Inteligência Artificial para Robótica Móvel

Introdução ao Aprendizado por Reforço

Professor: Marcos Maximo

Roteiro

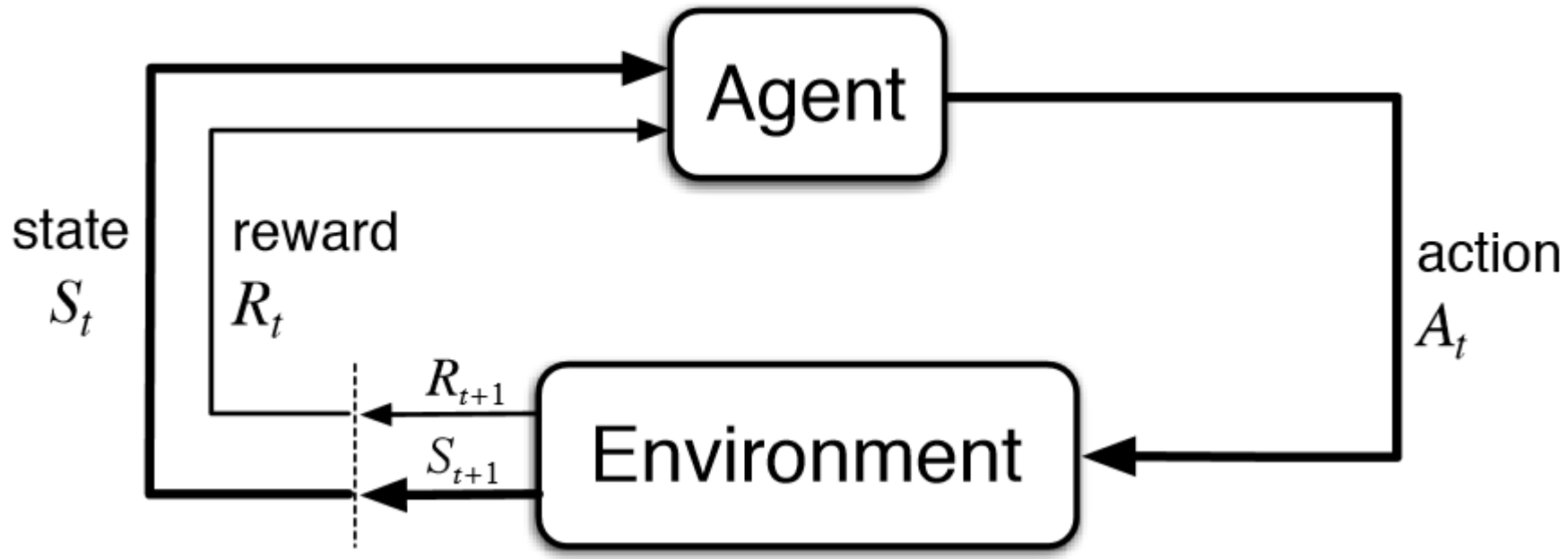
- Motivação;
- Revisão de Probabilidade 2;
- Problema de RL;
- Agente de RL;
- Equação de Bellman;
- Programação Dinâmica.

Motivação

Aprendizado por Reforço

- Inglês: *Reinforcement Learning* (RL).
- Ao contrário de Aprendizado Supervisionado, não há dados anotados.
- Há apenas um sinal de recompensa (bom ou ruim).
- Aprendizado difícil. Antigamente só resolvia problemas brinquedo…
- Também se beneficiou de *Deep Learning*.
- Muito interessante para tarefas de controle (forte paralelo).
- Agente observa o mundo e toma ações.
- *Artificial General Intelligence*.

Aprendizado por Reforço

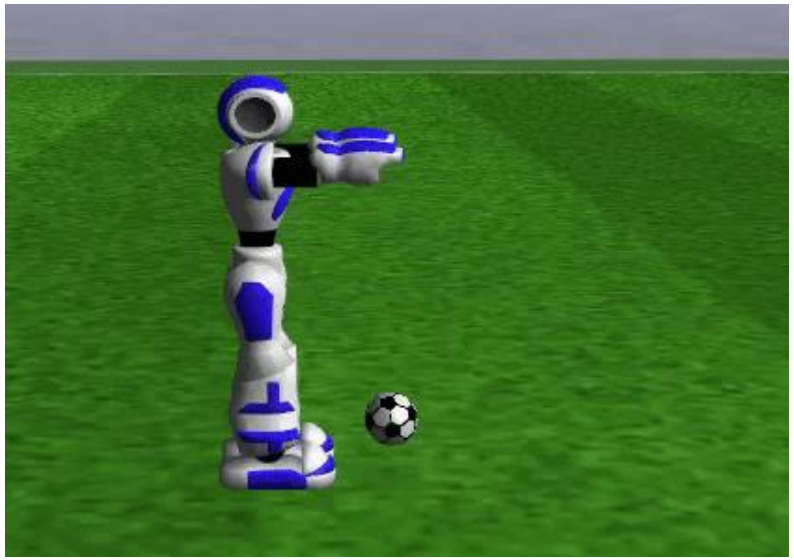


Exemplos de Aprendizado por Reforço

- Fazer um robô humanoide andar.
- Fazer um robô humanoide chutar a bola o mais distante possível.
- Driblar um robô adversário enquanto conduz a bola.
- Jogar jogos de Atari com habilidade superhumana.
- Derrotar o melhor jogador de Go do mundo.
- Derrotar o melhor time do mundo de Dota 2.
- Derrotar jogadores profissionais de Starcraft 2.

Chute da Bola (Melo et al, 2018)

- Paralelização em *cluster* Intel DevCloud: mais de 100 agentes aprendendo simultaneamente.
- Obtenção de experiência equivalente a 6 meses do robô chutando a bola.



Trained Dribbling Agent with PPO

Deep Mind

Atari (NIPS 2013)

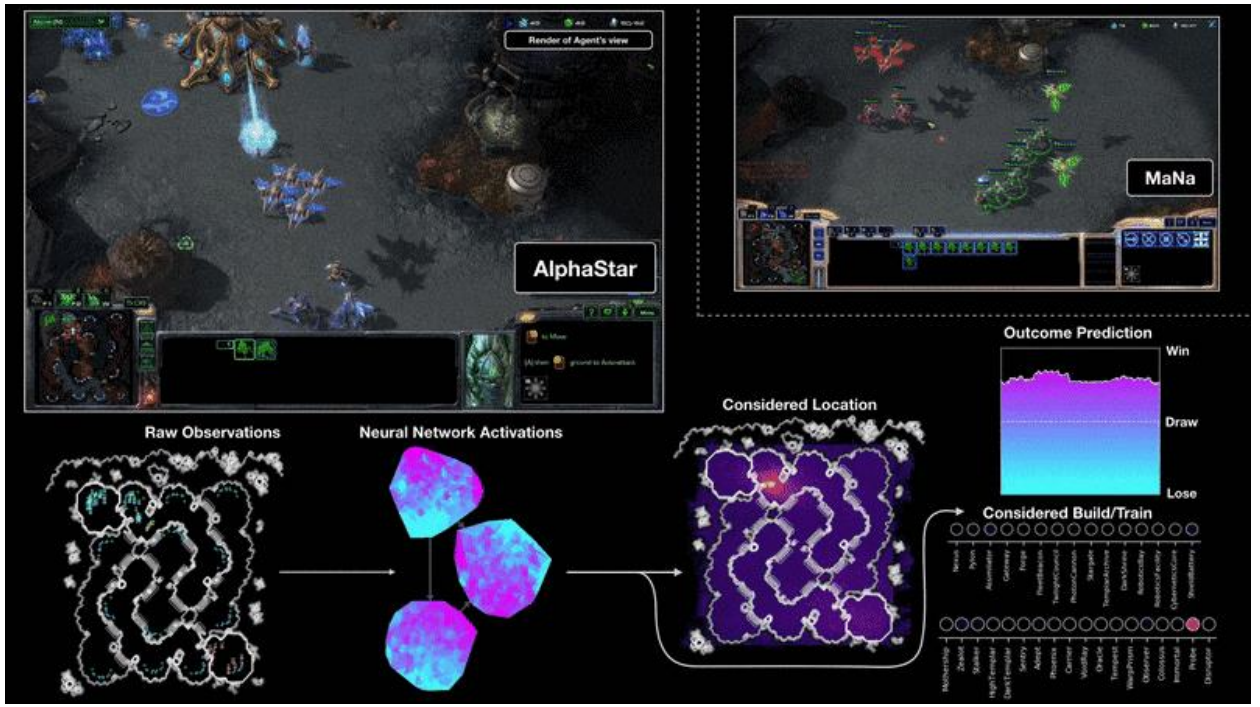
AlphaGo (2016)

OpenAI

- OpenAI Five: venceu melhor time de humanos no Dota 2 (13 de abril).
- 180 anos de treino por dia. 128k núcleos de CPUs e 256 GPUs.

Starcraft 2

- Venceu jogadores profissionais em Starcraft 2.
- Por enquanto, joga apenas com Protoss.



Controle e RL

- Quem já estudou controle, percebeu que a descrição de problema RL é muito semelhante a de problema controle.
- Agente: controlador.
- Ambiente: planta.
- Estado: estado.
- Ação: esforço de controle.
- Recompensa: medida de qualidade.

Controle e RL

- Ambas as áreas lidam com tomada de decisão, logo isso é esperado.
- RL aproveitou muito da Literatura de Controle.
- Em geral, Controle está mais preocupado em resolver o problema quando se tem o modelo do ambiente.
- RL está mais preocupado em resolver quando **não** se tem o modelo.
- Na prática, as técnicas de controle não funcionam bem para modelos complicados (muito não-lineares).
- Controle tem garantias matemáticas interessantes (comportamento conhecido, robustez etc.).

Controle e RL

- Controle é usado na indústria há muitos anos (funciona no mundo real).
- RL ainda é “brinquedo” (pesquisa).
- Difícil fazer funcionar no mundo real.
- RL tem mais promessa.
- Em resumo (situação atual):
  - Se tem modelo e é linear (ou facilmente linearizável), use controle.
  - Se não tem modelo ou é **muito** complexo, use RL.

Revisão de Probabilidade 2

Probabilidade Conjunta e Condicional

- Pode-se ter mais de uma variável aleatória.
- Probabilidade conjunta: $P(x, y) = P(X = x\ e\ Y = y)$.
- X e Y **independentes**: $P(x, y) = P(x)P(y)$.
- Probabilidade condicional: $P(x|y) = \frac{P(x,y)}{P(y)}$.
- X e Y independentes: $P(x|y) = P(x)$.

Independentes x Mutuamente Exclusivos

- Independentes:

$$P(x, y) = P(x)P(y)$$
$$P(x \cup y) = P(x) + P(y) - P(x)P(y)$$

- Mutuamente exclusivos:

$$P(x, y) = 0$$
$$P(x \cup y) = P(x) + P(y)$$

Esperança

- Resultado médio se repetir o experimento infinitas vezes.
- Matematicamente:

$$E[X] = \mu_x = \sum_x xP(x)$$

$$E[X] = \mu_x = \int_x xp(x)dx$$

- Propriedades:

$$E[X + Y] = E[X] + E[Y]$$
$$E[\alpha X] = \alpha E[X], E[cte] = cte$$

Problema de RL

Recompensas

- Uma recompensa R_t é um sinal de retroalimentação (*feedback*) escalar.
- Indica quão bem um agente está indo no tempo de amostragem t.
- O objetivo do agente é maximizar a recompensa acumulada.
- Hipótese da recompensa: todos os objetivos podem ser descritos como a maximização de recompensa acumulada esperada.

Exemplos de Recompensa

- Fazer um robô humanoide andar:
  - +1 por andar para frente.
  - -10 por cair.
- Derrotar o melhor time do mundo de Dota 2:
  - +1 se derrotar herói do time adversário.
  - -1 se perder herói do nosso time.
  - +20 se ganhar a partida.
  - -20 se perder a partida.

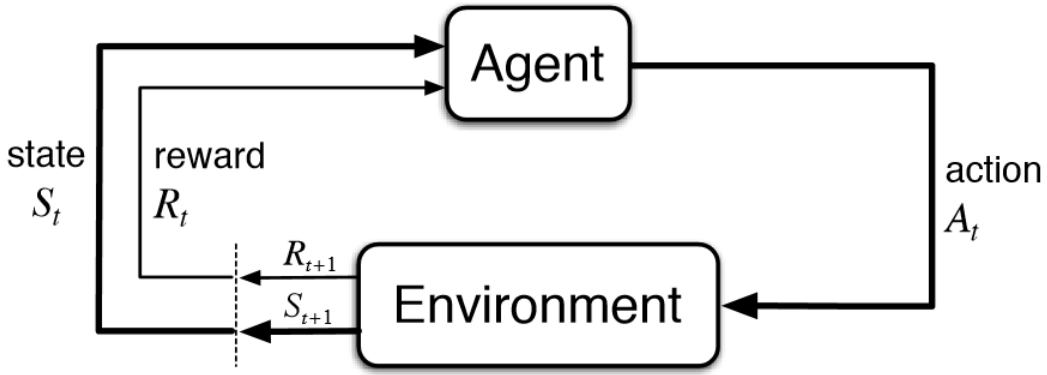
Recompensas Atrasadas

- Um grande desafio em RL é que às vezes a recompensa para uma ação é muito atrasada.
- Exemplo: uma ação que você toma no começo do jogo de Starcraft tem impacto no resultado final do jogo.
- A relação entre causa e efeito é atrasada nesse caso.
- Seres humanos são muito bons em fazer essa ligação, mas nossos algoritmos de RL ainda não são.
- É comum adicionar “recompensas intermediárias” para facilitar o aprendizado.

Tomada de Decisão Sequencial

- Objetivo: decidir ações que maximizem a recompensa acumulada futura.
- Ações podem ter consequências de longo prazo.
- Recompensa pode ser atrasada.
- Pode ser interessante sacrificar recompensa imediata para ganhar mais recompensa no longo prazo.
- Exemplo: para driblar oponente no futebol, às vezes necessário passar muito perto dele.

Agente e Ambiente



- Estado: representação "completa" do mundo.
- Em cada passo t, o agente:
  - Recebe estado S_t.
  - Recebe recompensa R_t.
  - Executa ação A_t.
- O ambiente:
  - Recebe ação A_t.
  - Emite novo estado S_{t+1}.
  - Emite nova recompensa R_{t+1}.

Histórico (Trajetória)

- Isso gera um histórico (trajetória):

$$H_t = \tau_t = S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$$

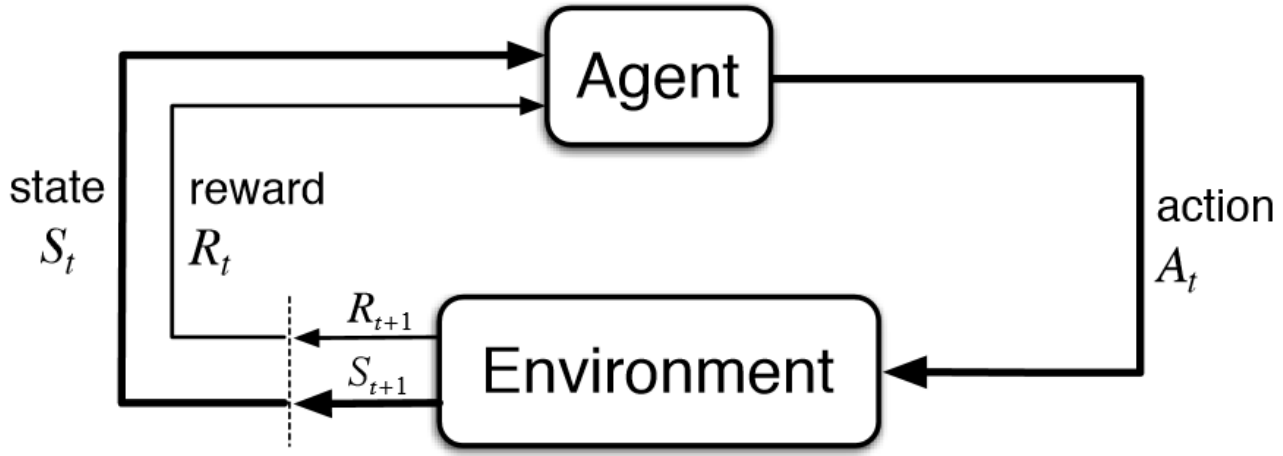
Dinâmica do Ambiente

- A função de dinâmica do ambiente é dada por:

$$p(s'|s,a) = P(S_{t+1} = s'|S_t = s, A_t = a)$$

- Alguns autores (e.g. Sutton) preferem incluir a recompensa junto:

$$p(s',r|s,a) = P(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$$



Ambiente Markoviano

- Diz-se que um ambiente é Markov se respeitar a propriedade de Markov:

$$P(S_{t+1}|S_t, A_t, S_{t-1}, A_{t-1}, \dots, S_0, A_0) = P(S_{t+1}|S_t, A_t)$$

- Isto é, o estado S_t caracteriza completamente a dinâmica (dinâmica não depende do passado).
- Em geral, é uma boa aproximação do mundo real…
- Na prática, se depende de estados anteriores, basta definir estado "expandido".
- Exemplo: se depende de S_t e S_{t-1}, definir $S'_t = [S_t\ S_{t-1}]^T$.

Observação x Estado

- Na verdade, muitas vezes, o agente apenas observa “observações”, e não estado.
- Quando não observa tudo, tem-se “observabilidade parcial”.
- Em geral, ação ótima precisa de conhecimento de todo o estado.
- Às vezes, parte do estado/observação não importa para decisão: posição da lua para tomar decisão para onde chutar a bola.

Observação x Estado

- Muitas vezes, é possível reconstruir o estado a partir do histórico de observações.
- Exemplo:
  - Observando apenas posição, pode-se calcular velocidade a partir de duas posições consecutivas.
  - Observando histórico de linhas, traves, cruzes, círculo central etc., robô pode encontrar sua posição no campo de futebol.
- Obter estado a partir de observação: estimação de estados (Processamento de Sinais).
- No curso, vamos considerar estado = observação.

Tarefas Episódicas x Tarefas Continuadas

- Algumas tarefas são naturalmente quebradas em “episódios”, i.e. a tarefa começa e termina.
- Outras tarefas podem continuar indefinidamente.
- Exemplo de tarefa episódica: robô chutar a bola.
- Exemplo de tarefa continuada: manter robô em pé o máximo de tempo possível.

Retorno

- O agente busca maximizar o acúmulo de compensa futura. Isto é formalizado através do conceito de retorno:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T$$

- No caso de tarefas continuadas, tem-se $T = \infty$, logo G_t pode ser infinito. Assim, é comum usar o conceito de desconto (γ):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

- $\gamma \in [0,1]$ implementa um conceito "humano" interessante: recompensas imediatas valem mais que recompensas futuras.
- Também pode usar em tarefas episódicas:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$$

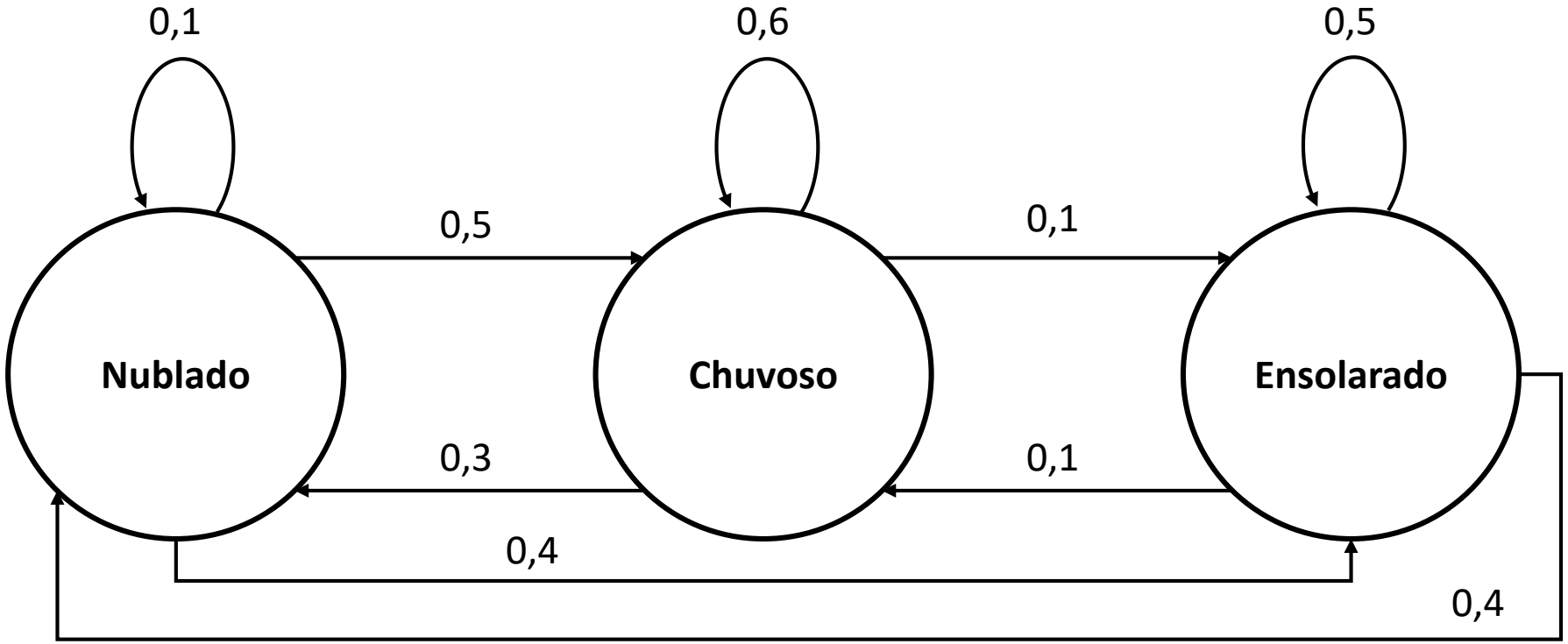
Processo de Markov (MP)

Um processo de Markov (cadeia de Markov) é a tupla (S, p):

- S é um conjunto (finito) de estados.
- $p(s'|s) = P(S_{t+1} = s'|S_t = s)$ é a matriz de probabilidade de transição.

Exemplo de Processo de Markov

- Mudança de clima de um dia para o outro.



Exemplo de Processo de Markov

No exemplo do *slide* anterior:

- Estados: $\{Nublado, Chuvoso, Ensolarado\}$.
- t é de um dia.

$$\mathbf{P} = \begin{bmatrix} 0{,}1 & 0{,}3 & 0{,}4 \\ 0{,}5 & 0{,}6 & 0{,}1 \\ 0{,}4 & 0{,}1 & 0{,}5 \end{bmatrix}$$

- Perceba que se somarmos os elementos de uma mesma coluna, o resultado dá 1. Por que?

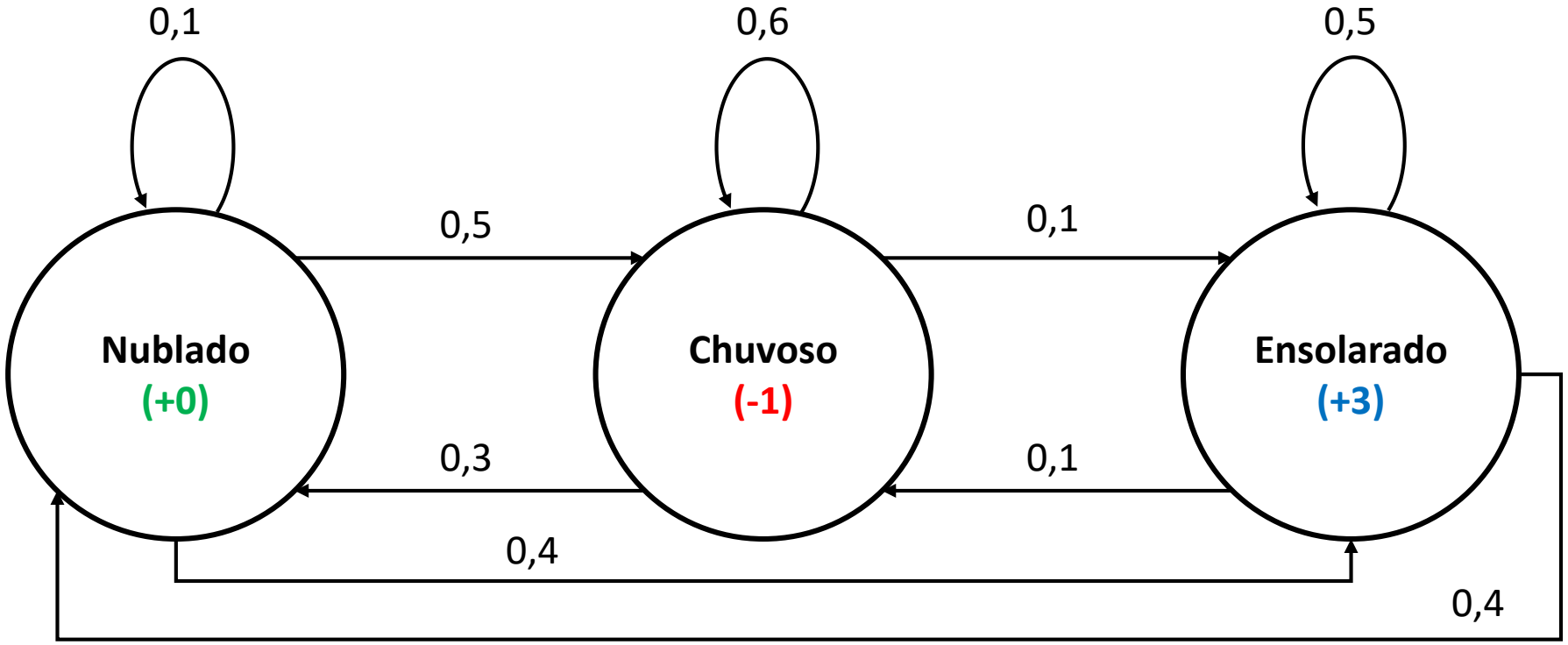
Processo de Markov com Recompensa (MRP)

Um processo de Markov com recompensa é a tripla (S, p, r, γ):

- S é um conjunto (finito) de estados.
- $p(s'|s) = P(S_{t+1} = s'|S_t = s)$ é a matriz de probabilidade de transição.
- $r(s, s') = E[R_{t+1}|S_t = s, S_{t+1} = s']$ é a função de recompensa.
- γ é um fator de desconto, $\gamma \in [0,1]$.

Exemplo de MRP

- **Nesse exemplo**, considerar recompensa quando “sai” do estado.

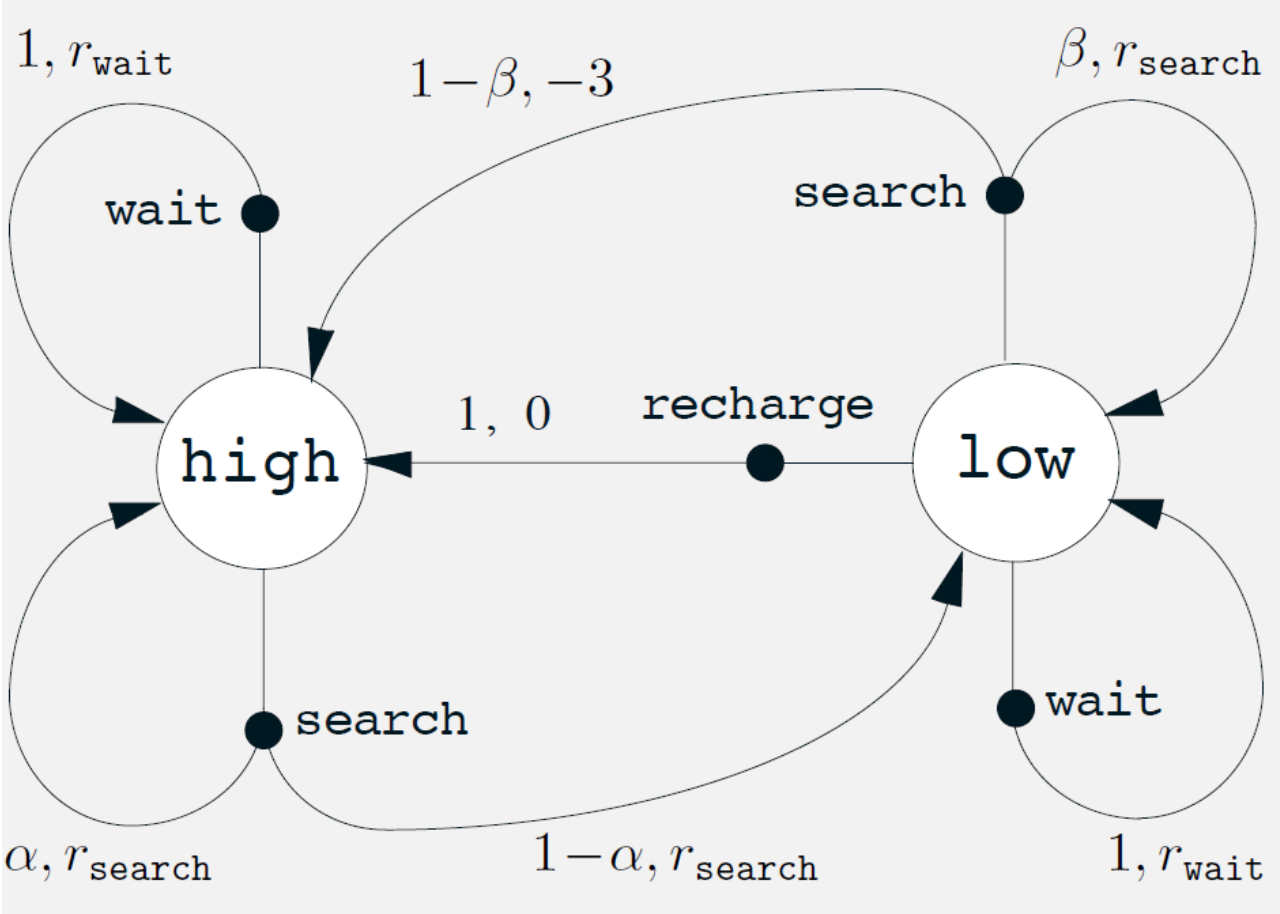


Processo Decisório de Markov (MDP)

Um Processo Decisório de Markov (MDP) é dado pela 4-tupla (S, A, p, r, γ):

- S é um conjunto (finito) de estados.
- A é um conjunto (finito) de ações.
- $p(s'|s, a) = P(S_{t+1} = s'|S_t = s, A_t = a)$ é a função de probabilidade de transição.
- $r(s, a, s') = E[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s']$ é a função de recompensa.

Exemplo de MDP (Robô Limpador)



Fonte: Sutton & Barto, Reinforcement Learning: An Introduction. The MIT Press.

Exemplo de MDP (Robô Limpador)

- $S = \{high, low\}$.
- $A(high) = \{search, wait\}$.
- $A(low) = \{search, wait, recharge\}$.

Exemplo de MDP (Robô Limpador)

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1 - \alpha$	$r_{\texttt{search}}$
low	search	high	$1 - \beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-

Fonte: Sutton & Barto, Reinforcement Learning: An Introduction. The MIT Press.

Agente de RL

Agente de RL

Pode possuir um ou mais dos seguintes componentes:

- Política: função de comportamento.
- Função valor: mede quão bom é certo estado (ou par estado-ação).
- Modelo: representação da dinâmica do ambiente.

Política

- Comportamento do agente.
- Qual ação o agente escolhe em cada estado.
- Matematicamente: $a = \pi(s)$.
- Política estocástica: $\pi(a|s) = P(A_t = a|S_t = s)$.

Função Valor

- Retorno esperado a partir de certo estado.
- Só faz sentido se associada a uma política.
- Avalia um determinado estado (começa em s e segue π):

$$v_\pi(s) = E_\pi[G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s]$$

- Função ação-valor (começa em s, executa a e depois segue π):

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$$

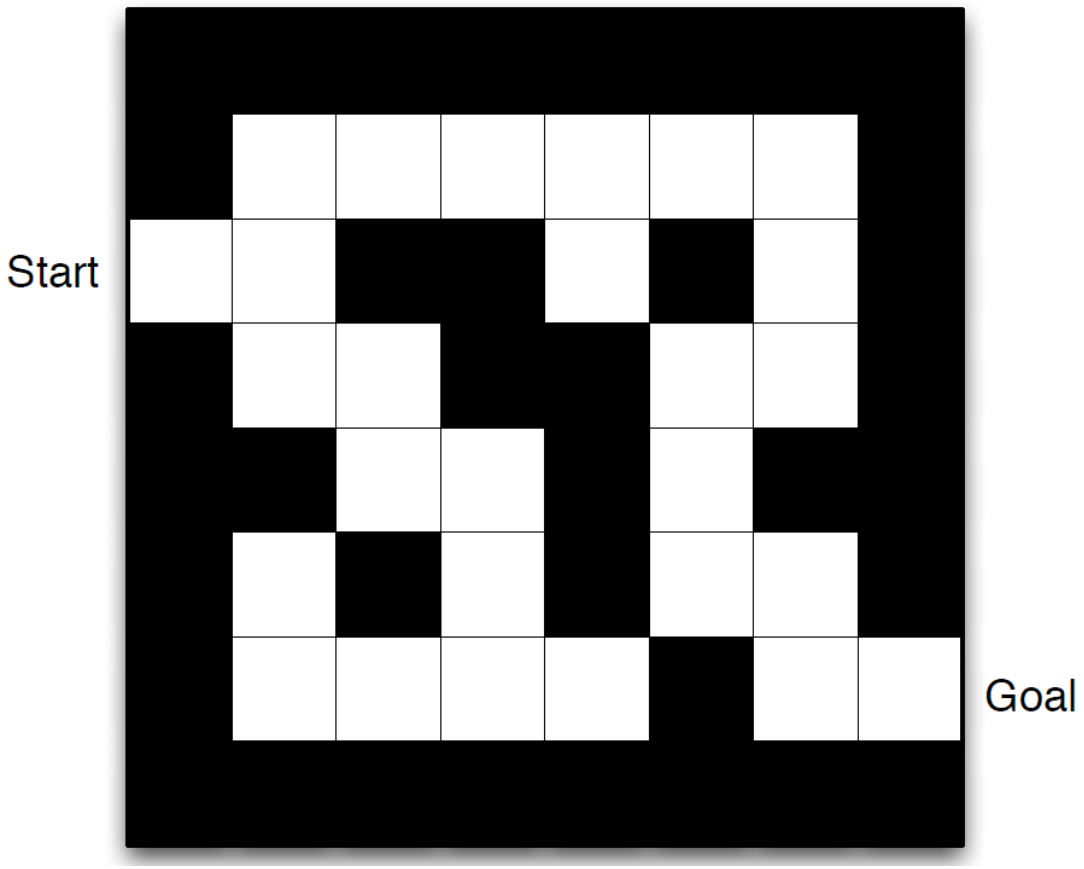
- Perceba que ao impor π para um MDP, obtém-se em um MRP.

Modelo

- Agentes podem possuir modelo (fornecido pelo projetista) ou construir com base em experiência.

$$p(s'|s,a) = P(S_{t+1} = s'|S_t = s, A_t = a)$$
$$r(s,a,s') = E[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s')$$

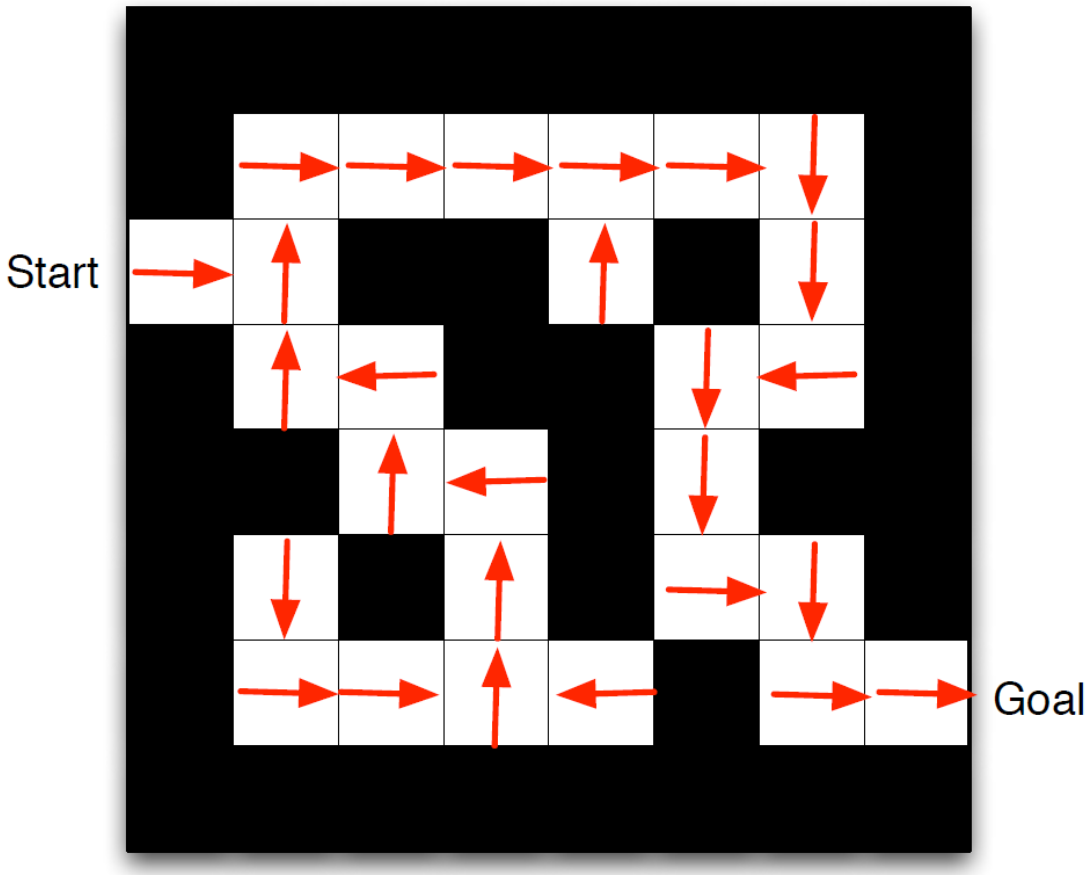
Exemplo do Labirinto



- Recompensa: -1 por tempo de amostragem.
- Ações: N, E, S, W (4-conectado).
- Estados: posições do agente dentro do labirinto.

Fonte: Lecture 1: Introduction to RL. UCL Course on RL. David Silver.

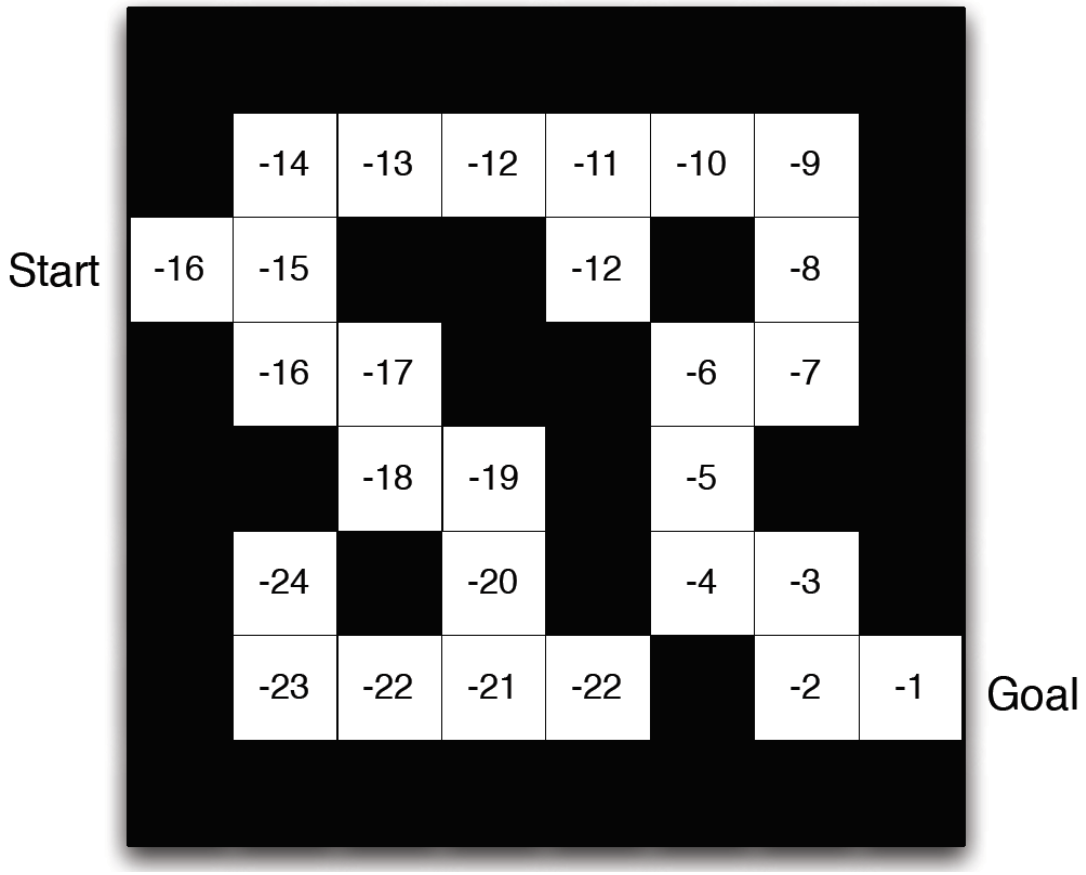
Exemplo do Labirinto – Política



- Seta representa ação em cada estado, i.e. $\pi(s)$.
- Política determinística.

Fonte: Lecture 1: Introduction to RL. UCL Course on RL. David Silver.

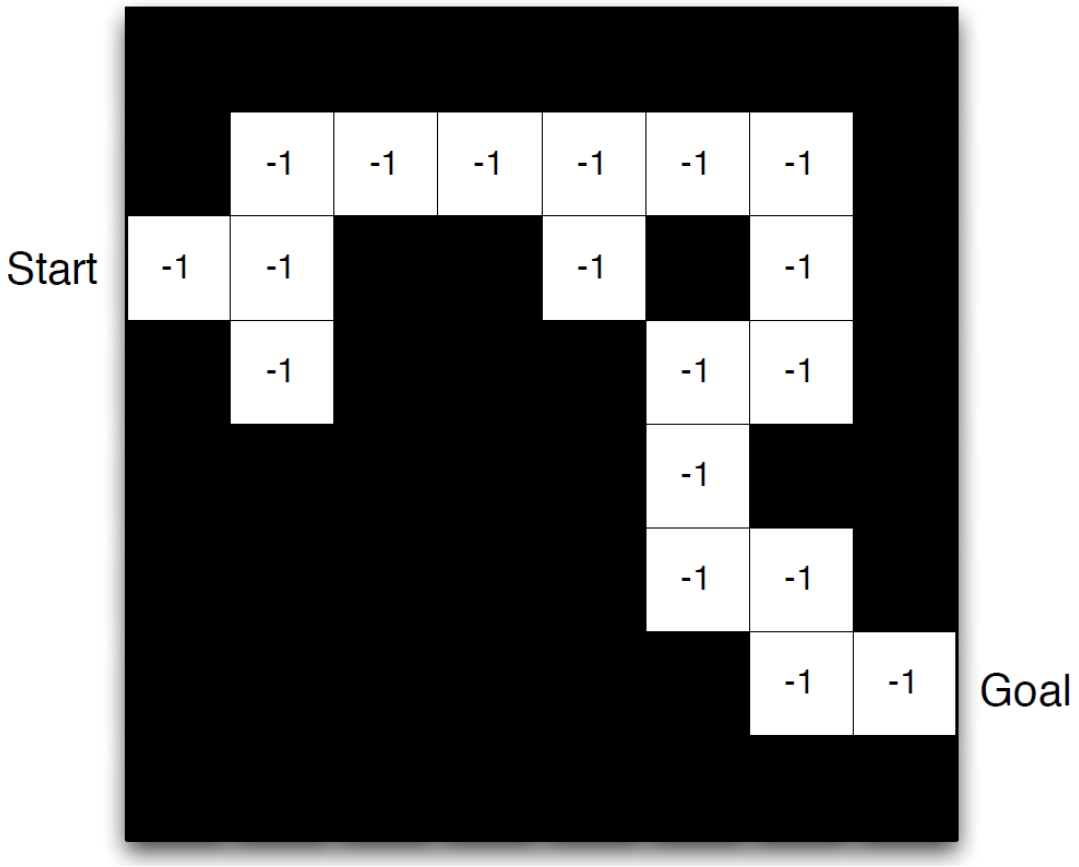
Exemplo do Labirinto – Função Valor



- Números representam função valor em cada estado, i.e. $v_\pi(s)$.

Fonte: Lecture 1: Introduction to RL. UCL Course on RL. David Silver.

Exemplo do Labirinto – Modelo



- Estrutura do *grid* contém transições.
- Números representam recompensa em cada estado, i.e. $r(s, a, s') = -1$.
- Modelo imperfeito.
- Agente tem que refinar modelo com base em experiência.

Fonte: Lecture 1: Introduction to RL. UCL Course on RL. David Silver.

Categorias de Agentes

- Baseados em Valor: possui apenas função valor.
- Baseados em Política: possui apenas política.
- *Actor-Critic*: possui política e função valor.

Categorias de Agentes

- Baseado em modelo (*model-based*): usa modelo para melhorar política.
- Livre de modelo (*model-free*): aprende política diretamente da experiência, sem ter ou construir modelo.

RL x Planejamento

- Duas formas de tomar decisão:
  - Planejamento: visto na Aula 2, usando modelo do ambiente, a partir do estado atual, realiza busca para encontrar a melhor ação.
  - RL: agente interage com o ambiente (ou com um modelo dele) e constrói política, que armazena ação para cada estado.
- No fundo, RL troca a "deliberação" do planejamento por um *cache* (tabelão) que contém a ação para cada situação possível.
- RL "pega o macaco".
- É possível juntar as duas abordagens (AlphaGo).

Exploration x *Explotation*

- Mesmo sentido de antes: repetir o que já descobri que é bom ou tentar coisas novas?
- Para maximizar recompensa no curto prazo, faz sentido repetir o que é bom.
- Para aprender, é importante tentar coisas novas.
- Importante balançear.

Predição x Controle

- Predição: prever função valor para uma dada política.
- Controle: determinar política ótima.

Equação de Bellman

Equação de Bellman de Expectativa

- Função (estado-)valor pode ser decomposta em recompensa imediata e valor do próximo estado:

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi[R_{t+1} + \gamma R_{t+2} + \cdots |S_t = s]$$
$$= E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$

- Mesma coisa pode ser feita com função valor ação-valor:

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a]$$
$$= E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

Equação de Bellman de Expectativa

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$

$$= r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_\pi(s')$$

$$\therefore v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_\pi(s') \right)$$

Equação de Bellman de Expectativa

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) r(s,a) + \gamma \sum_{a \in A} \sum_{s' \in S} \pi(a|s) p(s'|s,a) v_\pi(s')$$

$$v_\pi(s) - \gamma \sum_{a \in A} \sum_{s' \in S} \pi(a|s) p(s'|s,a) v_\pi(s') = \sum_{a \in A} \pi(a|s) r(s,a)$$

$$\mathbf{A}\mathbf{v}_\pi = \mathbf{b}$$

$$\mathbf{A} = \mathbf{I} - \gamma \mathbf{P}^\pi_{ss'}, \mathbf{b} = \mathbf{r}^\pi_s$$

- Basta resolver o sistema linear para encontrar $\mathbf{v}_\pi$!
- Na prática, sistema fica muito grande.
- Solução do problema de predição.

Equação de Bellman de Expectativa

- Para função valor estado-ação, chega-se em resultado semelhante:

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) \sum_{a' \in A} \pi(a'|s') q_\pi(s',a')$$

- Novamente, para achar $q_\pi(s,a)$, basta resolver o sistema.

Política Ótima

- Ordenação parcial de políticas:

$$\pi' \geq \pi \; se \; v_{\pi'}(s) \geq v_{\pi}(s), \forall s$$

- Teorema: existe uma política ótima π_* tal que: $\pi_* \geq \pi, \forall \pi$.
- Todas as políticas ótimas atingem as funções valores ótimas:

$$v_{\pi_*}(s) = v_*(s)$$
$$q_{\pi_*}(s, a) = q_*(s, a)$$

Encontrando a Política Ótima

- Pode-se encontrar uma política ótima maximizando o $q_*(s,a)$:

$$\pi_*(a|s) = \begin{cases} 1, se\ a = \underset{a' \in A}{\operatorname{argmax}}\, q_*(s,a') \\ 0, caso\ contrário \end{cases}$$

- Embora guloso, é ótimo, pois $q_*(s,a')$ considera o futuro.
- Perceba que a política ótima é determinística.
- Também pode-se afirmar que:

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

$$q_*(s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_*(s')$$

Equação de Bellman de Otimalidade

- Juntando as equações do *slide* anterior:

$$v_*(s) = \max_a \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_*(s') \right)$$

$$q_*(s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) \max_{a' \in A} q_*(s',a')$$

- $\max$ torna a equação não-linear.
- Não dá para só resolver sistema linear.

Programação Dinâmica

Avaliação de Política Iterativa

- Voltando na equação de Bellman de expectativa:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) r(s,a) + \gamma \sum_{a \in A} \sum_{s' \in S} \pi(a|s) p(s'|s,a) v_\pi(s')$$

- Ideia:

$$v_{k+1}(s) = \sum_{a \in A} \pi(a|s) r(s,a) + \gamma \sum_{a \in A} \sum_{s' \in S} \pi(a|s) p(s'|s,a) v_k(s')$$

- Vetorizado:

$$\mathbf{v}_{k+1} = \mathbf{r}_s^\pi + \gamma \mathbf{P}_{ss'}^\pi \mathbf{v}_k$$

- Converge para $v_\pi(s)$.

Iteração de Política

- Iniciar com política e função valor aleatórias.
- Loop:
  - Avaliar a política usando avaliação de política iterativa.

$$\mathbf{v}_{k+1} = \mathbf{r}_s^{\pi} + \gamma \mathbf{P}_{ss'}^{\pi} \mathbf{v}_k$$

  - Melhorar a política agindo de forma gulosa em relação a $v_k(s)$:

$$\pi'(s) = greedy(v_\pi(s))$$

Observação: durante a avaliação, não precisa iterar até convergir.

Esse algoritmo converge para a política ótima.

Escolha da Ação Ótima

$$\pi'(s) = greedy(v_\pi(s))$$

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\, q_\pi(s,a)$$

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}} \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_\pi(s') \right)$$

Observação: nesse caso, $\pi'(a|s)$ é determinística e $\pi'(s)$ determina a ação tomada em cada estado s.

Iteração de Valor

- Pode-se iterar diretamente sobre a equação de Bellman:

$$v_{k+1}(s) = \max_{a \in A} \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_k(s') \right)$$

- Pode-se mostrar que iteração de valor converge para $v_*(s)$.
- Por fim, quando o algoritmo tiver convergido, encontra-se a política ótima:

$$\pi_*(s) = greedy(v_*(s))$$

Observações sobre Programação Dinâmica

- Pode usar valores de $v_{k+1}(s)$ já atualizados no cálculo de $v_{k+1}(s)$ (Gauss-Jacobi x Gauss-Seidel).
- Iteração de valor síncrona:

$$v_{new}(s) = \max_{a \in A} \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_{old}(s') \right)$$

$$v_{old} = v_{new}$$

- "In-place":

$$v(s) = \max_{a \in A} \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v(s') \right)$$

Observações sobre Programação Dinâmica

- Algoritmo iterativo, logo requer critério de parada:

$$\max_{s \in S} |v_{k+1}(s) - v_k(s)| < \varepsilon$$

- Interessante: método usado na Literatura de Controle Ótimo.
- Resolve problemas de predição e controle “na moral”.
- Convergência garantida com tabelão.
- Método escala mal: capaz apenas de resolver problema “brinquedo”.
- Métodos práticos precisam aproximar $v(s)$.

Para Saber Mais

- Curso do David Silver (aulas 1, 2 e 3): http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html.
- Capítulos 3 e 4 do livro: SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction, second edition*. The MIT Press, 2017.
- Open AI Five (Dota 2): https://openai.com/five/
- AlphaStar (Starcraft 2): https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/

Laboratório 11

Laboratório 11

- Implementação de programação dinâmica para resolver tabuleiro.
- Problema de predição e de controle com modelo.