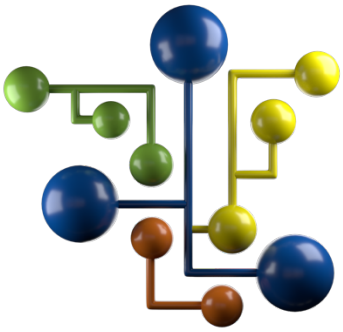


Data Science Academy

www.datascienceacademy.com.br

Introdução à Inteligência Artificial

Convergência da Iteração de Valor

Dissemos que a iteração de valor eventualmente converge para um único conjunto de soluções das equações de Bellman. Agora explicaremos por que isso acontece. Introduziremos algumas ideias matemáticas úteis ao longo do processo e obteremos alguns métodos para avaliar o erro na função utilidade devolvida quando o algoritmo é terminado prematuramente; isso é útil porque significa que não teremos de continuar para sempre.

O conceito básico usado para mostrar que a iteração de valor converge é a noção de contração. A grosso modo, uma contração é uma função de um único argumento que, ao ser aplicada a duas entradas diferentes, uma de cada vez, produz dois valores de saída que estão "mais próximos entre si" por pelo menos um fator constante, em relação às entradas originais. Por exemplo, a função "dividir por dois" é uma contração porque, depois de dividirmos dois números quaisquer por dois, sua diferença é reduzida à metade. Note que a função "dividir por dois" tem um ponto fixo, isto é, zero, que não é alterado pela aplicação da função. A partir desse exemplo, podemos distinguir duas propriedades importantes de contrações:

• Uma contração tem apenas um ponto fixo; se houvesse dois pontos fixos, eles não ficariam mais próximos um do outro quando a função fosse aplicada e não seria uma contração.
• Quando a função é aplicada a qualquer argumento, o valor deve ficar mais próximo do ponto fixo (porque o ponto fixo não se move) e, assim, a aplicação repetida de uma contração sempre alcança o ponto fixo no limite.

Agora, vamos supor que visualizamos a atualização de Bellman como um operador B que é aplicado simultaneamente para atualizar a utilidade de todo estado. Seja Ui o vetor de utilidades para todos os estados na i-ésima iteração. Então, a equação da atualização de Bellman pode ser escrita como

$$U_{i+1} \leftarrow BU_i.$$

Em seguida, precisamos de um modo de medir distâncias entre vetores de utilidade. Utilizaremos a norma max, que mede o "comprimento" de um vetor pelo valor absoluto de seu maior componente:

$$||U|| = \max_s |U(s)|$$

Com essa definição, a "distância" entre dois vetores, ||U – U'||, é a diferença máxima entre dois elementos correspondentes quaisquer. O principal resultado desta seção é: sejam Ui e dois vetores de utilidade quaisquer. Então, temos:

$$||BU_i - BU'_i|| \leq \gamma ||U_i - U'_i||.$$

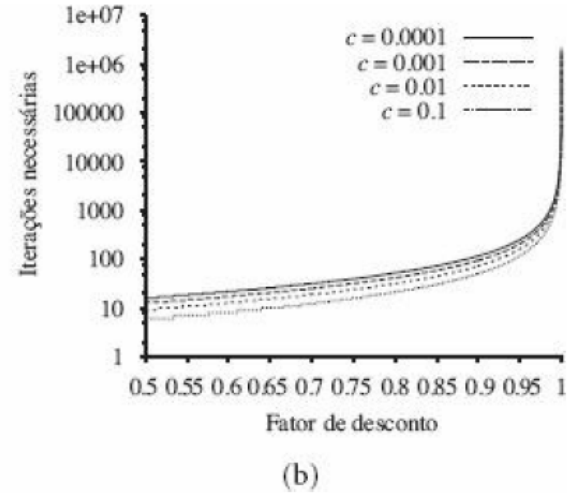
Isto é, a atualização de Bellman é uma contração por um fator γ no espaço de vetores de utilidade. Assim, a partir das propriedades de contrações em geral, segue que a iteração de valor sempre converge para uma solução única das equações de Bellman, sempre que γ < 1. Podemos também utilizar a propriedade de contração para analisar a taxa de convergência para uma solução. Em particular, podemos substituir na equação pelas utilidades verdadeiras U, para as quais BU = U. Então, obtemos a desigualdade:

$$||BU_i - U|| \leq \gamma ||U_i - U||$$

Assim, se visualizarmos ||Ui – U|| como o erro na estimativa Ui, veremos que o erro é reduzido por um fator de pelo menos γ em cada iteração. Isso significa que a iteração de valor converge de forma exponencialmente rápida. Podemos calcular o número de iterações necessárias para alcançar um limite de erro especificado ε, da seguinte forma: primeiro, vimos que as utilidades de todos os estados são limitadas por ±Rmax/(1 – γ). Isso significa que o erro inicial máximo ||U0 – U|| ≤ 2Rmax/(1 – γ). Vamos supor que sejam realizadas N iterações para alcançar um erro de no máximo ε. Então, como o erro é reduzido por pelo menos γ em cada vez, é necessário que (γN · 2Rmax/(1 – γ) ≤ ε. Usando logaritmos, descobrimos que

$$N = \lceil \log(2R_{max}/\epsilon(1-\gamma))/\log(1/\gamma) \rceil$$

iterações bastam.



(b)

A Figura acima mostra como N varia com γ para diferentes valores da razão ε/Rmax. A boa notícia é que, devido à convergência exponencialmente rápida, N não depende muito da razão ε/Rmax. A má notícia é que N cresce rapidamente à medida que γ se aproxima de 1. Podemos obter a convergência rápida se tornarmos γ pequeno, mas isso efetivamente dá ao agente um horizonte curto e pode anular os efeitos a longo prazo das ações do agente.

O limite de erro no parágrafo anterior nos dá alguma ideia dos fatores que influenciam o tempo de execução do algoritmo, mas às vezes é excessivamente conservador como um método para decidir quando interromper a iteração. Para este último propósito, podemos utilizar um limite relacionando o erro ao tamanho da atualização de Bellman em qualquer iteração dada. A partir da propriedade de contração, podemos mostrar que, se a atualização for pequena (isto é, se a utilidade não mudar muito para nenhum estado), então o erro, comparado à função utilidade verdadeira, também será pequeno. Mais precisamente,

$$\text{se} \quad ||U_{i+1} - U_i|| < \epsilon(1-\gamma)/\gamma, \quad \text{então} \quad ||U_{i+1} - U|| < \epsilon$$

Essa é a condição de término usada no algoritmo ITERAÇÃO-DE-VALOR:

função ITERAÇÃO-DE-VALOR(*mdp*, ε) **retorna** uma função utilidade
entradas: *mdp*, um MDP com estados *S*, ações *A*(*s*), modelo de transição *P*(*s'*|*s*,*a*), recompensas *R*(*s*), desconto γ,
ε, o erro máximo permitido na utilidade de qualquer estado
variáveis locais: *U*, *U'*, vetores de utilidades para estados em *S*, inicialmente zero
δ, a mudança máxima na utilidade de qualquer estado em uma iteração

repita
$U \leftarrow U'$; $\delta \leftarrow 0$
para cada estado *s* **em** *S* **faça**
$U'[s] \leftarrow R[s] + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
se $|U'[s] - U[s]| > \delta$ **então** $\delta \leftarrow |U'[s] - U[s]|$
até $\delta < \epsilon(1-\gamma)/\gamma$
retornar *U*

Referências:

Livro: Inteligência Artificial
Autor: Peter Norvig