دانشکده ریاضی و آمار

مدرس: **دکتر منصوری** نیمسال اول ۱۴۰۲-۱۴۰۱

مسائل MDP

هوش مصنوعی

نگارنده: عباسعلی رضائی

۱۴ دی ۱۴۰۱

فهرست مطالب

۱ Policy Iteration

Value iteration از نظر زمان اجرا بسیار کند است زیرا در هر تکرار، همانطور که ما Q-value را برای هر عمل محاسبه می کنیم، باید مقادیر تمام حالت‌هایی را که نیاز به تکرار در تمام اعمال را دارد به‌روز کنیم. محاسبه هر یک از این Q-value به نوبه خود نیاز به تکرار دوباره در هر یک از حالت‌ها دارد که این کار منجر به زمان اجرای ضعیف $O(|S|^2|A|)$ دارد.
همانطور که گفتیم در مسئله MDP هدف ما پیدا کردن سیاست بهینه است. Value iteration نیاز به محاسبات بیش از حد دارد، زیرا سیاستی که ما از طریق policy extraction محاسبه می کنیم بسیار سریعتر از خود مقادیر همگرا می شود.
برای رفع این نقص از الگوریتم Policy Iteration به عنوان یک جایگزین استفاده می کنیم، الگوریتمی که بهینه بودن Value iteration را حفظ می کند و در عین حال دستاوردهای قابل توجهی را ارائه می دهد.
Policy Iteration به شرح زیر عمل می کند:

۱. ابتدا باید یک سیاست اولیه را تعریف کنیم، برای انتخاب سیاست اولیه می توانیم از هر سیاستی استفاده کنیم، اما هرچه سیاست اولیه به سیاست بهینه نزدیک تر باشد، Policy Iteration سریعتر همگرا می شود.

۲. موارد زیر را تا زمان همگرایی تکرار کنید:

- با استفاده از Policy evaluation سیاست جاری را ارزیابی می کنیم. Policy evaluation برای یک سیاست π به معنای محاسبه $U^{\pi}(s)$ برای همه حالت های s است، به طوری که $U^{\pi}(s)$ سودمندی مورد انتظار از حالت شروع تا حالت s با دنبال کردن سیاست π است:

$$U^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma U^{\pi}(s')]$$

یک سیاست از Policy Iteration در تکرار i ام را به صورت π_i تعریف می کنیم. از آنجایی که ما به دنبال یک عمل واحد برای هر حالت هستیم، دیگر به عملگر max نیازی نداریم. برای محاسبه $U\pi_i$ ها از update rule زیر تا زمانی که به همگرایی برسد استفاده می کنیم، درست مانند کاری که در value iteration می کردیم :

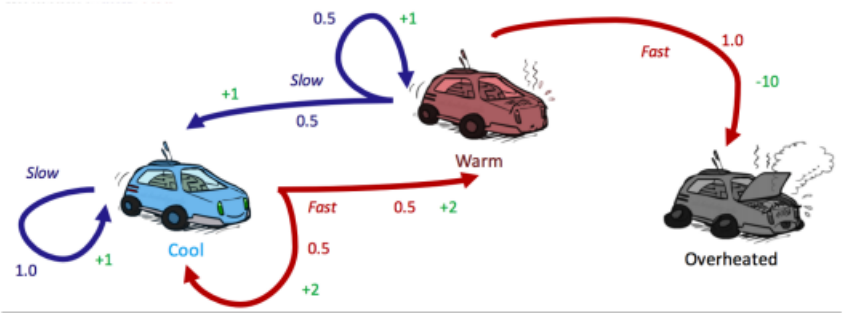
$$U^{\pi_i}_{k+1}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma U^{\pi_i}(s')]$$

- هنگامی که سیاست جاری را ارزیابی کردیم،از policy improvement برای ایجاد سیاست بهتر استفاده می کنیم. policy improvement از policy extraction روی ارزش حالت های که توسط Policy evaluation محاسبه شده است، به منظور ایجاد سیاست و بهبود آن استفاده می کند:

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma U^{\pi_i}(s')]$$

- اگر $\pi_{i+1} = \pi_i$ آنگاه الگوریتم همگرا شده است و می‌توان نتیجه گرفت $\pi_{i+1} = \pi_i = \pi^*$.

مثال ماشین مسابقه ای را این بار با استفاده از الگوریتم Policy Iteration حل می کنیم. در این مثال از $\gamma = 0.5$ استفاده می کنیم.



سیاست اولیه را همیشه slow انتخاب می کنیم.

	cool	warm	overheated
π_0	*slow*	*slow*	–

هیچ سیاستی نمی تواند مقداری رابه یک حالت پایانی اختصاص دهد، زیرا در حالت های پایانی هیچ عملی قابل انجام نیست. از این رو منطقی است که حالت overheated را نادیده بگیریم یعنی به صورت کلی برای هر حالت پایانی داریم که $\forall i,\ U^{\pi_i}(s) = 0$.
حال یک دور از policy evaluation روی π_0 اجرا می کنیم:

$$U^{\pi_0}(cool) = 1 \times [1 + 0.5 \times U^{\pi_0}(cool)]$$
$$U^{\pi_0}(warm) = 0.5 \times [1 + 0.5 \times U^{\pi_0}(cool)]$$

با حل معادلات بالا داریم که :

	cool	warm	overheated
U^{π_0}	2	2	0

policy extraction را با مقادیر به دست آمده اجرا می کنیم:

$$\begin{aligned}\pi_1(cool) &= \arg\max\{slow : 1 \times [1 + 0.5 \times 2], fast : 0.5 \times [2 + 0.5 \times 2] + 0.5 \times [2 + 0.5 \times 2]\} \\ &= \arg\max\{slow : 2, fast : 3\} \\ &= fast\end{aligned}$$

$$\begin{aligned}\pi_1(warm) &= \arg\max\{slow : 0.5 \times [1 + 0.5 \times 2] + 0.5 \times [1 + 0.5 \times 2], fast : 1 \times [-10 + 0.5 \times 0]\} \\ &= \arg\max\{slow : 3, fast : -10\} \\ &= slow\end{aligned}$$

با اجرای policy iteration برای دور دوم داریم که $\pi_2(cool) = fast$ و $\pi_2(warm) = slow$. از آنجا که نتیجه بدست آمده دقیقا مشابه π_1 است، می توانیم نتیجه بگیریم که $\pi_1 = \pi_2 = \pi^*$.

	cool	warm
π_0	*slow*	*slow*
π_1	*fast*	*slow*
π_2	*fast*	*slow*

این مثال قدرت الگوریتم policy iteration را نشان می دهد. زیرا تنها با دوبار تکرار، به سیاست بهینه برای MDP ماشین مسابقه ای خود رسیده ایم.

۱.۱ مقایسه

الگوریتم های value iteration و policy iteration هر دو یک کمیت را محاسبه می کنند: مقادیر بهینه!

در الگوریتم value iteration:

- در هر تکرار، مقادیر و سیاست هر دو به روز رسانی می شوند.
- ما به دنبال سیاست نیستیم اما با محاسبه ی ماکزیمم بر روی عملیات، به طور ضمنی آن را محاسبه می کنیم

در الگوریتم policy iteration:

- چند گذر وجود دارد که در آنها مقادیر سودمندی را تنها برای یک سیاست ثابت به روز رسانی می کنیم
- پس از ارزیابی سیاست، یک سیاست جدید انتخاب می شود. [استخراج سیاست]
- سیاست جدید نسبت به سیاست قبلی بهتر است. [در غیر این صورت الگوریتم همگرا شده است]

۲ جمع بندی

الگوریتم های ارائه شده در این فصل یکسان به نظر می رسند. الگوریتم ها value iteration، policy iteration، policy extractionو policy evaluation را پوشش دادیم. همه‌ی این الگوریتم ها گونه های مختلف از به روز رسانی معادلات بلمن هستند.

- Value iteration:برای محاسبه مقادیر بهینه حالت ها، توسط به روزرسانی های تکراری تا زمان هم گرایی استفاده می شود.
- Policy evaluation: برای محاسبه مقادیر حالت ها تحت یک سیاست خاص استفاده می شود.
- Policy extraction: برای تعیین یک سیاست با توجه به تابع مقدار حالت استفاده می شود. اگر مقادیر حالت بهینه باشند، این سیاست بهینه خواهد بود. این روش پس از اجرای value iteration ، برای محاسبه یک سیاست بهینه از مقادیر حالت بهینه یا به عنوان یک زیر روال در policy iteration ، برای محاسبه بهترین سیاست برای مقادیر حالت تخمین زده شده فعلی استفاده می شود.
- Policy iteration: تکنیکی که هم policy evaluation و هم policy extraction را در بر می گیرد و برای هم گرایی یک سیاست بهینه استفاده می شود. به دلیل این که سیاست ها معمولا بسیار سریع تر از ارزش های حالت ها همگرا می شوند، تمایل دارد. از value iteration بهتر عمل می کند.