Aprendizado de Máquina

Aula 13

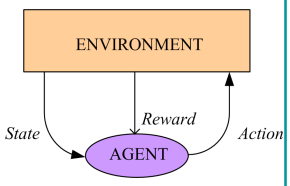
http://www.ic.uff.br/~bianca/aa/

Tópicos

1. Introdução – Cap. 1 (16/03)
2. Classificação Indutiva – Cap. 2 (23/03)
3. Árvores de Decisão – Cap. 3 (30/03)
4. Ensembles - Artigo (13/04)
5. Avaliação Experimental – Cap. 5 (20/04)
6. Aprendizado de Regras – Cap. 10 (27/04)
7. Redes Neurais – Cap. 4 (04/05)
8. Teoria do Aprendizado – Cap. 7 (11/05)
9. Máquinas de Vetor de Suporte – Artigo (18/05)
10. Aprendizado Bayesiano – Cap. 6 e novo cap. online (25/05)
11. Aprendizado Baseado em Instâncias – Cap. 8 (01/05)
12. Classificação de Textos – Artigo (08/06)
13. **Aprendizado por Reforço – Artigo (15/06)**

Introdução

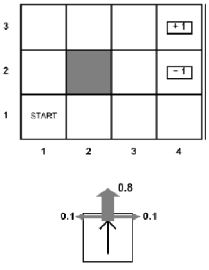
- **Jogos:** Aprender sequência de jogadas para ganhar o jogo
- **Robôs:** Aprender sequência de ações para chegar a um objetivo
- **Agente** está em um **estado** em um **ambiente**, realiza uma **ação**, recebe uma **recompensa** e os estado muda.
- Objetivo: aprender uma **política** que maximize as recompensas.



Processo de Decisão de Markov

- s_t : Estado do agente no instante t
- a_t: Ação tomada no instante t
- Em s_t, ação a_t é tomada. No instante seguinte a recompensa r_{t+1} é recebida e o estado muda para s_{t+1}
- Probabilidade do próximo estado: $P(s_{t+1} \mid s_t, a_t)$
- Distribuição de recompensas: $p(r_{t+1} \mid s_t, a_t)$
- Estado(s) inicial(is), estado(s) objetivos
- Episódios de ações do estado inicial ao estado final.
- (Sutton and Barto, 1998; Kaelbling et al., 1996)

Exemplo



- Agente tem probabilidade de 0.8 de se mover na direção desejada e 0.2 de se mover em ângulo reto.
- Os estados finais tem recompensa +1 e -1.
- Todos os outros estados tem recompensa -0.04.
- A medida de desempenho é a soma das recompensas.

Política e Recompensas Cumulativas

- Política, $\pi: S \rightarrow A \quad a_t = \pi(s_t)$
- Valor de uma política, $V^{\pi}(s_t)$
- Horizonte finito:

$$V^{\pi}(s_t) = E[r_{t+1} + r_{t+2} + \ldots + r_{t+T}] = E\left[\sum_{i=1}^{T} r_{t+i}\right]$$

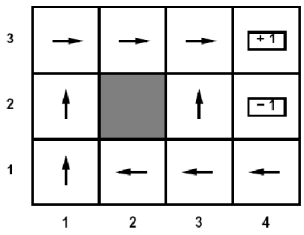
- Horizonte infinito:

$$V^{\pi}(s_t) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots] = E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right]$$

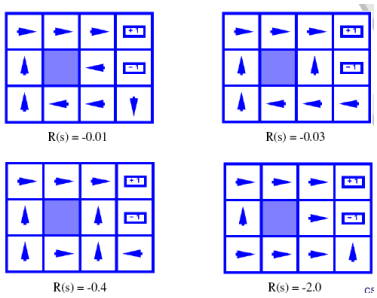
$0 \leq \gamma < 1$ é a taxa de desconto

Exemplo

- Política ótima quando os estados não-terminais tem recompensa $R(s) = -0.04$.



Exemplo



$$V^*(s_t) = \max_{\pi} V^{\pi}(s_t), \forall s_t$$

$$= \max_{a_t} E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right]$$

$$= \max_{a_t} E\left[r_{t+1} + \gamma \sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i+1}\right]$$

$$= \max_{a_t} E\left[r_{t+1} + \gamma V^*(s_{t+1})\right]$$

Equação de Bellman

$$V^*(s_t) = \max_{a_t}\left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})\right)$$

$$V^*(s_t) = \max_{a_t} Q^*(s_t, a_t) \quad \text{Value of } a_t \text{ in } s_t$$

$$Q^*(s_t, a_t) = E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$$

Modelo conhecido

- Comportamento do ambiente dado por $P(s_{t+1} \mid s_t, a_t)$, $p(r_{t+1} \mid s_t, a_t)$, é conhecido do agente.
- Não há necessidade de exploração.
- Podemos usar programação dinâmica.
- Resolver

$$V^*(s_t) = \max_{a_t}\left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})\right)$$

- Política ótima

$$\pi^*(s_t) = \arg\max_{a_t}\left(E[r_{t+1} \mid s_t, a_t] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})\right)$$

Iteração de Valor

- Calcular estimativas $V_i^*(s)$
  - Retorno esperado de se começar no estado s e agir de forma ótima por i passos.
  - Começamos com $i = 0$ e vamos aumentando o valor de i até a convergência (isto é, valores não mudam de i para $i + 1$).
    - A convergência é garantida com horizonte finito ou recompensas descontadas.

Iteração de Valor

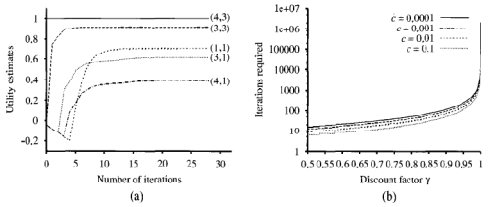
- Inicializar $V_0^*(s) = 0 \ \forall s$.
- Calcular $V_{i+1}^*(s)$ a partir de $V_i^*(s)$
- usando a equação:

$$V_{i+1}^*(s_t) = \max_{a_t}\left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V_i^*(s_{t+1})\right)$$

chamada de atualização de Bellman.

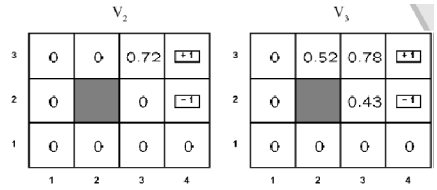
- Repetir o passo 2 até convergência isto é $V_{i+1}^*(s) \approx V_i^*(s) \ \forall s$

Exemplo: Iteração de Valor



Erro: $\epsilon = c \cdot R_{\max}$

Exemplo: Iteração de Valor



- A informação se propaga pra fora a partir dos estados terminais.

Modelo desconhecido

- O comportamento do ambiente dado por $P(s_{t+1} \mid s_t, a_t)$, $p(r_{t+1} \mid s_t, a_t)$ não é conhecido.
- É necessário **exploração**.
  - Executar ações para conhecer o ambiente e não tentando maximizar as recompensas.
- Usar a recompensa recebida no próximo passo para atualizar o valor do estado atual.

Estratégias de Exploração

- ε-gulosa: Com prob ε,escolher uma ação aleatória; e com probabilidade 1-ε escolher a melhor ação.
- Probabilística: $P(a \mid s) = \frac{\exp Q(s,a)}{\sum_{b=1}^{A} \exp Q(s,b)}$
- Ao longo do tempo diminuir a exploração:

$$P(a \mid s) = \frac{\exp\left[Q(s,a)/T\right]}{\sum_{b=1}^{A} \exp\left[Q(s,b)/T\right]}$$

Ações e Recompensas Determinísticas

$$Q^*(s_t,a_t) = E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t,a_t) \max_{a_{t+1}} Q^*(s_{t+1},a_{t+1})$$

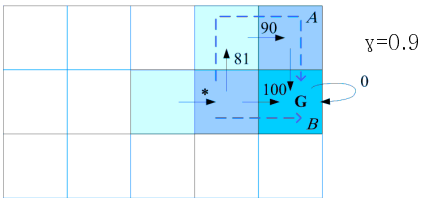
- Apenas uma recompensa e estado seguinte possíveis:

$$Q(s_t,a_t) = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1})$$

usar como regra de atualização:

$$\hat{Q}(s_t,a_t) \leftarrow r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1},a_{t+1})$$

Começando em zero, valores *Q* aumentam, nunca diminuem.



Considere o valor da ação marcada por um '*':
Se o caminho A é visto primeiro, Q(*)=0.9*max(0,81)=73
Depois B é visto, Q(*)=0.9*max(100,81)=90
Ou,
Se o caminho B é visto primeiro, Q(*)=0.9*max(100,0)=90
Depois A é visto, Q(*)=0.9*max(100,81)=90
Valores de Q só aumentam, nunca diminuem.

Recompensas e Ações Não-Determinísticas

- Quando estados e recompensas são não-determinísticas temos que manter valores esperados (médias).
- Q-learning (Watkins and Dayan, 1992):

$$\hat{Q}(s_t,a_t) \leftarrow \hat{Q}(s_t,a_t) + \eta\left(r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1},a_{t+1}) - \hat{Q}(s_t,a_t)\right)$$

backup

- Off-policy vs on-policy (Sarsa)

Q-learning

Initialize all $Q(s,a)$ arbitrarily
For all episodes
  Initalize s
  Repeat
    Choose a using policy derived from Q, e.g., ϵ-greedy
    Take action a, observe r and s'
    Update $Q(s,a)$:
      $Q(s,a) \leftarrow Q(s,a) + \eta(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$
    $s \leftarrow s'$
  Until s is terminal state

Sarsa

Initialize all $Q(s,a)$ arbitrarily
For all episodes
  Initalize s
  Choose a using policy derived from Q, e.g., ϵ-greedy
  Repeat
    Take action a, observe r and s'
    Choose a' using policy derived from Q, e.g., ϵ-greedy
    Update $Q(s,a)$:
      $Q(s,a) \leftarrow Q(s,a) + \eta(r + \gamma Q(s',a') - Q(s,a))$
    $s \leftarrow s',\ a \leftarrow a'$
  Until s is terminal state

Generalização

- Tabular: *Q*(*s*,*a*) ou *V*(*s*) guardados em uma tabela
- Funcional: Usar um método de regressão para representar *Q*(*s*,*a*) or *V*(*s*)