Markov Gambling

November 12, 2024

Simulate a gambling game where you either win or lose money based on specific probabilities based on the above parameters

1 Simulation Steps

1.1 Initialize Simulation:

Define the number of trials (num trials = 10) to run, starting wealth for the player, probability of winning each round, and the target wealth to reach.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

# Simulation parameters
num_trials = 10  # Number of trials to run
initial_wealth = 1
target_wealth = 5
ruin_wealth = 0
win_prob = 0.4  # Probability of winning each round
```

1.2 Conduct Trials:

– For each trial: ∗ Start with the initial wealth. ∗ Track the wealth changes throughout the game. ∗ Simulate rounds of play: · Continue playing until either the target wealth is reached or the player loses all their money. · In each round, determine the outcome based on the defined probability, updating the wealth accordingly.

[9]:
```
# Simulation results storage
wealth_progressions = []
win_counts = 0
total_rounds = []

# Run trials
for trial in range(num_trials):
    wealth = initial_wealth
    wealth_progression = [wealth]
    rounds = 0
    # Play until target wealth or ruin
    while wealth > ruin_wealth and wealth < target_wealth:
        if np.random.rand() < win_prob:
            wealth += 1  # Win: increase wealth by 1
        else:
            wealth -= 1  # Lose: decrease wealth by 1
        wealth_progression.append(wealth)
        rounds += 1

    wealth_progressions.append(wealth_progression)
    total_rounds.append(rounds)

    # Check if player won (reached target wealth)
    if wealth == target_wealth:
        win_counts += 1

# Calculate winning rate and average rounds
win_rate = win_counts / num_trials
avg_rounds = np.mean(total_rounds)

print(f"Winning rate: {win_rate}")
```

```
Winning rate: 0.3
```

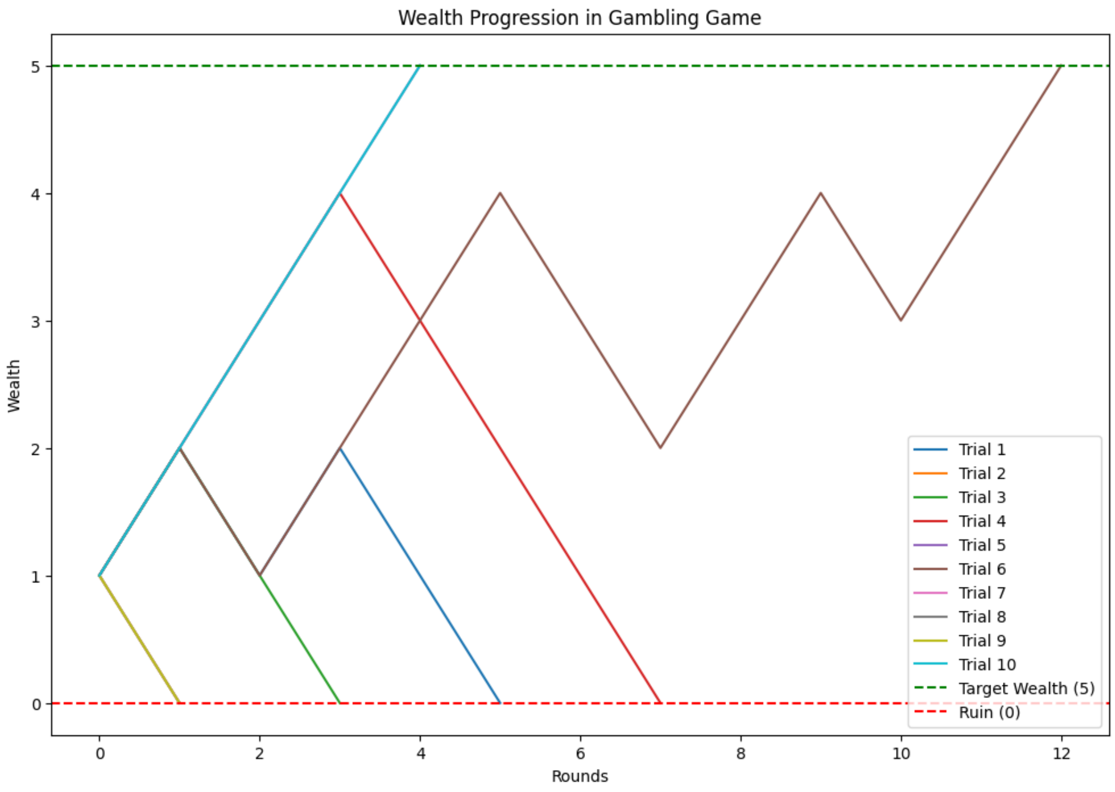
1.3 Store Results:

– Keep a record of the wealth progression for each trial.

1.4 Visualize Outcomes:

– Create plots to illustrate the wealth changes over the course of several trials. – Mark significant states, such as the target wealth and the point of ruin.

[10]:
```
# Visualize outcomes
plt.figure(figsize=(12, 8))
for i, wealth_prog in enumerate(wealth_progressions):
    plt.plot(wealth_prog, label=f'Trial {i+1}')
plt.axhline(y=target_wealth, color='g', linestyle='--', label='Target Wealth␣
 ↪(5)')
plt.axhline(y=ruin_wealth, color='r', linestyle='--', label='Ruin (0)')
plt.xlabel('Rounds')
plt.ylabel('Wealth')
plt.legend()
plt.title('Wealth Progression in Gambling Game')
plt.show()
```



1.5 Analyze Results:

– Calculate the number of times the player wins or loses across all trials. – Determine the overall winning rate and the average number of rounds played per trial. – Present these statistics for review.

```
[11]: # Analysis results
print(f"Win rate: {win_rate:.2f}")
print(f"Average rounds played per trial: {avg_rounds:.2f}")
```

```
Win rate: 0.30
Average rounds played per trial: 3.90
```