Capítulo 4 - Sutton

Paulo Henrique Albuquerque

2023-04-25

1 Programação Dinâmica

Algoritmos clássicos de programação dinâmica são de uso limitado pois eles assumem um modelo perfeito, além de terem um grande custo computacional.

Porém, os algoritmos de DP formam uma base essencial para entender outros métodos. Esses outros métodos geralmente tentam replicar os algoritmos de DP com menos computação e sem assumir um modelo ideal. O objetivo primário dos algoritmos de DP é computar funções valor.

1.1 Avaliação de Política

Consideramos o problema de calcular v_π para um política π arbitrária. Lembre-se de que,

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')].$$

A equação acima representa um sistema linear de $|\mathcal{S}|$ equações lineares, que pode ser resolvido atráves de uma computação direta. Para os nossos propósitos, um método interativo é mais adequado. Considere uma sequência de aproximações para a função valor: $v_1, v_2, \ldots$. A primeira aproximação é escolhida de forma arbitrária (exceto para estados terminais, que devem ter valor 0), e cada aproximação sucessiva é obtida usando a equação de Bellman como uma regra de atualização:

$$v_{k+1}(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s', r, |s, a)[r + \gamma v_k(s')],$$

para todo $s \in \mathcal{S}$. Claramente, v_π é um ponto fixo para essa regra de atualização, pois a equação de Bellman garante a igualdade nesse caso. De fato, a sequência $\{v_k\}$ converge para v_π a medida que $k \to \infty$ sob as mesmas condições que garantem a existência de v_π. Esse algorimo é chamado de *avaliação iterativa de política*.

Na implementação do algoritmo podemos utilizar um array e atualizar os valores in place. Pode-se ser, então, que novos valores sejam usados em vez de valores antigos no lado direito da equação da regra de atualização. Essa algoritmo levemente modificado também funciona e, usualmente, converge mais rápido. Geralmente, utilizamos a versão in place quando pensamos em algoritmos de DP.

O algoritmo de avaliação iterativa de política é dado abaixo.

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Figure 1: Algoritmo de avaliação iterativa de política

1.2 Melhoria de política

Para sabermos se uma política pode ser melhorada comparamos o valor de $v_\pi(s)$ com $q_\pi(s,a)$ para toda ação $a \in \mathcal{A}(s)$. Caso algum $q_\pi(s,a)$ seja maior que $v_\pi(s)$, a política pode ser melhorada dando preferência à ação a quando o agente está no estado s. Anunciamos essa observação através do seguinte teorema.

Melhoria de política: Sejam π e π' duas políticas determinísticas tais que, para todo $s \in \mathcal{S}$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s).$$

Então, a política π' é tão boa quanto π, ou até melhor:

$$v_{\pi'}(s) \geq v_\pi(s),$$

para todo $s \in \mathcal{S}$. Podemos estender essa argumentação para todos os estados. Então, para cada estado, procuramos a ação que maximize o retorno:

$$\pi' = \arg\max_a q_\pi(s,a) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')].$$

O processo de construir uma nova política melhorada, ao fazê-la gulosa em relação às funções valor da política original, é chamado de *melhoria de política*.

Suponha que ao construir uma nova política π' através da melhoria de política, obtemos uma política não melhor que π. Ou seja, $v_{\pi'} = v_\pi$. Pela equação acima, segue que, para todo $s \in \mathcal{S}$:

$$v_\pi(s') = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_{\pi'}(s')].$$

Mas a equação acima é idêntica à equação de Bellman ótima. Ou seja, $v_{\pi'} = v_\pi = v_\star$.

Até agora, consideramos políticas determinísticas. Ao considerarmos políticas estocásticas, basta fazermos a seguinte modificação natural:

$$q_\pi(s, \pi'(s)) = \sum_a \pi'(a|s) q_\pi(s,a).$$

Além disso, se várias ações são maximizadoras, não precisamos selecionar somente uma ação. Cada ação dessas pode ser dada uma porção da probabilidade de ser selecionada na nova política gulosa. É claro que toda ação sub-maximal deve ter probabilidade zero.

1.3 Iteração de política

A partir do momento que uma política π foi melhorada usando v_π para obter uma política melhorada $v_{\pi'}$, podemos então computar $v_{\pi'}$ e melhora-lá denovo para obter uma política ainda melhor π''. Podemos, portanto, obter uma sequência de políticas monotonicamente melhoradas e funções valor:

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \dots \xrightarrow{\text{I}} \pi_\star \xrightarrow{\text{E}} v_\star,$$

onde $\xrightarrow{\text{E}}$ denota uma avaliação de política e $\xrightarrow{\text{I}}$ denota uma melhoria de política. Cada política é, garantidamente, uma melhoria estrita em relação à passada se não for uma política ótima. Para MDPs finitos, o processo converge para uma política ótima em um número finito de passos, visto que o número de políticas é finito. (Porque? Só para políticas determinísticas, não?). Veja o algoritmo abaixo.

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

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

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      $a \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
      If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2

Figure 2: Algoritmo de iteração de políticas

Observe que há um bug. É possível que o algoritmo acima nunca termine, quando o processo fica alterando entre duas políticas igualmente boas (como isso é possível?). Note também que em cada avaliação de política, a função valor é inicializada com o valor da função da política anterior. Isso faz que o processo, em geral, convirga mais rapidamente.

1.4 Jack's Car Rental

Nessa seção, fazemos uma apresentação do problema *Jack's Car Rental* junto com uma solução. A descrição do problema é a seguinte:

Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited 10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of 2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables, meaning that the probability that the number is , where is the expected number. Suppose is 3 and 4 for rental requests at the first and second locations and 3 and 2 for returns. To simplify the problem slightly, we assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night. We take the discount rate to be = 0.9 and formulate this as a continuing finite MDP, where the time steps are days, the state is the number of cars at each location at the end of the day, and the actions are the net numbers of cars moved between the two locations overnight.

Para formular o problema como um MDP finito, definimos as seguintes variáveis aleatórias:

$X_t =$ "*quantidade de carros na localização 1 (L_1) ao fim do dia t, após contabilizar as transferências de carros entre localizações na noite do dia anterior e os pedidos e devoluções ao longo do dia t.*"

$Y_t =$ "*quantidade de carros na localização 2 (L_2) ao fim do dia t, após contabilizar as transferências de carros entre localizações na noite do dia anterior e os pedidos e devoluções ao longo do dia t.*"

$A_t =$ *"quantidade de carros movidos ao final da noite do dia t da localização 1 para a localização 2"*

Note que,

$$-\min(y, MAX\_MOVES) \leq A_t \leq \min(x, MAX\_MOVES),$$

onde $MAX\_MOVES = 5$ para essa instância do problema.

$P_t^{(1)} =$ "*quantidade total de pedidos concretizados ao longo do dia $t+1$.*"

O valor desse variável é limitado pela quantidade de carros disponíveis no início do dia $t+1$, $X_t - A_t$. Se $\overline{P_t}^{(1)}$ é a quantidade de pedidos no dia $t+1$ (observe que essa quantidade pode ser maior que 20: é uma variável Poisson independente),

$$P_t^{(1)} = \min(\overline{P_t}^{(1)}, X_t - A_t).$$

Analogamente,

$$P_t^{(2)} = \min(\overline{P_t}^{(2)}, Y_t + A_t).$$

$D_t^{(1)} =$ *"quantidade total de devoluções concretizadas na localização 1 ao final do dia t."*

$D_t^{(2)} =$ *"quantidade total de devoluções concretizadas na localização 2 ao final do dia t."*

Consideramos que o carros são devolvidos no final do dia, após terem sidos processados os pedidos naquele dia. Além disso, a quantidade final de carros na localização deve ser menor que $MAX\_CARS = 20$. Então, se $\overline{D_t}^{(1)}$ é a quantidade de devoluções totais no dia $t+1$,

$$D_t^{(1)} = \min(\overline{D_t}^{(1)}, MAX\_CARS - (X_t - A_t - P_t^{(1)})).$$

Analogamente,

$$D_t^{(2)} = \min(\overline{D_t}^{(2)}, MAX\_CARS - (Y_t + A_t - P_t^{(2)})).$$

As variáveis $\overline{P_t}^{(1)}$, $\overline{P_t}^{(2)}$, $\overline{D_t}^{(1)}$ e $\overline{D_t}^{(2)}$ seguem distribuições de Poisson bem definidas, conforme a descrição do problema.

$R_t =$ *"recompensa ao final do dia t, contabilizando o custo de transferências de carros da noite do dia anterior t e o lucro devido aos pedidos concretizados ao longo do dia $t+1$."*

Dadas essas definições, podemos escrever as seguintes relações.

$$X_{t+1} = X_t - A_t + D_t^{(1)} - P_t^{(1)}$$

$$Y_{t+1} = Y_t + A_t + D_t^{(2)} - P_t^{(2)}$$

$$R_{t+1} = -2|A_t| + 10(P_t^{(1)} + P_t^{(2)})$$

Agora, descrevemos o MDP arcaboço para o problema. O estado do MDP é o par $S_t = (X_t, Y_t)$. Então, o conjunto de estados S é dado por

$$S = \{(x, y) : x, y \in [0, MAX\_CARS]\}.$$