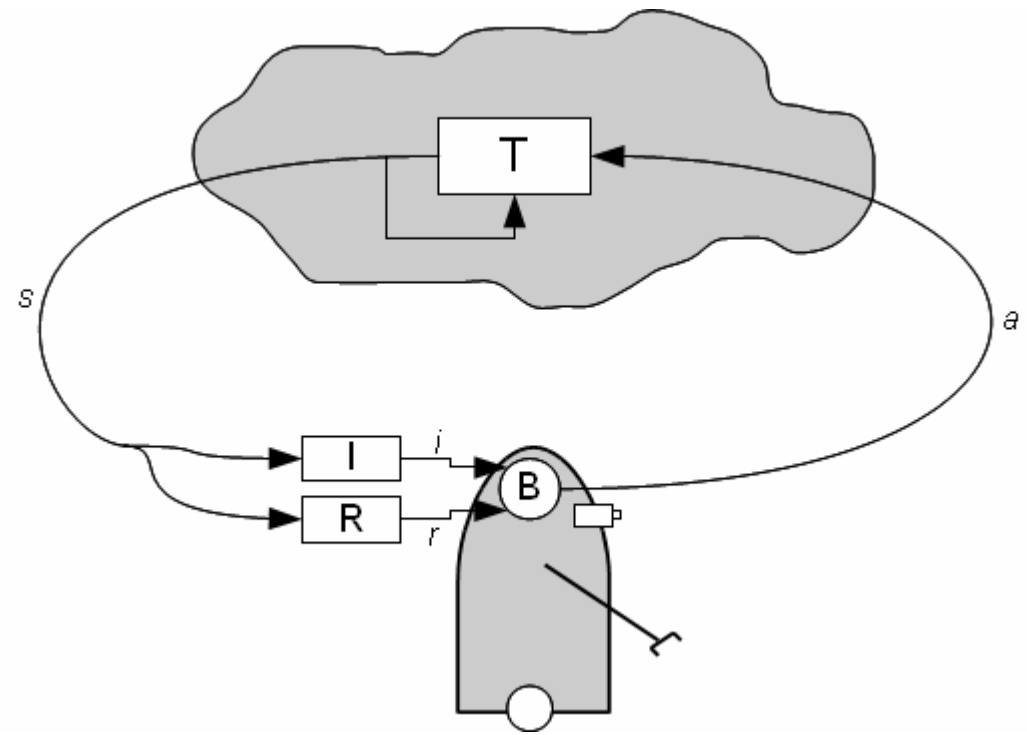
Aprendizado por Reforço

SCC5865-Robótica

Roseli A F Romero

Introdução

O modelo padrão de aprendizado por reforço



Aprendizado por Reforço

Formalmente, o modelo consiste de:

- Um conjunto discreto de estados do ambiente, S;
- Um conjunto discreto de ações do agente, A;
- Um conjunto de sinais de reforço, normalmente {0,1}, ou um número real.

Aprendizado por Reforço

AMBIENTE NON – DETERMINISTICO

Exemplo:

Ambiente: Você está no estado 65. Você possui 4 possíveis ações.
Agente: Eu realizo a ação 2.

Ambiente: Você recebeu um reforço de 7 unidades. Você agora está no estado 15. Você possui 2 possíveis ações.
Agente: Eu realizo a ação 1.

Ambiente: Você recebeu um reforço de -4 unidades. Você agora está no estado 65. Você possui 4 possíveis ações.
Agente: Eu realizo a ação 2.

Ambiente: Você recebeu um reforço de 5 unidades. Você agora está no estado 44. Você possui 5 possíveis ações.

.....

Modelos de Comportamento Ótimo

1- Finite-horizon model

$$E\left(\sum_{t=0}^{h} r_t\right)$$

Otimizar sua recompensa esperada para os prox. h passos

em que, r_t: recompensa escalar recebida no passo t

2 – Infinite-horizont discounted model

$$E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

em que, γ: fator de desconto ($0 <= \gamma < 1$) - taxa de interesse, probabilidade de vida ou limitador da soma infinita.

Modelos de Comportamento Ótimo

3) Average-reward model

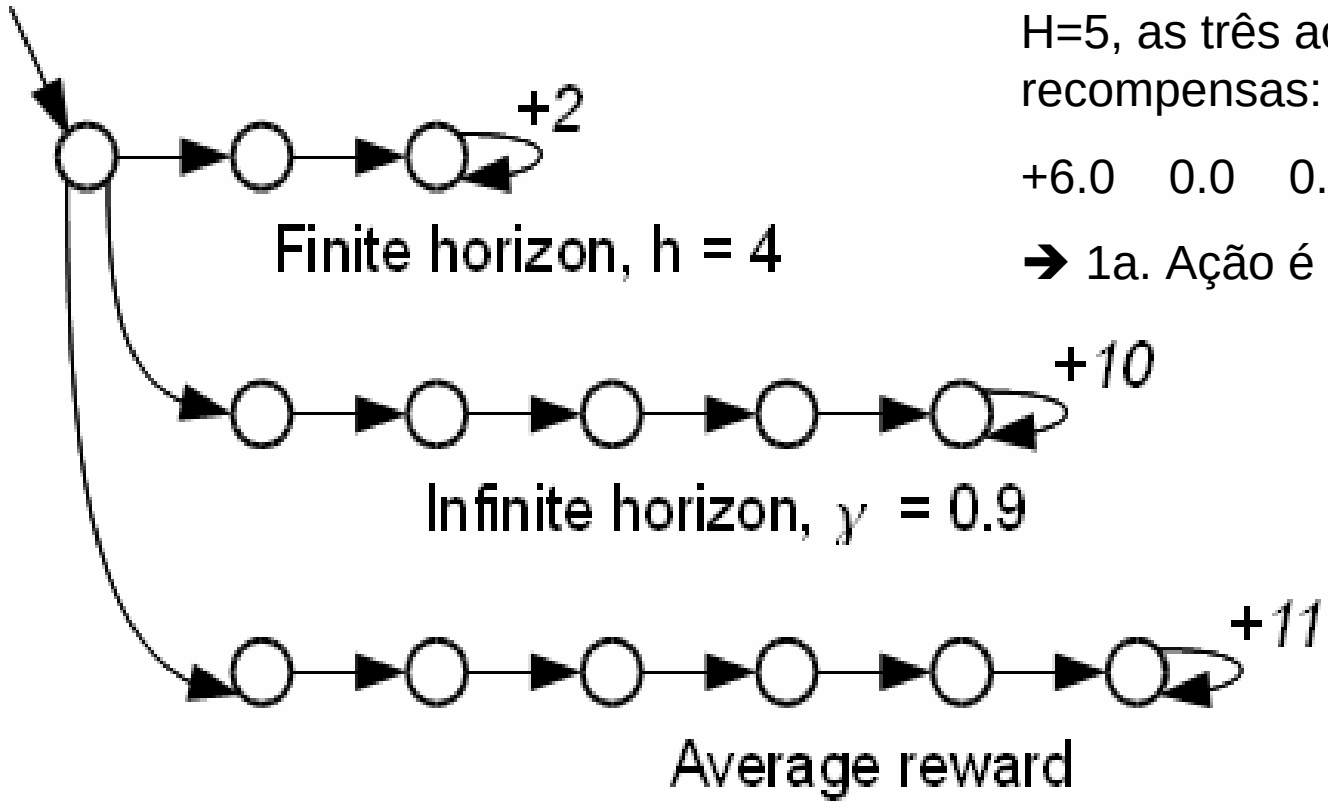
Otimizar a recompensa média de longo termo

$$\lim_{h\to\infty} E\left(\frac{1}{h}\sum_{t=0}^{h} r_t\right)$$

Tal política é conhecida como política de ganho ótima. Pode ser vista como caso limite do 2o. Modelo quando o fator de desconto se aproxima de 1 (Bertsekas, 1995).

Modelos de Comportamento Ótimo

Comparando modelos



H=5, as três ações produzirão recompensas:

+6.0 0.0 0.0

➔ 1a. Ação é escolhida

Medidas de Desempenho do Aprendizado

- Convergência para o ótimo;
- Velocidade de convergência para o ótimo;
- *Regret*;
- Aprendizado por reforço e controle adaptativo.

Programação Dinâmica

Exemplo 1

Considere o problema de programação linear:

$$\max f = 8x_1 + 10x_2$$

$$\text{sujeito a } 4x_1 + 2x_2 \leq 12$$

$$x_1, x_2 \geq 0$$

Resolvendo o problema acima (método simplex), obtém-se:

$$x_1 = 0; x_2 = 6; f = 60$$

Programação Dinâmica

Uma proposta alternativa é determinar a variável gradativamente, através de decomposição do problema em uma série de estágios. A decomposição do problema (1) é ilustrada na Figura 1 abaixo.

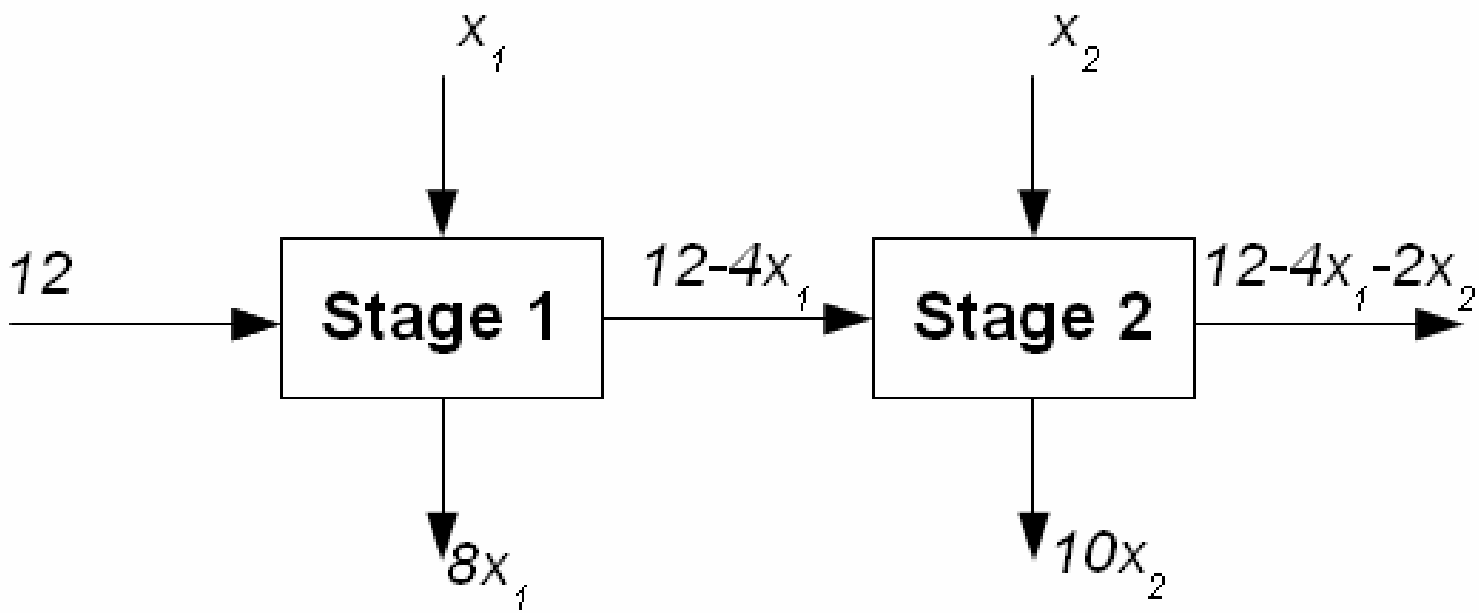


Figura 1 – Diagrama de Fluxo

Programação Dinâmica

$\max x_1 \; f_1 = 10x_2$

sujeito a $2x_2 \leq 12 - 4x_1^*$

$x_2 \geq 0$

Solução:

$x_2 = 6 - 2x_1^*; \; f_1 = 60 - 20x_1^*$

Programação Dinâmica

maxx_1 $f_2 = 8x_1 + f_1 = 60 - 12x_1$

sujeito a $4x_1 \leq 12$

$x_1 \geq 0$

Solução:

$x_1 = 0;\ f_2 = 60$

$x_2 = 6$

Programação Dinâmica

O estágio de entrada do estágio no. n, y_{n-1}, é transformado em um estado de saída y_n através da mudança provocada pela variável de decisão x_n. As mudanças sucessivas do estado do sistema pode ser formalmente descrita pelas equações de tranformação da forma

$$y_n = t_n(y_{n-1}, x_n) \qquad n = 1, 2, ..., N$$

No exemplo elas tem a forma

$$y_1 = y_0 - 4x_1$$

$$y_2 = y_1 - 2x_2$$

Programação Dinâmica

Juntas com as restrições de não negatividade $x_1, x_2, y_1, y_2 \geq 0$, elas são equivalentes as restrições do problema original (1), y_0 sendo igual a 12 e y_2 representando a variável de folga (*slack variable*).

O retorno de cada estágio será dependente, em geral, do estado de entrada e das variáveis de decisão:

$$r_n = r_n(y_{n-1}, x_n) \qquad n = 1, 2, ..., N$$

No exemplo em questão as funções de retorno são da forma simples

$$r_1 = 8x_1 \qquad r_2 = 10x_2$$

Programação Dinâmica

Introduzindo estes símbolos na Figura 1, o diagrama de fluxo do problema de dois estágios é ilustrado segundo a Figura 2.

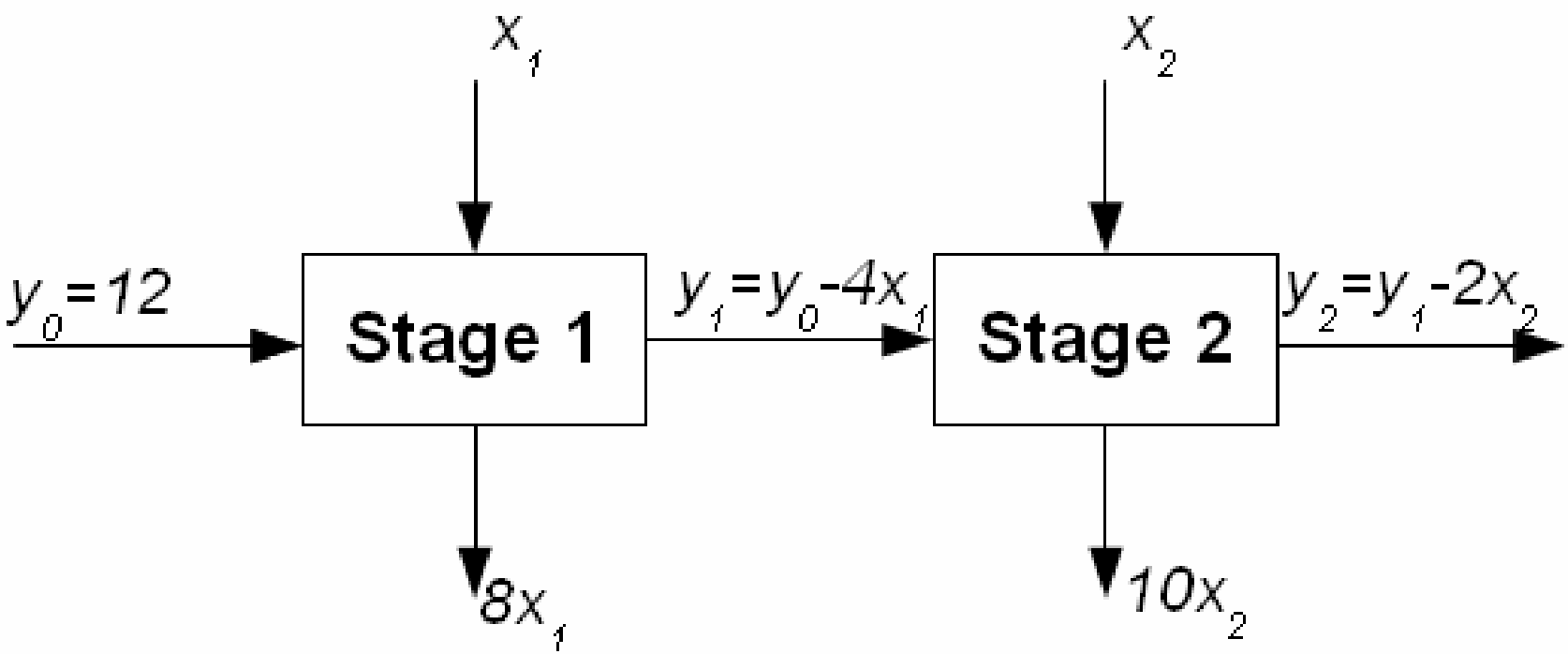


Figura 2 – Diagrama de Fluxo

Programação Dinâmica

maxx_2 $f_1 = r_2(x_2) = 10x_2$

sujeito a $2x_2 \leq y_1$

$x_2 \geq 0$

Solução:

$$x_2(y_1) = 0.5y_1$$

$$F_1(y_1) = 5y_1$$

em que, F_1 denota o valor máximo da função de decisão do estágio,

$$F_1 = \max_{x_2} f_1$$

Programação Dinâmica

$$\max_{x_2} \; f_2 = r_1(x_1) + F_1(y_1) = 8x_1 + 5y_1 =$$

$$= 8x_1 + 5(y_0 - 4x_1) = 5y_0 - 12x_1$$

sujeito a $4x_1 \leq y_0$

$$x_1 \geq 0$$

Solução:

$$x_1(y_0) = 0$$

$$F_2(y_0) = 5y_0$$

em que, $F_2 = \max_{x_1} \; f_2$

Programação Dinâmica

$$F_1(y_1) = \max_{x_2} 10x_2 \quad (0 \le x_2 \le 0.5y_1)$$

$$F_2(y_0) = \max_{x_1} (8x_1 + F_1(y_1))$$

$$= \max_{x_1} [8x_1 + F_1(y_0 - 4x_1)] \quad (0 \le x_1 \le 0.25y_0)$$

em que, $f = F_2(y_0)$

$$\max\ f = \max_{x_1, x_2} [r_1(y_0, x_1) + r_2(y_1, x_2)]$$

Programação Dinâmica

$$F_1(y_1) = \max_{x_2}[r_2(y_1, x_2)]$$

$$F_2(y_0) = \max_{x_1}[r_1(y_0, x_1) + F_1(t_1(y_0, x_1))]$$

O diagrama de fluxo para um sistema de N-estágios é apresentado na Figura 3 (próximo slide).

Programação Dinâmica

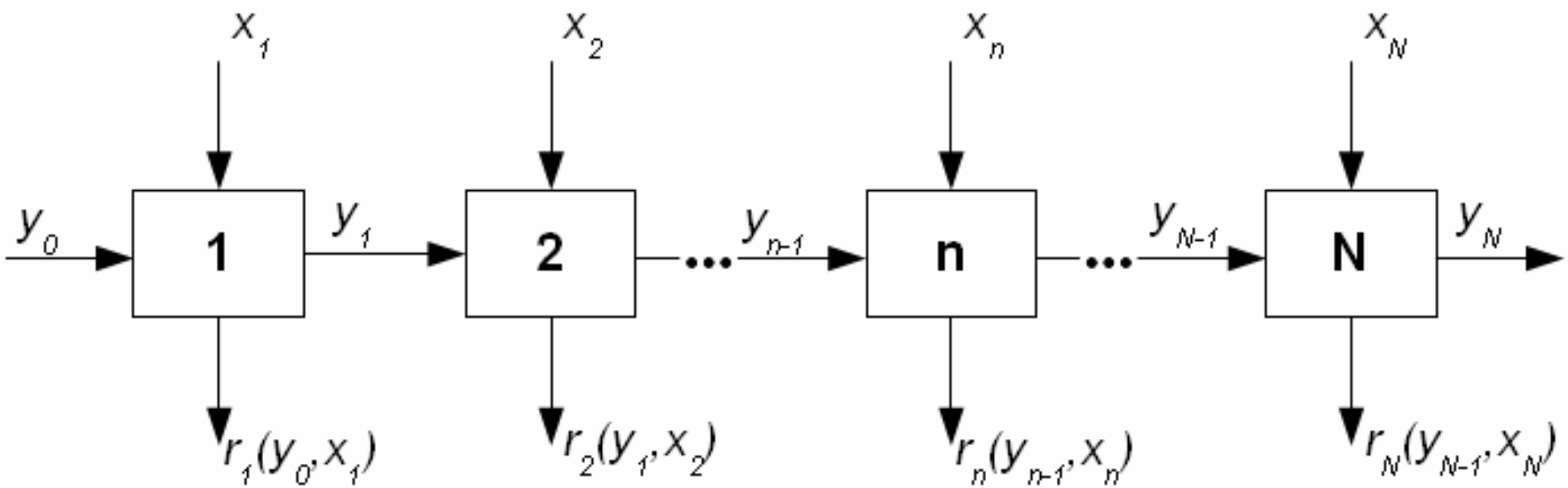


Figura 3 – Diagrama de Fluxo

Programação Dinâmica

As funções de decisão dos estágios $N, N-1, ..., 2, 1$ são respectivamente

$$f_1 = r_N(y_{N-1}, x_N)$$

$$f_2 = r_{N-1}(y_{N-2}, x_{N-1}) + F_1(y_{N-1}); y_{N-1} = t_{N-1}(y_{N-2}, x_{N-1})$$

...

$$f_i = r_{N-(i-1)}(y_{N-i}, x_{N-(i-1)}) + F_{i-1}(y_{N-(i-1)}); y_{N-(i-1)} = t_{N-(i-1)}(y_{N-i}, x_{N-(i-1)})$$

...

$$f_N = r_1(y_0, x_1) + F_{N-1}(y_1)$$

em que, $y_1 = t_1(y_0, x_1)$ e F_j é o máximo de f_j, $j = 1,2,...,N$.

Programação Dinâmica

Maximizando a função de decisão de cada estágio em relação a sua variável de decisão e tratando o estado de entrada como um parâmetro, obtêm-se as soluções de estágio paramétricas

$$x_n = x_n(y_{n-1}) \quad n = N, N-1, \ldots, 2, 1$$

Programação Dinâmica

Caso de Otimização Discreta

Suponha um exemplo no qual r_1 e r_2 são definidas apenas para os intervalos $x_1 \leq 3$ e $x_2 \leq 6$:

x_1	0	1	2	3
$r_1(=8x_1)$	0	8	16	24

x_2	0	1	2	3	4	5	6
$r_2(=10x_2)$	0	10	20	30	40	50	60

Programação Dinâmica

Caso de Otimização Discreta

	$y_1 (=y_0-4x_1)$			
$y_0 \backslash x_1$	0	1	2	3
12	12	8	4	0

	$y_2 (=y_1-2x_2)$						
$y_1 \backslash x_2$	0	1	2	3	4	5	6
0	0						
4	4	2	0				
8	8	6	4	2	0		
12	12	10	8	6	4	2	0

Programação Dinâmica

Caso de Otimização Discreta

Stage2			$f_1=r_2(x_2)=10x_2$					$F_1(y_1)$	$x_2(y_1)$	$y_2(y_1)$
$y_0 \backslash x_1$	0	1	2	3	4	5	6			
0	**0**							0	0	0
4	0	10	**20**					20	2	0
8	0	10	20	30	**40**			40	4	0
12	0	10	20	30	40	50	**60**	60	6	0

Stage1	$f_2=r_1(x_1)+F_1(y_1)=8x_1+F_1(y_1)$						
$y_0 \backslash x_1$	0	1	2	3	$F_2(y_0)$	$x_1(y_0)$	$y_1(y_0)$
12	**0+60**	8+40	16+20	24+0	60	0	12

Programação Dinâmica

Referência

ROMERO, R. A. F. ; GOMIDE, F A C . **A Neural Network to Solve Discrete Dynamic Programming Problems**. Campinas-SP: Relatório Técnico no 16/92, DCA/FEE/UNICAMP, 1992 (Relatório Técnico)

Técnicas Justificadas Formalmente

Programação dinâmica

$$V^*(n_1, w_1, ..., n_k, w_k) = \max_i E\left[\text{Future payoff if agent takes action } i, \text{ then acts optimally for remaining pulls}\right]$$

$$V^*(n_1, w_1, ..., n_k, w_k) = \max_i \begin{pmatrix} \rho_i V^*(n_1, w_i, ..., n_i + 1, w_i + 1, ..., n_k, w_k) + \\ (1 - \rho_i) V^*(n_1, w_i, ..., n_i + 1, w_i, ..., n_k, w_k) \end{pmatrix}$$

Recompensa Atrasada

Processos de Decisão de Markov (MDP)

Um MDP consiste de:

- Um conjunto de estados *S*;
- Um conjunto de ações *A*;
- Uma função de recompensa $R : SxA \rightarrow \Re$;

Recompensa Atrasada

Processos de Decisão de Markov (MDP)

- Uma função de transferência de estado $T : SxA \rightarrow \prod(S)$, em que um membro de $\prod(S)$ é uma distribuição de probabilidade sobre o conjunto *S*. Dessa maneira, *T(s,a,s')* é a probabilidade de realizar a transição do estado *s* para o estado *s'* através da ação *a*.

Recompensa Atrasada

Definindo uma política de um modelo

$$V^*(s) = \max_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

$$V^*(s) = \max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')\right), \forall s \in S$$

(1)

RAFR

Recompensa Atrasada

Definindo uma política de um modelo

$$\pi^*(s) = \arg\max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right)$$

Algoritmo Value Iteration

Definindo uma política de um modelo

initialize V(s) arbitrarily
loop until policy good enough
  loop for $s \in S$
    loop for $a \in A$

$$Q(s,a) := R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s')$$

$$V(s) := \max_a Q(s,a)$$

  end loop
end loop

Complexidade: quadratica no no. de estados e linear no no. de ações

Recompensa Atrasada

Definindo uma política de um modelo baseada na eq. 1

$$Q(s,a) := Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

r é uma amostra com média R(S,a)
e variancia limitada e s´ é
amostrado da distr. T(s,a,s´)

RAFR

Algoritmo de Policy Iteration

Definindo uma política de um modelo

choose an arbitrary policy π'
loop

$\pi := \pi'$

compute the value function of policy π*:*

solve the linear equations

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V_\pi(s')$$

improve the policy at each state:

$$\pi'(s) := \arg\max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_\pi(s') \right)$$

until $\pi = \pi'$

RAFR

Métodos Livres de Modelos

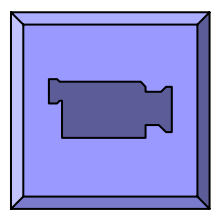
Q-Learning

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^*(s',a')$$

$$Q(s,a) := Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

Q-Learning [Watkins, 1989]

- No tempo *t, o agente*:
  - observa estado s_t e reforço R_t, seleciona ação a_t;
- No tempo *t+1, o agente*:
  - observa estado s_{t+1} ;
  - atualiza o **valor de ação** $Q_t(s_t, a_t)$ de acordo com
- $\Delta\, \mathbf{Q_t(s_t, a_t) = \alpha_t [R_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]}$



Características do Q-Learning

- ☺ **Convergência garantida**
- ☺ **Ações de treino podem ser escolhidas livremente**

- ☹ **Convergência extremamente lenta**

Q-Learning

- Exploração X Exploitação

O algoritmo pode ser tendioso se a escolha for sempre para a melhor ação (exploração). Então deve-se colocar uma taxa de exploitação (ainda que pequena) para que uma ação seja escolhida de forma aleatória.

R-learning [Schwartz, 1993]

- Semelhante ao Q-learning, mas:
  - Maximiza a recompensa média para cada passo.
  - Não utiliza descontos (γ).
- Atualiza o **valor de ação** $R_t(s_t, a_t)$ de acordo com

$$\Delta R_t(s_t, a_t) = \alpha_t [r_t - \rho + max_a R_t(s_{t+1}, a) - R_t(s_t, a_t)]$$

- ρ só é atualizado quando a não for aleatório:

$$\rho_t = \rho_t + \beta [r_t + max_a R_t(s_{t+1}, a) - max_a R_t(s_t, a) - \rho_t]$$

Referências

Kaelbling, L. P.; Littman, M. L. ; Moore, A. W. (1996). Reinforcement Learning: A Survey. **Journal of Artificial Intelligence Research**, vol. 4, pg. 237 – 285.