



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO INGENIERIA EN SISTEMAS COMPUTACIONALES

Ejercicio de laboratorio 1

“Tic Tac Toe y Búsqueda Aleatoria”

Presentan

GOMEZ HERNANDEZ ALAN JAVIER
HERNÁNDEZ PÉREZ JUAN MANUEL
JIMÉNEZ CRUZ DANIEL
RIVAS CARRERA DIANA LAURA

Profesor

FLORIANO GARCÍA ANDRÉS

Septiembre 2023



INTRODUCCION

Tic-Tac-Toe es un juego de estrategia sencillo y popular que se juega en un tablero 3x3. El objetivo del juego es lograr que tres de tus fichas (X u O) estén alineadas en fila, columna o diagonal antes que tu oponente. Cada jugador, uno utilizando "X" y el otro "O", coloca sus fichas en el tablero de manera alternada hasta que se complete el juego o se alcance un empate si el tablero se llena sin un ganador.

El Tic-Tac-Toe es un campo de estudio clásico en la inteligencia artificial debido a su simplicidad y estructura perfecta de información. En este juego, todas las posibles posiciones del tablero y movimientos son conocidos por completo, lo que lo convierte en un escenario ideal para el desarrollo y la prueba de algoritmos de IA.

A lo largo de los años, se han desarrollado varias estrategias y algoritmos de IA para jugar Tic-Tac-Toe, como por ejemplo:

- Búsqueda minimax: Explora todas las posibles jugadas y calcula el valor de cada jugada, suponiendo que el oponente juega de manera óptima.
- Bases de conocimiento: Se usan bases de conocimientos predefinidas para tomar decisiones, las cuales tienen reglas y estrategias específicas del juego.
- Aprendizaje por reforzamiento: Se entrenan IA's mediante la experiencia, así como mediante recompensas y/o penalizaciones.

La búsqueda aleatoria es un enfoque simple en la resolución de problemas y la toma de decisiones que implica tomar acciones al azar en lugar de seguir una estrategia deliberada o planificada. Este enfoque es utilizado en una variedad de contextos, incluyendo inteligencia artificial, optimización y toma de decisiones. En la búsqueda aleatoria, las decisiones se toman sin un patrón predefinido o lógica detrás de ellas, y en cambio, se basan en la probabilidad.

La búsqueda aleatoria no se considera una estrategia eficiente, cuando se trata de resolver problemas complejos o de objetivos específicos, aún así se encuentran diversos ejemplos en los que este mismo se utiliza, como los siguientes:

- Optimización aleatoria: Se exploran soluciones potenciales al seleccionar dentro de un conjunto de posibles soluciones.
- Experimentación y pruebas: Se utiliza para cuando no se conoce la relación entre las variables o se busca encontrar patrones inesperados.
- Selección aleatoria: Se aplica más que nada en donde las opciones son difíciles de evaluar objetivamente.

DESARROLLO

1. Desarrolla una versión tonta del juego del gato (tic tac toe):
 - Deberá ofrecer la posibilidad de jugar humano vs computadora y computadora vs computadora.
 - La selección de movimientos la hará de forma aleatoria y no se permitirá sobrescribir una jugada previa.
 - Deberán validarse las combinaciones ganadoras y de empate.

El juego comienza haciendo una validación, la cual empieza con las filas, después con las columnas y finalmente en diagonal, para saber si las casillas están llenas y así mismo saber si el usuario o la maquina, han ganado el juego.

El usuario tiene la posibilidad de jugar con la maquina o en su caso, que la maquina juegue consigo misma. Así mismo el usuario solo tiene permitido 9 tiradas para poder ganar.

Cuando el usuario hace una tirada, esta igualmente se valida para saber si en el lugar que ha decidido ponerla, esta vacía y en su caso de estar llena, escoja otra casilla, así hasta que haya un ganador o empate.

En el modo de juego donde la maquina juega consigo misma, se hace uso de números aleatorios y se lleva acabo el mismo algoritmo que en humano vs máquina.

```
# This is a sample Python script.

# Press Mayús+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool
windows, actions, and settings.
import random
import os
import time
def validar(casillas):
    print(f"    {casillas[1]} | {casillas[2]} | {casillas[3]}")
    print(f"    {casillas[4]} | {casillas[5]} | {casillas[6]}")
    print(f"    {casillas[7]} | {casillas[8]} | {casillas[9]}")
    if casillas[1] == casillas[2] == casillas[3] == "X" or casillas[4] ==
casillas[5] == casillas[6] == "X" or casillas[7] == casillas[8] ==
casillas[9] == "X" or casillas[1] == casillas[4] == casillas[7] == "X" or
casillas[2] == casillas[5] == casillas[8] == "X" or casillas[3] ==
casillas[6] == casillas[9] == "X" or casillas[1] == casillas[5] ==
casillas[9] == "X" or casillas[3] == casillas[5] == casillas[7] == "X":
        print("Ganaste jugador 1  (X)")
        return True
    elif casillas[1] == casillas[2] == casillas[3] == "O" or casillas[4]
```

```

== casillas[5] == casillas[6] == "O" or casillas[7] == casillas[8] ==
casillas[9] == "O" or casillas[1] == casillas[4] == casillas[7] == "O" or
casillas[2] == casillas[5] == casillas[8] == "O" or casillas[3] ==
casillas[6] == casillas[9] == "O" or casillas[1] == casillas[5] ==
casillas[9] == "O" or casillas[3] == casillas[5] == casillas[7] == "O":
    print("Ganaste jugador 2 (O)")
    return True
else:
    return False

casillas = {1: "", 2: "", 3: "", 4: "", 5: "", 6: "", 7: "", 8: "", 9:
""}

def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'menu') # Press Ctrl+F8 to toggle the breakpoint.
    print(f'Opción 1: humano vs maquina')
    print(f'Opción 2: maquina vs maquina')
    print(f'EL TABLERO ES DE LA SIGUIENTE MANERA:')
    print(f'  1 | 2 | 3')
    print(f'  4 | 5 | 6')
    print(f'  7 | 8 | 9')

    option = input("Ingrese una opción: ")

    if option in "1":
        print(f'humano vs maquina')
        for i in range(10): #Es para las tiradas totale

            while True: # Para que el usuario ingrese un numero valido
                try:
                    entrada = int(input("Ingresa una casilla del 1 al 9:
"))
                    if 1 <= entrada <= 9:
                        if casillas[entrada] == "":
                            casillas[entrada] = "X"
                            break
                        else:
                            print("Esta casilla ya está ocupada")
                    else:
                        print("El número debe estar en el rango del 1 al
9.")
                except ValueError:
                    print("Por favor, ingresa un número válido.")
            if i == 10:
                print("Empate")
                break

            while True:#Para la tirada de la maquina
                numero = random.randint(1, 9)
                if casillas[numero] == "":
                    casillas[numero] ="O"
                    print(f'la maquina selecciono la casilla:{numero}')
                    break
            if validar(casillas):
                break
            time.sleep(2) # Espera 2 segundo antes de verificar de nuevo

```

```

print("Gracias por jugar")

if option == "2":
    print(f'maquina vs maquina')
    for i in range(10): #Es para las tiradas totale

        while True: # PArA la tirada de la maquina
            numero = random.randint(1, 9)
            if casillas[numero] == "":
                casillas[numero] = "X"
                print(f'la maquina 1 selecciono la casilla:{numero}')
                break
        if i == 10:
            print("Empate")
            break

        while True:#PArA la tirada de la maquina
            numero = random.randint(1, 9)
            if casillas[numero] == "":
                casillas[numero] = "O"
                print(f'la maquina 2 selecciono la casilla:{numero}')
                break
        if validar(casillas):
            break
        time.sleep(2) # Espera 2 segundo antes de verificar de nuevo
    print("Gracias por jugar")

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/

```

Pruebas de funcionamiento

```
Opción 1: humano vs maquina
Opción 2: maquina vs maquina
EL TABLERO ES DE LA SIGUIENTE MANERA:
  1 | 2 | 3
  4 | 5 | 6
  7 | 8 | 9
Ingrese una opción: 2
maquina vs maquina
la maquina 1 selecciono la casilla:6
la maquina 2 selecciono la casilla:2
  | 0 |
  | | X
  | |
la maquina 1 selecciono la casilla:1
la maquina 2 selecciono la casilla:9
  X | 0 |
  | | X
  | | 0
la maquina 1 selecciono la casilla:8
la maquina 2 selecciono la casilla:5
  X | 0 |
  | 0 | X
  | X | 0
la maquina 1 selecciono la casilla:3
la maquina 2 selecciono la casilla:7
  X | 0 | X
  | 0 | X
  0 | X | 0
la maquina 1 selecciono la casilla:4
Empate
  X | 0 | X
  X | 0 | X
  0 | X | 0
```

```
menu
Opción 1: humano vs maquina
Opción 2: maquina vs maquina
EL TABLERO ES DE LA SIGUIENTE MANERA:
  1 | 2 | 3
  4 | 5 | 6
  7 | 8 | 9
Ingrese una opción: 1
humano vs maquina
Ingresa una casilla del 1 al 9: 1
la maquina selecciono la casilla:3
  X | | 0
  | |
  | |
Ingresa una casilla del 1 al 9: 4
la maquina selecciono la casilla:9
  X | | 0
  X | |
  | | 0
Ingresa una casilla del 1 al 9: 5
la maquina selecciono la casilla:7
  X | | 0
  X | X |
  0 | | 0
Ingresa una casilla del 1 al 9: 6
la maquina selecciono la casilla:2
  X | 0 | 0
  X | X | X
  0 | | 0
Ganaste jugador 1 (X)

Process finished with exit code 0
```

DESARROLLO

2. Desarrolla un programa que mediante búsqueda aleatoria intente resolver el siguiente sistema de ecuaciones:

$$16B - 6D + 4E + F = -36$$

$$B - 8D + E + F = -64$$

$$16B + 2D - 4E + F = -4$$

$$9B + 8D - 3E + F = -64$$

Para la realización de esta práctica, se define el sistema de ecuaciones anteriormente mencionado, en este mismo se contemplan las pruebas con números enteros del -100 al 100.

Las variables a encontrar, B, D, E y F, se les asignan un valor random de -100 a 100.

Para la evaluación, se hace un análisis de 9999999999 pruebas, en cada una de las cuales, evalúa si cada una de las ecuaciones se cumple, en caso de que alguna no se cumpla (debido a la evaluación en corto circuito de Python) ya no se evalúan las demás y comenzará otro ciclo probando con otros valores. Así hasta encontrar los números correctos para dar solución a cada una de estas.

```
import random

def ecuacion_1(b, d, e, f):
    return 16*b - 6*d + 4*e + f == -36

def ecuacion_2(b, d, e, f):
    return b - 8*d + e + f == 64

def ecuacion_3(b, d, e, f):
    return 16*b + 2*d - 4*e + f == -4

def ecuacion_4(b, d, e, f):
    return 9*b + 8*d - 3*e + f == -64

def solucion_aleatoria():
    max_intentos = 9999999999
    intentos = 0

    while intentos < max_intentos:
        intentos += 1

        b = random.randint(-100, 100)
        d = random.randint(-100, 100)
        e = random.randint(-100, 100)
        f = random.randint(-100, 100)

        if (ecuacion_1(b, d, e, f) and
```

```

    ecuacion_2(b, d, e, f) and
    ecuacion_3(b, d, e, f) and
    ecuacion_4(b, d, e, f)):

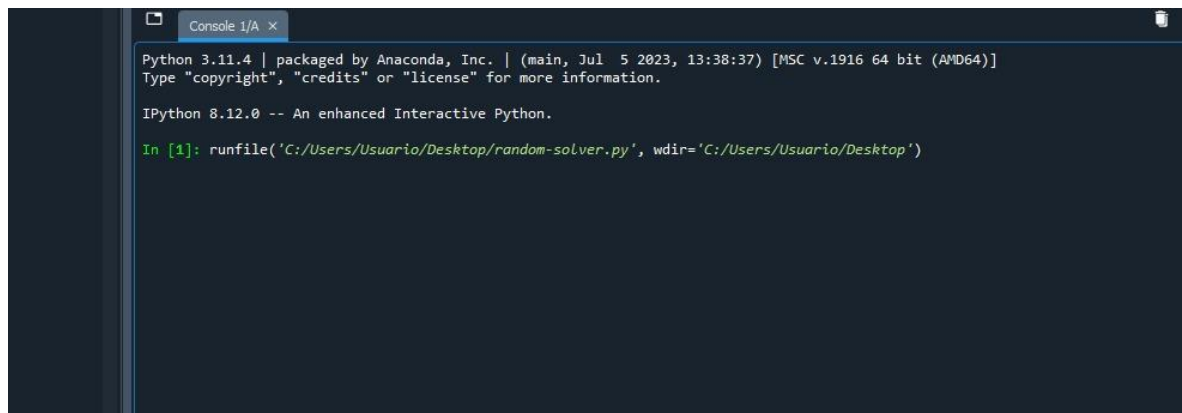
    print('Solución encontrada:')
    print(f'b: {b}')
    print(f'd: {d}')
    print(f'e: {e}')
    print(f'f: {f}')
    return

    print('No se encontró solución después de', intentos, 'intentos.')
    print(f'Últimos valores: b={b}, d={d}, e={e}, f={f}')

if __name__ == '__main__':
    solucion_aleatoria()

```

PRUEBAS DE FUNCIONAMIENTO



```

Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.12.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Usuario/Desktop/random-solver.py', wdir='C:/Users/Usuario/Desktop')

```

En este caso no se muestra el resultado debido a que intenta resolver un sistema de ecuaciones lineales utilizando la fuerza bruta. Esto significa que selecciona valores al azar para las variables b , d , e y f hasta encontrar una solución que cumpla con todas las ecuaciones o, en nuestro caso, hasta que haya realizado demasiados intentos. Si la solución está dentro del rango prescrito (-100 a 100), eventualmente encontrará una solución, aunque de una manera poco eficiente. Si la solución está fuera de este rango o si las ecuaciones no tienen solución, este método no funcionará.

CONCLUSIÓN

Los códigos presentados anteriormente demuestran diferentes aspectos de la programación y la toma de decisiones que son fundamentales en la construcción de sistemas de IA como, por ejemplo:

- **Interacción Humano-Máquina:** El primer código, que implementa un juego de Tic-Tac-Toe, representa la interacción entre humanos y máquinas en un contexto de juego. La IA se utiliza comúnmente en juegos para crear oponentes virtuales desafiantes, como se muestra en el código cuando la máquina toma decisiones sobre qué movimientos realizar. La IA en juegos puede ser desde muy simple hasta muy compleja, dependiendo de la experiencia que se desee proporcionar
- **Toma de Decisiones y Búsqueda Exhaustiva:** El segundo código aborda la resolución de un problema matemático mediante una búsqueda exhaustiva de soluciones. Aunque este código no utiliza técnicas de IA avanzadas, ilustra el concepto de cómo una IA podría explorar un espacio de soluciones para encontrar la mejor. En problemas más complejos, como el aprendizaje automático, las IA utilizan algoritmos de búsqueda y optimización para tomar decisiones basadas en datos y mejorar su rendimiento.

La IA implica la automatización de tareas y la toma de decisiones basadas en datos o algoritmos, y estos ejemplos simples destacan la capacidad de las máquinas para realizar tareas repetitivas o complejas, así como también se centra en resolver problemas, pero a menudo se utiliza para problemas más complejos que no se pueden resolver fácilmente mediante enfoques convencionales.