Path Planning and Path Covering using Q Learning Algorithm

Aluno: Daniel Werneck Rodrigues

Orientador: Evelyn Batista

Trabalho apresentado ao curso BI MASTER como pré-requisito para conclusão de curso e obtenção de crédito na disciplina "Projetos de Sistemas Inteligentes de Apoio à Decisão".

- Projeto Completo
- Projeto.pdf

Resumo

As calamidades sempre ocorreram no mundo e, muitas vezes, a dificuldade na identificação dos sinais que anunciam as suas ocorrências dificultam também, a adoção de medidas de precaução. Sendo assim, é desejável que se busque o melhor comportamento e planejamento diante de um desastre, de forma a maximizar a quantidade de sobreviventes eventualmente existentes e minimizar o tempo de exposição ao perigo ao qual as vítimas poderão estar submetidas.

O objetivo deste trabalho é otimizar o planejamento do resgate de vítimas, concluindo a busca com o menor tempo (menor quantidade de passos), dado um cenário em que conhecemos as prioridades de cada subespaço em nosso espaço de busca. Além do principal objetivo e da aplicabilidade direta na área de calamidades, tem-se como um dos objetivos desta pesquisa a capacidade de generalização, ou seja, que o trabalho aqui apresentado possa ser aplicado a outros problemas com características semelhantes. Para resolução deste problema, será utilizada o método Q Learning na área de Reinforcement Learning para a otimização da rota a ser percorrida pelo agente.

Abstract

Calamities have always occurred in the world and, many times, the improbability of their occurrence makes them difficult to predict and prevent. Therefore, the best behavior and planning in the face of a disaster is sought, in order to maximize the number of survivors and minimize the time of exposure to the danger to which the victims are being subjected.

The objective of this work is to optimize victim rescue planning, completing the search in the shortest time (fewer number of steps), given a scenario in which we know the priorities of each subspace in our search space. In addition to the main objective and direct applicability in the area of calamities, one of the objectives of this research is the ability to generalize, that is, that the work presented here can be applied to other problems with similar characteristics. To solve this problem, the Q Learning method will be used in the Reinforcement Learning area to optimize the route to be taken by the agent.

1. Introdução

Em Reinforcement Learning, o aprendizado ocorre através da interação de um agente com o ambiente, recebendo recompensas ou punições. Após n jogos jogados, temos um agente especializado no problema em questão que aprende através da maximização total das recompensas recebidas ao longo de cada jogo. Quase todos os problemas de Reinforcement Learning usa o framework matemático de Markov Decision Process (MDP), no qual estados futuros dependem apenas do estado presente.

A programação dinâmica é outro conceito importante para problemas de Reinforcement Learning, que se consiste em uma técnica para resolução de problemas complexos através da separação desse problema em subproblemas e do aproveitamento de soluções previamente estabelecidas para poupar esforço computacional, o que minimiza o tempo de processamento.

Ainda, para este trabalho será utilizada a abordagem de Diferença Temporal, que não requer conhecimento prévio a respeito da dinâmica do modelo. O algoritmo de aprendizado TD foi introduzido por Sutton em 1988. O algoritmo traz os benefícios do método de Monte Carlo e da programação dinâmica (DP). Assim como o método de Monte Carlo, não requer dinâmica de modelo e, assim como DP não requer o tempo de espera até o final do episódio para fazer uma estimativa da função de valor. Em vez disso, é realizada uma aproximação da estimativa atual com base na estimativa aprendida anteriormente.

Será utilizado o algorítimo de Q Learning, que se trata de algoritmo simples e muito popular em Temporal Difference Learning. Esse algoritmo tem como objetivo a atualização do Q Value(Q(s,a)), a cada passo do agente no ambiente, e ao final, encontrar os pares de estado ação que levarão a maximização da recompensa final.

2. Problema e Premissas

Cada subespaço do nosso espaço de busca será subdividido em quadrantes, onde cada quadrante terá uma prioridade pré-definida, sendo posteriormente utilizada como recompensa para o treinamento do agente. Essa prioridade pode ser estabelecida de acordo com: maior chance de sobrevivência das vítimas contidas no quadrante, maior quantidade de vítimas no quadrante, ou qualquer outra forma de priorização que se julgue adequada para o problema em questão. Para este trabalho, será considerada uma matriz de prioridades onde temos um epicentro da catástrofe com maior priorização e, quanto mais afastado deste epicentro, menores serão as prioridades (exemplo: queda de avião num ponto definido, em que suas áreas adjacentes, ou seja, quanto mais longe da aeronave, menores chances de localizarem-se vítimas).

Além disso, também serão pré-definidos obstáculos no espaço de busca que estarão contidos nas matrizes de prioridades e marcadas com valor -100.

O problema será resolvido em etapas, da mais simples para a mais complexa. Sendo assim serão considerados os cenários

```
i.  ‘matriz muito simples’ (matriz 4x3);
ii. ‘matriz simples’ (5x5);
iii.    ‘matriz média’ (8x8);
iv. ‘matriz complexa’ (9x13).
```

3. Modelagem

Para resolução do problema foi desenvolvido um ambiente com a definição de duas classes. A primeira de “estado”, que define os atributos de estado percorrido pelo agente, e “agente”, com os atributos e os métodos pertinentes, como o de execução de cada jogo e a atualização da tabela Q por cada passo (round). O objetivo de cada jogo é, sempre iniciando do estado (0,0), com a menor quantidade de passos, percorrer todos os quadrantes com valores maiores que zero, privilegiando os subespaços com maiores valores de prioridade e por fim, maximizando a recompensa total.

Dos principais hiperparâmetros do problema podemos destacar o Épsilon e o d_f. A ação é escolhida conforme épsilon atual, que segue função que retorna a ação com melhores recompensas para casos de épsilon mais próximos a zero (exploitation) e ações aleatórias para casos de épsilon mais próximos a um (exploration). Nos primeiros episódios, jogados têm maiores épsilons e nos últimos, menores, de forma a, inicialmente, explorar as possíveis soluções e mais tarde se especializar nas melhores soluções obtidas. Sendo assim, o fator de decaimento da taxa de exploração (ε_decay) é um dos hiperparâmetros importantes para o problema. Já a variável de deflação temporal (d_f), desenvolvida para este trabalho, não faz parte da modelagem clássica. Ela é responsável por reduzir as recompensas futuras de acordo com a quantidade de passos dados, atuando como “fator de urgência”, que deteriora a prioridade (recompensa) a cada passo do agente no sistema, simulando assim, a passagem do tempo de busca.

Como para cada cenário avaliado temos diferente quantidade de passos médios para conclusão do problema, se faz necessário identificar o número ideal da variável deflatora (d_f) para cada cenário. Inicialmente foi identificando, para cada cenário, o d_f ideal dentro do seguinte espaço de busca.

Cada configuração a seguir teve o modelo rodado 10 vezes para obtenção de resultados:

```
i.  ‘matriz muito simples’ (matriz 4x3) -> [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999];
ii. ‘matriz simples’ (5x5) -> [0.8, 0.9, 0.95, 0.99, 0.999, 0.9999];
iii.    ‘matriz média’ (8x8) -> [0.99, 0.999, 0.9999];
```

Como a recompensa final é maior quanto maior for o d_futilizado, essa não foi utilizada como métrica principal para escolha do d_f. Ao invés disso, foi selecionado o d_f que produziu menor quantidade de passos, solução mais estável, com convergência mais rápida para a melhor solução encontrada e com tempo de execução aceitável. Foi notado que menores d_fs aumentam consideravelmente o tempo de execução para cenários de maiores complexidades, sendo assim, este ponto também foi levado em consideração para avaliação e seleção de melhor d_f.

Posteriormente foi realizada uma busca de melhores hiperparâmetros para cada cenário, sendo importante destacar que, para o cenário 13x9 não foi realizada busca de melhor d_f e de hiperparâmetros e, ao invés disso, utilizou-se os valores ótimos avaliados no cenário 8x8. Dessa forma, é avaliado como seria realizado o planejamento em um cenário mais realista e complexo com melhores parâmetros escolhidos a partir de cenário de menor complexidade.

A busca por hiperparâmetros se deu da seguinte forma, em 8.000 episódios e com a mesma configuração executada 10 vezes:

i. ‘matriz muito simples’ (4x3):

```
o   Learning rate = [0.01, 0.001, 0.0001];
o   Gamma = [0.95, 0.99];
o   d_f = [0.95];
o   exp_rate decay (epsilon) = [0.001, 0.0005];
o   quantidade maxima de passos = [5000]
```

ii. ‘matriz simples’ (5x5):

```
o   Learning rate = [0.01]
o   Gamma = [0.95, 0.99]
o   d_f = [0.8]
o   exp_rate decay (epsilon) = [0.001, 0.0005]
o   quantidade maxima de passos = [10000]
```

iii. ‘matriz média’ (8x8):

```
o   Learning rate = [0.01]
o   Gamma = [0.97, 0.99]
o   d_f = [0.99]
o   exp_rate decay (epsilon) = [0.001, 0.0005]
o   quantidade maxima de passos = [10000]
```

4. Resultados

Os resultados foram obtidos com sucesso de acordo com a seleção das configurações e do instante (round) que produziu melhores recompensas totais, sendo esses:

i. Resultado Matriz Muito Simples (4x3)

Configuração 8 (lr=0.001; gamma=0.99; epsilon_decay_rate= 0.001) foi selecionada pois possui maior média geral de Total Rewards e menor média geral de quantidade de passos. Acima de 6000 episódios, possui solução estável e, aparentemente, possui convergência mais rápida que em outras configurações.

ii. Resultado Matriz Simples (5x5)

Configuração 2 (lr=0.01; gamma=0.95; epsilon_decay_rate= 0.001) foi selecionada pois possui maior média geral de Total Rewards e menor média geral de quantidade de passos. Acima de 6000 episódios, possui solução estável e, aparentemente, possui convergência mais rápida que em outras configurações.

iii. Resultado Matriz Média (8x8)

A configuração 3 foi selecionada pois possui maior média geral de Total Rewards e menor média geral de quantidade de passos. Aparentemente, possui convergência mais rápida que em outras configurações, apesar de que acima de 4000 episódios, resultados começarem a apresentar ruídos.

É importante notar que, para as configurações com taxa de decaimento da taxa de exploração de 0.001 e 0.0005, o resultado de Total Rewards começa a se deteriorar a partir de aproximadamente 2800 e 4500 episódios, respectivamente, sugerindo que modelo pode se beneficiar de maiores quantidade de episódios submetidos a maiores taxas de exploração.

No entanto, foi obtido resultado aceitável (sub-ótimo ou ótimo, pois não há garantia de máximo global) referente à configuração selecionada com quantidade de Timesteps de 29 e Total Reward de 13.37. Esse resultado foi obtido selecionando a instância com o maior Total Reward e menor quantidade de Timesteps obtida no treinamento. Dessa forma, a deterioração da convergência teve pouco impacto no resultado final.

iv. Resultado Matriz Complexa (13x9)

Foi percebida uma queda em Total Rewards e um aumento de Timesteps a partir de aproximadamente 2800 episódios. De maneira semelhante ao que ocorreu no cenário 8x8, que também teve piores resultados a partir de determinado limiar de episódios.

No entanto, ao contrário do resultado obtido no cenário 8x8, não foi percebido um resultado sub-ótimo aceitável, com a quantidade de Timesteps do agente no ambiente de 1013 e Total Reward máximo percebido de 28.96.

4. Conclusões

Os resultados foram obtidos com sucesso, para os cenários apresentados. Para futuros trabalhos é importante à utilização de outras técnicas de Reinfocerment Learning e de otimização para comparação com os resultados obtidos neste trabalho.

Foi percebido que para cada cenário há diferença nos melhores hiperparâmetros utilizados, o que demanda tempo e esforço computacional. Apesar disso, as diferenças nos resultados de recompensas totais foram sutis de configuração para configuração, sendo percebidos bons resultados em hiperparâmetros subótimos em relação aos escolhidos.

Para configuração da matriz de média complexidade (8x8) e alta complexidade (9x13) houve problemas na convergência do modelo, com a deterioração da recompensa total para episódios com menores taxas de exploração. Como sugestão para futuros trabalhos, é pertinente a utilização de outras curvas para redução do épsilon de maneira a aumentar a quantidade de episódios submetidos a uma maior taxa de exploração de forma a se obter maior convergência nesses cenários. Também pode ser interessante a definição de um valor mínimo para a taxa de exploração, de maneira que agente continue com chances de explorar mesmo em episódios mais avançados.

Também foi identificado que para cenários mais complexos, tem-se aumento substancial do tempo de processamento e maior dificuldade de convergência do modelo. Além disso, para estes cenários a redução no valor dos hiperparâmetros está, normalmente, associada com maiores tempos de processamento e esforço computacional. Essas características tornam difícil a utilização do trabalho proposto em cenários reais. Para futuros desenvolvimentos, verificar se treinamento de algoritmo de Deep Reinforcement Learning, pode ser

aplicado de forma a mitigar esses problemas de aplicação e melhorar desempenho do modelo. Além disso, pode ser pertinente à otimização do código python visando tornar o esforço computacional menor na execução do modelo e na busca por melhores hiperparâmetros e consequentemente menor tempo.

Ademais, de forma complementar ao trabalho aqui apresentado, para a obtenção da matriz de prioridades, além da possibilidade de a obtermos de acordo com a análise de um especialista, pode ser possível, mediante a testes, à utilização de modelo de Redes Neurais Convolucionais para obtê-la a partir de imagens de satélite de forma a classificar determinado quadrante em relação a sua prioridade.

Matrícula: 211.101.278

Pontifícia Universidade Católica do Rio de Janeiro

Curso de Pós Graduação *Business Intelligence Master*