Simple Game AI for Rock-Paper-Scissors

Title Page

Project Title: Simple Game AI for Rock-Paper-Scissors
Submitted by: shivam kumar
Date: 11/03/2025

---

1. Introduction

Rock-Paper-Scissors is a popular hand game that is simple yet interesting from an AI perspective. The goal of this project is to build an AI that can play against a human player and improve its performance over time. Instead of making purely random moves, the AI learns from the opponent's past choices and adapts to counter them.

This project aims to demonstrate basic AI techniques such as pattern recognition and predictive modelling . The AI system will collect user input, process past moves, and make optimized choices to increase its winning probability.

---

2. Methodology

The AI follows a structured methodology to enhance its decision-making:

1. **Random Choice (Initial Stage):** When the game starts, the AI makes random choices since no data is available.
2. **Opponent Move Tracking:** The AI keeps a history of the opponent’s choices.
3. **Prediction Model:** The AI predicts the opponent’s next move by analyzing the most frequent past choices.
4. **Counter Move Strategy:** Based on the prediction, the AI selects the move that can beat the predicted choice.
5. **Gameplay Loop:** The game continues in a loop until the user decides to exit.

---

3. Typed Code

Below is the complete Python code for the Rock-Paper-Scissors AI:

import random

```
class RockPaperScissorsAI:
    def __init__(self):
        self.choices = ["rock", "paper", "scissors"]
        self.opponent_history = []

    def get_ai_choice(self):
        if not self.opponent_history:
            return random.choice(self.choices)

        predicted_move = max(set(self.opponent_history), key=self.opponent_history.count)
        counter_moves = {"rock": "paper", "paper": "scissors", "scissors": "rock"}
        return counter_moves[predicted_move]

    def play_round(self, user_choice):
        if user_choice not in self.choices:
            return "Invalid choice! Choose rock, paper, or scissors."

        ai_choice = self.get_ai_choice()
        self.opponent_history.append(user_choice)

        if user_choice == ai_choice:
            result = "It's a tie!"
        elif (user_choice == "rock" and ai_choice == "scissors") or \
             (user_choice == "scissors" and ai_choice == "paper") or \
             (user_choice == "paper" and ai_choice == "rock"):
            result = "You win!"
        else:
            result = "AI wins!"

        return f"You chose {user_choice}, AI chose {ai_choice}. {result}"
```

```
if __name__ == "__main__":
    game_ai = RockPaperScissorsAI()

    while True:
        user_input = input("Enter rock, paper, or scissors (or 'quit' to exit): ").lower()
        if user_input == "quit":
            print("Thanks for playing!")
            break
        print(game_ai.play_round(user_input))
```

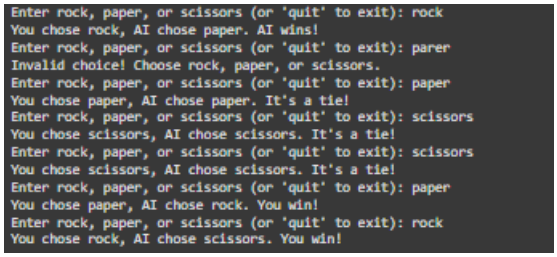
---

4. Screenshots

4.1 Code Screenshot

```python
import random

class RockPaperScissorsAI:
    def __init__(self):
        self.choices = ["rock", "paper", "scissors"]
        self.opponent_history = []

    def get_ai_choice(self):
        if not self.opponent_history:
            return random.choice(self.choices)

        # Predict opponent's next move based on history
        predicted_move = max(set(self.opponent_history), key=self.opponent_history.count)

        # Choose the move that beats the predicted move
        counter_moves = {
            "rock": "paper",
            "paper": "scissors",
            "scissors": "rock"
        }
        return counter_moves[predicted_move]

    def play_round(self, user_choice):
        if user_choice not in self.choices:
            return "Invalid choice! Choose rock, paper, or scissors."

        ai_choice = self.get_ai_choice()
        self.opponent_history.append(user_choice)

        if user_choice == ai_choice:
            result = "It's a tie!"
        elif (user_choice == "rock" and ai_choice == "scissors") or \
             (user_choice == "scissors" and ai_choice == "paper") or \
             (user_choice == "paper" and ai_choice == "rock"):
            result = "You win!"
        else:
            result = "AI wins!"

        return f"You chose {user_choice}, AI chose {ai_choice}. {result}"

# Run the game
if __name__ == "__main__":
    game_ai = RockPaperScissorsAI()

    while True:
        user_input = input("Enter rock, paper, or scissors (or 'quit' to exit): ").lower()
        if user_input == "quit":
            print("Thanks for playing!")
            break
        print(game_ai.play_round(user_input))
```

4.2 Output Screenshot



---

5. Conclusion

The Rock-Paper-Scissors AI demonstrates how a simple machine learning approach can be applied to a basic game. By tracking past moves and using a predictive strategy, the AI improves its decision-making over time.

Future Enhancements:

- Implement reinforcement learning for better adaptation.
- Introduce a Markov decision model for enhanced predictions.
- Develop a graphical user interface (GUI) for better user interaction.

References

1. Rock-Paper-Scissors Game Theory: https://en.wikipedia.org/wiki/Rock_paper_scissors
2. Machine Learning in Game: https://www.analyticsvidhya.com/blog/2023/03/ml-and-ai-in-game-development/

Prepared by: SHIVAM KUMAR
Date: 11/03/2025.