Relatório - Minimax (Connect-4)

Alunos:
- Caio Cargnin Cardoso
- Diego Marzarotto

Implementação

```
import numpy as np
from connect4 import Jogo, Tabuleiro, Minimax, Avaliacao
```

O programa consiste em um script python, que utiliza 4 classes para implementar o algoritmo minimax, com o objetivo de criar um jogador artificial de Lig-4 (Connect-4).

Além das bibliotecas nativas do python, o script usa também o *numpy*, portanto, antes de rodar o script é necessário instalar a biblioteca. Caso esteja usado o *pip* para gerenciar as dependências, basta rodar:

```
sudo pip install -r requirements.txt
```

Caso contrário, instale o pip antes. No Ubuntu basta executar:

```
sudo apt-get install python-pip
```

Para instruções em outros sistemas operacionais, consulte o google. :)
Para utilizar o script, ele deve ser executado em um terminal.

```
./connect4.py
```

Ao ser executado, o script apresenta a seguinte saída para o usuário, que deve então entrar com uma jogada válida. Em seguida as jogadas são alternadas entre o jogador humano e o adversário.

```
Jogo(demo=True).iniciar() # internamente o script executa isso, com demo=False
```

```
### Connect-4 ###

1| _ _ _ _ _ _ _ |
2| _ _ _ _ _ _ _ |
3| _ _ _ _ _ _ _ |
4| _ _ _ _ _ _ _ |
5| _ _ _ _ _ _ _ |
6| _ _ _ _ _ _ _ |
   1 2 3 4 5 6 7
```

```
Em qual coluna você deseja realizar sua jogada? 4

1| _ _ _ _ _ _ _ |
2| _ _ _ _ _ _ _ |
3| _ _ _ _ _ _ _ |
4| _ _ _ _ _ _ _ |
5| _ _ _ x _ _ _ |
6| _ _ _ o _ _ _ |
   1 2 3 4 5 6 7
```

O número de níveis que o algoritmo percorrerá para realizar a busca pode ser passado como parâmetro:

```
./connect4.py -n 1
./connect4.py -n 2
./connect4.py -n 3
```

A seguir, uma breve descrição das classes existentes:

Classe Jogo

Responsável por realizar a interação com o usuário e controlar o fluxo da partida.

Interage com a classe Tabuleiro para armazenar o estado do jogo, com a classe Avaliacao para interpretar o valor de cada estado do tabuleiro, e com a classe Minimax para realizar a busca das melhores jogadas.

Classe Minimax

Responsável por realizar a busca jogadas do jogador artificial.

O algoritmo utilizado é o Minimax com poda alfa-beta, que utiliza recursão para percorrer uma árvore de busca à procura da melhor jogada.

Durante a busca, a classe Minimax divide o valor final da avaliação pelo número de jogadas já realizadas. Divide-se o valor ponderado das ameaças pelo número de níveis percorridos até o atual estado na árvore de busca, com +1 para evitar a divisão por zero no nível raíz da árvore.

n - número de níveis percorridos na árvore de busca

$f(A_f, A_c) = \frac{A_f - A_c}{n+1}$

Classe Avaliacao

Responsável por implementar a função que calcula a utilidade e a heurística dos diferentes estados do jogo.

A função de avaliação realiza uma contagem das ameaças existentes para ambos os jogadores, calcula a diferença e utiliza pesos diferentes dependendo dos tipos de ameaças.

A - ameaças
A_1, A_2, A_3, A_4 - ameaças com 1, 2, 3 e 4 peças, respectivamente

A_f - valor calculado das ameaças a favor
A_c - valor calculado das ameaças contra

Ameaças são conjuntos de 4 posições adjacentes do tabuleiro (horizontal, vertical, ou diagonalmente), em que existam somente peças de um dos jogadores. Ameaças diretas são aquelas que podem ser realizadas sem que seja necessário realizar outras jogadas adicionais para preencher os espaços vazios abaixo da peça que está faltando na ameaça. Ameaças indiretas não podem ser realizadas sem que antes todas as posições abaixo da peça faltando na ameaça sejam ocupadas.

A implementação apresentada não diferencia as ameaças diretas das ameaças indiretas, sendo essa a sua principal limitação.

Os pesos utilizados para cada ameaças são os sequintes:

- (Ameaça com 1) $W_1 = 1$
- (Ameaça com 2) $W_2 = 50$
- (Ameaça com 3) $W_3 = 5000$
- (Ameaça com 4) $W_4 = 500000$

$A_f = \sum_{i=1}^{4} W_i A_{f,i}$

$A_c = \sum_{i=1}^{4} W_i A_{c,i}$

$f(A_f, A_c) = A_f - A_c$

Para fazer a contagem, foi utilizado um filtro (matriz) de 4x4 que percorre todo o tabuleiro e conta as ameaças verticais, horizontais e diagonais.

Além disso, a função é também responsável por identificar se o jogo está terminado e qual jogador saiu vitorioso.

Classe Tabuleiro

Responsável por armazenar o estado do tabuleiro e validar as jogodas possíveis de serem realizadas.

Limitações

O maior número de níveis que foi possível percorrer nos exemplos avaliados, foi um máximo de 6 níveis. A complexidade envolvida em percorrer o tabuleiro com o filtro inviabiliza o algoritmo de ser utilizado eficientemente em problemas de tempo real.

Outro problema encontrado com a implementação tem relação com o fato do número de níveis a serem percorridos ser par ou ímpar. Quando este número é ímpar, a última jogada avaliada é a jogada do jogador humano, fazendo com que uma ameaça com 3 para o jogador humano, que pode ser concluída já na sua próxima jogada não seja devidamente identificada como uma condição de vitória iminente.

Para "resolver" esse caso, identificamos as ameaças no último nível, caso este seja ímpar, como uma ameaça maior. Por exemplo, uma ameaça com 3 passa a ser identificada como uma ameaça com 4, e assim para todas as outros tipos de ameaças, exceto ameaças com 4, que continuam com a mesma avaliação.
