چه واکنشی نشان می‌دهد. کافی است که از رفتار دوستانش میانگین وزن‌دار گرفته بر مبنای احتمالاتی که بدست آورده ایم و رفتاری که دوستانش انجام داده اند و تبلیغی را نشان دهیم که بالاترین میانگین را دارد یا یک آستانه تعریف کنیم که اگر عدد مذکور از آن عدد بالاتر بود تبلیغ را نشان بده.

حال برای مدل barabsi-albert یک الگوریتم ارائه می‌دهیم.

در این مدل احتمالاتی ،هر چه راس درجه‌اش بیشتر باشد احتمال این که درجه‌اش باز هم بیشتر شود بیشتر است نسبت به راسی که درجه‌اش کمتر است. بنابراین باید ۲ کار انجام داد.

۱.بررسی و پیدا کردن عناصری که درجه‌ی بالایی دارند(دوستان زیادی دارند)

۲.برای هر عنصر جدید باید بررسی کنیم که با کدام یه از این افراد بیشتر دوست است و سپس یک میانگین وزن دار و کار تمام می‌شود.

حال باید الگوریتمی برای اولی بدهیم کافی است که تمام بردار ها را باهم جمع کرده و به ازای هر نفر بررسی کنیم که کدام نفر ضرب داخلی اش با این بردار بیشتر است ،چون که افرادی که دوستان زیادی دارند سلیقیشان در جمع trend شده و عملا جمع تابع آن شخص رفتار می‌کنند پس بدیهی است که ضرب داخلی بیشتری دریافت می‌کنند از محبوبیت بیشتری برخوردار هستند.

---

[۵] Barabási–Albert model

[۶] Scale-free

[۷] spectral bandit

برای حل مشکل دوم فرض کنید که ما M نفر محبوب را داریم وقرار است اگر فرد جدیدی را میخواهیم بررسی کنیم دوستانش بین این M نفر باشند،دو باره مانند حالت قبل ضرب داخلی با این بردار ها را انجام داده و حال این اعداد را بر مبنای این ضرب داخلی و میزان محبوبیت آن شخص ،اعداد را به احتمال تبدیل کرده و میانگین وزن دار میگیریم از اعداد دیگران.

۸ پیاده سازی و آزمایش

در هر جایی که یک الگوریتم با یکسری پارامتر استفاده کرده ایم آزمایش برای آن الگوریتم و پارامتر ها ۴۰ بار تکرار شده و از تمامی مقادیر به دست آمده روی این ۴۰ بار میانگین گرفته ایم. توزیع گزینه ها نرمال میانگین μ و واریانس ۱ است. اگر مسئله دارای K گزینه است، میانگین توزیع های گزینه ها را $1, 2, .., K$ گرفته ایم.

در این قسمت می‌خواهیم تاثیر پارامترهای مختلف را روی الگوریتم‌هایی که پیاده کرده‌ایم بررسی کنیم. ابتدا پارامترهای هر تابع را تعریف می‌کنیم.

```
epsilon_greedy(e_t =formula)
```

این پارامتر مقدار ϵ را در هر مرحله مشخص می‌کند.

```
explore_then_exploit(num_of_exploration)
```

این پارامتر تعداد بار تست کردن هر گزینه در ابتدا را مشخص می‌کند. همچنین تابع random به صورت تصادفی و یکنواخت یکی از گزینه‌ها را انتخاب می‌کند.

```
ubc1
successive_elimination
```

حال پارامترهای محیط را تعریف می‌کنیم. num_of_arms تعداد گزینه‌ها، time_horizon تعداد مراحل است

در ادامه نتیجه چند شبیه سازی از الگوریتم ها را خواهیم آورد. کد هر آزمایش در فایل experiment.py و در تابعی به شماره همان آزمایش است. مثلا تابع

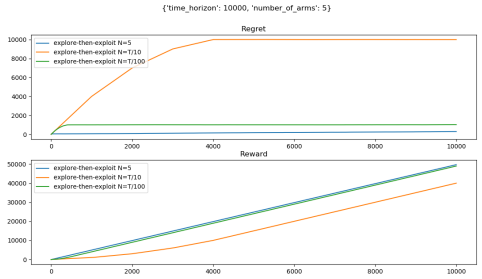
experiment1()

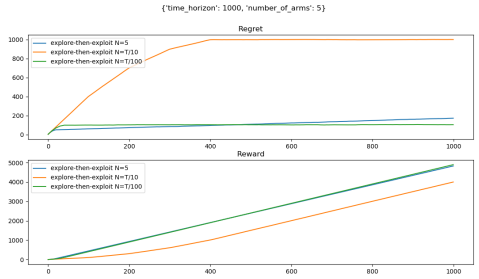
مربوط به آزمایش اول است.

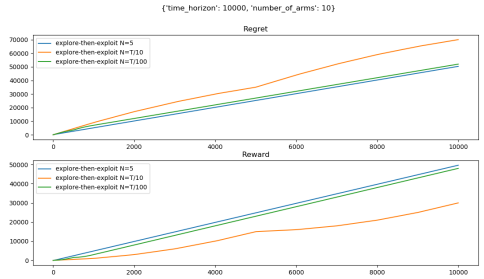
۱.۸ آزمایش اول

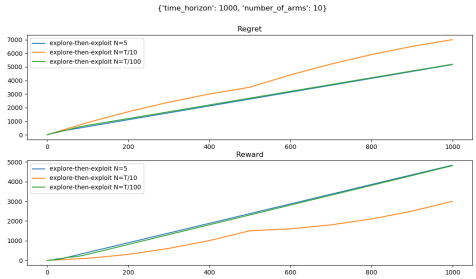
در این آزمایش قصد داریم تاثیر این که هر گزینه را چند بار تست کنیم را در الگوریتم explore-then-exploit بررسی نماییم. طبیعتا یکی از ایرادات این الگوریتم این است که یک برآورد انجام میدهد و همیشه از آن استفاده می کند و اگر تعداد آزمایش ها زیاد شود و در ابتدا خوب براورد نکرده باشد خود را اصلاح نمی کند،اما همان طور که در ابتدای این بخش مطرح کردیم ما آزمایش را تکرار می کنیم و این تا حدی این تاثیر را از بین می برد لذا صراحتا به این عیب که در نتایج احتمالا دیده نخواهد شد اشاره کردیم.

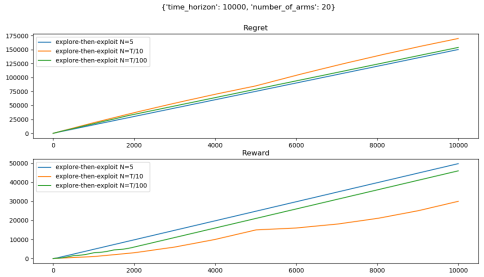
آزمایش با پارامتر ها و میزان پاداش و پشیمانی که در نمودار های زیر آمده و به ازای سه الگوریتم با تعداد تست هر گزینه همواره ۵ تا، یک دهم زمان و یک صدم زمان انجام شده است.

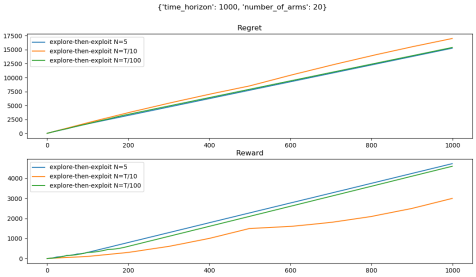






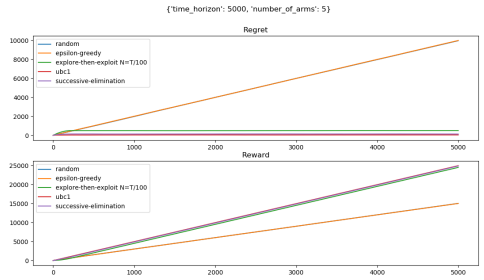


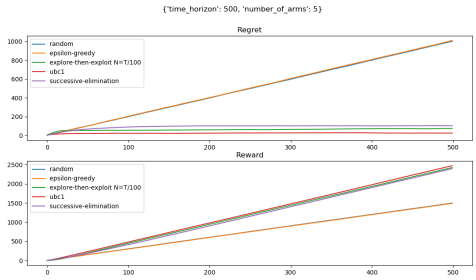


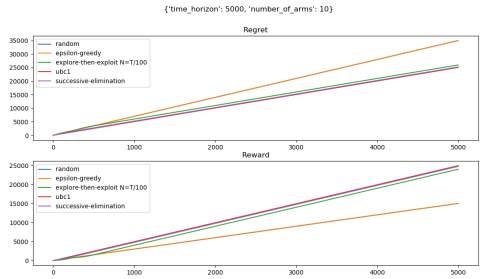


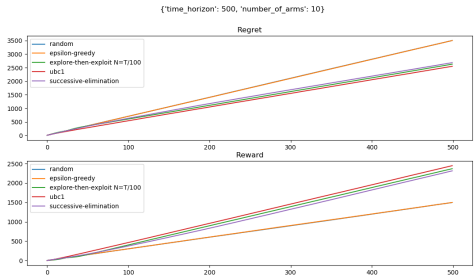
۲.۸ آزمایش دوم

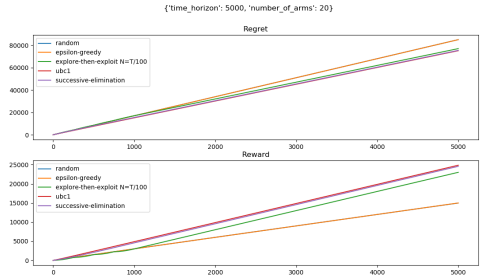
در این آزمایش قصد داریم عملکرد کلی الگوریتم ها را با هم مقایسه کنیم. الگوریتم های تصادفی انتخاب کردن، اپسیلون حریصانه با اپسیلون مناسبی که قبلا در گزارش در قضیه ای ذکر شده بود، successive_elimination، ubc۱ و explore-the-exploit با $N = \frac{T}{۱۰۰}$ الگوریتم ها و پارامتر های محیط مانند آمایش قبل اند (فقط زمان نصف شده است).

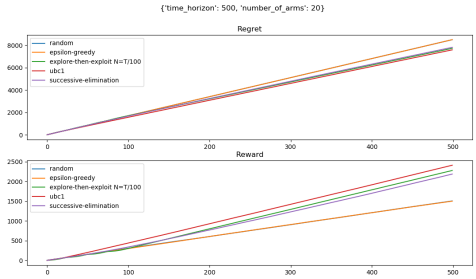












در مجموع الگوریتم های ubc و successive_elimination عملکرد بهتری دارند.

۹ نتیجه

دراین پروژه الگوریتم های راهزن چند دست را بررسی و پیاده سازی کرده و تاثیر تغییر پارامترهای مختلف مسئله راهزن را روی آن ها مقایسه کردیم. الگوریتم ucb۱ و بعد از آن successive_elimination پیشمانی کمتری داشتند. در نهایت یک روش برای مدل سازی نمایش کدام تبلیغ به یک کاربر که روی گراف دوستی یا شباهت برای نمایش تبلیغ اجرا می شد را پیاده سازی نمودیم. خروجی روش هر چه به ۱ نزدیک باشد احتمال اینه که کاربر تبلیغ را ببیند بیشتر و هر چه به ۱ ــ نزدیک تر باشد احتمال اینکه کار رو انجام ندهد بیشتر است

۱۰ منابع

1- Slivkins, A. 2018. “Introduction to Multi-Armed Bandits”. Foundations and Trends in Machine Learning.
2- Slivkins, A. 2014. “Contextual bandits with similarity information”. J. of Machine Learning Research (JMLR). 15(1): 2533–2568. Preliminary version in COLT 2011.
3- Michal Valko. Bandits on graphs and structures. Machine Learning [stat.ML]. École normale supérieure de Cachan - ENS Cachan, 2016.