Relatório da Atividade Sobre IA em Robótica Móvel

Introdução à Robótica

João Marcello Mendes Moreira

E-mail: joaomarcello.mm@gmail.com

Professor: Areolino de Almeida Neto

Resumo da Atividade. Implementar um programa que faça um agente decidir o caminho mais curto sem colisões até o destino utilizando aprendizado por reforço (SARSA) considerando um obstáculo movendo-se a uma velocidade variável. O agente pode mover-se uma célula por vez enquanto que o obstáculo move-se de 1 a 3 células por passo, sempre para baixo e, caso saia do limite inferior do mapa retorna para a parte superior. O mapa deve possuir 9 linhas e 14 colunas.

Implementação

Utilizou-se a linguagem Python 3.7 para a realização da atividade. O agente foi definido como um objeto que possui os atributos x e y, que representam a coluna e a linha em que ele está posicionado, respectivamente. Ele pode executar as seguintes ações:

- ficar parado
- ir para cima
- ir para cima-esquerda
- ir para esquerda
- ir para esquerda-baixo
- ir para baixo
- ir para direita-baixo
- ir para direita
- ir para direita-cima

No total, o agente pode fazer 9 ações. Caso ocorra de o agente tentar fazer um movimento que o faria sair do mapa (por exemplo, tentar ir para cima quando se está na borda superior do mapa), a ação não será executada e ele ficará parado.

O ambiente é composto por uma matriz M x N com M = 9 e N = 14, um obstáculo um ponto de chegada (ambos objetos com posição x e y).

Utilizou-se 4 variáveis de estado, que foram definidas como se segue:

- Posição do agente (varia de 0 a 127): Calcula a posição relativa do agente na matriz utilizando seus atributos x e y utilizando a fórmula (y * N) + x, onde N é a quantidade de colunas da matriz.
- Distância entre o agente e o objetivo (varia de 0 a 11): Quantidade mínima de células pelas quais o agente precisaria se mover para alcançar a posição do objetivo.
- Distância entre o agente e o obstáculo (varia de 0 a 8): Quantidade mínima de células pelas quais o agente precisaria se mover para alcançar a posição do obstáculo.
- Posição y do obstáculo (varia de 0 a 8): A linha em que o obstáculo se encontra na matriz.

Para o aprendizado, o algoritmo escolhido foi o SARSA. As constantes *alpha* e *gamma* do algoritmo foram escolhidas empiricamente. Para iniciar o treinamento, deve-se executar o script “train.py”. É possível alterar o valor das constantes nas primeiras linhas do código. Abaixo, uma descrição de cada constante:

- SHOW_EXECUTION (booleano): Se exibe ou não o treinamento na tela. Pode ser True ou False.
- SAVE_EACH (inteiro): Quantidade de episódios que devem ocorrer para salvar a matriz Q.
- total_episodes (inteiro): Quantidade de episódios que o experimento executará.
- max_steps: Número máximo de passos em cada episódio.

- epsilon (float, valores entre 0 e 1): Constante para a política ε-greedy (usada na escolha da ação). Quanto maior o valor, maior a probabilidade de o agente executar uma ação aleatória.
- alpha (float, valores entre 0 e 1): Constante α do algoritmo SARSA
- gamma (float, valores entre 0 e 1): Constante ϒ do algoritmo SARSA

Primeiramente, o programa tenta carregar a matriz Q do disco (para retomar o treinamento, caso já se tenha feito algum). Caso não consiga, o treinamento ocorrerá do zero e a matriz Q resultante será salva em disco em um arquivo de nome “q”. A matriz Q terá a dimensão (126, 12, 9, 9, 9), referentes aos valores máximos de cada variável de estado explicados anteriormente, mais a quantidade de ações do agente.

Para visualizar o resultado do treinamento, deve-se executar o script “test.py”, que recebe um parâmetro pela linha de comando. Para visualizar o teste execute a seguinte linha de comando “`python teste.py true`”. Se não quiser visualizar, use “`python teste.py false`”. Ao final, o programa exibe alguns resultados, como a quantidade de episódios que o agente conseguiu chegar no objetivo sem colisão, o número total de colisões, e a quantidade de vezes “imperfeitas” (vezes em que o agente chegou ao objetivo, mas não em 10 passos). A quantidade de episódios do experimento pode ser definida alterando o valor da constante MAX_EPISODES nas linhas iniciais do código. Caso o programa não encontre a matriz Q no disco, ele exibe uma mensagem de erro e finaliza a sua execução.

Experimentos

Cada experimento foi treinado com 100.000 episódios, N vezes, apenas variando os valores das constantes alpha, gamma e epsilon. Para avaliar o desempenho, usou-se a quantidade de colisões e a quantidade de vezes imperfeitas (chegada ao objetivo em mais de 10 passos) em 50.000 episódios.

Experimento 1

Alpha = 0,005

Gamma = 0,8

Epsilon: 0,2

Tabela 1-A: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	48.349	96,7

Tabela 1-B: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	44.500	89

Tabela 1-C: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	40.812	81,62

Tabela 1-D: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	39.270	78,54

Tabela 1-E: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	38.599	77,2

Experimento 2

Alpha = 0,05

Gamma = 0,8

Epsilon: 0,2

Tabela 2-A: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	29.733	59,47

Tabela 2-B: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	24.445	48,89

Tabela 2-C: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	16.306	32,61

Tabela 2-D: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	14.468	28,94

Tabela 2-E: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	29.733	59,47

Tabela 2-F: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	14.468	28,94

Experimento 3

Alpha = 0,1

Gamma = 0,8

Epsilon: 0,2

Tabela 3-A: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	31.765	63,47

Tabela 3-B: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	28.443	56,89

Tabela 3-C: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	27.914	55,83

Tabela 3-D: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	23,131	46,26

Tabela 3-E: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	28.080	56,16

Tabela 3-F: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	27.685	55,37

Experimento 4

Alpha = 0,05

Gamma = 0,9

Epsilon: 0,2

Tabela 4-A: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	32.391	64,78

Tabela 4-B: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	37.572	75,14

Tabela 4-C: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	26.205	52,41

Tabela 4-D: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	26.283	52,57

Tabela 4-E: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	25.235	50,47

Tabela 4-F: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	38.225	76,45

Experimento 5

Alpha = 0,05

Gamma = 0,6

Epsilon: 0,2

Tabela 5-A: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	31.432	62,86

Tabela 5-B: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	31.159	62,32

Tabela 5-C: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	27.727	55,45

Tabela 5-D: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	21.731	43,46

Tabela 5-E: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	24.873	49,75

Tabela 5-F: Resultado do experimento.

	Quantidade	Porcentagem
Colisões	0	0
Imperfeitos	23.996	47,99

Discussão dos resultados

A partir dos resultados, nota-se que o agente consegue aprender a evitar a colisão ainda nos primeiros 100.000 episódios (como foi constatado em cada um dos 5 experimentos). Contudo, para aprender o melhor o caminho é necessário uma quantidade maior de episódios.

No experimento 1, onde alpha foi definido como 0,005, a quantidade de vezes imperfeitas feitas pelo agente diminui conforme deixamos mais tempo no treinamento, apesar dessa diminuição ser bastante lenta por conta do valor muito pequeno da constante alpha. Ao final, este experimento totalizou 38.599 vezes imperfeitas.

No experimento 2, alpha foi definido como 0,05, e a diminuição das vezes imperfeitas foi mais acentuada, iniciando com 29.733 e finalizando com 14.468. Já no experimento 3, onde alpha foi de 0,1, a quantidade de vezes imperfeitas começou a oscilar. Assim, concluiu-se que o melhor valor para alpha é 0,05 considerando os 3 primeiros experimentos.

Nos próximos experimentos, buscou-se alterar os valores de gamma. No experimento 4, a constante gamma foi definida como 0,9 e a quantidade de vezes imperfeitas oscilou muito, assim como no experimento 5 (gamma = 0,6).

Logo, a melhor combinação de valores para as constantes foi o do experimento 2 (*alpha* = 0,05 / *gamma* = 0,8 / *epsilon*: 0,2).

Problemas conhecidos

Por conta da ação “ficar parado” ser a primeira opção do agente, foi necessário fazer uma pequena alteração na maneira como ocorre a recompensa. Esta se dá da seguinte maneira:

- Se o agente colidiu, o valor é -1.
- Se chegou ao objetivo, o valor é 1.
- Senão, o valor é – (10^{-10}).

O motivo é que se o valor permanecer zero, o agente só se moverá caso a probabilidade de uma ação aleatória da politica ε-greedy ocorrer, o que pode fazer com que tempo de treinamento aumente consideravelmente. Portanto, a solução utilizada para o problema foi atribuir um valor negativo próximo de zero como recompensa nos casos em que o agente não colidiu e nem chegou ao objetivo.