Universidade Federal de Viçosa
Centro de Ciências Exatas e Tecnológicas
Departamento de Informática

Processos de Decisão de Markov
Levi H. S. Lelis
levi.lelis@ufv.br

Algoritmos de Programação Dinâmica

Nessa aula veremos algoritmos de programação dinâmica para solucionar Processos de Decisão de Markov (PDM). O primeiro algoritmo que veremos é o de Avaliação de Política, que resolve o que é conhecido como o problema de previsão: dada uma política π, qual o valor v dos estados no problema?

Avaliação de Política (AP)

Denotaremos com a letra maiúscula $V(s)$ uma aproximação do valor de $v_\pi(s)$. No algoritmo de AP os valores de V são inicializados de forma arbitrária e então utilizamos a equação de Bellman como uma regra de atribuição.

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

O valor de $V(s)$ converge para $v_\pi(s)$ ao repetirmos várias vezes a atribuição acima para todos os estados s.

Exemplo Considere o problema do grid abaixo, onde o agente é permitido mover-se para cima (C), baixo (B), direita (D) ou esquerda (E). A dinâmica do ambiente é determinística. Por exemplo, se o agente está na célula 5 e se move para direita, ele para na célula 6. Se o agente tenta se mover para fora dos limites do grid ele fica parado. Os estados em cinza são terminais. Uma vez que um estado terminal é atingido o episódio é finalizado. A recompensa é de -1 para qualquer ação de qualquer estado do problema.

1	2	3
4	5	6
7	8	9

Os grids abaixo mostram os valores V para cada s, com $\gamma = 1$, depois das iterações $0, 1, 2, \infty$.

0	0	0
0	0	0
0	0	0

0	-1	-1
-1	-1	-1
-1	-1	0

0	-1.7	-2
-1.7	-2	-1.7
-2	-1.7	0

0	-7	-9
-7	-8	-7
-9	-7	0

Por exemplo, o valor de $V(2)$, temos o seguinte, dado que todos os valores de V são inicializados com 0:

$$V(2) = \overbrace{0.25(-1+0)}^{\text{ação = C}} + \overbrace{0.25(-1+0)}^{\text{ação = B}} + \overbrace{0.25(-1+0)}^{\text{ação = D}} + \overbrace{0.25(-1+0)}^{\text{ação = E}}$$
$$V(2) = -1$$

Após calcular os valores de V na primeira iteração, calculamos então os novos valores de V utilizando os novos valores. Por exemplo:

$$V(2) = \overbrace{0.25(-1-1)}^{\text{ação = C}} + \overbrace{0.25(-1-1)}^{\text{ação = B}} + \overbrace{0.25(-1-1)}^{\text{ação = D}} + \overbrace{0.25(-1+0)}^{\text{ação = E}}$$
$$V(2) = -1.75$$

Dado que S é o conjunto de estados, AP avalia π dado um valor de γ e um limiar de precisão numérica θ.

```
def AP(π, γ, θ):
  inicialize V(s) de forma arbitrária.
  (V(s) = 0 para estados terminais)
  Δ ← ∞
  while Δ ≥ θ:
    Δ ← 0
    for s in S:
      v ← V(s)
      V(s) ← Σ_a π(a|s) Σ_{s',r} p(s', r|s, a)[r + γV(s')]
      Δ ← max(Δ, |V(s) − v|)
  return V
```

Melhoramento de Política

O procedimento AP nos permite avaliar uma política. Considere a seguinte situação. Dado que sabemos os valores V para todos os estados s, o que acontece com o valores esperado de recompensa se o agente tomar uma ação específica em um estado? Essa ideia é capturada pela equação abaixo.

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

Se temos que $q_\pi(s,a) > v_\pi(s)$, então a política π' que é idêntica a π em todos os estados menos em s, onde π' escolhe a, é tão boa ou melhor que π: $v_{\pi'}(s) \geq v_{\pi'}(s)$ para todo s.

Essa ideia pode ser generalizada da seguinte forma: para todos os estados s, escolha a ação é que melhor em s de acordo com $q_\pi(s,a)$, gerando uma política gulosa π'.

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

Exemplo O valor de $\pi'(2)$ dada a política aleatória π para o problema do grid é E, que é de fato uma melhoria sobre a política anterior.

Convergência Se após uma melhoria da política nós tivermos que $\pi'(s) = \pi(s)$ para todos os s, então nós temos que,

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

que é a equação de otimalidade de Bellman para $v_*(s)$, vista na última aula. Portanto, quando não há melhorias a serem feitas, $v_{\pi'}$ convergiu em v_*.

Iteração de Política (IP)

O algoritmo de iteração de política alterna entre avaliar uma política e melhora-la. Os grids abaixo mostram como a política muda após o passo de melhoria. Inicialmente todas as ações são escolhidas com mesma probabilidade. Após o passo de melhoria, a política no estado 7 passa a ser ou C ou D. A política exibida na tabela à direita é ótima para esse problema.

0	0	0
0	0	0
0	0	0

0	-7	-9
-7	-8	-7
-9	-7	0

	C,B,D,E	C,B,D,E
C,B,D,E	C,B,D,E	C,B,D,E
C,B,D,E	C,B,D,E	

0	E	E,B
C	C,B,D,E	B
C,D	D	0

```
1 def IP(γ, θ):
2   inicialize V(s) de forma arbitrária.
3   (V(s) = 0 para estados terminais)
4   inicialize π de forma arbitrária.
5
6   estavel ← False
7   while not estavel:
8     estavel ← True
9     V = AP(π, γ, θ)
10    for s in S:
11      a ← π(s)
12      π(s) ← arg max_a Σ_{s',r} p(s', r|s, a)[r + γV(s)]
13      if π(s) ≠ a:
14        estavel ← False
15  return π
```

O algoritmo de IP não precisa executar o algoritmo AP até a convergência em cada uma de suas iterações. Ele pode executar alguns passos do AP e utilizar os valores calculados até então para atualizar a política. Um caso especial dessa ideia é quando o algoritmo AP termina após uma passagem por todos os estados, retornando então para a melhoria de política. Esse algoritmo é conhecido como Iteração de Valor (IV).

Iteração de Valor (IV)

Uma outra forma de ver o algoritmo IV é através do uso iterativo da equação de otimalidade de Bellman para v.

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')]\,,$$

onde os novos valores $v_{k+1}(s)$ são calculados através de uma escolha gulosa considerando os valores de v_k. Os valores de v_0 são inicializados de forma arbitrária e v_k possui a garantia de convergir para v_*.

```
def IV(γ, θ):
  inicialize V(s) de forma arbitrária.
  (V(s) = 0 para estados terminais)
  Δ ← ∞
  while Δ > θ:
    Δ ← 0
    for s in S:
      v ← V(s)
      V(s) ← max_a Σ_{s',r} p(s', r|s, a)[r + γV(s')]
      Δ ← max(Δ, |V(s) − v|)
  π(s) ← arg max_a Σ_{s',r} p(s', r|s, a)[r + γV(s')] para todo s ∈ S
  return π
```

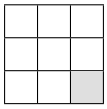
Programação Dinâmica Assíncrona

O algoritmo IV é uma versão do IP em que as políticas são avaliadas com apenas uma iteração sob todos os estados. Na verdade, a interação entre **avaliação** e **melhoria** pode ser ainda mais fina. O algoritmo IP pode alterar apenas um valor de v_k em uma iteração, antes de começar a próxima. Esse esquema é conhecido como programação dinâmica assíncrona. Um implementação assíncrona do IV atualiza e lê o mesmo vetor V em cada iteração do comando *while*.

Desde que $0 \leq \gamma < 1$ e que cada estado seja atualizado um número infinito de vezes, implementações assíncronas fazem com que v_k vá convergir para v_*.

Em termos de agentes que interagem com o mundo, essa ideia pode ser aplicada para atualizar os valores dos estados com os quais o agente interage, como veremos nas próximas aulas.

Exemplo: Considere o problema abaixo onde uma transição para o estado terminal (em cinza) fornece uma recompensa de +10. Todas as outras transições fornecem uma recompensa de -1. Os grids abaixo



mostram os valores de v como foram inicializados e após a primeira e a segunda iterações sobre os estados. Nesse exemplo assumimos uma implementação assíncrona onde o vetor v é utilizado para leitura e escrita simultaneamente; $\gamma = 1.0$.

0	0	0
0	0	0
0	0	

-1	-1	-1
-1	-1	10
-1	10	

-2	-2	9
-2	9	10
9	10	