گزارش تمرین شماره 2

نام و نام خانوادگی	حمیدرضا علی اکبری خویی
شماره دانشجویی	810196514

فهرست

سوال یک..... 3

درصد استفاده از تصمیم بهینه..... 3

بررسی Regret..... 5

سوال دو..... 6

تعریف مدل..... 6

تعریف پاداش ها..... 6

استفاده از عامل $\epsilon - greedy$..... 8

استفاده از عامل UBC..... 10

Cross validation..... 12

سوال سه..... 15

مدل سازی..... 15

توسعه عامل e-greedy..... 16

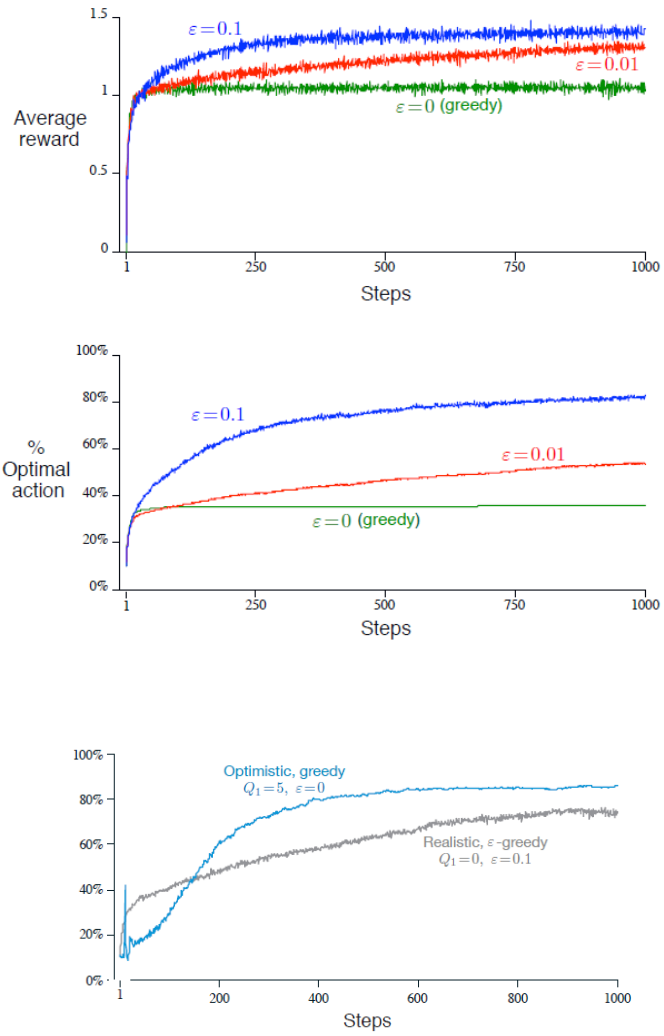
عامل gradient..... 18

مقایسه عامل ها برای مدل..... 20

سوال یک

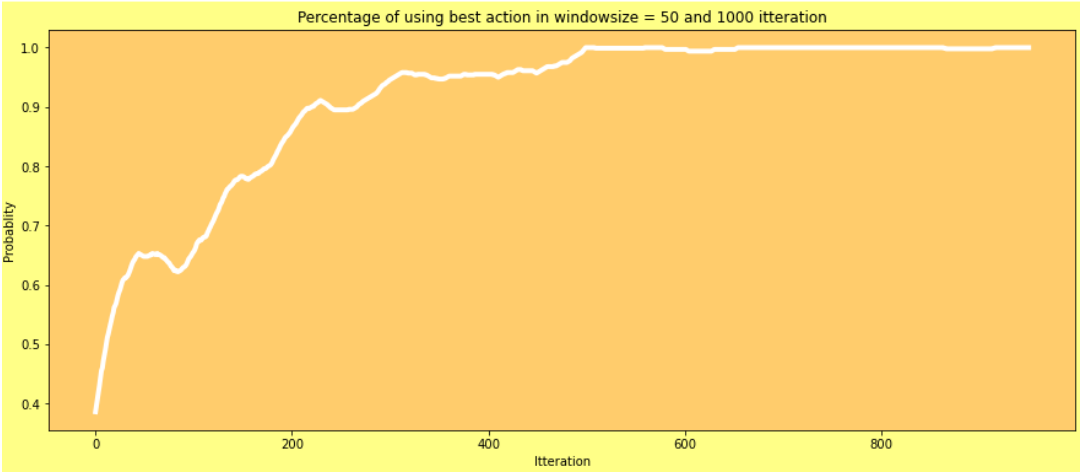
درصد استفاده از تصمیم بهینه

در کتاب Saton مطابق شکل های 2-2 و 3-2 کتاب داریم :



1Figure

با توجه به نمودار های بدست آمده برای الگوریتم Thompson_sampling طبق عامل یاده سازی شده برای سوال یک نتایج شکل 2 بدست آمد:



2Figure

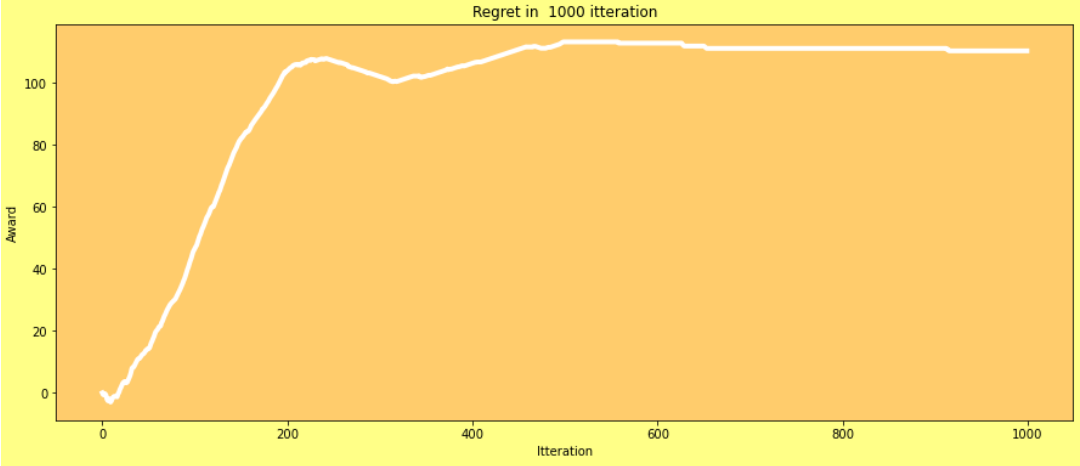
که مشخصا با استفاده از عامل Thompson_sampling به نمودار درصد استفاده از تصمیم بهینه بهتری رسیدیم که حتی با توجه به 20 بار اجرا، چون که درصد استفاده از تصمیم بهینه دارای یک توزیع است و آماره است به همین دلیل در 20 بار اجرا که در ابتدای هر بار اجرا سیستم به تنظیمات اولیه مثل حالت سکون برمیگشت، با استفاده از میانگین بر روی 20 بار اجرا میتوانیم تخمین خوبی از این عملگر(درصد استفاده از تصمیم بهینه داشته باشیم) البته در این الگوریتم مذکور با تقریبا خوبی در حدود تصمیم 500 عامل توانسته است به طور کامل به استفاده از تصمیم بهینه روی آورد، در صورتی که برای شکل 2-2 کتاب با استفاده از عامل e_greedy حتی به درصد بالایی هم دست پیدا نکرده است، این به این معنی است که الگوریتم Thompson_sampling خیلی بهتر از e-greedy کار میکند.

باید توجه داشت که باتوجه به نحوه کار الگوریتم Thompson_sampling میتوان این را دریافت که در واقع توزیع هایی از پاداش ها را سعی میکنیم تخمین بزنیم که البته این توزیع ها توزیع های نرمالی هستند.

و با هر بار به روز رسانی پارامتر های مربوط به توزیع هر بازو، آن توزیع جمع تر میشود، و اینکار باعث میشود که در حالت حدی انتخاب به تعداد نامحدود آن بازو به عنوان تصمیم بهینه، در نهایت به یک حالت ضربه ای برسیم که فقط آن مورد انتخاب میشود. فقط باید توجه داشت که چون نوع پاداش ها برای 10 بازو این مساله یکی است همه دارای میانگین و واریانس یکسانی هستند، پس در حقیقت هر چه قدر هم با استفاده از q_value ها مدل را بروز رسانی کنیم، باز هم هر کدام از بازو ها انتخاب بشود، در نهایت میزان اداشی که خواهیم گرفت دارای توزیع گوسی با میانگین و واریانس یکسان برای همه حالات خواهد بود. برای هر بار اجرا انتخاب یک بازو کاملا random است، چرا که هر کدام از بازوها توالی اداش های بزرگتری در همان iteration های اول داشته باشد، Q_value مربوط به آن بازو دارای میانگین بیشتری خواهد بود و از احتمال انتخاب به عنوان بازو بهینه بیشتری برخوردار است.

بررسی Regret

در رابطه با میزان Regret با توجه به شکل 3 داریم :



3Figure

در نمودار حاصل از مدلسازی مربوطه سوال اول که نشان دهنده Regret است، باید در ابتدا تصمیم بهینه مشخص بشود و بعد در رابطه با Regret حرف زده شود. برای مثال این شکل نشان میدهد که در نهایت همانند شکل 2 تصمیم بهینه برای افق 1000 تلاش و 20 بار اجرا، در حدود تلاش 600-700 به تصمیم بهینه همگرا میشود. همگرایی در نمودار Regret را اینگونه تعریف میکنیم که باید از یک نقطه مشخصی به بعد نمودار به حالت خطی با شیب صفر دربیاید، چرا که با توجه به تعریف، این عمل نشان میدهد که آن دسته از Reward هایی که میگیریم خیلی نزدیک به Reward بهینه است. اگر کل Reward های گرفته شده به صورت میانگین برای20 بار اجرا را بررسی کنیم، متوجه میشویم که چون همه توزیع پاداش ها یکسان است و در طول زمان تغییری نمیکند، پس سوای انتخاب هر تصمیمی، پاداش برای هر تلاش از یک توزیع خاصی می آید؛ یعنی احتمال دارد که مثلا برای تصمیم آخر مقدار پاداشی که گرفته اید منفی شده باشد ولی مثلا برای تصمیم تلاش 200 ام این مقدار بیشینه کل تلاش ها باشد! شکل کلی نمودار Regret نشان میدهد که در حالت کلی به یک فرم لگاریتیمی نزدیک میشویم که این خبر خوبی است. روش e_greedy دارای نوسان زیادی حول نقطه همگرایی است، ولی برای الگوریتم Thompson برای نمونه برداری، این مقدار خیلی کم است و حتی سرعت همگرایی مدل Thompson هم باتوجه به شکل های کتاب، بسیار بیشتر است.

سوال دو

تعریف مدل

با توجه به این که ما ساعت 7:30 کلاسمان شروع میشود و میزان زمانی که با استفاده از وسایل نقلیه لازم داریم که به کلاس برسیم، 25 دقیقه است؛ پس ما از زمان رسیدن به میدان انقلاب(6:55) تا زمان سوار شدن به هر وسیله نقلیه (اتوبوس سا تاکسی) برای رسیدن به دانشکده، حد اکثر 10 دقیقه میتوانیم صبر کنیم، وگرنه به کلاس دیر میرسیم. با توجه به این قضیه reward ها و utility-function ها را تعریف میکنیم.

برای این که نهایتا 10 دقیقه میتوانیم صبر کنیم، تعریف 12 بازو برای مدل کار دور از ذهنی نیست، چرا که اگر بازوی اول را متناظر با زمانمنتظر ماندن به میزان 0 دقیقه و تا بازوی 11ام را برای منتظر ماندن برای 10 دقیقه مدل بکنیم؛ و در نهایت بازی آخر را برای منتظر ماندن بیشتر از 10 دقیقه مدل بکنیم، همه حالات را پوشش دادیم.

تعریف پاداش ها

جدا گانه برای تعریف پاداش ها برای استفاده از تاکسی یا اتوبوس را به صورت زیر تعریف میکنیم:

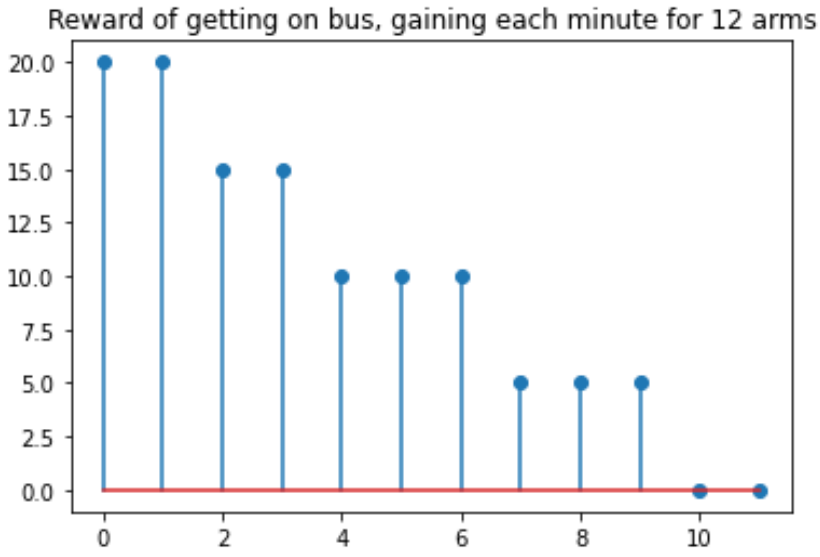
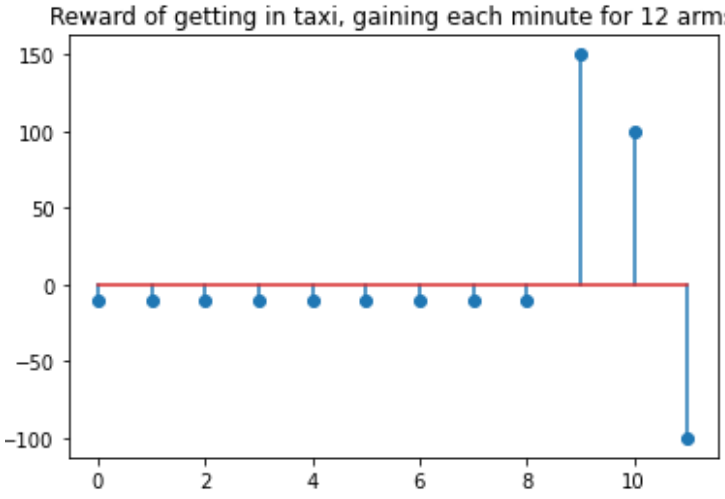


Figure 4

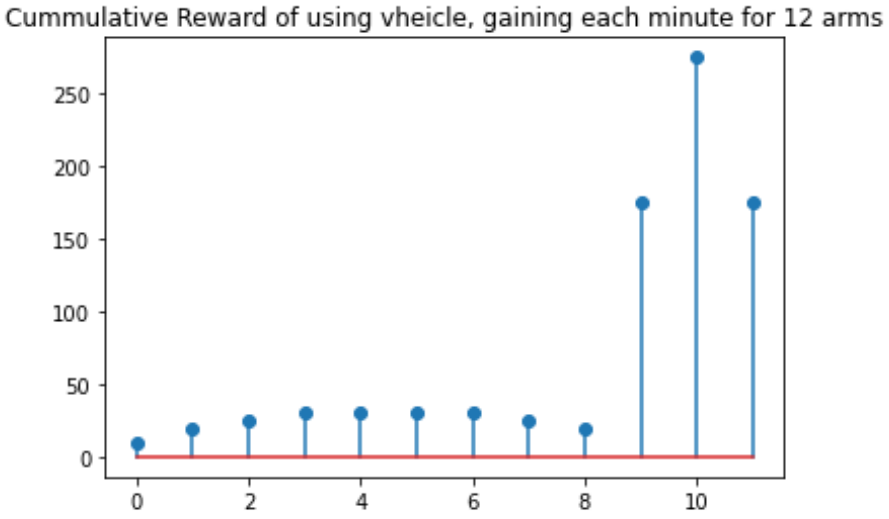


5Figure

با توجه به شکل 4 میزان پاداشی که در هر دقیقه میزان پاداش (تا سر همان دقیقه منتظر بون برای اتوبوس) را که تعریف میکنم، به صورت نزولی کاهش میابد تا وقتی که برای زمان 7:55 به صفر می رسد. البته باید توجه داشت که میزان پاداش کل ام مثلا برای 3 دقیقه منتظر ماندن برای اتوبوس، برابر مجموع تجمعی پاداش ها تا آن لحظه است. برای بیشتر از 10 دقیقه منظر ماندن پاداشی برایش تعریف نکرده ام و صفر است.

با توجه به شکل 5 میزان پاداشی که در هر دقیقه برای انتخاب تاکسی تعریف میکنم، تا لحظه7:54 دقیقه، منفی است؛ چرا که در هر زمانی از 6:55 تا 7:54 اگر مدل من تاکسی را انتخاب کرد بتواند بفهمد که انتخابش اشتباه بوده و اتوبوس برای آن انتخاب آن لحظه بهتر است. باید توجه کرد که برای تاخیر بیش از 10 دقیقه برای انتخاب تاکسی هم به صورت -100 تعریف کرده ام که اگر بیشتر از 10 دقیقه طول بکشد تاخیر منطقی است که از اتوبوس استفاده بشود تا میزان تنبیهی که شامل حال مدل من میشود کمتر شود(چون پاداش اتوبوس برای این شرایط صفر است). برای لحظه 7:54 و 7:55 پاداش منظور شده یک عدد مثبت بزرگی است که برای زمان 7:54 این عدد بیشتر است چرا که در نهایت با مقایسه انتخاب اتوبوس یا تاکسی برای زمان 7:54 یا 7:55 باید تاکسی انتخاب بشود تا به کلاس به موقع برسد.انتخاب پاداش بزرگتر برای زمان 7:54 به این دلیل است که میزان فاصله رسیدن از ایستگاه اتوبوس تا تاکسی هارا در نظر گرفته ام که تا حداکثر یک دقیقه فاصله داشته باشند، انتخاب یک دقیقه کمتر برای اتوبوس منتظر ماندن بهتر است.

برای پاداش استفاده است اتوبوس +200 و برای دیر رسیدن به کلاس -1000 منظور شده است. با توجه به تعاریف هنگام یادگیری مدل ما میزان پاداشی که در خروجی میبینیم بدون در نظر گرفتن استفاده از اتوبوس یا دیر رسیدن به کلاس به صورت زیر است :

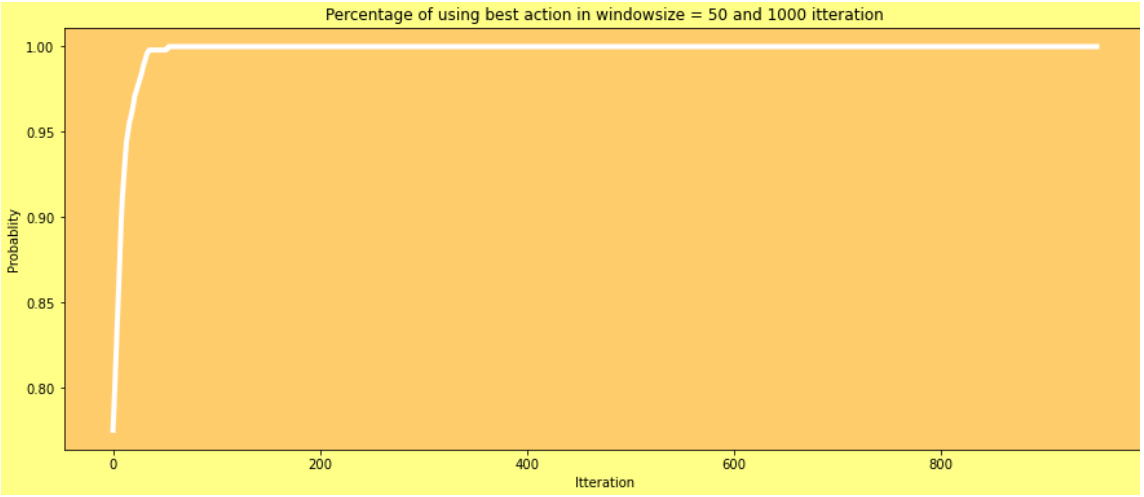


6Figure

با توجه به شکل 6 این ارجحیت با زمان 7:55 است.

استفاده از عامل $\epsilon - greedy$

خروجی های استفده از تصمیم بهینه و Regret را برای این مدل داریم :



7Figure

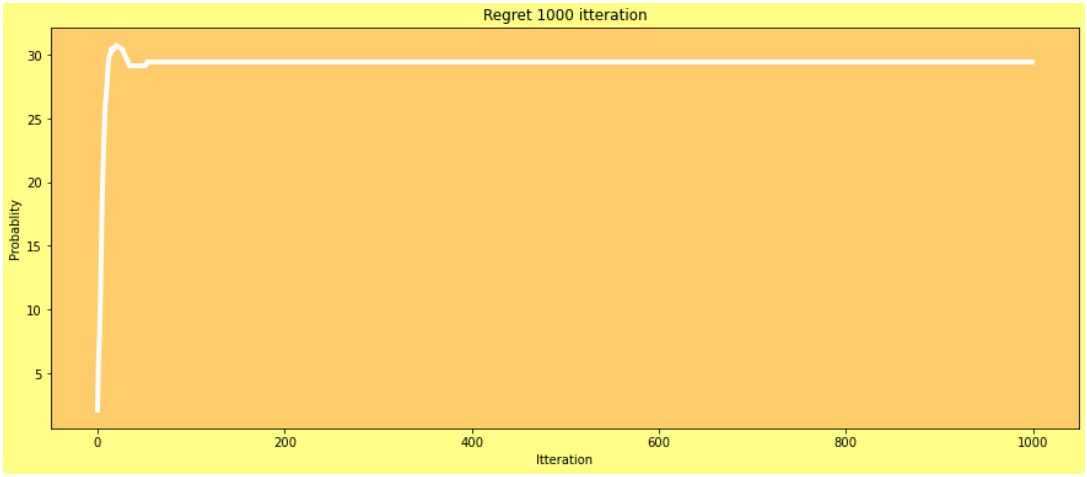
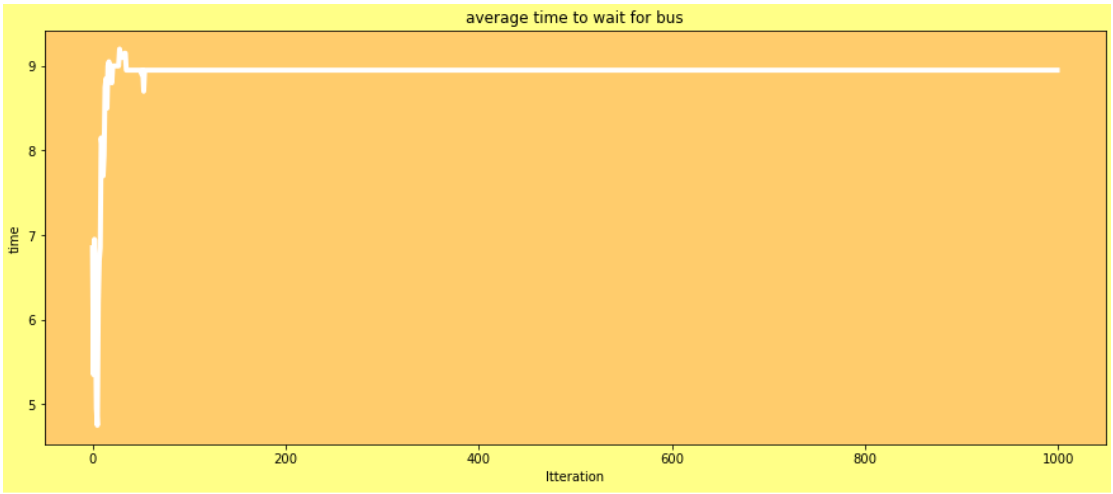


Figure 8

با توجه به شکل 7 و 8 میزان استفده از تصمیم بهینه و Regret را مشاهده میکنیم، که نمایانگر این است که در تعداد تلاش های خیلی کم مدل به حالت بهینه میرسد.

میزان منتظر بودن به صورت میانگین برای این مدل به صورت زیر است :



9Figure

با توجه به شکل 9 میزان متوسط منتظر ماندن برای اتوبوس برای این مدل برای 20 بار اجرا برابر 8دقیقه و 56 ثانیه است، که با استفاده از فرض های اولیه رسیدن این جواب قابل انتظار بود چرا که میزان استفاده از بازوی 10و9 ام بیشتر بوده است، که در عمل با توجه شکل 9 این عامل اول به صورت کم بازوی 10 را نتخاب میکند و بعد متناوباً به انتخاب بازوی 9 می انجامد که در نهایت میزان منتظر ماندن در حدود 9 دقیقه را حاصل میشود. پس چون کمتر از 10 دقیقه است مدل درست یادگرفته و به موقع به سر کلاس خواهد رسید. با توجه به یک اسپایک منفی در تلاش های اول این برداشت را میتوان کرد که در آن حالت

احتمالا اتوبوس به موقع رسیده بود و عامل چون میزان سوار شدن به اتوبوس با پاداش بیشتری برخوردار بود آن بازو را انتخاب کرده و بود که باعث شده بود میزان صبر کردن کاهی بیابد، ولی درنهیت همان حدود 9 دقیقه منتظر ماندن را انتخاب کرده است.

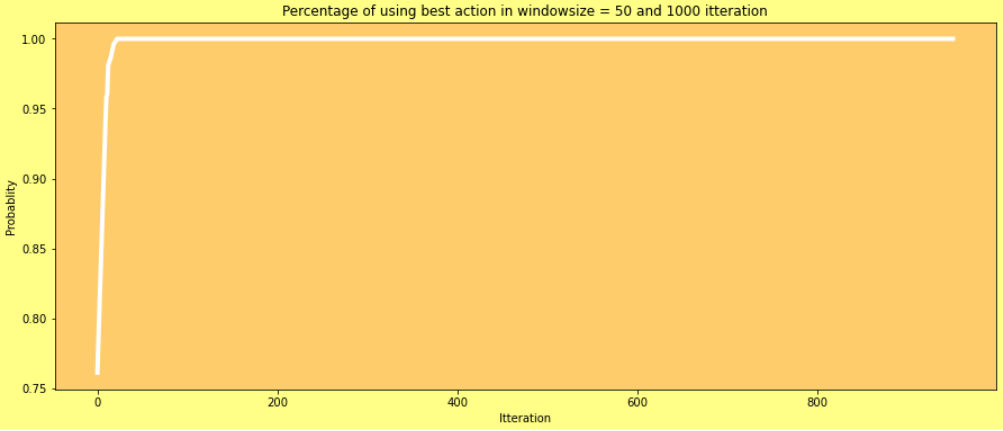
در این عامل مقدار اپسیلون در اول مقدار 1 گرفته است، چرا که ما پیش زمینه و اطلاعات قبلی از هیچ یک از تصمیم ها و پاداش آن نداشتیم و بعد در طی تلاش ها انی مقدار با ضریب 0.97 برای هر تلاش کم میشود که این کار باعث میشود مدل در نقاط ایستای محلی گیر نکند.

در هر مرحله با توجه به این که کدام یک از Q-value ها بیشتر است تصمیم بهینه را انتخاب میکردیم که در نهایت با انتخاب آن مقدار احتمالات انتخاب جدید برای همه تصمیم هارا بروط رسانی میکردیم به گونه ای که داشتیم:

$$\begin{cases} p = 1 - \epsilon + \frac{\epsilon}{||A||} & if\ action\ is\ optimal \\ p = \frac{\epsilon}{||A||} & else \end{cases}$$

استفاده از عامل UBC

خروجی های استفده از تصمیم بهینه و Regret را برای این مدل داریم :



10**Figure**

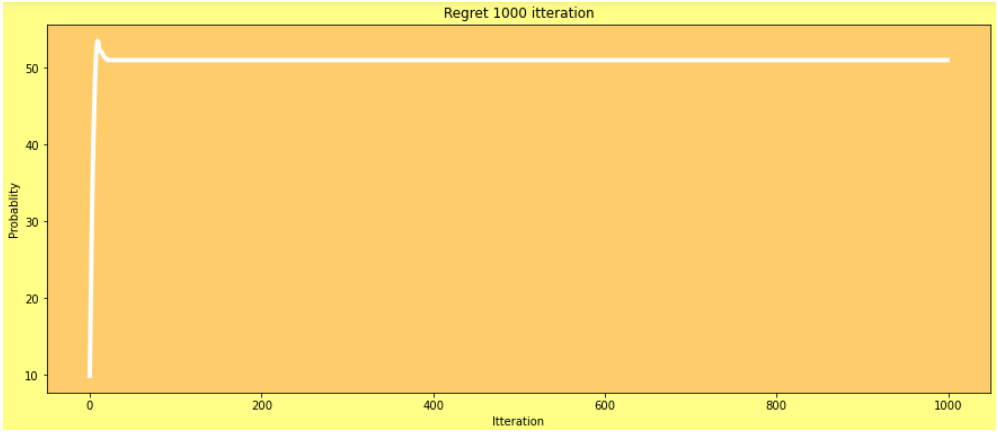
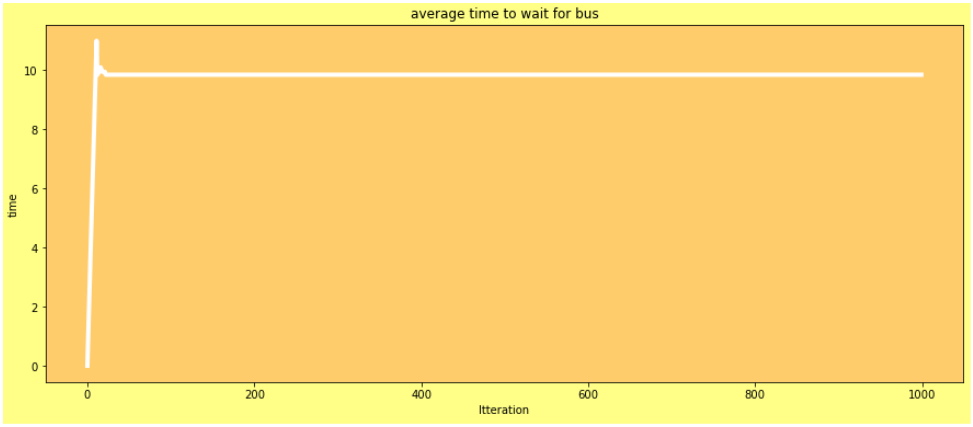


Figure 11

با توجه به شکل 10 و 11 میزان استفده از تصمیم بهینه و Regret را مشاهده میکنیم، که نمایانگر این است که در تعداد تلاش های خیلی کم مدل به حالت بهینه میرسد.



12**Figure**

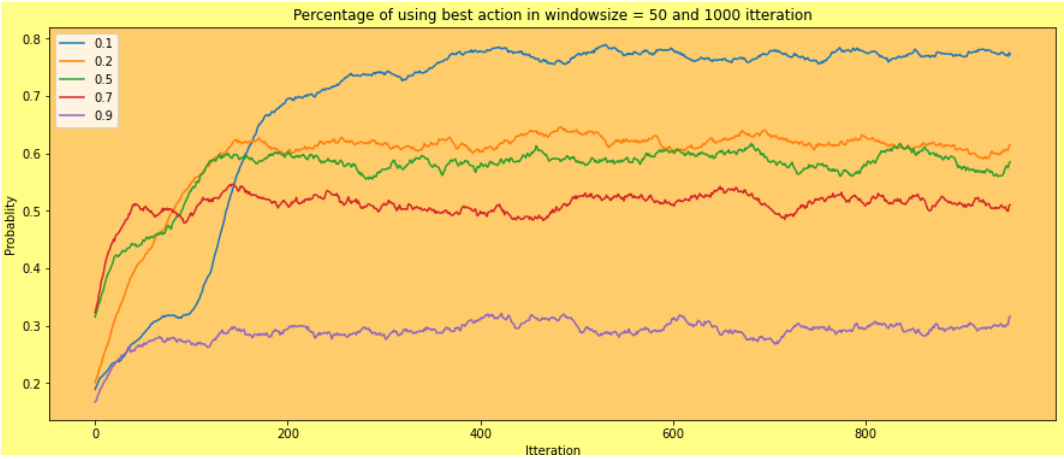
با توجه به شکل 12 میزان متوسط منتظر ماندن برای اتوبوس برای این مدل برای 20 بار اجرا برابر 9دقیقه و 50 ثانیه است، که با استفاده از فرض های اولیه رسیدم این جواب قابل انتظار بود چرا که میزان استفاده از بازوی 10و9 ام بیشتر بوده است، که در عمل با توجه شکل 12 این عامل اول به صورت کم بازوی 10 را نتخاب میکند که در نهایت میزان منتظر ماندن در حدود 10دقیقه را حاصل میشود. پس چون کمتر از 10 دقیقه است مدل درست یادگرفته و به موقع به سر کلاس خواهد رسید. البته این مدل در ابتدا عمل منتظر ماندن بیشتر از 10 دقیقه را هم انختاب میکند ولی به خاطر این که میزان پاداش تعریف شده خیلی کم است این بازو را دیگر انتخاب نمیکند، در حقیقت این عامل در ابتدا باید همه تصمیم هارا اتخاذ کند وبعد با توجه به Q_value ها و قسمتBonus این عامل، بیشترین احتمال تصمیم بهینه را اتخاذ میکند و چون در هر تلاش Q_value ها جمع تر میشوند، پس بعد از مدتی که در ابتدا تصمیم بهینه

را انتخاب کرد، معمولا از تصمیم های دیگه نزدیک بهینه هم انتخاب میکند و در نهایت به انتخاب کامل تصمیم بهینه منجر میشود.

برای انتخاب بهترین تصمیم در هر تلاش باید ابتدا Q-value هارا آپدیت کنیم و بعد در انتها با حساب bonus و اضافه کردن آن به این قسمت و انتخاب بیشترین آن ها بهترین تصمیم را اتخاذ میکنیم.

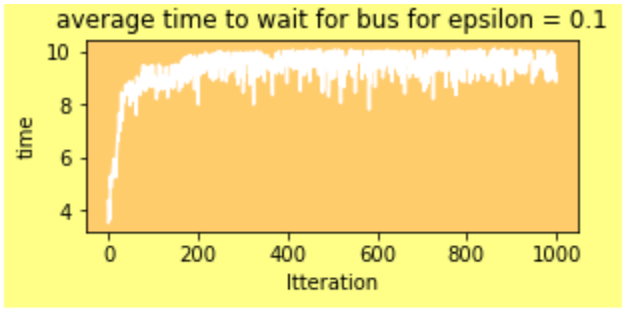
Cross validation

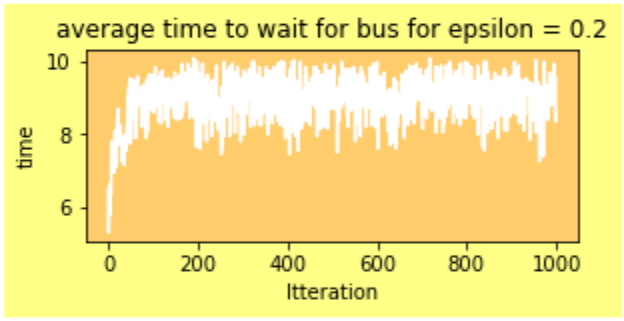
معنی cross validation این است که یک هایپر پارامتری را با امتحان کردن مقادیر مختلف آن به گونه ای انتخاب کنیم که از یک نظر خاص در آن مقدار از هایپر پارامتر خاص مدل ما بهینه تر از دیگر مقادیر کار کند ، در این جا مقادیر مختلف این هایپر پارامتر را میدهیم تا دید کلی راجع به مدل داشته باشیم. توجه کنید که در این حالت مقدار هایپر پارامتر برای همه تلاش ها ثابت خواهد بود و کاهشی نخواهد بود. با توجه به خروجی ها برای epsilon های مختلف داریم :

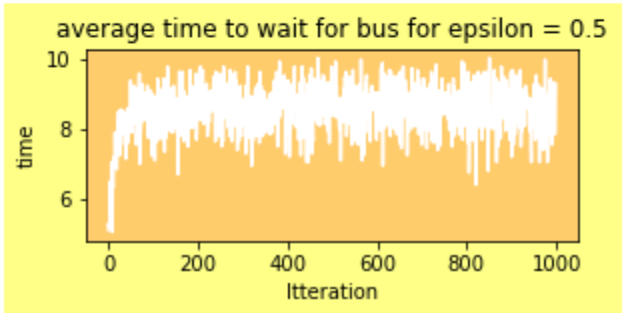


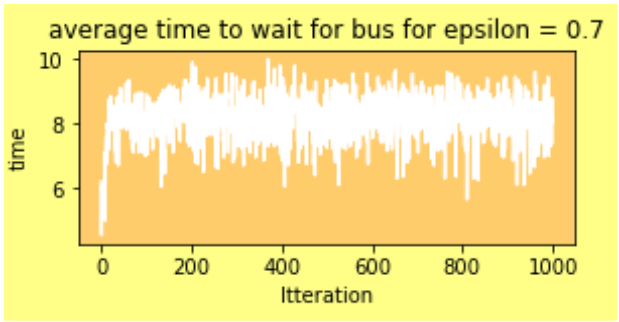
13**Figure**

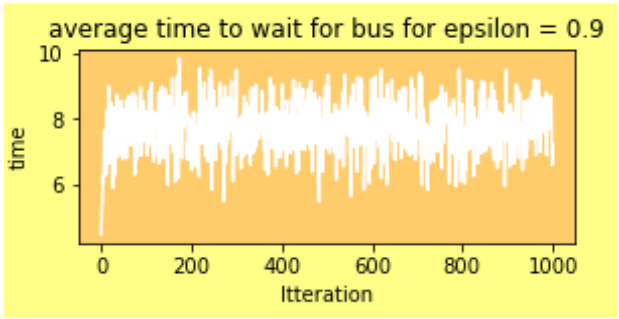
با توجه به شکل 13 میتوان فهمید که استفاده از $\epsilon = 0.1$ بهینه تر است از استفاده $\epsilon =$ 0.2,0.5,0.7,0.9 چرا که با کاهش اپسیلون مدل ما بیشتر Greedy تر عمل خواهد کرد. البته با تحلیل دقیقتر شکل 13 میتوان فهمید که تا تلاش حدود 150 به دلیل اهمیت موضوع exploration اپسلون برابر 0.7 بهتر نتیجه میدهد که ابته این مهم نیست و در نهایت در 1000 تلاش اپسیلون برابر 0.1 با اختلاف 20 درصد بهتر عمل کرده است.


average time to wait for bus for epsilon = 0.1
time
Itteration


average time to wait for bus for epsilon = 0.2
time
Itteration


average time to wait for bus for epsilon = 0.5
time
Itteration


average time to wait for bus for epsilon = 0.7
time
Itteration

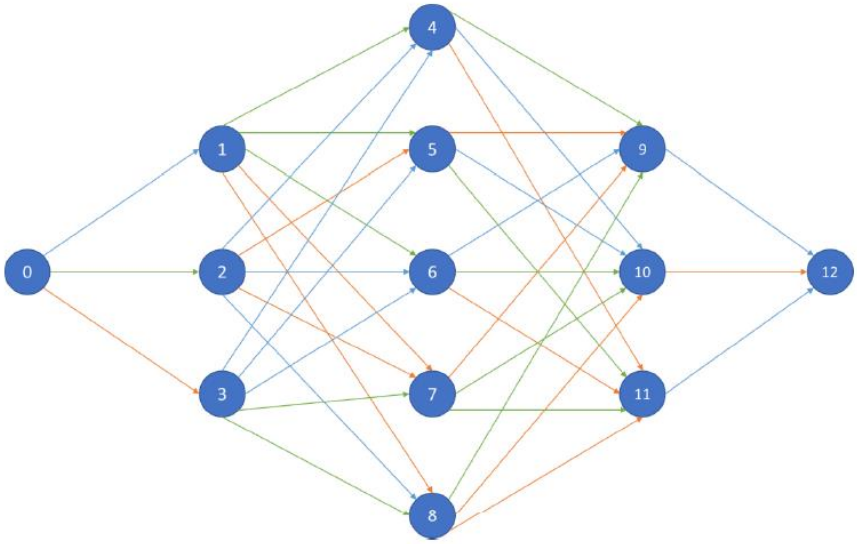

average time to wait for bus for epsilon = 0.9
time
Itteration

14**Figure**

میزان بهینه صبر کرد برای epsilon های مختلف میتوان فهمید که طبق شکل 14 با افزایش مقدار epsilon که به فاکتور Exploration می افزاییم، میزان مدت زمان بهینه صبر کردن ما کاهش میابد در epsilon = 0.9 به حدود 7 دقیقه میرسد که این خوب نیست چرا که هم از نظر منطقی باید حدود 10 دقیقه منتظر بماند که این عمل انتظار حدود7 دقیقه یعنی این که تاکسی سوار شدن را ترجیح میدهد چرا که در نهایت به کلاس زود میرسد و پاداش خوبی دریافت میکنید. چرا این اتفاق می افتد؟ چون که میزان هایپر پارامتر epsilon در این قسمت آنقدر بالا بود که باعث میشد به مقدار خوبی مدل با exploit نکند و این نقطه نقص قضیه است. برای حل مشکل باید مقدار این هایپرپارامتر را کاهش دهیم که همانند شکل 14 میتوان دید که برای epsilon های کم میزان منتظر بودن به مقدار بهینه یعنی حدود 10 دقیقه میرسد. البته باید این نکته را هم در نظر گرفت که چون از مقدار هایپر پارامتر کم میشود و این باعث میشود که تصمیم بهینه را بیشتر انتخاب کنیم نوسان خروجی برای 1000 تلاش طبق شکل 14 کمتر خواهد بود و هرچقد هایپر پارامتر را افزایش دهیم این مقدار افزایش خواهد یافت.

سوال سه

مدل سازی



15**Figure**

با توجه به این که تعداد کل مسیر های رسیدن از گره 0 به گره 12 برابر 45 تا است. من هم یک مدل با 45 بازو در نظر گرفته ام که برای تعریف پاداش های مربوط به ان نیز اینگونه عمل کرده ام که ، با توجه به این که در کل 3 توع link داریم پس من کل link هایی که آبی یا سبز باشند را پیدا کردم و در یک بردار ذخیره کردم، در نهایت با اعمال حلقه کل مسیر را با توجه به جایگشت های مختلف گره ها یافتم. در عمل میزان تاخیر این که بسته در نهایت به خروجی برسد در هر link برابر میزان سمپل هایی است که بر توزیع ها به ازای آن link خاص چه توسط گره ها و چه توسط شاخه ها گرفته ایم.

با توجه به این که هر عامل تلاش دارد که میزان پاداش گرفته شده را بیشینه کند، در اینجا هدف برعکس است و باید میزان تاخیر را حد اقل کنیم و هر عاملی یاد بگیرد که در نهایت با چند بار اجرا مسیر هایی که کمترین تاخیر را دارند را انتخاب کند. برای انجام این کار میتوان به صورت راه های مختلف پاداس های قسمت قبل را طوری تغییر داد که با همان عملرگد هامل ها بتوان به آن مسیر با حداقل تاخیر رسید:

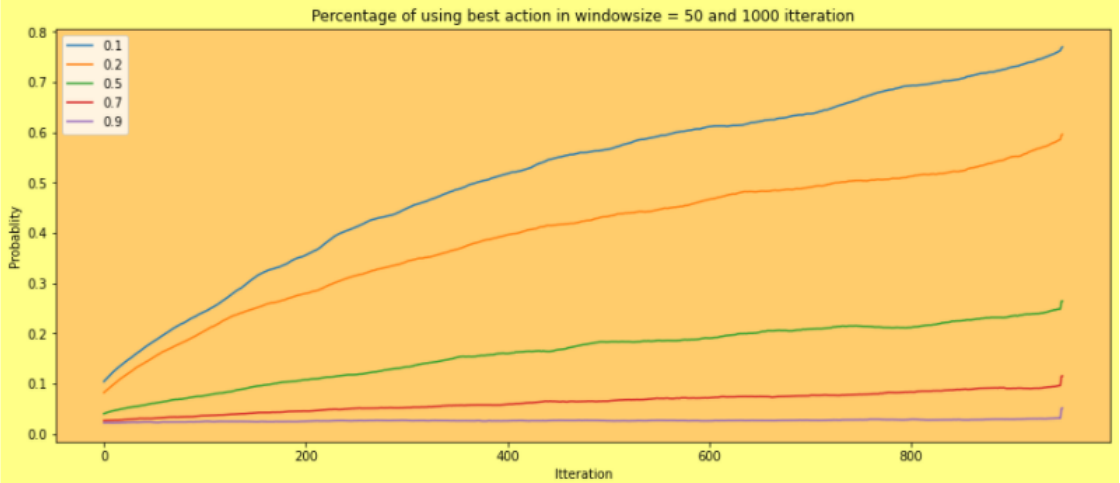
- اضافه کردن یک bias یه قرینه پاداش ها به گونه ای که $r = bias - r$ که این bias میتوان به گونه انتخاب بشود که همه پاداش ها مثبت بشوند .
- میتوان بجای این که bias بدهیم تنها مدل را قرینه کنیم به گونه ای که $r = -r$
- میتوان معکوس هر پاداش را به عامل داد مثلا $r = \frac{1}{r}$

برای حالت سوم میتوان به سادگی گفت این کار نتیجه درستی نخواهد داد یا اگر نتیجه درست بدهد باید تعداد خیلی زیادی اجرا داشته باشیم تا به پاداش بهینه برسیم

برای حالات های اول و دوم امتحان میکنیم. البته برای هر کدام از پاداش های حالت اول یا دوم میتوان مقدار پاداش های دریافتی را در یک ثاتی ضرب کرد که اسکیل بشوند و با این کار فاصله پاداش ها از هم زیاد خواهد بود به گونه ای که بشود راحت تفکیک کرد و در این حالت شاید عامل با تعداد اجرای کمتر به نتیجه برسد. حالت دوم را با استفاده از عامل e-greedyو حالت سوم رو با استفاده از عامل gradient آموزش دادم.

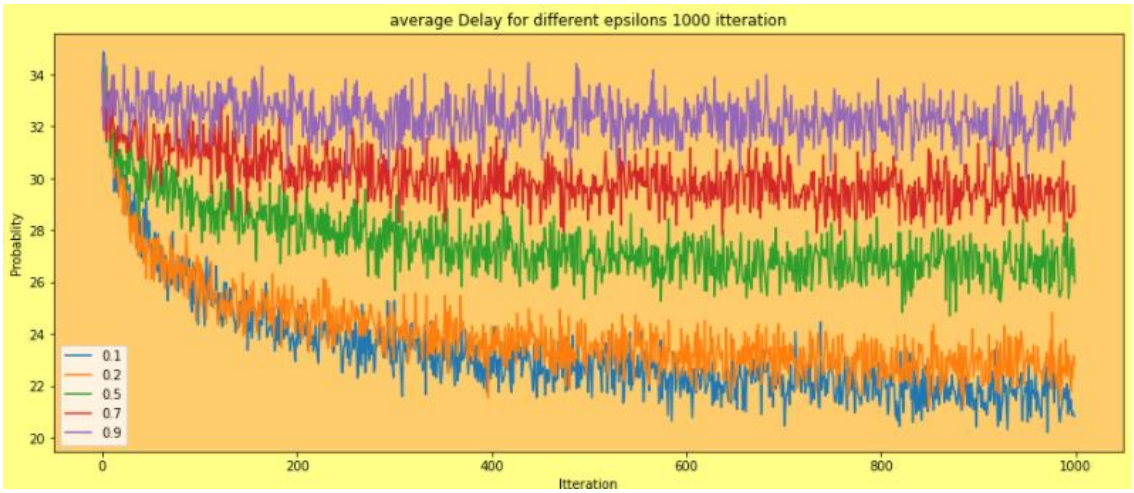
توسعه عامل e-greedy

با توجه به توصیف عملکرد این عامل در بخش سوال دوم، فقط به اعلام نتایج انتهایی بسنده میکنیم. با توجه به این که برای این عامل پاداش هارا با یک bias داده ام نتیجه در کل بهتر از عامل gradient شده است چرا که با 1000 اجرا تقریبا نتیجه عامل gradient با 2000 اجرا را میگیرد که نشان میدهد عمل bias کردن به مراتب بهتر است البته باید توجه داشت که چون عامل ها متفاوت است نمیتوان به صورت قاطع این حرف را زد ولی در ظاهر که این عمل دیده میشود. با توجه به مقایسه روش برای هایپر پارامتر epsilon های مختلف داریم:



16**Figure**

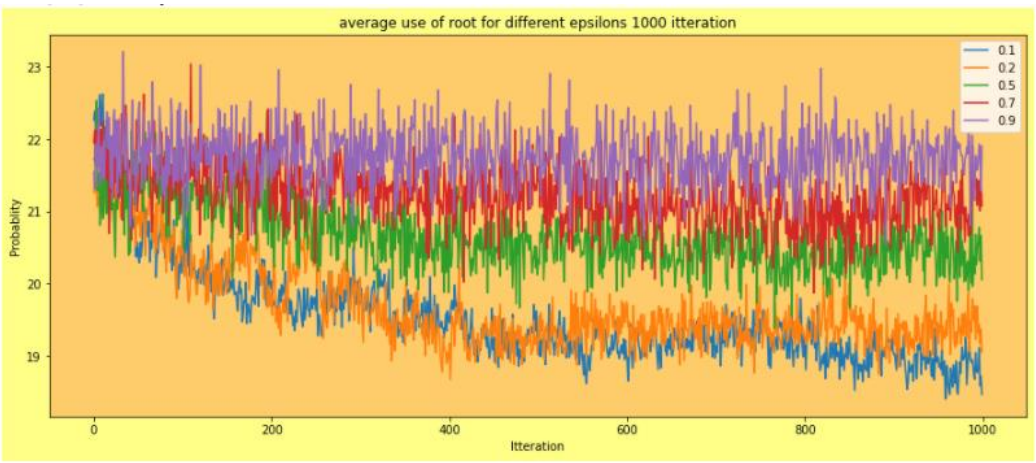
با توجه به شکل 16 برای هایپر پارامتر با مقادیر مختلف نشان میدهد که در افق 1000 تلاش مدل برای هایپر پارامتر با مقادیر کمتر بیشتر از تصمیم بهینه استفاده میکند که در بهترین حالت این مقدار 0.1 است که به درصد 80 در افق 1000 تلاش میرسد.



17**Figure**

با توجه به شکل 17 که نشان دهنده آن است که در نهایت مقدار تاخیر گرفته شده در افق 1000 تلاش برای کدام مقدار هایپر پارامتر بهتر است که میتوان دید که برای مقدار 0.1 از همه بهتر است و به تاخیر میانگین حدود 22 ثانیه میرسیم.

در نهایت برای میانگین تصمیم بهینه با توجه به شکل 18 داریم:



18**Figure**

با توجه به شکل 18 داریم که بازم اکشن بهینه برای هایپر پارامتر اپسیلون برابر 0.1 حول هوش اکشن 19 است که همسایگی های اکشن 19 عبارت اند از:

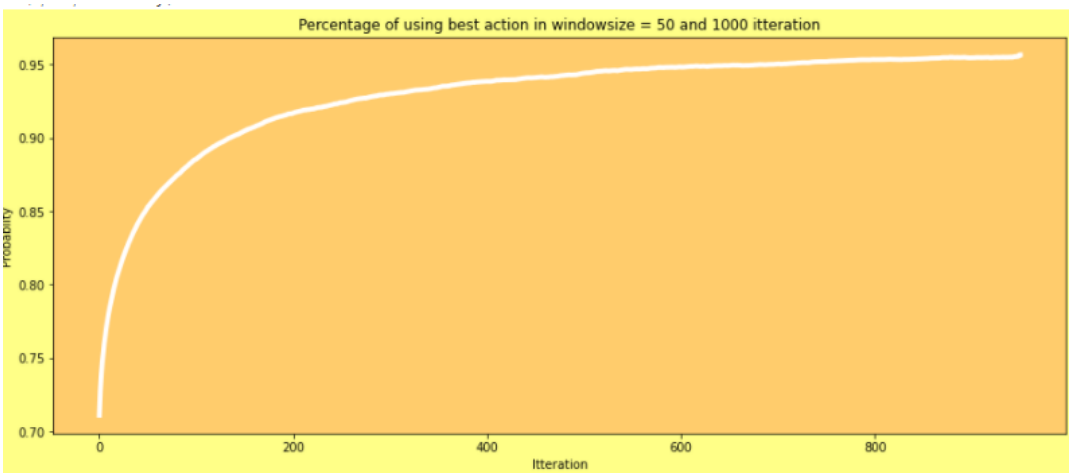
```
[22] links[19-3:19+3]

['2 4 10', '2 4 11', '2 5 9', '2 5 10', '2 5 11', '2 6 9']
```

19**Figure**

شکل 19 انتخاب گره های وسط را برای همسایگی های تصمیم 19 نشان میدهد.

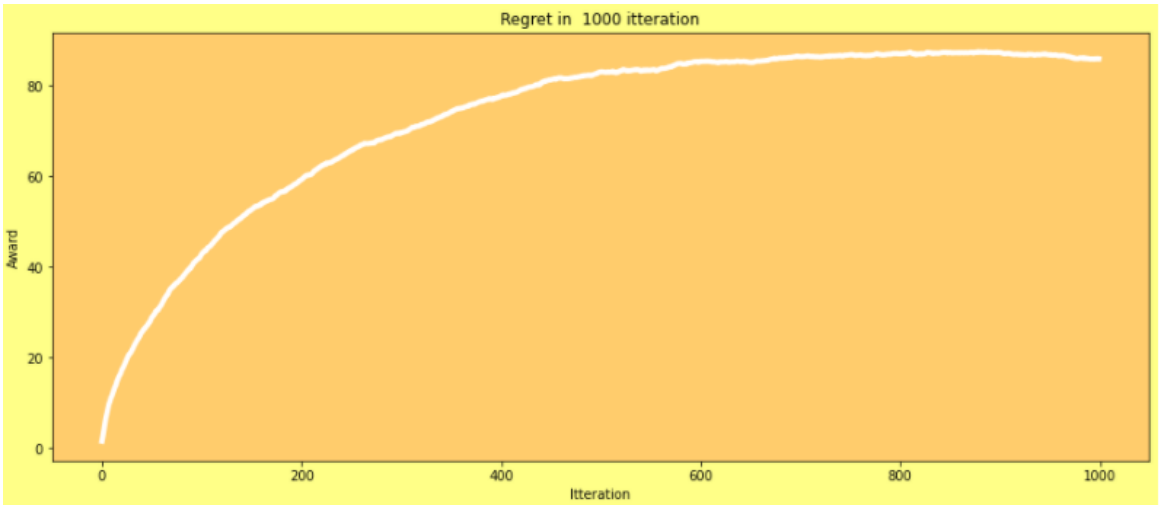
عامل gradient

خروجی های این عامل به صورت زیر است :



20**Figure**

این عامل در افق تلاش 100 و برای 2000 بار اجرا در نهایت خروجی ها را داده است. طبق شکل 20 این عامل در نهایت به درصد استفاده از عمل بهینه 95 درصد میرسد که 15 درصد بیشتر از در حالت بهترین هایپر پارامتر برای عامل قبلی است که این نکته مثبت این عامل را نشان میدهد.



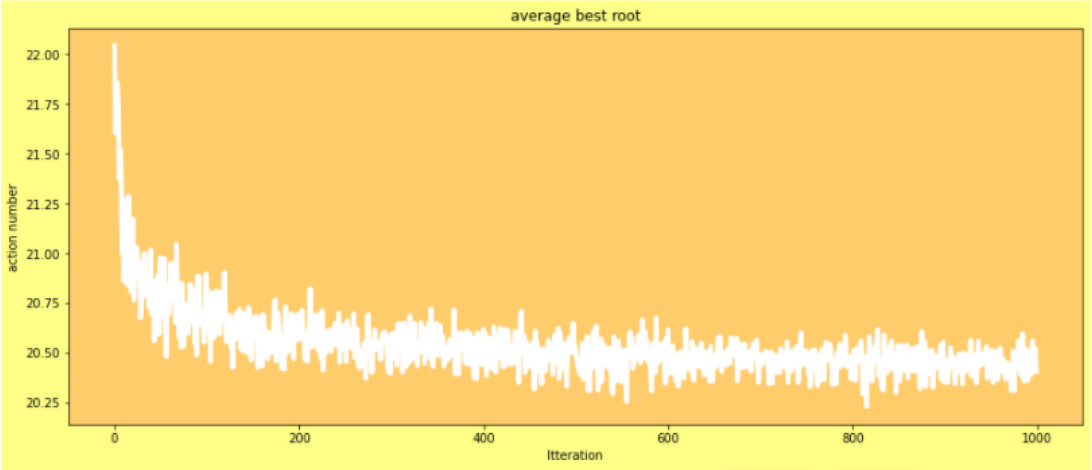
21**Figure**

با توجه به شکل 21 میزان Regret برای این عامل به صورت تقریبی به حالت $0.8\log(itteration)$ رسیده که خیلی خوب است و توانسته ایم عاملی را درست کنیم که رفتار لپاریتمی داشته باشد و حتی با سرعت افزایشی کمتر از آن.

22**Figure**

با توجه به شکل 22 این عامل در نهایت سیستم را به حالت تاخیر حدودی 26 ثانیه میرساند که با توجه به عامل قبلی که با بهترین حالت هایپر پارامتر به تاخیر حدود 22 ثانیه میرسید آن هم در 1000

افق تلاش بد تر است و این نشان میدهد که عامل قبلی با استفاده از bias داد به قرینه پاداش ها بهتر از این عامل با قرینه پاداش ها عمل کند که نشان میدهد تقریبا bias میتواند تاخیر را کمتر بکند، این بیشتر شدن تاخیر را از روی استفاده میانگینی از یک بازوی خاص هم میتوان در شکل 23 دید:



23**Figure**

با توجه به شکل 23 این عامل در نهایت به صورت میانگین بازو هایی را انتخاب میکند که در همسایگی بازوی شماره 29 و در حول هوش همسایگی های زیر است :

```
In [3]:  1  links[20-3:20+3]
Out[3]: ['2 4 11', '2 5 9', '2 5 10', '2 5 11', '2 6 9', '2 6 10']
```

Figure 24

شکل 24 نشان میدهد که برای همسایگی بازو 20 ام که توسط گره های وسطی با توجه به شکل 24 انتخاب شده اند عامل همگرا میشود.

مقایسه عامل ها برای مدل

در حالت کلی با توجه به این که این سیستم کاملا متغیر با زمان است پس اصولا برای این که زود تر بتوانیم با اجرای کمتر تصمیم بهینه را پیدا کنیم ترتیب استفاده از عامل ها و اولیت هایشان به صورت زیر خواهد بود:

$$full\ greedy < \epsilon - greedy < gradient < UBC < Thompson\ Sampling$$

انتخاب های بالا نشان میدهد که برای این سوال استفاه از bias خیلی بهتر از استفاده خالی از قرینه کردن بود چرا که مدل e-greedy دارای سرعت کمتری است نسب ت به عامل gradient و در این سوال در افق تلاش نصف عامل gradient جواب بهتر گرفته است که این خیلی خوب است. ترتیب نمایش داده بالا با توجه به این نکات انتخاب شده است که برای مد هایی که متغیر با زمان هستند باید عاملی انتخاب بشود که در ابتدا explorationه ای خودرا بین بازو هایش به تعدد و کامل انجام بدهد و در نهایت exploit بکند که برای UBC , Thompson Sampling عامل Thompson بهتر است چرا که در اول به مقدار خوب و بهتری نسبت به عامل UBC بین بازو ها میگردد تا پاداش های متفاوت از بازو هارا بگیرد و مدل سازی کند البته اینجا استفاده از مدل Thompson با استفاده از توزیع t-student میتوانست حتی تاثیر بسزایی در سرعت همگرایی روش داشته باشد چرا که توزیع هایمان هم در ابتدا گوسی نیست. در حال کلی UBC و Thonpson تقریبا در یک سطح هستند ولی دومی بهتر است.