Universidad Nacional de Colombia

Facultad de ciencias

Departamento de matemáticas

Cadenas de Markov 2025-II

Metodos para distribución estacionaria

Estudiantes:
Jose Miguel Acuña Hernandez
Andres Puertas Londoño
Guillermo Murillo Tirado

Docente:
Freddy Hernandez

Contenido

1. Marco Teórico

1.1. Cadenas de Markov y Distribución Estacionaria

Una cadena de Markov es un proceso estocástico $\{X_n\}_{n\geq 0}$ con espacio de estados finito $S = \{0, 1, \ldots, n-1\}$ que satisface la propiedad de Markov: $P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_{n+1} = j | X_n = i)$. La matriz de transición $P \in \mathbb{R}^{n\times n}$ está definida por $P_{ij} = P(X_{n+1} = j | X_n = i)$, donde cada elemento es no negativo y cada fila suma uno, constituyendo así una matriz estocástica.
La distribución estacionaria es un vector $\pi \in \mathbb{R}^n$ que satisface la ecuación fundamental $\pi = \pi P$ junto con las condiciones de normalización $\sum_{i=0}^{n-1} \pi_i = 1$ y no negatividad $\pi_i \geq 0$. Para matrices irreducibles y aperiódicas, el teorema de existencia y unicidad garantiza que existe una única distribución estacionaria π tal que $\lim_{n\to\infty} P^n = \mathbf{1}\pi^T$, donde $\mathbf{1}$ es el vector de unos.

1.2. Método del Autovector

1.2.1. Fundamentación Teórica

La distribución estacionaria π puede caracterizarse como el autovector izquierdo de la matriz P asociado al autovalor $\lambda = 1$, expresado mediante la relación $\pi P = \pi \Leftrightarrow \pi^T P^T = \pi^T$. Esto equivale a resolver el problema de autovalores $P^T \mathbf{v} = \mathbf{v}$, donde $\mathbf{v} = \pi^T$ es el autovector derecho de P^T correspondiente a $\lambda = 1$.
El Teorema de Perron-Frobenius establece que para una matriz estocástica irreducible P, el autovalor $\lambda = 1$ es simple y dominante con $|\lambda_i| \leq 1$ para $i \geq 2$, existe un único autovector izquierdo positivo π normalizado asociado a $\lambda = 1$, y todos los demás autovalores satisfacen $|\lambda_i| < 1$ si P es aperiódica. Esta fundamentación teórica garantiza la existencia y unicidad de la solución buscada.

1.2.2. Implementación Algorítmica

El algoritmo utilizado en este trabajo emplea la descomposición espectral directa mediante `np.linalg.eig(P.T)`, que calcula todos los autovalores y autovectores a través de tres etapas principales. Primero, se realiza la reducción a forma de Hessenberg usando transformaciones de Householder, seguida por la aplicación del algoritmo QR con desplazamientos para encontrar los autovalores, y finalmente el cálculo de autovectores por sustitución hacia atrás.

La complejidad temporal de este método es $\mathcal{O}(n^3)$ con contribuciones específicas: la reducción Hessenberg requiere aproximadamente $\frac{10n^3}{3}$ operaciones de punto flotante, las iteraciones QR demandan cerca de $6n^3$ operaciones en promedio, y el cálculo de autovectores consume alrededor de $3n^3$ operaciones, resultando en un total aproximado de $10n^3$ operaciones de punto flotante. La complejidad espacial es $\mathcal{O}(n^2)$ debido al almacenamiento de la matriz completa y los autovectores.

1.3. Método de Tiempos Medios de Primer Retorno

1.3.1. Fundamentación Teórica

El tiempo medio de primer retorno al estado j se define como $E[T_j] = E[\text{mín}\{n \geq 1 : X_n = j | X_0 = j\}]$. El teorema fundamental establece que para una cadena de Markov irreducible con distribución estacionaria π, se cumple la relación $\pi_j = \frac{1}{E[T_j]}$, indicando que la probabilidad estacionaria es inversamente proporcional al tiempo esperado de retorno.

Para derivar el sistema lineal correspondiente, sea $m_j^{(i)}$ el tiempo esperado de primer retorno al estado j comenzando desde el estado $i \neq j$. Entonces se cumple $m_j^{(i)} = 1 + \sum_{k \neq j} P_{ik} m_j^{(k)}$, lo que genera un sistema lineal de dimensión $(n-1) \times (n-1)$ expresado como $(I - P_{-j})\mathbf{m}_j = \mathbf{1}$, donde P_{-j} es la matriz P con la fila y columna j eliminadas, y $\mathbf{m}_j$ es el vector de tiempos de retorno desde todos los estados excepto j. El tiempo medio de retorno desde j se calcula entonces como $E[T_j] = 1 + \sum_{k \neq j} P_{jk} m_j^{(k)}$.

1.3.2. Implementación y Análisis de Complejidad

El algoritmo estándar procede iterativamente para cada estado $j = 0, 1, \ldots, n-1$, construyendo la matriz reducida $P_{-j} \in \mathbb{R}^{(n-1)\times(n-1)}$, resolviendo el sistema lineal $(I - P_{-j})\mathbf{m}_j = \mathbf{1}$, calculando $E[T_j] = 1 + \mathbf{p}_{j,-j}^T \mathbf{m}_j$, y finalmente obteniendo $\pi_j = 1/E[T_j]$.

La construcción de P_{-j} requiere eliminar la fila j copiando $n \times (n-1)$ elementos y eliminar la columna j copiando $(n-1) \times (n-1)$ elementos, resultando en una complejidad $\mathcal{O}(n^2)$ para cada matriz reducida. La resolución del sistema lineal se realiza mediante factorización LU, donde el costo de factorización es $\sum_{k=0}^{n-2}(n-1-k)^2 \approx \frac{(n-1)^3}{3} = \mathcal{O}(n^3)$, la sustitución hacia adelante ($L\mathbf{y} = \mathbf{1}$) tiene costo $\sum_{i=0}^{n-2} i = \frac{(n-1)(n-2)}{2} = \mathcal{O}(n^2)$, y la sustitución hacia atrás ($U\mathbf{m}_j = \mathbf{y}$) también requiere $\mathcal{O}(n^2)$ operaciones, resultando en un costo total por sistema de $\mathcal{O}(n^3)$.

Al considerar que este proceso debe repetirse para n estados, la complejidad total del algoritmo es $n \times \mathcal{O}(n^3) = \mathcal{O}(n^4)$. En términos de operaciones de punto flotante específicas, cada factorización LU requiere aproximadamente $\frac{2(n-1)^3}{3}$ flops, las sustituciones demandan cerca de $2(n-1)^2$ flops, resultando en un total por estado de aproximadamente $\frac{2n^3}{3}$ flops y un total para todo el algoritmo de cerca de $\frac{2n^4}{3}$ flops.

1.4. Análisis Comparativo de Complejidad Computacional

La comparación entre ambos métodos revela diferencias significativas en términos de eficiencia computacional. El método del autovector mediante descomposición espectral presenta complejidad temporal $\mathcal{O}(n^3)$ y espacial $\mathcal{O}(n^2)$ con excelente estabilidad numérica, mientras que el método de tiempos medios en su implementación estándar tiene complejidad temporal $\mathcal{O}(n^4)$ y espacial $\mathcal{O}(n^2)$ manteniendo también excelente estabilidad.

El análisis de constantes multiplicativas muestra que para una matriz $n \times n$, el método del autovector requiere aproximadamente $10n^3$ operaciones de punto flotante distribuidas entre la reducción Hessenberg ($\frac{10n^3}{3}$ flops), el algoritmo QR ($6n^3$ flops en promedio), y el cálculo de autovectores ($3n^3$ flops). En contraste, el método de tiempos en su implementación estándar demanda aproximadamente $0.67n^4$ flops, considerando $\frac{2(n-1)^3}{3}$ flops por factorización LU multiplicado por n estados.

La razón de tiempos de ejecución entre ambos métodos puede expresarse como $\frac{\text{Tiempo(Tiempos)}}{\text{Tiempo(Autovector)}} \approx \frac{0.67n^4}{10n^3} = 0.067n$, lo que implica que para matrices de tamaño $n = 100$ el método de tiempos es aproximadamente 6.7 veces más lento, para $n = 500$ es 33.5 veces más lento, y para $n = 1000$ alcanza una razón de 67 veces más lento que el método del autovector.

1.4.1. Efectos de Optimizaciones Computacionales

Las bibliotecas BLAS/LAPACK proporcionan optimizaciones significativas incluyendo manejo eficiente de caché y vectorización, paralelización automática para operaciones matriciales, y un speedup típico de $10 - 100$ veces comparado con implementaciones ingenuas. Varios factores afectan las mediciones empíricas: para tamaños pequeños ($n < 100$) los términos de orden inferior dominan el comportamiento asintótico, la jerarquía de memoria hace que matrices que

caben en caché L2/L3 sean significativamente más rápidas, BLAS puede utilizar múltiples threads automáticamente, y el overhead del intérprete constituye una constante aditiva significativa para valores pequeños de n.

1.4.2. Consideraciones de Estabilidad Numérica

El condicionamiento de la matriz P afecta sustancialmente la precisión de los resultados. El número de condición $\kappa(P) = \frac{\sigma_{\text{máx}}}{\sigma_{\text{mín}}}$ determina la calidad numérica: cuando $\kappa(P) < 10^{12}$ es posible alcanzar precisión de máquina, mientras que $\kappa(P) > 10^{12}$ resulta en pérdida significativa de dígitos significativos. La propagación de errores se rige por la desigualdad $\frac{\|\Delta\pi\|}{\|\pi\|} \leq \kappa(P)\frac{\|\Delta P\|}{\|P\|}$, donde el método del autovector es generalmente más robusto ante perturbaciones en P que el método de tiempos medios.
Para la selección algorítmica, matrices pequeñas ($n \leq 100$) se benefician de la descomposición espectral directa donde el overhead de configuración es despreciable y se obtiene máxima precisión numérica. Para matrices medianas ($100 < n \leq 1000$) se recomienda la descomposición espectral cuando se requiere precisión, evitando el método de tiempos no optimizado. En matrices grandes ($n > 1000$) conviene considerar métodos iterativos especializados que puedan explotar estructura dispersa cuando sea aplicable.

2. Función de las 3 Cadenas

2.1. Caminata Aleatoria Simple

La caminata aleatoria simple implementada corresponde a un proceso en una estructura cíclica de n estados, donde las probabilidades de transición están definidas por p (probabilidad de avanzar al siguiente estado) y q = 1-p (probabilidad de retroceder al estado anterior). La matriz de transición P presenta la forma donde cada estado i puede transitar al estado (i+1) mod n con probabilidad p, o al estado (i-1) mod n con probabilidad q.
La implementación se realiza mediante la función `generar_caminata_aleatoria(n, p)` que construye la matriz de transición correspondiente:

```python
def generar_caminata_aleatoria(n, p):
    """
    Genera matriz de transición para caminata aleatoria cíclica.

    Args:
        n: Número de estados
        p: Probabilidad de ir al siguiente estado (q = 1-p al anterior)

    Returns:
        P: Matriz de transición
    """
    q = 1 - p
    P = np.zeros((n, n))

    for i in range(n):
        P[i, (i + 1) % n] = p  # Probabilidad de ir al siguiente (mod n para ciclo)
        P[i, (i - 1) % n] = q  # Probabilidad de ir al anterior (mod n para ciclo)

    return P
```

La configuración experimental utiliza un rango de estados desde n = 10 hasta n = 560 con incrementos de 50, y cuatro valores de probabilidad p = [0.2, 0.4, 0.6, 0.8]. Para cada combinación de parámetros, se genera la matriz de transición correspondiente y se aplican ambos métodos de cálculo de distribución estacionaria, midiendo los tiempos de ejecución mediante `perf_counter()` y calculando el error de convergencia entre los resultados obtenidos por ambos métodos.

3. Bloques de Código que Generan los Cambios en las Cadenas

4. Resultados de la Comparación con las Gráficas