NXT MindStorms e Aprendizagem por Reforço

João Coelho m6416@alunos.uevora.pt
Teresa Gonçalves tcg@uevora.pt

Universidade de Évora

Resumo Este artigo analisa o comportamento de um robô implementado com um algoritmo de aprendizagem por reforço. A aprendizagem por reforço é uma técnica de aprendizagem por tentativa e erro onde o agente, através da interacção com o ambiente, aprende a realizar uma tarefa com base em recompensas positivas e negativas. Para avaliar o processo de aprendizagem foi construído um robô com o auxílio do kit de robótica Lego NXT MindStorms, cuja tarefa era seguir uma linha e implementado o algoritmo Q-learning. Foi realizada uma experiencia com o propósito de determinar quais os valores óptimos das três variáveis fundamentais do algoritmo Q-learning, sendo elas a taxa de aprendizagem, o factor de desconto e a taxa de exploração. Conclui-se então, que a taxa de aprendizagem e a de exploração necessitam de ser baixos e o factor de desconto necessita de um valor médio, para que o robô tenha um bom desempenho.

1 Introdução

Tem surgido um enorme interesse na utilização da aprendizagem por reforço na robótica, o que é bastante positivo para que haja evolução na investigação relacionada com este tema [1]. Com a aprendizagem por reforço implementada em robôs reais, a descrição de tarefas é definida através de uma função de recompensa, em vez de instruções específicas. Isto é, em vez de indicar ao robô a acção que deve tomar numa determinada situação, o robô, através da interacção com o ambiente, descobre que acções têm maior recompensa nessa situação. Para avaliar o comportamento de um método de aprendizagem por reforço foi o utilizado o robô NXT Mindstorms da Lego, que se encontra já na terceira geração [2]. Como algoritmo de aprendizagem foi utilizado o algoritmo Q-learning pois este não precisa de um modelo de ambiente[1], o que é indicado para este tipo de problema. Na (sub-secção 3.1), serão explicados sucintamente o modelo de ambiente assim como os elementos fundamentais na aprendizagem por reforço. A **secção 2** dedica-se à descrição dos componentes do robô educacional da Lego, Nxt Mindstoms, e da linguagem utilizada nesse robô. Na **secção 3** é apresentada alguns dos conceitos fundamentais da aprendizagem por reforço.Tais como, os elementos da aprendizagem por reforço e o algoritmo Q-learning. Na **secção 4** é explicado a parte experimental feita com o robô. É descrito o ambiente, o

[1] do inglês, *model-free*.

conjunto de acções e a função de recompensa. A **secção 5** mostra o resultado da experiência realizada com o robô, para determinar a sua performance em realizar a tarefa proposta. E por último na **secção 6**, a conclusão, é descrito as conclusões e também algum do trabalho futuro que podirá vir a ser realizado.

2 O Lego NXT Mindstorms

O kit básico inclui um bloco, três motores e quatro sensores: luz, ultra-som, som e toque, mas podem ser adquiridos outro tipo de acessórios. O bloco NXT, conhecido por *smart brick*, é o "cérebro" de qualquer robô lego mindstorms. É um bloco controlado por computador que da vida ao robô e que vai efectuar diferentes operações [13]. Este bloco contém quatro entradas na parte inferior e três na parte superior e uma entrada USB. Na parte superior do bloco, onde existem três entradas, são ligados os motores às portas A, B e C, que os controlam. Na parte inferior do bloco, onde existem quatro entradas, servem para ligar os sensores às portas de entrada 1, 2,3 e 4 que manipulam os quatro sensores (**ultra-som, toque, luz e som**). A Porta **USB** serve para conectar um cabo USB para que se possa fazer o *download* do programa para o bloco a partir do computador. Os **motores NXT** permitam ao robô moverem-se no ambiente em que se encontram. Estes motores têm a capacidade ajustar a velocidade, de pararem imediatamente ou irem parando ao longo do tempo. Estes podem ser utilizados em robôs móveis, em robôs humanóides e em máquinas que agarram um objecto. Estes sabem com precisão as voltas ou graus que o motor efectuou. Os quatro sensores base, no NXT Mindstorms que podem ser ligados a cada uma das quatro entradas, esses sensores são:

- O **sensor de som** permite medir o volume do som, ou seja, consegue determinar se o nível do som detectado é alto ou se é baixo;
- O **sensor de toque** tem a capacidade perceber se este está a ser pressionado ou não. Este dá ao robô a sensação de toque;
- O **sensor de infravermelhos** permite ao robô distinguir se está claro ou escuro, ou seja, consegue distinguir a luminosidade. Este consegue detectar, através de um LED[2] vermelho incorporado, os valores da luz reflectida, por outras palavras consegue detectar a intensidade das cores;
- O **sensor de ultra-som** tem a capacidade de medir a distância a que o robô se encontra de um obstáculo. Este sensor é utilizado para evitar a colisão de obstáculos ou interagir com as pessoas quando estas se encontram a uma determinada distância.

Para implementar este sistema de aprendizagem por reforço utilizou-se o lejOS NXJ [4], um projecto open-source que usa a *Java Virtual Machine* e fornece uma poderosa *API* que implementa várias funções de alto nível, como a navegação e comportamento robótico e as ferramentas necessárias para descarregar o código para o bloco NXT.

[2] *light-emitting diode* .

3 Aprendizagem por Reforço

A aprendizagem por reforço é um campo da aprendizagem automática, onde um agente através de tentativa e erro, tenta maximizar a recompensa que recebe ao interagir com o ambiente para realizar uma tarefa específica [11]. Ao agente não são ditas quais as acções que deve tomar numa determinada situação, mas ele deverá descobrir quais as acções que produzem maior recompensa através da experiência. Nas sub-secções seguintes apresentam-se os conceitos básicos da aprendizagem por reforço (sub-secção 3.1), os métodos de pesquisa que permitem o equilíbrio entre a exploração e examinação (sub-secção 3.2) e o algoritmo Q-learning (sub-secção 3.3).

3.1 Conceitos básicos

Nesta secção serão descritos alguns dos conceitos da aprendizagem por reforço. Tais como, a política, a funcão de recompensa, a função de valor e o modelo de ambiente. A **política** é uma regra estocástica que define o comportamento do agente. Esta analisa um estado analisado pelo agente, onde executa uma acção nesse estado que maximiza a satisfação dos seus objectivos. A política é representada por uma função π, que analise o estado, s, e acções, a, num valor $\pi(s, a)$ que corresponde à probabilidade do agente tomar a acção $a \in A(S)$ quando este se encontrar no estado $s \in S$. Por outras palavras, uma política é uma distribuição de probabilidades da acções a em cada estado s. A política é o núcleo de um agente implementado com um sistema de aprendizagem por reforço no sentido que sozinha é capaz de determinar o comportamento deste. A **função de recompensa** define o objectivo a atingir num sistema de aprendizagem por reforço. Esta diz ao agente qual é o seu objectivo, que consiste em maximizar as recompensas adquiridas durante sua interacção com o ambiente, explicitando ao agente quais foram às acções boas e as más desempenhadas por ele. O agente associa um estado (ou par estado-acção) do ambiente a um número, mais conhecido por recompensa. A **função de valor**, como já foi referido anteriormente, indica o ganho total que será acumulado no futuro ao iniciarmos um determinado estado. Todos os algoritmos de aprendizagem por reforço são baseados na estimativa de funções de valor, esta estima quão bom é a realizar uma determinada acção num determinado estado. Esta define em termos de recompensas futuras que podem ser esperadas. Existem dois tipos de função valor, a função estado-valor, $V^\pi(s)$, e a função acção-valor, $Q^\pi(s, a)$. Nos Processos de Decisão de Markov, a função $V^\pi(s)$, define-se por:

$$V^\pi(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}, \tag{1}$$

onde $V^\pi(s)$ representa o valor esperado que o agente vai receber quando seguir a política π, num determinado instante t. $V^\pi(s)$ é o somatório das recompensas.

função acção-valor, $Q^{\pi}(s,a)$, considerando o par estado-acção, define-se por:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}, \quad (2)$$

onde a representa uma acção, no estado s sobre a política π, num determinado instante t. A função $Q^{\pi}(s,a)$ calcula as recompensas esperadas começando em s.
Estas duas funções, $V^{\pi}(s)$ e $Q^{\pi}(s,a)$, podem ser estimadas por experiência. Isto é, um agente segue a política π e mantém a média das recompensas actuais, que acompanham um estado. Depois essa média vai convergir para $V^{\pi}(s)$. Se as médias separadas forem mantidas para cada acção, executada em cada estado, então vão convergir para $Q^{\pi}(s,a)$. O **modelo do ambiente** é algo que imita o comportamento do ambiente, onde existe interacção entre o agente e o ambiente. Com isto, o agente que se encontra num estado, ao realizar uma determinada acção consegue prever qual será o resultado do próximo estado e da recompensa. Existem vários métodos que não utilizam este modelo, mais conhecidos por métodos livres de modelo do ambiente , apesar destes métodos serem mais simples que os outros, os que precisam de modelo de ambiente, demoram mais tempo até este chegar ao estado óptimo.

3.2 Métodos de pesquisa

Um dos desafios surgidos na aprendizagem por reforço é a existência de conflitos entre a exploração[3] e a examinação[4]. Na **examinação** a escolha baseia-se na informação que o agente dispõe no momento, ou seja, para uma determinada situação o agente escolhe a acção com maior recompensa. À primeira vista parece ser uma boa opção mas ela impede o agente a procura de melhores acções para um determinado estado. Na **exploração** a escolha permite obter novas informações sobre o ambiente com o objectivo de alcançar níveis de desempenho maiores no futuro, ou seja, o agente explora acções ainda não executadas ou estados não visitados, para ter uma visão mais abrangente do ambiente onde está inserido sempre com o objectivo de maximizar o seu desempenho. Uma das grandes diferenças entre aprendizagem por reforço e aprendizagem supervisionada, dá-se ao simples facto do agente da aprendizagem por reforço ter que, explicitamente, explorar o ambiente em que se encontra [8]. Existem vários métodos que permitem o equilíbrio entre a examinação e a exploração, sendo dois dos mais conhecidos os métodos ϵ-*Greedy* e o *Softmax*. O método ϵ-*Greedy* na maioria das vezes escolhe a acção com maior recompensa, mas de vez em quando, a acção é escolhida aleatoriamente independentemente da estimativa acção-valor; a melhor acção conhecida é escolhida com a probabilidade de $1-\epsilon$, e a acção aleatória é escolhida com a probabilidade ϵ, ou seja, para acção a^*, a política é

[3] Do inglês, *exploration*.
[4] Do inglês, *exploitation*.

$\pi_t(s, a^*) = 1 - \epsilon$. No final, são feitas tentatitvas suficientes, assegurando assim serem descobertas as acções óptimas. O método *Softmax* foi desenvolvido para ultrapassar esta questão atribuindo um peso para cada acção, de acordo com a sua estimativa acção-valor [12], ou seja, uma acção é seleccionada de acordo com o peso que lhe está associado. Desta forma, é muito improvável a pior acção ser escolhida e a acção com maior valor de recompensa, tem maior probabilidade de ser executada. Para calcular as probabilidades é utilizada a distribuição de Gibbs e a política é dada pela equação

$$\pi_t(a) = \frac{e^{\frac{Q_t(a)}{T}}}{\sum_{b=1}^{n} e^{\frac{Q_t(a)}{T}}} \tag{3}$$

onde T é um valor positivo designado temperatura.

3.3 Q-learning

O Q-learning desenvolvido por Watkins em 1989 [7] é considerado um dos maiores avanços na área da aprendizagem por reforço. Trata-se de um algoritmo que não necessita de um modelo do ambiente[5], ou seja, aprende a política óptima sem criar um modelo do ambiente. Neste método, a função valor é ampliada para se tornar uma função acção-valor, que armazena o valor de todas acções em cada estado. A função acção-valor $Q^*(s, a)$ expressa o valor da acção a realizada no estado s num comportamento óptimo. A relação entre a função valor e a função acção-valor é:

$$V^*(s) = \max_a Q^*(s, a) \tag{4}$$

A implementação do algoritmo é simples pois esta função acção-valor é a única coisa que é preciso armazenar e actualizar. A política é definida de acordo com a equação

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{5}$$

Cada experiência é descrita por um tuplo (s, a, r, s'). A actualização da função acção-valor é dada pela equação (está certa a equação?)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t) \right] \tag{6}$$

onde Q é a tabela de valores dos pares estado-acção, a_t é a acção actual, s_t o estado actual, s_{t+1} é o novo estado que resulta da acção no estado actual, $\alpha \in]0, 1[$ corresponde à taxa de aprendizagem, $\gamma \in]0, 1[$ é factor de desconto, $\max_a Q(s_{t+1}, a_t)$ é a acção com maior recompensa da tabela Q e a r_{t+1} é a recompensa que o agente recebe no próximo estado. Este método é bastante conhecido por ser um método pioneiro que verdadeiramente estuda o a aprendizagem por reforço com o propósito do controlo, com uma forte prova da sua convergência [10].

[5] Do inglês, *model-free*.

4 Tarefa de seguir a linha

Neste secção será descrito

4.1 Ambiente

O ambiente é composto por uma superfície iluminada, neste caso uma cartolina branca, onde está desenhado o percurso que o robô deve seguir. Em seguida podemos verificar com a ajuda da tabela 1, os estados possíveis. Em seguida vão

Estado	**Sensor nº 1**	**Sensor nº 2**
A	0	0
B	0	1
C	1	0
D	1	1

Tabela 1: Estados possíveis

ser descritos para uma melhor compreensão:

- Estado um, quando os dois sensores não estão na linha a ser seguida;
- Estado dois, quando tem o sensor direito na linha e o sensor esquerdo não está na linha;
- Estado três, quando não tem o sensor direito na linha e o sensor esquerdo está na linha;
- E por ultimo o estado quatro, quando os dois sensores de infravermelhos estão na linha.

4.2 Conjunto de Acções

Foram definidas quatro acções, andar para a frente, andar para trás, virar à direita e, por último, virar à esquerda. A tabela seguinte mostra as acções que o robô pode realizar em cada estado.

4.3 Função de recompensa

Esta função é a maneira mais natural de especificar qual é a tarefa que o robô tem de realizar, ou seja, qual é o objectivo que o robô tem de alcançar. É mais fácil de especificar ao agente que tem um comportamento bom ou mau num determinado estado do que especificar que acção deve ser realizada num dada

Acção	Motor A	Motor B
Andar para trás	roda para trás	roda para trás
Virar à direita	roda para trás	roda para a frente
Virar à esquerda	roda para a frente	roda para trás
Andar para a frente	roda para a frente	roda para a frente

Tabela 2: Acções possíveis

situação ou estado.
Quando o robô está com os dois sensores na linha recebe uma recompensa positiva pois este é o estado óptimo, por outras palavras, é este comportamento que o robô deve ter. Quando o robô está com os dois sensores fora da linha o robô recebe uma recompensa negativa alta. No caso de estar um sensor fora da linha e outro na linha o robô é punido por não ter os dois sensores na linha, mas esta punição não é tão elevada como a anterior.

5 Resultados e Conclusões

5.1 Configuração de variáveis

Neste experiencia vão ser alterados os valores da taxa de aprendizagem[6](α) , do factor de desconto(γ)[7] e da taxa de exploração [8](ϵ) para determinar que combinação, das três variáveis principais, será mais viável para este tipo de tarefa, isto é, verificar quais das combinações dá um maior desempenho ao robô. Esta experiência consiste em verificar o número de iterações (passos) que o agente executa até conseguir seguir uma linha. Foi feito uma média de vinte experiências para cada combinação. Os valores podem ser caracterizados por valor alto (0,9), por valor médio (0,5) e por valor baixo (0,3). A seguinte tabela mostra as vinte e sete combinações possíveis, das três variáveis, com a média das iterações que o robô fez para aprender a seguir uma linha:

[6] Do inglês, *learning rate*.
[7] Do inglês, *discount factor*.
[8] Do inglês, *exploration rate*.

α	γ	ϵ	**Média**	**Desvio Padrão**
0,9	0,9	0,9	300	24
0,9	0,9	0,5	142	17
0,9	0,9	0,3	131	31
0,9	0,5	0,9	330	48
0,9	0,5	0,5	141	21
0,9	0,5	0,3	106	20
0,9	0,3	0,9	215	21
0,9	0,3	0,5	123	20
0,9	0,3	0,3	113	21
0,5	0,9	0,9	300	33
0,5	0,9	0,5	136	16
0,5	0,9	0,3	147	37
0,5	0,5	0,9	389	28
0,5	0,5	0,5	149	15
0,5	0,5	0,3	101	18
0,5	0,3	0,9	482	13
0,5	0,3	0,5	144	16
0,5	0,3	0,3	102	11
0,3	0,9	0,9	589	11
0,3	0,9	0,5	130	14
0,3	0,9	0,3	157	20
0,3	0,5	0,9	357	14
0,3	0,5	0,5	117	18
0,3	0,5	0,3	**85**	12
0,3	0,3	0,9	304	12
0,3	0,3	0,5	114	18
0,3	0,3	0,3	111	11

Tabela 3: Média e Desvio Padrão dos valores registados para cada combinação de variáveis

Através da tabela 3, pode-se concluir que o robô tem um melhor desempenho quando os valores da taxa de aprendizagem e da exploração são baixos, e o valor do factor de desconto for médio. O robô com estes valores consegue aprender a seguir uma linha com uma média de 85 iterações. Quando a taxa de exploração apresentar um valor alto, o desempenho do robô é afectado. A média das iterações para o robô aprender a tarefa, quando a exploração tem valores altos, é de 363.

6 Conclusões

Um robô implementado com um sistema de aprendizagem por reforço consegue aprender uma determinada tarefa em poucas iterações (passos). Através da experiencia, concluiu-se que o robô tenha um melhor desempenho os valores da taxa de aprendizagem e da exploração têm de ser baixos e o valor do factor de desconto tem de ser médio e a melhor localização dos sensores para qualquer

percurso, é lado a lado na parte frontal do robô Como trabalho futuro construir outro robô com um propósito diferente do robô apresentado neste artigo, um robô que tenha um estado final. Por exemplo, um robô com o objectivo de seguir uma fonte de luz, ou som.

Referências

1. Jeremy Wyatt, Issues in Putting Reinforcement Learning Onto Robots, Mobile Robotics Workshop, 10th Biennial Conference of the AISB. 1995
2. Juan Antonio Breña Moral, Develop leJOS programs Step by Step. 2009
3. Sutton, Richard S. and Barto, Andrew G, Reinforcement Learning: An Introduction, MIT Press, 1998
4. Juan Antonio Breña Moral, Multithreading with Java leJOS, 2008
5. , Jeremy Wyatt, Reinforcement Learning: a brief overview, Perspectives on adaptivity and learning, 243–264, 2002
6. Peter Dayan and Christopher Watkins, Reinforcement Learning, 2001
7. C. J. C. H. Watkins, Learning from Delayed Rewards, Cambridge University, 1989
8. L.P. Kaelbling and M.L. Littman and Andrew Moore, Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, 4, 237-285, 1996
9. Lin, Long-Ji and Mitchell, Tom M., Reinforcement learning with hidden states, Proceedings of the second international conference on From animals to animats 2 : simulation of adaptive behavior: simulation of adaptive behavior, 271–280, Cambridge, MA, USA, MIT Press, 0-262-63149-0, Honolulu, Hawai, United States, 10, http://dl.acm.org/citation.cfm?id=171174.171206, 1993
10. Tommi Jaakkola and Michael I. Jordan and Satinder P. Singh, Convergence of Stochastic Iterative Dynamic Programming Algorithms, Neural Computation, 6, 1185–1201, 1994
11. Haykin, Simon, Neural Networks: A Comprehensive Foundation Second Edition, 1998
12. Viviane Margarida Gomes, Controle Inteligente de Tempo Livre em Tutoria Multissessão, Universidade Federal de Goiás, 2009
13. Octávia Raquel Gomes Figueira, DROIDE M.L.P Potencializando a Plataforma, Universidade da Madeira, 2008