Aprendizado por Reforço Relacional

Aluno: Thiago Yukio Sikusawa

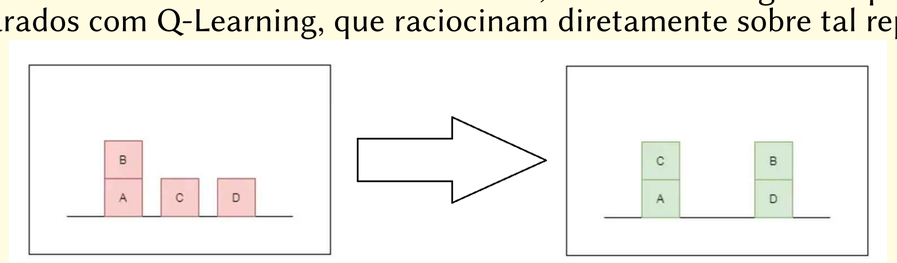
Supervisora: Leliane Nunes de Barros

IME-USP

Introdução

Uma criança aprende interagindo com o mundo em sua volta, o que a ajuda a entender as consequências de suas ações. No Aprendizado por Reforço (RL) podemos simular este processo de aprendizado humano adotando como modelo um Processo de Decisão Markoviano (*Markovian Decision Process - MDP*), formalmente definido pela tupla $\langle S, \mathcal{A}, \mathcal{R}, p \rangle$, sendo S o conjunto de estados no ambiente, $\mathcal{A}$ o conjunto de ações do agente, $\mathcal{R}$ a função de recompensa por ação e p a função de transição probabilística. O objetivo do agente é selecionar ações que maximizem o total de recompensa acumulada por um dado horizonte de interações. Um algoritmo clássico que resolve problemas deste tipo é o **Q-Learning** (Sutton e Barto, 2015), que raciocina sobre uma representação enumerativa de estados e ações.

Objetivo: estudar como representar um MDP de forma relacional, baseada em lógica de predicados, e utilizar algoritmos mais eficientes, quando comparados com Q-Learning, que raciocinam diretamente sobre tal representação.



Aprendizado por Reforço

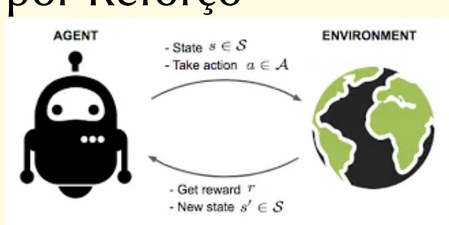


Figura 1: Interação entre agente e ambiente em um MDP.

Q-Learning

	a1	a2	a3	a4
s1	-1.2	0.5	1.1	1.9
s2	0.2	1.4	1.5	-0.1
s3	0.3	-0.1	-0.2	-0.1
s4	-2.7	0.7	0.9	0.9
s5	1.1	0.2	-0.3	1.1
s6	0.1	0.4	0.3	-0.5
s7	-2.2	-1.5	-0.9	-0.1
s8	0.0	0.2	1.4	2.0

Figura 2: Representação da função $Q(s, a)$ usada pelo algoritmo Q-Learning em que as linhas indicam os estados enumerados e as colunas as ações enumerados.

O algoritmo Q-Learning tenta aproximar a função $q^* : S \times \mathcal{A} \to \mathbb{R}$, a função-valor de uma política ótima. Uma política é uma forma do agente comportar-se no ambiente, e uma política ótima é aquela que maximiza o valor esperado da recompensa acumulada. Assim, q^* é uma função que indica, para cada par (s, a), quão bom é executar a ação a quando o ambiente está no estado s.

O algoritmo Q-Learning começa inicializando uma função de estimativa da função ótima q^*, chamada de $Q : S \times \mathcal{A} \to \mathbb{R}$, que tentará aproximar a função q^*. Uma forma de pensar sobre a estimativa Q é como se fosse uma tabela, em que para cada $s \in S$ e $a \in \mathcal{A}$ estamos guardando um valor, como na Figura 2.

Para melhorarmos a estimativa Q, o algoritmo Q-Learning faz uso de ideias da Equação de Bellman de Otimalidade, a qual diz que:

$$q^*(s, a) = \mathbb{E}[R_{t+1} + \gamma \max_{a' \in \mathcal{A}} q^*(S_{t+1}, a') \mid S_t = s, A_t = a],$$

para cada $(s, a) \in S \times \mathcal{A}$. Assim, toda vez que o agente estiver em um estado $s \in S$, executar uma ação $a \in \mathcal{A}$, obter uma recompensa $r \in \mathbb{R}$ e o ambiente mudar de estado para $s' \in S$, o algoritmo faz a seguinte atualização para a estimativa Q:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s, a)),$$

em que $\alpha, \gamma \in [0, 1]$ são parâmetros escolhidos pelo usuário.

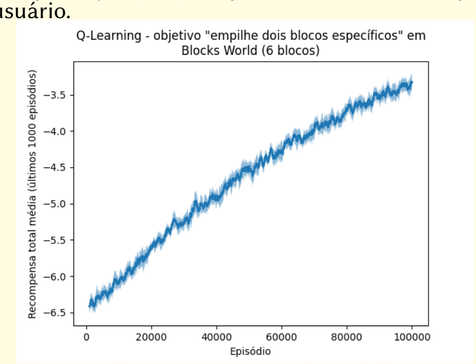


Figura 3: Resultado do algoritmo Q-Learning no Mundo dos Blocos. Recompensa esperada acumulada a cada 1000 episódios. Média de 10 épocas completas de aprendizado com intervalo de confiança de 95%.

RRL-TG

Representação relacional de estados e ações:

move(X, Y)
on(X, Y)
clear(X)
above(X, Y)
equals(X, Y)

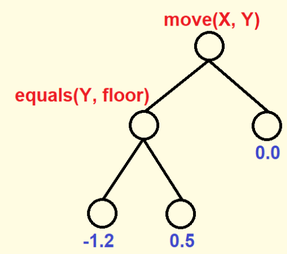


Figura 4: Representação da função $Q(s, a)$, pelo o algoritmo RRL-TG, como uma árvore binária de decisão sobre símbolos da lógica de primeira ordem.

O algoritmo RRL-TG faz uso do fato de que estados e ações são representados como relações da lógica de primeira ordem. Para isso, o algoritmo usa uma estrutura chamada de **árvore de decisão lógica de primeira ordem** (abreviada para FOLDT, do inglês *First-order Logical Decision Tree*). Em uma FOLDT, cada nó interno tem uma proposição lógica, possivelmente com variáveis, e cada folha tem um valor de retorno. Um exemplo de uma FOLDT pode ser visto na Figura 4. Assim, a estimativa do valor de um par estado e ação $(s, a) \in S \times \mathcal{A}$ com uma FOLDT T é feita usando o seguinte algoritmo:

1. Comece na raiz da árvore de decisão T.
2. Se o nó atual for uma folha, devolva o valor guardado na folha e pare o algoritmo.
3. Caso contrário, verifique a seguinte condição (C1): se existe alguma forma de substituír as variáveis no nó interno atual, junto com as proposições anteriores que sucederam, de tal forma que todas as proposições resultantes estejam em (s, a)
   (a) Se a condição C1 for verdadeira, mova para o nó à esquerda do nó atual.
   (b) Caso contrário, mova para o nó à direita do nó atual
4. Volte para o passo (2)

O algoritmo RRL-TG usa uma FOLDT T para estimar a função q^*, então precisamos de alguma forma de mudar T ao longo do algoritmo para melhorar essa estimação. Para fazer isso, guardamos diversas estatísticas em cada folha em T durante o aprendizado, e verificamos se existe alguma proposição lógica r tal que, se dividirmos uma folha f, transformando-a em um nó interno com proposição r e criando duas folhas novas abaixo de f, a variação das estimações tem uma redução estatisticamente significante. Ou seja, para cada folha f em T, verificamos se existe alguma proposição lógica r tal que, comparando os valores de:

$$\frac{n_p^{f,r}}{n^f}(\sigma_p^{f,r})^2 + \frac{n_n^{f,r}}{n^f}(\sigma_n^{f,r})^2 \quad \text{e} \quad (\sigma_{total}^f)^2$$

um Teste F suceda com nível de significância β, para algum $\beta \in (0, 0.5]$ pré-definido pelo usuário.

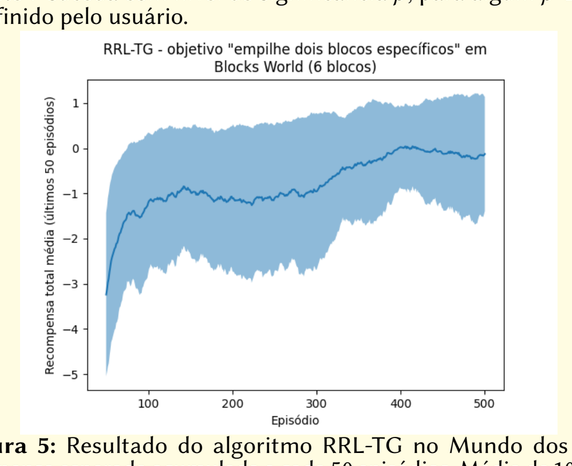


Figura 5: Resultado do algoritmo RRL-TG no Mundo dos Blocos. Recompensa esperada acumulada a cada 50 episódios. Média de 10 épocas completas de aprendizado com intervalo de confiança de 95%.

RRL-RIB

Representação relacional de estados e ações:

move(X, Y)
on(X, Y)
clear(X)

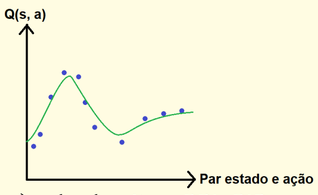


Figura 6: Representação da função $Q(s, a)$ pelo algoritmo RRL-RIB, o qual usa um conjunto de exemplos (representado pelos pontos azuis) para criar uma aproximação da função q^* (representada pela linha verde)

Para usarmos o algoritmo RRL-RIB, o maior desafio é o fato que precisamos definir uma função para determinar a distância entre dois pares estado e ação. Ou seja, precisamos definir uma função $dist : (S \times \mathcal{A}) \times (S \times \mathcal{A}) \to \mathbb{R}_+$, tal que, dados $i, j \in S \times \mathcal{A}$, o valor de $dist(i, j)$ indica o nível de semelhança entre os dois pares estado e ação, com valores maiores significando que i e j são mais diferentes. Dado uma definição da função $dist$, intuitivamente o que criamos é um senso de proximidade entre os elementos em $S \times \mathcal{A}$, e a ideia principal do algoritmo RRL-RIB é guardar um conjunto E de exemplos de elementos em $S \times \mathcal{A}$ junto com as estimações de q^* realizadas pelo algoritmo. Uma forma de visualisar o conjunto de exemplos E, junto com o espaço métrico criado pela função $dist$, é mostrado na Figura 6.

Com o conjunto E e a função $dist$, dado um par estado e ação $i \in S \times \mathcal{A}$, podemos usar a ideia de que a estimação da função valor-ação de i deve ser próximo da estimação de outros pares estado e ação próximos de i. Formalmente, a estimação para i usando E é:

$$\hat{q}^E(i) := \frac{\sum_{j \in E} \frac{q_j}{dist(i,j)+\delta}}{\sum_{j \in E} \frac{1}{dist(i,j)+\delta}},$$

em que δ é uma constante real pequena para evitar divisão por zero. Podemos usar a recompensa r e o próximo estado s', junto com a estimação $\hat{q}^E$, para computar $q^E(i)$, uma aproximação de $q^*(i)$, usando uma regra semelhante ao Q-Learning:

$$q^E(s, a) := \widehat{q}^E(s, a) + \alpha(r + \gamma \max_{a' \in \mathcal{A}} \widehat{q}^E(s', a') - \widehat{q}^E(s, a)).$$

Quanto mais exemplos tivermos em E, melhor será a estimação $\hat{q}^E$, porém, na prática ter muitos exemplos deixa o algoritmo RRL-RIB muito lento, então precisamos limitar a quantidade de exemplos que adicionamos em E. Para fazer isso, dado um novo par estado e ação $i \in S \times \mathcal{A}$, calculamos $\sigma_{local}^E(i)$, o desvio padrão das estimações em E de exemplos próximos de i, e também calculamos σ_{global}^E, o desvio padrão das estimações de todos os exemplos em E. Assim, adicionamos i em E se pelo menos um dos seguintes critérios for satisfeito:

$$|\hat{q}^E(i) - q^E(i)| > \sigma_{local}^E(i) \cdot F_l, \quad \text{ou} \quad \sigma_{local}^E(i) > \frac{\sigma_{global}^E}{F_g},$$

em que $F_l, F_g \in \mathbb{R}$ são constantes escolhidas pelo usuário. Além disso, se o tamanho de E ficar muito grande, precisamos determinar um exemplo em E para excluir. A ideia usada pelo algoritmo RRL-RIB é escolher um exemplo que mais contribui para os erros de estimações. Isso é feito calculando, para cada $i \in E$, o valor de:

$$EP\text{-}score(i) := \sum_{\substack{j \in E \\ j \neq i}} \frac{|q^E(j) - \hat{q}^E(j)|}{\text{dist}(i, j) + \delta},$$

e remover um exemplo em E que maximiza esse valor.

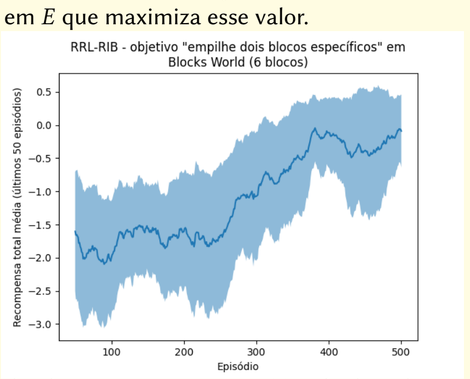


Figura 7: Resultado do algoritmo RRL-RIB no Mundo dos Blocos. Recompensa esperada acumulada a cada 50 episódios. Média de 10 épocas completas de aprendizado com intervalo de confiança de 95%.

Conclusões

Com os resultados obtidos ao compararmos o aprendizado por reforço relacional (RRL-TG e RRL-RIB) e o enumerativo (Q-Learning), é possível chegar nas seguintes conclusões:

- Algoritmos de RRL em geral precisam de bem menos episódios de treinamento para chegar no mesmo desempenho ou melhor que Q-Learning.
- Algoritmos de RRL têm uma variância bem maior no desempenho do agente comparado com Q-Learning. Isso significa que há uma inconsistência maior no resultado final com RRL comparado com Q-Learning, dado que em lógica relacional são feitas abstrações sobre a função $Q(s, a)$.
- Diferente de Q-Learning, é possível observar que algoritmos de RRL, apresentam uma queda de desempenho durante o treinamento. Uma justificativa para isso pode ser a maneira como as relações fazem abstrações de conjuntos de estados.
- Algoritmos de RRL em geral precisam de bem mais tempo de computação por episódio comparado com Q-Learning (vide monografia).

Bibliografia

- Driessens, Kurt (mai. de 2004). "Relational Reinforcement Learning". Tese de dout. Leuven, Bélgica: Universidade Católica de Lovaina.
- Sutton, Richard S. e Andrew G. Barto (2015). *Reinforcement Learning: An Introduction*. 2ª ed. The MIT Press.

Bolsa durante a graduação

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

Universidade de São Paulo

Instituto de matemática e estatística

Site do TCC: https://doublefloat1.github.io/mac0499/

Universidade de São Paulo