به نام خدا

عنوان:

تکلیف دوازدهم درس یادگیری ماشین

استاد:

دکتر پدرام

دانشجو:

محمدعلی مجتهدسلیمانی

۴۰۳۳۹۰۴۵۰۴

تاریخ:

۱۴۰۳/۱۰/۱۴

سوال اول

نقش برنامه نویسی پویا در یادگیری مانند پیدا کردن کوتاه ترین مسیر از خانه تا دانشگاه است. ما میتوانیم راه های مختلفی را برای رسیدن به دانشگاه امتحان کنیم ولی ناکارآمد هستند. در عوض میتوانیم از یک نقشه (environment) استفاده بکنیم تا یک مسیری را برنامه ریزی بکنیم. ما مسئله را به قسمت های کوچک تری تقسیم میکنیم و سعی میکنیم در هر کدام از این قسمت های کوچک تر مسیر دنبال کوتاه ترین مسیر بگردیم و مجموع اینها باعث پیدا شدن کوچکترین مسیر بشوند. این ایده همان ایده برنامه نویسی پویا خواهد بود. حل کردن یک مسئله پیچیده با شکستن به قسمت های کوچکتر که با همدیگر همپوشانی دارند و حل کردن هر کدام از این قسمت های کوچکتر باعث ساختن راه حل مسئله اصلی ما میشود.

برنامه نویسی پویا و یادگیری تقویتی با همدیگر مرتبط هستند که میتوانیم در رویکرد های مختلف رد این ارتباط را ببینیم:

- رویکرد model-based RL:

برنامه نویسی پویا به طور اصلی در این شیوه به کار میرود، جایی که agent به یک مدلی از محیط (environment) دسترسی دارد. این مدل شامل ۲ ویژگی است:

۱. Transition probability یا احتمال حرکت از یک state به state دیگر وقتی یک action را انجام میدهیم. ۲. Reward function یا پاداشی که برای هر action در یک state دریافت میشود.

- رویکرد محاسبه تابع value:

نقش اصلی برنامه نویسی پویا در یادگیری تقویتی محاسبه کردن مقدار بهینه تابع value است. این تابع ارزش (value) دراز مدت یک action را در یک حالت (state) خاص تخمین میزند. مقدار این تابع بیشینه پاداش جمع شده ممکن را که یک agent میتواند بدست بیاورد از هر حالت تخمین میزند. البته با فرض اینکه مدل دارد از سیاست (policy) بهینه تبعیت میکند. الگوریتم های پویایی مثل value iteration و policy iteration در این مرحله به کار میروند.

- رویکرد value iteration:

در این رویکرد ما به صورت تکرار شونده مقدار تخمینی value function را بهبود میدهیم تا به مقدار بهینه همگرا شود. در این رویکرد از معادله Bellman استفاده میشود.

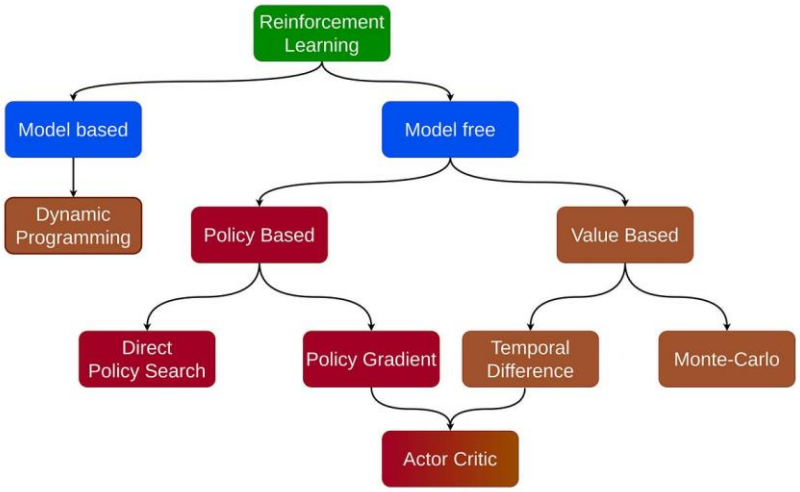
- رویکرد policy iteration:

در این رویکرد ما ۲ مرحله داریم: ۱. Policy evaluation که در این مرحله ما مقدار تابع value را برای سیاست مد نظر محاسبه میکنیم. ۲. Policy improvement که در این مرحله بر اساس مقدار فعلی تابع value یک سیاست بهتری را پیدا میکنیم که از سیاست فعلی بهتر باشد، این پیدا کردن سیاست بهتر توسط انتخاب یک action در هر حالت که ما را به بشینه پاداش مورد انتظار ما بر طبق تابع value فعلی برساند صورت میگیرد.

- پیدا کردن سیاست بهینه:

وقتی مقدار بهینه تابع value محاسبه شد توسط هر کدام از رویکرد های بالا، ما سیاست بهینه را انتخاب میکنیم و agent به سادگی در هر حالت actionیی را انجام میدهد که پاداش مورد انتظار خودش را زیاد کند.

اگر چه که برنامه نویسی یک رویکرد model-based است، در رویکرد model-free هم از ایده های برنامه نویسی پویا استفاده میشود.



سوال ۲

در یادگیری تقویتی ما با ۲ محیط قطعی و غیر قطعی روبرو هستیم. محیط در یادگیری تقویتی بسیار اهمیت دارد زیرا تعیین میکند agent چگونه آموزش میبیند. یک راهی برای دسته بندی محیط ها بر اساس قابل پیشبینی آنها است که باعث میشود ما با ۲ محیط قطعی و غیر قطعی روبرو شویم:

- محیط قطعی:

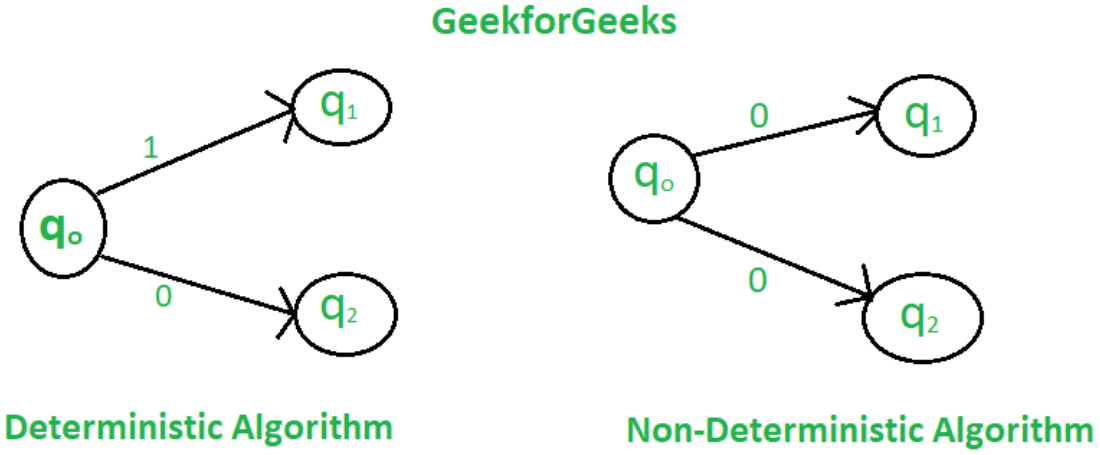
در یک محیط قطعی، یک action یکسان در یک حالت یکسان همیشه باعث رفتن به یک حالت یکسان دیگر و پاداش یکسان میشود. به عبارت دیگر هیچ تصادفی یا عدم قطعیتی در انتقال در این محیط ها وجود ندارد. این امر باعث میشود که نتایج actionها کاملا قابل پیشبینی باشد. یعنی اگر ما حالت فعلی و action را بدانیم با قطعیت کامل میتوانیم حالت بعدی و مقدار پاداش را بگوییم. بازی شطرنج مثالی از این محیط است که محیط قطعی به شمار میرود زیرا با action و حالت مشخص به یک نتیجه یکسان همیشه ختم میشود.

- محیط غیرقطعی (تصادفی):

در یک محیط تصادفی یک action یکسان در حالت یکسان ممکن است باعث شود که ما به حالت متفاوتی همراه با پاداش متفاوت برویم. عنصر تصادفی و عدم قطعیت در انتقال این محیط وجود دارد. این عدم قطعیت توسط احتمال بیان میشود. خروجی action ها در این محیط لزوما قابل پیشبینی نیست حتی اگر ما حالت فعلی و action فعلی بدانیم. البته میتوانیم بین حالت های مختلف و پاداش های مختلف یک توزیع احتمالاتی داشته باشیم. منبع این عدم قطعیت میتواند علل مختلفی داشته باشد: ۱. ویژگی خود محیط، مثلا خود محیط شاید درگیر noise باشد یا وقفه های پیشبینی نشده خارجی. ۲. نیمه مشاهده پذیر بودن محیط یعنی اینکه agent اطلاعات کافی نسبت به محیط نداشته باشد. ۳. انتقال تصادفی، به این معنی که خود محیط در انتقال بین حالت های مختلف دارای عدم قطعیت است و یک action میتواند خروجی های مختلفی داشته باشد.

محیط های قطعی راحت تر قابل یادگیری دارند برای عامل. همچنین این محیط ها امکان planning را به عامل میدهد به دلیل اینکه خروجی ها مشخص و معین هستند و با بهره گیری از برنامه نویسی پویا میتوانیم سیاست بهینه را انتخاب بکنیم.

در مقابل محیط های غیرقطعی نیاز به استفاده از رویکرد model-free دارند و عامل ما احتمالات و چالش های یادگیری در محیط غیر قطعی سروکار دارد.



سوال سوم

مسائلی که در RL وجود دارند میتوانند توسط MDP به صورت زیر مدل شوند:

اجزای MDL:

States (S): حالت های مسئله ما هستند که میتوانند بیانگر یک موقعیت خاص از شرایط اصلی مسئله باشند.

Actions: اقداماتی هستند که عامل میتواند در محیط انجام دهد.

Transition probabilities: در این مرحله جایی است که عنصر عدم قطعیت وارد مسئله میشود. برای هر action در هر حالتی که انجام میدهیم یک احتمالی بین حالت بعدی توزیع میشود.

Reward: پاداشی است که هر حالت به ما میدهد. پاداش میتواند مثبت باشد به معنای نزدیک تر شدن مثلا به هدف و یا جریمه باشد.

Goal: هدف پیدا کردن سیاستی است که به ما میگوید هر حالت چه actionیی را انجام بدهیم. به دنبال سیاست بهینه ای هستیم که پاداش انباشته شده مورد انتظار ما را در طول زمان بیشینه کند. (به نوعی امید ریاضی پاداش انباشته شده را میخواهیم بدست بیاوریم).

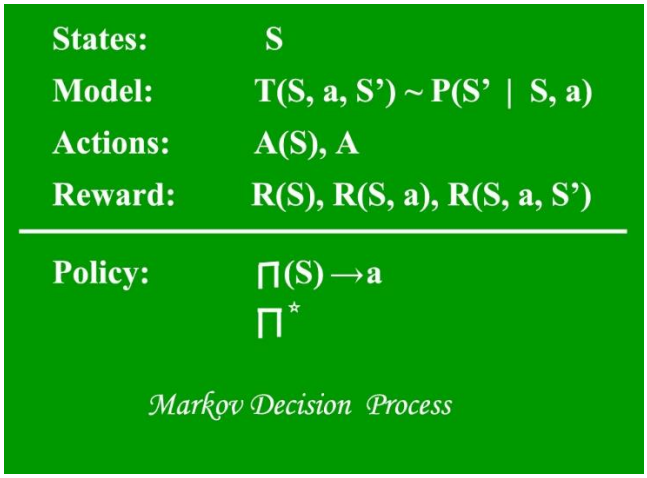
فرآیند تصمیم گیری (پیدا کردن سیاست بهینه):

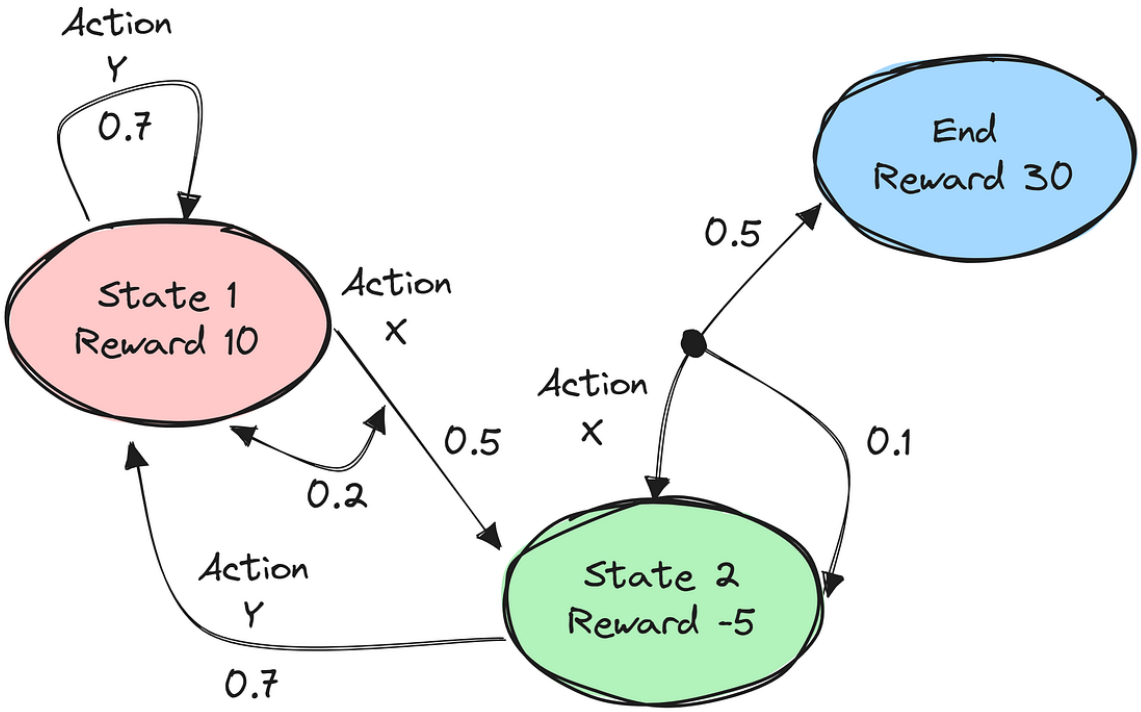
ابتدا برای هر حالت ما یک تابع value تعریف میکنیم تا ببینیم با رسیدن به این حالت چه مقدار پاداش به عامل داده میشود. بعد از آن باید معادله bellman را حل کنیم. این معادله به ما میگوید که value یک حالت برابر است با پاداش لحظه ای در این حالت فعلی بعلاوه value مورد انتظار در حالت بعدی. (امید ریاضی گرفتن از value های حالت بعدی).

بعد از اینکه value را برای هر حالت بدست آوردیم، به دنبال انتخاب سیاست بهینه میرویم. در هر حالت به دنبال actionیی میرویم که بیشترین امید ریاضی یا مقدار مورد انتظار برای value را به ما بدهد.

پس به طور کلی ما با استفاده از MDP شرایط محیطی را توصیف میکنیم که در آن ما در یک حالت قرار داریم و یک action انجام میدهیم که انجام این کار یک سری نتایج به همراه دارد همراه با یک مقدار عدم قطعیت یعنی نتیجه action از قبل مشخص نیست و ممکن است به حالت جدید برویم. بر اساس حالتی که به آن رفتیم پاداش دریافت میکنیم یا جریمه میشویم. هدف این است که پاداش خودمان را در طول زمان بیشینه کنیم. رویکرد Markov به ما میگوید که آینده فقط به حال فعلی بستگی دارد نه به گذشته یعنی حالت بعدی و پاداشی که دریافت خواهی کرد به حالت فعلی و actionیی که انجام میدهی بستگی دارد و اهمیتی ندارد چطور به حالت فعلی رسیدی در واقع Markov به ما یک نبود حافظه یا memorylessness میدهد

تا مسئله ما ساده شود. در عکس زیر به خوبی این نماد پردازی در تصمیم مارکوف نمایان گر است اگر چه که به ضریب discount اشاره نشده است. این ضریب یک عددی بین ۰ و ۱ است که اهمیت پاداش های آینده را نسبت به پاداش لحظه ای در حالت فعلی را مشخص میکند. هر چه این مقدار بزرگتر باشد ما به پاداش های دراز مدت بیشتر اهمیت میدهیم، این ضریب را با gamma یونانی نمایش میدهیم. سیاست بهینه با π^* مشخص شده است. همچنین در این تصویر به تابع action value که با Q نمایش داده میشود اشاره نشده است که امید ریاضی یا مقدار مورد انتظار پاداش discounted تجمعی را بر اساس حالت شروع، action هایی که انجام میدهیم و انتخاب یک سیاست به ما میدهد. همچنین دو معادله bellman برای تابع value و action داریم. که جمع امید ریاضی هایی مختلف را برای حالت های مختلف به ما میدهد.




Action
Y
0.7
State 1
Reward 10
Action
X
0.5
0.2
Action
Y
0.7
State 2
Reward -5
Action
X
0.5
End
Reward 30
0.1