INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Introdução à Aprendizagem Automática — 2024/2025
Aprendizagem por Reforço

> Estes exercícios devem ser resolvidos utilizando Python notebooks devido à possibilidade de gerar um relatório integrado com o código. É assumido que o estudante é proficiente em programação. Todas as respostas devem ser justificadas e os resultados discutidos e comparados com os *baselines* apropriados.
> A pontuação máxima da tarefa é de 2 pontos. Os exercícios opcionais (se existentes) ajudam a alcançar a pontuação máxima, complementando erros ou falhas.
> **Deadline:** final da aula da semana de 11 a 15 de novembro, 2024

Imagine uma situação em que um robot (*Smiley*) precisa de **aprender a sequência de acções** que o levam de uma posição inicial (estado 1) para a tomada eléctrica (estado 100, marcada com **X**). Imagine que pode experimentar a aprendizagem num ambiente simulado (simplificado), como o cenário representado na Figura 1.

Suponhamos (por sobre-simplificação) que a sala pode ser dividida em quadrados e que estas **"áreas" (a que chamaremos estados) estão numeradas** para que o robot possa identificar cada posição diferente na sala. Imagine também que o robot tem **quatro acções (cima / baixo / esquerda / direita)** e que estas vão do meio de um quadrado para o meio de um quadrado adjacente. Do ponto de vista deste robot, ele sabe que está no estado 1 e se se mover para a direita receberá a informação de que chegou ao estado 2, se se mover para baixo será informado de que chegou ao estado 11, se tentar mover-se noutra direção (considerando que há paredes para cima e para a direita) será informado de que permaneceu no estado 1.

Ao chegar ao estado 100, o robot recebe uma recompensa.

Exercício 1

Construa o ambiente de simulação. Desenvolva as seguintes funcionalidades:

a) A função de transição de estado ($s' = f(s, a)$), onde um estado (s) e uma ação (a) são

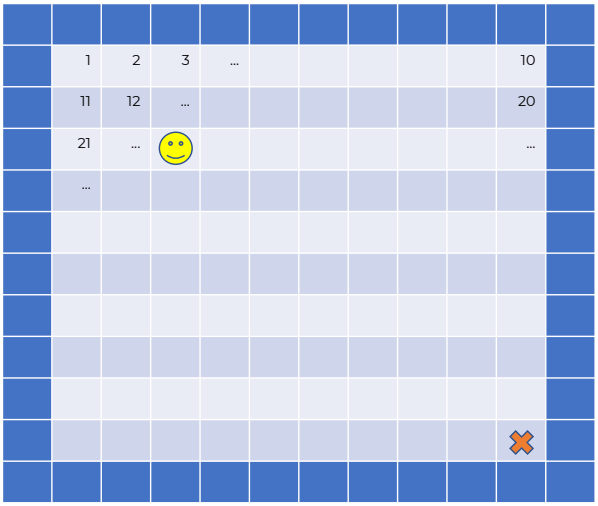

Figura 1: Ambiente simplificado (os estados são numerados de 1 a 100). Os quadrados azuis escuros são paredes.

dados como argumento, e um novo estado (o estado de chegada, s') é devolvido, de modo que: $f(1, right) = 2$, $f(1, down) = 11$, $f(1, up) = 1$, etc

b) Uma função de recompensa $r(s)$ que recompensa todos os estados com 0 e o estado objetivo (estado 100) com 100 pontos.

c) Uma função que escolhe uma ação aleatoriamente.

d) Defina o fim do episódio – quando o robot atingir o estado objetivo (a tomada elétrica, marcada com um X, estado 100), este deve ser re-colocado na posição inicial após receber a recompensa. Quando o robot executar 1000 ações sem atingir o objetivo, este deve ser re-colocado à posição inicial sem recompensa.

e) Simule o robot a realizar um episódio e repita-o 30 vezes. Meça e registe a recompensa média por passo em cada episódio e o número de passos para atingir o objetivo em cada episódio. Calcule a média e o desvio padrão do número de passos para atingir o objetivo, tempos de execução e recompensas para os 30 testes. Estes serão os resultados de referência e serão usados para testar se o sistema está a ter um desempenho melhor do que apenas adivinhar aleatoriamente.

f) Represente a média e o desvio padrão (recompensa, passos para atingir o objetivo e tempos de execução), cada um num *boxplot* diferente com caixas verticais (Figura 2).

Dicas sobre a apresentação de resultados para todos os exercícios. Em todas as situações que envolvem algum processo estocástico (pseudo-aleatoriedade ou aleatoriedade), é necessário: 1) armazenar a semente aleatória para repetir a mesma experiência exata, se necessário; 2) repetir a experiência 30 vezes com os mesmos parâmetros; 3) calcular a média e o desvio padrão dos 30 testes para cada quantidade me-

dida. Uma representação gráfica comum desses resultados é o *boxplot with whiskers* (`matplotlib.axes.Axes.boxplot` in Python).

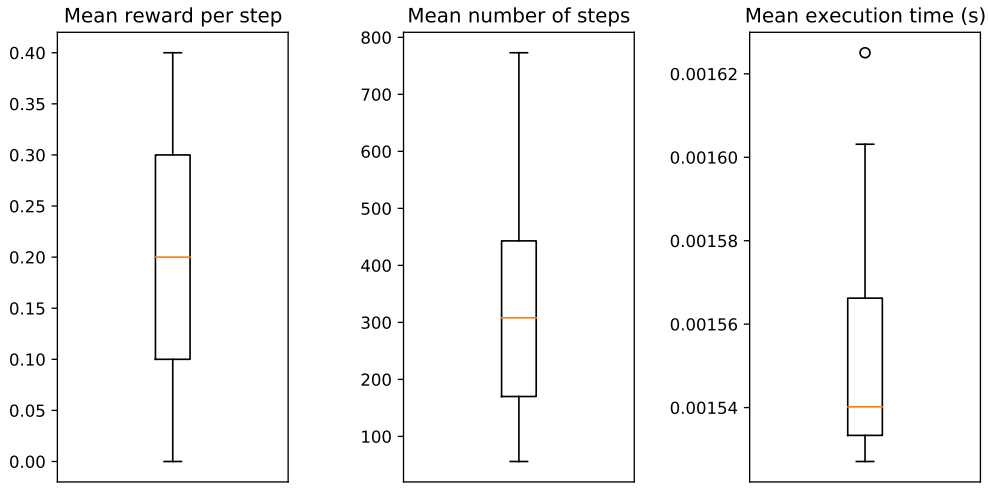

Figura 2: *Boxplots* mostrando a recompensa média por passo, o número médio de passos para atingir o objetivo e o tempo médio de execução para os 30 testes (máximo de 1000 passos cada).

Exercício 2

Crie uma matriz Q, indexada por estado e ação, $Q[s, a]$, e certifique-se de que é inicializada com zeros. Ao chegar a um estado s', atualize a utilidade do estado de onde o robot veio (s), usando a seguinte função de atualização (apresentada na aula sobre aprendizagem por reforço):

$$Q[s, a] = (1 - \alpha)Q[s, a] + \alpha\Big(r(s') + \gamma(\max_{a'} Q[s', a'])\Big) \quad (1)$$

onde $max_{a'}Q[s', a']$ é o melhor $Q[s', a']$ para todas as ações a' disponíveis no estado s' e $r(s')$ é a recompensa dada no estado s'. Use os seguintes valores para α e γ: $\alpha = 0,7$ e $\gamma = 0,99$.

Consegue dizer qual é a melhor ação a partir de qualquer estado dado? Compare os testes a) e b) e tire as suas conclusões.

a) Faça uma caminhada aleatória (*random walk*, como no exercício 1) e execute esta função de atualização após cada transição de estado para 20000 passos em cada experiência. Repita a experiência 30 vezes. Em cada uma das 30 experiências, aos passos 100, 200, 500, 600, 700, 800, 900, 1000, 2500, 5000, 7500, 10000, 12500, 15000, 17500 e 20000 (ou outros pontos intermédios que sejam considerados úteis), pare para executar um teste.

Um teste consiste em executar o sistema durante 1000 passos usando a tabela Q atual (sem a alterar) e escolhendo sempre a melhor ação a cada passo. Meça a recompensa média por passo nestes 1000 passos.

Meça também o tempo de execução de cada teste completo (todos os 20000 passos) e calcule a média e o desvio padrão dos tempos de execução para os testes. Trace um gráfico dos passos (eixo $\mathbf{x}$) versus a recompensa média (eixo $\mathbf{y}$) dos testes nos pontos medidos. Uma série de diagramas de caixas também pode ser usada para uma visão mais informativa da evolução do comportamento do robot.

Represente a utilidade final de cada estado (a qualidade da melhor ação para cada estado), representando o que o agente aprendeu sobre o ambiente, usando um mapa de calor (Figura 3).

Dica: Um mapa de calor é uma boa maneira de visualizar a informação na matriz Q. Se precisar de ver a política completa, o melhor processo é ter um mapa de calor da qualidade máxima para cada estado (`matplotlib.pyplot.imshow` em Python) e / ou uma matriz com a melhor ação para cada estado.

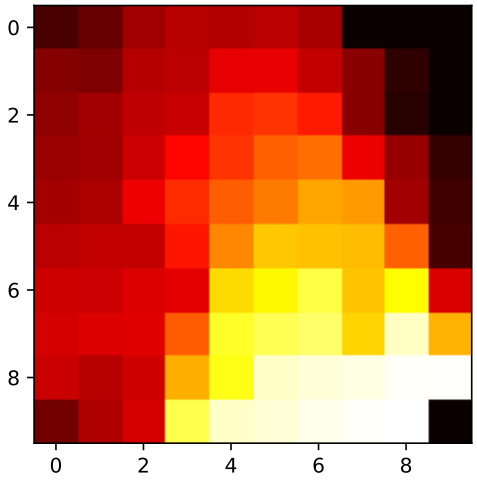

Figura 3: Mapa de calor da utilidade aprendida representando a utilidade máxima (utilidade da melhor ação) por estado (obtido com seed $= 10$, no ponto de teste 5000).

b) Faça o mesmo teste do exemplo anterior, mas em vez de uma caminhada aleatória, use sempre os valores da tabela Q para escolher a melhor ação. Tenha cuidado para fazer os desempates aleatoriamente.

Exercício 3

Use uma mistura das duas estratégias descritas acima: **inclua um termo (*greed*) na função de seleção** de ação que determinará a probabilidade de escolher uma ação aleatória. Por exemplo, se *greed* for 0,9, aproximadamente 10% das ações escolhidas devem ser aleatórias e os restantes 90% devem ser a melhor ação disponível de acordo com Q. Se *greed* for baixo, por exemplo, 0,2, aproximadamente 80% das ações são aleatórias. Experimente três parâmetros *greed* diferentes e compare os resultados. Finalmente, experimente um parâmetro *greed* crescente começando em 30%, para os primeiros 30% dos passos do teste, e aumentando lentamente até 100% no final do teste. Compare os resultados dos testes e as tabelas Q.

Exercício 4

Altere a simulação para incluir paredes (como na Figura 4) e que bater numa parede dê uma pequena recompensa de penalização (-0,1). Compare com os resultados anteriores.

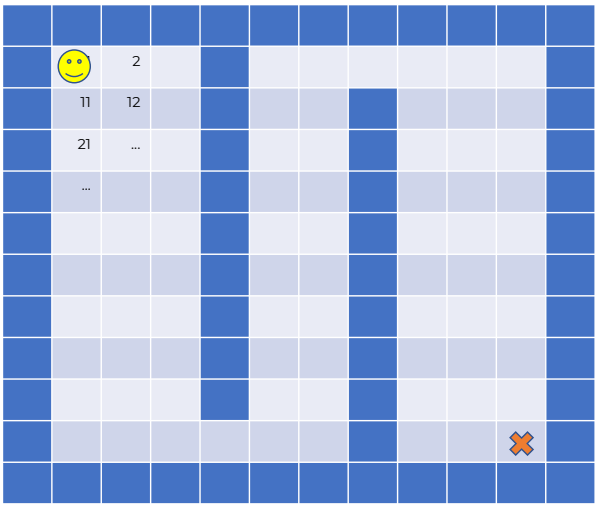

Figura 4: As paredes tornam o problema mais difícil. Use recompensas de penalização (pequenas em comparação com a recompensa final) para manter o agente no caminho certo.

Exercício 5 [Opcional]

Imagine que a mesma ação não leva sempre o robot para o mesmo estado **(sistema não-determinista)**. Com uma probabilidade de 5%, pode levar o robot a qualquer estado vizinho do estado atual. Como é que isso afeta o resultado?

Exercício 6 [Opcional]

Agora, imagine uma situação, mais próxima do cenário real, onde os estados não são numerados e **o agente só pode perceber a sua posição pelos ecos nas paredes**. A "percepção" do agente do que está à sua volta é uma matriz de valores de ponto flutuante que representam a distância à parede para cada lado UP, LEFT, DOWN, RIGHT, e.g., (NA, 0.56, NA, 0.14) significa: nenhuma parede encontrada acima, parede a 0,56 metros à esquerda, nenhuma parede abaixo, parede a 14 cm à direita. Como podem estes estados ser simplificados para um número que possa ser usado como índice? Quais são os riscos desta transformação num cenário como o da Figura 4 (pense nos estados da coluna 2 e 8, por exemplo)?