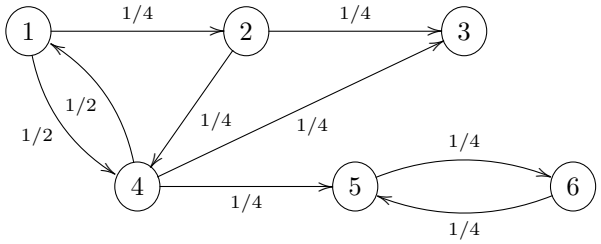
Sistemas Estocásticos 2024-25

Examen Final (Primera parte)

Sistemas a tiempo discreto

A. Consideremos la cadena de Markov definida en el siguiente diagrama:



Se asuma que cada estado tiene un arco hacia el estado mismo con la probabilidad necesaria para hacer que la suma de las probabilidades en salida sea 1.

i) Determinar la matriz de transición de la cadena

ii) Determinar la(s) clase(s) comunicante(s)

iii) Determinar la(s) clase(s) absorbentes

iv) Determinar la probabilidad que saliendo de 1 se llegue al estado 3

v) Determinar (sin necesidad de cálculo) la probabilidad que saliendo de 1 se llegue a 5

vi) Determinar (sin necesidad de cálculo) el tiempo medio necesario para ir de 1 a 3

vii) Determinar (sin necesidad de cálculo) la probabilidad de pasar por 3 saliendo de 5

B. Un *paseo aleatorio por popularidad* es una cadena de Markov en un grafo definida como sigue. Sea $G = (V, E)$ un grafo no dirigido, con $V = \{1, \ldots, n\}$ y $E \subseteq V \times V$. Sea d_i el *grado* del nodo i, es decir, el número de vecinos de i:

$$d_i = \left|\{j|(i,j) \in E\}\right| \tag{1}$$

El paseo, a tiempo discreto, se desarrolla como sigue:

a. Al instante t el paseo se encuentra en un nodo i

b. Al instante $t+1$ el paseo se mueve de i a uno de sus vecinos. La probabilidad que se mueva al vecino j es proporcional a d_j^α, donde α es un parámetro que caracteriza el paseo. Es decir la probabilidad de moverse de i a j es

$$p(j|i) = \begin{cases} \dfrac{d_j^\alpha}{\displaystyle\sum_{k:(i,k)\in E} d_k^\alpha} & \text{si } (i,j) \in E \\ 0 & \text{si } (i,j) \notin E \end{cases} \tag{2}$$

En este apartado se implementará una simulación del paseo por popularidad. En particular analizaremos la velocidad con que se visita el grafo en función de α.

Las pruebas se harán con grafo aleatorios de tipo Barabasi-Albert. El fichero `barabasi.py` define la función `Barabasi(n,m)`, que proporciono con este enunciado, crea un grafo del tipo que necesitamos. La función de creación recibe dos parámetros: el número de nodos del grafo, n, y el número de arcos que cada nodo genera cuando es creado, m. El número total de arcos (dirigidos) del grafo será $2nm$. El valor de m no es crítico, siempre que sea $m \ll n$. Para grafos con 1.000 nodos o más, un valor $m = 10$ debería funcionar bien, para grafos más pequeños, un valor $m = 5$ es más adecuado. Grafos con menos de 100 nodos se pueden usar al principio para ver si todo funciona, pero no son fiables para ejecutar las pruebas: para esas es aconsejable un grafo con por lo menos 1000 nodos. La función devuelve una representación muy minimalista: una lista de listas con sólo la estructura del grafo. El elemento k de la lista es la lista de adyacencia del nodo k, y contiene los índices de los nodos a que k está conectado.

En estos grafos se ejecutarán las simulaciones. Para obtener resultados más estables, se efecturá un número `NTESTS` de paseos por cada valor de α. En cada simulación se empezará desde u nodo aleatorio y se pasará de un nodo a otro según las probabilidades definidas arriba. Se mantendrá, por cada nodo, un indicador que indica si el nodo ya ha sido visitado o no y por cada t e contarán los nodos que se han visitado. La simulación continuará por un tiempo `Tmax` que será por lo menos 10 veces el número de nodos del grafo (si durante la simulación se llega a un punto en que ya se ha visitado todo el grafo—raro, pero puede pasar—, la simulación se puede interrumpir). El resultado de una simulación será un array de `Tmax` elementos con, en el elemento t, la fracción del grafo visitada al tiempo t.

Esta simulación se repitirá `NTESTS` veces y se calculará la media de las fracciones del grafo visitada. Para evitar dependencias de características específica de un grafo, cada simulación se hará en un grafo distinto generado con los mismos parámetros.

Esto nos da la fracción de grafo visitada en función del tiempo por un valor de α. Ahora habrá que repitir la simulación para varios valores de α y ver para que valor la visita es más rápida. Aconsejo empezar usando los valores $\alpha = [-2, -1, -0.5, 0, 1, 2, 5]$ y ver un poco como van las cosas. El resultado a presentar será una gráfica con la fracción de grafo visitada en función del tiempo para varios valores representativos de α. Si es el caso, podéis intentar con otros valores.

Valores razonables son unos $n \sim 1.000$ nodos para el número de nodos del grafo, $m \sim 10$, `NTEST` más o menos del órden de magnitud del número de nodos y α en el intervalo $[-2, 2]$ (ese es el intervalo donde creo que pasan las cosas más interesantes, pero si parece que algo curioso pasa cerca de los extremos, no es una mala idea alargar un poco el intervalo).

Opcional: una gráfica interesante es usar α como variable independiente en lugar del tiempo, es decir, elegir una gracción de grafo (p.ej. 0.5) y hacer una gráfica del tiempo necesario para visitar esta gracción de grafo en función de α.

Para la entrega

Entregar un fichero llamado

`STOCH1-2425-<1er-apellido>{.zip,.tar.gz,.tgz,.tar}`

En formado `tar` (preferido), `tar` comprimido o `zip` (si no hay otra posibilidad...). Este fichero contendrá:

i) Un fichero `report.pdf` (forato `pdf`, por favor: evitar otros formatos) con las respuestas a la primera parte, las gráficas de la segunda parte, un comentario sobre como se ha efectuado la simulación, y una discusión de los resultados (¿Cuál es al valor de α que da la exploración más rápida? ¿Por qué pensáis que es este valor y no otros?)

ii) El (los) fichero(s) con el código de la simulación. El lenguaje de programación que usáis es de viestra elección. EL código que proporciono para crear los grafos Barabasi está escrito en `python`, pero no es obligatorio usar este lenguaje.

Para Dudas

Me podéis escribir en cualquier momento a `simone.santini@uam.es`.