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 sua solução.

O problema consiste