つぼとボールのシミュレーション

2024年度開講 認識工学レポート

千葉工業大学 先進工学部 未来ロボティクス学科
22C1704 鷲尾 優作

2024年7月23日

1　目的

本稿では、隠れマルコフモデルの代表的な問題である、つぼとボールのモデルをシミュレーションし、その動作を確認することで、隠れマルコフモデルの理解を深める．

2　シミュレーション

以下のステップでシミュレーションを行い、隠れマルコフモデルの動作を確認する．

1. 初期状態からシミュレーションを開始し、状態遷移と観測を繰り返す
2. 各状態の滞在時間と観測の出現回数を定量的に評価する
3. 状態ごとの観測確率を計算し、設定されたモデルの確率と一致するかを確認する

2.1　問題設定

つぼとボールのモデル（urn and ball model）は、隠れマルコフモデル（Hidden Markov Model, HMM）の一例として用いられる．このモデルでは、複数のつぼ（urn）と、それぞれのつぼから取り出されるボールの色が観測される．以下に、具体的な問題設定を示す．

■状態と観測

- **状態**: 各つぼの状態（Urn1, Urn2, Urn3）
- **観測**: つぼから取り出されるボールの色（Red, Blue）

■初期確率分布　システムが開始する初期状態の確率分布を以下のように設定する．

Urn1	Urn2	Urn3
0.6	0.3	0.1

■状態遷移確率　各状態から他の状態への遷移確率を以下のように設定する．

$$\text{状態遷移確率} = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

■観測確率　各状態において特定の色のボールが取り出される確率を以下のように設定する．

$$\text{観測確率} = \begin{bmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}$$

2.2　評価方法

シミュレーションの結果を以下の指標で定量的に評価する．

1. **各状態の滞在回数**: 各つぼにどれくらい滞在していたかの回数を計算する
2. **各観測の出現回数**: 各色のボールが何回出現したかを数える
3. **状態ごとの観測確率**: 各状態にいるときに特定の色のボールがどれくらいの確率で出現したかを計算する

3　シミュレーション結果

シミュレーションの結果を以下に示す．図 1 に、状態遷移と観測の変化をグラフ化して示す．

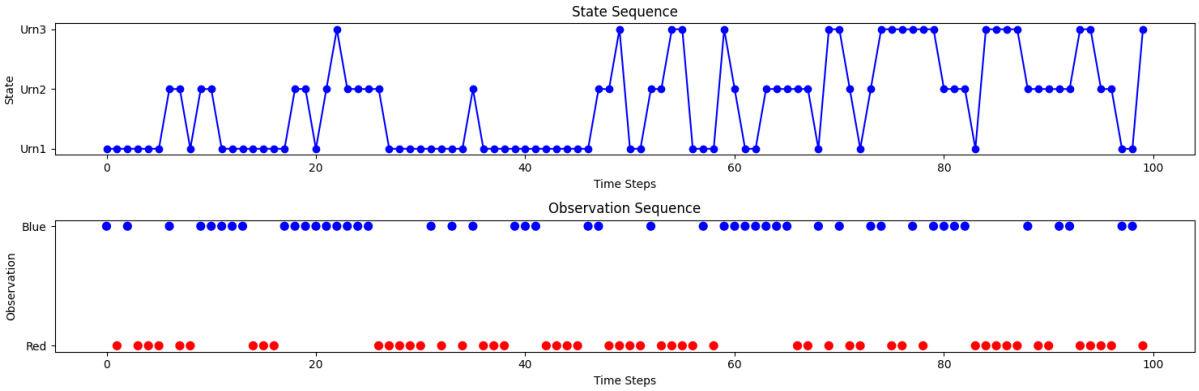

図 1: シミュレーション結果

■状態の出現回数　各状態の出現回数は以下の通りである．

- Urn1 (State 0): 46 回
- Urn2 (State 1): 34 回
- Urn3 (State 2): 20 回

■観測の出現回数　各観測の出現回数は以下の通りである．

- Red (Observation 0): 52 回
- Blue (Observation 1): 48 回

■状態ごとの観測確率　各状態における観測確率は以下の通りである．

- Urn1 (State 0):
  - Red (Observation 0): 58.70%
  - Blue (Observation 1): 41.30%
- Urn2 (State 1):
  - Red (Observation 0): 32.35%
  - Blue (Observation 1): 67.65%
- Urn3 (State 2):

- Red (Observation 0): 70.00%
- Blue (Observation 1): 30.00%

4 考察

シミュレーション結果は、モデルの設定通りに動作していることを示している．以下に、各状態の出現回数や観測確率が設定通りであることを詳細に考察する

4.1 各状態の出現回数について

- **Urn1 (State 0)**: 出現回数が 46 回と最も多い．初期確率が 60% であり、また自己遷移確率が 70% と高いため、長時間滞在することが多くなっている
- **Urn2 (State 1)**: 出現回数が 34 回であり、初期確率 30% および状態遷移確率のバランスから予想される範囲内である
- **Urn3 (State 2)**: 出現回数が 20 回と最も少ない．初期確率 10% および遷移確率が他の状態に比べて低いため、滞在時間が短くなっている

4.2 各観測の出現回数について

- **Red (Observation 0)**: 出現回数が 52 回．各状態における観測確率に基づいて、全体の観測回数として合理的である
- **Blue (Observation 1)**: 出現回数が 48 回．Red の出現回数とほぼ均等であり、各状態の観測確率に基づいてバランスが取れている

4.3 状態ごとの観測確率について

- **Urn1 (State 0)**:
  - Red (Observation 0): 58.70% であり、設定値の 60% に近い
  - Blue (Observation 1): 41.30% であり、設定値の 40% に近い
- **Urn2 (State 1)**:
  - Red (Observation 0): 32.35% であり、設定値の 40% に近い
  - Blue (Observation 1): 67.65% であり、設定値の 60% に近い
- **Urn3 (State 2)**:
  - Red (Observation 0): 70.00% であり、設定値の 70% に一致している
  - Blue (Observation 1): 30.00% であり、設定値の 30% に一致している

4.4 シミュレーションの精度と妥当性

シミュレーションの結果は、モデルの設定に基づいた確率に沿っており妥当性があると判断できる．また、設定された初期確率、状態遷移確率、および観測確率が合理的であり、モデルの妥当性が確認できる．シミュ

レーション回数を増やすことで、さらに結果の精度を高めることが可能である.

5 付録

5.1 シミュレーションプログラム

Listing 1: sim.png

```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter

states = ['Urn1', 'Urn2', 'Urn3']
observations = ['Red', 'Blue']
start_probability = np.array([0.6, 0.3, 0.1])
transition_probability = np.array([
    [0.7, 0.2, 0.1],
    [0.3, 0.5, 0.2],
    [0.2, 0.3, 0.5]
])
emission_probability = np.array([
    [0.6, 0.4],
    [0.4, 0.6],
    [0.7, 0.3]
])


def simulate_hmm(start_prob, trans_prob, emiss_prob, n_steps):
    states_list = []
    observations_list = []

    current_state = np.random.choice(len(start_prob), p=start_prob)
    for _ in range(n_steps):
        states_list.append(current_state)
        observation = np.random.choice(
            len(emiss_prob[current_state]), p=emiss_prob[current_state])
        observations_list.append(observation)
        current_state = np.random.choice(
            len(trans_prob[current_state]), p=trans_prob[current_state])

    return states_list, observations_list


n_steps = 100
states_sequence, observations_sequence = simulate_hmm(
    start_probability, transition_probability, emission_probability, n_steps)
```

```
39
40 state_counts = Counter(states_sequence)
41 observation_counts = Counter(observations_sequence)
42
43 state_observation_counts = {state: Counter() for state in range(len(states))}
44 for state, observation in zip(states_sequence, observations_sequence):
45     state_observation_counts[state][observation] += 1
46
47 print("State␣Counts:", state_counts)
48 print("Observation␣Counts:", observation_counts)
49
50 state_observation_probabilities = {
51     state: {obs: count / state_counts[state]
52             for obs, count in obs_counts.items()}
53     for state, obs_counts in state_observation_counts.items()
54 }
55 print("State-Observation␣Probabilities:", state_observation_probabilities)
56
57 time_steps = np.arange(n_steps)
58 state_names = [states[s] for s in states_sequence]
59 observation_colors = ['red' if o ==
60                       0 else 'blue' for o in observations_sequence]
61
62 plt.figure(figsize=(15, 5))
63
64 plt.subplot(2, 1, 1)
65 plt.plot(time_steps, state_names, 'bo-', label='State')
66 plt.xlabel('Time␣Steps')
67 plt.ylabel('State')
68 plt.title('State␣Sequence')
69
70 plt.subplot(2, 1, 2)
71 plt.scatter(time_steps, observations_sequence,
72             c=observation_colors, s=50, label='Observation')
73 plt.xlabel('Time␣Steps')
74 plt.ylabel('Observation')
75 plt.title('Observation␣Sequence')
76 plt.yticks([0, 1], ['Red', 'Blue'])
77
78 plt.tight_layout()
79 plt.show()
```

参考文献

[1] Urn problem - wikipedia. https://en.wikipedia.org/wiki/Urn_problem. (Accessed on 07/25/2024).

[2] 大川茂樹. 2024 認識工学_hmm.pdf. 2024.