Sistemas Inteligentes - Computação Natural

1. Implementação

Na resolução desta ficha foi implementado tanto o algoritmo Q-Learning como o SARSA para o problema do Grid World:

- **Grid world:** foi criada uma class com os métodos que permitem definir/atualizar as recompensas, ações e estados, para além de permitir a movimentação do agente e ainda definir todos os estados possíveis no jogo bem como os locais com recompensas. Tendo em conta que se pretende minimizar o número de movimentos, cada movimento terá um custo associado.
- **Agente com Q-Learning:** neste algoritmo utilizei a grid em que o movimento tem custos associados, sendo que comecei por instanciar esta grid e inicializar o Q(s,a). De seguida, enquanto não convergir e não terminar o jogo escolhemos uma posição aleatória através do *Epsilon-greedy*. A diferença entre este algoritmo e o seguinte é que neste calculamos o max[a']{Q(s',a')} para fazer a atualização. Assim, podemos calcular o Q(s,a) e verificar qual foi a alteração que aconteceu.
- **Agente com SARSA:** este algoritmos teve uma implementação muito semelhante ao anterior, sendo que neste não calculamos o max[a']{Q(s',a')}

2. Resultados

Assim, obtemos que os rewards em ambos são:

```
-------------------------
|-0.10|-0.10|-0.10| 1.00|
-------------------------
|-0.10| 0.00|-0.10|-1.00|
-------------------------
|-0.10|-0.10|-0.10|-0.10|
-------------------------
```

E podemos observar nos seguintes gráficos, que o primeiro algoritmo convergiu muito mais depressa, em pouco mais das 2000 iterações enquanto que o segundo não chegou a convergir totalmente em 10000 interações.

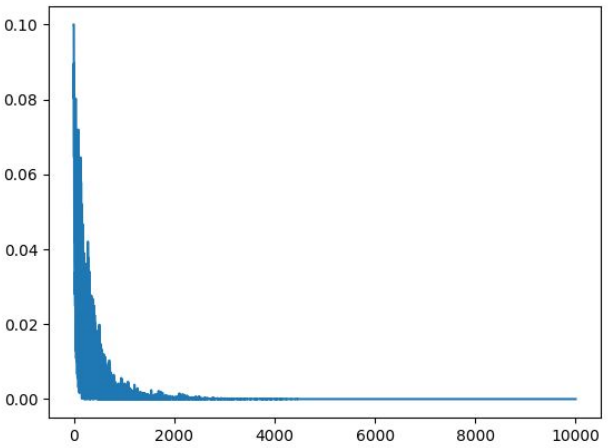

Ilustração 1 - Convergência em função do número de iterações: Q-Learning

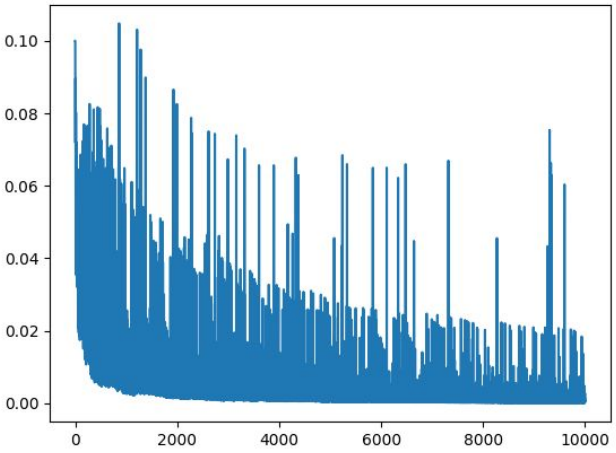

Ilustração 2 - Convergência em função do número de iterações: SARSA

3. Dificuldades

Este foi um projeto que deu bastante trabalho de implementação e em grande parte de investigação, sendo que foi bastante o tempo dispendido neste, principalmente relativamente como deveria abordar a implementação do algoritmo em termos dos valores necessários para as fórmulas, como por exemplo os valores do epsilon e do alpha que se iam adaptando. Esta foi a parte de acabou por levar mais tempo uma vez que fui melhorando os algoritmos percebendo os erros através dos resultados que eram obtidos.