Morpion-IA

October 15, 2020

1 Artificial Intelligence Tic Tac Toe

This code is writen by Yan PODOLAK and myself

1.1 The purpose of this project is to create an unbeatable AI at Tic Tac Toe using minmax algorithm and Alpha Beta pruning

https://en.wikipedia.org/wiki/Minimax https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

1.1.1 This class is the State class which aims to keep the state of the array

[15]:
```
class State:
    empty_array = [[0, 0, 0],
                   [0, 0, 0],
                   [0, 0, 0]]

    def __init__(self, array=empty_array):
        self.array = array;

    def print(self):
        for i in range(len(self.array)):
            for j in range(len(self.array[i])):
                if self.array[i][j] == 1:
                    print("O", end=" ")
                if self.array[i][j] == -1:
                    print("X", end=" ")
                if self.array[i][j] == 0:
                    print("-", end=" ")
            print()
```

1.1.2 The Actions function create a list of all possible actions for a player at a given state

[16]:
```
def Actions(state, joueur):
    liste_action = []
    liste_action_state = []
    state_array = state.array
    for i in range(3):
        for j in range(3):
            if state.array[i][j] == 0:
                array = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
                for l in range(3):
                    for m in range(3):
                        array[l][m] = state_array[l][m]
                array[i][j] = joueur
                liste_action.append(array)
    for i in range(len(liste_action)):
        liste_action_state.append(State(liste_action[i]))
    return liste_action_state
```

1.1.3 The result function is used to change the array state in the Main code

[17]:
```
def Result(state, a):
    for i in range(3):
        for j in range(3):
            state.array[i][j] = a.array[i][j]
    return state;
```

1.1.4 The terminal state is a functions that tell if we reached a terminal state (Full array or Win/Lose state)

[18]:
```
def Terminal_Test(state):
    # print(state)
    var = 0
    for i in range(3):
        for j in range(3):
            if state.array[i][j] == 0:
                var = var + 1
    if var == 0:
        # print("Rempli")
        return True

    if state.array[0][0] == state.array[1][1] and state.array[0][0] == state.
↪array[2][2] and state.array[0][0] != 0:
        # print("Desc")
        return True
    else:
        if state.array[2][0] == state.array[1][1] and state.array[2][0] ==␣
↪state.array[0][2] and state.array[2][0] != 0:
            # print("Asc")
            return True
        else:
            for i in range(len(state.array)):
                if state.array[i][0] == state.array[i][1] and state.array[i][0]␣
↪== state.array[i][2] and state.array[i][
                    0] != 0:
                    # print("Ligne")
                    return True

            for j in range(len(state.array)):
                if state.array[0][j] == state.array[1][j] and state.array[2][j]␣
↪== state.array[0][j] and state.array[0][
                    j] != 0:
                    # print("Colonne")
                    return True
            return False
```

1.1.5 The Utility function for the evaluation of a state (To know if it leads to a win, tie or lose)

[19]:
```
def Utility(state):
    if state.array[0][0] == state.array[1][1] and state.array[0][0] == state.
↪array[2][2] and state.array[0][0] != 0:
        # print("Desc")
        return state.array[0][0]
    else:
        if state.array[2][0] == state.array[1][1] and state.array[2][0] ==␣
↪state.array[0][2] and state.array[2][0] != 0:
            # print("Asc")
            return state.array[2][0]
        else:
            for i in range(len(state.array)):
                if state.array[i][0] == state.array[i][1] and state.array[i][0]␣
↪== state.array[i][2] and state.array[i][
                    0] != 0:
                    # print("Ligne")
                    return state.array[i][0]

            for j in range(len(state.array)):
                if state.array[0][j] == state.array[1][j] and state.array[0][j]␣
↪== state.array[2][j] and state.array[0][
                    j] != 0:
                    # print("Colonne")
                    return state.array[0][j]
            return 0
```

1.1.6 Here is the minmax algorithm applied to our Tic Tac Toe game

[20]:
```
def minmax(state, joueur):
    if Terminal_Test(state):
        return Utility(state)
    if joueur == False:
        maxi = -1000
        liste_action = Actions(state, 1)
        for i in range(len(liste_action)):
            val = minmax(liste_action[i], True)
            maxi = max(val, maxi)
        return maxi
    else:
        mini = 1000
        liste_action = Actions(state, -1)
        for i in range(len(liste_action)):
            val = minmax(liste_action[i], False)
            mini = min(mini, val)
        return mini
```

1.1.7 Here is the alpha beta pruning aimed to optimize the minmax function by skipping the calculation of non relevent states

[21]:
```
def alphabeta(state, joueur, alpha, beta):
    if Terminal_Test(state):
        return Utility(state)
    if joueur == False:
        maxi = -1000
        liste_action = Actions(state, 1)
        for i in range(len(liste_action)):
            maxi = max(maxi, alphabeta(liste_action[i], True, alpha, beta))
            if maxi >= beta:
                return maxi
            alpha = max(alpha, maxi)
        return maxi
    else:
        mini = 1000
        liste_action = Actions(state, -1)
        for i in range(len(liste_action)):
            mini = min(beta, alphabeta(liste_action[i], False, alpha, beta))
            if alpha >= mini:
                return mini
            beta = min(beta, mini)
        return mini
```

1.1.8 Turn by Turn - Player against AI

[10]:
```
def Jeu():
    state = State()
    joueur = -1
    while Terminal_Test(state) == False:
        if joueur == -1:
            print(" A vous de jouer, voici la grille : ")
            state.print()
            print("")
            print("")
            jeu = Actions(state, -1)
            for i in range(len(jeu)):
                print("Choix", i, ":")
                val =alphabeta(jeu[i], False, -10000, 10000)
                jeu[i].print()
            i = -1
            while (i < 0 or i > 8):
                i = int(input("Selectionnez votre choix :"))
            state = Result(state, jeu[i])
            joueur = 1
        else:
            score = -100
            liste_action = Actions(state, 1)
            index = 0
            for i in range(len(liste_action)):
                val = minmax(liste_action[i],True)
                #val = alphabeta(liste_action[i], True, -10000, 10000)
                if val > score:
                    score = val
                    index = i
            state = Result(state, liste_action[index])
            joueur = -1
            print("")
            print("")

    if Utility(state) == -1:
        print("Vous avez gagné ! Bravo !")
    if Utility(state) == 1:
        print("Vous avez perdu ! Dommage !")
    if Utility(state) == 0:
        print("Egalité")
    state.print()
```

1.1.9 To play against the AI

[109]:
```
Jeu()
```

```
 A vous de jouer, voici la grille :
- - -
- - -
- - -



Choix 0 :
X - -
```

```
- - -
- - -
Choix 1 :
- X -
- - -
- - -
Choix 2 :
- - X
- - -
- - -
Choix 3 :
- - -
X - -
- - -
Choix 4 :
- - -
- X -
- - -
Choix 5 :
- - -
- - X
- - -
Choix 6 :
- - -
- - -
X - -
Choix 7 :
- - -
- - -
- X -
Choix 8 :
- - -
- - -
- - X
Selectionnez votre choix :4


 A vous de jouer, voici la grille :
O - -
- X -
- - -


Choix 0 :
O X -
- X -
- - -
Choix 1 :
```

```
O - X
- X -
- - -
Choix 2 :
O - -
X X -
- - -
Choix 3 :
O - -
- X X
- - -
Choix 4 :
O - -
- X -
X - -
Choix 5 :
O - -
- X -
- X -
Choix 6 :
O - -
- X -
- - X
Selectionnez votre choix :4


 A vous de jouer, voici la grille :
O - O
- X -
X - -


Choix 0 :
O X O
- X -
X - -
Choix 1 :
O - O
X X -
X - -
Choix 2 :
O - O
- X X
X - -
Choix 3 :
O - O
- X -
X X -
```

```
Choix 4 :
O - O
- X -
X - X
Selectionnez votre choix :0


 A vous de jouer, voici la grille :
O X O
- X -
X O -


Choix 0 :
O X O
X X -
X O -
Choix 1 :
O X O
- X X
X O -
Choix 2 :
O X O
- X -
X O X
Selectionnez votre choix :0


 A vous de jouer, voici la grille :
O X O
X X O
X O -


Choix 0 :
O X O
X X O
X O X
Selectionnez votre choix :0
Egalité
O X O
X X O
X O X
```

1.1.10 Turn by Turn - 2 AI against each other leading to a tie

[22]:
```python
def AI():
    state = State()
    joueur = -1
    while Terminal_Test(state) == False:
        print("State")
        state.print()
        if joueur == -1:
            score = 100
            liste_action = Actions(state, -1)
            index = 0
            for i in range(len(liste_action)):
                # val = minmax(liste_action[i],True)
                val = minmax(liste_action[i], False)
                if val < score:
                    score = val
                    index = i
            state = Result(state, liste_action[index])
            joueur = 1
            print("")
            print("")
        else:
            score = -100
            liste_action = Actions(state, 1)
            index = 0
            for i in range(len(liste_action)):
                # val = minmax(liste_action[i],True)
                val = alphabeta(liste_action[i], True, -10000, 10000)
                if val > score:
                    score = val
                    index = i
            state = Result(state, liste_action[index])
            joueur = -1
            print("")
            print("")

    if Utility(state) == -1:
        print("Vous avez gagné ! Bravo !")
    if Utility(state) == 1:
        print("Vous avez perdu ! Dommage !")
    if Utility(state) == 0:
        print("Egalité")
    state.print()
```

[23]:
```python
AI()
```

```
State
```

```
- - -
- - -
- - -


State
X - -
- - -
- - -


State
X - -
- O -
- - -


State
X X -
- O -
- - -


State
X X O
- O -
- - -


State
X X O
- O -
X - -


State
X X O
O O -
X - -


State
X X O
O O X
X - -


State
```

```
X X O
O O X
X O -


Egalité
X X O
O O X
X O X
```