Dispense del corso di Laboratorio di Sistemi Stocastici

Dott. Marco Caliari

a.a. 2013/14

Questi appunti non hanno alcuna pretesa di completezza. Sono solo alcune note ed esercizi che affiancano il corso di Sistemi Stocastici. Sono inoltre da considerarsi in perenne "under revision" e pertanto possono contenere discrepanze, inesattezze o errori. Gli esempi ed gli esercizi proposti sono stati implementati e risolti in GNU Octave 3.4.x. Matlab® potrebbe dare risultati diversi.

Per l'esame, dopo il colloquio orale si deve discutere la risoluzione degli esercizi di questa dispensa contrassegnati da un punto interrogativo. Tipicamente, si porta all'esame la stampa dei codici di risoluzione e dei risultati ottenuti.

Indice

Capitolo 0

Algebra lineare

0.1 Matrici a banda

Per generare una matrice diagonale si usa il comando

```
diag([1,2,3,4])
```

il cui risultato è

```
ans =

   1   0   0   0
   0   2   0   0
   0   0   3   0
   0   0   0   4
```

È possibile specificare la posizione della diagonale:

```
> diag([1,2,3,4])+diag([5,6,7],1)+diag([8,9,10],-1)
ans =

   1   5    0   0
   8   2    6   0
   0   9    3   7
   0   0   10   4
```

0.2 Matrici di Toeplitz

Una matrice di *Toeplitz* ha la forma

$$\begin{bmatrix} c_1 & r_2 & r_3 & \dots & r_n \\ c_2 & c_1 & r_2 & \dots & r_{n-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & \ddots & \ddots & r_2 \\ c_n & c_{n-1} & \dots & c_2 & c_1 \end{bmatrix}$$

Per generarla, si usa il comando `toeplitz(c,r)`, ove `c` $= [c_1, c_2, \dots, c_n]$ e `r` $= [c_1, r_2, \dots, r_n]$. Per esempio,

```
> toeplitz([0,1/5,0,0],[0,4/5,0,0])
ans =

   0.00000   0.80000   0.00000   0.00000
   0.20000   0.00000   0.80000   0.00000
   0.00000   0.20000   0.00000   0.80000
   0.00000   0.00000   0.20000   0.00000
```

0.3 Sistemi lineari e autovettori

Il sistema lineare $Ax = b$ si risolve con il comando

```
x = A\b
```

ove il risultato è il vettore *colonna* `x`. Se invece si intende risolvere il sistema lineare $b^{\mathrm{T}} = yA$, si deve usare il comando

```
y = b'/A
```

ove il risultato è il vettore *riga* `y`. Alternativamente, si riscrive il problema come $A^{\mathrm{T}}y^{\mathrm{T}} = b$. Nel seguito, l'orientamento di un vettore (riga o colonna) sarà chiaro dal contesto.

Il problema della ricerca dell'autovettore *sinistro* v della matrice P relativo all'autovalore 1 si può risolvere considerando il problema $P^{\mathrm{T}}v^{\mathrm{T}} = v^{\mathrm{T}}$. Per la soluzione, si usa il comando

```
[V,D] = eig(P')
```

D'ora in poi per autovettore intenderemo autovettore sinistro. Le matrici V, D e P^{T} soddisfano l'equazione $P^{\mathrm{T}}V = VD$, con $D = \mathrm{diag}(d_1, d_2, \ldots)$ matrice diagonale degli autovalori e V matrice le cui colonne sono gli autovettori. L'autovettore v, se unico, corrisponde alla colonna i-esima di V (trasposta), se $d_i = 1$. Ovviamente gli autovettori sono definiti a meno di una costante moltiplicativa. Una matrice di elementi *non negativi* e le cui righe hanno somma 1 si dice *stocastica*. Vale la pena ricordare il seguente

Teorema (Markov–Kakutani)**.** *Una matrice stocastica ha almeno un autovettore, a elementi dello stesso segno, relativo all'autovalore 1.*

Una parte del teorema è banale: se P è una matrice stocastica, allora dato il vettore $v = [1, 1, \ldots, 1]$ si ha

$$Pv^{\mathrm{T}} = v^{\mathrm{T}}$$

Dunque, 1 è autovalore di P e quindi anche di P^{T} (gli autovalori delle matrici trasposte coindidono). Quindi esiste w tale che

$$P^{\mathrm{T}}w^{\mathrm{T}} = w^{\mathrm{T}} \Rightarrow wP = w$$

Dunque P ha l'autovalore 1 relativo ad un autovettore sinistro.

Teorema (Cerchi di Gerschgorin)**.** *Sia $A = (a_{ij})$ una matrice quadrata di dimensione n. Allora gli autovalori sono compresi nella regione*

$$\left(\bigcup_{i=1}^{n} R_i\right) \cap \left(\bigcup_{j=1}^{n} C_j\right)$$

ove

$$R_i = \left\{ z \in \mathbb{C}\colon\ |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \right\}, \quad C_j = \left\{ z \in \mathbb{C}\colon\ |z - a_{jj}| \leq \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| \right\}$$

Dimostrazione. Dimostriamo che gli autovalori di A stanno in $\cup_i R_i$: seguirà che gli autovalori di A^{T} stanno in $\cup_j C_j$ e poiché i due insiemi di autovalori coincidono, staranno nell'intersezione. Sia λ un autovalore e v l'autovettore associato, normalizzato in modo che

$$\max_k |v_k| = |v_i| = 1, \quad 1 \leq i \leq n$$

allora

$$\sum_{j=1}^{n} a_{ij}u_j - \lambda u_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}u_j + a_{ii}u_i - \lambda u_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}u_j + (a_{ii} - \lambda)u_i = 0$$

Passando ai moduli

$$|a_{ii} - \lambda| = \left| \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}u_j \right| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|\,|u_j| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|$$

e dunque $\lambda \in R_i$. □

P ha righe la cui somma vale 1. Quindi i dischi R_i del teorema di Gerschgorin hanno tutti centro nel segmento $[0, 1]$, passano per 1 e sono contenuti del disco unitario. Quindi, tutti gli autovalori di P (e quindi anche di P^{T}) hanno modulo minore o uguale a 1. Per calcolare l'autovettore destro di P^{T} (sinistro di P) relativo all'autovalore 1 è possibile dunque usare il metodo delle *potenze*. Tale metodo potrebbe risultare estremamente vantaggioso nel caso di matrici sparse e di grande dimensione, per le quali il costo del calcolo di tutti gli autovalori/autovettori potrebbe essere proibitivo (cubico nella dimensione). Usando il linguaggio dei sistemi stocastici, applicare il metodo delle potenze significa calcolare la probabilità invariante mediante il calcolo della probabilità limite. Riassumiamo il legame tra il linguaggio dell'algebra lineare (numerica) e dei sistemi stocastici

`invariantPotenze`

1. Una matrice di transizione P ha almeno una probabilità invariante (P ha almeno un autovettore, a elementi dello stesso segno, relativo all'autovalore 1).

2. Se la probabilità invariante non è unica, il metodo delle potenze, se converge, converge ad una combinazione lineare degli autovettori relativi all'autovalore (di molteplicità maggiore di 1) 1.

3. Se la catena ad essa associata è irriducibile, esiste un'unica probabilità invariante. Dunque la matrice ha un solo autovalore pari ad 1, ma potrebbe avere l'autovalore -1, come per

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

4. L'irriducibilità non è sufficiente affinché esista la probabilità limite. Cioè non è sufficiente affinché il metodo delle potenze converga nel calcolo dell'autovettore relativo all'autovalore 1.

5. Se P è regolare (e dunque irriducibile) esiste un'unica probabilità invariante e limite. Per definizione, il metodo delle potenze vi converge (è l'unico caso). Significa che gli autovalori diversi da 1 sono in modulo minori di 1.

0.4 Altri comandi utili

La somma sulle righe di una matrice A si ottiene con il comando

```
sum(A,2)
```

Una matrice stocastica le cui colonne hanno somma 1 si dice *bistocastica*. La somma sulle colonne di una matrice A si ottiene con il comando

```
sum(A)
```

La norma-1 di un vettore v (somma dei moduli degli elementi) si ottiene con il comando `norm(v,1)`. Chiameremo un vettore *normalizzato* se tutti i suoi elementi sono *positivi* e la sua norma-1 è 1.

Il comando `repmat` permette di formare matrici ripetendo, per righe o per colonne, un vettore dato. Per esempio,

```
repmat([1,2,3],3,1)
```

produce

```
ans =

   1   2   3
   1   2   3
   1   2   3
```

0.5 Risoluzione di un sistema lineare sottodeterminato

Supponiamo che S sia una matrice stocastica di oridne n con un solo autovettore relativo all'autovalore 1. Allora, $S^{\mathrm{T}}v^{\mathrm{T}} - v^{\mathrm{T}} = (S^{\mathrm{T}} - I)v^{\mathrm{T}} = 0$, con v non nullo. Possiamo considerare la fattorizzazione $QRP^{\mathrm{T}} = A$ (ove $A = S^{\mathrm{T}} - I$)

che si ottiene con il comando `qr`. La matrice triangolare superiore R avrà gli elementi in diagonale ordinati in maniera decrescente per modulo. Poiché le matrici Q e P sono ortogonali (non singolari), R deve avere $r_{nn} = 0$ (visto che A è singolare). Allora, definendo $y = P^{\mathrm{T}}x$, è possibile risolvere il sistema lineare $\tilde{R}y = \tilde{b}$, $b = 0$, ponendo $\tilde{r}_{nn} = 1$ e $\tilde{b}_n = 1$. A quel punto, $x = Py$ è soluzione (non nulla, non normalizzata) di $Ax = 0$.

Nelle stesse ipotesi, si può risolvere il sistema rettangolare

$$\begin{bmatrix} & & A & & \\ 1 & 1 & \dots & 1 & 1 \end{bmatrix} x = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow x^{\mathrm{T}} \begin{bmatrix} & & 1 \\ & S - I & \vdots \\ & & 1 \end{bmatrix} = \begin{bmatrix} 0 & \dots & 0 & 1 \end{bmatrix}$$

di rango n con il risolutore standard `\` (oppure `/`). La soluzione risulta già normalizzata.

0.6 Esercizi

1. Implementare una function che determini se una matrice è stocastica, a meno dell'errore di macchina.

`isstoc`

2. Si implementino due functions che costruiscono le matrici

$$\begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ q & 0 & p & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & q & 0 & p \\ 0 & \dots & \dots & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & p_1 & 0 & \dots & \dots & 0 \\ q_1 & r_2 & p_2 & 0 & \dots & 0 \\ 0 & q_2 & r_3 & p_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & q_{n-2} & r_{n-1} & p_{n-1} \\ 0 & \dots & \dots & 0 & q_{n-1} & r_n \end{bmatrix}$$

`catenanm` di ordine n ($n \geq 3$).

3. Data la matrice

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

si calcoli, se esiste, l'autovettore sinistro normalizzato relativo all'autovalore 1.

4. Si ripeta l'esercizio precedente con la matrice

$$P = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

5. Si ripeta l'esercizio precedente con la matrice

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6. Si generi una matrice di elementi random di ordine 5. Scalandone opportunamente le righe, la si renda stocastica.

`randstoc`

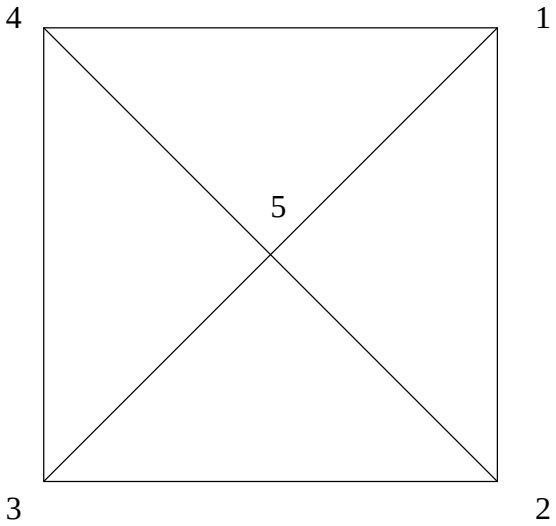
7. Implementare una function che calcoli, se esiste, l'unico autovettore sinistro normalizzato relativo all'autovalore 1 di una matrice stocastica, usando il comando `eig`.

`invariantVD`

8. Implementare una function che calcoli, se esiste, l'unico autovettore sinistro normalizzato relativo all'autovalore 1 di una matrice stocastica, usando il comando `qr`. `invariantQR`

9. Generare un vettore di lunghezza 10 di numeri casuali compresi tra -2 e 3.

10. Si costruisca la matrice di transizione P relativa alla passeggiata casuale sul grafo



Si verifichi che $v = [3/16, 3/16, 3/16, 3/16, 4/16]$ è reversibile per P e dunque invariante.

11. Si generi una matrice di transizione

$$P = \begin{bmatrix} r_1 & p_1 & 0 & \dots & \dots & 0 \\ q_1 & r_2 & p_2 & 0 & \dots & 0 \\ 0 & q_2 & r_3 & p_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & q_{n-2} & r_{n-1} & p_{n-1} \\ 0 & \dots & \dots & 0 & q_{n-1} & r_n \end{bmatrix}$$

di ordine $n = 10$, con $p_1 \neq 0$ e $q_{n-1} \neq 0$. Si verifichi che l'unica probabilità invariante è un multiplo di

$$\xi = \left[1, \frac{p_1}{q_1}, \frac{p_1 p_2}{q_1 q_2}, \dots, \frac{p_1 \cdot \ldots \cdot p_{n-1}}{q_1 \cdot \ldots \cdot q_{n-1}}\right]$$

12. Costruire una matrice bistocastica di ordine 4, non simmetrica e con tutti gli elementi diversi da 0. Verificare che la probabilità invariante è $v = [1/4, 1/4, 1/4, 1/4]$. *(Sugg.: usare il comando* `magic`*...)*

13. Dimostrare che se esiste una probabilità limite per una catena, essa è anche invariante.

Capitolo 1

Algoritmo di Metropolis

1.1 Generazione di matrici stocastiche simmetriche irriducibili

Supponiamo di dover costruire una matrice di transizione simmetrica irriducibile di ordine n *qualunque*. Possiamo partire dalla prima riga (e dunque colonna) con un vettore random normalizzato `randstocsym`

```
v = rand(1,n);
v = v/sum(v);
Q(1,:) = v;
Q(:,1) = v';
```

Per la seconda riga (e colonna) abbiamo bisogno di un vettore w tale che $q_{21} + w_1 + w_2 + \ldots + w_{n-1} = 1$. A partire da un vettore random $\tilde{w}$ dobbiamo normalizzarlo in modo che $q_{21} + \alpha(\tilde{w}_1 + \tilde{w}_2 + \ldots + \tilde{w}_{n-1}) = 1$. Dunque $w = \tilde{w}\alpha = \tilde{w}\left(\frac{1-q_{21}}{\tilde{w}_1+\tilde{w}_2+\ldots+\tilde{w}_{n-1}}\right)$. Per la terza riga (e colonna) è $z = \tilde{z}\beta = \tilde{z}\left(\frac{1-(q_{31}+q_{32})}{\tilde{z}_1+\tilde{z}_2+\ldots+\tilde{z}_{n-2}}\right)$. E così via. Poiché non è garantito che tutti gli elementi della matrice siano non negativi, serve una correzione finale. La matrice che si ottiene sarà anche "quasi certamente" irriducibile. L'algoritmo per la costruzione di una matrice stocastica simmetrica è riportato in Tabella 1.1.

Una "banale" matrice di transizione simmetrica irriducibile di ordine n è quella con elementi tutti uguali a $1/n$.

1.2 Algoritmo di Metropolis

Data una distribuzione di probabilità v di lunghezza n ed una matrice Q di ordine n stocastica simmetrica irriducibile, la catena associata alla matrice

```
function Q = randstocsym(n)
%
% Q = randstocsym(n)
%
% Genera una matrice stocastica simmetrica random di ordine n
Q = zeros(n);
v = rand(1,n);
v = v / sum(v);
for i = 1:n-1
  Q(i,i:n) = v;
  Q(i:n,i) = v';
  v = rand(1,n-i);
  v = v * ((1 - sum(Q(i + 1,1:i))) / sum(v));
end
Q(n,n) = v;
Q = Q - 2 * min([min(Q(:)),0]);
Q = Q / (sum(Q(:))/n);
```

Tabella 1.1: Algoritmo per la generazione di una matrice stocastica simmetrica

di transizione $P = (p_{ij})$ definita da

$$p_{ij} = \begin{cases} q_{ij} & v_j \geq v_i,\ i \neq j \\ q_{ij}\dfrac{v_j}{v_i} & v_j < v_i,\ i \neq j \\ 1 - \displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{n} p_{ij} & i = j \end{cases} \tag{1.1}$$

ammette $\hat{v}$ come unica probabilità invariante ($\hat{v}$ è v normalizzata). La costruzione di P non richiede che v sia normalizzata. L'algoritmo riportato in Tabella 1.2 costruisce in maniera efficiente, per mezzo del comando `find`, la matrice di transizione P irriducibile e con la probabilità invariante $\hat{v}$ data (algoritmo di Metropolis). Se v non è uniforme, la catena di Markov di matrice di transizione P è regolare e dunque

$$\lim_{m\to\infty} p_{ij}^{(m)} = \hat{v}_j = \left(\sum_{i=1}^{n} w_i\right)\hat{v}_j = \lim_{m\to\infty} (wP^m)_j$$

per qualunque probabilità w.

```
function P = metropolis(v,Q)
%
% P = metropolis(v)
% P = metropolis(v,Q)
%
n = length(v);
if (nargin == 1)
  Q = randstocsym(n);
end
V = repmat(v,n,1);
W = repmat(v',1,n);
P = zeros(n);
P(V>=W) = Q(V>=W);
P(V<W) = Q(V<W) .* V(V<W) ./ W(V<W);
P = P - diag(diag(P));
d = 1 - sum(P,2);
P = P + diag(d);
```

Tabella 1.2: Algoritmo di Metropolis

Il calcolo di P^m secondo l'algoritmo

$$P^k = PP^{k-1}, \quad k = 1, 2, \ldots, m$$

costa $\mathcal{O}(mn^3)$, se n è l'ordine della matrice P. Il calcolo di wP^m secondo l'analogo algoritmo costa invece $\mathcal{O}(mn^2)$.

1.2.1 Simulated annealing

Dato uno spazio degli stati E ed una funzione $H\colon E \to \mathbb{R}$, si considera la distribuzione di probabilità di elementi $\hat{v}_i^\varepsilon$, $v^\varepsilon = \mathrm{e}^{-H(i)/\varepsilon}$, $\hat{v}^\varepsilon$ normalizzata. Tramite l'algoritmo di Metropolis, si può costruire una catena con matrice di transizione P_ε avente $\hat{v}^\varepsilon$ come probabilità invariante. La matrice $P_\varepsilon = (p_{ik}^\varepsilon)$ è definita da

$$p_{ij}^\varepsilon = \begin{cases} q_{ij} & H(j) \le H(i),\ j \ne i \\ q_{ij}\mathrm{e}^{-(H(j)-H(i))/\varepsilon} & H(j) > H(i),\ j \ne i \\ 1 - \displaystyle\sum_{\substack{j=1 \\ j\ne i}}^{n} p_{ij}^\varepsilon & j = i \end{cases}$$

Il calcolo di v^ε può dare problemi di overflow. Per esempio, per $H(1) = -1$ e $H(2) = 1$ e $\varepsilon = 10^{-3}$ si ha

```
H(1) = -1;
H(2) = 1;
epsilon = 1e-3;
v = exp(-H/epsilon);
```

da cui

```
v =
   Inf     0
```

Per evitare questi problemi, conviene usare la formula equivalente

$$\hat{v}_i^\varepsilon = \frac{\mathrm{e}^{-H(i)/\varepsilon}}{\sum_{j=1}^n \mathrm{e}^{-H(j)/\varepsilon}} = \left(\frac{\sum_{j=1}^n \mathrm{e}^{-H(j)/\varepsilon}}{\mathrm{e}^{-H(i)/\varepsilon}} \right)^{-1} = \left(\sum_{j=1}^n \mathrm{e}^{-(H(j)-H(i))/\varepsilon} \right)^{-1} \tag{1.2}$$

che si può calcolare mediante

```
v = 1./sum(exp(-(repmat(H',1,2)-repmat(H,2,1))/epsilon))
```

da cui

```
v =
     1     0
```

Se H assume il suo valore minimo nell'unico punto $\bar{i} \in E$, allora, da (1.2), si ha

$$\lim_{\varepsilon \to 0} \hat{v}_j^\varepsilon = \delta_{\bar{i}j}$$

Questa osservazione suggerisce l'algoritmo di ottimizzazione globale seguente: se si deve determinare il minimo assoluto di una funzione H su un insieme E, a partire da una qualunque matrice di transizione Q irriducibile simmetrica su E si simula la catena di Markov associata alla matrice di transizione P, *senza calcolare esplicitamente* la matrice (vedremo al capitolo seguente come operare in pratica). Infatti, se ε è piccolo, per n grande la catena si trova con grande probabilità in uno stato i in cui H prende un valore molto vicino al minimo. Questa procedura è effettivamente molto utile in alcuni problemi di ottimizzazione in cui l'insieme E ha una cardinalità talmente grande che la strategia ovvia di calcolare H su tutti gli stati $i \in E$ e confrontarli è inutilizzabile. Si può anzi pensare di costruire una catena in cui ad ogni transizione si considera un valore di ε più piccolo. Ovvero, più precisamente, di scegliere una successione $\{\varepsilon_n\}_n$ decrescente a 0 e di considerare la catena di Markov *non omogenea* associata alla matrice di transizione di elementi $p_{ij}^{\varepsilon_n}$. Questa procedura viene chiamata *simulated annealing*. Un valore di ε_n troppo piccolo all'inizio della simulazione potrebbe portare la catena ad assumere gli stati in corrispondenza a minimi *locali*.

epsmet

1.3 Il problema del commesso viaggiatore

Supponiamo di avere N città connesse tra loro. Un *circuito* è un percorso che, partendo da una città, le visita tutte una sola volta e torna alla città iniziale. Fissata, per comodità, la città iniziale, il numero di circuiti possibili è $n = (N-1)!$. Dato un circuito $\omega = (i_2, i_3, \ldots, i_N)$, definiamo *adiacente* un circuito in cui i soli i_h e i_k, $h \neq k$, risultino scambiate di posto. Il numero di circuiti adiacenti ad un circuito dato ω sono $\binom{N-1}{2} = (N-1)(N-2)/2$. La matrice di transizione Q associata alla passeggiata su questo grafo avrà elementi

$$q_{\omega\omega'} = \begin{cases} \dfrac{2}{(N-1)(N-2)} & \text{se } \omega \text{ e } \omega' \text{ sono adiacenti} \\ 0 & \text{altrimenti} \end{cases}$$

1.4 Esercizi

1. Data una distribuzione di probabilità v si costruisca una matrice irriducibile P che ammetta v come probabilità invariante. Si calcoli poi
$$wP^n$$
per $n = 1, 2, \ldots, 100$, w distribuzione di probabilità. Si produca un grafico semilogaritmico nelle ordinate dell'errore $\|v - wP^n\|_1$.

2. Si implementi un algoritmo deterministico che calcola il minimo tra gli elementi di un vettore dato.

3.? Si consideri la funzione $H(i) = x_i^2 \operatorname{sen}(3x_i)$ sull'insieme $\{x_i\}_i$ dei 100 punti equispaziati tra $\pi/2$ e 2π. Si costruisca la matrice di transizione P_ε della catena che, ad un tempo n grande, ha una distribuzione che si concentra su quegli stati in cui H è più piccola. Si verifichi che per ε sufficientemente piccolo e n sufficientemente grande, $(wP_\varepsilon^n)_{\bar{i}} \approx 1$, dove w è una distribuzione di probabilità e $\bar{i}$ è tale per cui $\min_i H(i) = H(\bar{i})$.

4.? Per la funzione $H(i)$ dell'esercizio precedente, si consideri la matrice di transizione P_ε associata alla probabilità normalizzata $\hat{v}^\varepsilon$, $\varepsilon = 0.01$, $v_i^\varepsilon = \mathrm{e}^{-H(i)/\varepsilon}$. Data una distribuzione di probabilità w, si calcoli $\|wP_\varepsilon^n - \hat{v}^\varepsilon\|_1$, $n = 0, 1, \ldots, 1000$ e si verifichi, con un grafico semilogaritmico nelle ordinate, che $\|wP_\varepsilon^n - \hat{v}^\varepsilon\|_1 = \mathcal{O}(\lambda_2^n)$, ove λ_2 è il secondo autovalore di modulo massimo di P_ε. Perché si usa proprio λ_2?

`testpf`

5.? Per la funzione $H(i)$ dell'esercizio precedente, si consideri come matrice di transizione (di una catena *non* omogenea) P_{ε_n}, $\varepsilon_n = 1.01^{-n}$. Si facciano tante transizioni quante servono perché $(wP_{\varepsilon_n}^n)$ abbia una componente che dista da 1 meno di 10^{-4}. Si verifichi che ciò avviene in corrispondenza del minimo di H.

`sam`

6.? Si ripeta l'esercizio precedente per il calcolo del massimo di H.

7. Si implementi un algoritmo che calcola la potenza n-esima di una matrice usando le seguenti relazioni
$$n = c_1 + 2c_2 + 2^2 c_3 + \ldots + 2^m c_m, \quad c_i \in \{0, 1\},\ c_m \neq 0$$
$$P^n = P^{c_1 + 2c_2 + 2^2 c_3 + \ldots + 2^m c_m} = P^{\sum_{i=0}^m 2^i c_i} = \prod_{i=0}^m P^{2^i c_i} = \prod_{i=0}^m \left(P^{2^i}\right)^{c_i}$$
$$P^{2^i} = P^{2^{i-1}} P^{2^{i-1}}$$

Capitolo 2

Simulazione di catene di Markov

2.1 Simulazione di variabili aleatorie discrete

Sia U la variabile aleatoria uniforme su $[0,1]$, $I_{[a,b]}$ la funzione indicatrice dell'intervallo $[a,b]$ e X una variabile aleatoria discreta di distribuzione $[v_1, v_2, \ldots, v_n]$. La variabile aleatoria Y definita da

$$Y = \sum_{k=1}^{n} k I_{\left[\sum_{j=1}^{k-1} v_j, \sum_{j=1}^{k} v_j\right]}(U)$$

è distribuita come X (si pone $\sum_{j=1}^{0} v_j = 0$).

Il comando `rand(n)` genera una matrice di ordine n di numeri casuali, distribuiti in maniera uniforme nell'intervallo $[0,1]$. Dato il vettore v di lunghezza n, il comando `cumsum(v)` genera il vettore

$$[v_1, v_1 + v_2, \ldots, v_1 + v_2 + \ldots + v_n] = \left[\sum_{j=1}^{1} v_j, \sum_{j=1}^{2} v_j, \ldots, \sum_{j=1}^{n} v_j\right]$$

Dunque, il comando `randdisc`

```
k = find(rand < cumsum(v),1)
```

simula la variabile aleatoria Y

$$Y = \begin{cases} 1 & 2 & \ldots & n \\ v_1 & v_2 & \ldots & v_n \end{cases}$$

cioè tale che

$$\mathrm{P}(Y = k) = v_k$$

```
function j = simula(P,N,i)
%
% j = simula(P,N)
% j = simula(P,N,i)
%
% Simula la catena di Markov di matrice di transizione P,
% per N transizioni
n = length(P);
if (nargin == 2)
  i = randdisc(ones(1,n)/n);
end
for k = 1:N
  j = randdisc(P(i,:));
  i = j;
end
```

Tabella 2.1: Simulazione di una C.d.M.

Per rappresentare variabili aleatorie discrete, mettendo in ascissa il valore assunto y e in ordinata la probabilità di assumerlo v, si può usare il comando `bar(y,v)`.

2.2 Simulazione di catene di Markov

Data la matrice di transizione $P = (p_{ij})$ associata ad una catena di Markov e supponendo di essere nello stato i, per conoscere lo stato della catena alla transizione successiva basta simulare la variabile aleatoria che assume valore j con probabilità p_{ij}, come implementato in Tabella 2.1

`testsimula`

Per verificare la corretta implementazione della simulazione si può procedere come segue: a partire da una probabilità iniziale $v^{(0)} = [0, \ldots, 0, 1, 0, \ldots, 0]$ (1 in posizione i), si eseguono un numero N di transizioni di stato $v^{(n+1)} = v^{(n)}P$ tramite la matrice di transizione P. Poi, a partire dallo stato i, si esegue un numero molto alto di simulazioni attraverso la function `simula(P,N,i)` e si calcolano le frequenze relative di arrivo in uno stato j. Infine, si confrontano $v^{(N)}$ e il vettore delle frequenze relative.

2.2.1 Simulazione di catene speciali

Vediamo come simulare catene di Markov con matrice di transizione di tipo (1.1). Supponiamo di essere nello stato $X_n = i$: scegliamo lo stato j con

probabilità q_{ij}. Dopo di che

1. se $v_j \geq v_i$, accettiamo la transizione con probabilità uguale a 1
2. se $v_j < v_i$, accettiamo la transizione con probabilità v_j/v_i, altrimenti rifiutiamo la transizione. Questo si ottiene generando in modo indipendente una variabile aleatoria di Bernoulli

$$Y_n^{ji} = \begin{cases} 1 & 2 \\ \dfrac{v_j}{v_i} & 1 - \dfrac{v_j}{v_i} \end{cases}$$

Dunque, vediamo quanto vale $\mathrm{P}(X_{n+1} = j | X_n = i)$

1. se $v_j \geq v_i$ e $j \neq i$, $\mathrm{P}(X_{n+1} = j | X_n = i) = \mathrm{P}(\text{“uscito } j\text{”}) = q_{ij}$
2. se $v_j < v_i$ e $j \neq i$,

$$\begin{aligned} \mathrm{P}(X_{n+1} = j | X_n = i) &= \mathrm{P}(\text{“uscito } j\text{”}, Y_n^{ji} = 1) = \\ &= \mathrm{P}(\text{“uscito } j\text{”}) \cdot \mathrm{P}(Y_n^{ji} = 1 | \text{“uscito } j\text{”}) = q_{ij} \frac{v_j}{v_i} \end{aligned}$$

3. se $j = i$,

$$\begin{aligned} \mathrm{P}(X_{n+1} = i | X_n = i) &= \mathrm{P}(\text{“uscito } i\text{”}) + \\ &+ \sum_{\substack{j \neq i \\ v_j < v_i}} \mathrm{P}(\text{“uscito } j \neq i\text{”}) \cdot \mathrm{P}(Y_n^{ji} = 2 | \text{“uscito } j\text{”}) = \\ &= q_{ii} + \sum_{\substack{j \neq i \\ v_j < v_i}} q_{ij} \left(1 - \frac{v_j}{v_i}\right) = \\ &= 1 - \left(\sum_{j \neq i} q_{ij} + q_{ii}\right) + q_{ii} + \sum_{\substack{j \neq i \\ v_j < v_i}} q_{ij} \left(1 - \frac{v_j}{v_i}\right) = \\ &= 1 - \sum_{\substack{j \neq i \\ v_j < v_i}} q_{ij} - \sum_{\substack{j \neq i \\ v_j \geq v_i}} q_{ij} + \sum_{\substack{j \neq i \\ v_j < v_i}} q_{ij} - \sum_{\substack{j \neq i \\ v_j < v_i}} q_{ij} \frac{v_j}{v_i} = \\ &= 1 - \sum_{\substack{j \neq i \\ v_j \geq v_i}} q_{ij} - \sum_{\substack{j \neq i \\ v_j < v_i}} q_{ij} \frac{v_j}{v_i} = 1 - \sum_{\substack{j=1 \\ j \neq i}}^{n} p_{ij} \end{aligned}$$

La catena di Markov ottenuta ha dunque come matrice di transizione $P = (p_{ij})$. L'algoritmo proposto permette però di simulare la catena riducendo notevolmente il numero di elementi v_i da valutare.

Come criterio d'arresto per la simulazione della catena, si può considerare il numero di permanenze in uno stato: quando questo è maggiore di un numero prefissato, si interrompe la simulazione.

2.3 Esercizi

1. Si considerino le variabili aleatorie $\{X_n\}$ i.i.d., distribuite come la variabile aleatoria di Bernoulli

$$X = \begin{cases} 1 & 0 \\ p & q \end{cases}$$

con $p = 1/3$. Le si simuli, verificando che $\frac{1}{n}\sum_{i=1}^{n} X_n \approx \mathrm{E}(X) = p$.

2? Si implementi l'algoritmo di Metropolis senza usare la matrice di transizione e lo si testi per trovare il massimo della funzione $H(i)$ definita nell'Esercizio 3 del Capitolo 1.

3. Si ripeta l'esercizio precendente considerando 200 e 400 stati. Si implementi l'algoritmo salvando le valutazioni di H negli stati già visitati. `sas`

Capitolo 3

Legge dei grandi numeri per C.d.M.

3.1 Alcuni modelli

3.1.1 Code a stati numerabili

Sia $X(t)$ il numero di clienti in coda al tempo t, $X(t) \in \mathbb{N}$.

Coda M/G/1

Siano $T_1, T_2, \ldots$ gli istanti di compimento di servizio. Definiamo $X_n = X(T_n+)$ il numero di clienti in coda immediatamente dopo il compimento del servizio al tempo T_n e A_{n+1} il numero di clienti che arrivano nel periodo $[T_n, T_{n+1})$. Allora vale la relazione

$$X_{n+1} = (X_n - 1)_+ + A_{n+1}$$

Se $\{A_n\}$ sono i.i.d. e indipendenti da X_1 e $\mathrm{P}(A_2 = k) = a_{k+1}$, allora la matrice di transizione associata alla catena di Markov $\{X_n\}$ è

$$P = \begin{bmatrix} a_1 & a_2 & a_3 & \ldots & \ldots \\ a_1 & a_2 & a_3 & \ldots & \ldots \\ 0 & a_1 & a_2 & a_3 & \ldots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

Coda G/M/1

Siano ora $\tau_1, \tau_2, \ldots$ gli istanti in cui arriva un cliente, S_{n+1} il numero di potenziali compimenti del servizio nell'intervallo $[\tau_n, \tau_{n+1})$ e $X_n = X(\tau_n-)$ il

numero di clienti in coda immediatamente prima dell'n-esimo arrivo. Allora vale la relazione

$$X_{n+1} = (X_n + 1 - S_{n+1})_+$$

Se $\{S_n\}$ sono i.i.d. e indipendenti da X_1 e $\mathrm{P}(S_2 = k) = s_{k+1}$, allora la matrice di transizione associata alla catena di Markov $\{X_n\}$ è

$$P = \begin{bmatrix} \sum_{i=2}^{\infty} s_i & s_1 & 0 & \dots & \dots \\ \sum_{i=3}^{\infty} s_i & s_2 & s_1 & 0 & \dots \\ \sum_{i=4}^{\infty} s_i & s_3 & s_2 & s_1 & \ddots \\ \vdots & s_4 & s_3 & s_2 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix} = \begin{bmatrix} 1 - \sum_{i=1}^{1} s_i & s_1 & 0 & \dots & \dots \\ 1 - \sum_{i=1}^{2} s_i & s_2 & s_1 & 0 & \dots \\ 1 - \sum_{i=1}^{3} s_i & s_3 & s_2 & s_1 & \ddots \\ \vdots & s_4 & s_3 & s_2 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}$$

Si noti che, pur essendo P una matrice di "ordine" infinito, ogni sua riga è composta da un numero finito di elementi diversi da 0. Inoltre, il primo elemento di ogni riga si ottiene con una somma finita. `rigacodagm1`

3.1.2 Code a stati finiti

Un modello di coda a stati finiti $E = \{0, 1, 2, \dots, n\}$ può essere ottenuto definendo gli stati i, $i < n$ come "il numero di persone in coda è i" e lo stato n come "il numero di persone in coda è maggiore o uguale a n.

3.2 Calcolo di probabilità invarianti: `ones`

Vale il seguente

Teorema. *Sia P una matrice stocastica irriducibile di ordine n. Se v è una probabilità invariante, allora*

$$v = [1, 1, \dots, 1](I - P + \texttt{ones})^{-1}$$

ove `ones` *è una matrice di ordine n con tutti gli elementi pari ad 1.*

Il comando per calcolare v è dunque

```
v = ones(1,n)/(eye(n)-P+ones(n))
```

3.3 Legge dei grandi numeri per C.d.M.

Vale il seguente

Teorema (delle medie temporali)**.** *Sia $f\colon E \to \mathbb{R}$ una funzione non negativa o limitata e $\{X_n\}$ C.d.M. con spazio degli stati E, irriducibile e positivamente ricorrente e v l'unica probabilità invariante. Allora*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} f(X_n) = \sum_{j\in E} f(j) v_j \ .$$

invariantSIM

Un'applicazione importante del teorema è la seguente: sia $f(j) = \delta_{ij}$ (si può implementare tramite il comando `(i == j)`) allora

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \delta_{iX_n} = v_i \ .$$

Dunque è possibile approssimare v_i calcolando

$$\frac{1}{N} \sum_{n=1}^{N} \delta_{iX_n} \tag{3.1}$$

per N sufficientemente grande. Da notare che v è la probabilità invariante, mentre la probabilità limite potrebbe non esistere. Per calcolare (3.1) non serve la matrice di transizione della catena $\{X_n\}$: basta saper passare dallo stato $i = X_n$ allo stato $j = X_{n+1}$ con probabilità p_{ij}.

3.4 Verifica di medie

Sia $\{X_n\}$ una catena di Markov. Si vuole *stimare* $\mathrm{P}(X_n = j | X_0 = i) = p_{ij}^{(n)}$ senza calcolare esplicitamente la potenza n-esima della matrice di transizione associata alla catena. Indicando con A_j l'evento $\{X_n = j | X_0 = i\}$, si ha

$$\mathrm{E}(I_{A_j}) = \mathrm{P}(A_j)$$

Consideriamo adesso un campione statistico $\{I_{A_j}^{(i)}\}_{i=1}^{m}$ i.i.d. (simulazioni ripetute in maniera indipendente della catena, dallo stato i per n transizioni). Uno stimatore (corretto e consistente) della media di I_{A_j} è

$$\overline{I}_{A_j} = \frac{1}{m} \sum_{i=1}^{m} I_{A_j}^{(i)}$$

Per la legge dei grandi numeri

$$\lim_{m\to\infty} \overline{I}_{A_j} = \mathrm{E}(I_{A_j}^{(1)}) = \mathrm{P}(A_j) = p_{ij}^{(n)}$$

3.5 Esercizi

1. Happy Harry gioca a basket. La sua "produttività" oscilla tra 1 (quando realizza 0 o 1 punto), 2 (tra 2 e 5 punti) e 3 (più di 5 punti). Quando in una partita realizza molti punti, nella partita successiva i suoi compagni tendono a non passargli la palla. Dunque, la matrice di transizione tra i suoi stati di produttività potrebbe essere `harry`

$$P = \begin{bmatrix} 0 & \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & 0 & \frac{2}{3} \\ 1 & 0 & 0 \end{bmatrix}$$

   Sul lungo periodo, in quale proporzione Harry realizza il massimo di produttività? Harry è pagato 40 dollari a partita se realizza la massima produttività, 30 dollari se realizza la produttività intermedia e 20 dollari altrimenti. Sul lungo periodo, qual è il suo guadagno medio per partita?

2. Costruire le matrici di transizione delle code M/G/1 e G/M/1 a stati finiti.

3.? Si consideri una coda (a stati numerabili) G/M/1. Il numero di potenziali compimenti di servizio negli intervalli $[\tau_n, \tau_{n+1})$ sono dati da variabili aleatorie S_{n+1} i.i.d. distribuite come una variabile aleatoria di *Poisson* $\mathcal{P}(\lambda)$ di distribuzione

$$\mathrm{P}(\mathcal{P}(\lambda) = k) = s_{k+1} = \frac{\mathrm{e}^{-\lambda}\lambda^k}{k!},\ k \geq 0$$

   con $\lambda = 1.5$. Se all'istante τ_1 vi sono 3 clienti in coda, qual è la probabilità che vi siano 2 clienti in coda all'istante τ_{30}? Qual è la probabilità che, sul lungo periodo, non vi siano clienti in coda? Si risolva l'esercizio senza costruire la matrice di transizione associata alla catena. `simulacodagm1`

Indice dei comandi