

## Trabalho Prático II: Aprendizado por Reforço

Valor: 15 pontos

Data de entrega: 29/01/2025

### 1. Definição do Problema

Iremos revisitar o cenário do Trabalho Prático I. Dessa vez, o objetivo será praticar conceitos e algoritmos de aprendizado por reforço. Além disso, proporcionará uma forma de entender como funções de recompensa e o ambiente afetam a política gerada pelo agente.

Relembrando o problema, será proposto a resolução do Path-Finding, o problema de encontrar um caminho entre dois pontos. Sua aplicação é essencial para diversas áreas. Em jogos, os métodos são usados para guiar a movimentação de NPCs. O mundo do jogo (podendo ser 2D ou 3D) é discretizado em áreas geométricas, como mostrado na Figura 1. Com isso, o tamanho do espaço de estados é reduzido. As possíveis ações do agente são caminhar entre essas áreas, executando um movimento entre quatro direções (cima, baixo, direita e esquerda). Quando uma solução é encontrada, o agente apenas executa as ações até alcançar o alvo.

Uma questão importante é que, muitas vezes, alguns caminhos são preferíveis a outros. Por exemplo, entre escolher o caminho pegando fogo, que é direto ao objetivo, ou decidir contorná-lo gastando mais passos. Para este trabalho, iremos lidar com esse problema atribuindo diferentes recompensas baseadas no terreno (estado) que o agente move.

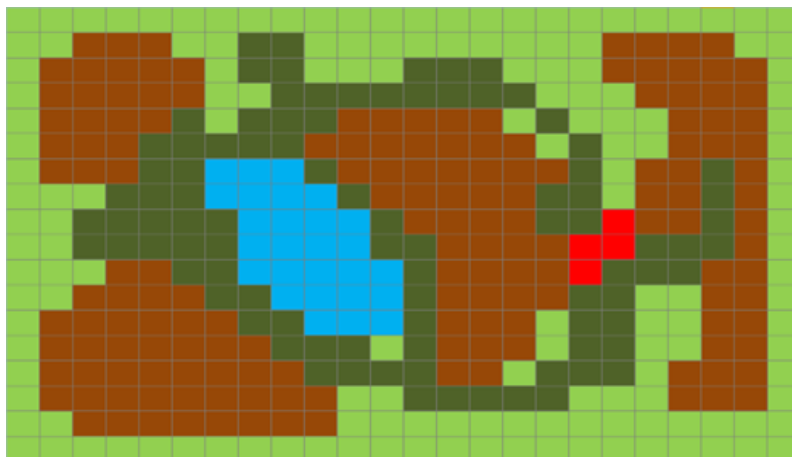


Figure 1. Exemplo de um mapa discretizado a partir de um mundo 3D.

Você deve implementar o método Q-Learning. O seu programa receberá uma representação discretizada do mapa, um estado inicial e um número de passos que o agente deve ser treinado. A saída será a política gerada após o número de passos descrito.

## 2. Mapa

O mapa que seu programa receberá será a versão discretizada de um mundo. Esta é modelada por uma matriz bidimensional. Cada posição da matriz representa uma área do mapa original do jogo e terá um símbolo identificando o tipo de terreno. Os símbolos e suas respectivas recompensas estão descritos na Tabela 1.

Terreno	Símbolo	Recompensa
Grama	.	-0.1
Grama Alta	;	-0.3
Água	+	-1.0
Fogo	x	-10.0
Objetivo	O	10.0
Parede	@	$-\infty$

Table 1. Símbolos e recompensas associados a cada terreno.

Alguns detalhes para essa nova modelagem:

- Quando o agente se encontrar em um estado representado por fogo ou objetivo, o episódio acaba e ele receberá a recompensa apresentada na tabela.
- Um terreno de parede representa novamente um terreno intransponível. Dessa vez, a ação em direção a esse terreno é **válida**, porém o agente **se manterá no estado em que se encontra**. Um movimento para fora dos limites do mapa deve ser considerado como um movimento em direção a uma parede.

As coordenadas assumidas neste trabalho têm como  $(0, 0)$  o canto superior esquerdo do mapa. Um ponto  $(x, y)$  representa a altura  $y$  e o comprimento  $x$  daquele ponto.

## 3. Ações

O agente pode se movimentar em quatro direções (cima, baixo, direita e esquerda). Movimentos para fora dos limites do mapa ou para terrenos intransponíveis são válidos, mas o estado resultante se manterá o mesmo.

## 4. Métodos

Para o trabalho, você deve implementar o método Q-Learning padrão. Para a taxa de aprendizado, utilize o valor  $\alpha = 0.1$  e para a taxa de desconto  $\gamma = 0.9$ . A sua implementação deve fazer uso de um **método de exploração**. Para o trabalho, implemente o  $\epsilon$ -greedy com o valor de  $\epsilon = 0.1$ .

Além disso, outras duas versões modificadas devem ser realizadas para fins de comparação de políticas:

#### 4.1. Recompensas Positivas

Essa versão apenas modifica a recompensa atribuída para os terrenos. A nova função de recompensa é dada pela Tabela 2.

Terreno	Símbolo	Recompensa
Gramma	.	3.0
Gramma Alta	;	1.5
Água	+	1.0
Fogo	x	0.0
Objetivo	O	10.0
Parede	@	$-\infty$

Table 2. Símbolos e recompensas associados a cada terreno na versão modificada.

#### 4.2. Ambiente Estocástico

Nesta versão, ao escolher e executar uma ação, existe uma chance de 20% de uma ação perpendicular à escolhida ser executada pelo ambiente, sendo 10% para a perpendicular direita e 10% para a esquerda. Assim, a ação escolhida será realizada sem alteração pelo ambiente com 80% de chance.

Por exemplo, quando o agente escolhe ir para cima  $\hat{}$ , existe uma chance de 20% da ação executada ser  $<$  ou  $>$ . A grande diferença é que, na visão do agente a ação executada foi  $\hat{}$ , logo os pesos atualizados serão do estado  $Q(s, \hat{})$ .

### 5. Entrada do Programa

O seu programa receberá como argumento uma string que representa o nome do arquivo texto onde o mapa e suas informações serão descritas. Além disso, uma outra string que identifica qual modificação que deve ser utilizado. Por fim, 3 inteiros que representam as coordenadas  $(x_i, y_i)$  do estado inicial e  $n$  o número de passos que o agente deve ser treinado. Um exemplo de um programa em Python nomeado como *qllearning*, o seguinte comando deve ser usado para rodá-lo:

```
python qllearning [caminho_para_arquivo_mapa] [identificador_modificacao]  $x_i$   $y_i$   $n$ 
```

Em que [identificador\_modificacao] pode ser *standard*, *positive*, *stochastic* representando: **Q-Learning padrão sem modificações**, **Q-learning com recompensas apenas positivas**, **Q-Learning com ambiente estocástico** respectivamente.

[caminho\_para\_arquivo\_mapa] será a string com o caminho para o arquivo texto.

O arquivo texto apresentará todas as informações referentes ao mapa. A primeira linha do arquivo possui dois inteiros  $W$  e  $H$ , representando o número de colunas e linhas da matriz respectivamente. Em seguida, as próximas  $H$  linhas terão  $W$  caracteres representando o tipo de terreno naquela posição sem espaço entre os caracteres. Um exemplo de entrada é descrito a seguir:

```
1 5 4
2 +++ .O
3 .@@.x
```

```

4  . @ @ . .
5  . ; ; ; .

```

## 6. Saída do Programa

Seu programa deve computar os valores  $Q$  de cada estado do mapa ao longo do treino partindo do estado inicial dado na entrada. Ao final dos  $n$  passos, deve-se imprimir na tela o mapa, e para cada posição transponível, um indicador mostrando qual a melhor ação a ser tomada naquele estado. Utilize os indicadores  $>$ ,  $<$ ,  $^$ ,  $\vee$  para indicar as direções direita, esquerda, cima e baixo respectivamente. Para estados terminais (como fogo e objetivo) e intransponíveis, o símbolo de seu terreno deve ser impresso. **Não deve** ter espaços em branco nem nenhum outro caractere entre os símbolos. Seu programa deve imprimir **apenas** o descrito.

Um exemplo de política gerada após 100.000 passos com o estado inicial (0,3) é dada a seguir:

```

1  \>>>O
2  \@@@^x
3  \@@@^<
4  >>>^^

```

## 7. Entrega

Você deverá entregar um arquivo **ZIP** no Moodle da disciplina. O arquivo **ZIP** deve conter o código-fonte da sua implementação em **Python**. Além disso, deve conter uma documentação em formato **PDF**. Essa deve descrever de forma sucinta as decisões de implementação e análises. Mais especificamente, deve-se incluir:

- Apresentação das estruturas usadas e da modelagem dos componentes (estado, agente, ambiente, etc.).
- Breve descrição do método utilizado e das eventuais modificações.
- Análise comparando as políticas geradas pelo método original e suas modificações. Qual o efeito da mudança? A política se alterou? Porque?

Por fim, deve-se ter um **README** descrevendo como rodar (e compilar caso a linguagem necessite) o seu programa. Assim como uma lista de bibliotecas utilizadas.

## 8. Considerações Finais

- Sinta-se livre para discutir o trabalho com seus colegas. Entretanto, o compartilhamento de códigos ou texto é plágio e será devidamente punido. Seu trabalho deve ser de sua autoria.
- Use o fórum de discussão do Moodle em caso de dúvidas. Caso prefira, pode enviar um email para o monitor da disciplina.
- **Bom trabalho!**