دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

گزارش تمرین شماره ۲

درس یادگیری تعاملی

پاییز ۱۴۰۱

نام و نام خانوادگی

محمد سعادتی

شماره دانشجویی

۸۱۰۱۹۸۴۱۰

فهرست

چکیده

هدف از این تمرین، مدل سازی و حل مساله های دنیای واقعی با استفاده از مساله Multi-Armed Bandit می باشد.

بخش ۱ – سوالات تحلیلی

در بخش سوالات تحلیلی، سه مساله از مسائل دنیای واقعی انتخاب شده اند و برای هرکدام، یک مدل مبتنی بر مساله Multi-Armed Bandit ارائه می دهیم.

سوال ۱

برای ارائه یک مدل مبتنی بر مساله Multi-Armed Bandit ، نیاز است تا مجموعه بازوها، پاداش و نحوه پاسخ دهی به مساله با توجه به حالات مختلف ورودی تبیین شود.

منظور از مجموعه بازوها، مجموعه عمل(Action) هایی است که می توانیم انجام دهیم. در مساله فعلی، عمل هایی که توانایی انجام آنها را داریم همان انتخاب هایی هستند که برای ترتیب جایشگت انجام حرکات تمرینی در هر یک از سه برنامه تمرینی داریم؛ که این مجموعه برای هر یک از سه برنامه تمرینی، یک مجموعه با اندازه تعداد حرکات تمرینی آن برنامه و با عضو های حرکات تمرینی آن برنامه می باشد.

فرض می کنیم هدف از حل مساله، بیشینه کردن مقدار انرژی مصرف شده در انجام حرکات تمرینی است (فرض کرده ایم که هر چه انرژی مصرف شده بیشتر باشد، وضعیت سوخت و ساز بدن در شرایط مطلوب تری قرار می گیرد و عضلات بیشتری درگیر می شوند) (حالتی که هدف کمینه کردن مقدار انرژی مصرف شده در انجام حرکات تمرینی است نیز بطور مشابه قابل اثبات است). در نتیجه ابزاری که مقدار انرژی مصرف شده را برای ما اندازه گیری می کند میتواند به عنوان پاداش در نظر گرفته شود.

منظور از "حالات مختلف ورودی"، مفهوم Context است (میتوان آن را به State هم تعبیر کرد). در مساله فعلی، میخواهیم توالی بهینه انجام حرکات را برای ۳ روز مجزا - که پاداش دریافتی از یک حرکت یکسان در دو روز متفاوت می تواند تفاوت داشته باشد - پیدا کنیم. در نتیجه برای مساله ما سه Context یا State متفاوت وجود دارد که برای هرکدام باید یک عامل یادگیر Multi-Armed Bandit را تمرین داد.

سوال ۲

برای ارائه یک مدل مبتنی بر مساله Multi-Armed Bandit ، نیاز است تا مجموعه بازوها، پاداش و نحوه پاسخ دهی به مساله با توجه به حالات مختلف ورودی تبیین شود.

منظور از مجموعه بازوها، مجموعه عمل(Action) هایی است که می توانیم انجام دهیم. در مساله فعلی ما ۳۱ عمل به شماره ۰ تا ۳۰ تعریف می کنیم که هر یک از این اعمال به معنای آن است که به اندازه شماره عمل پشت چراغ قرمز تقاطع برای سبز شدن آن صبر می کنیم و اگر پیش از تمام شدن مدت انتظار چراغ سبز شد، به مسیر مستقیم ادامه می دهیم و از مسیر اصلی و اولی (مسیری که ۱۰ دقیقه طول می کشد تا به دانشگاه برسیم) به دانشگاه می رویم و متناسب با مدتی که حدس زده بود باید در صف بایستد، به علاوه تعداد دقایق رسیدن در مسیر انتخابی به دانشگاه (برای مسیر اول ۱۰ دقیقه) پاداش منفی دریافت می کند. اگر چراغ قرمز تقاطع تا زمان حدس زده شده سبز نشد، عامل تغییر مسیر می دهد و از سمت راست تقاطع و مسیر جایگزین (مسیر ۳۰ دقیقه ای) به دانشگاه می رود و جریمه ای (پاداش منفی) متناسب مدت زمان انتظار پشت چراغ قرمز به علاوه تعداد دقایق رسیدن در مسیر انتخابی به دانشگاه (برای مسیر دوم و جایگزین ۳۰ دقیقه) دریافت می کند.

با توجه به صورت سوال، هدف از حل مساله کمینه کردن مدت زمان رسیدن به دانشگاه می باشد. در نتیجه مجموع مدت زمانی که پشت چراغ قرمز صبر می کنیم، با مدت زمانی که طول می کشد در مسیر انتخابی بعد از تقاطع (مسیر مستقیم و اصلی یا گردش به راست و مسیر جدید و جایگزین) به دانشگاه برسیم می تواند به عنوان جریمه (پاداش منفی) در نظر گرفته شود. ما ۳۱ عمل به شماره ۰ تا ۳۰ تعریف کرده ایم که بطور مثال عمل شماره ۱۵ یعنی عامل به اندازه ۱۵ دقیقه پشت چراغ قرمز صبر می کند؛ اگر چراغ زودتر از ۱۵ دقیقه سبز شد که از مسیر اولی و اصلی به سمت دانشگاه می رود و پاداش منفی ای متناسب با ۱۵ دقیقه صبر کردن و ۱۰ دقیقه مدت زمان مسیر اولی دریافت می کند. اما اگر چراغ در مدت زمان ۱۵ دقیقه سبز نشد، پاداش منفی ای منتاسب با ۱۵ دقیقه صبر کردن پشت چراغ قرمز به علاوه ۳۰ دقیقه مدت زمان مسیر دوم و جایگزین دریافت می کند. توجه کنید که در این مثال حتی اگر چراغ در دقیقه اول نیز سبز بشود، عامل به اندازه ۱۵ دقیقه صبر کردن جریمه می شود زیرا عامل سعی کند زمان بهینه صبر کردن قبل از تغییر مسیر را بیاموزد و هر دفعه حدس های دقیق تری را ارائه بدهد. در واقع اگر این کار انجام نمی شد عامل زمان بهینه صبر کردن را یاد نمی گرفت و همیشه آخرین عمل (عمل ۳۱ ام) را انتخاب می کرد.

منظور از "حالات مختلف ورودی"، مفهوم Context است (میتوان آن را به State هم تعبیر کرد). در مساله فعلی، میخواهیم بهترین استراتژی ممکن برای رفتن به دانشگاه از مسیر مدنظر را پیدا کنیم. در نتیجه برای مساله ما یک Context یا State وجود دارد که باید یک عامل یادگیر Multi-Armed Bandit را تمرین داد. برای این منظور ما اعمال را طوری طراحی کردیم که پس از انتخاب هر عمل مسئله پایان یابد و بنابراین عامل بتواند زمان بهینه صبر کردن پشت چراغ قرمز را بیاموزد و همچنین با شرط تک حالته بودن مسئله نیز سازگار باشد.

سوال ۳

برای ارائه یک مدل مبتنی بر مساله Multi-Armed Bandit ، نیاز است تا مجموعه بازوها، پاداش و نحوه پاسخ دهی به مساله با توجه به حالات مختلف ورودی تبیین شود.

منظور از مجموعه بازوها، مجموعه عمل(Action) هایی است که می توانیم انجام دهیم. در مساله فعلی، عمل هایی که توانایی انجام آنها را داریم همان گزینه هایی هستند که برای انتخاب درگاه مناسب جهت ارسال بسته وارد شده به مسیریاب به سمت مقصد مد نظر داریم؛ که این مجموعه، یک مجموعه سه عضوی متشکل از سه تا از چهار درگاه مسیریاب می باشد (فرض کرده ایم درگاهی که بسته از طریق آن از طرف مبدا وارد مسیریاب شده است، از درگاه های قابل انتخاب برای ارسال بسته از مسیریاب به مقصد موردنظر حذف می شود).

با توجه به صورت سوال، هدف از حل مساله، کمینه کردن مدت زمان ارسال بسته ها به مقصدشان با توجه به مقصدشان می باشد (کمینه کردن تاخیر دریافت سیگنال Acknowledgement از طرف مقصد در مبدا). در نتیجه مدت زمانی بسته در مبدا ارسال می شود تا زمانی که طول می کشد تا سیگنال Acknowledgement در مبدا دریافت شود میتواند به عنوان پاداش منفی (جریمه) در نظر گرفته شود. بطور مثال اگر بعد از ارسال بسته در مبدا ۳۰ ثانیه طول بکشد تا سیگنال Acknowledgement از طرف مقصد در مبدا دریافت شود، پاداش درگاه انتخاب شده در مسیریاب برای آن مقصد برابر ۳۰- می شود. بدیهی است که هر چی این مقدار بیشتر باشد، یعنی پاداش بیشتری (جریمه کمتری) دریافت کرده ایم و برایمان مطلوب تر است.

منظور از "حالات مختلف ورودی"، مفهوم Context است (میتوان آن را به State هم تعبیر کرد). در مساله فعلی، میخواهیم بهترین درگاه مسیریاب جهت ارسال بسته به مقصد را برای ۵ کشور مقصد مجزا - که پاداش دریافتی از یک درگاه یکسان برای دو کشور متفاوت می تواند تفاوت داشته باشد - پیدا کنیم. در نتیجه برای مساله ما پنج Context یا State متفاوت وجود دارد که برای هرکدام باید یک عامل یادگیر Multi-Armed Bandit را تمرین داد.

بخش ۲ – سوال پیاده سازی

در بخش سوال پیاده سازی، یک سوال وجود دارد که متشکل از ۵ بخش است. در این سوال، ابتدا برای یک مساله دنیای واقعی یک مدل مبتنی بر مساله Multi-Armed Bandit ارائه می دهیم. در بخش های بعدی نیز به پیاده سازی و بررسی الگوریتم های مختلف حل مساله Multi-Armed Bandit در شرایط مختلف می پردازیم.

سوال ۱

برای ارائه یک مدل مبتنی بر مساله Multi-Armed Bandit ، نیاز است تا مجموعه بازوها، پاداش و نحوه پاسخ دهی به مساله با توجه به حالات مختلف ورودی تبیین شود.

منظور از مجموعه بازوها، مجموعه عمل(Action) هایی است که می توانیم انجام دهیم. در مساله فعلی، عمل هایی که توانایی انجام آنها را داریم همان گزینه هایی هستند که برای انتخاب نوع تسهیلاتی که میخواهیم به افراد بدهیم؛ که این مجموعه، یک مجموعه سه عضوی متشکل از "تسهیلات ۵ میلیون تومانی" ، "تسهیلات ۲۰ میلیون تومانی" و "تسهیلات ۱۰۰ میلیون تومانی" است.

با توجه به صورت سوال، هدف از حل مسئله بیشینه کردن سود بانک از ارائه تسهیلات وام به افراد می باشد. در نتیجه ابزاری که تفاوت مقدار پول بازپرداخت تسهیلات داده شده از مقدار پول تسهیلات اختصاص شده را برای ما اندازه گیری می کند (کلاس Reward) می تواند به عنوان پاداش در نظر گرفته شود. بطور مثال، اگر یک مشتری ۲۰ میلیون تسهیلات وام بگیرد و بتواند مقدار ۱۵ میلیون از آن تسهیلات را برگرداند، مقدار پاداش بانک برابر ۵- =۲۰ - ۱۵ میلیون تومان می شود.

مقدار حسرت (Regret) برابر تفاوت مقدار مطلوب بانک از پازپرداخت تسهیلات (مقدار تسهیلات داده شده + کارمزد) از مقدار پول بازپرداخت تسهیلات داده شده تعریف شده است. بطور مثال، اگر یک مشتری ۲۰ میلیون تسهیلات وام بگیرد و بتواند مقدار ۱۵ میلیون از آن تسهیلات را برگرداند، با توجه به اینکه مقدار کارمزد تسهیلات ۲۰ میلیون تومانی برابر ۷۵۰ هزار تومان است، مقدار حسرت بانک برابر ۵.۷۵۰ = ۱۵ - (۰.۷۵۰ + ۲۰) میلیون تومان می شود.

منظور از "حالات مختلف ورودی"، مفهوم Context است (میتوان آن را به State هم تعبیر کرد). در مساله فعلی، میخواهیم بهترین نوع تسهیلات را برای ۳ گروه مشتری مجزا - که پاداش دریافتی از یک تسهیلات یک وام یکسان برای دو گروه مشتری متفاوت می تواند تفاوت داشته باشد - پیدا کنیم. در نتیجه برای مساله ما سه Context یا State متفاوت وجود دارد که برای هرکدام باید یک عامل یادگیر Multi-Armed Bandit را تمرین داد.

سوال ۲

هدف سوال

در این سوال به پیاده سازی محیطی که برای مسئله در قسمت قبل ارائه کردیم پرداختیم.

توضیح پیاده سازی

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Part 2 - Environment** قرار دارد.

کلاس **Environment** یک نمونه از کلاس **Reward** را با توجه به نوع مشتری در خود دارد. در این کلاس پاداش عمل عامل در محیط محاسبه و به عامل اطلاع داده می شود.

تابع ***calc_reward*** پاداش عملی که عامل انجام داده است را توسط کلاس **Reward** محاسبه و خروجی می دهد.

تابع ***get_available_actions*** عمل های مجازی که عامل می تواند انجام دهد را خروجی می دهد.

روند اجرای کد پیاده‌سازی

برای اجرا کد های این بخش لازم است تا تمام کد های ماقبل این بخش و کد های این بخش را اجرا کنید.

سوال ۳

هدف سوال

به ازای هرکدام از الگوریتم های Epsilon-Greedy، Gradient-Based و Upper Confidence Bound ، یک عامل یادگیر Multi-Armed Bandit پیاده سازی کردیم.

توضیح پیاده سازی

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Part 3** قرار دارد.

عامل پایه (Base)

در این قسمت به پیاده سازی عامل پایه پرداختیم. کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Part 3 - Agent Base** قرار دارد.

تابع ***calculate_utility*** برای محاسبه مطلوبیت با توجه به رابطه ارائه شده در صورت تمرین پیاده سازی شده است. این تابع مقدار پاداش را می گیرد و مقدار utility این پاداش را محاسبه و خروجی می دهد.

تابع ***choose_action*** برای انتخاب اکشن مناسب به کار می رود. این تابع با توجه به سیاستی که برای انتخاب اکشن داریم، یک اکشن را انتخاب می کند و خروجی می دهد. جزییات پیاده سازی این تابع با توجه به نوع عامل و سیاستی که هر عامل در انتخاب اکشن دارد که داریم در بخش مربوط به پیاده سازی این تابع برای هر عامل در فایل ***Codes.ipynb/html*** قسمت **Part 3** قابل مشاهده است.

تابع ***take_action*** برای انجام اکشن به کار می رود. ورودی این تابع اکشن انتخاب شده است. این تابع ابتدا پاداش اکشن انتخاب شده را از محیط دریافت می کند. سپس مقدار utility پاداش دریافت شده را محاسبه می کند. سپس به آپدیت value هایی که داریم توسط تابع *update_value* می پردازد. در نهایت مقدار پاداش و utility را خروجی می دهد.

تابع ***update_value*** به بروزرسانی متغییر هایی که برای هر عامل داریم می پردازد. جزییات پیاده سازی این تابع با توجه به نوع عامل و متغییر هایی که برای هر عامل داریم در بخش مربوط به پیاده سازی این تابع برای هر عامل در فایل ***Codes.ipynb/html*** قسمت **Part 3** قابل مشاهده است.

تابع ***step*** به یک بار به طور کامل عملیات یادگیری را انجام می دهد. ورودی این تابع تعداد trial ها است. این تابع در هر تریال، اکشن مناسب را توسط تابع *choose_action* انتخاب می کند. سپس مقدار

پاداش و utility حاصل از انجام آن اکشن را توسط تابع *take_action* محاسبه می کند. در انتها این تابع مقدار حسرت را با توجه به تعریفی که در قسمت ۱ ارائه دادیم محاسبه می کند و مقدار پاداش و حسرت این تریال را در متغییر های مربوطه دخیره می کند.

تابع ***reset*** تمام متغییر هایی که در محاسبات یادگیری یک عامل به کار رفته اند را مقدار دهی اولیه می کند. جزییات پیاده سازی این تابع با توجه به نوع عامل و متغییر هایی که برای هر عامل داریم در بخش مربوط به پیاده سازی این تابع برای هر عامل در فایل ***Codes.ipynb/html*** قسمت **Part 3** قابل مشاهده است.

تابع ***get_rewards*** پاداش های حاصل از یادگیری عامل در هر تریال را خروجی می دهد.

تابع ***get_regret*** حسرت های حاصل از یادگیری عامل در هر تریال را خروجی می دهد.

عامل Epsilon-Greedy

در این قسمت به پیاده سازی عامل Epsilon-Greedy پرداختیم. کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Part 3 – Epsilon Greedy Agent** قرار دارد.

تعریف متغیر های خاص این عامل برای یادگیری در تابع ***__init__*** کلاس ***EpsilonGreedyAgent*** در فایل ***Codes.ipynb/html*** قسمت **Part 3 – Epsilon Greedy Agent** به صورت کامنت آورده شده است.

جزییات پیاده سازی سه تابع ***choose_action*** ، ***update_value*** ، ***reset*** کلاس ***EpsilonGreedyAgent*** در فایل ***Codes.ipynb/html*** قسمت**Part 3 – Epsilon Greedy Agent** قابل مشاهده است.

- **choose action :**

$$Action\ at\ time(t) : \pi_{act}(s) = \begin{cases} argmax_{a \in Actions} \hat{Q}_{opt}(s,a) & probability\ 1-\epsilon \\ random\ from\ Actions(s) & probability\ \epsilon \end{cases}$$

- **update value :**

$$Number\ of\ doing\ an\ Action : N(A) \leftarrow N(A) + 1$$

$$Action - Value\ Fuction : Q(A) \leftarrow Q(A) + \frac{1}{N(A)} \times [R(A) - Q(A)]$$

عامل Gradient-Based

در این قسمت به پیاده سازی عامل Gradient-Based پرداختیم. کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Part 3 – Gradient Based Agent** قرار دارد.

تعریف متغیر های خاص این عامل برای یادگیری در تابع ***__init__*** کلاس ***GradientBasedAgent*** در فایل ***Codes.ipynb/html*** قسمت **Part 3 – Gradient Based Agent** به صورت کامنت آورده شده است.

جزییات پیاده سازی سه تابع ***choose_action*** ، ***update_value*** ، ***reset*** کلاس ***GradientBasedAgent*** در فایل ***Codes.ipynb/html*** قسمت **Part 3 – Gradient Based Agent** قابل مشاهده است.

- **choose action :**

$$Action\ at\ time(t) : \pi_{act}(s) = random\ from\ Actions(s) \quad , \quad probability\ Pr\{A_t = a\}$$

- **update value :**

$$Number\ of\ Trials : N_t \leftarrow N_t + 1$$

$$Average Reward : \bar{R}_t \leftarrow \bar{R}_t + \frac{1}{N_t} \times \left[R_t - \bar{R}_t\right]$$

$$Action - Preferences\ Function : H_t(A) = \begin{cases} H_{t+1}(A_t) \doteq H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)) & and \\ H_{t+1}(a) \doteq H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a) & for\ all\ a \neq A_t \end{cases}$$

$$Probability\ of\ an\ Action : Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

عامل Upper Confidence Bound

در این قسمت به پیاده سازی عامل Upper Confidence Bound پرداختیم. کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Part 3 – Upper Confidence Bound Agent** قرار دارد.

تعریف متغیر های خاص این عامل برای یادگیری در تابع ***__init__*** کلاس ***UpperConfidenceBoundAgent*** در فایل ***Codes.ipynb/html*** قسمت **Part 3 – Upper Confidence Bound Agent** به صورت کامنت آورده شده است.

جزییات پیاده سازی سه تابع ***choose_action*** ، ***update_value*** ، ***reset*** کلاس ***UpperConfidenceBoundAgent*** در فایل ***Codes.ipynb/html*** قسمت **Part 3 – Upper Confidence Bound Agent** قابل مشاهده است.

- **choose action :**

$$Action\ at\ time(t) : \pi_{act}(s) = argmax_{a \in Actions} A_t(a)$$

- **update value :**

$$Total\ Trials : t \leftarrow t + 1$$

$$Number\ of\ times\ that\ Action\ a\ has\ been\ selected : N(A) \leftarrow N(A) + 1$$

$$Action - Value\ Fuction : Q(A) \leftarrow Q(A) + \frac{1}{N(A)} \times [R(A) - Q(A)]$$

$$Upper\ Confidence\ Bound\ for\ each\ Action : A_t \doteq \underset{a}{argmax} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

روند اجرای کد پیاده‌سازی

برای اجرا کد های این بخش لازم است تا تمام کد های ماقبل این بخش و کد های این بخش را اجرا کنید.

سوال ۴

هدف سوال

در این سوال به اجرا هر یک از عامل های یادگیری بر روی گروه مشتریان مدنظر می پردازیم.

توضیح پیاده سازی

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Part 4** قرار دارد.

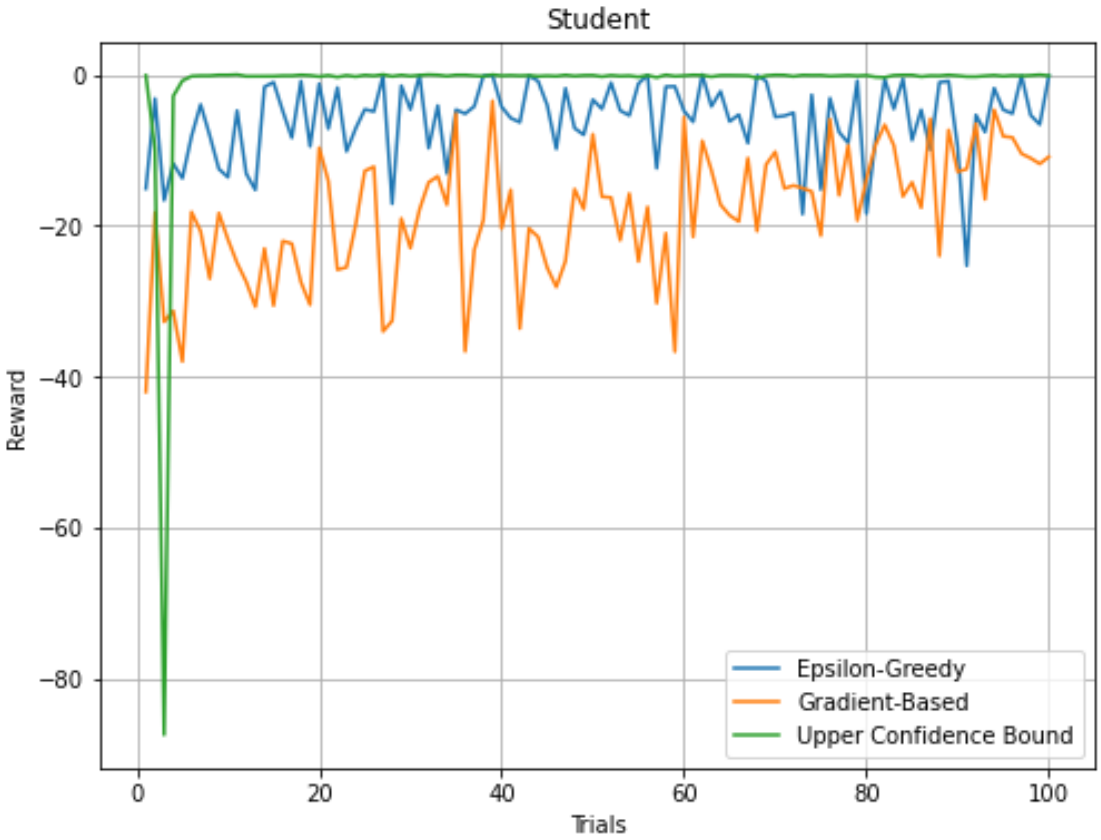
تابع ***draw_ylabel_trial_plot_part_4*** برای نمایش نمودار میانگین پاداش دریافتی و میانگین مقدار پشیمانی یادگیری هر یک از سه عامل بر روی یک گروه از مشتریان می پردازد.

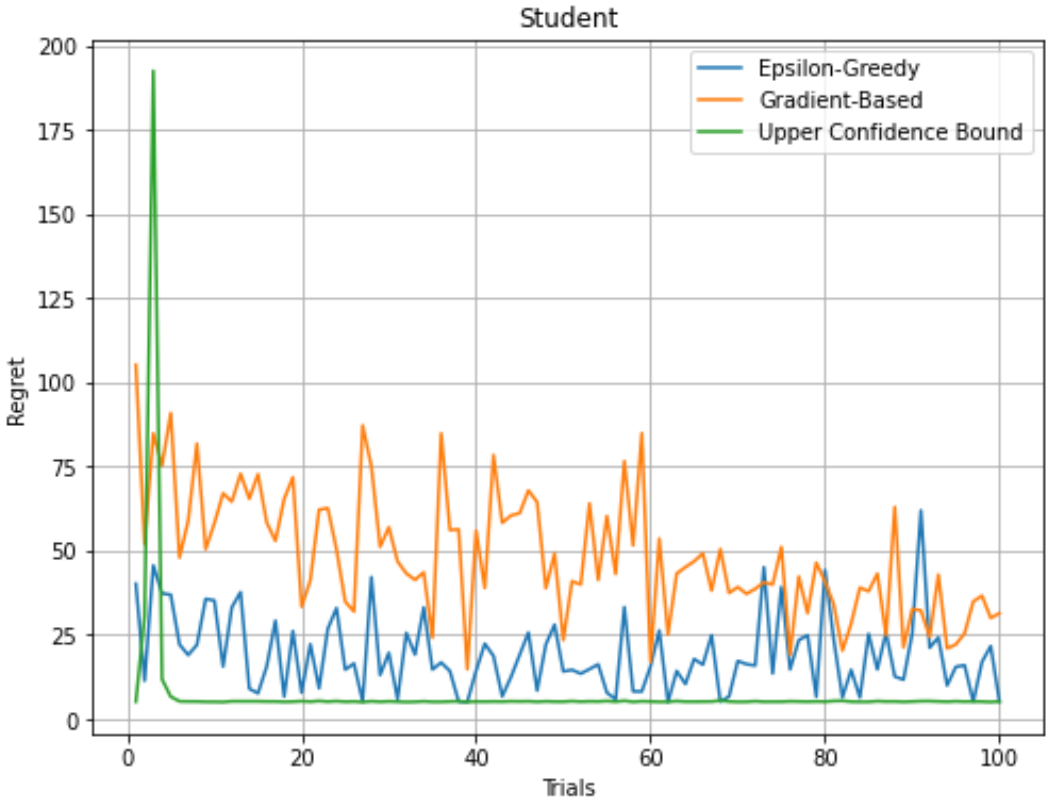
تابع ***run_part_4*** برای اجرا عمل یادگیری هر یک از سه عامل بر روی یک گروه از مشتریان پیاده سازی شده است. این تابع نام و نوع پاداش مشتری، مقادیر هایپرپارامتر های هر یک از سه عامل یادگیری، تعداد دفعات اجرا هر الگوریتم و تعداد تریال ها را ورودی می گیرد. سپس به دفعات اجرا (number of run)، هر یک از سه الگوریتم یادگیری با تعداد تریال و هایپرپارامتر های مدنظر اجرا می کند و میانگین پاداش و پشمیانی دریافتی هر الگوریتم در هر بار اجرا الگوریتم ها را محاسبه می کند. در انتها این تابع به رسم نمودار میانگین پاداش دریافتی و میانگین مقدار پشیمانی یادگیری هر یک از سه عامل بر روی یک گروه از مشتریان با استفاده از تابع *draw_ylabel_trial_plot_part_4* می پردازد.

نتایج

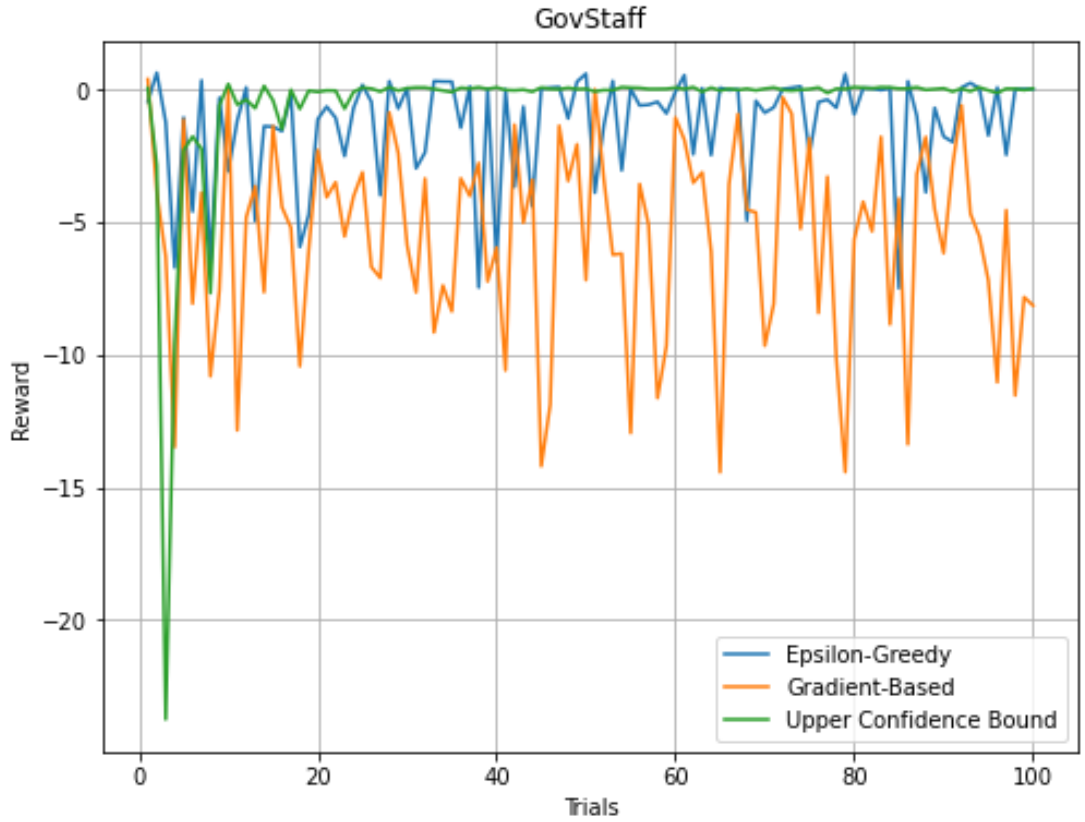
نتایج مربوط به یادگیری سه عامل با مقادیر اپسیلون برابر ۰.۲ ، نرخ یادگیری (Learning Rate) برابر ۰.۰۰۱ و c برابر ۲ به عنوان هایپرپارامتر هر یک از سه الگوریتم یادگیری بر روی هر گروه از مشتریان در ادامه آورده شده است.

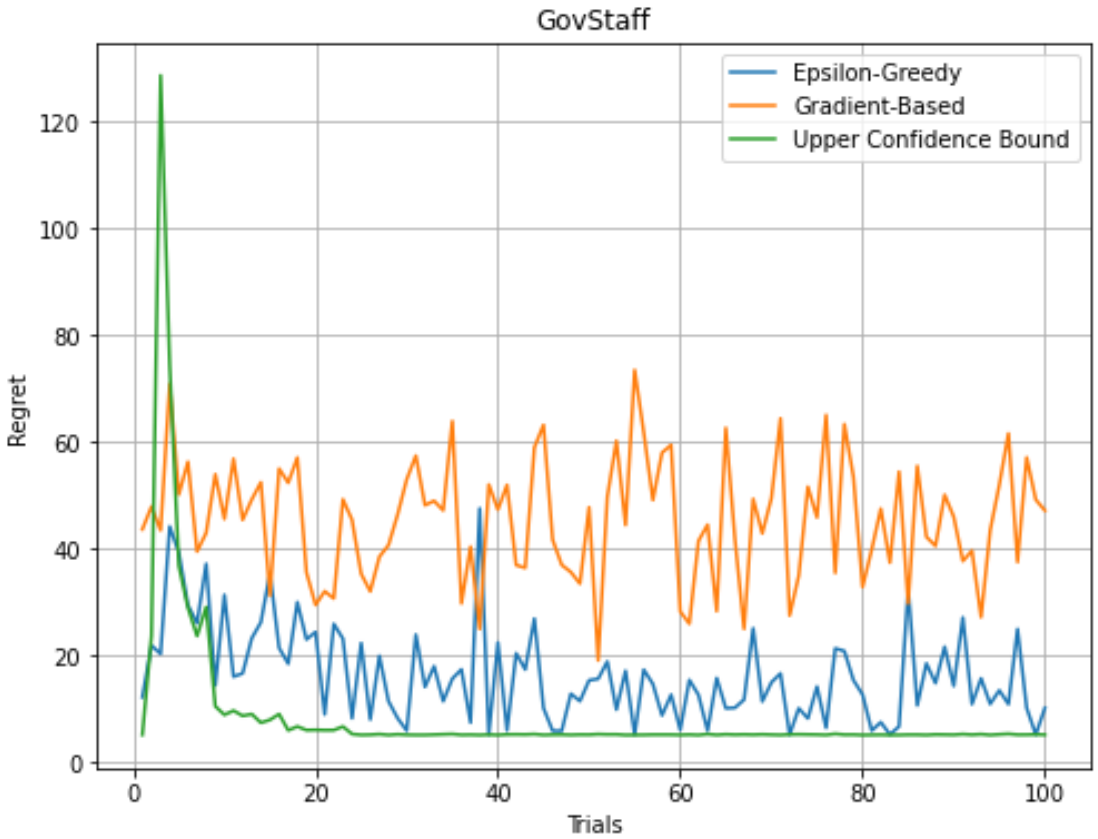
گروه مشتری دانشجویان (Student)



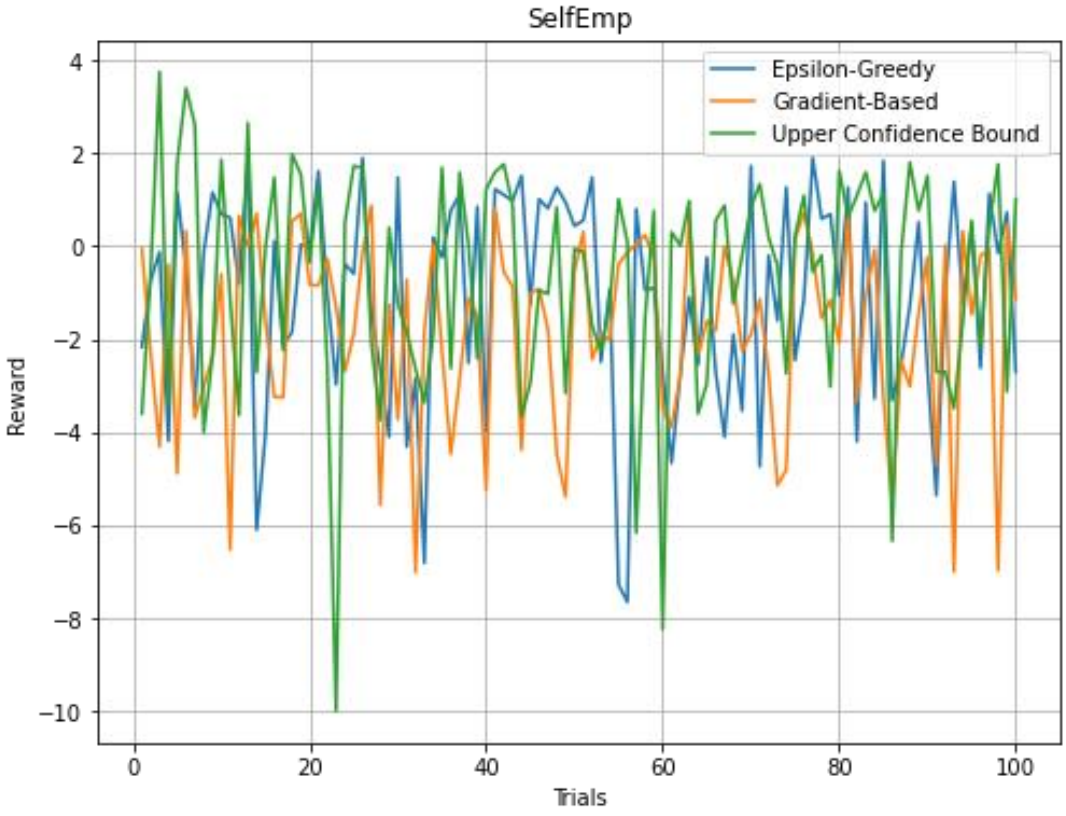


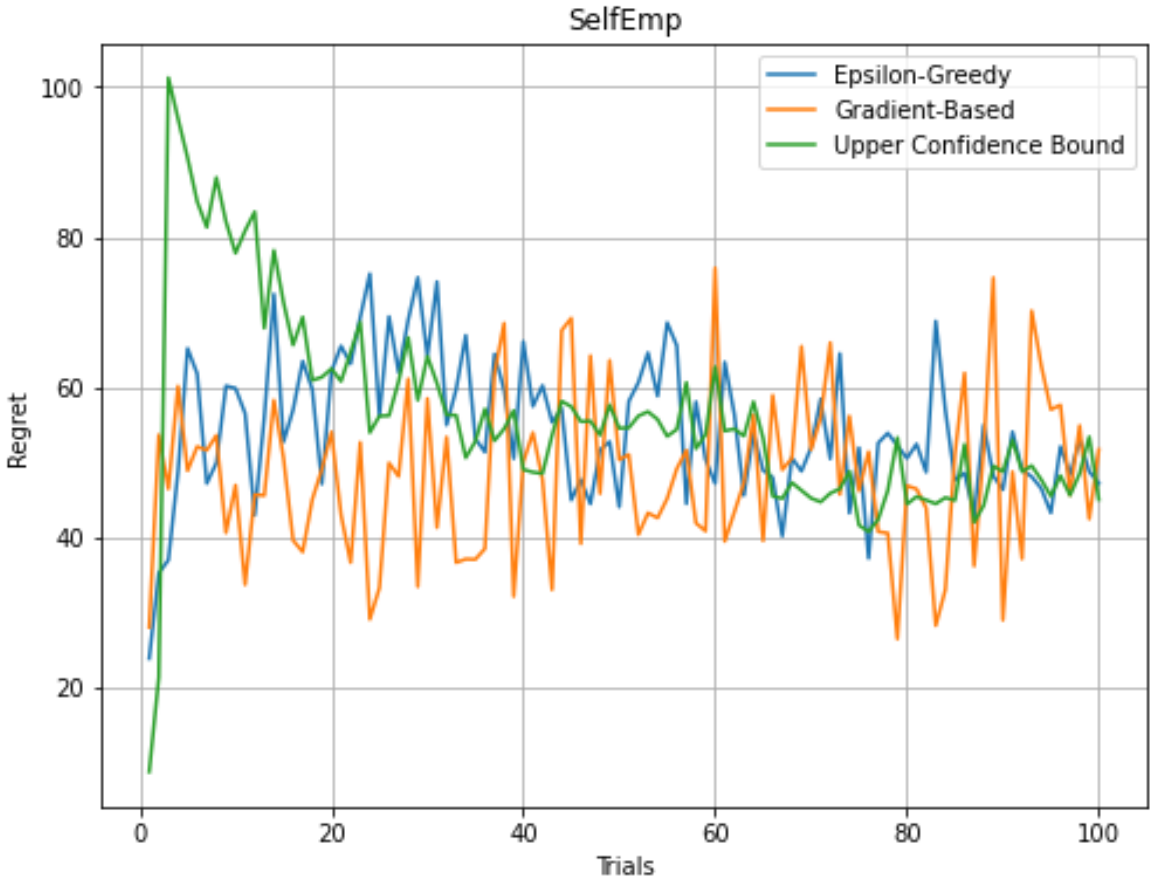
گروه مشتری کارمندان دولتی (GovStaff)





گروه مشتری صاحبان مشاغل آزاد (SelfEmp)





روند اجرای کد پیاده‌سازی

برای اجرا کد های این بخش لازم است تا تمام کد های ماقبل این بخش و کد های این بخش را اجرا کنید.

سوال ۵

هدف سوال

در این سوال به بررسی و پیدا کردن نرخ یادگیری بهینه برای الگوریتم Gradient-Based پرداختیم.

توضیح پیاده سازی

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Part 5** قرار دارد.

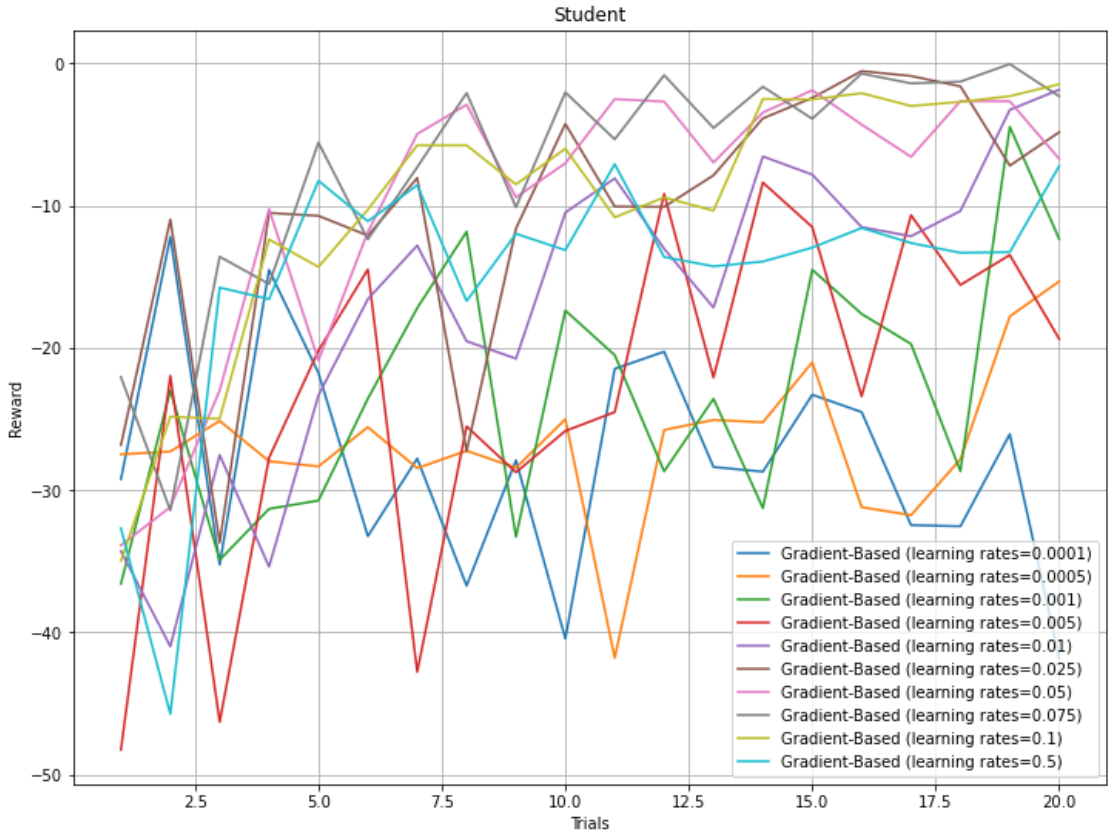
تابع ***draw_ylabel_trial_plot_part_5*** برای نمایش نمودار میانگین پاداش دریافتی و میانگین مقدار پشیمانی یادگیری عامل الگوریتم Gradient-Based با نرخ های یادگیری مختلف بر روی یک گروه از مشتریان می پردازد.

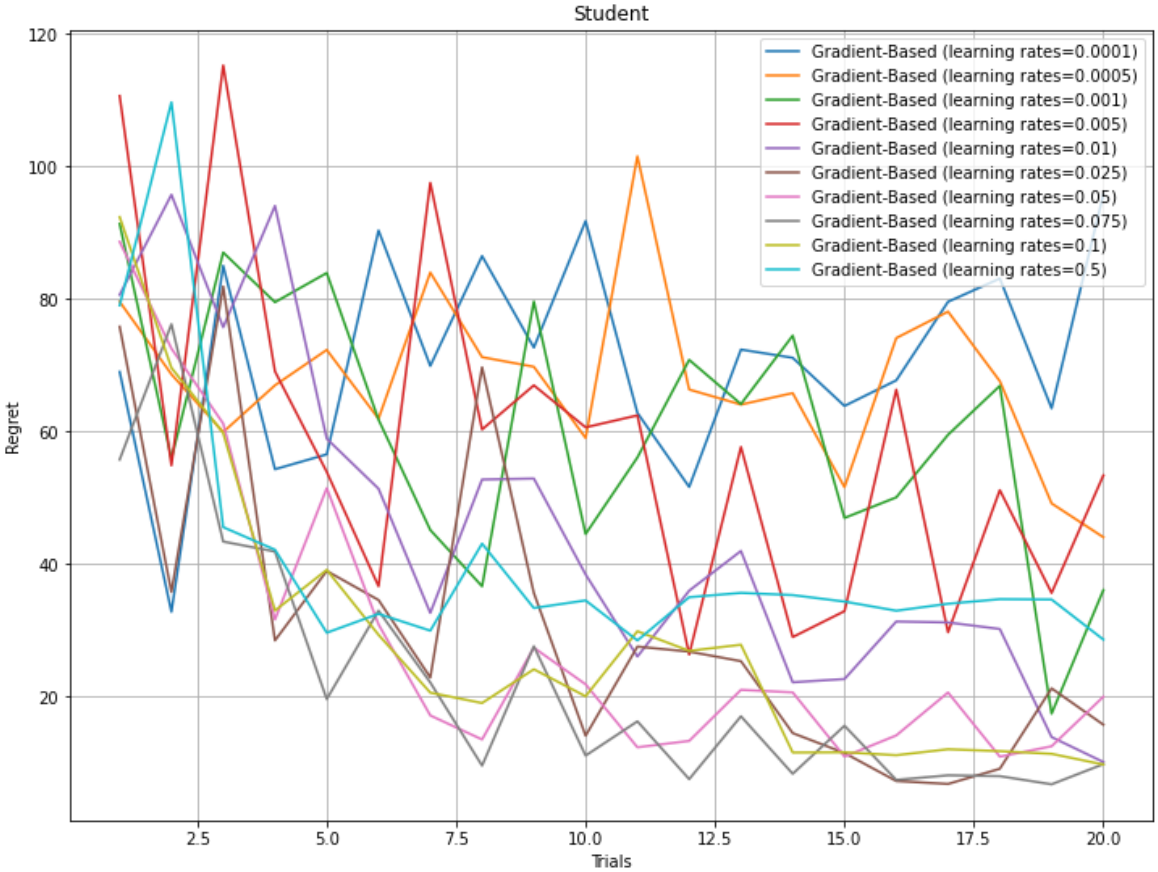
تابع ***run_part_5*** برای اجرا عمل یادگیری عامل الگوریتم Gradient-Based با نرخ های یادگیری مختلف بر روی یک گروه از مشتریان پیاده سازی شده است. این تابع نام و نوع پاداش مشتری، مقادیر نرخ های یادگیری عامل Gradient-Based ، تعداد دفعات اجرا الگوریتم و تعداد تریال ها را ورودی می گیرد. سپس به ازای هر یک از نرخ های یادگیری، به دفعات اجرا (number of run)، الگوریتم Gradient-Based را با تعداد تریال و نرخ یادگیری مدنظر اجرا می کند و میانگین پاداش و پشمیانی دریافتی الگوریتم در هر بار اجرا الگوریتم را برای یک نرخ یادگیری محاسبه می کند. در انتها این تابع به رسم نمودار میانگین پاداش دریافتی و میانگین مقدار پشیمانی یادگیری عامل Gradient-Based بر روی یک گروه از مشتریان با استفاده از تابع *draw_ylabel_trial_plot_part_5* می پردازد.

نتایج

هدف این سوال، یافتن تسهیلات بهینه برای هر کدام از گروه مشتریان است. تسهیلاتی بهینه است که مشتری با احتمال بالاتری موفق به بازپرداخت آن بشود؛ یعنی مقدار پشیمانی کمتری برای بانک ایجاد کند. در نتیجه باید مقدار نرخ یادگیری ای را انتخاب کنیم که در نمودار میانگین پشیمانی بانک، نرخ پیشمانی به حداقل برسد.

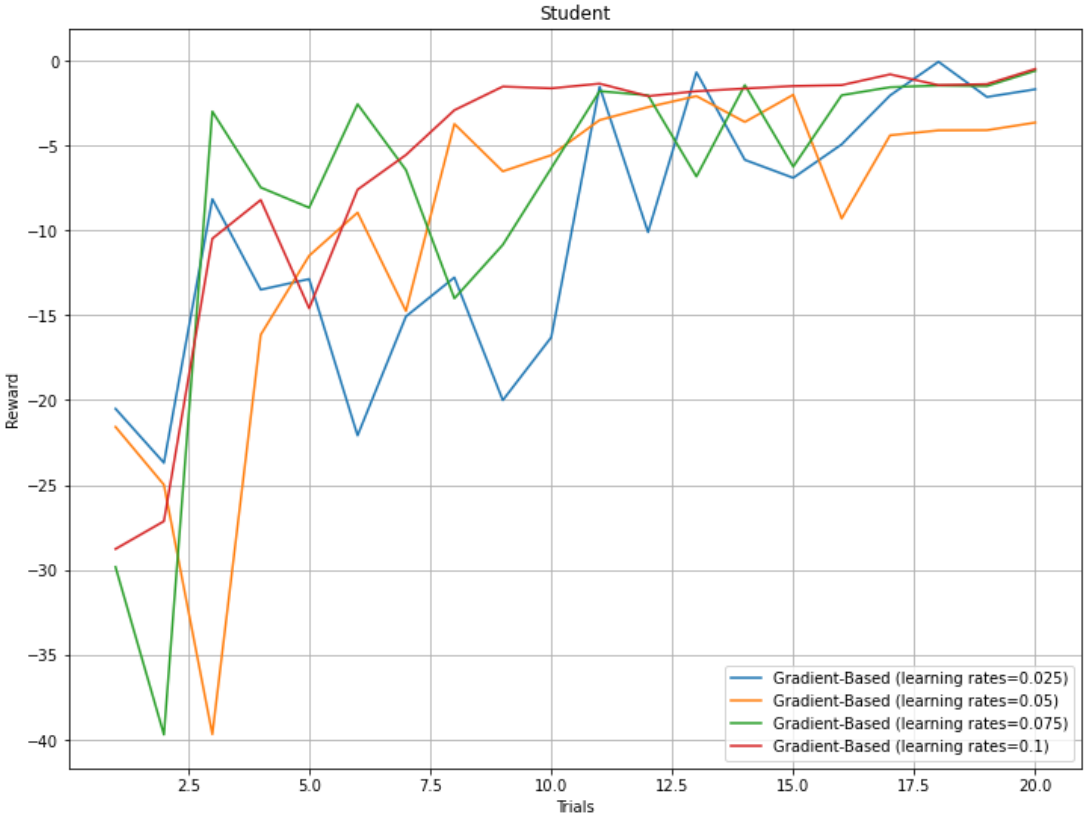
برای این سوال ما از مقادیر متفاوتی برای نرخ یادگیری استفاده کردیم که بطور مثال عملکرد نرخ های یادگیری انتخاب شده بر روی گروه مشتری دانشجویان به شکل زیر است:

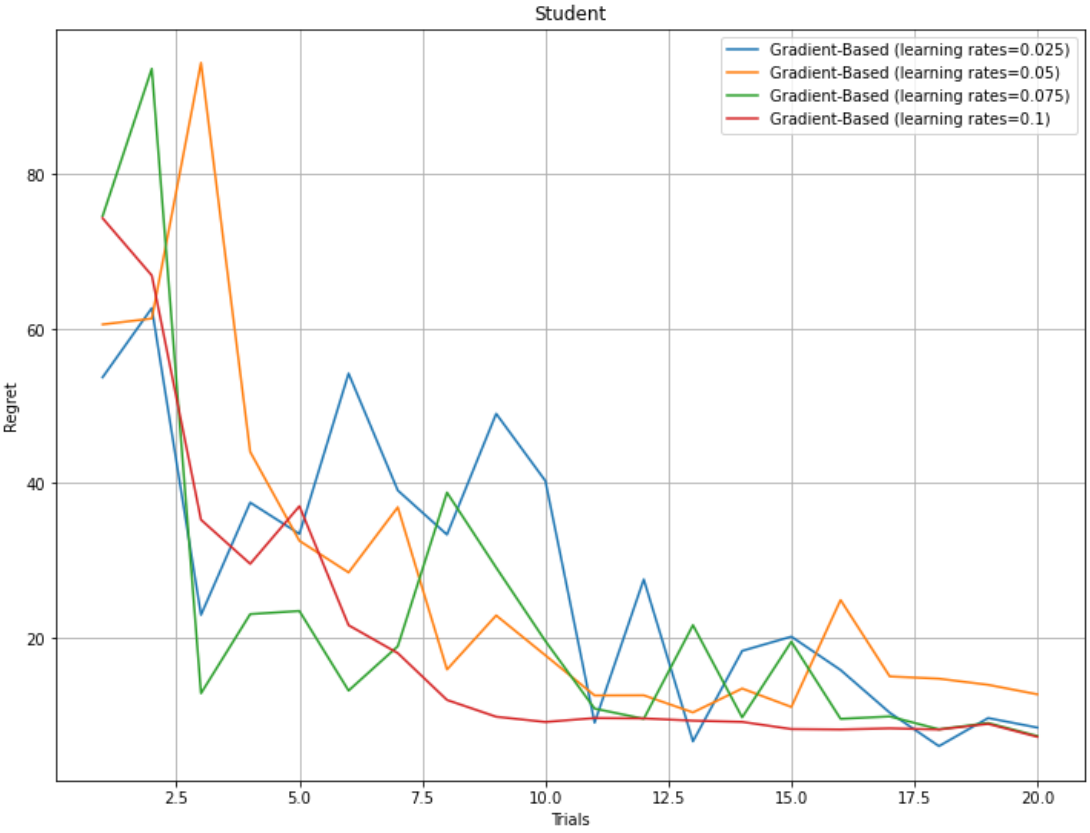

Student
Reward
Trials
0
−10
−20
−30
−40
−50
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Gradient-Based (learning rates=0.0001)
Gradient-Based (learning rates=0.0005)
Gradient-Based (learning rates=0.001)
Gradient-Based (learning rates=0.005)
Gradient-Based (learning rates=0.01)
Gradient-Based (learning rates=0.025)
Gradient-Based (learning rates=0.05)
Gradient-Based (learning rates=0.075)
Gradient-Based (learning rates=0.1)
Gradient-Based (learning rates=0.5)

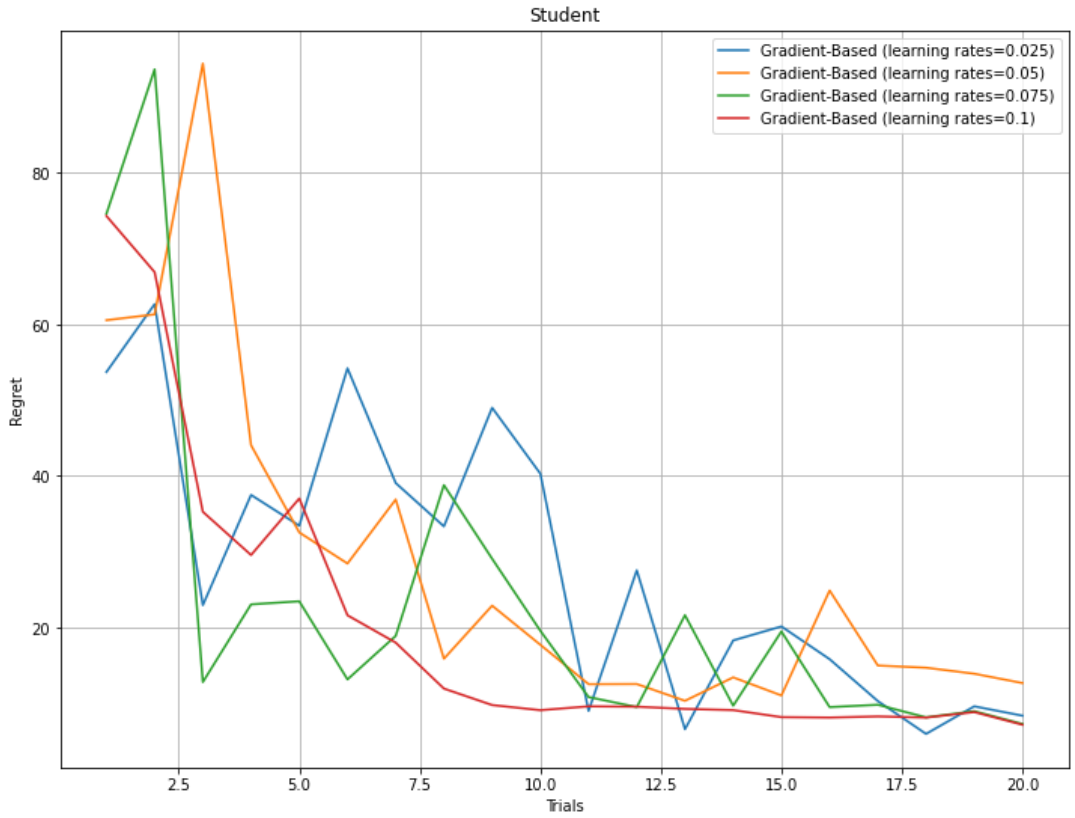

Student
Regret
Trials
120
100
80
60
40
20
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Gradient-Based (learning rates=0.0001)
Gradient-Based (learning rates=0.0005)
Gradient-Based (learning rates=0.001)
Gradient-Based (learning rates=0.005)
Gradient-Based (learning rates=0.01)
Gradient-Based (learning rates=0.025)
Gradient-Based (learning rates=0.05)
Gradient-Based (learning rates=0.075)
Gradient-Based (learning rates=0.1)
Gradient-Based (learning rates=0.5)

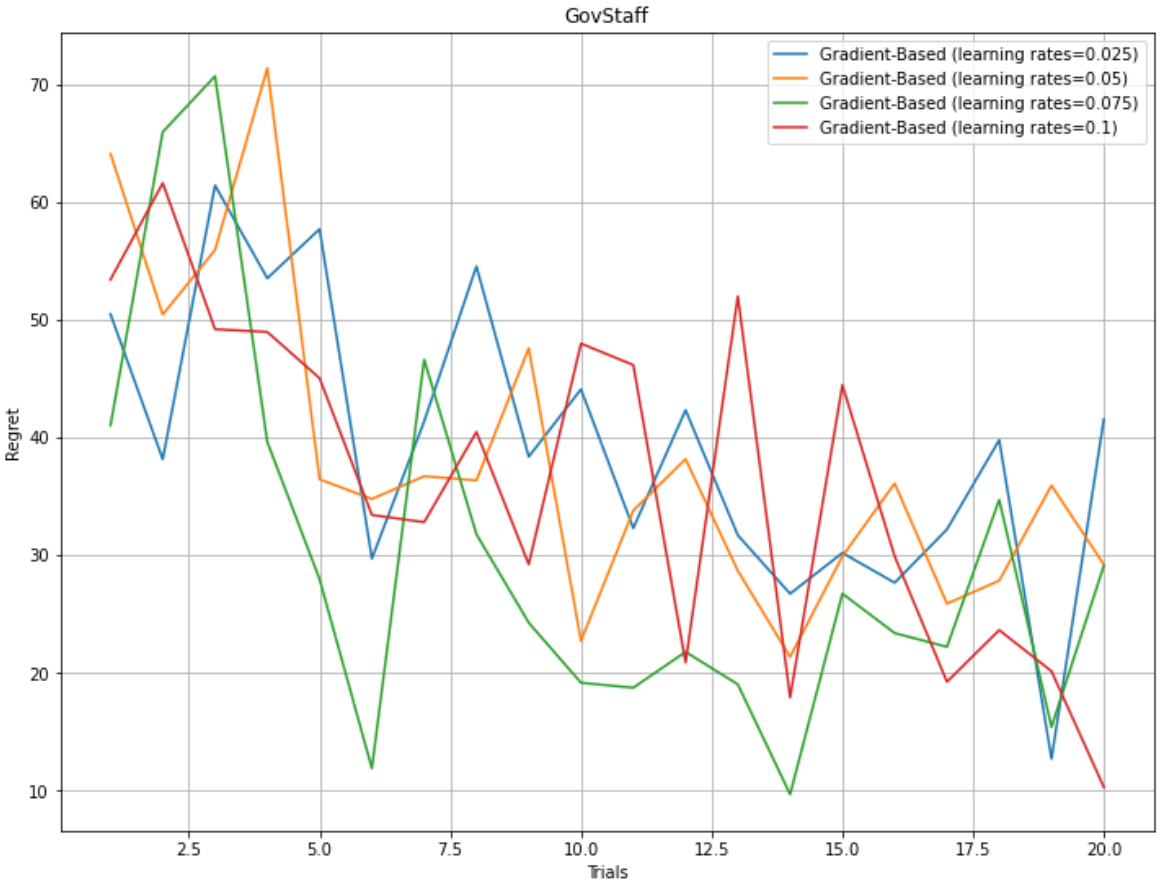
در اینترنت پیشنهاد شده بود که نرخ یادگیری بهتر است بین ۰.۰۱ و ۰.۱ باشد. در نتیجه با توجه به نمودار های بالا و مطالعاتی که در اینترنت داشتم، تصمیم گرفتم از چهار مقدار ۰.۰۲۵، ۰.۰۵، ۰.۰۷۵ و ۰.۱ به عنوان نرخ یادگیری الگوریتم Gradient-Based استفاده بکنم. نتایج مربوط به اجرا الگوریتم Gradient-Based برای هر کدام از گروه های مشتریان در ادامه آورده شده است:

گروه مشتری دانشجویان (Student)

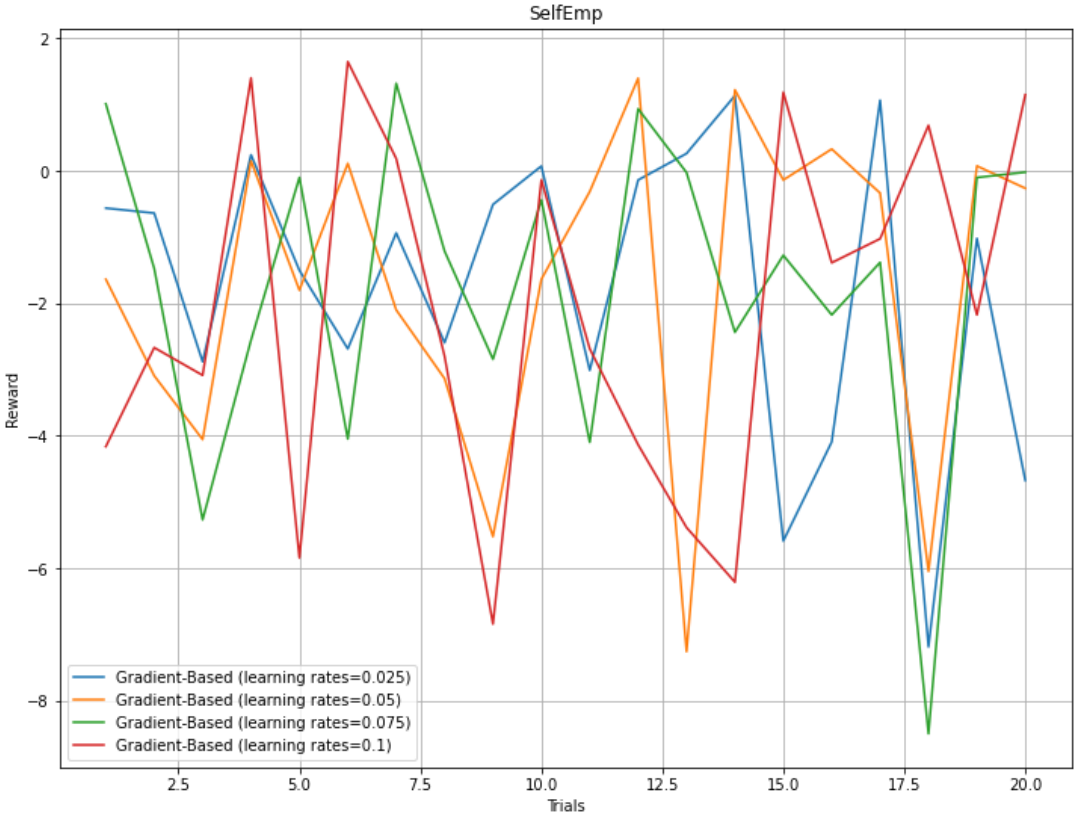


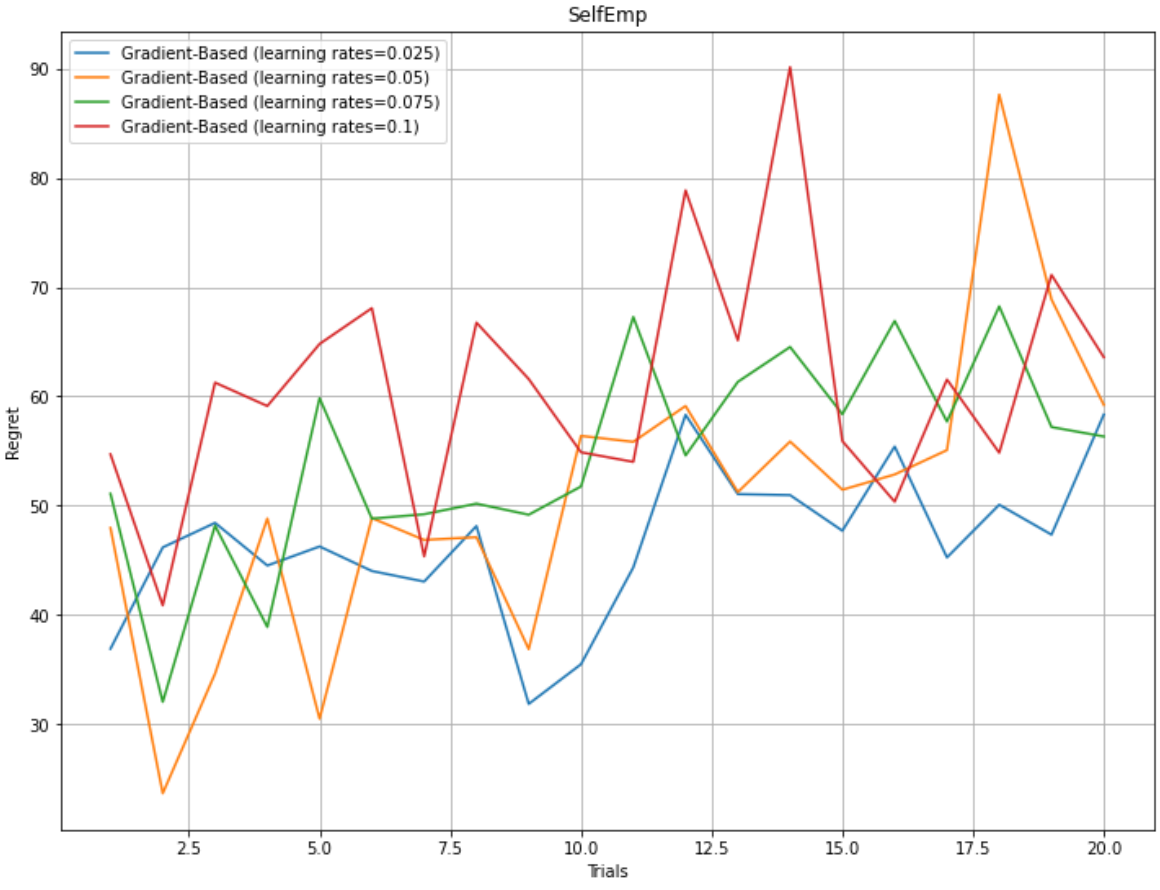



Student
Gradient-Based (learning rates=0.025)
Gradient-Based (learning rates=0.05)
Gradient-Based (learning rates=0.075)
Gradient-Based (learning rates=0.1)
Regret
Trials
20
40
60
80
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


GovStaff
Gradient-Based (learning rates=0.025)
Gradient-Based (learning rates=0.05)
Gradient-Based (learning rates=0.075)
Gradient-Based (learning rates=0.1)
Regret
Trials
10
20
30
40
50
60
70
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

گروه مشتری صاحبان مشاغل آزاد (SelfEmp)





نتیجه گیری

با توجه به نمودار های بالا، بهترین نرخ یادگیری الگوریتم Gradient-Based جهت به حداقل رساندن میزان پشیمانی بانک و پیدا کردن تسهیلات بهینه برای گروه مشتری دانشجویان، کارمندان دولتی و صاحبان مشاغل آزاد به ترتیب برابر **۰.۱** ، **۰.۰۷۵ و ۰.۰۲۵** می باشد. بین این سه مقدار نیز بطور میانگین مقدار **۰.۰۷۵** عملکرد بهتری در مجموع سه گروه مشتری در به حداقل رساندن میزان پشیمانی بانک داشت.

روند اجرای کد پیاده‌سازی

برای اجرا کد های این بخش لازم است تا تمام کد های ماقبل این بخش و کد های این بخش را اجرا کنید.

منابع

1. https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/
2. https://www.baeldung.com/cs/epsilon-greedy-q-learning
3. https://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/
4. https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/#:~:text=A%20traditional%20default%20value%20for,starting%20point%20on%20your%20problem.&text=this%20default%20value-,%E2%80%94%20Practical%20recommendations%20for%20gradient%2Dbased,training%20of%20deep%20architectures%2C%202012.
5. https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0
6. https://www.andreaperlato.com/theorypost/the-learning-rate/