

Dispense del corso di
Laboratorio di Sistemi Stocastici

Dott. Marco Caliari

a.a. 2008/09

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 3.0.x. Matlab[®] potrebbe dare risultati diversi.

Indice

0	Algebra lineare	5
0.1	Matrici a banda	5
0.2	Matrici di Toeplitz	6
0.3	Sistemi lineari e autovettori	6
0.4	Altri comandi utili	8
0.5	Esponenziale di matrice	8
0.6	Esercizi	9
1	Algoritmo di Metropolis	12
1.1	Generazione di matrici stocastiche simmetriche	12
1.2	Algoritmo di Metropolis	12
1.2.1	Simulated annealing	15
1.3	Il problema del commesso viaggiatore	16
1.4	Esercizi	16
2	Simulazione di catene di Markov	18
2.1	Simulazione di variabili aleatorie	18
2.2	Simulazione di catene di Markov	19
2.2.1	Simulazione di catene speciali	19
2.3	Alcuni modelli	20
2.3.1	Code a stati numerabili	20
2.3.2	Code a stati finiti	21
2.4	Calcolo di probabilità invarianti: <code>ones</code>	22
2.5	Legge dei grandi numeri per C.d.M.	22
2.6	Verifica di medie	23
2.7	Esercizi	23
3	Catene di Markov in tempo continuo	25
3.1	Simulazione di variabili aleatorie continue	25
3.1.1	Simulazione di variabili aleatorie	25
3.1.2	Simulazione di variabili aleatorie	26

3.2	Processo di Poisson omogeneo	26
3.3	Coda M/M/1	26
3.4	Processi continui a stati finiti	27
3.5	Esercizi	27
4	Processi con rinnovo	29
4.1	Teoremi limite	29
4.2	Coda G/G/1	30
4.3	Esercizi	30
	Indice dei comandi	31

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 $\mathbf{c} = [c_1, c_2, \dots, c_n]$ e $\mathbf{r} = [c_1, r_2, \dots, r_n]$. 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* \mathbf{x} . Se invece si intende risolvere il sistema lineare $b^T = yA$, si deve usare il comando

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

ove il risultato è il vettore *riga* \mathbf{y} . Alternativamente, si riscrive il problema come $A^T y^T = b$. 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 risolve considerando il problema $P^T v^T = v^T$. Per la soluzione, si usa il comando

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

Le matrici V , D e P' soddisfano l'equazione $P^T V = VD$, con $D = \text{diag}(d_1, d_2, \dots)$ 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.*

Inoltre, per il teorema dei cerchi di Gershgorin, tutti gli autovalori hanno modulo minore o uguale a 1. Per calcolare l'autovettore associato 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. 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

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 converge ad una combinazione lineare degli autovettori relativi all'autovalore 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
```

Vediamo come risolvere il problema

$$Ax = 0$$

per una matrice di ordine n e rango $n - 1$. Possiamo considerare la fattorizzazione $QRP^T = A$ 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 $r_{nn} = 0$. Allora, definendo $y = P^T x$, è possibile risolvere il sistema lineare $\tilde{R}y = \tilde{b}$, $b = 0$, risolvendo ponendo $\tilde{r}_{nn} = 1$ e $\tilde{b}_n = 1$. A quel punto, $x = Py$ è soluzione (non nulla, non normalizzata) di $Ax = 0$.

0.5 Esponenziale di matrice

Data una matrice A , si definisce $\exp(A)$ come

$$\exp(A) = \sum_{i=0}^{\infty} \frac{A^i}{i!}$$

Valgono le seguenti

$$\begin{aligned}\exp(0) &= I \\ \exp(\text{diag}(d_1, d_2, \dots, d_n)) &= \text{diag}(e^{d_1}, e^{d_2}, \dots, e^{d_n}) \\ \exp(2A) &= \exp(A) \exp(A)\end{aligned}$$

ma, in generale,

$$\exp(A + B) \neq \exp(A) \exp(B)$$

Per l'approssimazione di $\exp(A)$, *non* si usa una troncata della serie, in quanto la convergenza è lenta e prona ad errori di cancellazione. Il comando di GNU Octave che approssima $\exp(A)$ è `expm(A)`.

0.6 Esercizi

1. Implementare una function che determini se una matrice è stocastica, a meno dell'errore di macchina.
2. Si implementino due functions che costruiscono le matrici

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

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.

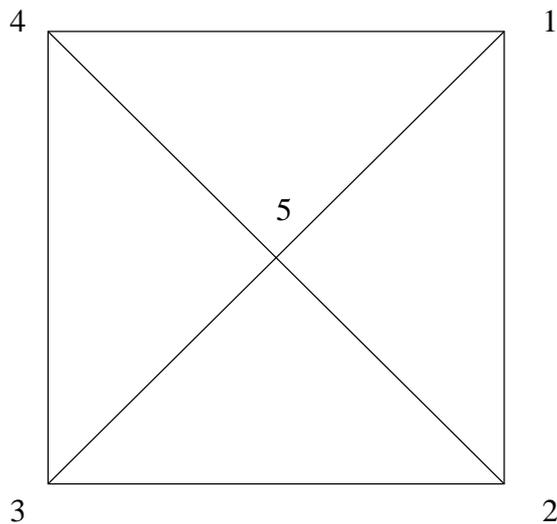
4. Si ripeta l'esercizio precedente con la matrice

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

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.
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`.
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`.
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

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

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

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

```
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 $q_{21} + w_1 + w_2 + \dots + w_{n-1} = 1$. A partire da un vettore random \tilde{w} dobbiamo normalizzarlo in modo che $q_{21} + \alpha(\tilde{w}_1 + \tilde{w}_2 + \dots + \tilde{w}_{n-1}) = 1$. Dunque $w = \tilde{w}\alpha = \left(\frac{1-q_{21}}{\tilde{w}_1+\tilde{w}_2+\dots+\tilde{w}_{n-1}}\right)$. Per la terza riga (e colonna) è $z = \tilde{z}\beta = \left(\frac{1-(q_{31}+q_{32})}{\tilde{z}_1+\tilde{z}_2+\dots+\tilde{z}_{n-2}}\right)$. 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](#)

1.2 Algoritmo di Metropolis

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

```

function Q = stocsym(n)
%
% Q = stocsym(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(min(Q)),0]);
Q = Q/(sum(sum(Q))/n);

```

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

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

$$p_{ij} = \begin{cases} q_{ij} & v_j \geq v_i, i \neq j \\ q_{ij} \frac{v_j}{v_i} & v_j < v_i, i \neq j \\ 1 - \sum_{\substack{j=1 \\ j \neq i}}^n p_{ij} & i = j \end{cases} \quad (1.1)$$

ammette v come unica probabilità invariante. 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 v data (algoritmo di Metropolis). Se v non è uniforme, la catena di Markov di matrice di transizione P è regolare e dunque

$$\lim_{m \rightarrow \infty} p_{ij}^{(m)} = v_j = \lim_{m \rightarrow \infty} (wP^m)_j$$

per qualunque probabilità w .

Il calcolo di P^m secondo l'algoritmo

$$P^m = PP^{m-1}$$

```
function P = metropolisminimo(H,varargin)
%
% P = metropolisminimo(H,varargin)
%
n = length(H);
if (nargin == 1)
    Q = stocsym(n);
else
    Q = varargin{1};
end
V = repmat(H,n,1);
W = repmat(H',1,n);
P = zeros(n);
index = find(V <= W);
P(index) = Q(index);
index = find(V > W);
P(index) = Q(index).*exp(-(V(index)-W(index)));
P = P-diag(diag(P));
d = 1-sum(P,2);
P = P+diag(d);
```

Tabella 1.2: Algoritmo di Metropolis

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 $H: E \rightarrow \mathbb{R}$, si considera la distribuzione di “probabilità” di elementi $v_i^\varepsilon = e^{-H(i)/\varepsilon}$ (in realtà, non è una probabilità perché non normalizzata). Tramite l'algoritmo di Metropolis, si può costruire una catena con matrice di transizione P_ε avente $v^\varepsilon / \|v^\varepsilon\|_1$ come probabilità invariante. La matrice $P_\varepsilon = (p_{ik}^\varepsilon)$ è definita da

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

Per evitare problemi di overflow/underflow nel calcolo di v^ε per ε piccolo, conviene adattare l'algoritmo di Metropolis in Tabella 1.2 passando come input il vettore $[H(1), H(2), \dots]$. Se H assume il suo valore minimo nell'unico punto $\bar{i} \in E$, allora

$$\lim_{\varepsilon \rightarrow 0} \frac{v_j^\varepsilon}{\|v^\varepsilon\|_1} = \delta_{\bar{i}j}$$

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*.

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 $\omega = (i_2, i_3, \dots, i_N)$, 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

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

1.4 Esercizi

1. Data una distribuzione di probabilità v si costruisca una matrice irriducibile che ammetta v come probabilità invariante. Si calcoli poi

$$wP^n$$

per $n = 1, 2, \dots, 100$, w distribuzione di probabilità. Si produca un grafico semilogaritmico nelle ordinate dell'errore $\|v - wP^n\|_\infty$.

2. Si implementi un algoritmo deterministico che calcola il minimo tra gli elementi di un vettore dato.
- 3.? Si consideri la funzione $H(i) = x_i^2 \sin(3x_i)$ 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, $(wP_\varepsilon^n)_{\bar{i}} \approx 1$, se $\min_i H(i) = H(\bar{i})$.
- 4.? Si ripeta l'esercizio precedente, considerando come matrice di transizione (di una catena *non* omogenea) P_{ε_n} , $\varepsilon_n = 1.01^{-n}$. Si arresti il calcolo di $(wP_{\varepsilon_n}^n)$ non appena una delle sue componenti dista da 1 meno di 10^{-4} e si verifichi che ciò avviene in corrispondenza del minimo di H .
- 5.? Si ripeta l'esercizio precedente per il calcolo del massimo di H .

6. Si implementi un algoritmo che calcola la potenza n -esima di una matrice usando le seguenti relazioni

$$n = c_1 + 2c_2 + 2^2c_3 + \dots + 2^m c_m, \quad c_i \in \{0, 1\}, \quad c_m \neq 0$$

$$P^n = P^{c_1+2c_2+2^2c_3+\dots+2^m c_m} = P^{\sum_{i=0}^m 2^i c_i} = \prod_{i=0}^m P^{2^i c_i} = \prod_{i=0}^m \left(P^{2^i}\right)^{c_i}$$

$$P^{2^i} = P^{2^{i-1}} P^{2^{i-1}}$$

Capitolo 2

Simulazione di catene di Markov

2.1 Simulazione di variabili aleatorie

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 densità $[v_1, v_2, \dots, v_n]$. La variabile aleatoria Y definita da

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

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

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

Dunque, il comando

```
k = min(find(rand < cumsum(v)))
```

simula la variabile aleatoria Y tale che

$$P(Y = k) = v_k$$

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)`.

```

function i = simula(P,N,varargin)
n = length(P);
if (nargin == 2)
    i = min(find(rand < cumsum(ones(1,n)/n)));
else
    i = varargin{1};
end
for k = 1:N
    i = min(find(rand < cumsum(P(i,:))));
end

```

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

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 al tempo n , per conoscere lo stato della catena al tempo successivo basta simulare la variabile aleatoria che assume valore j con probabilità p_{ij} , come implementato in Tabella 2.1

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

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

Dunque,

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

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

3. se $j = i$,

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

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 Alcuni modelli

2.3.1 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, \dots 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 $P(A_2 = k) = a_{k+1}$, allora la matrice di transizione associata alla catena di Markov $\{X_n\}$ è

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

Coda G/M/1

Siano ora τ_1, τ_2, \dots gli istanti in cui arrivano clienti, S_{n+1} il numero di potenziali complimenti del servizio nell'intervallo $[\tau_n, \tau_{n+1})$ e $X_n = X(\tau_n-)$ il numero di clienti immediatamente prima dell' n -esimo arrivo. Allora vale la relazione

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

Se $\{S_n\}$ sono i.i.d. e indipendenti da X_1 e $P(S_2 = k) = s_{k+1}$, 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.

2.3.2 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 ”.

2.4 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*

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

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

Il comando per calcolare v è dunque

$$v = \text{ones}(1, n) / (\text{eye}(n) - P + \text{ones}(n))$$

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

Vale il seguente

Teorema (delle medie temporali). *Sia $f: E \rightarrow \mathbb{R}$ 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*

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

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

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

Dunque è possibile approssimare v_i calcolando

$$\frac{1}{N} \sum_{n=1}^N \delta_{iX_n} \tag{2.1}$$

per N sufficientemente grande. Da notare che v è la probabilità invariante, mentre la probabilità limite potrebbe non esistere. Per calcolare (2.1) 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} .

2.6 Verifica di medie

Sia $\{X_n\}$ una catena di Markov. Si vuole *stimare* $P(X_n = j | X_0 = i) = p_{ij}^{(n)}$ 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

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

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

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

Per la legge dei grandi numeri

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

2.7 Esercizi

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

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

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

- 2.? Si ripetano gli Esercizi 3, 4 e 5 del Capitolo 1 simulando la catena di Markov coinvolta.
3. 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

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

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?

4. Costruire le matrici di transizione delle code M/G/1 e G/M/1 a stati finiti.
5. 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

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

con $\lambda = 2.5$. 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.

Capitolo 3

Catene di Markov in tempo continuo

3.1 Simulazione di variabili aleatorie continue

Sia X una variabile aleatoria con densità $f(x)$ e funzione di ripartizione

$$F_X(t) = \int_{-\infty}^t f(x)dx$$

e U la variabile aleatoria uniforme su $[0, 1]$. Allora $F_X^{-1}(U)$ è distribuita come X .

3.1.1 Simulazione di variabili aleatorie per cui è nota l'inversa della funzione di ripartizione

La variabile aleatoria *esponenziale* $\mathcal{E}(\lambda)$ ha come funzione di ripartizione $F_{\mathcal{E}_\lambda}(t) = 1 - e^{-\lambda t}$, la cui inversa è $F_{\mathcal{E}_\lambda}^{-1}(u) = -\log(1 - u)/\lambda$. Dunque, il comando

```
-log(1-rand)/lambda;
```

simula la variabile aleatoria esponenziale di parametro λ .

3.1.2 Simulazione di variabili aleatorie per cui *non* è nota l'inversa della funzione di ripartizione

La variabile aleatoria *normale* $\mathcal{N}(\mu, \sigma)$ di densità

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

ha come funzione di ripartizione

$$F_{\mathcal{N}(\mu, \sigma)}(t) = \frac{1}{2} \left(1 + \operatorname{erf} \left(\frac{t - \mu}{\sqrt{2}\sigma} \right) \right)$$

ove

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

La funzione inversa $x = \operatorname{erf}^{-1}(y)$ non è nota esplicitamente: dato y si tratta allora di risolvere l'equazione, nell'incognita x , $y - \operatorname{erf}(x) = 0$. Il metodo di Newton per tale equazione necessita solo delle funzioni $\operatorname{erf}(x)$ e $\operatorname{erf}'(x) = 2e^{-x^2}/\sqrt{\pi}$. La funzione `erfinv` usa proprio il metodo di Newton per il calcolo di erf^{-1} . Dunque, $F_{\mathcal{N}(\mu, \sigma)}^{-1}(u) = \mu + \sqrt{2}\sigma \operatorname{erf}^{-1}(2u - 1)$.

3.2 Processo di Poisson omogeneo

La catena immersa di un processo di Poisson omogeneo ha matrice di transizione

$$Q = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

e la successione dei tempi aleatori $\{T_n\}$ soddisfa $T_1 = 0$, $T_{n+1} - T_n \sim \mathcal{E}(\lambda)$. Allora il numero di cambiamenti di stato $N(0, t]$ nell'intervallo di tempo $(0, t]$ è distribuito come una variabile aleatoria di Poisson $\mathcal{P}(\lambda t)$.

3.3 Coda M/M/1

In una coda M/M/1 i tempi di inter-arrivo sono distribuiti come $\mathcal{E}(a)$ (dunque il numero di arrivi è poissoniano) e i tempi di inter-servizio sono distribuiti come $\mathcal{E}(b)$. Il processo $\{X(t)\}$ conta il numero di persone in coda e, per le

proprietà delle variabili aleatorie esponenziali, la matrice di transizione della catena immersa è

$$Q = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots \\ \frac{b}{a+b} & 0 & \frac{a}{a+b} & 0 & \dots \\ 0 & \frac{b}{a+b} & 0 & \frac{a}{a+b} & \dots \\ \vdots & 0 & \ddots & \ddots & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}$$

con tempi di soggiorno nello stato i dati distribuiti come $\lambda(i) = \mathcal{E}(a)$ se $i = 1$ e $\lambda(i) = \mathcal{E}(a + b)$ se $i > 1$. Per una coda M/M/1 la probabilità invariante è data da $v_i = (1 - \rho)\rho^{i-1}$, $\rho = a/b$, se $a < b$.

La misura invariante η può essere approssimata per mezzo di

$$\eta_j = E_i \int_0^{T_{\tau_i(1)}} I_{[X(s)=j]} ds$$

Una volta simulata la catena, per ogni i -ciclo si considerano i tempi di permanenza nello stato j e se ne fa la media aritmetica.

3.4 Processi continui a stati finiti

Per un processo continuo a stati finiti, data la matrice di transizione Q della catena immersa e la funzione $\lambda(i)$, si può calcolare il generatore mediante il comando

```
A = diag(lambda)*(Q-eye(size(Q)))
```

Se la catena immersa è irriducibile e ricorrente, allora esiste un'unica misura invariante η per il processo, data dalla soluzione non banale (vedi Sezione 0.4) di

$$\eta A = 0$$

È inoltre possibile determinare la probabilità di passare dallo stato i allo stato j nell'arco di tempo Δt : essa è data dall'elemento p_{ij} della matrice $\exp(\Delta t A)$.

3.5 Esercizi

1. Verificare la correttezza della simulazione di variabili aleatorie continue.
2. Verificare che il numero di cambiamenti di stato $N(0, t]$ nell'intervallo di tempo $(0, t]$ di un processo di Poisson è distribuito come una variabile aleatoria di Poisson $\mathcal{P}(\lambda t)$.

3. Approssimare la probabilità invariante di una coda M/M/1 di parametri $a = 1$ e $b = 2$.
- 4.? Quando Happy Harry gioca a basket, è soggetto ad infortuni: essi variano dallo stato 1 (nessun infortunio), allo stato 2 (infortuni minori), allo stato 3 (infortuni seri che impediscono di giocare). La transizione tra gli stati avviene secondo la matrice di transizione

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

I tempi di permanenza nello stato i sono distribuiti come $\lambda(1) = \mathcal{E}(1/3)$, $\lambda(2) = \mathcal{E}(1/3)$ e $\lambda(3) = \mathcal{E}(1/6)$. Consideriamo il processo di Markov $\{X(t)\}$ che descrive lo stato di salute di Harry al tempo t . Qual è il generatore? Qual è, sul lungo periodo, la frazione di tempo in cui non può giocare? Harry è pagato 40 dollari quando è in buona forma, 30 dollari quando può comunque giocare e 20 dollari se non può giocare. Qual è il guadagno medio per partita sul lungo periodo? Si determini infine la probabilità di passare dallo stato di forma allo stato di infortunio serio in un arco di tempo $\Delta t = 10$. La si verifichi simulando più volte il processo.

5. Implementare una function `x = fgammainv(alpha, gamma, y)` che calcola l'inversa della funzione di ripartizione $F_{\Gamma(\alpha, \lambda)}(t)$ della variabile aleatoria $\Gamma(\alpha, \lambda)$, $\alpha > 0$, $\lambda > 0$, di densità

$$f(x) = \begin{cases} \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x} & x > 0 \\ 0 & x \leq 0 \end{cases}$$

(Sugg.: occorre scegliere con cura il punto di partenza per il metodo di Newton. In particolare, se $\alpha \geq 1$, conviene prendere come punto di partenza lo zero della derivata seconda di $F_{\Gamma(\alpha, \lambda)}(t)$. Altrimenti, osservare che $F_{\Gamma(\alpha, \lambda)}(t) < \int_0^t \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} dx$ e prendere come punto di partenza lo zero dell'equazione $\frac{\lambda^\alpha}{\Gamma(\alpha)} \frac{x^\alpha}{\alpha} - y = 0$.)

Capitolo 4

Processi con rinnovo

Useremo il comando `randg(alpha)/lambda` per simulare la variabile aleatoria $\Gamma(\alpha, \lambda)$ di densità

$$f(x) = \begin{cases} \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x} & x > 0 \\ 0 & x \leq 0 \end{cases}$$

e il comando `mu+sigma*randn` per simulare la variabile aleatoria $\mathcal{N}(\mu, \sigma)$.

4.1 Teoremi limite

Indicata con $\{S_n\}_{n=1}^\infty$ la successione dei rinnovi

$$S_n = \sum_{i=1}^n Y_i$$

($S_1 = 0$), e con $N(t)$ il numero di rinnovi nell'intervallo $[0, t]$, se $E(Y_1) < \infty$, allora

- $\frac{N(t)}{t} \xrightarrow{\text{q.c.}} \frac{1}{E(Y_1)}$ per $t \rightarrow \infty$
- $\frac{E(N(t))}{t} \rightarrow \frac{1}{E(Y_1)}$ per $t \rightarrow \infty$
- se $\text{Var}(Y_1) < \infty$, allora $N(t) \xrightarrow{d} \mathcal{N}\left(\frac{t}{E(Y_1)}, \sqrt{\frac{t \text{Var}(Y_1)}{E(Y_1)^3}}\right)$ per $t \rightarrow \infty$

4.2 Coda G/G/1

In una coda G/G/1, i tempi di inter-arrivo dei clienti sono $\{\sigma_n\}$, i.i.d. con media $1/a$ (a è il *tasso di arrivo*). Dunque i clienti arrivano ai tempi

$$A_{n+1} = \sum_{i=1}^n \sigma_i$$

($A_1 = 0$). Il cliente i -esimo viene servito in un tempo τ_i , con $\{\tau_n\}$ i.i.d. con media $1/b$ (b è il *tasso di servizio*). Il rapporto $\rho = a/b$ è detto *intensità di traffico*. Un processo di interesse è $\{W_n\}$, ove W_i è il tempo di attesa dell' i -esimo cliente. È facile rendersi conto che

$$\begin{aligned} W_1 &= 0 \\ W_{n+1} &= (W_n + \tau_n - \sigma_n)^+, \quad n > 0 \end{aligned}$$

Un altro processo interessante è $Q(t)$, il numero di clienti nel sistema (sia in coda che in servizio) al tempo t . Si può definire poi il tempo passato nel sistema dall' i -esimo cliente, $S^{(i)} = W_i + \tau_i$. Vale allora la formula di Little:

$$E(Q(\infty)) = aE(S^{(\infty)})$$

4.3 Esercizi

1. Verificare i teoremi limite per i processi di rinnovo.
2. Verificare la formula di Little per una coda G/G/1.

Indice dei comandi

\, 6
bar, 16

cumsum, 16

diag, 5

eig, 6
erfinv, 24
expm, 9

find, 13

magic, 11

norm, 8

qr, 8

rand, 16
randg, 27
randn, 27
repmat, 8

/, 6
sum, 8

toeplitz, 6