RL aula 2

Markov Decision Processes and Solutions

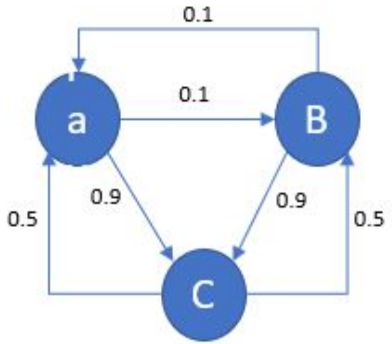
Plano de hoje

1. De Processos de Markov a Processos de Decisão de Markov
2. Equações de Bellman
3. Solução por DP

De Processos de Markov a Processos de Decisão de Markov

- Processos de Markov
- Processos de Recompensa de Markov
  - Solução
- Processos de Decisão de Markov

Processos de Markov



$$P = \begin{bmatrix} 0 & 0.1 & 0.9 \\ 0.1 & 0.0 & 0.9 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

- Processos aleatórios sem memória
- São uma tupla <S, P>
    - S são os estados (finito)
    - P é uma matriz de probabilidades de transição
- Pode ser visto como uma máquina de estados probabilística
- Assume a propriedade Markoviana (se não a matriz de transição não faria sentido)

Processos de Recompensa de Markov

- Uma tupla <S, P, R, γ>
  - S são os estados (finito)
  - P é uma matriz de probabilidades de transição
  - R é uma função de Recompensa
  - $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \mid S_t = s\right]$
  - γ é um fator de desconto
- Retorno
  - A recompensa total descontada a partir do tempo t
  - $$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$
- Efeitos do desconto
  - gamma pequeno -> agente míope
  - gamma grande -> agente focado no futuro

Função de Valor

- Valor esperado do retorno partindo do estado s
- A recompensa a longo prazo de um estado s
- Resolver um MRP é achar v

$$v(s) = \mathbb{E}\left[G_t \mid S_t = s\right]$$

Bellman Equation

- Podemos quebrar a função de valor para encontrar uma formulação recursiva:

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \dots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}
$$

Bellman Equation Versão Matricial

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Resolvendo a Equação de Bellman

- Podemos resolver diretamente:

$$v = (I - \gamma \mathcal{P})^{-1} \mathcal{R}$$

- Solução cúbica no número de estados, pesada para $|S| > 10^5$, impossível para coisas como Go
- Métodos iterativos:
  - DP (essa aula)
  - Monte-Carlo Evaluation (Próxima aula)
  - Temporal-Difference Learning (Próxima aula)

Processo de Decisão de Markov

- MRP com livre arbítrio (ações)
- Uma tupla <S, A, P, R, γ>
  - S é o conjunto de estados (finito)
  - A é o conjunto de ações
  - P é um tensor de probabilidades de transição

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$$

  - R é uma função de Recompensa
  - $\mathcal{R}_{s}^{a} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
  - γ é um fator de desconto

Policy

- Mapa de estados para ações ou distribuição de ações

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

- Define completamente as ações de um agente, por isso muitas vezes se usa os termos de maneira permutável
- Dependem só do estado e não do tempo
- Dado um MDP e uma policy podemos reduzir o MDP a um MRP (uma vez que a policy determina as ações)
- Resolver um MDP é encontrar **π***, a policy ótima

Funções de Valor

- Temos novamente a v, porém agora dependendo da policy

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$

- Temos agora q, a função de valor de ação, que é o retorno esperado de tomar a ação **a** no estado **s** e a partir daí seguir a policy **π**

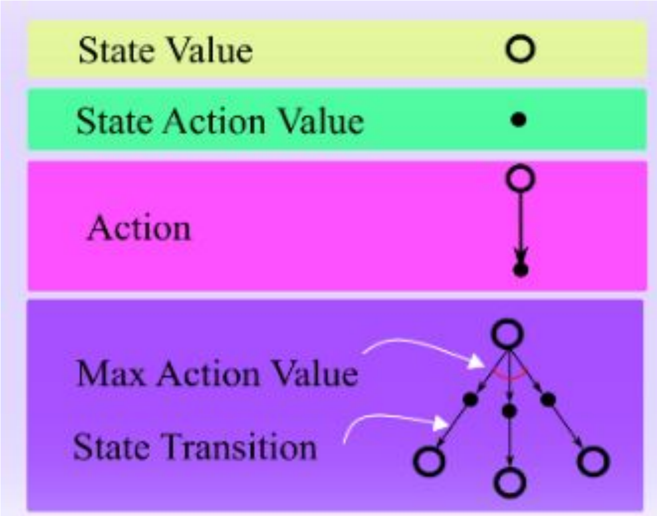
$$q_\pi(s, a) = \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right]$$

- Policies podem ser ordenadas de maneira fraca, uma policy **π'** é melhor ou igual que **π** se para todo estado $\mathbf{V_{\pi'}(s) \geq V_{\pi}(s)}$

Prediction vs Control

- **Prediction**:
  - Descobrir o quão boa é uma policy
  - Encontrar **v** e/ou **q** dada **π**
  - Resolver o MRP induzido pelo seu MDP e **π**
- **Control:**
  - Otimizar a policy
  - Resolver o MDP

Diagramas de Backup



Bellman Expectation Equation

Usada para prediction

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Funções de Valor Ótimas

- A função de valor de estado ótima v*

$$v_*(s) = \max_{\pi} v_\pi(s)$$

- A função de valor de estado ótima q*

$$q_*(s, a) = \max_{\pi} q_\pi(s, a)$$

- Especificam a melhor performance possível no MDP

Policies ótimas

- Para toda policy ótima **π*** e policy **π** temos **π* ≥ π**
- Toda policy ótima tem gera **v*** e **q***
- Para MDPs sempre existe uma policy ótima determinística
- Se sabemos q*:

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, q_*(s,a) \\ 0 & \textit{otherwise} \end{cases}$$

Bellman Optimality Equation

$$v_*(s) = \max_a q_*(s, a)$$

$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

Resolvendo a BOE

- Não é linear :(
- Em geral não tem solução fechada
- Soluções iterativas
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - Sarsa

Solução Baseada em DP

- Para resolver um problema por DP precisamos de duas condições:
  - Optimal Substructure: Garantida pela equação de Bellman
  - Overlapping Subproblems: Funções de valor
- É model-based: você precisa conhecer todo o MDP
- Pode ser usada para prediction e control

Iterative Policy Evaluation

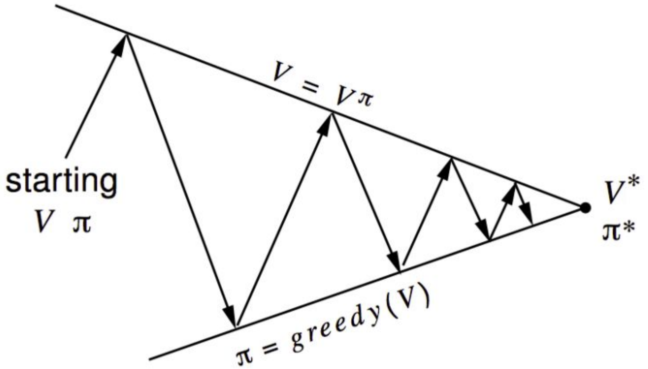
- Prediction: Saber o quão boa é uma policy
- Usa a Bellman Expectation Equation
- Para cada estado s:
  - Atualize $v^{k+1}(s)$ usando $v^{k}(s')$, onde s' é um sucessor de s

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

$$\mathbf{v}^{k+1} = \mathcal{R}^{\boldsymbol{\pi}} + \gamma \mathcal{P}^{\boldsymbol{\pi}} \mathbf{v}^k$$

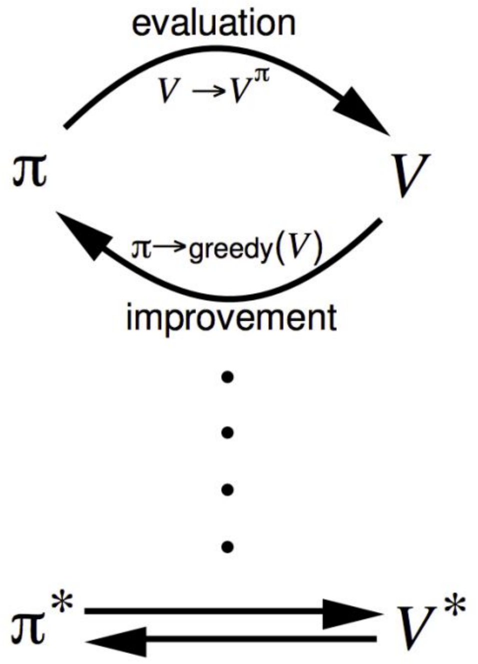
Policy Iteration

- Até convergência
  - Evaluate the policy: calcular **v** usando IPE
  - Melhore ela agindo **gulosamente** em relação a **v**
- Sempre converge para **π***



Generalised Policy Iteration

- Até convergência
  - Evaluate the policy: calcular **v** usando qualquer método de policy evaluation
  - Melhore ela usando qualquer método de policy improvement



Value Iteration

- Se temos os valor de **v*(s')**, podemos encontrar **v*(s)**:

$$v_*(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

- Basicamente Policy Evaluation, mas usamos a Equação de Otimalidade de Bellman
- Converge para v*
- Não tem uma policy explícita

Exemplo: Menor caminho

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

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

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

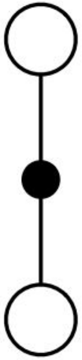
Resumo dos Algoritmos

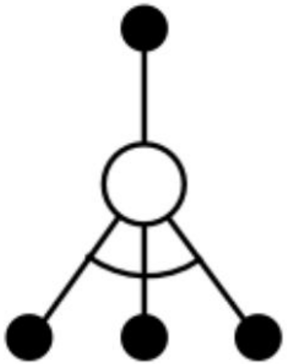
Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

- $O(AS^2)$ por iteração
- Podemos usar os mesmos algoritmos para computar **q**, ficando com $O(A^2S^2)$

Extensões e fraquezas

- Extensões
  - Updates assíncronos
  - Prioritized sweeping
- Fraquezas:
  - Fazemos um full-width backup toda vez, computacionalmente caro





Recursos úteis

- Markov Chains Visually Explained
- Lilian Weng's A (Long) Peek into Reinforcement Learning
- Backup Diagrams
- Stack Overflow: Policy Iteration vs Value Iteration
- Gifs meus