Trabalho Prático II - Introdução à Inteligência Artificial

Luís Felipe Ramos Ferreira

lframos.lf@gmail.com

1 Introdução

O Trabalho Prático II da disciplina de Introdução a Inteligência Artificial teve como objetivo a implementação do algoritmo de `Q-Learning` para encontrar o melhor caminho entro um ponto inicial e um objetiov em uma mapa bidimensional.

2 Implementação

O projeto foi implementado na linguagem Python, versão 3.12.3. Um arquivo `requirements.txt` com os pacotes utilizados no ambiente virtual criado para desenvolvimento está disponibilizado. O único pacote fora dos já disponibilizados por padrão na Linguagem foram Numpy, Pandas e Matplotlib. Instruções para rodar o programa estão disponibilizadas no arquivo `README.md` disponibilizado.

3 Q-learning

O algoritmo de `Q-Learning` é um algoritmo de aprendizado por reforço, em que o agente, ao explorar o ambiente e interagir com o que ele possui, passa a aprender mais sobre como alcançar seus objetivos. Para cada possível estado s, ele seleciona a melhor ação a para se tomar, obtendo assim o estado s' que será alcançado ao aplicar a em s, assim como uma recompensa r por aplicar essa ação. Por fim, ele atualiza o valor de $Q(s, a)$, que é mantido para todo par de estado e ação e indica quão bom é executar aquela ação partindo daquele estado.

O pseudocódigo abaixo, inspirado nos slides da disciplina, indica, em alto nível, como funciona o algoritmo.

- Inicializa $Q(s, a)$ para todos os estados e ações
- $s \leftarrow$ estado inicial

- $n \leftarrow$ número de episódios
- $\epsilon \leftarrow$ limiar $\epsilon - greedy$
- $\gamma \leftarrow$ taxa de desconte
- $\alpha \leftarrow$ taxa de aprendizado
- Para todo $i = 0$ até n
  - se $random() < \epsilon : a \leftarrow random\_action()$
  - senão $a \leftarrow argmax_a(Q(s, a))$
  - Executa ação a, observa recompensa r e próximo estado s'
  - $Q(s, a) \leftarrow Q(s, a) + \alpha * [r + \gamma * max_{a' \in A} Q(s', a') - Q(s, a)]$
  - $s \leftarrow s'$

Uma característica do `Q-Learning` é que ele é um algoritmo chamado de *model free*, que em alto nível significa dizer que em nenhum momento o agente precisa aprender ou estimar a função de transição que ele utiliza. Ele também é um algoritmo *off-policy*, ou seja, a política utilizada pelo agente na exploração pode ser diferente da política que ele está aprendendo (*target policy*).

O algoritmo possui algumas pequenas variações, as quais exploramos no trabalho, e estão descritas a seguir.

- **STANDARD**: nesta versão, utilizamos o algortimo normalmente, com a tabela de recompensas descrita na especificação.
- **POSITIVE**: também utilizamos a tabela descrita na especificação do trabalho. Nesa versão, toda recompensa é positiva.
- **STOCHASTIC**: nesta versão, após escolher uma ação, o algortimo tem 20% de chance de na verdade executar outra ação perpendicular à ação escolhida, conforme descrito na especificação.

4 Estruturas e modelagem

A linguagem *Python* facilitou muito o trabalho. Para modelar as 4 ações possíveis, utilizamos números inteiros. EM particular, para ficar mais organizado, utilizamos a estrutura *IntEnum*. O mapa é um *array* da biblioteca *numpy*, que armazena caracteres. A matriz de pesos é uma matriz tridimensional, também *numpy*, que armazena floats. Ela possui as dimensões do mapa de entrada, mas cada posição é composta por um vetor de 4 posições, uma para cada possível ação.

As variações **STANDARD** e **POSITIVE** foram implementadas na mesma função, chamada de *qlearning*, pois elas são idênticas a menos do valor das recompensas, que é mudado dinamicamente conforme os parâmetros de linha de

comando. A variação **STOCHASTIC** é implementada numa função diferente, chamada *stochastic*, para facilitar a organização.

As implementações seguem exatamente as especificações do algoritmo apresentadas na disciplina.

5 Análise das diferentes políticas

Para analisar as diferentes políticas, testamos elas com os diferentes mapas disponibilizados no *moodle* da disciplina. Os resultados obtidos foram os seguintes. Nas tabelas, a célula verde indica o ponto inicial e a célula vermelha indica o ponto onde se encontra o objetivo.

5.1 Mapa teste

Para o mapa teste inicial, que pode ser visto abaixo, as políticas encontradas para cada uma das variações foram as seguintes.

+	+	+	.	O
.	@	@	.	x
.	@	@	.	.
.	;	;	;	.

STANDARD

$>$	$>$	$>$	$>$	O
v	@	@	$\hat{}$	x
v	@	@	$\hat{}$	$<$
$>$	$>$	$>$	$\hat{}$	$\hat{}$

POSITIVE

$>$	$>$	$>$	$<$	O
v	@	@	$\hat{}$	x
v	@	@	$>$	v
$>$	$>$	$>$	$\hat{}$	$\hat{}$

STOCHASTIC

$>$	$>$	$>$	$>$	O
$\hat{}$	@	@	$>$	x
$\hat{}$	@	@	$>$	$<$
$\hat{}$	$>$	$>$	$\hat{}$	$\hat{}$

A política foi encontrada para o ponto inciial $(0, 3)$, com 10000 episódios. EM particular, para a variação **POSITIVE**, que demora bastante, utilizamos 500 iterações. Podems ver que todas encontram uma política satisfatória para encontrar o melhor caminho para o objetivo. Em particular, a variação **STANDARD** foi a mais rápida e a que, a princípio, encontrou o melhor resultado.

5.2 Mapa labirinto (*Maze*)

O mapa *Maze* está exposto abaixo.

x	@	x	@	x	@	x	@	x	@	.	@
.	.	.	.	.	.	.	.	.	.	.	@
.	x	@	@	@	@	.	@	.	@	@	@
.	x	@	.	.	.	.	.	.	@	.	@
.	x	@	.	@	@	@	@	@	@	@	@
.	x	@	.	.	.	.	.	.	@	.	.
.	x	@	@	@	@	@	@	@	@	@	.
O	.	.	.	.	.	.	.	.	.	.	.

STANDARD

x	@	x	@	x	@	x	@	x	@	v	@
v	$<$	$<$	$<$	$<$	$<$	$<$	$<$	$<$	$<$	$<$	@
v	x	@	@	@	@	@	@	@	@	ˆ	@
v	x	@	$<$	$<$	$>$	$<$	$>$	$<$	$>$	ˆ	@
v	x	@	$>$	@	@	@	@	@	@	@	@
v	x	@	ˆ	$<$	$<$	ˆ	v	$<$	$>$	$>$	x
v	x	@	@	@	@	@	@	@	@	$>$	@
O	v	ˆ	$>$	ˆ	ˆ	v	$<$	v	$>$	$>$	@

POSITIVE

x	@	x	@	x	@	x	@	x	@	v	@
v	$<$	$<$	$<$	$<$	$<$	$<$	$<$	$>$	$<$	$<$	@
v	x	@	@	@	@	@	@	@	@	ˆ	@
v	x	@	v	ˆ	v	v	ˆ	ˆ	$<$	ˆ	@
v	x	@	ˆ	@	@	@	@	@	@	@	@
v	x	@	v	$>$	$<$	ˆ	$>$	ˆ	v	ˆ	x
ˆ	x	@	@	@	@	@	@	@	@	v	@
O	ˆ	$<$	v	v	$>$	v	ˆ	v	$<$	v	@

STOCHASTIC

x	@	x	@	x	@	x	@	x	@	v	@
v	$<$	$<$	$>$	ˆ	$<$	ˆ	$>$	ˆ	$<$	$<$	@
v	x	@	@	@	@	@	@	@	@	ˆ	@
$>$	x	@	v	$<$	$<$	$<$	v	$>$	$>$	ˆ	@
v	x	@	v	@	@	@	@	@	@	@	@
v	x	@	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$	x
$>$	x	@	@	@	@	@	@	@	@	ˆ	@
O	$<$	$<$	$<$	$<$	$<$	$<$	$<$	$<$	$>$	ˆ	@

Podemos notar nesse mapa maior que, mais uma vez, a variação **STANDARD** apresentou uma política melhor que a dos demais. No entanto, podemos ver que a performance das outras duas deixou a desejar. Para um número de episódios igual a 30000, a variação **STOCHASTIC** não conseguiu encontrar uma política "perfeita" que acha o melhor caminho em direção ao objetivo. Em muitos estados, a melhor

ação a se tomar calculada ao final do algoritmo não aparenta ser de fato a melhor. A variação **POSITIVE** é ainda pior, as vezes achando caminhos opostos ao ótimo.

Vale a pena citar que, assim como no mapa teste, a variação **POSITIVE** demorou muito, e o número de episódios teve que ser diminuído para obtermos uma resposta rápidamente.

5.3 Mapa escolhas (*Choices*)

O mapa *choices* está exposto abaixo.

@	.	.	.	.	.	.	.	.	.	@
@	+	@	.	x	@	x	.	@	;	@
@	+	@	.	@	@	x	.	@	;	@
@	+	@	.	x	@	x	.	@	;	@
@	+	.	.	@	@	x	.	.	;	@
@	+	@	.	x	@	x	.	@	;	@
@	+	@	.	@	@	x	.	@	;	@
@	+	.	.	.	O	.	.	.	;	@

STANDARD

@	>	>	>	>	>	>	v	<	<	@
@	^	@	^	x	@	x	v	@	^	@
@	^	@	v	@	@	x	v	@	^	@
@	^	@	v	x	@	x	v	@	^	@
@	>	>	^	@	@	x	v	<	<	@
@	<	@	v	x	@	x	v	@	^	@
@	>	@	^	@	@	x	v	@	^	@
@	v	>	^	<	O	<	<	<	<	@

POSITIVE

@	v	>	v	<	<	<	<	<	<	@
@	v	@	v	x	@	x	v	@	>	@
@	v	@	v	@	@	x	^	@	v	@
@	v	@	v	x	@	x	<	@	>	@
@	>	>	<	@	@	x	<	<	v	@
@	^	@	v	x	@	x	<	@	^	@
@	v	@	v	@	@	x	v	@	>	@
@	>	>	>	>	O	<	^	<	>	@

STOCHASTIC

@	>	>	v	v	>	v	v	<	<	@
@	^	@	>	x	@	x	<	@	v	@
@	v	@	v	@	@	x	<	@	v	@
@	<	@	>	x	@	x	<	@	v	@
@	>	>	^	@	@	x	^	>	<	@
@	^	@	>	x	@	x	^	@	v	@
@	^	@	^	@	@	x	<	@	v	@
@	>	>	^	<	O	>	^	>	<	@

Algumas análises interessantes podem ser feitas nesse mapa. Mais uma vez, a variação **STANDARD** foi superior às outras e obteve uma política satisfatória. A variação **POSITIVE**, apesar de mais uma vez muito lenta, também achou uma política satisfatória. O curioso nesse caso é que o caminho mais "indicado" pela política é o que desce para o objetivo pela esquerda, ao contrátio da variação anterior, que indicava que o melhor caminho era descer pela direita. A variação **STOCHASTIC**, no entanto, teve um desempenho extremamente ruim nesse mapa. A política encontrada não apresenta qualquer tipo de sentido e não aponta boas escolhas para se chegar ao objetivo. Meu palpite inicial era de que talvez o número de iterações fosse muito baixo para que o algoritmo desempenha-se bem, mas mesmo aumentando o número de episódio a performance não melhorou.

6 Conclusão

O trabalho foi uma excelente maneira de aprender mais e colocar em prática os conhecimentos adquiridos sobre aprendizado por reforço, em particular em relação ao algoritmo *Q-Learning*. O algoritmo possui uma implementação simples e a linguagem *Python* ajudou muito também, por ser de fácil uso. Em relação às variações propostas, a **STANDARD** aparentou ser a melhor de todas, como esperado. A variação **STOCHASTIC** teve um desempenho bom nos primeiros testes, mas no último mapa os resultados não foram satisfatórios. Um estudo mais profundo sobre o caso desse mapa em específico e como ocorreu o aprendizado do agente é necessário para compreender o que está acontecendo. A variação **POSITIVE**, por sua vez, até conseguia encontrar políticas interessantes para o agente, mas seu tempo de execução era extremamente lento, o que classificou essa variação negativamente.