Value iteration implementation for the Gambler's Problem

در این سوال قصد داریم در قالب حل مسئله قمارباز (Gambler Problem) با الگوریتم value iteration آشنا شویم.

مسئله مطابق با فرض های صورت تمرین تعریف می‌شود. مطابق با الگوریتم value iteration در پیاده سازی، state ها، دارایی هستند. همچنین action ها مقدار سرمایه ای است که روی آن شرط بندی می‌شود. برای ارزش state ها یک آرایه با مقادیر اولیه صفر تعریف می‌شود. همچنین در نهایت، بهترین action در هر state را در یک آرایه (policy) ذخیره می‌کنیم. همچنین آرایه ای که مطابق با پاداش هر state است در تمامی اندیس ها صفر بوده و تنها در state آخر (پایان بازی) ۱ می‌باشد. برای پیاده سازی الگوریتم مطابق با تصویر ۲-۱ عمل می‌کنیم. به این منظور که در یک حلقه بی‌نهایت یک Δ با مقدار اولیه صفر تعریف می‌کنیم و سپس بعد از به‌روز کردن ارزش هر state مقدار آن را با اختلاف ارزش قبلی و بعدی state جایگزین می‌کنیم. (در صورتی که این مقدار از مقدار سابق Δ کوچکتر باشد، جایگزینی صورت نمی‌گیرد.) در حقیقت Δ بزرگترین تغییر ارزش state ها را در هر بار به‌روزرسانی تمامی ارزش ها نشان می‌دهد. برای به روز کردن ارزش ها مطابق با رابطه تصویر ۲-۱، عمل می‌کنیم. نکته قابل توجه این است که از یک state خاص نمی‌توانیم به تمامی state های دیگر برویم. زیرا در صورتی که کمتر از 50$ داشته باشیم نمی‌توانیم روی مقداری بیشتر از سرمایه خود شرط بندی کنیم و در صورتی که بیشتر از 50$ نیز داشته باشیم، معقولانه است که نهایتاً طوری شرط بندی کنیم که به 100$ برسیم. در نتیجه state های بعدی را فقط state هایی در نظر می‌گیریم که در این دو شرط صدق کند. برای به روز کردن ارزش ها تابع Best_NextState() نوشته شده است. در انتها نیز برای انتخاب بهترین action در هر state یک‌بار Best_NextState() را فراخوانی می‌کنیم. این الگوریتم به طور کلی در تابع Value_Iteration() پیاده سازی شده است.

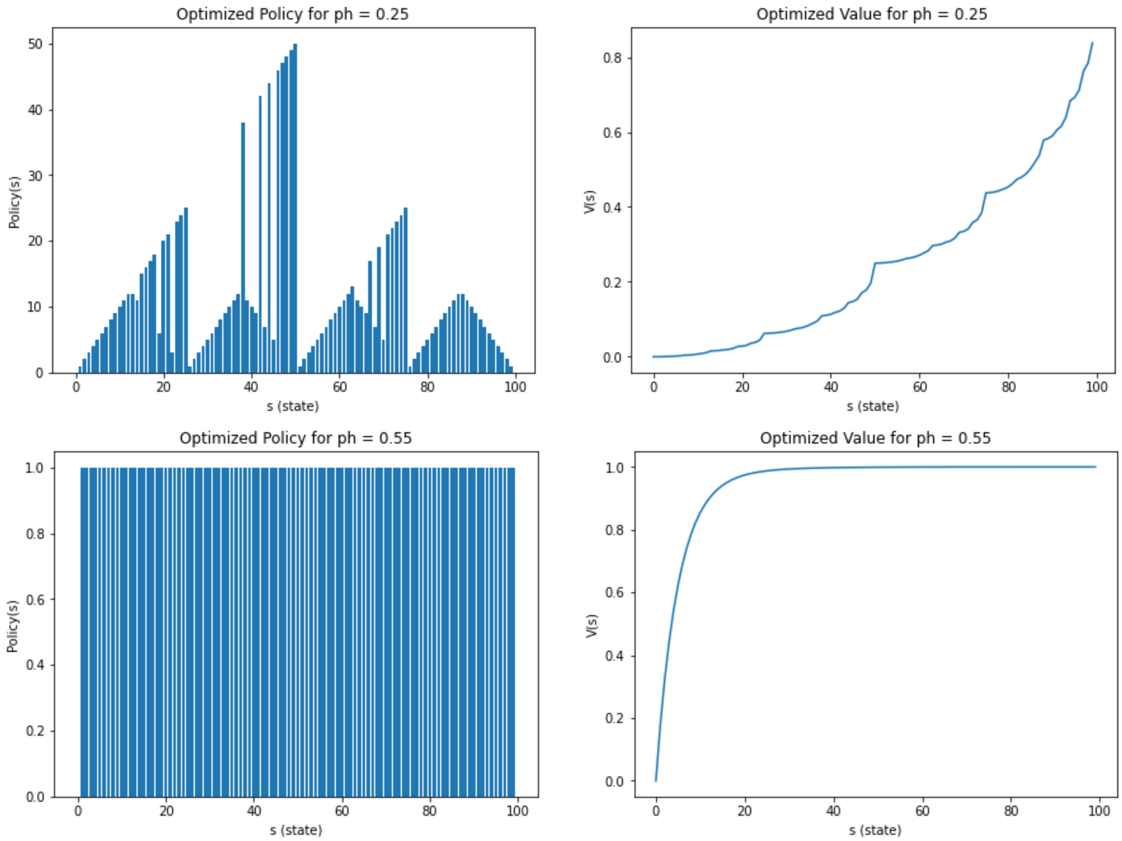
Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
$\quad \Delta \leftarrow 0$
$\quad$ For each $s \in \mathcal{S}$:
$\qquad v \leftarrow V(s)$
$\qquad V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
$\qquad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
$\quad \pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

تصویر ۲-۱: الگوریتم value iteration

تصویر ۲-۲ نمودار های ارزش و policy برای دو احتمال مختلف $p_h = 0.25$ و $p_h = 0.55$ نشان می‌دهد.



تصویر ۲-۲: نمودار های ارزش و policy در مسئله قمارباز برای دو احتمال مختلف

همانطور که در تصاویر بالا دیده می‌شود به نکات زیر می‌رسیم:

۱- هر چه به state های انتهایی نزدیک‌تر می‌شویم، ارزش آن بیشتر می‌شود.

۲- اگر احتمال شیر آمدن بزرگتر از 0.5 باشد، می‌دانیم که در هر صورت (اگر چه کند باشد) ولی بالاخره برنده بازی خواهیم شد و بنابراین همیشه روی 1$ شرط بندی می‌کنیم. در نتیجه هنگامی که در state های بالاتر هستیم در صورت باخت ارزش state چندان تغییری نمی‌کند. به بیان دیگر چون state های مجاور در صورت برد یا باخت عوض می‌شوند (چون روی 1$ شرط بندی کرده ایم، تغییری در ارزش های state های بالاتر احساس نمی‌شود.

۳- اگر احتمال شیر آمدن کوچکتر از 0.5 باشد، می‌دانیم که شانس برنده شدن در هر بار کم است؛ در نتیجه روی مقدار زیادی شرط بندی می‌کنیم. یعنی اگر تعداد دفعات زیادی شرط بندی کنیم می‌دانیم در تعداد کمی از آنها برنده هستیم. پس در بعضی مواقع مثلا state های 25 و 50 روی کل پول شرط بندی می‌کنیم تا زودتر برنده یا بازنده شویم. (اگر تعداد دفعات شرط بندی

زیاد باشد در هر صورت تعداد باخت ها از برد ها بیشتر است. اگر تعداد دفعات کم باشد، ممکن است با یک شیر آمدن بازی را برنده شویم.)