MACROMONOPOLY

"K y Matriz P"

Daniel

17 de agosto de 2025

Resumen

Este documento recoge de forma exhaustiva y rigurosa la justificación matemática y técnica de la construcción de la matriz de transición P usada en el modelo del Monopoly modificado, la motivación para incluir la variable k (contador de pares consecutivos), los pasos exactos de cálculo de la matriz en el código, los fundamentos teóricos (espacio vectorial, autovalores y autovectores, teoremas relevantes como Perron-Frobenius), consideraciones numéricas y de modelado, y verificaciones recomendadas para una defensa académica completa. Está redactado para ser presentado ante un tribunal o audiencia técnica y contiene explicaciones detalladas desde primeros principios.

Índice

1. Introducción 3

2. Resumen 3

3. Detalle literal del cálculo de la matriz P según el código 3
   - 3.1. Espacio de estados y mapeo a índice único . . . 3
   - 3.2. Inicialización . . . 3
   - 3.3. Enumeración de transiciones (bucle principal) . . . 4
   - 3.4. Reglas concretas implementadas . . . 4
   - 3.5. Acumulación iterativa (push a la matriz) . . . 4
   - 3.6. Normalización por fila . . . 4
   - 3.7. Guardado . . . 5

4. Qué representa cada elemento de P 5

5. Por qué el proceso es (o no es) Markoviano: la función de k 5
   - 5.1. Definición formal de la propiedad de Markov . . . 5
   - 5.2. Por qué sin k el proceso no es Markoviano . . . 5
   - 5.3. La solución: incluir k (aumento del espacio de estados) . . . 5
   - 5.4. Ejemplo ilustrativo . . . 5

6. La matriz P como operador lineal y el espacio vectorial 6
   - 6.1. Espacio vectorial . . . 6
   - 6.2. P como aplicación lineal . . . 6

7. Autovalores y autovectores: por qué importan aquí 6
   - 7.1. Definición . . . 6
   - 7.2. Interpretación para cadenas de Markov . . . 6

8. Teoremas que sustentan el enfoque 6
8.1. Perron-Frobenius (versión para cadenas) 6
8.2. Teorema espectral (finito) 7

9. Cómo se calcula π en la práctica 7

10. Interpretaciones prácticas del espectro 7

11. Condiciones de existencia y unicidad de π 7

12. Errores numéricos y consideraciones computacionales 7
12.1 Posibles fuentes de error 7
12.2 Recomendaciones prácticas 7

13. Interpretación con teoría de grafos 8

14. Sesgos de modelado y supuestos 8

15. Posibles fallos lógicos en la implementación (lista de chequeo) 8

16. Conclusión 8

1. Introducción

En este documento explicamos de forma pormenorizada cómo se construyó la matriz de transición P usada para modelar el Monopoly modificado, por qué la inclusión de la variable k (pares consecutivos) es necesaria para asegurar la propiedad de Markov de primer orden, y qué implicaciones matemáticas y numéricas tiene esa decisión. La exposición está diseñada para quien no asume conocimientos previos: todas las construcciones se fundamentan y se conectan con los teoremas que justifican su uso en análisis y simulación.

2. Resumen

- Definimos un espacio de estados ampliado: cada estado es la tripleta (t, c, k) con $t \in \{1, 2\}$ (tablero), $c \in \{0, \ldots, 23\}$ (casilla), $k \in \{0, 1, 2\}$ (pares consecutivos).
- Construimos la matriz de transición $P \in \mathbb{R}^{144\times144}$ enumerando, para cada estado, las 36 combinaciones posibles de dos dados y acumulando la probabilidad $1/36$ en la entrada destino correspondiente.
- Normalizamos por fila por robustez numérica y guardamos la matriz.
- Calculamos la distribución estacionaria π como autovector izquierdo asociado al autovalor $\lambda = 1$ de P (mediante autovectores de $P^\top$).
- Justificamos teóricamente la inclusión de k usando el teorema de aumento del espacio de estados y argumentos prácticos.

3. Detalle literal del cálculo de la matriz P según el código

A continuación se describe paso a paso *lo que hace el código* para construir P.

3.1. Espacio de estados y mapeo a índice único

Se define el número total de estados como

$$N_{\text{ESTADOS}} = 2 \times 24 \times 3 = 144.$$

La función de mapeo usada convierte la tripleta (tablero, casilla, k) en un índice entero

$$\texttt{estado\_a\_indice}(t, c, k) = (t - 1) \cdot 72 + c \cdot 3 + k,$$

con $t \in \{1, 2\}$, $c \in \{0, \ldots, 23\}$, $k \in \{0, 1, 2\}$. Este mapeo es inyectivo y total: cada estado físico se asocia a una fila/columna de la matriz P.

3.2. Inicialización

Se crea la matriz de ceros

$$P = \mathbf{0}_{144\times144},$$

que será rellenada de forma iterativa. Cada fila representa la distribución condicional $P(X_{n+1} \mid X_n = i)$ para un estado i dado.

3.3. Enumeración de transiciones (bucle principal)

Para cada estado actual i (recorrido mediante las tres variables t, c, k), el código:

1. Calcula el índice del estado actual: $i = \texttt{estado\_a\_indice}(t, c, k)$.
2. Itera sobre todos los pares ordenados de dados (d_1, d_2), con $d_1, d_2 \in \{1, \ldots, 6\}$. Cada combinación tiene probabilidad

$$\mathbb{P}(d_1, d_2) = \frac{1}{36}$$

bajo el supuesto de dados justos e independientes.

3. Para cada combinación calcula:

$$s = d_1 + d_2, \qquad c' = (c + s) \bmod 24, \qquad \texttt{es\_par} = (d_1 = d_2).$$

4. Actualiza la lógica de transición en función de es_par y de las reglas especiales (FORTUNA, cárcel, cambio de tablero), determinando un nuevo estado $j = \texttt{estado\_a\_indice}(t_{\text{nuev}}$
5. Acumula la probabilidad:

$$P[i, j] \mathrel{+}= \frac{1}{36}.$$

3.4. Reglas concretas implementadas

En resumen, las reglas que alteran la transición para un resultado (d_1, d_2) son:

- Si sale par: incrementar $k \leftarrow k+1$. Si $k \geq 3$ entonces ir a la cárcel y fijar $k \leftarrow 0$. Si $k < 3$, se vuelve a tirar (se modela como permanecer en la misma casilla con nuevo k).
- Si no sale par: $k \leftarrow 0$. Si la casilla destino es FORTUNA y el tiro anterior (es decir, el estado actual) tiene $k > 0$, entonces se cambia de tablero según la regla de superposición; en caso contrario no se cambia de tablero.
- En todos los demás casos, el nuevo estado resulta de mover la casilla c por s posiciones y fijar $k = 0$.

3.5. Acumulación iterativa (push a la matriz)

Cada combinación de dados añade su probabilidad $\frac{1}{36}$ a la entrada destino. Por tanto la fila i de la matriz P se construye como suma de cargas $\frac{1}{36}$ sobre las columnas correspondiente Se trata de una acumulación iterativa, no de una asignación única. Formalmente:

$$P_{ij} = \sum_{\substack{(d_1,d_2) \\ \texttt{imagen}(i,d_1,d_2)=j}} \frac{1}{36}.$$

3.6. Normalización por fila

Al final se aplica

$$P \leftarrow \frac{P}{P\mathbf{1}}$$

(operación vectorial que divide cada fila por su suma) para robustecer frente a errores numéricos. Teóricamente, si la construcción fue exacta, cada fila ya suma 1, pero la normalización corrige pequeñas desviaciones de coma flotante o casos anómalos.

3.7. Guardado

La matriz final se guarda binariamente (por ejemplo con numpy.save) para su análisis posterior: matriz_transicion_monopoly.npy.

4. Qué representa cada elemento de P

$$P_{ij} = \mathbb{P}(X_{n+1} = j \mid X_n = i),$$

con $i, j \in \{0, \ldots, 143\}$ índices del espacio de estados. Las condiciones de no negatividad y suma por filas:

$$P_{ij} \geq 0, \qquad \sum_j P_{ij} = 1$$

garantizan que cada fila es una distribución de probabilidad discreta.

5. Por qué el proceso es (o no es) Markoviano: la función de k

5.1. Definición formal de la propiedad de Markov

Un proceso estocástico $(X_t)_{t\geq 0}$ es Markoviano de primer orden si

$$\mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0) = \mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t)$$

para todo t y posibles realizaciones.

5.2. Por qué sin k el proceso no es Markoviano

La regla de ‘‘tres pares consecutivos $\Rightarrow$ cárcel’’ introduce memoria. Si el estado se define solo como (t, c) (tablero, casilla) entonces conocer (t, c) no es suficiente para decidir la probabilidad de la transición: hace falta saber cuántos pares consecutivos se llevan (información histórica). Por tanto, sin k la propiedad de Markov falla: la ley condicional del siguiente paso depende de información previa no contenida en (t, c).

5.3. La solución: incluir k (aumento del espacio de estados)

Teorema de aumento de orden (equivalencia de orden). Todo proceso de orden m puede representarse como un proceso de primer orden en un espacio de estados ampliado que contiene la información de los $m-1$ pasos anteriores. (Referencia clásica: *Norris, Markov Chains*, Teorema 1.2.1.)

Aplicando ese principio, definimos el estado como $S_t = (t, c, k)$. Entonces:

$$\mathbb{P}(S_{t+1} \mid S_t, S_{t-1}, \ldots) = \mathbb{P}(S_{t+1} \mid S_t),$$

porque k contiene la información esencial de la memoria relevante (conteo de pares consecutivos

5.4. Ejemplo ilustrativo

Considere la siguiente trayectoria:

1. $S_0 = (1, 0, 0)$ (sin pares).
2. Sale par $\Rightarrow S_1 = (1, 7, 1)$.
3. Sale par $\Rightarrow S_2 = (1, 14, 2)$.
4. En S_2, si sale par entonces S_3 será la cárcel: la probabilidad de S_3 depende únicamente de S_2 (y del resultado del lanzamiento), no de S_1 ni S_0, porque la información previa relevante está codificada en $k = 2$.

Conclusión: Al incluir k en la definición de estado hemos "proyectado" todas las dependencias pasadas relevantes dentro del estado actual, recuperando la propiedad de Markov de primer orden.

6. La matriz P como operador lineal y el espacio vectorial

6.1. Espacio vectorial

Las filas (o las columnas) de P son vectores en $\mathbb{R}^{144}$. El conjunto $\mathbb{R}^{144}$ es un espacio vectorial real: podemos sumar vectores y multiplicarlos por escalares.

6.2. P como aplicación lineal

La matriz P actúa como un operador lineal sobre $\mathbb{R}^{144}$:

$$\begin{cases} \text{distribución fila:} & x \mapsto xP, \\ \text{distribución columna:} & x \mapsto Px. \end{cases}$$

Esto permite usar técnicas de álgebra lineal (autovalores, autovectores, descomposición espectral) para analizar la dinámica de la cadena.

7. Autovalores y autovectores: por qué importan aquí

7.1. Definición

Un vector $v \neq 0$ y un escalar λ tales que

$$Pv = \lambda v$$

son respectivamente un autovector y su autovalor.

7.2. Interpretación para cadenas de Markov

- $\lambda = 1$ es siempre un autovalor de una matriz estocástica. El autovector izquierdo asociado a $\lambda = 1$ es la distribución estacionaria π que satisface

$$\pi^\top P = \pi^\top, \qquad \sum_i \pi_i = 1.$$

- Los autovalores λ con $|\lambda| < 1$ corresponden a modos transitorios que decaen con el tiempo. El segundo autovalor en módulo λ_2 (SLEM) controla la velocidad de mezcla.

8. Teoremas que sustentan el enfoque

8.1. Perron-Frobenius (versión para cadenas)

Si P es estocástica, no negativa, irreducible y aperiódica, entonces:

- El autovalor $\lambda = 1$ es simple.
- El autovector asociado π tiene entradas estrictamente positivas.
- P^n converge a la proyección $\mathbf{1}\pi^\top$ cuando $n \to \infty$.

Esto garantiza existencia, unicidad y convergencia hacia la distribución estacionaria.

8.2. Teorema espectral (finito)

La descomposición del operador lineal P en sus modos propios permite escribir la evolución de una distribución inicial como combinación de modos que decaen según $|\lambda|^n$.

9. Cómo se calcula π en la práctica

En el código se usa la factorización espectral:

1. Se calcula con eig los autovalores y autovectores de $P^\top$.
2. Se localiza el autovalor $\lambda \approx 1$ y se toma el autovector asociado v.
3. Se normaliza: $\pi = v / \sum_i v_i$ (y se usa la parte real si hay ruido numérico).

Matemáticamente, esto resuelve $P^\top v = v$, es decir $v^\top P = v^\top$.

10. Interpretaciones prácticas del espectro

- Modo estacionario: $\lambda = 1$ y π describen la frecuencia a largo plazo de cada estado.
- Gaps espectrales: $1 - |\lambda_2|$ controla la rapidez de convergencia (mixing time).
- Autovectores transitorios: muestran patrones de oscilación o agrupaciones de estados que comparten dinámica.

11. Condiciones de existencia y unicidad de π

- Existencia: siempre existe al menos una distribución estacionaria para matrices estocásticas.
- Unicidad: requiere irreducibilidad y aperiodicidad. Si la cadena tiene varias clases cerradas, hay múltiples estacionarias.
- Periodicidad: si hay periodo $d > 1$ el comportamiento asintótico puede ser oscilatorio; aun así existe solución de $\pi^\top P = \pi^\top$, pero la convergencia desde condiciones iniciales puede fallar.

12. Errores numéricos y consideraciones computacionales

12.1. Posibles fuentes de error

- Filas con suma 0 (división por cero en la normalización).
- Ruidos de coma flotante al acumular $1/36$ repetidas veces.
- Autovectores con pequeñas partes imaginarias por errores numéricos.
- Selección inexacta de $\lambda = 1$ si hay valores propios cercanos (tolerancia).

12.2. Recomendaciones prácticas

- Comprobar con np.allclose(P.sum(axis=1),1) que las filas suman 1.
- Detectar componentes fuertemente conexas en el grafo asociado para chequear irreducibilida
- Si N crece, usar matrices dispersas (scipy.sparse) y métodos iterativos (power method, eigs).
- Verificar la no-negatividad y que la ℓ_1-normalización dé vectores no negativos.

13. Interpretación con teoría de grafos

- El grafo dirigido $G = (V, E)$ con $V = \{0, \ldots, 143\}$ y pesos $w_{ij} = P_{ij}$ es la representación natural de la cadena.
- La existencia de caminos entre nodos determina clases comunicantes; la estacionaria se distribuye según la estructura de clases cerradas.
- La reversibilidad (balance detallado) raramente se cumple aquí; en general la cadena es *no reversible*.

14. Sesgos de modelado y supuestos

1. Dados justos e independientes: $\mathbb{P}(d_1, d_2) = 1/36$.
2. Reglas del juego implementadas exactamente como en el código (FORTUNA, cárcel, vuelvo a tirar).
3. Exclusión de estrategia de jugadores (no se modela compra/venta; solo movimiento).
4. Limitación del contador k a $\{0, 1, 2\}$ (si hubiera memoria más larga habría que ampliar el espacio).
5. Coarse-graining de casillas ‘‘vacías’’ o similares: decisiones de modelado que sesgan π respecto a métricas diferentes a frecuencia de visita.

15. Posibles fallos lógicos en la implementación (lista de chequeo)

- Filas con suma 0 tras bucles (debe detectarse y corregirse).
- Errores de mapeo en estado_a_indice (off-by-one).
- Implementación incorrecta de cambios de tablero por FORTUNA (mapeo de superposiciones).
- Selección errónea del autovector asociado a $\lambda = 1$ si hay autovalores muy cercanos.

Recomendado :

- Comprobaciones de irreducibilidad y aperiodicidad.
- Cálculo del gap espectral y mixing time cuantitativo.
- Análisis de sensibilidad y robustez numérica.
- Manejo explícito y seguro de filas con suma 0.
- Uso de matrices dispersas si la dimensión escala.

16. Conclusión

La construcción de la matriz P se hizo de forma consistente con la teoría: se modelaron todas las transiciones elementales (36 combinaciones de dados) y se acumuló su probabilidad en la entrada destino. La inclusión de k permite, por el principio de ampliación del espacio de estados, transformar un proceso con memoria en un proceso Markoviano de primer orden, habilitando el uso de la teoría espectral (autovalores, autovectores) y el teorema de Perron-Frobenius para razonamientos sobre existencia, unicidad y convergencia hacia la distribución estacionaria. No obstante, para una defensa académica completa, se recomiendan verificaciones numéricas y tests de sensibilidad que fortalezcan la argumentación.

Apéndice A: Fragmento de pseudo-código (para claridad)

```
# Definir N_ESTADOS = 2 * 24 * 3
P = zeros((N_ESTADOS, N_ESTADOS))

for tablero in [1,2]:
  for casilla in range(24):
    for k in range(3):
      estado_actual = estado_a_indice(tablero, casilla, k)
      for d1 in range(1,7):
        for d2 in range(1,7):
          prob = 1/36
          suma = d1 + d2
          nueva_casilla = (casilla + suma) % 24
          es_par = (d1 == d2)
          if es_par:
            nuevo_k = k + 1
            if nuevo_k >= 3:
              nueva_casilla = CARCEL_T1 if tablero==1 else CARCEL_T2
              nuevo_k = 0
              nuevo_estado = estado_a_indice(tablero, nueva_casilla, nuevo_k)
            else:
              nuevo_estado = estado_a_indice(tablero, casilla, nuevo_k)
          else:
            nuevo_k = 0
            # manejar FORTUNA y cambio de tablero si corresponde
            nuevo_estado = estado_a_indice(tablero_o_nuevo, nueva_casilla, nuevo_k)
          P[estado_actual, nuevo_estado] += prob

# normalizar filas por robustez
P = P / P.sum(axis=1, keepdims=True)
save("matriz_transicion_monopoly.npy", P)
```

Apéndice B: justificación de k

Justificación de la inclusión de k (contador de pares consecutivos): La regla de "tres pares seguidos $\Rightarrow$ cárcelïntroduce memoria en la dinámica: la probabilidad de la transición en el paso siguiente depende de resultados previos. Para aplicar la teoría de Markov de primer orden (y sus herramientas espectrales), ampliamos el espacio de estados para que incluya la información necesaria de la memoria (k). Este es un procedimiento estándar: cualquier proceso de orden m puede representarse como un proceso de primer orden en un espacio aumentado. Con esto, toda la información relevante para la transición futura está contenida en el estado actual (t, c, k) y la propiedad de Markov se recupera.

Referencias y lecturas recomendadas

- Norris, J. R. (1998). *Markov Chains*. Cambridge University Press.
- Levin, D. A., Peres, Y., & Wilmer, E. L. (2009). *Markov Chains and Mixing Times*. American Mathematical Society.
- Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations*.