UNIVERSITA' DEGLI STUDI DI MILANO-BICOCCA

Scuola di Economia e Statistica

Corso di Laurea Magistrale in

Scienze Statistiche ed Economiche

RETI BAYESIANE A TEMPO CONTINUO

con

DISTRIBUZIONE DI FASE

Tesi di Laurea di:
Federico MELOGRANA
Matricola 789256

Relatore:
Prof. Fabio Antonio STELLA
Correlatore:
Prof.ssa Sonia MIGLIORATI

Anno Accademico 2018 - 2019

Sommario

Le reti Bayesiane a tempo continuo (CTBN) descrivono processi stocastici con stati finiti che evolvono in tempo continuo. Una CTBN è un grafico di dipendenze dirette su un insieme di variabili, ognuna delle quali rappresenta un processo di Markov a tempo continuo e stati finiti il cui modello di transizioni è funzione dei propri genitori. La rappresentazione di sistemi complessi con reti Bayesiane è da tempo un problema aperto. Riuscire ad inserire memoria in tali sistemi in maniera pulita rappresenta un miglioramento nella capacità rappresentativa di tali reti per applicazioni reali. Presentiamo una nuova rappresentazione Hidden variable, la rappresentazione bipartite che consente di ottenere la stessa espressività delle reti dirette con una complessità minore rispetto alla rappresentazione Hidden variable canonica. Possibili sviluppi in questa direzione possono portano a reti con grande espressività, interpretabilità e capacità inferenziali con complessità decisamente minore rispetto alle attuali.

Indice

Capitolo 1

Introduzione

In questo lavoro l'obbiettivo è quello di definire e imparare un nuovo tipo di Hidden Markov Model con distribuzione di fase. Potenziali applicazioni di tali reti si riscontrano in svariati campi, quali il campo biologico, medico-sanitario, neurologico ed in generale ovunque si voglia imparare una struttura che consenta di avere memoria. In tutti i sopracitati domini, ed in molti altri, si entra in contatto con strutture di serie storiche multivariate nelle quali è importante riuscire a cogliere sia le dipendenze di breve che di lungo periodo. Un metodo efficace per catturare queste dipendenze è usare modelli di Deep Learning, usando reti ricorrenti o ricorsive come le Long-Short Term Memory. Tale approccio, benchè molto potente, è limitato ad essere un approccio black-box. Quando le previsioni generate dal modello servono da supporto per prendere decisioni rilevanti un aspetto cruciale è quello di avere *interpretabilità* del risultato ottenuto. Essa è cruciale per supportare i decisori nelle loro scelte, ed è necessaria anche per capire le motivazioni sottostanti tali scelte, senza che i decisori debbano fidarsi ciecamente di un modello black-box.

L'aggiunta della distribuzione di fase ci serve per superare l'assenza di memoria di una rete Bayesiana classica, con transizioni regolate da variabili esponenziali/geometriche, che costituiscono un limite in situazioni reali. Introdurremo una nuova architettura di rete a variabile nascosta, chiamata rappresentazione bipartite. Tale rappresentazione ha due punti di forza; il primo è rappresentato dai vantaggi dell'approccio Hidden, ovvero migliore inferenza, facilità interpretativa, possibilità di incrementare il numero di fasi senza penalizzare eccessivamente la complessità. Secondo punto di forza è mantenere comunque, in maniera sostanziale, la possibilità di avere una transizione contemporanea tra la variabile di interesse (X) e la sua variabile Hidden (H), quindi nessuna perdita di efficienza rispetto alla rappresentazione diretta. Tale rappresentazione ha un significativo vantaggio in termini di semplicità ed efficienza rispetto alla rappresentazione full. La rapresentazione full rappresenta il modello canonico per un approccio Hidden variables.

Dopo aver definito questa rappresentazione lo step successivo è il Learning dei parametri, ovvero imparare i parametri di tale rete tramite delle traiettorie parzialmente osservate mediante l'algoritmo di Expectation-Maximization. In seguito, tramite alcune prove, vediamo come tali matrici abbiano degli effettivi vantaggi. L'estensione di tali traiettore con variabili genitori Y che influenzano X costituiscono il mattoncino base con il quale è possibile imparare la struttura complessiva di una rete.

Capitolo 2

Reti Bayesiane a tempo continuo

Le reti Bayesiane a tempo continuo (CTBN) coniugano le reti Bayesiane e i processi di Markov omogenei per modellare in maniera efficiente sistemi dinamici a stati discreti e tempo continuo. Tempo continuo in quanto non esistono degli intervalli di tempo predefiniti e preimpostati con i quali il nostro sistema evolve ma tale evoluzione avviene in maniera dinamica nel continuo. Tale strutturazione è particolarmente utile negli ambiti nei quali sono presenti numerosi variabili che hanno una granularità differente - ovvero evolvono con tempi molto diversi tra di loro.

2.1 Processi di Markov omogenei

Introduciamo in questa sezione *Processi di Markov omogenei a stati finiti e tempo continuo*. Supponiamo di avere una variabile X, che può assumere valori in $Val(X) = \{x_1, ..x_n\}$. Con $x_1, ..x_n$ l'insieme finito di tutti gli stati che può assumere X. Prendiamo ora una distribuzione iniziale P_X^0 su X. Ovvero una distribuzione di partenza di probabilità nella variabile X oggetto di interesse.

$$Q = \begin{bmatrix} -q_{x_1} & q_{x_1x_2} & \cdots & q_{x_1,x_n} \\ q_{x_2x_1} & -q_{x_2} & \cdots & q_{x_2,x_n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ q_{x_nx_1} & q_{x_n,x_2} & \cdots & -q_{x_n} \end{bmatrix}$$

Questa matrice n x n è una matrice di *intensità di transizioni*. Ovvero sulla diagonale è presente $-q_{x_i}$, l'intensità con la quale la distribuzione lascia lo stato i-esimo. Fuori dalla diagonale il generico elemento è invece $q_{x_ix_j}$, ovvero l'intensità di probabilità con la quale il processo transiziona dallo stato i-esimo allo stato j-esimo. E' importante notare come la sommatoria delle intensità in ogni riga debba essere uguale a zero, pertanto $q_{x_i} = \sum_{j \neq i}^{n} q_{x_ix_j}$ vale per ogni riga della matrice Q.

Quindi, il processo $X(t)$ che descrive transizioni di stato lungo il tempo nella variabile X è un *Processo di Markov omogeneo a stati discreti e tempo continuo* se valgono le seguenti condizioni:

- $Pr\{X(t+\Delta t) = x_j | X(t) = x_i\} \approx q_{x_i x_j} \Delta t$, *per* $i \neq j$
- $Pr\{X(t+\Delta t) = x_i | X(t) = x_i\} \approx 1 - q_{x_i} \Delta t$

quando $\Delta t \to 0$

La prima condizione significa che la propabilità istantanea ($\Delta t \to 0$) di transizione tra lo stato i e lo stato j è proporzionale all'intensità di transizione $q_{x_i x_j}$. La seconda condizione invece specifica che la probabilità instantanea di non transizionare, rimanere nello stato i, è inversamente proporzionale a q_{x_i}. Insieme, le due condizioni sopracitate significano che se al tempo t la variabile X è nello stato x_i allora resta nello stato i per un periodo di tempo T, distribuito come una *variabile esponenziale* il cui parametro θ è pari a q_{x_i}. La funzione di densità f per X che rimane in x_i è data dalla densità di una esponenziale:

$$f(q_{x_i}, t) = q_{x_i} exp(-q_{x_i} t)$$

con t>0.

Per definizione di variabile esponenziale di parametro q_{x_i} il valore atteso, ovvero il tempo medio nel quale il sistema lascia lo stato $i-esimo$, è pari a $1/q_{x_i}$. La probabilità di transizionare dallo stato $i-esimo$ allo stato $j-esimo$, dato che X ha lasciato lo stato i-esimo, è pari a $\theta_{x_i xj} = q_{x_i x_j}/q_{x_i}$. Ovvero, dato che il sistema ha lasciato lo stato $i-esimo$ la probabilità di transizionare in un altro stato generico j è regolata da una *multinomiale* con parametri $\theta_{x_i x_j}$.

La distribuzione della transizione in un sistema è dunque definita da una coppia di variabili casuali. Una Esponenziale per *quando* avviene la transizione e una variabile Multinomiale per *quale stato* risulta dalla transizione. Il comportamento del sistema X è pienamente determinato dalla coppia q_{x_i} e $q_{x_i}/q_{x_i x_j}$ che abbiamo definito come $\theta_{x_i xj}$. Questi parametri, insieme alla distribuzione iniziale, definiscono appieno il sistema.

La distribuzione $P_X(t)$ sugli stati di X in un generico punto temporale futuro t è definita da:

$$P_X(t) = P_X^0 expm^1(Q_x t)$$

Dove $P_X(t)$ rappresenta la distribuzione sulla variabile X al tempo t, $P_X(0)$ rappresenta la distribuzione iniziale su X, $expm(Q_x t)$ è l'esponenzializzazione matriciale della matrice di intensità di transizioni Q_X moltiplicata per il tempo t (differenza tra t e il punto di partenza 0).

[1] *expm* si riferisce all'esponenzializzazione di una matrice, con la seguente formula $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$ con $X^0 = I$ matrice identità

La notazione e le formula sopra introdotte definiscono un processo di Markov omogeneo a tempo continuo,definito d'ora in avanti più semplicemente *Processo di Markov omogeneo*, permettendone di comprendere struttura e comportamento. Tale processo è Markoviano in quanto è un processo aleatorio in cui la probabilità di transizione che determina il passaggio a uno stato di sistema dipende solo dallo stato del sistema immediatamente precedente e non da come si è giunti a tale stato[2]. E' omogeneo in quanto le intensità di transizione sono fisse e non variano nel tempo. Mentre è a tempo continuo poichè non vengono fissati dei tagli temporali (slice) tramite i quali il processo può variare ma esso transiziona dinamicamente nel tempo.

La limitazione principale di tale processo è la sua impossibilità di modificare le intensità di transizione nel tempo. La proprietà di omogeneità risulta essere un vincolo poco credibile in applicazioni reali, nelle quali le intensità variano nel tempo. Per sopperire a tale limitazioni bisogna fare riferimento alla classe dei *conditional* Markov process.

2.2 Conditional Markov process

I conditional Markov process sono un caso particolare della ampia classe dei processi di Markov non omogenei, dove le intensità variano nel tempo, ma non come funzione diretta del tempo. Nei conditional Markov process le intensità sono una funzione solamente del valore corrente di un insieme di altre variabili che evolvono anche esse come un processo di Markov. Tali variabili sono definite variabili *genitori* e il loro insieme viene indicato con $P_a(X_i)$ spesso abbreviato più semplicemente in $\mathbf{U}$. Questo insieme di variabili $\mathbf{U}$ definiscono il *Markov Blanket* della nostra variabile X.

Definition 2.2.1. Si definisce Markov Blanket $\mathbf{B_Z}$ di un insieme di variabili $\mathbf{Z} \subseteq \mathbf{X}$ l'insieme di variabili di X che non sono membri di Z ma sono genitori o figli di qualche $Z \in \mathbf{Z}$ o co-genitori[3] di qualche variabile $Z \in \mathbf{Z}$.

Il Markov Blanket B_Z separa i nodi $\mathbf{Z}$ dal resto dei nodi. Ovvero $\mathbf{Z}$ è indipendente dalle restanti variabili del sistema dato il suo Markov Blanket.

Per chiarire meglio cosa è un conditional Markov process, supponiamo di avere la variabile X che evolve come un processo di Markov $X(t)$ le cui dinamiche sono condizionate da un insieme $\mathbf{U}$ di variabili, i genitori di X. Dove anche ogni variabile appartenente a $\mathbf{U}$ evolve nel tempo. La dinamica di X(t) è totalmente descritta da un *insieme* di matrici di intensità, una per ogni allocazione di $\mathbf{U}$, ognuna della forma:

$$Q_{X|\mathbf{u}} = \begin{bmatrix} -q_{x_1|\mathbf{u}} & q_{x_1x_2|\mathbf{u}} & \cdots & q_{x_1,x_n|\mathbf{u}} \\ q_{x_2x_1|\mathbf{u}} & -q_{x_2|\mathbf{u}} & \cdots & q_{x_2,x_n|\mathbf{u}} \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ q_{x_nx_1|\mathbf{u}} & q_{x_n,x_2|\mathbf{u}} & \cdots & -q_{x_n|\mathbf{u}} \end{bmatrix}$$

[2]Proprietà di Markov

[3]Due variabile Y e Z sono co-genitori se hanno almeno un figlio in comune.

In particolare la *conditional intensity matrix* (CIM) $Q_{X|\mathbf{U}}$ è composta dall'insieme di matrici di intensità $Q_{X|\mathbf{u}}$, tante quante le diversi allocazioni possibili di $u \in U$. Ovvero avremo un numero di matrici $Q_{X|\mathbf{u}}$ pari a $Val(\mathbf{U})$. In altri termini abbiamo $Q_{X|\mathbf{U}} = \{Q_{X|\mathbf{u}} : \mathbf{u} \in Val(\mathbf{U})\}$

2.3 Continuos time Bayesian network

Il modello continuos time Bayesian network si definisce come segue:

Definition 2.3.1. Continuous time Bayesian network (Nodelman et al.,2002 [4]) Sia $\mathbf{X}$ un insieme di variabili locali $\mathbf{X} = \{X_1, .., X_n\}$. Ogni variabile X_i ha un numero finito di di valori $Val(X_i)$. Una Continuos time Bayesian network N su $\mathbf{X}$ consiste di due componenti: La prima è una distribuzione iniziale P_X^0 mentre la seconda è un continuous time transition model specificato come:

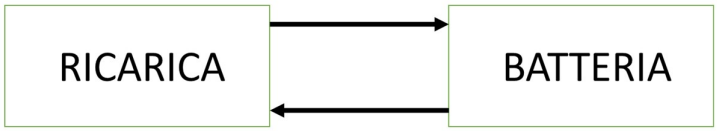
- Un grafico diretto (può anche essere ciclico) G i cui nodi sono $X_1, X_2, ...X_n$; $Pa_G(X_i)$ spesso abbreviato in $\mathbf{U_i}$, che denota i genitori di X_i in G.
- Una conditional intensity matrix (CIM) $Q_{X_i|U_i}$ per ogni variabile $X_i \in \mathbf{X}$

Ogni conditional intensity matrix $\mathbf{Q_{X|U}}$ può essere riassunta tramite due insieme indipendenti: $\mathbf{q_{X|U}} = \{q_{x_i|U} : 1 \leq i \leq n\}$, ovvero l'insieme delle intensità nella diagonale della CIM che caratterizzano la distribuzione esponziale su *quando* avverrà la transizione, e $\theta_{\mathbf{X|U}} = \{\theta_{x_i x_j|u} = \frac{q_{x_i x_j|u}}{q_{x_i|u}} : 1 \leq i, j \leq n, i \neq j\}$, l'insieme della probabilità che parametrizzano tutte le distribuzioni multinomiali su in *quale stato* transizionerà.

Una assunzione del modello CTBN è che solamente una variabile possa transizionare in un determinato momento, quindi non è possibile effettuare una doppia transizione. Le dinamiche con le quali avviene tale transizione sono determinate dall'instanziazione delle variabili genitori di X , tramite la corrispettiva CIM, e tali dinamiche sono indipendenti da tutte le altre variabili dato il proprio *Markov Blanket.*

Esempio 2.3. *Consideriamo un semplice sistema nel quale viene utilizzato un sistema di ricarica per le batterie di una macchina utensile, che si scarica durante l'utilizzo. L'operazione di ricarica (a cui ci riferimemo anche con RC) può assumere i valori {no, si}. Dove "no" significa che la macchina viene utilizzata e non ricaricata e "si" significa che viene ricaricata la macchina. Supponiamo che l'operazione di ricarica duri una determinata quantità di tempo. Supponiamo inoltre che la variabile Batteria (a cui ci riferiremo anche con BT) sia associata con il livello di batteria della nostra macchina utensile e assuma tre livelli $Val(Batteria) = \{zero, media, alta\}$. Dove "zero" corrisponde alla batteria completamente scarica. Vediamo dunque il modello di continuous time Bayesian network associato a questo sistema.*

$$\mathrm{Q}_{RC|BT=zero} = \begin{array}{c} \\ no \\ si \end{array} \begin{array}{c} \begin{array}{cc} no & si \end{array} \\ \begin{bmatrix} -\infty & \infty \\ 0 & 0 \end{bmatrix} \end{array} \qquad Q_{RC|BT=media} = \begin{array}{c} \\ no \\ si \end{array} \begin{array}{c} \begin{array}{cc} no & si \end{array} \\ \begin{bmatrix} -5 & 5 \\ 3 & -3 \end{bmatrix} \end{array} \qquad Q_{RC|BT=alta} = \begin{array}{c} \\ no \\ si \end{array} \begin{array}{c} \begin{array}{cc} no & si \end{array} \\ \begin{bmatrix} 0 & 0 \\ 4 & -4 \end{bmatrix} \end{array}$$



$$\mathrm{Q}_{BT|RC=no} = \begin{array}{c} \\ zero \\ media \\ alta \end{array} \begin{array}{c} \begin{array}{ccc} zero & media & alta \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 2 & -2 & 0 \\ 0 & 2 & -2 \end{bmatrix} \end{array} \qquad Q_{BT|RC=si} = \begin{array}{c} \\ zero \\ media \\ alta \end{array} \begin{array}{c} \begin{array}{ccc} zero & media & alta \end{array} \\ \begin{bmatrix} -5 & 5 & 0 \\ 0 & -5 & 5 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

Descriviamo il sistema sovrastante. Le matrici $Q_{RC|BT}$ si riferiscono alle transizioni della ricarica nel caso in cui la batteria sia zero, media o alta. Possiamo osservare che se la batteria è completamente scarica allora immediatamente la macchina viene ricaricata. Nelle matrici che descrivono il sistema $Q_{BT|RC}$ invece possiamo notare come la batteria si carichi molto più velocemente, tempo medio per passare allo stato di ricarica superiore $\frac{1}{5} = 0.2$, di quanto si scarichi, tempo medio per passare allo stato di ricarica inferiore $\frac{1}{2} = 0.5$.

2.4 Memoria e distribuzioni di fase

Le CTBN sono sistemi senza memoria. Tale proprieta è legata alla distribuzione esponenziale che ne definisce i tempi di transizione. Tale proprietà significa che lo stato $\mathbf{X}(t+s)$, ovvero lo stato che assume la variabile X al tempo $t+s$, dipende solamente dallo stato della variabile X al tempo t e non dallo stato $\mathbf{X}(t-v)$. Questa assunzione vale nel caso in cui ogni valore che viene assunto dalle componenti di $\mathbf{X}$ al tempo t è noto. L'assunzione di assenza di memoria costituisce però una limitazione importante in quanto limita grandemente l'espressività e la flessibilità del sistema. In molte applicazioni reali tale assunzione è poco credibile ed è pertanto necessario passare a forme più sofisticate di modellazione. Un sistema con assenza di memoria fallisce nel tener conto di modifiche temporali. Nell'esempio di un sistema con due variabili quali l'ingerimento di un certo farmaco e il dolore provato, con l'assunzione di assenza di memoria è impossibile prendere in considerazione la crescente diminuzione di efficacia del farmaco dovuta all'assuefazione.

La limitazione dovuta all'assenza di memoria viene risolta tramite le *distribuzioni di fase*. La restrizione delle CTBN che porta alla proprietà di assenza di memoria è la limitazione a modellare i tempi di durata in uno stato, quanto tempo trascorre la variabile in un certo stato, tramite una singola distribuzione esponenziale. Come riportato nelle sezioni precedenti infatti, una variabile resta in uno stato i per un periodo di tempo t che si distribuisce come una variabile esponenziale di parametro q_{x_i}. Le distribuzioni di fase (phase distribution) riescono a superare questa limitazione; esse sono una classe ricca e semiparametrica in grado, tramite un numero finito di stati, di modellare qualsiasi processo stocastico. Sono dunque molto potenti e versatili.

Tali distribuzioni si basano sulla distribuzione esponenziale e, informalmente, consistono di un insieme di fasi S tramite le quali il processo $X(t)$ considerato evolve. Ogni fase $s \in \mathbf{S}$ è associata con una distribuzione esponenziale che codifica il tempo

speso dal processo stocastico di interesse $X(t)$ in tale fase s. Quindi, il processo stocastico $X(t)$ entra in una fase s per poi lasciarla dopo un tempo t, realizzazione di una variabile esponenziale con parametro λ_s associato alla fase s.

Secondo *Nodelman et Al (2005)* [5], il processo stocastico $X(t)$ può essere rappresentato come un grafo diretto, che ammette cicli, consistente di un *insieme di fasi S*. Quindi, ci è permesso creare combinazoni di catene, misture e loop di tali fasi distribuite esponenzialmente in molti diversi modi. Il processo stocastico $X(t)$ spende una certa quantità di tempo per transizionare da uno stato s ad uno stato s' ma lascia infine l'insieme delle fasi S tutte assieme. La distribuzione complessiva con la quale il processo stocastico $X(t)$ lascia un certo insieme di fasi S si chiama *distribuzione di fase*.
La ricchezza della classe generale delle distribuzioni di fase può essere sfruttata per migliorare l'espressività delle reti Bayesiane standard per ottenere *Phase-Type continuous Bayesian network*.

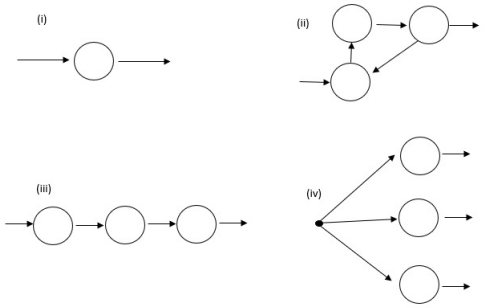


Figura 2.1: Quattro diverse tipologie di Phase-type distribution. Rispettivamente: (i) è una singola esponenziale: è presente una singola fase che corrisponde con lo stato. (ii) è un ciclo: è possibile ritornare da una fase s_j ad una fase s_i con $j > i$. Resta fissa in questa rappresentazione l'entrata nella distribuzione di fase tramite una fase di ingresso s_1 e l'uscita tramite l'ultima fase s_N. (iii) è una Erlangiana (catena) (iv) è una mistura.

Capitolo 3

Phase-Type continuous Bayesian network

Definition 3.0.1. Phase-Type continuous Bayesian network. Una continuous time Bayesian network N su $\mathbf{X}$ è una *Phase-Type continuous Bayesian network* se uno o più stati appartenenti a $Val(X_i)$ per una o più delle componenti $x_i \in \mathbf{X}$, sono associate con una distribuzione di fase.

Secondo Nodelman [3] esistono due alternative principali per sfruttare la classe generale delle distribuzioni di fase per ottenere delle Phase-Type continuous time Bayesian network. Tali alternative sono la *rappresentazione diretta* e la *rappresentazione Hidden variables*. Di seguito riporteremo un esempio di una rete semplice nelle due rappresentazioni per mostrare la differenza tra le due. Nell'esempio sottoriportato faremo riferimento ad una rete con $X \in \mathbf{X}$ nella rete Bayesiana a tempo continuo G senza alcun genitore, ed ogni stato $Val(X)$ supporremo essere composto dallo stesso numero di fasi N. La generalizzazione al caso più generale nel quale X ha dei genitori ed ogni stato x_i è associato con un diverso numero di fasi N_i è molto semplice e lineare.

Specificatamente, per mostrare come le rappresentazioni *diretta* e *Hidden variables* sfruttano le distribuzioni di fase, prendiamo $X \in \mathbf{X}$ una variabile casuale tale che $Val(X) = \{x_1, .., x_k\}$, e strutturiamo il tempo in modo tale che la durata media nel generico stato x_j, con $1 \leq j \leq k$, sia una variabile casuale che si distribuisce secondo una distribuzione di fase consistente di N_j fasi.
Come riportato nel paragrafo soprastante possiamo considerare $N_j = N$, lo stesso numero di fasi per ogni variabile senza perdita di generalità. Questa è una semplificazione illustrativa molto comoda, che porta anche ad una definizione più immediata della matrice, ma la sua generalizzazione in una rete con N_j fasi è diretta.

3.1 Rappresentazione diretta

La rappresentazione diretta [5] ci permette di descrivere la dinamica del processo stocastico $X(t)$ associato alla variabile casuale X usando una matrice di intensità della forma $[Nk \text{ x } Nk]$ con numero di righe e di colonne pari al prodotto del numero degli stati della variabile X di interesse e del numero di fasi N all'interno di ogni stato. La

strutturazione adottata viene mostrata nella tabella sottostante.

		x_1			x_j			x_n	
		$h_{1,1}$	$h_{N,1}$	.	$h_{1,j}$	$h_{N,j}$	.	$h_{1,k}$	$h_{n,k}$
x_1	$h_{1,1}$. $h_{N,1}$	Q_{x_1}		.	$Q_{x_1x_j}$		.	$Q_{x_1x_n}$	
x_j	$h_{1,j}$. $h_{N,j}$	$Q_{x_jx_1}$		.	Q_{x_j}		.	$Q_{x_jx_n}$	
x_k	$h_{1,k}$. $h_{N,k}$	$Q_{x_kx_1}$		.	$Q_{x_kx_j}$		.	Q_{x_k}	

Analizzando nel dettaglio la tabella sovrastante, notiamo che per ogni stato x_j, $j = 1, .., k$ della variabile X sono associate un insieme di N fasi $S_j = \{h_{1,j}, .., h_{N,j}\}$. Tali fasi formano il *sottosistema* della variabile x_j. Ovvero le transizioni che effettua restando sempre all'interno dello stato j-esimo. Ogni stato x_j è quindi associato con una *within system matrix* ed alcune *between system matrix*. La matrice *within system* Q_{x_j} associata a x_j descrive l'evoluzione nel sottosistema S_j all'interno di x_j. Le $k-1$ matrici *between system* associate sempre al generico x_j, $Q_{x_ix_k}$ con $i \neq j$ descrivono come avvengono le transizioni tra due stati x_i e x_j. Ovvero quando il sistema transiziona dal sottosistema S_i al sottosistema S_j.

La struttura all'interno di ogni *within subsystem matrix* ci permette di creare qualsivoglia tipo di catene, misture e cicli, tenendo come struttura fondamentale la distribuzione esponenziale. Possiamo ad esempio modellare una mistura di esponenziali con parametri θ differenti, oppure catene di esponziali o ancora cicli dove è possibile una transizione *backward* delle fasi. Definendo la struttura dei sottosistemi si definiscono anche i vincoli che vogliamo imporre al modello, ad esempio potremmo voler imporre l'impossibilità di formare un ciclo. Nella fase successiva di Learning vedremo che usando il sistema di *Expectation Maximization* non dovremo vincolarci a strutture straightforward ma ammetteremo anche la presenza di cicli.

Per quanto riguarda invece la struttura delle *between subsystem matrix*, esse definiscono la dinamica del processo stocastico $X(t)$ associato con la variabile X. E' possibile imporre vincoli anche in tal caso, ad esempio si potrebbe porre il vincolo che le transizioni tra x_i ed x_j avvengano solo tramite l'ultimo sottostato appartenente S_i al *primo* sottostato di S_j. Tale vincolo non causa perdite di generalità e apporta numerosi vantaggi di interpretabilità alla rete e viene pertanto usato frequentemente. In generale sotto tale vincolo vale la condizione che quando il sistema lascia lo stato x_j transiziona nello stato x_i con probabilità

$$\frac{q_{h_{N,j}h_{1,i}}}{q_{h_{N,j}} - \sum_{1 \leq l \leq N} q_{h_{N,j}h_{l,j}}}$$

Ovvero la proporzione tra l'intensità della transizione che ci interessa e l'intensità con la quale lascia lo stato x_j. Quest'ultima viene ricavata come differenza tra l'in-

tensità di lasciare lo stato $q_{h_{N,j}}$ *meno* la probabilità di transizionare all'interno del sottosistema S_j. Una nota importante sulle matrici *between system* e *within system* è che esse, prese da sole, non costituiscono un sistema valido in quanto non sommano a 0.

Per chiarirne meglio la struttura della matrice Q_x e delle sue decomposizioni in matrici whithin subsystem Q_{x_i} e between subsystem $Q_{x_i x j}$ presentiamo il seguente esempio.

Esempio 3.1 *Supponiamo che la variabile X assuma valori in $Val(X) = \{x_1, x_2, x_3, x_4\}$ e supponiamo che il numero di fasi sia uguale per ogni stato $N = 3$. Il tempo trascorso in uno stato x_j è una determinazione da una variabile casuale che si distribuisce secondo una distribuzione di fase consistente di tre fasi. Quindi ogni x_j è associato con un sottosistema $S_j = \{h_{1,j}, h_{2,j}, h_{3,j}\}$. Ne segue la matrice Q_x*

		x_1			x_2			x_3			x_4		
		$h_{1,1}$	$h_{2,1}$	$h_{3,1}$	$h_{1,2}$	$h_{2,2}$	$h_{3,2}$	$h_{1,3}$	$h_{2,3}$	$h_{3,3}$	$h_{1,4}$	$h_{2,4}$	$h_{3,4}$
x_1	$h_{1,1}$	-1	0.9	0.1	0	0	0	0	0	0	0	0	0
	$h_{2,1}$	1.2	-2	0.8	0	0	0	0	0	0	0	0	0
	$h_{3,1}$	0.5	1.5	-3	0.5	0	0	0.3	0	0	0.2	0	0
x_2	$h_{1,2}$	0	0	0	-4	3	1	0	0	0	0	0	0
	$h_{2,2}$	0	0	0	1	-5	4	0	0	0	0	0	0
	$h_{3,2}$	0.6	0	0	2	4	-8	1	0	0	0	0	0
x_3	$h_{1,3}$	0	0	0	0	0	0	-9	7	2	0	0	0
	$h_{2,3}$	0	0	0	0	0	0	-5	-7	2	0	0	0
	$h_{3,3}$	1	0	0	1	0	0	5	1	-9	1	0	0
x_4	$h_{1,4}$	0	0	0	0	0	0	0	0	0	-5	4	1
	$h_{2,4}$	0	0	0	0	0	0	0	0	0	1	-2	1
	$h_{3,4}$	1	0	0	0.5	0	0	0.5	0	0	2	3	-7

Quindi la matrice *within subsystem* relativa allo stato x_1 è la seguente:

$$Q_{x_1} = \begin{bmatrix} -1 & 0.9 & 0.1 \\ 1.2 & -2 & 0.8 \\ 0.5 & 1.5 & -3 \end{bmatrix}$$

Contiene quindi tutte le transizioni interne al sottosistema S_1. Si può notare come le righe non sommino a 0 ma nella terza riga la somma totale sia -1, il che significa che l'intensità complessiva con la quale il sistema abbandona lo stato x_1 dato che è in $h_{3,1}$ è pari ad 1.
Le corrisponenti matrici *between subsystem* sono invece

$$Q_{x_1x_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.5 & 0 & 0 \end{bmatrix} \qquad Q_{x_1x_3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.3 & 0 & 0 \end{bmatrix} \qquad Q_{x_1x_4} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.2 & 0 & 0 \end{bmatrix}$$

Sommando le righe della matrice within subsystem con le matrici between subsystem si ritorna ad avere le prime tre righe di un sistema valido, che sommano dunque a zero. Guardando queste matrici in sequenza si nota come è possibile effettuare un cambio di stato solamente se si è nell'N-esimo stato, il terzo, di x_1, e solo al primo sottostato della variabile in cui transiziona. In tale stato l'intensità di transizione è pari a 3, ma di questi con probabilità $\frac{0.5+1.5}{3} = 0.66$ la transizione avviene *all'interno* del sottosistema S_{x_1}, mentre, supponendo che il sistema transizioni in un sottosistema diverso le probabilità sono le seguenti: probabilità di transizionare da x_1 a x_2 data la transizione è pari a $\frac{0.5}{3-2} = 0.5$. Per quanto riguarda lo stato x_3 e x_4 rispettivamente le probabilità sono $\frac{0.3}{3-2} = 0.3$ e $\frac{0.2}{3-2} = 0.2$.
Questo sottosistema, come si vede nelle matrici within subsystem, implementa un ciclo, in quanto arrivati alla fase $h_{j,1}$ è permessa la transizione in una fase precedente $h_{i,1}$ con $i < j$.

3.2 Rappresentazione Hidden variables

Nodelman, in [5], propone l'uso di una variabile nascosta H_X come genitore della variabile X. Questo è un modo molto pulito per aggiungere potere espressivo alle reti Bayesiane a tempo continuo. Una tale rappresentazione ci permette di usare una distribuzione più complessa sulla traiettoria di X. Infatti permette alle intensità che controllano le dinamiche del processo stocastico $X(t)$ associate ad X di cambiare più frequentemente.

Sempre Nodelman nel suo paper del 2005 nota anche come per un numero fissato di fasi N una variabile X di una rete Bayesiana a tempo continuo modellata con rappresentazione diretta è sempre *strettamente più espressiva* rispetto alla stessa variabile modellata con rappresentazione Hidden con lo stesso numero di fasi N. Infatti molte transizioni che sono possibili con rappresentazione diretta *non* lo sono invece con rappresentazione Hidden variables.Il motivo è il vincolo in una rete Bayesiana che due variabili non possano transizionare contemporaneamente. Pertanto non è possibile avere uno scatto contemporaneo della variabile Hidden e della variabile X della quale la variabile nascosta H_X ne è il genitore. Questa limitazione potrà essere aggirata supponendo transizioni con intensità pari a ∞ il cui tempo medio di permanenza è quindi $\frac{1}{\infty} = 0$.

Nelle sezioni seguenti illustreremo due metodi per implementare la rappresentazione a Hidden variables, e infine le compareremo con la rappresentazione diretta in termini di espressività e complessità computazionale.

3.2.1 Full rappresentation

La rappresentazione full è il primo approccio basic alla rappresentazione Hidden variables.Per continuità con le sezioni precedenti prendiamo la medesima struttura di rete. X quindi resta uguale alla sezione precedente e H_X è una variabile che consiste di Nk stati, dove k rappresenta il numero di stati di X, ad esempio

$Val(H_X) = \{h_1, h_2, ..h_{Nk}\}$.
Sotto la rappresentazione full, in questo semplice esempio, lasciamo che la variabile nascosta H_X sia l'unico genitore di X. H_X e X formano un loop, sono entrambi genitore dell'altro.

Pertanto per definire il sistema formato da H_X e X dobbiamo definire le due matrici condizionali $Q_{X|H_X}$ e $Q_{H_X|X}$. Definiamo per prima le matrici $Q_{X|H_X}$ per $1 \leq j \leq k$.

Quando $l = N(j-1)+1$ le matrici sono le seguenti:

$$q_{x_i x_m|h_l} = \begin{cases} \text{-}\infty & 1 \leq i \leq k, i = m, i \neq j \\ \infty & 1 \leq i \leq k, j = m, i \neq j \\ 0 & \text{altrimenti} \end{cases}$$

Questa matrice serve solamente a *sincronizzare* X e riportarlo in accordo con H. Ovvero le transizioni nella variabile H_x che portano ad un cambiamento di sottosistema, ovvero di variabile X, producono una transizione istantanea ($tempo = \frac{1}{\infty} = 0$) che ha l'obbiettivo di riallineare X con H

Quando invece $l \neq N(j-1)+1$ la matrice di intensità è la matrice nulla $Q_{X|h_l} = \mathbf{0_{k,k}}$. Ovvero una matrice di dimensioni k x k di tutti zero.

Per quanto riguarda invece la matrice $Q_{X|H_X}$, i suoi elementi hanno la seguente distribuzione:

$$q_{h_l h_m|x_j} = \begin{cases} \text{-q}_{h_{i,j} h_{l,u}} & \text{N(j-1)}<\text{l}\leq Nj, \quad 1\leq m \leq Nk, l = m \\ \text{q}_{h_{i,j} h_{l,u}} & \text{N(j-1)}<\text{l}\leq Nj, \quad 1\leq m \leq Nk, l \neq m \\ 0 & \text{altrimenti} \end{cases}$$

Con gli indici u, i e l come segue:
$u := \lceil \frac{m}{N} \rceil$, $i := l - N(j-1)$, $l := m - N(u-1)$.

Volendo dare una spiegazione alla formula soprastante possiamo partire dalla condizione $N(j-1) < l \leq Nj$, che significa che abbiamo delle intensità *diverse* da zero solamente nelle righe della matrice $Q_{H_x|X}$ corrisponenti allo stato x_j a cui ci condizioniamo. A titolo di esempio, sono le prime tre righe per x_1, dalla quarta alla sesta riga per x_2 e così via. Un altra considerazione è che la prima riga della formula tratta gli elementi diagonali della nostra matrice, pertanto quelli con intensità negativa, mentre la seconda riga della formula tratta gli elementi *fuori* dalla diagonale.
E' importante notare come la rappresentazione full incrementi il numero di stati della matrice $Q_{H_x|X}$ da N, come nella rappresentazione diretta, a Nk. Tutto ciò avviene mentre il numero di parametri liberi, da stimare successivamente tramite la fase di learning, restano i medesimi.

Esempio 3.2.1 *Consideriamo la stessa struttura vista precedentemente nell'esempio 3.1. La variabile X assume sempre valori $Val(X) = \{x_1, x_2, x_3, x_4\}$ mentre la variabile Hidden H_X assume valori $Val(H_X) = \{h_1, ..., h_{12}\}$.*

Dalla prima condizione $l = N(j-1)+1$ *otteniamo che* $Q_{X|h_l} = \mathbf{0_{4,4}}$ *per* $l \neq \{1,4,7,10\}$. *Inoltre*

$$Q_{X|h_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \infty & -\infty & 0 & 0 \\ \infty & 0 & -\infty & 0 \\ \infty & 0 & 0 & -\infty \end{bmatrix} \qquad Q_{X|h_4} = \begin{bmatrix} -\infty & \infty & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \infty & -\infty & 0 \\ 0 & \infty & 0 & -\infty \end{bmatrix}$$

$$Q_{X|h_7} = \begin{bmatrix} -\infty & 0 & \infty & 0 \\ 0 & -\infty & \infty & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \infty & -\infty \end{bmatrix} \qquad Q_{X|h_{10}} = \begin{bmatrix} -\infty & 0 & 0 & \infty \\ 0 & -\infty & 0 & \infty \\ 0 & 0 & -\infty & \infty \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Che rispettivamente sincronizzano $X = x_1, x_2, x_3, x_4$ se H_X assume i valori h_1, h_4, h_7, h_{10} abilitando una transizione immediata.

Per quanto riguarda $Q_{H_X|x_j}$ abbiamo:

$$\mathrm{Q}_{H_X|x_1} = \begin{bmatrix} -1 & 0.9 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.2 & -2 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 1.5 & -3 & 0.5 & 0 & 0 & 0.3 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathrm{Q}_{H_X|x_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -5 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.6 & 0 & 0 & 2 & 4 & -8 & 1 & 0 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$
Q_{H_X|x_3} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -9 & 7 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 5 & -7 & 2 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 5 & 1 & -9 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
Q_{H_X|x_4} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -5 & 4 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\
1 & 0 & 0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 & 2 & 3 & -7
\end{bmatrix}
$$

Queste matrici mostrano le transizioni che avvengono in H_X condizionate ai quattro stati che può assumere x.

Le due rappresentazioni, diretta e full, sono equivalenti in quanto rappresentano, in modo diverso e con matrici diversi, lo stesso processo stocastico $X(t)$. Si può dimostrare tale equivalenza comparando il loro *diagramma di transizione di stato*. Mostriamo innanzitutto il diagramma di transizione di stato corrispondente alla *rappresentazione diretta*.

Il rettangolo colorato in bianco rappresenta l'i-esimo sottosistema S_i. Ogni sottosistema è composto da N fasi ordinate. Mostriamo come il sistema descritto consenta transizioni da e verso ogni stato del sottosistema. Nel diagramma le transizioni all'interno del sottosistema sono identificate da una riga tratteggiata. Le transizioni che portano ad un cambio del sottosistema sono invece marcate da una linea continua. Come mostrato in figura il sistema entra nell'i-esimo sottosistema tramite la fase di ingresso h_{i1} ed esce dal sottosistema mediante la fase di uscita h_{iN}.

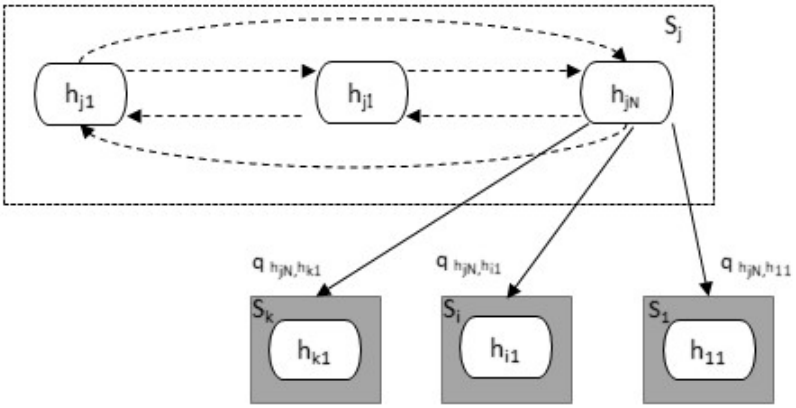


Figura 3.1: Diagramma di transizione della rappresentazione *diretta* di $X(t)$

Nella figura sottostante viene invece mostrato il diagramma di transizione della rappresentazione full del processo $X(t)$. Esso è l'approccio Naive alla rappresentazione a variabile nascosta. Vediamo che il sistema descritto è *sostanzialmente* identico rispetto all'approccio precedentemente preso in esame della rappresentazione diretta. La fila intermedia di blocchi di tale rappresentazione è semplicemente una fila di stati *transienti*, altresì detti *vanishing states*. Sono stati nel quale il sistema in media non si sofferma, la cui unica funzione è quella di abilitare la transizione da uno stato $h_{jN}x_j$ ad uno stato $h_{i1}x_i$, $\forall i, j \in 1..k$. Tale transizione non è rappresentabile mediante una singola intensità in quanto uno dei vincoli delle reti Bayesiane a tempo continuo è che solamente una variabile alla volta (quindi solamente h o x) possa transizionare. Pertanto l'artificio che si usa è scomporre questa doppia transizione in h e in x in due transizioni nelle quali cambia solamente una delle due variabili, con lo stato raggiunto dopo la prima transizione che è uno stato transiente nel quale il sistema non spende (in media) tempo. Pertanto, la velocità con la quale il sistema transiziona da uno stato $h_{jN}x_j$ ad uno stato $h_{i1}x_i$ è regolata solamente dall'intensità $q_{h_{jN}h_{i1}}$.

Come descritto nei paragrafi soprastanti possiamo concludere che la rappresentazione diretta e full sono *equivalenti* ed hanno lo stesso numero $k(N(N-1)+(k-1))$ di parametri liberi. Pertanto la complessità di apprendimento della rete è la medesima nelle due rappresentazioni.

Tuttavia il vantaggio interpretativo e inferenziale della rappresentazione full, porta ad un incremento della complessità, per via dell'aumento delle dimensioni della matrice amalgamata. L'amalgamazione è una combinazione tra due CIM per produrre una singola CIM più grande; tale operazione serve per riuscire a vedere un intero sistema come un singolo processo Markoviano e verrà approfondita nelle sezioni seguenti.
Per quanto riguarda la matrice Q_X della rappresentazione diretta essa è già di per sè la matrice *amalgamata* del processo ed ha numero di entrate $Q_X = [kN \text{ x } kN]$. Per quanto riguarda invece la rappresentazione full, l'amalgamata è la combinazione tra le due variabili che compongono il sistema, H_X e X.

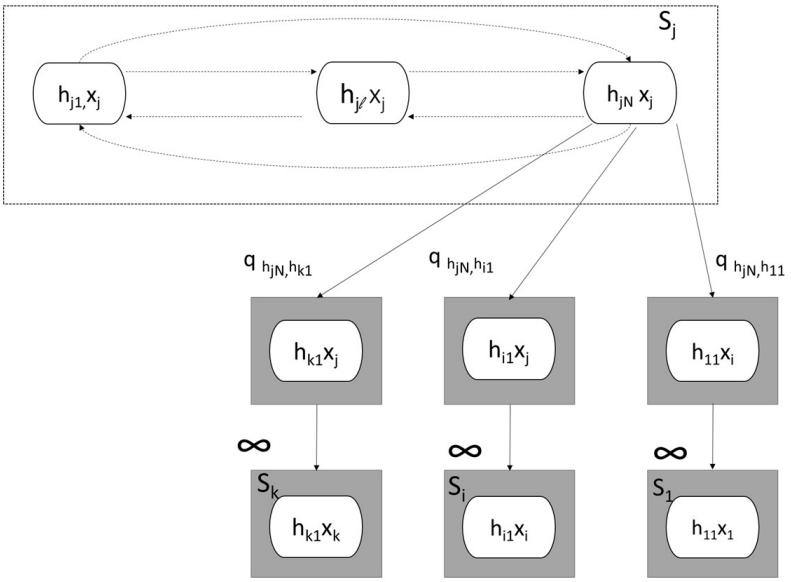


Figura 3.2: Diagramma di transizione della rappresentazione *full* di $X(t)$

La matrice amalgamata $Q_{H_X X}$ risulterà avere $[k^2N$ x $k^2N]$ entrate. Quindi il numero di entrate, passando dalla rappresentazione diretta alla rappresentazione full, si incrementa di un fattore quadrato. Tale incremento nelle dimensioni della matrici di intensità porta ad un aumento del tempo necessario per l'inferenza.

3.2.2 Rappresentazione bipartite

La *rappresentazione bipartite* riduce il numero di stati nascosti usati dalla rappresentazione full. Riesce a raggiungere questo risultato scambiando alternativamente la *fase di ingresso* e la *fase di uscita* della distribuzione di fase.

L'idea di base dietro tale rappresentazione è che l'incremento del numero di stati di cui soffre la rappresentazione full è dovuta al vincolo delle reti Bayesiane a tempo continuo per il quale solamente una variabile alla volta può cambiare il suo stato in un determinato istante temporale. Pertanto la variabile X e la sua variabile Hidden H_X non possono cambiare contemporaneamente i loro valori. La soluzione proposta dalla rappresentazione full, che ricordiamo essere la soluzione semplicistica, è di incrementare il numero di righe della variabile nascosta H_X da N - il numero di fasi- a Nk, ovvero il prodotto tra numero di fasi e numero di stati di X.

Esiste però una soluzione al problema della doppia transizione che consente di ridurre il numero di righe per la variabile H_X. Tale metodo si basa sul partizionare l'insieme degli stati di X, $Val(X) = \{x_1, ...x_k\}$ in due sottoinsiemi disgiunti A e B, con numero di stati uguali, per quanto possibile, nei due sottoinsiemi. Per quanto possibile significa che nel caso in cui il numero k di stati è pari, allora essi verrano divisi equamente nei due sottoinsiemi, con la convenzione che gli stati demarcati dall'indice dispari (es: x_1, x_3) confluiranno in A e i restanti in B. Contrariamente, con k dispari,

lasciamo per convenzione i $\lceil k \rceil$ stati *dispari* in A e i rimanenti $\lfloor k \rfloor$ stati *pari* in B.

La riduzione del numero di stati della variabile H_X viene ottenuta dalla seguente scelta di costruzione. Per tutti gli stati x_i appartenenti ad A poniamo h_1 come fase di ingresso (es: $x_i h_1$) e h_N come fase di uscita. Mentre per gli stati appartenenti a B poniamo h_N come fase di ingresso e h_1 come fase di uscita, invertendo l'ordine designato nel sottoinsieme di stati A. Questa scelta di costruzione della rete ci permette di definire le transizioni $x_i \to x_j$ da uno stato $x_i \in A$ ad uno stato $x_j \in B$ come transizioni *legittime*. Non necessitanti dunque di stati transienti di supporto. Ciò avviene in quanto la transizione $x_i \to x_j$, per costruzione della rete, significa una transizione dalla fase di uscita h_N dello stato x_i, ovvero $x_i h_N$, alla fase di *entrata* $\mathbf{h_N}$ dello stato x_j, ovvero $x_j h_N$. Pertanto solamente **una** delle due variabili che compongono il sistema, nello specifico X, modifica il suo stato nella transizione, rendendola perfettamente legittima in una rete Bayesiana a tempo continuo.
L'inversione di fase di entrata e uscita nei due sottoinsieme A e B ci permette dunque di rendere legittime transizioni che erano vietate nella rappresentazione full, con conseguente risparmio di numero di entrate per H_X. Allo stesso modo, la transizione $x_j \to x_i$, con $x_j \in B$ e $x_i \in A$ è legittima. Significa una transizione dalla fase di uscita h_1 dello stato x_j, ovvero $x_j h_1$, alla fase di *entrata* $\mathbf{h_1}$ dello stato x_i, ovvero $x_i h_1$. Anche in questo caso solamente la variabile X si modifica nella transizione, portando alla legittimità della stessa.

Per illustrare meglio il funzionamento della rappresentazione bipartite riprendiamo l'esempio precedente con la variabile X.

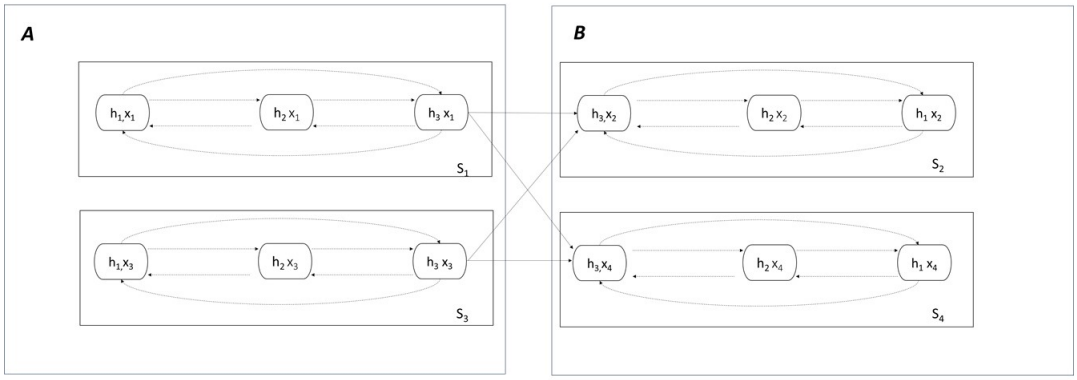


Figura 3.3: Grafico associato con la rappresentazione bipartite di $X(t)$

Nella figura soprariportata si può osservare come le righe tratteggiate, che vanno da un sottosistema all'altro, corrispondano tutte a transizioni legittime. Tutte le transizioni tra fasi di *uscita* del sottoinsieme A e di entrata di B sono legittime. Altre transizioni legittime, specificatamente quelle tra fasi di *uscita* del sottoinsieme B e di entrata di A, non sono state riportate nel grafico.
Non sono legittime invece le transizioni tra stati appartenti allo stesso sottoinsieme (es: transizioni $x1 \rightarrow x3$, poichè appartengono tutti e due ad A).

Nella figura sottostante invece mostriamo il meccanismo tramite il quale è possibile una transizione tra due stati S_i differenti, nel caso in esame $S_1 \rightarrow S_3$, appartenenti allo stesso sottoinsieme A.

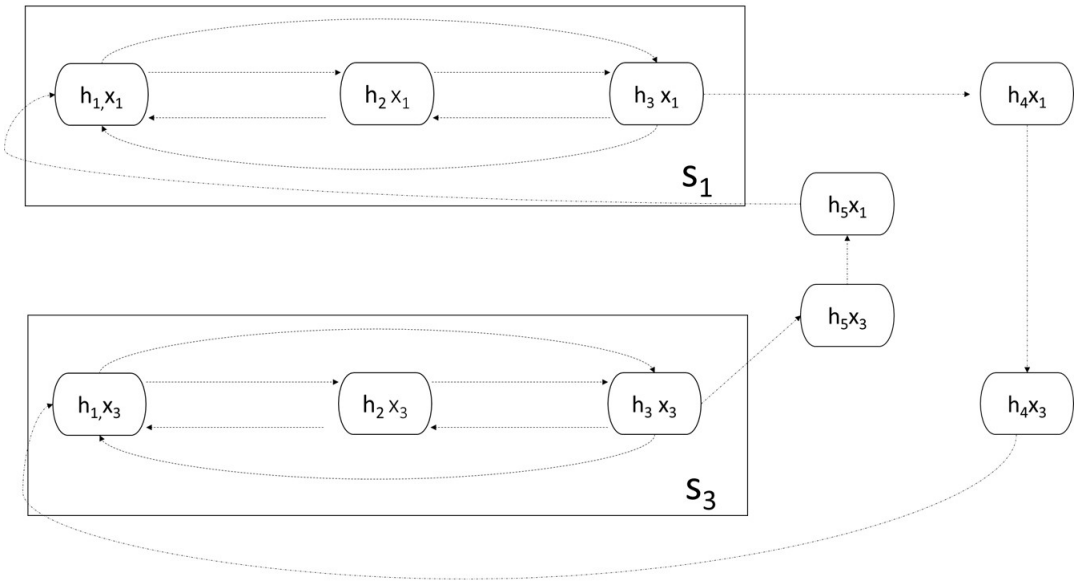


Figura 3.4: Diagramma bipartite di $X(t)$

Possiamo vedere che il meccanismo è il medesimo che nella raprpesentazione full. La differenza fondamentale è invece la presenza di *due*, e non solamente uno, fasi *vanishing*, ad esempio h_4x_3-h_4x_1, che permettono al sistema di compiere la transizione ma nei quale il sistema passa un tempo atteso pari a zero.

Il numero di fasi da aggiungere alla variabile Hidden H_X nella rappresentazione bipartite dipende dal numero di stati della variabile X, poichè essi determinano il numero di transizioni non legittime.
Considerando k stati per la variabile X, il numero di transizioni *legittime* sotto la rappresentazione bipartite è dato dalla somma tra il numero di connessioni da variabili $x_i \in A$ a $x_j \in B$ con il numero di connessioni da variabili $x_j \in B$ a $x_i \in A$.
Ognuno di questi addendi è pari a $\lceil \frac{k}{2} \rceil \lfloor \frac{k}{2} \rfloor$, pertanto la loro somma è:

$$2(\left\lceil \frac{k}{2} \right\rceil \left\lfloor \frac{k}{2} \right\rfloor)$$

Il numero di transizioni *non legittime* è invece dato dalla somma tra $\lceil \frac{k}{2} \rceil (\lceil \frac{k}{2} \rceil - 1)$,il numero di transizioni non legittimi del sottoinsieme A, e $\lfloor \frac{k}{2} \rfloor (\lfloor \frac{k}{2} \rfloor - 1)$, il corrispettivo nel sottinsieme B. La loro somma è dunque pari a:

$$\left\lceil \frac{k}{2} \right\rceil (\left\lceil \frac{k}{2} \right\rceil - 1) + \left\lfloor \frac{k}{2} \right\rfloor (\left\lfloor \frac{k}{2} \right\rfloor - 1) = v_a + v_b = v$$

La rappresentazione bipartite richiede che la variabile Hidden H_X abbia numero di stati *pari* a $N + v$. Rispetto alla rappresentazione full, il risparmio è notevole, soprattutto per N grande: si passa da Nk stati a $N + v$.

Definiamo ora gli elementi delle due matrici condizionali che definiscono il sistema, $Q_{X|H_X}$ e $Q_{H_X|X}$. Gli elementi della matrice $Q_{X|H_X}$ sono definiti nel modo seguente:

$$q_{x_i x_j|h_1} = \begin{cases} \text{-}\sum_{1 \le l \le k} q_{h_{N,i} h_{1,l}} & 1 \le i = j \le k, i\ pari, l\ dispari \\ q_{h_{N,i} h_{1,j}} & 1 \le i \ne j \le k, i\ pari, j\ dispari \\ 0 & \text{altrimenti} \end{cases}$$

$$q_{x_i x_j|h_N} = \begin{cases} \text{-}\sum_{1 \le l \le k} q_{h_{N,i} h_{1,l}} & 1 \le i = j \le k, i\ dispari, l\ pari \\ q_{h_{N,i} h_{1,j}} & 1 \le i \ne j \le k, i\ dispari, j\ pari \\ 0 & \text{altrimenti} \end{cases}$$

Queste sono le definizioni *element-wise* delle matrici $Q_{X|H}$ condizionate alle fasi h_1 e h_N, la prima e l'ultima fase. Per quanto riguarda la definizione *element-wise* delle matrici condizionate alle fasi "interne", ovvero per $1 < l < N$, abbiamo la matrice $Q_X|h_l = \mathbf{0_{k,k}}$, la matrice nulla [$k$ x k]. Per quanto riguarda le matrici per $N < l \le N + v$, la definizioni delle matrici è la seguente, per $1 \le i, j \le k$:

$$q_{x_i x_j|h_l} = \begin{cases} \text{-}\infty & \text{i=j=a} \\ \infty & \text{i=a, j=b} \\ 0 & \text{altrimenti} \end{cases}$$

con a e b definiti nel seguente modo.

$(a, b) = f^{-1}(l - N)$, dove f è una funzione che mappa gli interi a e b su numeri naturali crescenti, ognuno dei quali corrispondente ad una transizione non legittima. In particolare questa mappatura segue le regole sottostanti di ordinamento $f(a, b)$ per $a < b$:

- La mappatura avviene sui numeri naturali dispari $2\mathbb{Z} + 1$

- a è la variabile primaria per l'ordinamento. es. ad $(a,b) = (1,7)$ verrà assegnato un numero $f(1,7)$ certamente *inferiore* rispetto a $(a,b) = (2,4)$.
- b è la variabile secondaria per l'ordinamento: a parità di a, per $b' < b''$ $f(a,b')$ assegna un numero *minore* rispetto a $f(a,b'')$.

Per $a > b$ invece $f(a,b) = f(b,a) + 1$, pertanto opera una mappatura sui numeri pari. Tale regola verrà esemplificata nelle sezione successiva.

Di seguito illustreremo la costruzione *element-wise* delle matrici $Q_{H_X|X}$. Per poter definire tale matrice, e in particolare le transizioni con intensità pari a ∞ sarà necessario definire la funzione s. Tale funzione $s(k,N,j)$ ha come parametri k numero di stati di X, N numero di fasi e un indice j con $j = 1..k$.
La funzione, per un determinato x_j, restituisce un vettore di numeri corrispondenti agli indici i delle fasi h_i tramite le quali avvengono le transizioni non legittime da x_j. Esistono due definizioni differenti per x_j pari e dispari. La prima formula mostrata è per j dispari:

$$q_{h_l h_m|x_j} = \begin{cases} -q_{h_{l,j}h_{l,j}} & i \leq l = m < N \\ -q_{h_{N,j}h_{N,j}} + \sum_u q_{h_{N,j}h_{N,u}} & l = m = N,\ 1 \leq u \leq k, u\ pari \\ q_{h_{l,j}h_{m,j}} & i \leq l \neq m \leq N \\ q_{h_{N,j}h_{1,b}} & l = N, N < m \leq N + v_A, \\ & (a,b) = f^{-1}(m-N), a = j \\ -\infty & N < l \leq N + v_A, l \in s, m = l \\ \infty & N < l \leq N + v_A, l \in s, m = 1 \\ 0 & \text{altrimenti} \end{cases}$$

Per j pari invece:

$$q_{h_l h_m|x_j} = \begin{cases} -q_{h_{N-l+1,j}h_{N-l+1,j}} & i < l = m \leq N \\ -q_{h_{N,j}h_{N,j}} + \sum_u q_{h_{N,j}h_{1,u}} & l = m = 1,\ 1 \leq u \leq k, u\ dispari \\ q_{h_{N-l+1,j}h_{N-m+1,j}} & i \leq l \neq m \leq N \\ q_{h_{N,j}h_{1,b}} & l = 1, N + v_A < m \leq N + v, \\ & (a,b) = f^{-1}(m-N), a = j \\ -\infty & N + v_A < l \leq N + v, l \in s, m = l \\ \infty & N + v_A < l \leq N + v, l \in s, m = N \\ 0 & \text{altrimenti} \end{cases}$$

Per illustrare meglio la rappresentazione *bipartite* e le formule soprastanti presentiamo l'esempio seguente.

Esempio 3.2.2 *Consideriamo la stessa struttura vista precedentemente nell'esempio 3.1. La variabile X assume sempre valori $Val(X) = \{x_1, x_2, x_3, x_4\}$ mentre la variabile Hidden H_X assume valori $Val(H_X) = \{h_1, ..., h_7\}$. Quindi, abbiamo $Q_{X|h_2} = \mathbf{0_{4,4}}$.*

$$Q_{X|h_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.6 & -1.6 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0.5 & -1.5 \end{bmatrix} \qquad Q_{X|h_2} = \begin{bmatrix} -0.7 & 0.5 & 0 & 0.2 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q_{X|h_3} = \begin{bmatrix} -\infty & 0 & \infty & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad Q_{X|h_4} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \infty & 0 & -\infty & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q_{X|h_5} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\infty & 0 & \infty \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad Q_{X|h_6} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \infty & 0 & -\infty \end{bmatrix}$$

Per quanto riguarda invece le matrici $Q_{H_X|X}$:

$$Q_{H_X|x_1} = \begin{bmatrix} -1 & 0.9 & 0.1 & 0 & 0 & 0 & 0 \\ 1.2 & -2 & 0.8 & 0 & 0 & 0 & 0 \\ 0.5 & 1.5 & -2.3 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 & -\infty & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad Q_{H_X|x_2} = \begin{bmatrix} -6.4 & 4 & 2 & 0 & 0 & 0.4 & 0 \\ 4 & -5 & 1 & 0 & 0 & 0 & 0 \\ 1 & 3 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \infty & 0 & 0 & 0 & -\infty \end{bmatrix}$$

$$Q_{H_X|x_3} = \begin{bmatrix} -9 & 7 & 2 & 0 & 0 & 0 & 0 \\ 5 & -7 & 2 & 0 & 0 & 0 & 0 \\ 5 & 1 & -7 & 0 & 1 & 0 & 0 \\ \infty & 0 & 0 & -\infty & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad Q_{H_X|x_4} = \begin{bmatrix} -5.5 & 3 & 2 & 0 & 0 & 0 & 0.5 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & -5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \infty & 0 & 0 & -\infty & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

La funzione $f(a, b)$ in tale esempio è definita nel modo seguente:

$$f(a,b) = \begin{cases} 1 & \text{a=1,b=3} \\ 2 & \text{a=3,b=1} \\ 3 & \text{a=2,b=4} \\ 4 & \text{a=4,b=2} \end{cases}$$

Pertanto la funzione inversa $f^{-1}(c)$ è definita nel seguente modo:

$$f^{-1}(c) = \begin{cases} (1,3) & \text{c=1} \\ (3,1) & \text{c=2} \\ (2,4) & \text{c=3} \\ (4,2) & \text{c=4} \end{cases}$$

Per quanto riguarda invece la funzione $s(K, N, j)$, essa è definita nel seguente modo:

$$s(K = 4, N = 3, j) = \begin{cases} 5 & \text{j=1} \\ 7 & \text{j=2} \\ 4 & \text{j=3} \\ 6 & \text{j=4} \end{cases}$$

3.2.3 Comparare la complessità delle rappresentazioni

Nelle sezioni precedenti abbiamo mostrato l'equivalenza dei sistemi delle due rappresentazioni Hidden, la full e la bipartite, e la rappresentazione diretta. La rappresentazione più efficiente è comunque la rappresentazione *diretta*. E' più efficiente in quanto richiede il minor numero di stati delle matrici di intensità per modellare il processo $X(t)$. I vantaggi derivanti dall'adottare un approccio Hidden derivano dalla semplicità interpretativa, inferenziale e dalla capacità di aggiungere in maniera lineare e pulita ulteriore complessità al modello.

Per confrontare tra loro le complessità derivanti dai tre differenti approcci, in settaggi differenti di N e k, prendiamo una variabile $X \in \mathbf{X}$ con $Val(X) = \{x_1, .., x_k\}$. Supponiamo che il tempo che la variabile trascorre in uno stato x_j $(1 \leq j \leq k)$ prima di lasciarlo si distribuisca come una distribuzione di fase consistente di N fasi. La matrice delle intensità Q_X sotto rappresentazione diretta consiste di Nk stati, ovvero è proporzionale linearmente in ognuna delle due variabili N e k, fissata l'altra. Sotto la rappresentazione full il numero degli stati della matrice amalgamata delle intensità $Q_{H_X X}$ è pari a k^2N. Infine, sotto rappresentazione bipartite la matrice amalgamata delle intensità $Q_{H_X X}$ ha numero di stati uguale a $k(N + v)$, dove $v = \lceil \frac{k}{2} \rceil (\lceil \frac{k}{2} \rceil - 1) + \lfloor \frac{k}{2} \rfloor (\lfloor \frac{k}{2} \rfloor - 1)$. Spezzando la formula si ottiene $kN + kv$, con $v \propto k^2$. Tale rappresentazione "soffre", aumenta la complessità, all'aumentare del numero di stati k, ma è più robusta all'aumentare del numero di fasi N.

La tabella che segue riassume il numero di stati della matrice amalgamata nelle diverse rappresentazione sotto diverse configurazioni di N e k.

	Numero di stati della matrice di intensità amalgamata								
	N=3			N=10			N=20		
k	diretta	full	bipartite	diretta	full	bipartite	diretta	full	bipartite
2	6	12	6	20	40	20	40	80	40
3	9	27	15	30	90	36	60	180	66
4	12	48	28	40	160	56	80	320	96
5	15	75	55	50	250	90	100	500	140
6	18	108	90	60	360	132	120	720	192
7	21	147	147	70	490	196	140	980	266
8	24	192	216	80	640	272	160	1280	352
9	27	243	315	90	810	378	180	1620	468
10	30	300	430	100	1000	500	200	2000	600

La tabella soprastante mostra come per k basse, la rappresentazione bipartite è *decisamente* più efficiente rispetto alla rappresentazione full. All'aumentare del numero di stati k questo vantaggio si riduce fino ad annullarsi completamente.

Per $N = 3$, il vantaggio si annulla per $k = 7$, mentre per N maggiori, il vantaggio diminuisce molto più gradualmente e anche per $k = 10$ la rappresentazione bipartita ha meno della metà degli stati rispetto alla rappresentazione full. Per valori di N elevati, nella tabella si mostrano $N = 10$ e $N = 20$, la rappresentazione bipartite è altamente competitiva con la rappresentazione diretta, almeno per k modeste.

Il modo ottimale per sfruttare le potenzialità della rappresentazione bipartite introdotta è attraverso una configurazione con k modesto, $2 \leq k \leq 5$, e numero di fasi N elevate, $N \geq 10$. Pertanto una situazione nella quale la variabile X ha pochi stati possibili ma il tempo trascorso in ognuno di tali stati ha una *distribuzione complessa* e con memoria, impossibile da rappresentare con una semplice esponenziale ($N = 1$) o con poche fasi.

Degna di nota è l'osservazione che per $k = 2$, ovvero una variabile X *binaria*, il numero di stati della rappresentazione *bipartite* è uguale al numero di stati della rappresentazione diretta, ma è un modo più pulito per introdurre le distribuzioni di fase.

3.3 Amalgamazione

La matrice amalgamata di un processo è una matrice che rappresenta *tutto* il processo. L'amalgamazione consente di visualizzare l'intero processo ed è utile per *sommare* tra di loro più processi. Per definire l'operazione di amalgamazione è necessario dare delle definizioni preliminari. L'obbiettivo è quello di mettere assieme più matrici condizionali, in modo da espanderle in una singola matrice su *tante variabili*. Dopodichè descriveremo l'operazione di amalgamazione che porta ad avere una matrice di intensità *congiunta* che esprime il comportamento di una CTBN come un processo di Markov omogeneo.

3.3.1 Variable Orderings

Tale metodo fornisce una *mappatura*, un legame, tra numeri di riga/colonna e instanziazioni della variabile. Tale mappatura è definita dal variable orderings. In altre

parole, è un metodo per associare in maniera univoca ogni cella della matrice finale ad una variabile, e riesce ad effettuare questa associazione tramite un *ordinamento* delle variabili.

Definition 3.3.1. ϵ_s. Se ϵ_s è il *variable orderings* per le variabili dell'insieme **S**, allora ϵ_s può essere visto come mapping dai numeri di riga ad una instanzazione $S \in \mathbf{S}$, adottando la convenzione di iterare sulla prima variabile, poi sulla seconda, fino alla s-esima variabile.

Esemplificando, $\epsilon_s[i]$ è l'i-esima instanzazione con mappatura ϵ dell'insieme **S**.

Proprietà 3.3.1 *Consistenza. Siano ϵ_s e $\epsilon_s{}'$ due ordinamenti su* **S** *e* **S** $'$ *distinti. Essi sono* ***consistenti*** *se hanno lo stesso ordine nelle variabili di intersezione* $\mathbf{S} \cap \mathbf{S}'$ *e lo scriviamo* $\epsilon_s[i] \cong \epsilon_s{}'[k]$

Esempio 3.3.1 *Consideriamo quattro variabili binarie X, Y, Z, W. Assumono quindi rispettivamente valori:*
$Val(X) = \{x_1, x_2\}$, $Val(Y) = \{y_1, y_2\}$, $Val(Z) = \{z_1, z_2\}$, $Val(W) = \{w_1, w_2\}$.
L'ordinamento ϵ_s è definito come $\epsilon_s =< X, Y, Z >$. Pertanto la sequenza degli stati di ϵ_s è:

$\epsilon_s[1] =< x_1, y_1, z_1 >$; $\epsilon_s[2] =< x_2, y_1, z_1 >$; $\epsilon_s[3] =< x_1, y_2, z_1 >$; $\epsilon_s[4] =< x_2, y_2, z_1 >$;

$\epsilon_s[5] =< x_1, y_1, z_2 >$; $\epsilon_s[6] =< x_2, y_1, z_2 >$; $\epsilon_s[7] =< x_1, y_2, z_1 >$; $\epsilon_s[8] =< x_2, y_2, z_1 >$;

ϵ_s è **consistente** con un ordinamento $\epsilon'_s =< Y, W, Z >$ in quanto le variabili comuni ai due ordinamenti, $< Y, Z >$ hanno lo stesso ordinamento sia in ϵ_s che in $\epsilon_s{}'$.

Definition 3.3.2. Possiamo dire che due instanziazioni $\epsilon_s[j]$ e $\epsilon_s{}'[k]$ sono consistenti se assegnano lo stesso valore alle variabili nell'intersezione $S \cap S'$.

Possiamo notare come le instanziazioni possono essere consistenti anche se gli ordinamenti ϵ_s e $\epsilon_s{}'$ sono inconsistenti. Per chiarire meglio la definizioni, $\epsilon_s{}'[j] \cong \epsilon_s[i]$ significa che le variabili nell'intersezione $S \cap S'$ assumono in $\epsilon_s{}'[j]$ ed in $\epsilon_s[i]$ lo stesso valore.

3.3.2 Espansione delle CIM

Prima di combinare le CIM per amalgamazione abbiamo bisogno di scriverle come singole matrici sullo *stesso* insieme di variabili. Una CIM $Q_{S|C}$ sulle variabili $S \subseteq X$ condizionate a $C \subset X$ definisce la dinamica di S dato C. Possiamo riscrivere $Q_{S|C}$ come una singola matrice a blocchi sullo spazio congiunto $Val(S)$ x $Val(C)$

$$Q_{S|C} = \begin{bmatrix} Q_{S|c_1} & 0 & .. & 0 & 0 \\ 0 & Q_{S|c_2} & .. & 0 & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & .. & Q_{S|c_{N-1}} & 0 \\ 0 & 0 & .. & 0 & Q_{S|C_N} \end{bmatrix}$$

con $c_1, ..c_N$ gli stati assunti dalle variabili nell'insieme C. La matrice sovrastante è la CIM espansa sulle variabili $S \cup C$.

La CIM scritta in tale maniera induce una distribuzione sulla dinamica delle variabili S data la traiettoria delle variabili C. Per definire in modo formale questo processo di espansione, definiamo l'eespansione della CIM in termini di matrice di espansione $N_{S|C}$.

Definition 3.3.3. Fissiamo $\epsilon_{s,c}$ un ordinamento, e siano ϵ_s e ϵ_c dei sottordinamenti consistenti. Sia $c_i = \epsilon_c[i]$. Allora, la matrice espansa $N_{S|c_i}$ è una matrice rettangolare di dimensioni $Val(S \cup C)$ dove l'entrata $[j,k]$ è definita come:

$$N_{S|c_i}[j,k] = \begin{cases} 1 & \text{se} \quad \epsilon_{s,c}[j] \cong \epsilon_c[i] \; e \\ & \qquad \epsilon_{s,c}[j] \cong \epsilon_s[k] \\ 0 & \text{altrimenti} \end{cases}$$

Considerata la definizione, $N_{S|c_i}$ è ovunque 0 tranne nella sottomatrice quadrata definita dall'insieme delle righe consistente con c_i. In tale sottomatrice quadrata $N_{S|c_i}$ è la matrice identità; in altre parole $N_{S|c_i}$ è una matrice con 0 ovunque tranne lungo la diagonale principale corrispondente alle entrate dove le righe e le colonne sono consistenti alle istanziazioni di S.

Un esempio è ora di rigore per chiarire meglio le dinamiche di funzionamento della matrice di espansione sopra definita.

Esempio 3.3.2 *Consideriamo due variabili binarie X, Y. Con rispettivamente $Val(X) = \{x_1, x_2\}$, $Val(Y) = \{y_1, y_2\}$ e l'ordinamento $< X, Y >$.*
Quindi, la sequenza degli stati per le righe è: $< x_1, y_1 >$; $< x_2, y_1 >$; $< x_1, y_2 >$; $< x_2, y_2 >$. Le matrici di espansioni sono dunque le seguenti:

$$N_{X|y_1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad N_{X|y_2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Abbiamo inoltre:

$$N_{Y|x_1} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad N_{Y|x_2} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Su tali matrici possiamo osservare che per $N_{X|y_1}$, solamente le prime due righe hanno entrate diverse da 0. In quanto per l'ordinamento $< X, Y >$ solamente le prime righe hanno $Y = y_1$, e sono quindi consistenti. Allo stesso modo possiamo notare

come in $N_{Y|x_1}$ solamente la prima e la terza riga non sono nulle, corrisponenti alle righe dove l'ordinamento $< X, Y >$ ha $X = x_1$.

Avendo definito ed esemplificato le matrici di espansioni siamo ora pronti a definire l'espansione di una generica CIM $Q_{S|C}$ in una singola matrice.

Definition 3.3.4. La forma espansa di $Q_{S|C}$ dato l'ordinamento $\epsilon_{s,c}$ è una matrice di dimensioni $Card(S \cup C)$ x $Card(S \cup C)$ definita in termini delle matrici $Q_{S|c_i}$ di dimensioni $Card(S)$ x $Card(S)$ con la seguente formula:

$$Q_{S|C} = \sum_{c_i \in Val(C)} N_{S|c_i} Q_{S|c_i} N'_{S|c_i}$$

Dove $N'_{S|c_i}$ altro non è che che la matrice $N_{S|c_i}$ trasposta.

Per trovare la forma espansa della matrice stiamo essenzialmente espandendo ogni matrice $Q_{S|c_i}$ su S in una matrice su $S \cup C$. Iteriamo su ogni instanziazione dell'ordinamento ϵ_c per poi sommare queste matrici.

Presentiamo ora un Teorema e un corollario che ci mostreranno la potenza di tale rappresentazione.

Theorem 3.3.1. Ogni intensità nella matrice completamente espansa $Q_{S|C}$ corrisponde ad una singola intensità da una $Q_{S|ci}$ per qualche $c_i \in Val(C)$. Inoltre, scegliando una appropriata sottomatrice da $Q_{S|C}$, possiamo ricondurci a $Q_{S|c_i}$ per ogni $c_i \in Val(C)$.

Teorema a cui fa seguito il seguente corolalrio

Corollary .1. Ogni intensità **positiva** della matrice completamente espansa $Q_{S|C}$ corrisponde ad una singola transizione di una singola variabile di S. Le intensità negative di $Q_{S|C}$ sono tutte site nella diagonale principale e rendono la somma di ogni riga pari a zero.

3.4 Amalgamazione di una CIM

Grazie all'espansione delle CIM definita nella sezione precedente possiamo definire formalmente l'operazione di amalgamazione sulle CIM. Definiamo tale operazione tramite la forma espansa per le CIM che abbiamo appena definito.

Definition 3.4.1. L'amalgamazione è una operazone che prende due CIM, $Q_{S_1|C_1}$, $Q_{S_2|C_2}$, per formare la nuova CIM $Q_{S|C}$, dove $S = S_1 \cup S_2$ e $C = (C_1 \cup C_2) \setminus S$. Dato un ordinamento fisso $\epsilon_{S,C}$, espandiamo $Q_{S_1|C_1}$ e $Q_{S_2|C_2}$ in singole matrici sullo spazio S x C dimensionale. Definiamo quindi la matrice amalgamata come la somma $Q_{S|C} = Q_{S_1|C_1} + Q_{S_2|C_2}$.

L'inverso dell'operazione di amalgamazione prende il nome di *matrix substraction*. L'operazione di amalgamazione soddisfa le proprietà *commutativa* e *associativa*. Nel caso della matrice che abbiamo portato ad esempio nei paragrafi e nelle sezioni precedenti in rappresentazione bipartite, l'operazione di amalgamazione di $Q_{H_X|X}$ e $Q_{X|H_X}$

portano alla creazione di una singola matrice $Q_{X\ H_X}$ di dimensioni 28 x 28, numero derivante dal prodotto del numero di stati di X, $k = 4$, e il numero di stati di H_X, N=7. Tale matrice sarà importantissima nella sezione seguente, dove faremo il learning dei parametri della rete. Tale apprendimento viene effettuato dando all'algoritmo di Expectation Maximization in input la matrice amalgamata. La matrice amalgamata trovata a partire dalle CIM dell'esempio 3.1 viene allegata a questo lavoro.

Nel caso invece di rappresentazione diretta, la matrice amalgamata è semplicemente la matrice di partenza illustrata nella sezione 3.1, quindi una matrice di dimensione $12x12$, decisamente più agevole da trattare.

Infine, nel caso di rappresentazione full, la matrice amalgamata avrebbe dimensioni $48x48$, consistendo dunque di 2304 entrate. Il numero di righe e colonne derivano dal prodotto tra numero degli stati di X, 4, con il numero degli stati di H_X, 12. Il numero di stati di H_X è comprendente delle fasi aggiuntive ausiliarie *vanishing*. Il numero di entrate risulta decisamente elevato e cresce molto velocemente, portando alla creazioni di amalgamate intrattabili anche per numero di stati di X e H_X modesti.

Capitolo 4

Learning dei parametri

Come definito nei capitoli precedenti stiamo considerando un approccio che modelli un sistema come evolvente in tempo continuo. Tale approccio è molto potente in quanto elimina i problemi derivanti da approcci che discretizzano il tempo ad intervalli fissi. Tali approcci, come Hidden Markov model o dynamic Bayesian network, sono limitati dalla diversa granularità degli eventi componenti il sistema, con variabili che possono evolvere molto più velocemente di altre.

Uno dei problemi principali in ogni modello è costruirlo adeguatamente. Un approccio per derivare tale modello è tramite dati campionari. Esiste un metodo per derivare il modello a partire da traiettorie - realizzazioni di un certo modello in un dato lasso temporale - *completamente* osservate. Ma in applicazioni reali è raro avere dati completamente osservati. Quindi in tale fase useremo una tecnica per imparare la struttura della rete tramite *traiettorie parzialmente osservate*. Per apprendere i parametri della CTBN useremo l'algoritmo (Dempster et al, 1997) di **Expectation Maximization**, abbreviato in EM. Tale modello permette di imparare una rete nella quale viene superata la limitazione di assenza di memoria dei modelli DBN e CTBN. In particolare, ci permette di imparare i parametri di una rete costruita tramite le *distribuzioni di fase*.

Come mostrato nei precedenti capitoli, la fase preliminare si basa sull'integrare le distribuzioni di fase con le reti Bayesiane a tempo continuo, mentre in questo capitolo tratteremo nello specifico come usare l'algoritmo di EM per imparare i parametri della rete dai dati. Cominceremo mostrando l'algoritmo di EM per un generico processo di Markov, per poi estenderlo ad un processo nel quale sono presenti variabili genitori u. Ricordiamo che tramite l'amalgamazione della matrice riusciamo ad esprimere una CTBN come un processo di Markov omogeneo, pertanto la matrice amalgamata Q_{X,H_X} verrà indicata come generico processo di Markov Q_X.

4.1 Dati incompleti

Per un processo di Markov X, i nostri dati sono un insieme di *traiettorie parzialmente osservate* denotate con $D = \{\sigma[1], .., \sigma[\omega]\}$ che descrivono il comportamento di X. Una traiettoria completa può essere specificata come una sequenza di stati x_i appartenti ad X, ognuno dei quali associato con la relativa durata. Questo significa

che osserviamo *ogni* transizione del sistema da uno stato al successivo e il tempo al quale tale transizione occorre. Al contrario, una traiettoria parzialmente osservata $\sigma \in D$ è specificata come una sequenza di *sottosistemi* S_i di X_i, ognuno con la durata associata. Un sottosistema è semplicemente un sottoinsieme non vuoto di stati di X - quindi un insieme di uno o più stati di X.
Alcune transizioni sono parzialmente osservate: sappiamo solo che il sottosistema passa da un sottosistema all'altro. Tali transizioni fanno cambiare sottosistema nel quale giace la variabile. Transizioni da uno stato all'altro *all'interno* dello stesso sottosistema sono invece completamente non osservate. Ovvero non risultano in alcun modo dalle traiettorie σ parzialmente osservate. Pertanto, non abbiamo alcun modo per stabilire dove e quante siano tali transizioni.

In tale framework è possibile che l'osservazione dello stato del sistema sia momentanea, chiamata anche *point evidence*. In altri termini è possibile che nella sequenza di sottosistemati osservati la durata sia zero, ovvero osserviamo solamente un punto nella linea temporale ad ogni transizione tra sottosistemi. Questo è possibile ricorrendo ad EM per imparare i parametri della rete ma non lo è invece con altre tecniche: rappresenta dunque un importante vantaggio.

Per ogni traiettoria $\sigma[i]$ possiamo considerare uno spazio $\mathbf{H}[i]$ di possibili completamenti della traiettoria. Ogni completamento $h[i] \in \mathbf{H}[i]$ specifica, per ogni transizione in $\sigma[i]$, quale transizione sta avvenendo sottotraccia tra gli stati di X. Specifica inoltre tutte le transizioni tra stati X che sono completamente non osservate in $\sigma[i]$.
In altre parole un completamento $h[i]$ indica un settaggio compatibile con la traiettoria parzialemente osservata $\sigma[i]$. Combinando $\sigma[i]$ con $h[i]$ otteniamo una *traiettoria completamente osservata* $\sigma^+[i]$ su X. Ovvero una traiettoria nella quale sono evidenziate *tutte* le transizioni effettuate da ogni stato x_i ad ogni stato successivo x_j, sia che esso sia all'interno o all'esterno del sottosistema di x_i.
Possiamo notare come, in una traiettoria parzialmente osservata, il numero di possibili transizioni non osservate sottostanti non sia noto. Ci sono un numero non-numerabile di possibili tempi nel quale ogni transizione può avere luogo. Non sappiamo dunque quante transizioni non osserviamo nè quando esse avvengono. Nonostante ciò, la nozione di tutti i possibili completamenti è ben definita.
Possiamo definire l'insieme $D^+ = \{\sigma^+[1], .., \sigma^+[\omega]\}$ come l'insieme dei completamenti di tutte le traiettorie parzialmente osservate in D.

Per chiarire meglio i concetti sovrastanti, illustriamo il seguente esempio.

Esempio 4.1. *Supponiamo di avere un processo Z con $Val(Z) = X\ x\ Y = \{x_1, x_2\}\ x\ \{y_1, y_2\}$ consideriamo gli esempi seguenti di una traiettoria completamente osservata σ^+ su un intervallo $[0, 1)$. Z parte in $< x_1, y_1 >$ al tempo 0; al tempo 0.3 transiziona in $< x_2, y_1 >$, al tempo 0.8 transiziona in $< x_1, y_2 >$.*
Supponiamo ora di avere una traiettoria parzialmente osservata nella quale osserviamo solamente la variabile X, ovvero i due sottosistemi consistono in $< x_1, \cdot >$ - per gli stati $< x_1, y_1 >$ e $< x_1, y_2 >$, e $< x_2, \cdot >$ per gli stati $< x_2, y_1 >$ e $< x_2, y_2 >$. Pertanto la variabili Y risulta "invisibile" nella traiettoria osservata. Una possibile traiettoria parzialemente osservata σ per tali sottosistema sull'intervallo [0,1) è la seguente: Z parte in $< x_1, \cdot >$ al tempo zero, transiziona in $< x_2, \cdot >$ al tempo 0.3. Possiamo notare come σ^+ sia un completamento di σ. Un altro possibile completamento di σ è il seguente: Z parte in $< x_1, y_1 >$ al tempo 0; al tempo 0.2 transiziona

in $< x_1, y_2 >$; al tempo 0.3 transiziona in $< x_2, y_2 >$; al tempo 0.5 transiziona in $< x_2, y_1 >$.
Possiamo descrivere una traiettoria parzialmente osservata σ' con una point evidence a tempi 0.1 e 0.6. Z parte in $< \cdot, \cdot >$ al tempo zero; osserviamo Z in $< x_1, \cdot >$ al tempo 0.1; osserviamo Z in $< x_2, \cdot >$ al tempo 0.6. Da 0.6 in poi osserviamo Z in $< \cdot, \cdot >$. Possiamo notare come σ^+ sia un completamento anche di questa traiettoria definita in termini di point evidence.

4.2 Statistiche sufficienti attese e verosimiglianza

Si definisce statistica sufficiente una statistica che riesce a rappresentare in maniera sintetica una informazione contenuta nel campione. Per un processo di Markov le statistiche sufficienti di un insieme di traiettorie D^+ sono $T[x]$, la quantità totale di tempo nel quale $X = x$, e $M[x, x']$, il numero di volte nelle quali X transiziona da x a x'.
$T[x]$ misura il tempo totale nel quale il sistema si è soffermato nello stato x. Se definiamo $M[x] = \sum_{x'} M[x, x']$, allora possiamo scrivere la log-verosimiglianza per X:

$$\begin{aligned} l_X(q, \theta; D^+) &= l_X(q : D^+) + l_X(\theta : D^+) \\ &= \sum_x (M[x] ln(q_x) + q_x T[x] + \sum_{x \neq x'} M[x, x'] ln(\theta_{xx'})) \end{aligned}$$

Nell'equazione soprastante con q si intende l'intensità diagonale della riga corrispondente allo stato x. Ovvero l'intensità con la quale X lascia lo stato x. $\theta_{xx'}$ è invece la probabilità di transitare nello stato x' *dato* che il sistema lascia lo stato x.

Sia r una densità di probabilità su ogni completamento $\mathbf{H}[i]$ che porta ad avere una densità su ogni possibile completamento dei dati D^+. Possiamo scrivere il valore atteso delle statistiche sufficienti rispetto alle densità di probabilità su ogni possibile completamento dei dati: $\bar{T}[x]$, $\bar{M}[x, x']$, $\bar{M}[x]$. Queste statistiche sufficienti *attese* ci permettono di scrivere il valore atteso della log-verosimiglianza per X come:

$$\begin{aligned} E_r[l_X(q, \theta : D^+) &= E_r[l_X(q : D^+) + E_r[l_X(\theta : D^+) \\ &= \sum_x (\bar{M}[x] ln(q_x) + q_x \bar{T}[x] + \sum_{x \neq x'} \bar{M}[x, x'] ln(\theta_{xx'})) \end{aligned}$$

Tale formula sostituisce alle quantità *teoriche* $M[x], M[x, x'], T[x]$ le quantità *osservate*, campionarie, $\bar{M}[x], \bar{M}[x, x'], \bar{T}[x]$. L'oggetto risultante da tale formula è quindi la log verosimiglianza *attesa* per X.

Tale funzione obbiettivo verrà massimizzata, nei suoi parametri $q; \theta$, in iterazioni successive di EM. Fino a trovarne un punto di massimo dove non si verifichino scostamenti- o essi siano di grandezza irrisoria - della log verosimiglianza attesa in un'iterazione rispetto alla successiva.

4.3 L'algoritmo EM

Per trovare i parametri di massima verosimiglianza q, θ di X usiamo expectation maximization. Tale algoritmo comincia con una assegnazione casuale iniziale q^0, θ^0. Dopodichè ripete l'*Expectation Step* e il *Maximization Step*, aggiornando l'insieme dei parametri, fino ad arrivare a convergenza. Di seguito mostriamo tali step nel dettaglio, supponendo di essere alla k-esima iterazione, con valori di partenza dei parametri dunque q^k e θ^k.

Expectation Step. Usando l'insieme corrente di parametri definiamo per ogni traiettoria $\sigma[i] \in D$ la densità di probabilità:

$$r^k(h[i]) = p(h[i] \mid \sigma[i], q^k, \theta^k)$$

Computiamo quindi le statistiche sufficienti attese $\bar{M}[x], \bar{M}[x, x'], \bar{T}[x]$ secondo tale probabilità a *posteriori* sul completamento delle traiettorie, date le traiettorie e il modello. Semplificando, con l'equazione sopra riportata calcoliamo la probabilità a posteriori di un generico completamento $h[i]$ *dati* i parametri e la traiettoria parziale osservata. Tale probabilità viene poi utilizzata nel calcolo delle statistiche sufficienti attese.

Maximization Step. Usiamo le statistiche sufficienti attese precedentemente calcolate come se provenissero da una traiettoria completamente osservata σ^+. Tramite tali statistiche definiamo i nuovi parametri q^{k+1}, θ^{k+1}. Essi saranno i parametri di massima verosimiglianza del nostro modello e il loro calcolo avviene come segue:

$$q_x^{k+1} = \frac{\bar{M}[x]}{\bar{T}[x]} \qquad \theta_x^{k+1} = \frac{\bar{M}[x, x']}{\bar{M}[x]}$$

Aggiorniamo quindi i parametri di massima verosimiglianza in accordo con le nuove statistiche sufficienti.

Abbiamo quindi mostrato un passo arbitrario del nostro algoritmo. La parte problematica in questo algoritmo è nell'*Expectation Step*. In quanto lo spazio di integrazione è decisamente complesso e non esiste un modo chiaro per calcolare le statistiche sufficienti attese in maniera trattabile. La prossima sezione vuole risolvere tale problema

4.4 Calcolo statistiche sufficienti attese

Per applicare l'algoritmo di EM dobbiamo necessariamente calcolare le statistiche sufficienti attese con rispetto alla densità di probabilità a posteriori sul completamento delle traiettorie date le osservazioni e il modello corrente.

4.4.1 Notazione

Per calcolare le statistiche sufficienti attese su D, le calcoliamo per ogni singola traiettoria $\sigma \in D$ singolarmente e ne combiniamo i risultati.

Una traiettoria parzialmente osservata σ è composta da una sequenza di N sottosistemi tali che lo stato è vincolato al sottosistema S_i durante l'intervallo $[t_i, t_{i+1})$ per $0 \leq i \leq (N-1)$. Supponiamo, senza perdita di generalità, che σ inizi al tempo $t = 0$ e termini al tempo $t = \tau$, avendo pertanto $t_0 = 0$ e $t_N = \tau$.

Per il sottosistema S, sia Q_S la matrice di dimensioni $n\ x\ n$ di intensità con tutte le intensità poste uguali a zero *eccetto* le intensità corrispondenti alle transizioni all'interno. Essa è una matrice di dimensioni pari alla matrice Q_X amalgamata di partenza, con intensità diverse da zero solamente nelle entrate corrispondenti alla matrice *within system* relativa al sottosistema S.

Per i sottosistemi S_1, S_2, sia Q_{S_1,S_2} la matrice di intensità con tutte le intensità poste uguali a zero, eccetto quelle corrispondenti a transizioni dal sottosistema S_1 al sottosistema S_2. Notiamo come anche le intensità corrispondenti a transizioni all'interno di S_1 o di S_2 vengano poste uguali a zero. Le uniche entrate diverse da zero sono quelle in corrispondenza della matrice *between system* relativa ai sottosistemi S_1, S_2. Tali entrate sono uguali ai valori delle transizioni tra i sottosistemi S_1, S_2 nella matrice Q_X.

Ci sarà utile nel proseguio riferirci alle transizioni osservate delle traiettorie (chiamate anche *prove*) fornite da un intervallo di tempo arbitrario. Definiamo dunque $\sigma_{t_1:t2}$ come la prova fornita da σ sull'intervallo $[t_1, t_2)$. Quindi $\sigma_{0:\tau}$ è la prova di cui disponiamo in *tutta* la traiettoria σ. Sia poi $\sigma_{t_1:t_2^+}$ la prova fornita sull'intervallo *chiuso* $[t_1, t_2]$. Specularmente, sia invece $\sigma_{t_1^-:t_2}$ la prova derivante dall'intervallo *aperto* (t_1, t_2). Sia e un vettore colonna di dimensione $[n\ x\ 1]$ di uno. Sia poi e_j un vettore colonna di dimensione $[n\ x\ 1]$ con zero ovunque e uno nella j-esima posizione. Sia $\Delta_{j,k}$ una matrice di dimensioni $n\ x\ n$ di zeri con uno in posizione $[j, k]$. Pertanto per ottenere tale matrice è sufficiente combinare opportunamente due vettori colonna e_i: $\Delta_{j,k} = e_j e_k'$. Definiamo dunque ora i vettori α_t^- e β_t^+ element-wise nella seguente maniera:

$$\alpha_t^-[i] = p(X_t^- = i, \sigma_{0:t}) \qquad \beta_t^+[i] = P(\sigma_{t^+:\tau} | X_t^+ = i)$$

Dove X_t^- è il valore assunto da X appena prima di una eventuale transizione in t. Allo stesso modo X_t^+ è il valore assunto da X appena dopo l'eventuale transizione in t. Tali valori hanno interpretazione analoga a limite destro e limite sinistro per una funzione. Essi infatti coincidono se non è presente una transizione in t. E' importante notare come dalle notazioni soprastanti abbiamo che $\sigma_{0:t}$ rappresenta l'informazione dell'intervallo $[0, t)$, con t non inclusa, mentre $\sigma_{t^+:\tau}$ rappresenta la prova fornita dall'intervallo (t, τ), con τ non inclusa. Pertanto, nessuno dei vettori α^-, β^+ definiti includono prova di una transizione al tempo t.

Definiamo inoltre i vettori:

$$\alpha_t[i] = p(X_t = i, \sigma_{0:t^+}) \qquad \beta_t[i] = P(\sigma_{t:\tau} | X_t = i)$$

I quali al contrario includono entrambi l'informazione relativa all'eventuale transizione avvenuta al tempo t. Tali vettori costituiscono i passi *backward* e *forward* del nostro algoritmo e ci serviranno per calcolare le statistiche sufficienti attese $\bar{M}[x], \bar{M}[x, x'], \bar{T}[x]$.

Nelle prossime sezioni vedremo come calcolare operativamente- tramite le quantità appena definite - le statistiche sufficienti

4.4.2 Durata attesa

Per brevità in tale sezione vengono riportate solamente le formule finali. Per i passaggi su come ricondursi a tali formule la referenza è [Expectation Maximization and Complex Duration Distributions for Continuous Time Bayesian Network, Nodelamn and al., 2005].

La statistica sufficiente $T[j]$ è il quantitativo di tempo che X spende nello stato j nel corso della traiettoria σ. Possiamo scrivere il valore atteso di $T[j]$ con la seguente formula:

$$E[T[j]] = \frac{1}{p(\sigma_{0:\tau})} \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} p(X_t, \sigma_{0:\tau}) e_j dt$$

La frazione costante all'inizio dell'equazione è semplicemente un moltiplicatore che serve per rendere la somma del totale del tempo atteso su ogni j pari a τ. Il simbolismo con $p(\sigma_{0:\tau})$ è pertanto fuorviante, è semplicemnte una riscalatura dell'integrale precedente che viene effettuata a posteriori dopo aver calcolato $E[T[j]] \; \forall \; j \in val(X)$. Rende la somma di tali $E[T[j]]$ pari alla lunghezza della traiettoria.

Per calcolare tale integrale dunque abbiamo spezzato in N intervalli, ognuno con *informazione costante*, ovvero ognuno di tali intervalli non ha transizioni osservate al suo interno.

Mostriamo ora il calcolo del generico elemento di tale sommatoria. Sia $[v, w)$ un intervallo di informazione costante, e sia S il sottosistema nel quale lo stato è vincolato in tale intervallo. Abbiamo dunque

$$\int_v^w p(X_t, \sigma_{0:\tau}) e_j dt = \int_v^w \alpha_v exp(Q_S(t-v)) \Delta_{j,j} exp(Q_S(w-t)) \beta_w dt$$

Per calcolare tale integrale, e in generale ogni integrale necessario per la fase di learning usiamo il metodo di Simpson, detto anche *Simpson's Rule*. Si tratta di un metodo per il calcolo approssimato di integrali più raffinato rispetto al semplice campionamento dalla funzione con l'idea dell'integrale di Riemann.

In sistesi dunque possiamo calcolare il tempo totale atteso $E[T[j]]$ sommando l'espessione precedente su tutti gli intervalli con informazione costante.

4.4.3 Numero atteso di transizioni

Anche in tale sezione vengono riportate solamente le formule finali. Per i passaggi su come ricondursi a tali formule la referenza è [Expectation Maximization and Complex Duration Distributions for Continuous Time Bayesian Network, Nodelamn and al.,

2005].

La statistica sufficiente $M[j,k]$ è il numero di volte X transiziona dallo stato j allo stato k nel corso della traiettoria σ. Per calcolare il suo valore atteso ricorriamo ad una approssimazione; discretizziamo la traiettoria in intervalli di lunghezza ϵ e incrementiamo $M_e[j,k]$ per ogni intervallo ϵ nel quale il sistema entra in j ed esce in k. In formula:

$$M_\epsilon[j,k] = \sum_{t=0}^{\tau/\epsilon-1} \mathbf{1}\{X_{t\epsilon} = j, X_{(t+1)\epsilon} = k\}$$

Si può dimostrare che per $\epsilon \to 0$ talle somma coincide con la quantità reale ricercata:

$$E[M[j,k]] = \lim_{\epsilon \to 0} E[M_\epsilon[j,k]]$$

Il valore atteso di destra può essere riscritto come:

$$E[M_\epsilon[j,k]] = \sum_{t=0}^{\tau/\epsilon-1} \frac{p(X_{t\epsilon}, X_{(t+1)\epsilon} = k, \sigma_{0:\tau})}{p(\sigma_{0:\tau})}$$

Possiamo notare come anche in tale caso il simbolismo $p(\sigma_{0:\tau})$ sia fuorviante, esso si riferisce semplicemente allo stesso moltiplicatore calcolato tramite $E[T]$ per avere una somma dei tempi pari alla lunghezza della traiettoria. Possiamo inoltre osservare come, per ϵ sufficientemente piccolo, possiamo osservare *al più* una transizione per intervallo. Pertanto, tutti gli intervalli della sommatoria appartengono ad una delle seguenti due categorie: contengono una transizione parzialmente osservata, oppure l'intervallo ha informazione costante. Trattiamo tali casi separatamente

Sia $[t\epsilon, (t+1)\epsilon]$ un intervallo contenente una transizione parzialmente osservata al tempo t_i. Noi osserviamo solamente che stiamo transizionando da uno degli stati di S_i ad uno degli stati di S_{i+1}. Possiamo calcolare il contributo di questo intervallo alla statistica sufficiente attesa, ignorando la costante moltiplicativa $p(\sigma_{0:\tau})$, come:

$$q_{jk} \alpha^-_{t_i} \Delta_{j,k} \beta^+_{t_i}$$

$q_{j,k}$ rappresenta l'intensità di transizione allo stato corrente della matrice. $\alpha^-_{t_i}$ è invece il vettore di probabilità di osservare $X_{t^-} = i$ *e* $\sigma_{0:t^+}$, ovvero la probabilità di essere in tale stato *prima della transizione in* t_i avendo osservato la traiettoria fino a tale t_i. $\beta^+_{t_i}$ è invece la probabilità *da quel momento* t_i *in avanti* di osservare la traiettoria $\sigma_{t:\tau}$ dato di aver osservato $X_{t^+} = i$, X dopo la transizione. La matrice Δ serve per selezionare dai due vettori solamente il j-esimo elemento del primo e il k-esimo elemento del secondo.

Consideriamo ora il caso opposto, ovvero un intervallo $[v,w) = [t_i, t_{i+1})$ di informazione costante - senza alcuna transizione parzialmente osservata. Tale intervallo sarà generalmente di lunghezza maggiore rispetto al precedente. Sia $\Delta t = w - v$ e sia S il sottosistema nel quale tale stesso è vincolato nell'intervallo. Per $\epsilon \to 0$, il contributo di tale intervallo alla somma è esprimibile come:

$$q_{j,k} \int_v^w \alpha_v exp(Q_S(t-v)) \Delta_{j,k} exp(Q_S(w-t) \beta_w dt$$

Tale formula, come la formula precedente, sono solamente il risultato finale di una serie di calcoli che non riporniamo per brevità, la cui referenza è [Expectation Maximization and Complex Duration Distributions for Continuous Time Bayesian Network, Nodelamn and al., 2005].

Abbiamo una espressione del genere per ogni intervallo di informazione costante. Stiamo essenzialmente integrando la probabilità *instantanea* di transizionare dallo stato j allo stato k nell'intervallo data l'informazione. Esiste una somiglianza di base tra questa formula e quella usata per calcolare $E[T[j]]$: quest'ultima differisce solamente per la matrice $\Delta_{j,k}$ e il termine $q_{j,k}$.
Per ottenere la statistica sufficiente complessiva, dobbiamo dunque combinare le somme dei termini nelle formule per informazine costante e per intervalli con transizioni parzialmente osservate. L'equazione complessiva possiamo scriverla come:

$$\frac{q_{j,k}}{p(\sigma_{0:\tau})}\left[\sum_{i=1}^{N-1}\alpha^{-}_{t_i}\Delta_{j,k}\beta^{+}_{t_i}+\sum_{i=1}^{N-1}\int_{v}^{w}\alpha_v exp(Q_S(t-v))\Delta_{j,k}exp(Q_S(w-t)\beta_w dt\right]$$

Le costanti moltiplicative, per $E[M[j,k]]$ sono dunque $q_{j,k}$ e $p(\sigma_{0:\tau})$. La prima è specifica per ogni coppia di valori j e k, mentre la seconda è la stessa costante moltiplicativa già incontrata nel calcolo della durata attesa in ogni stato. La prima sommatoria rappresenta il contributo a $E[M[j,k]]$ delle transizioni parzialmente osservate su tutta la traiettoria σ, mentre la seconda sommatoria rappresenta il contributo degli intervalli di informazione costante.

Pertanto per $j,k \in S_i$, appartenenti dunque allo stesso sottosistema, la *prima* sommatoria da un contributo nullo, in quanto se viene osservata una transizione in un tempo t_i, allora è impossibile che tale transizione sottenda una transizione tra due stati dello stesso sottosistema.
Analogamente per j,k appartenenti a sottosistemi diversi, la seconda sommatoria porterà un contributo nullo, in quanto è impossibile che avvenga una transizione tra due stati che appartengono a sottosistemi diversi in intervalli di informazione costante, ovvero senza transizione parzialmente osservate.

4.4.4 Calcolo α_t e β_t

Gli ultimi tasselli che dobbiamo calcolare per far funzionare l'algoritmo di Expectation Maximization sono α_t e β_t. Calcoliamo tali quantità in maniera iterativa con punti di partenza diversi. Per α_t la partenza è α_0, una inizializzazione sugli stati consistente con la traiettoria. Ad esempio si da lo stesso valore a tutti gli stati appartenenti al sottosistema di partenza della traiettoria. Tale valore viene utilizzato come punto di partenza per calcolare α_1, che a sua volta viene utilizzato per il calcolo di α_2, e così procedendo fino all'elemento α_τ. Per il calcolo dei beta viene invece utilizzato il procedimento inverso. Il punto di partenza è β_τ che viene inizializzato con 1 in ogni stato, $\beta_\tau = e$ un vettore di uno. A partire da tale vettore vengono calcolati iterativamente $\beta_{\tau-1}$, $\beta_{\tau-2}$, ... fino a β_0. Tale metodo per calcolare α_t e β_t si basa su un algoritmo *forward-backward*.
Operativamente le formule per calcolare α_{t+1} e β_t a partire dai loro predecessori α_t e

β_{t+1} sono:

$$\alpha_{t_{i+1}} = \alpha_{t_i} exp(Q_{s_i}(t_{i+1} - t_i))Q_{S_i,S_{i+1}}$$
$$\beta_{t_i} = Q_{S_{i-1},S_i} exp(Q_{s_i}(t_{i+1} - t_i))\beta_{t_{i+1}}$$

Per escludere l'incorporazione dell'informazione sulla transizione da entrambi i vettori forward e backward (oppure se il tempo t_i in questione non è un tempo di transizione), possiamo semplicemente rimuovere dalle equazioni sovrastanti la matrice di intensità di transizione tra sottosistemi $Q_{S_i,S_{i+1}}$. Per esempio dato che i tempi estremi, ovvero la partenza $t = 0$ e il punto di arrivo $t = \tau$ non sono tempi di transizione , abbiamo:

$$\alpha_{t_\tau} = \alpha_{t_{N-1}} exp(Q_{S_{N-1}}(\tau - t_{N-1}))$$
$$\beta_0 = exp(Q_{s_0}(t_i - 0))\beta_{t_1}$$

Calcoliamo anche le versioni iterative dei vettori α_t^- e β_t^+ con tali formule:

$$\alpha^-_{t_{i+1}} = \alpha_{t_i} exp(Q_{s_i}(t_{i+1} - t_i))$$
$$\beta^+_{t_i} = exp(Q_{s_i}(t_{i+1} - t_i))\beta_{t_{i+1}}$$

Tali formule differiscono dal calcolo per α_t e β_t solamente per l'esclusione della matrice $Q_{S_i,S_{i+1}}$, data dalla transizione osservata in t_i. Notiamo come per calcolare il generico elemento α^-_{t+1} non partiamo da α_t^- ma bensì da α_t; analogamente con β_t^+. Questi due vettori saranno necessari per calcolare il numero atteso di transizioni $E[M[j,k]]$.

4.5 EM con variabili genitori

Ora siamo in possesso di tutti gli strumenti per effettuare una Expectation Maximization su un processo di Markov. Avendo noi amalgamato la matrice, trasformandola in un processo di Markov, possiamo applicarle tale algoritmo.
Abbiamo inoltre dimostrato come la rappresentazione diretta e quella bipartitica rappresentano esattamente lo stesso processo. Pertanto, data la semplicità computazionale, nei vari esempi sviluppati abbiamo imparato una matrice in rappresentazione diretta.
Da tale matrice esiste una conversione biunivoca (uno a uno) ad un insieme di matrici rappresentanti lo stesso processo in forma bipartite. Il codice allegato permette di imparare direttamente una matrice in forma bipartite, tramite la sua amalgamata. Ii parametri da imparare sono i medesimi nelle due rappresentazioni.

Tale algoritmo non ci soddisfa appieno in quanto riesce ad imparare una matrice di Markov, ma non una rete completa.
Per imparare una rete è sufficiente essere in grado di stabilire, tramite confronto tra

log-verosimiglianze, se un certo nodo è o meno genitore di un altro nodo. Tale processo, se ripetuto per tutta la rete, riesce a ricostruire la rete di parentela completa, costituendo il mattoncino chiave nell'apprendimento della rete. E' necessario dunque essere in grado di imparare i parametri q, θ per la matrice amalgamata di intensità Q_X supponendo tale nodo X senza genitori e di imparare i parametri $q_{X|Y}, \theta_{X|Y}$ per le matrici amalgamate $Q_{X|Y}$ supponendo che Y sia un genitore di X. Dobbiamo inoltre essere in grado di *confrontare* questi due casi e stabilire, tramite log-verosimiglianza, quale dei due casi è maggiormente supportato dall'evidenza empirica.
Ci sono varie tecniche che consentono di imparare i parametri caratterizzanti le matrici $Q_{X|Y}$, supponendo Y genitore di X. La tecnica che abbiamo utilizzato si basa su una modifica dell'algoritmo realizzato per l'apprendimento di una rete senza genitori.

La traiettoria parzialmente osservata che abbiamo a disposizione in questo caso differisce rispetto al caso precedente in quanto è presente anche l'indicazione riguardante quale valore assume la variabile genitore Y in ogni momento. Di seguito vediamo un esempio di un estratto di una traiettoria di un sistema nella quale la variabile Y è genitore di X.

t_{inizio}	t_{fine}	$stato_y$	$stato_x$
0.0000000	1.1325524	1	2
1.1325524	10.0737447	1	2
10.0737447	11.2014548	2	1
11.2014548	13.0038702	1	1
13.0038702	13.4006153	1	2

In tale esempio, con sia X che Y binarie, notiamo come osserviamo sia ogni transizione per la variabile X che ogni transizione della variabile Y. Quello che non osserviamo, analogamente al caso senza senza genitori, sono le transizioni tra le variabili Hidden H_X.

Per sfruttare l'informazione di questa traiettoria è necessario estrapolarne le statistiche sufficienti. Per costruirle dobbiamo partire dal definire $\alpha_t[i]$ e $\beta_t[i]$ in tale impostazione.

Le defizioni continuano ad essere iterative ma si modificano nel modo che segue: $\alpha_t[i]$ viene calcolata in ogni istante temporale nel quale avviene una transizione nella variabile X *o* nella variabile Y. Nel primo caso la definizione è quella canonica e non si discosta dalla definizione fornita nella struttura di rete senza genitori, mentre nel secondo caso dalla formula viene omesso il termine $Q_{S_i,S_{i+1}}$.

Riassumendo in formula :

$$\alpha_{t_{i+1}} = \begin{cases} \alpha_{t_i} exp(Q_{s_i}(t_{i+1} - t_i)) Q_{S_i,S_{i+1}} & \text{se } \mathrm{x}_t \neq x_{t+1} \\ \alpha_{t_i} exp(Q_{s_i}(t_{i+1} - t_i)) & \text{se } \mathrm{x}_t = x_{t+1} \end{cases}$$

Analogamente viene calcolato $\beta_t[i]$:

$$\beta_{t_i} = \begin{cases} \mathrm{Q}_{S_{i-1},S_i} exp(Q_{s_i}(t_{i+1} - t_i))\beta_{t_{i+1}} & \text{se } \mathrm{x}_t \neq x_{t+1} \\ \exp(\mathrm{Q}_{s_i}(t_{i+1} - t_i))\beta_{t_{i+1}} & \text{se } \mathrm{x}_t = x_{t+1} \end{cases}$$

Tali formule derivano dalla definizione di $\alpha_t[i]$ e $\beta_t[i]$. I vettori sono in funzione di X_t solamente e non entra in esame la variabile genitore Y_t.
I vettori α_t^- e β_t^+ vengono calcolati per via iterativa con la medesima formula applicata nell'expectation maximization senza genitori, senza alcuna modifica.

Tali vettori ci serviranno per definire le statistiche sufficienti. Notiamo che le statistiche sono della struttura $T[x|u]$, il tempo totale in cui $X = x$, mentre $U = u$, mentre $M[x, x'|u]$ è il numero totale di volte che X transiziona da x a x' sotto $U = u$. Ricordiamo che $U = u$ significa per una particolare istanza dell'insieme di variabili genitori di X. Nel caso riportato nella tabella soprastante le istanze U sono solamente due: y_1 e y_2.

Per calcolare le statistiche sufficienti attese $\bar{M}[x, x'|u]$ e $\bar{T}[x|u]$ vengono utilizzati i vettori sopracalcolati nel modo seguente: per ogni periodo temporale $[t_i, t_{i+1}]$ il sistema è in un determinato stato dell'insieme di variabili genitori $U = u$, ovvero nel nostro esempio $Y = y_1$ *oppure* $Y = y_2$. Pertanto la sommatoria complessiva che viene utilizzata per il calcolo delle statistiche sufficienti viene divisa in due sommatorie, rispettivamente per intervalli consistenti con y_1 e per intervalli consistenti con y_2. In generale la sommatoria su tutta la traiettoria viene divisa in sommatorie sugli intervalli della traiettoria consistenti con una istanza di U.
Traducendo in formula, la statistica sufficiente $T[j \mid U = u]$ è il quantitativo di tempo che X spende nello stato j nel corso della traiettoria σ condizionato ai suoi genitori U. Possiamo scrivere il valore atteso di $T[j \mid U = u]$ con la seguente formula:

$$E[\, T[j \mid U = u]] = \frac{1}{p(\sigma_{0:\tau})} \sum_{i=0}^{N-1} \mathbb{1}(U_i = u) \int_{t_i}^{t_{i+1}} p(X_t, \sigma_{0:\tau}) e_j dt$$

Dove $\mathbb{1}(U_i = u)$ è la funzione indicatrice che vale 1 se l'instanziazione dell'insieme delle variabili genitori U è u nell'i-esimo intervallo della traiettoria σ.

Per ricondurci a $E[\, T[j]\,]$ è sufficiente sommare $E[\, T[j \mid U = u]]\ \forall u \in Val(U)$. Il resto della formula è identico alla formula per calcolare $E[\, T[j]\,]$ nel caso senza genitori.

Analogamente viene calcolata anche $E[M_\epsilon[j, k] \mid U = u]]$:

$$E[M_\epsilon[j, k] \mid U = u]] = \sum_{t=0}^{\tau/\epsilon - 1} \mathbb{1}(U_t = u) \frac{p(X_{t\epsilon} = j, X_{(t+1)\epsilon} = k, \sigma_{0:\tau})}{p(\sigma_{0:\tau})}$$

Dove l'interpratazione di $\mathbb{1}(U_t = u)$ è la medesima che nel calcolo della durata media in uno stato precedente.

Abbiamo mostrato nei paragrafi soprastanti come calcolare le statistiche sufficienti condizionate $\bar{M}[x, x'\,|u], \bar{M}[x\,|u], \bar{T}[x\,|u]$, ovvero il passo di *Expectation* dell'algoritmo

EM. Possiamo dunque ora sfruttare le statistiche suffcienti attese condizionate appena calcolate, come se venissero da un dataset completo, per aggiornare i valori di q e θ condizionati. Nello specifico tali parametri sono:

$$q_{x|u}^{k+1} = \frac{\bar{M}[x\ |u]}{\bar{T}[x\ |u]} \qquad \theta_{x|u}^{k+1} = \frac{\bar{M}[x, x'\ |u]}{\bar{M}[x\ |u]}$$

Esiste dunque una coerenza intrinseca tra il passo di Maximization in tale struttura e nel caso senza genitori.

Essendo cambiata la definizione dei parametri componenti l'Expectation Maximization cambia conseguentemente anche la log-verosimiglianza da massimizzare. Tale verosimiglianza si può decomporre nel modo seguente:

$$\begin{aligned} l_X(q, \theta; D^+) &= l_X(q : D^+) + l_X(\theta : D^+) \\ &= \sum_u \sum_x (M[x|u] ln(q_{x|u}) + q_{x|u} T[x|u]) \\ &+ \sum_u \sum_x \sum_{x \neq x'} M[x, x'|u] ln(\theta_{xx'|u})) \end{aligned}$$

Grazie alla proprietà di linearità del valore atteso, la funzione di log-verosimiglianza attesa si decompone nella medesima maniera, e possiamo dunque scrivere $E_r[l_X(q, \theta : D^+)]$ come una somma negli stessi termini dell'equazione soprastante, ma sostituendo alle statistiche sufficienti la loro versione *attesa* $\bar{M}[x]$, $\bar{T}[x]$ e $\bar{M}[x, x']$:

$$\begin{aligned} E_r[l_X(q, \theta : D^+)] &= E_r[l_X(q : D^+) + E_r[l_X(\theta : D^+) \\ &= \sum_u \sum_x (\bar{M}[x|u] ln(q_{x|u}) + q_{x|u} \bar{T}[x|u]) \\ &+ \sum_u \sum_x \sum_{x \neq x'} \bar{M}[x, x'|u] ln(\theta_{xx'|u})) \end{aligned}$$

Tale log-verosimiglianza attesa rappresenta la funzione obbiettivo che viene massimizzata in iterazioni successive dell'algoritmo di EM, fino a raggiungere convergenza. Dobbiamo notare che però tale punto di massimo raggiunto è solamente un *massimo locale* e non esiste alcuna assicurazione che non esistano dei valori per l'insieme di parametri che permetterebbero di raggiungere valori migliori di log-verosimiglianza.

Nello studio di reti Bayesiane è importante riuscire ad avere uno strumento che ci dica se una certa variabile Y è genitore di una variabile X. Ovvero che ci riesca a dire se la configurazione di X con genitore Y è maggiormente supportata dall'evidenza empirica - le traiettorie osservate - rispetto alla configurazione opposta. Tale confronto avviene tramite la log-verosimiglianza. La tecnica per operare tale raffronto è la seguente:

- Costruiamo sia il modello con Y che è genitore di X sia il caso opposto
- Impariamo i parametri di entrambi i modelli mediante le traiettorie a nostra disposizione con l'algoritmo di EM, fino a raggiungere convergenza

- Compararíamo le log-verosimiglianza attese ottenute nelle due configurazioni.

Risulta maggiormente supportata la configurazione che ha ottenuto la verosimiglianza maggiore.

Capitolo 5

L'algoritmo EM nell'apprendimento di una CTBN con distribuzione di fase

La fase di apprendimento dei parametri di una rete CTBN con distribuzione di fase è stata molto dispendiosa in termini di tempo. E' presente sul software statistico R un pacchetto che si occupa di apprendere reti Bayesiane tramite traiettorie parzialmente osservate ma non era adatto allo scopo perchè soffre di un importante bug. Una rete Bayesiana si caratterizza per le entrate *pari a zero* della matrice amalgamata. Ovvero una rete è definita per *quello che non può fare*. Essendo il passaggio di aggiornamento dei parametri di EM, in sommi termini e per una generica iterazione, un semplice prodotto tra il valore dell'iterazione precedente e il valore risultante dall'evidenza empirica allora inizializzare delle entrate pari a zero nella matrice di partenza deve risultare in tali entrate pari a zero anche nella matrice risultante. Questa caratteristica, fondamentale per la definizione di una rete Bayesiana a tempo continuo, non era soddisfatta provando ad apprendere tale rete con il pacchetto di funzioni pre esistente.

Pertanto, abbiamo dovuto partire da zero e scrivere un insieme di funzioni che ci potesse portare all'obbiettivo: imparare una rete Bayesiana tramite traiettorie parzialmente osservate.

5.1 Generazione traiettorie

Il primo passo è stato generare tali traiettorie parzialmente osservate, che serviranno successivamente come dati di training per imparare la matrice. L'obbiettivo è quello di creare una o più traiettorie che siano realizzazioni casuali dei cambiamenti di stato di un sistema descritto da una matrice Q_{XH_X} in una arco di tempo pari a $t = \tau$. Per generare una traiettoria il procedimento che abbiamo adottato è basato sulla definizione di rete Bayesiana. Abbiamo inizializzato lo stato di partenza in maniera randomica tra tutti gli stati della matrice Q_{XH_X}. Dopodichè sappiamo che la transizione è regolata da una coppia di variabili casuali: una esponenziale per *quando* avviene la transizione e una multinomiale per in quale stato risulta la transizione.

Poniamo particolare attenzione che benchè siamo in un sistema di distribuzione di fase dove la durata in uno stato x_i *non* è distribuita come una esponenziale, il tempo di permanenza nella singola *fase* dello stato x_i, ovvero l'entrata x_i, h_j della matrice Q_{XH_X}, è regolata da una variabile esponenziale. I parametri della multinomiale che definiscono la probabilità di transizionare in un certo stato sono esattamente i parametri $q_{x_i h_j, x_i h_n}$ della matrice Q_{XH_X}. Generando quindi delle realizzazioni casuali da una esponenziale t_0 e da una multinomiale $stato_{t=0}$ siamo in grado di descrivere completamente la transizione. Al passo successivo il sistema partità dal tempo t_1 e sarà nello stato $stato_{t=1}$. Iterando tale procedimento fino a τ avremo generato la traiettoria di nostro interesse.

Di seguito vediamo un estratto di un esempio di una traiettoria generata.

Realizzazione traiettoria con k=3 ; N=4			
stato Q_{XH_X}	inizio	fine	stato x
4	0.00000000	0.05529582	1
3	0.05529582	0.30888997	1
4	0.30888997	0.37593480	1
5	0.37593480	0.43795402	2
6	0.43795402	0.60686499	2

La colonna "stato Q_{XH_X}" contiene l'indicazione sullo stato $Val(X) \cup Val(H)$ della matrice Q_{XH_X} espresso in forma di un numero incrementale.
La colonna "inizio" conteine l'indicazione del tempo t di quando il sistema entra nello stato corrisponendente. La colonna fine, quando *esce* da tale stato. La colonna "stato x" invece è una traslazione dello stato Q_{XH_X} dell'amalgamata allo stato x. Nel nostro esempio le prime quattro fasi di Q_{XH_X} fanno riferimento allo stato $x = 1$, mentre le fase $5 - 8$ allo stato $x = 2$. Tale colonna verrà utilizzata per la fase di learning

E' importante notare come la colonna riguardante lo stato di h sia importante e presente in fase di generazione ma debba venire oscurata in fase di apprendimento della rete. In quanto in applicazioni nel mondo reale - con traiettorie empiriche e non generate - non conosciamo tipicamente gli stati nascosti di una certa variabile ma *osserviamo* solamente data variabile.

Un estratto di una traiettoria che utilizzeremo per la fase di learning è dunque la seguente:

Traiettoria per learning k=3 ; N=4		
inizio	fine	stato x
0.0000000	0.3759348	1
0.3759348	0.9103758	2
0.9103758	1.4866120	1
1.4866120	2.6297739	2
2.6297739	4.0868043	1

Come possiamo osservare, in questo data frame sono contenuti solo le indicazioni degli estremi temporali e dello stato di X. In più, ogni riga racchiude tutto il tempo trascorso nello stato x_i, che è dunque la somma del tempo trascorso nell'insieme delle fasi che compongono x_i.

Dopo avere definito come è possibile generare delle traiettorie, passiamo alla definizioni dei vari passi che compongono l'algoritmo di Expectation Maximization.

5.2 Expectation Maximization

Alla base di tutto il procedimento vi sono delle funzioni che data una certa matrice la trasformano nella matrice *within* o *beetwen* di un certo sottosistema. Ovvero la prima di tali funzioni, data una generica matrice 12 x 12 con 4 stati x ognuno con 3 fasi e dato in input uno stato x_i , calcolano una matrice sempre 12 x 12 ma con le entrate a zero ovunque tranne che nelle righe e colonne consistenti con lo stato x_i. Tale matrice si chiama matrice *within subsystem* e la indicheremo con Q_{s_i}

La seconda funzione calcola invece la matrice *between subsystem* consistente con un certo stato x_i di partenza e uno stato x_j di arrivo. Il risultato è dunque una matrice con tutte le entrate pari a zero tranne nella sottomatrice che regola la transizione tra x_i ed x_j, le cui righe sono consistenti con x_i e le colonne sono consistenti con x_j.

5.2.1 Calcolo α e β

Il passaggio successivo è il calcolo dei vettori α_t, α_t^-, β_t e β_t^+. Tale calcolo avviene in modo ricorsivo, come illustrato nella sezione esplicativa dell'algoritmo EM. Il singolo passo viene calcolato tramite la specifica formula mediante l'esponenzializzazione della matrice Q_{s_i}.

$$
\begin{aligned}
\alpha_{t_{i+1}} &= \alpha_{t_i} exp(Q_{s_i}(t_{i+1} - t_i))Q_{S_i,S_{i+1}} \\
\beta_{t_i} &= Q_{S_{i-1},S_i} exp(Q_{s_i}(t_{i+1} - t_i))\beta_{t_{i+1}} \\
\alpha^-_{t_{i+1}} &= \alpha_{t_i} exp(Q_{s_i}(t_{i+1} - t_i)) \\
\beta^+_{t_i} &= exp(Q_{s_i}(t_{i+1} - t_i))\beta_{t_{i+1}}
\end{aligned}
$$

Per quanto riguarda α_t e α_t^- il risultato finale è una matrice il cui singolo elemento t-esimo è un vettore di dimensioni $k * N$. Con k numero di stati di X e N numero di fasi di ogni variabile $x \in X$. L'elemento i-esimo di tale vettore, ovvero $\alpha_t[i]$ ed $\alpha_t^-[i]$ sono le probabilità di essere nello stato i-esimo al tempo t considerata la traiettoria osservata fino a t $\sigma_{0:t}$.
In $\alpha_t[i]$ è anche considerata la transizione al tempo t. Per calcolare tali matrici viene inizializzato α_0 con una probabilità costante nelle fasi consistenti allo stato x_0 di partenza e 0 altrove. Dopodichè iterativamente viene chiamata la funzione che calcola

$\alpha_{t_{i+1}}$ dato l'elemento precedente α_t.

Di seguito un estratto di α:

Esempio di α_t con k=2; N=2				
t_0	t_1	t_2	t_3	t_4
0	0.340318	0.0000000	0.2103252	0.0000000
0	0.5344029	0.0000000	0.1242667	0.0000000
0.5	0.0000000	0.273024	0.0000000	0.0956367
0.5	0.0000000	0.2240789	0.0000000	0.0873366

Alcune osservazioni importanti su questa tabella.

Innanzitutto possiamo notare come questa α sia stata calcolata in corrispondenza di una traiettoria il cui stato iniziale è x_2; infatti il vettore α_{t_0} è stato inizializzato con zero nelle prime due righe -consistenti con x_1, e con un valore costante e sommante a 1 nelle due righe seguenti - corrisponenti alla coppia di fasi che compone x_2. Possiamo osservare inoltre che in ogni matrice α_t, ed anche α_t^-, il valore tenda a degradarsi per istanti temporali successivi.

Gli istanti temporali di riferimento vanno da t_0, che considera le probabilità iniziali, fino a $t_{\tau-1}$, ovvero fino alla penultima transizione. E' possibile computare anche t_τ ma è inutile nel successivo calcolo delle statistiche sufficienti attese.

In ogni istante temporale, $t_0, t_1, .., t_{N-1}$, solamente le fasi consistenti ad uno stato hanno valore positivo. Nel nostro caso, solamente le prime due righe - stato x_1 - oppure le ultime due righe hanno valore positivo. Le probabilità dunque degradano nel tempo ma sempre con valori positivi compresi tra 0 ed 1.

Il degrado progressivo delle probabilità, dovuto a considerare porzioni di traiettoria via via più lunghe, porta ad un limite tecnico della lunghezza massima della traiettoria con la quale imparare tale α. Tale limite non è fisso, ma è negativamente influenzato dal numero di fasi e dal numero di stati. In altre parole, al crescere della complessità della rete tale degrado avviene più rapidamente pertanto il limite tecnico si raggiunge prima.

Per ovviare a tale problema la soluzione che abbiamo trovato è stata *dividere* una traiettoria che supera tale limite tecnico in due traiettorie di dimensione ridotta, e considerarle due traiettorie distinte. Andrebbero però effettuate ricerche successive se il dividere due traiettorie porta ad avere degli effetti collaterali che degradano l'informazione apportata da tale traiettoria.

Per quanto riguarda β_t e β_t^+ Per avere in conclusione la matrice il cui singolo elemento è un vettore di dimensioni $k * N$, con k numero di stati di X e N numero di fasi di ogni variabile $x \in X$. L'elemento i-esimo di tale vettore, ovvero $\beta_t[i]$ ed $\beta_t^+[i]$ sono le probabilità che la probabilità di osservare la porzione di traiettoria da t fino ad τ, dato che $X_t = i$. In $\beta_t[i]$ è anche considerata la transizione al tempo t. Per calcolare tale

matrici viene inizializzato β_τ con una probabilità costante pari ad 1. Dopodichè iterativamente viene chiamata la funzione che calcola β_{t_i} dato l'elemento successivo $\beta_{t_{i+1}}$.

Vediamo di seguito un estratto da β:

Esempio di β_t con k=2; N=2								
t_0	t_1	t_2	t_3	..	t_{N-3}	t_{N-2}	t_{N-1}	t_N
0.000	$5.24 * 10^{-16}$	0.000	$6.38 * 10^{-16}$	..	0.000	0.696	0.000	1
0.000	$4.66 * 10^{-16}$	0.000	$9.05 * 10^{-16}$	..	0.000	0.834	0.000	1
$4.25 * 10^{-16}$	0.000	$5.56 * 10^{-16}$	0.000	..	0.811	0.000	0.983	1
$4.29 * 10^{-16}$	0.000	$9.98 * 10^{-16}$	0.000	..	0.696	0.000	0.967	1

Nella tabella sono rappresentate le prime e le ultime quattro colonne della matrice dei β.

Le osservazioni scritte in precedenza per gli α sono le medesime anche per i β. In questo caso, però, essendo u processo *backward*, il rischio di saturazione è presente nelle colonne iniziali della matrice dei β, che operativamente vengono calcolate per ultime.

5.2.2 Calcolo statistiche sufficienti attese

Il passo successivo è l'utilizzo di tali matrice per risalire alle statistiche sufficienti attese $\bar{M}[x, x']$ e $\bar{T}[x]$ (che scriveremo anche $E[M[j, k]]$ e $E[T[j]]$) che permettono di calcolare la massima verosimiglianza.

Per computare gli integrali alla base di tali matrici il metodo che abbiamo implementato è il *composite Simpson*. Si tratta di una tecnica per approssimare il valore di un integrale del quale è impossibile calcolare una primitiva. Si tratta dunque di un metodo di integrazione numerica che approssima un integrale definito.

In particolare, è la seguente approssimazione per $n + 1$ suddivisioni equamente spaziate.

$$\int_a^b f(x)dx \approx \frac{\Delta_x}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_f) + ... + 4f(x_{n-1}) + f(x_n))$$

dove $\Delta_x = \frac{b-a}{n}$ e $x_i = a + i\Delta_x$.

Pertanto la precisione di tale metodo - ovvero la capacità di approssimare correttamente un dato integrale - è diretta conseguenza del valore n. A valori più elevati di n corrispondono precisioni *non peggiori*. Ad n elevati il calcolo è però più oneroso e i tempi computazionali aumentano conseguentemente.
Tale metodo implementa una media pesata del campionamento dell'integrale in zone diverse ed equidistanziate. Il peso è ridotto per i valori di inizio e fine dell'integrale (x_0 e x_n) in quanto l'informazione che apportano è limitata rispetto ai valori interni. Infatti solamente da uno dei due lati essi hanno una spazio dove la funzione viene

computata.

Nella funzione da noi implementata abbiamo fissato $n = 10$, che ci è sembrato un compomesso accettabile tra precisione e velocità di esecuzione. Tale valore è pero un parametro da passare alle funzioni e può dunque essere adattato in base alle esigenze ed alle specifiche di tempo e precisione. Tale funzione calcola dunque un generico integrale dati gli estremi di integrazione e l'integranda.

Il passo successivo consiste nell'includere il *Composite Simpson* in una funzione che integra, dato un determinato stato della matrice amalgamata Q_{XH_X}, il tempo trascorso in tale stato su tutta la traiettoria di training: $E[T]$. La funzione viene poi iterata e il risultato finale è la quantità di tempo trascorsa dal sistema in ogni singola fase di ogni singolo stato. Dunque l'integrale da approssimare in maniera numerica è:

$$\int_v^w p(X_t, \sigma_{0:\tau}) e_j dt = \int_v^w \alpha_v exp(Q_S(t-v)) \Delta_{j,j} exp(Q_S(w-t)) \beta_w dt$$

Che viene poi utilizzato per il calcolo della durata attesa in un particolare stato j tramite la seguente formula:

$$E[T[j]] = \frac{1}{p(\sigma_{0:\tau})} \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} p(X_t, \sigma_{0:\tau}) e_j dt$$

.

Iterando per $\forall j \in Val(X)$ x (H) otteniamo come risultato una matrice di questa forma:

Durata attesa				
	x_1	x_2	x_3	x_4
h_1	0.21	0.04	0.13	0.04
h_2	0.07	0.04	0.09	0.02
h_3	0.14	0.06	0.11	0.04

Nella matrice mostrata la somma delle durate è stata normalizzata ad uno. Nell'esempio mostrato in tabella il numero di stati è uguale a 4 e il numero di fasi per ogni stato è uguale a 3.

Lo step più oneroso computazionalmente però è il calcolo di $E[M[j,k]]$, il numero atteso di transizioni osservate tra due stati. Tale onerosità deriva dalla necessità di calcolare un integrale con la medesima complessità che nel medesimo precedente ma con un numero di iterazioni proporzionale al quadrato del numero degli stati di Q_{XH_X}. $E[M[j,k]]$, per j e k generici, si compone di due addendi, calcolati su intervalli di evidenza costante e dove invece viene registrata una transizione. La base è un integrale calcolato per via numerica tramite *Composite Simpson* su tutta la traiettoria per ogni coppia di stati j e k. Ogni stato è una particolare configurazione della matrice amalgamata Q_{XH_X}, quindi è una coppia di x_i e h_j.

Il generico elemento $E[M[j,k]]$ viene calcolato tramite la seguente formula:

$$E[M[j,k]] = \frac{q_{j,k}}{p(\sigma_{0:\tau})} \left[\sum_{i=1}^{N-1} \alpha^{-}_{t_i} \Delta_{j,k} \beta^{+}_{t_i} + \sum_{i=1}^{N-1} \int_{v}^{w} \alpha_v exp(Q_S(t-v)) \Delta_{j,k} exp(Q_S(w-t) \beta_w dt \right]$$

Come possiamo notare, esistono due costanti moltiplicative uguali $\forall j,k \in Val(Q_{XH_X})$. In particolare $q_{j,k}$ è importante in quanto riusciamo a ridurre di molto il numero di integrali da calcolare effettivamente. Essendo $q_{j,k} = 0$ per ogni transizione non consentita, possiamo calcolare effettivamente l'integrale *solamente* per le coppie j,k che hanno $q_{j,k} \neq 0$. Anche in una semplice matrice amalgata con $k = 2$ e $h = 2$ le entrare sono 16 nella matrice, cui 4 sono elementi diagonali dei quali non dobbiamo calcolare $E[M[j,k]]$. Dei restanti 12 elementi solamente 6 hanno $q_{j,k}$ non nullo, pertanto sarà necessario eseguire 6 integrali.

Il procedimento rappresentato nella formula precedente viene iterato per ogni coppia di stati e il risultato finale è una tabella come la seguente:

Numero atteso di transizioni				
	x_1, h_1	x_1, h_2	x_2, h_1	x_2, h_2
x_1, h_1	-	107.0	0	0
x_1, h_2	6.7	-	99	0
x_2, h_1	0	0	-	134.9
x_2, h_2	99.1	0	36.1	-

Tale tabella deriva da una traiettoria generata dalla seguente matrice con $k = 2$ e $N = 2$:

		x_1		x_2	
		h_1	h_2	h_1	h_2
x_1	h_1	-3	3	0	0
	h_2	1	-3	2	0
x_1	h_1	0	0	-3	3
	h_2	2	0	1	-3

Avendo adesso costruito le funzioni necessarie per calcolare le statistiche sufficienti, ed essendo il passo di *Maximization* la semplice applicazione di formule, possiamo combinare tutto il lavoro fin qui nell'applicazione dell'algoritmo di Expectation Maximization.

5.2.3 Applicazione algoritmo

Tale algoritmo richiama le funzioni precedentemente definite di calcolo delle statistiche sufficienti attese $E[M[j,k]]$ e $E[T[j]]$. Dopodichè per trovare $E[M[j]]$- il numero di volte nelle quale il sistema transiziona *dallo* stato j- è sufficiente sommare $E[M[j,k]]$ $\forall k \in Val(Q_{XH_X})$.

Il Maximization step viene definito direttamente al suo interno mediante la formula

$$q_j^{k+1} = \frac{E[M[j]]}{E[T[j]]} \qquad \theta_j^{k+1} = \frac{E[M[j,k]]}{E[M[j]]}$$

La matrice risultante da questo step di EM viene determinata tramite le due quantità sovrastanti. Contestualmente viene anche calcolata la log-verosimiglianza attesa. Tale indicatore è molto importante per capire se le funzioni da noi sviluppate diano effettivamente i risultati corretti. Qualora infatti trovassimo che per due step successivi la log-verosimiglianza attesa aumenti sarebbe un indicatore che le formule da noi sviluppate siano incorrette.

Il metodo utilizzato è dunque quello di partite con una generica matrice di partenza con la medesima struttura della matrice della quale vogliamo imparare i parametri e aggiornarli iterativamente con passi successivi di EM, fino a raggiungere convergenza. L'output tipico di uno step di Expectation Maximization ha la struttura mostrata dall'esempio seguente, con struttura $k = 2$,$N = 2$:

- **Data-step traiettoria**=100
- **Log-Verosimiglianza**=107.39
- **Matrice E[T]**:

Durata attesa		
	x_1	x_2
h_1	12.9	35.5
h_2	21.8	31.4

- **Matrice E[M]**

Numero atteso di transizioni				
	x_1, h_1	x_1, h_2	x_2, h_1	x_2, h_2
x_1, h_1	-	107.0	0	0
x_1, h_2	6.7	-	99	0
x_2, h_1	0	0	-	134.9
x_2, h_2	99.1	0	36.1	-

- **Matrice risultante**

		x_1		x_2	
		h_1	h_2	h_1	h_2
x_1	h_1	-8.77	8.77	0	0
	h_2	0.55	-8.67	8.12	0
x_1	h_1	0	0	-3.57	3.57
	h_2	2.62	0	0.96	-3.58

Con quest'ultima che è semplicemente la matrice finale risultante dopo uno step di Expectation Maximization. Chiaramente anche essa è una CIM valida e in quanto tale ha la somma per riga uguale a zero. Tale matrice costituirà la matrice di input dello step successivo di EM.

Iterando il passo di Expectation Maximization finchè la differenze tra verosimiglianze non è trascurabile, ovvero finchè tale algoritmo non converge, si trova la matrice di massima verosimiglianza per la/e traiettorie sulle quali abbiamo trainato la nostra rete.

In applicazioni reali è limitativo però trovare un modello che aderisca nel miglior modo possibile ad una singola traiettoria, spesso è più utile addestrare una rete su un numero maggiore di traiettorie, per dare più informazioni alla rete. Anche per via della limitazione tecnica sulla lunghezza della rete di cui parlavamo precedentemente, è talvolta necessario spezzare una traiettoria osservata in più traiettorie di lunghezza inferiore.

L'architettura descritta quindi precedentemente è insufficiente ed è necessario sviluppare una funzione EM che possa prendere in input *più* traiettorie contemporaneamente.

In tale framework richiamiamo le funzioni che computano le statistiche sufficienti attese *per ogni traiettoria* di input. Applicando la relativa costante moltiplicativa $p(\sigma_{0:\tau})$ ad ogni traiettoria abbiamo che nella somma finale delle traiettorie ognuna di esse pesi esattamente la sua lunghezza definita in data-step. Dopo aver effettuato tale somma alla fase successiva utilizziamo $E[M[j,k]]$ e $E[T]$ totale per il calcolo della log-verosimiglianza e l'aggiornamento dei parametri della matrice finale. L'output tipico di tale funzione è dunque il medesimo visto nel caso con singola traiettoria in input, con una differenza che "Data-step traiettoria" viene calcolato come la somma dei data step di tutte le traiettorie.

5.3 EM con genitori

Nella sezione precedente abbiamo descritto le matrici tramite le quali abbiamo fatto imparare alla rete i suoi parametri. Rete che ricordiamo, per semplicità, abbiamo supposto essere composta solamente da un nodo X e dalla sua variabile Hidden H. In reti reali però il numero di nodi della rete è generalmente maggiore, e tali nodi sono in relazione tra di loro. Un nodo ha quindi una matrice differente per ogni instanziazione del suo insieme di variabili genitori U.

Tali variabili sono osservate e si modificano anche esse nel tempo. Vogliamo dunque applicare l'algoritmo di Expectation Maximization ad un nodo X della rete con dei genitori U. Tale passaggio risulterà particolarmente utile per confrontare tale approccio con la log-verosimiglianza di una EM con il nodo X senza genitori.

La generazione delle traiettorie avviene tramite un procedimento leggermente più complesso rispetto al caso senza genitori. In tale impostazione c'è una competizione

tra la matrice amalgamata Q_{XH_X} e la matrice delle variabili genitori Q_U.

Al principio si inizializzano entrambe le matrici in uno stato, avremo quindi uno stato u_{t_0} $x_{t_0}|u_{t_0}$, $h_{t_0}|u_{t_0}$. Dopodichè si effettuano due estrazioni da una esponenziale. La prima da una esponenziale di parametro $q_{x_{t_0},h_{t_0}|u_{t_0}}$, definita t_{xh_0}, che regola l'intensità di transizione dalla matrice amalgamata $Q_{XH_X|u_{t_0}}$. La seconda da una esponenziale di parametro $q_{u_{t_0}}$, definita t_{u_0} che regola l'intensità di transizione della matrice Q_U.

Dopodichè avviene un confronto tra questi due tempi; supponendo che venga estratto un tempo minore dalla matrice $Q_{XH_X|u_{t_0}}$ allora viene estratto uno stato di tale matrice dove avverrà la transizione da una multinomiale con i $\theta_{j,k}|u_{t_0}$ come parametri, chiamiamo lo stato dove transiziona la matrice x_{t_1}. Dopodichè viene estratta ancora una volta da una esponenziale con parametro $q_{x_{t_1}|u_{t_0}}$.

Tale tempo viene sommato al precedente $t = t_{x_0} + t_{x_1}$. Tale t viene confrontato con l'estrazione t_{u_0}. Nel caso risulti ancora minore allora si iterano i passaggi precedenti, continuando a definire t come la somma di tutte le estrazione dall'amalgamata $t = t_{x_n} + t_{x_{n-1}} + .. + t_{x_1} + t_{x_0}$. Fintantochè non si ottenga $t > t_{u_0}$. A quel punto avviene una transizione nelle variabili U, sempre tramite una estrazione da una multinomiale definita dalle matrici Q_U.

In tal caso, poichè configurazioni diverse nei genitori portano ad avere matrici molto diverse nelle variabili figlie X, H_X la scelta che facciamo è quella di mantenere consistente lo stato, ovvero non re-inizializzare lo stato e mantenere quello corrente. Re inizializziamo però il tempo dell'esponenziale derivante dalla matrice $Q_{XH_X|u_{t_0}}$, estraendo dallo stato corrente della matrice $Q_{XH_X|u_{t_1}}$. Tali passaggi vengono iterati fino al raggiungimento del tempo massimo di data-step.

Nell'esempio che mostreremo d'ora in avanti supporremo che l'insieme delle variabili genitori sia composto dalla sola variabile Y. Tratteremo dunque come sinonimi i termini U e Y, Q_Y e Q_U. L'estensione al caso caso multigenitoriale è chiara, è necessario solamente calcolare l'amalgamata delle matrici $U_1, U_2, ..U_N$.

Realizzazione traiettoria con k=3 ; N=4				
stato Q_{XH_X}	inizio	fine	stato x	stato y
2	0.00000000	0.1583213	1	1
3	0.1583213	1.0242059	2	1
4	1.0242059	1.5246304	2	1
2	1.5246304	1.6256492	1	1
2	1.6256492	2.5899489	1	2

Nella quale in fase di *training* della rete viene oscurata la colonna Q_{XH_X}.
Il risultato è:

Realizzazione traiettoria con k=3 ; N=4			
inizio	fine	stato x	stato y
0.00000000	0.1583213	1	1
0.1583213	1.0242059	2	1
1.5246304	1.6256492	1	1
1.6256492	2.5899489	1	2
2.5899489	3.1654299	2	2

Come possiamo osservare, in questo estratto di tabella sono contenuti solo le indicazioni degli estremi temporali, dello stato di X e dello stato di Y. In più, ogni riga racchiude tutto il tempo trascorso nello stato x_i per una determinata configurazione $Y = y_i$, che è dunque la somma del tempo trascorso nell'insieme delle fasi che compongono x_i dato y_i.
Tale traiettoria verrà data in input alle funzioni che realizzeranno l'Expectation Maximization.

Alla base di tutto il procedimento vi sono le medesime funzioni che abbiamo visto nella sezione precedente. Ovvero le funzioni che data una certa matrice la trasformano nella matrice *within* (Q_{s_i}) o *beetwen* di un certo sottosistema. Q_{s_i,s_j}

Un cambiamento si registra invece nel calcolo di α e β. L'interpretazione è la medesima ma cambia la formula per il loro calcolo.
E' necessario un distinguo nei casi nei quali nella traiettoria osservata si sia in presenza di un cambiamento nello stato X oppure di un cambiamento nella variabile genitore Y.
Nel primo caso la formula iterativa per il calcolo è invariante rispetto al caso senza genitori. Mentre con un cambiamento della variabile genitore Y la formula per il calcolo è diversa, in quanto *omette* l'informazione relativa alla transizione nel sottosistema Q_S associato alla X corrente. Le inizializzazioni sono le medesime mostrate nel caso precedente sia per α che per β. Mostrando in tabella un estratto da β_t:

Esempio di β_t con k=2; N=2				
	t_0	t_1	t_2	t_3
x_1, h_1	$7.67 * 10^{-18}$	$1.28 * 10^{-17}$	0.000	$1.96 * 10^{-17}$
x_1, h_2	$8.51 * 10^{-18}$	$1.96 * 10^{-17}$	0.000	$3.41 * 10^{-17}$
x_2, h_1	0	0	$4.12 * 10^{-17}$	0.000
x_2, h_2	0	0	$2.38 * 10^{-16}$	0.000

Come possiamo notare anche in questo caso in un dato tempo $(t_0, t_1, ..)$ sono diverse da zero solo le righe corrispondenti alle fasi di un unico stato X: quello nel quale il sistema è vincolato nel dato tempo. Al contrario che nella rappresentazione precedente è pero possibile in questo caso che due colonne adiacenti abbiamo le stesse righe non nulle. Questo avviene in quanto in $\Delta_t = t_1 - t_0$ che viene riportato è presente una transizione della variabile Y, con X a evidenza costante nella porzione di traiettoria.

Invece per quanto riguarda α^- e β^+ esse sono definite in modo iterativo partendo da α e β, pertanto la loro formula non è variata in alcun modo ma i risultati differiscono a causa della differente definizione delle matrici α e β.

Particolarmente interessante è invece il calcolo delle statistiche sufficienti attese. Esse non sono più generali ma sono specifiche per ogni instanziazione di Y. Esiste quindi una durata attesa *per ogni stato* per ogni instanziazione di $y = Y$. Anche la statistica sufficiente che regola il numero di transizioni, $E[M[i,j]]$ non è più generale su ogni traiettoria ma è specifica per ogni instanziazione delle variabili genitori.

Pertanto esistono tante coppie di statistiche sufficienti attese quante sono le instanziazioni delle variabili genitori.
La formula per il calcolo della durata attesa in un generico stato j *data* $Y = y$ è:

$$E[\, T[j] \mid Y = y] = \frac{1}{p(\sigma_{0:\tau})} \sum_{i=0}^{N-1} \mathbb{1}(Y_i = y) \int_{t_i}^{t_{i+1}} p(X_t, \sigma_{0:\tau}) e_j dt$$

Ovvero si sommano tutti gli integrali dove $Y = y$. Iterando per ogni stato di X ed ogni instanza di Y si ottengono tutte le durate medie attese di interesse.
Il risultato finale è una tabella di questa forma per una configurazione nella quale supponiamo numero di stati di X pari a 2, numero di fasi pari a 2 e una singola variabile genitore Y con due stati:

Durata attesa $Y = y_1$			
	x_1	x_2	x_3
h_1	1.48	1.64	2.25
h_2	0.55	1.05	0.89

Durata attesa $Y = y_2$			
	x_1	x_2	x_3
h_1	0.57	4.32	3.05
h_2	0.48	3.24	0.46

Come possiamo vedere abbiamo un numero di tabella pari alle diverse instanziazioni della variabile genitore Y. Anche in questo primo semplice esempio è possibile notare come il tempo trascorso in ogni singolo stato può differire di molto al cambiamento dello stato Y. Anche la somma dei tempi totali trascorsi in ogni stato è diversa per la tabella dato y_1 e dato y_2.

Per quanto riguarda il numero atteso di transizioni condizionato $E[M[j,k]|Y = y]$ la formula che ne permette il calcolo segue lo stesso principio di dividere la sommatoria che genera $E[M[j,k]]$ globale in più sommatorie, a seconda dello stato $Y = y$ per trovare $E[M[j,k]|Y = y]$.

Il risultato finale è una tabella di questa forma per una configurazione nella quale supponiamo numero di stati di X pari a 3, numero di fasi pari a 2 e una singola variabile genitore Y con solamente due stati:

Numero atteso di transizioni per $Y = y_1$						
	x_1, h_1	x_1, h_2	x_2, h_1	x_2, h_2	x_3, h_1	x_3, h_2
x_1, h_1	-	15.2	0	0	0	0
x_1, h_2	3.3	-	4.0	0	0.4	0
x_2, h_1	0	0	-	7.1	0	0
x_2, h_2	1.1	0	3.6	-	2.2	0
x_3, h_1	0	0	0	0	-	2.4
x_3, h_2	2	0	0.2	0	0.4	-

Numero atteso di transizioni per $Y = y_2$						
	x_1, h_1	x_1, h_2	x_2, h_1	x_2, h_2	x_3, h_1	x_3, h_2
x_1, h_1	-	2.9	0	0	0	0
x_1, h_2	0.6	-	2.0	0	0.5	0
x_2, h_1	0	0	-	3.9	0	0
x_2, h_2	1.0	0	0.5	-	3.0	0
x_3, h_1	0	0	0	0	-	3.9
x_3, h_2	2	0	1	0	0.9	-

Come possiamo osservare ad ogni instanziazione di Y corrisponde una tabella diversa. Le uniche transizioni lecite sono quelle con numero atteso di transizioni diverso da zero. Tali transizioni possono essere diverse nelle due tabelle diverse, portando alcune transizioni ad essere possibili *solamente* per certi valori dei genitori Y.

Dopo aver quindi definito tutte le matrici delle statistiche sufficienti attese possiamo quindi passare allo step di Maximization. Tale step, data la sua semplicità, è incorporato all'interno della funzione che computa uno step di Expectation Maximization e che prende in input la funzioni che calcolano le statistiche sufficienti attese condizionate. Tale funzione prende in input una traiettoria e le matrici di partenza, la matrici al passo k. Tante matrici quante instanze delle variabili genitori e restituisce le stesse matrici con i parametri aggiornati grazie alle informazioni delle traiettoria: le matrici al passo $k + 1$.

Il maximization step definito al suo interno ha una formula diversa rispetto al caso senza genitori. Per ogni matrice da aggiornare, ovvero per ogni istanza di Y, la formula è la seguente:

$$q_j^{k+1}|y_i = \frac{E[M[j]|y_i]}{E[T[j]|y_i]} \qquad \theta_j^{k+1}|y_i = \frac{E[M[j,k]|y_i]}{E[M[j]|y_i]}$$

In tale step viene anche calcolata la log-verosimiglianza attesa globale. Che altro non è che la somma pesata della verosimiglianza per ogni istanziazione di Y, con peso il numero di data-step nel quale il sistema è stato nello stato y_i. Tale log-verosimiglianza è la funzione da massimizzare e l'indice di buon andamento dell'algoritmo è il non-aumento della log-verosimiglianza per iterazioni successive dell'algoritmo. Infatti l'al-

goritmo viene iterato per passi successivi fino all'avvenuta convergenza.

L'output caratteristico del k-esimo passo di Expectation Maximization è il seguente, supponendo di avere numero di stati di X pari a 3 numero di fasi pari a 2 e un solo genitore Y con due stati y_1, y_2:

1. $Y = y_1$

   - **Data-step traiettoria**=76.45953
   - **Log-Verosimiglianza condizionata**=-99.72384
   - **Matrice E[T|Y]**:

Durata attesa			
	x_1	x_2	x_3
h_1	25.7	13.2	12.8
h_2	8.9	6.6	9.2

   - **Matrice E[M|Y]**

Numero atteso di transizioni						
	x_1, h_1	x_1, h_2	x_2, h_1	x_2, h_2	x_3, h_1	x_3, h_2
x_1, h_1	-	49.8	0	0	0	0
x_1, h_2	11.5	-	19	0	0	0
x_2, h_1	0	0	-	25.3	0	0
x_2, h_2	2.0	0	4.2	-	18.0	0
x_3, h_1	0	0	0	0	-	30.0
x_3, h_2	36.0	0	4.0	0	8.4	-

   - **Matrice risultante**

		x_1		x_2		x_3	
		h_1	h_2	h_1	h_2	h_1	h_2
x_1	h_1	-1.94	1-94	0	0	0	0
	h_2	1.30	-5.36	2.14	0	1.92	0
x_1	h_1	0	0	-1.91	1.91	0	0
	h_2	0.30	0	0.64	-3.68	0	2.73
x_3	h_1	0	0	0	0	-2.34	2.34
	h_2	3.89	0	0.43	0	0.91	-5.23

2. $Y = y_2$

   - **Data-step traiettoria**=123.53153
   - **Log-Verosimiglianza condizionata**=-107.48721

- **Matrice E[T|Y]**:

Durata attesa			
	x_1	x_2	x_3
h_1	8.9	18.6	33.6
h_2	8.4	19.4	34.5

- **Matrice E[M | Y]**

Numero atteso di transizioni						
	x_1, h_1	x_1, h_2	x_2, h_1	x_2, h_2	x_3, h_1	x_3, h_2
x_1, h_1	-	33.6	0	0	0	0
x_1, h_2	2.9	-	30.0	0	3	0
x_2, h_1	0	0	-	60.6	0	0
x_2, h_2	6.1	0	12.9	-	43.2	0
x_3, h_1	0	0	0	0	-	47.7
x_3, h_2	25.3	0	15.3	0	9.0	-

- **Matrice risultante**

		x_1		x_2		x_3	
		h_1	h_2	h_1	h_2	h_1	h_2
x_1	h_1	-3.77	3.77	0	0	0	0
	h_2	0.34	-4.26	3.55	0	0.36	0
x_1	h_1	0	0	-3.25	3.25	0	0
	h_2	0.32	0	0.66	-3.19	2.21	0
x_3	h_1	0	0	0	0	-1.42	1.42
	h_2	0.73	0	0.44	0	0.26	-1.44

L'output risultante è dunque una coppia di liste degli stessi oggetti già presenti per la configurazione senza genitori. Per trovare la verosimiglianza totale è sufficienti sommare le due *Log-verosimiglianze condizionate* delle due liste.

Anche per l'Expectation Maximization di un nodo con genitori è possibile sfruttare l'informazione di più traiettorie per imparare i parametri delle matrici. L'algoritmo si limita a computare le statistiche sufficienti attesi per ogni traiettoria, sommarle e infine passare al Maximization step per aggiornare i parametri.

Capitolo 6

Risultati

Abbiamo implementato l'algoritmo di EM descritto sopra nella configurazione senza genitori. Partendo da una matrice in forma diretta espressa dall'esempio 3.1 abbiamo creato la rappresentazione bipartite di tale matrice. Dopo aver generato delle traiettorie campionarie, le abbiamo usato per imparare i parametri della rete tramite Expectation Maximization. Questo passaggio è avvenuto nascondendo le transizioni che riguardavano la variabile Hidden H_X, supponendole ignote. Mostrando all'algoritmo di EM solamente le transizioni tra sottosistemi S_i, S_j. I risultati sono stati molto soddisfacenti, l'algoritmo riduceva ad ogni iterazione la log-verosimiglianza fino a raggiungere convergenza. Siamo dunque riusciti ad apprendere una rete con memoria, data dalle distribuzioni di fase, in un contesto di variabili Hidden. Tale rappresentazione presenta gli stessi parametri liberi della rappresentazione diretta ma con numerosi vantaggi di interpretabilità e inferenza.

6.1 Memoria vs assenza di memoria

Una rete CTBN con distribuzione di fase offre numerosi vantaggi rispetto ad una rete senza distribuzioni di fase. Riesce infatti a superare la limitazione dell'assenza di memoria e risulta quindi una rappresentazione più ricca e completa del sistema. Per dimostrare l'effetiva superiorità delle reti con *distribuzioni di fase* su reti semplici abbiamo svolto i seguenti esperimenti con configurazioni molto diverse tra di loro per provare la generalità dei vantaggi di tali reti.

In primo luogo abbiamo generato delle traiettorie da una CTBN con distribuzione di fase, con numero variabile di fasi. In seguito abbiamo utilizzato tali traiettorie per imparare i parametri di una CTBN *senza distribuzione di fase* e abbiamo applicato EM per imparare una CTBN *con* memoria.

Per confrontare le due reti finali abbiamo calcolato la log-verosimiglianza su dati di test. Ovvero abbiamo studiato come la nostra rete riesce ad essere aderente a dati che non ha mai visto prima. Dati che, ovviamente, sono stati generati *indipendentemente* dalla stessa distribuzione usata per la fase di learning. Abbiamo quindi confrontato i due modelli tramite previsioni *out-of-sample*. Il confronto quindi avviene mediante log-verosimiglianza.

Per il primo esempio abbiamo utilizzato un nodo con variabile X binaria e con una distribuzione consistente di due fasi. La matrice dalla quale sono stati generati i dati è la seguente rappresentata:

$$Q_{XH_X} = \begin{bmatrix} -2.0 & 2.0 & 0.0 & 0.0 \\ 1.5 & -2.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & -7.0 & 7.0 \\ 4.0 & 0.0 & 3.0 & -7.0 \end{bmatrix}$$

Nella quale $Val(X) = \{x_1, x_2\}$ e $Val(H) = \{h_1, h_2\}$.

La matrice Q_{XH_X} è la matrice in forma diretta con due stati X e due fasi H. Si entra nello stato tramite la fase di entrata h_1, dopodichè all'interno dello stato è permessa la transizione forward $h_1 \to h_2$ ma anche backward $h_2 \to h_1$, uscendo dallo stato tramite la fase di uscita h_2. Le prime due righe rappresentano gli stati h_1, x_1 e h_2, x_1, ovvero le fasi interni allo stato x_1. Le ultime due righe rappresentano invece gli stati h_1, x_2 e h_2, x_2, ovvero le fasi intenrni allo stato x_2.

Tramite le traiettorie generate da tale matrice sono state imparate due matrice, una CTBN con distribuzione di fase e una senza. Esse sono state messe a confronto con traiettorie di test di lunghezza diversa, e il risultato risultante è riassunto in una tabella con la seguente forma

Memoria vs senza memoria		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	..	..
100	-..	..
150	..	..
200	..	..
250	..	..

Dalle tabelle che presenteremo successivamente strutturate in questa maniera è possibile mettere a confronto architetture con e senza memoria, imparate sui medesimi dati di train e su dati di test indipendenti tra di loro e diversi per numero di data-step. Il risultato che ci auspichiamo è che il modello con memoria raggiunga performance migliori - ovvero verosimiglianze più basse - rispetto al suo omologo senza memoria.

Un altro risultato ottenuto è rappresentato nelle figure sottostanti. Nella figura sottostante sono rappresentati log-verosimiglianze su dati di test, quindi mai osservati prima dalla rete, in configurazioni con e senza memoria per reti imparate con dati diversi. Le traiettorie sono generate da una CTBN con una distribuzione di fase composta da due fasi.

La matrice che ha generato i dati è la medesima per ogni grafico della figura, nei vari grafici cambia solamente la quantità di informazione, definita in data-step, data alla rete in fase di *learning* dei parametri.
Per comporre il grafico sono state valutate le due reti imparate - con e senza memoria

- per dataset di test diversi e di lunghezza diversa.

6.1.1 Distribuzione di fase N=2

Vediamo che anche solamente una distribuzione di fase molto semplice, composta solamente da due fasi, è rappresentabile in maniera generalmente migliore con una rete con memoria rispetto che senza.

Figura 6.1: Log-verosimiglianza CTBN con distribuzioni di fase N=2 vs log-verosimiglianza CTBN senza memoria

Partendo dalla figura in alto a destra abbiamo il grafico per dati imparati con traiettorie di training di lunghezza 100 data-step. In alto a sinistra la traiettoria usata per l'apprendimento ha lunghezza 200 data step. In basso a destra 300 datastep mentre in basso a sinistra sono rappresentate le log veromiglianze su dati di test di lunghezza diverse per traiettorie imparate con 400 datastep.

Nei grafici soprastanti la linea rossa rappresenta la log-verosimiglianza delle reti con distribuzione di fase, mentre la linea blu rappresenta la log-verosimiglianza delle reti con assenza di memoria. Possiamo osservare come nei due grafici nella riga superiore e nel grafico in basso a sinistra risulta supportato il modello con memoria. L'unica eccezione è il modello con training di 300 data-step in basso a sinistra, nel quale risulta migliore la configurazione senza memoria.

Data step=100:

Memoria vs senza memoria 2 fasi training 100 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	+77.6	-32.4
100	+149.1	-62.16
150	256.85	-95.1
200	+329.81	-112.11

Data step=200:

Memoria vs senza memoria 2 fasi training 200 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-33.2	-25.5
100	-27.3	-65.4
150	-66.5	-105.7
200	-100.8	-111.9

Data step=300:

Memoria vs senza memoria 2 fasi training 300 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-38.4	-33.5
100	-93.5	-63.7
150	-127.6	-88.2
200	-183.4	-127.2

Data step=400:

Memoria vs senza memoria 2 fasi training 400 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	75.9	-36.3
100	13.7	-65.1
150	-38.3	-80.8
200	-0.78	-126.5

Risulta dunque che con due sole fasi a generare i dati il riconoscimento della memoria non è scontato. Per riuscire a supportare il modello corretto è necessario aumentare la lunghezza delle traiettorie di training.

6.1.2 Distribuzione di fase = 3

.

In questa sezione vediamo invece un confronto tra log-verosimiglianze quando il numero di fasi per ogni stato è pari a 3.

Aumentando il numero di fasi, rendiamo possibile per la distribuzione di fase una flessibilità maggiore che dovrebbe risultare in un miglioramento più accentuato che nella sezione precedente confrontandosi con reti senza memoria.

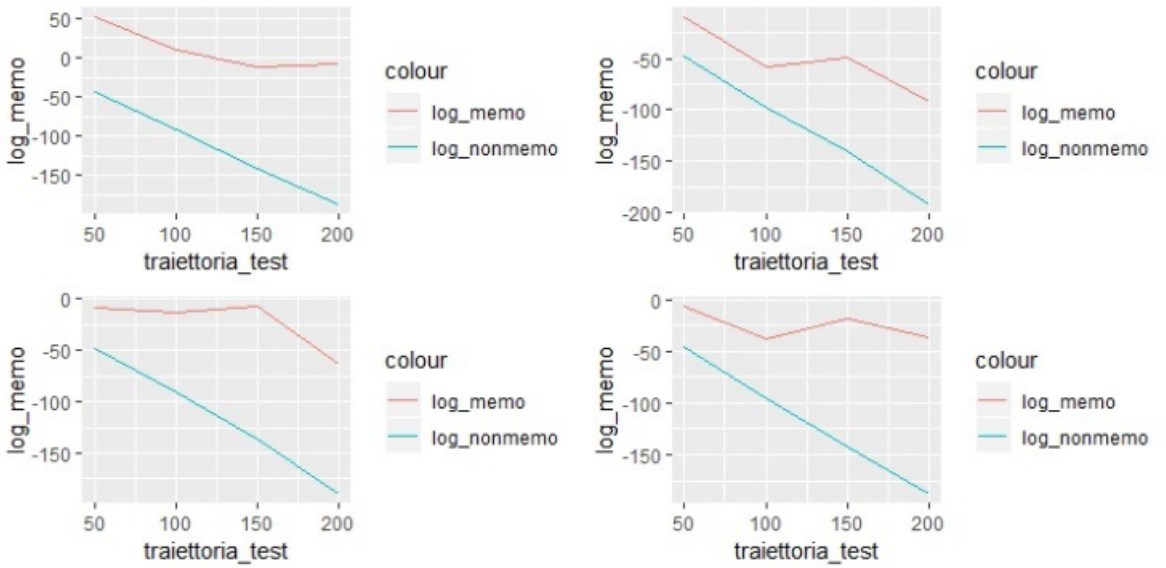


Figura 6.2: Log-verosimiglianza CTBN con distribuzioni di fase N=3 vs log-verosimiglianza CTBN senza memoria

Nella figura sono rappresentati log-verosimiglianze su dati di test, quindi mai osservati prima dalla rete, in configurazioni con e senza memoria per reti imparate con dati diversi.

Partendo dalla figura in alto a destra abbiamo il grafico per dati imparati con traiettorie di training di lunghezza 100 data-step. In alto a sinistra la traiettoria usata per l'apprendimento ha lunghezza 200 data step. In basso a destra 300 datastep

mentre in basso a sinistra sono rappresentate le log veromiglianze su dati di test di lunghezza diverse per traiettorie imparate con 500 datastep.

In ognuno di questi grafici vediamo come sia preferito il modello con memoria, colorato in rosso. Possiamo osservare come con tre fasi le differenze in termini di log-verosimiglianza siano marcate anche avendo a disposizione un numero limitato di dati di training. Questo poichè la ricchezza di una distribuzione di fase è più difficile da sintetizzare con una singola esponenziale al crescere delle fasi che la compongono.

Riportiamo di seguito i risultati in maniera numerica dai quali si è generato il grafico sovrastante:

Data step=100:

Memoria vs senza memoria 3 fasi training 100 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	+52.1	-44.6
100	+10.5	-91.46
150	-11.9	-140.7
200	-8.2	-187.4

Data step=200:

Memoria vs senza memoria 3 fasi training 200 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-9.2	-47.3
100	-58.0	-98.6
150	-49.8	-141.2
200	-92.2	-192.37

Data step=300:

Memoria vs senza memoria 4 fasi training 300 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-8.8	-47.6
100	-12.9	-91.2
150	-6.9	-135.62
200	-62.9	-189.3

Data step=500:

Memoria vs senza memoria 4 fasi training 500 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-6.41	-45.6
100	-37.7	-95.5
150	-18.5	-143.2
200	-36.7	-187.6

Con traiettorie di training di lunghezza diversa i risultati sono i medesimi, con la configurazione con memoria che si dimostra sempre più efficiente. All'aumentare dei dati a disposizione la differenza tra log-verosimiglianza in media aumenta. Avendo quindi a disposizione più informazione è possibile sfruttare meglio la flessibilità del modello con memoria.

6.1.3 Distribuzione di fase = 4

.

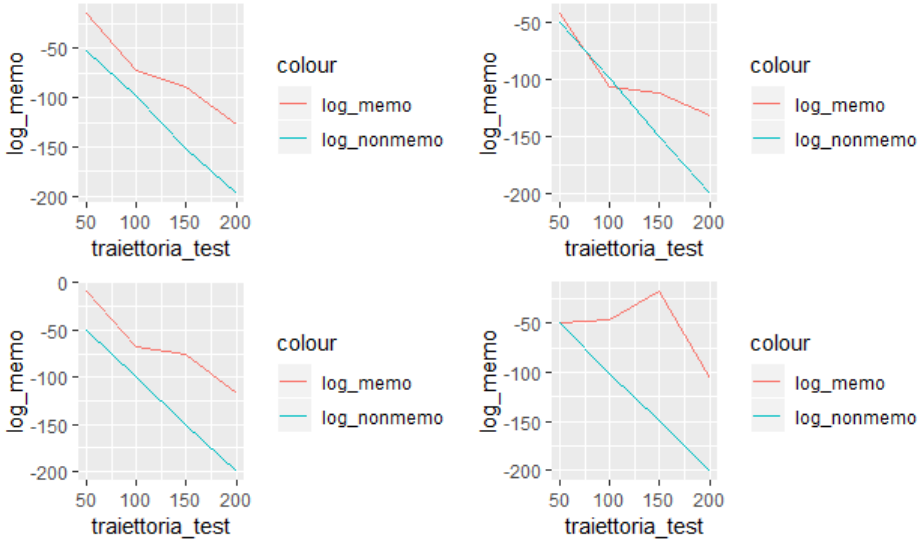


Figura 6.3: Log-verosimiglianza CTBN con distribuzioni di fase N=4 vs log-verosimiglianza CTBN senza memoria

Aumentiamo ancora il numero di fasi all'interno di ogni variabile X della nostra rete. Generando i dati da un modello con numero di fasi pari a quattro stiamo utilizzando una distribuzione *più* complessa per generare le traiettorie, discostandosi in maniera netta dalla distribuzione esponenziale che viene supposta da un modello senza memoria.

Mettiamo dunque ancora a confronto reti imparate da traiettorie con time-step differenti su vari test set tutti indipendenti tra loro e difformi per time-step.

Il grafico è strutturato alla stessa maniera dei precedenti: in ogni figura è presente un confronto tra la verosimiglianza su dati di test per reti con memoria e reti senza memoria. In alto a sinistra entrambe le reti vengono imparate con una traiettoria di 100 data-step; 200 per la figura in alto a sinistra, 300 per la figura in basso a sinistra e 500 per la figura in basso a destra.

Riportiamo di seguito i risultati in maniera numerica dai quali si è generato il grafico sovrastante:

Data step=100:

Memoria vs senza memoria 4 fasi training 100 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-13.9	-52.0
100	-71.7	-98.9
150	-89.1	-151.5
200	-128.0	-197.55

Data step=200:

Memoria vs senza memoria 4 fasi training 200 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-41.9	-49.5
100	-106.3	-98.3
150	-111.7	-149.2
200	-130.8	-199.6

Data step=300:

Memoria vs senza memoria 4 fasi training 300 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-8.5	-50.1
100	-68.4	-99.1
150	-75.9	-151.0
200	-116.9	-199.7

Data step=500:

Memoria vs senza memoria 4 fasi training 500 data-step		
data-step traiettoria	log-vero memoria	log-vero senza memoria
50	-49.8	-49.3
100	-46.3	-100.9
150	-16.7	-148.9
200	-106.3	-200.9

Possiamo notare come in ogni configurazione risulta essere supportata la rete con memoria, almeno per dataset di test con numero di data-step elevato. Soprattutto nella rete che è stata imparata con 500 time-step, ovvero quella che ha ricevuto più informazione, il valore della log-verosimiglianza è decisamente differente per test set da almeno 100 time-step.

6.2 Genitore vs senza genitore

.

Abbiamo quindi implementato l'algoritmo EM descritto sopra nei due casi, con genitori o senza genitori. Senza perdita di generalità abbiamo supposto che la variabile Y - possibile genitore di X - sia binaria e possa assumere solamente gli stati y_1 e y_2.

6.2.1 Configurazione k=2, N=2

Per semplicità supponiamo in primo luogo che anche X sia binaria, pertanto $Val(X) = \{x_1, x_2\}$. Con ognuno dei due stati che si distribuisce secondo una distribuzione di fase data dalla variabile H_X anche essa binaria. Pertanto le matrici condizionali in forma diretta sono della forma:

$$Q_{XH_X|y_1} = \begin{bmatrix} -2.0 & 2.0 & 0.0 & 0.0 \\ 1.5 & -2.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & -7.0 & 7.0 \\ 4.0 & 0.0 & 3.0 & -7.0 \end{bmatrix} \qquad Q_{XH_X|y_2} = \begin{bmatrix} -7.0 & 7.0 & 0.0 & 0.0 \\ 4.0 & -7.0 & 3.0 & 0.0 \\ 0.0 & 0.0 & -2.0 & 2.0 \\ 0.5 & 0.0 & 1.0 & -1.5 \end{bmatrix}$$

Dove Q_{XH_X} rappresenta la matrice del nodo X in forma diretta, corrispondente anche all'amalgamata di Q_{XH_X}.

Per l'apprendimento di entrambe le reti abbiamo oscurato l'informazione relativa allo stato H e abbiamo lavorato su traiettorie parzialmente osservata σ. Per imparare i parametri di tali matrici abbiamo usato l'algoritmo di expectation maximization descritto nella fase di learning nei casi con genitore e senza genitore.

Questo confronto è molto importante ai fini pratici in quanto è tramite il confrontro tra queste verosimiglianze che si decide se una variabile è o meno genitore di un'altra. Tramite l'iterazione di questo passaggio base è quindi possibile imparare l'intera struttura della rete.

Come primo esempio generiamo una traiettoria da una rete Bayesiana con numero di stati pari a due e numero di fasi anche essi pari a due dalle matrici soprastanti. La traiettoria è stata generata supponendo che la variabile X sia figlia della variabile Y anche essa binaria. tale traiettoria verrà poi utilizzata, mascherando le informazioni relative agli stati nascosti H, per imparare due CTBN con distribuzione di fase.
La prima CTBN suppone che Y sia genitore di X e stima quindi due matrici per le due diverse instanze di Y. Mentre la seconda CTBN suppone che Y non sia genitore di X e stima una sono matrice.

Dalla stessa matrice che ha generato i dati di training con i quali abbiamo imparato le matrici generiamo delle ulteriori traiettorie, tutte indipendenti, che non diamo alla reti in fase di apprendimento. Tali traiettorie serviranno per valutare le reti imparate, secondo il principio di separazione tra train e test. I risultati sono riassunti nella tabella seguente nella quale sono riassunte le verosimiglianze osservate per diversi dataset di test, ognuno con numero di data-step diverso. Il dataset di training usato per imparare le due matrici in questo primo esempio è composto da 400 data-step.

Genitore vs non genitore		
data-step traiettoria	log-vero Y genitore	log-vero senza genitore
50	484	1567
100	3618,4	2037
200	3024	4481
300	5238	5280
400	8778	7371
500	10612	9266
600	12689	11921

Vediamo come il nostro modello è in grado di riconoscere che Y è genitore di X solamente quando messo di fronte a traiettorie di test di lunghezza superiore a 400 data-step. La mancanza di capacità di identificare la relazione tra X ed Y con test set limitati è probabimente dovuta alla insufficiente quantità di dati somministrati al modello. Sarebbero comunque utili ulteriori verifiche.

Il modello con Y genitore di X ha esattamente il doppio dei parametri liberi da stimare rispetto al modello senza genitori, in quanto deve stimare una matrice per ogni

instanziazione di Y. Pertanto necessita di una quantità sufficiente di informazione per stimare tale modello.

Nell'esempio successivo usiamo un training set per imparare i parametri più ampio, composto da 600 data-step. Avendo maggiore informazione riusciamo a identificare meglio i parametri delle matrici sotto l'ipotesi che Y sia genitore di X. Pertanto possiamo osservare nella tabella seguente come riusciamo ad identificare correttamente Y come genitore di X molto più semplicemente rispetto all'esempio sovrastante.

Genitore vs non genitore		
data-step traiettoria	log-vero Y genitore	log-vero senza genitore
50	-41	-51
100	-76	-87
200	-153	-187
300	-279	-235
400	-325	-373
500	-455	-479
600	-514	-573

Abbiamo quindi appurato come siamo in grado di riconoscere se Y è genitore di X ma abbiamo bisogno di addestrare la nostra rete con un numero di dati sufficienti per stimare adeguatamente tutti i parametri delle matrici.

6.2.2 Configurazione k=2, N=3

Adesso invece consideriamo una configurazione nella quale il numero di fasi associate ad ogni stato X passa da due a tre, consentendo quindi una maggiore flessibilità.

I risultati sono riassunti nella tabella seguente nella quale sono riassunte le verosimiglianze osservate per diversi dataset di test, ognuno con numero di data-step diverso.

Genitore vs non genitore		
data-step traiettoria	log-vero Y genitore	log-vero senza genitore
50	98190	77,66
100	292743,4	1011,403
200	325190	675
300	1399405	3186
250	646462	2000
300	727687	2142.671
250	1224773	2254.361
250	1444736	3015.373

I risultati sono molto positivi: abbiamo costruito un framework in grado di identificare in maniera chiara che la variabile Y influenza la variabile X. Una log-verosimiglianza maggiore significa che condizionare X a Y porta ad una modello che

aderisce in maniera migliore ai dati di test.

Generando i dati da due matrici diversi, a seconda che siano condizionate ad y_1 o y_2 abbiamo raggiunto un risultato migliore, in termini di log-verosimiglianza, nella configurazione con Y genitore di X. Tale risultato ci conforta poichè abbiamo generato i dati supponendo vera tale ipotesi. Notiamo inoltre come la convergenza si raggiunga più lentamente nella configurazione con Y genitore di X.

6.2.3 Configurazione k=3, N=2

Supponiamo adesso di generare le traiettorie di training e di test da una coppia di matrice con numero di stati X pari a 3 e numero di fasi uguale per ogni stato e pari a 2.

Le matrici di partenza hanno dunque la forma seguente:

$$Q_{XH_X|y_1} = \begin{bmatrix} -5.0 & 5.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.5 & -5.0 & 4.0 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & -3.0 & 3.0 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.5 & -3.0 & 2.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -2.0 & 2.0 \\ 1.0 & 0.0 & 0.5 & 0.0 & 0.5 & -2.0 \end{bmatrix}$$

$$Q_{XH_X|y_2} = \begin{bmatrix} -2.0 & 2.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.5 & -2.0 & 1.0 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & -3.0 & 3.0 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.5 & -3.0 & 2.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -4.0 & 4.0 \\ 3.0 & 0.0 & 0.5 & 0.0 & 0.5 & -4.0 \end{bmatrix}$$

Possiamo osservare che in entrambe le matrici sono lecite le stesse transizioni con parametri e dunque pesi però diversi.

La matrice Q_Y che regola invece le transizioni per il nodo genitore Y è binaria ed è strutturata nel modo seguente:

$$Q_Y = \begin{bmatrix} -0.1 & 0.1 \\ 0.1 & -0.1 \end{bmatrix}$$

Come nelle sottosezioni precedenti, alleniamo la nostra rete con delle traiettorie di training generate dalle matrici sovrastanti. Per indagare al meglio il comportamento di questa rete per scenari diversi valutiamo la rete modificando la lunghezza della traiettoria di training. Per ogni traiettoria la valutazione avviene mediante la log-verosimiglianza su traiettorie di test mai osservate in precedenza dal modello. Tali traiettorie sono generate dalle medesime matrici che generano la traiettoria di training in modo indipendente e con lunghezza diversa.

Da questo insieme di prove emerge la complessità e i punti di forza e debolezza dell'approccio. Il grafico seguente sintetizza tutte le analisi in modo visuale.

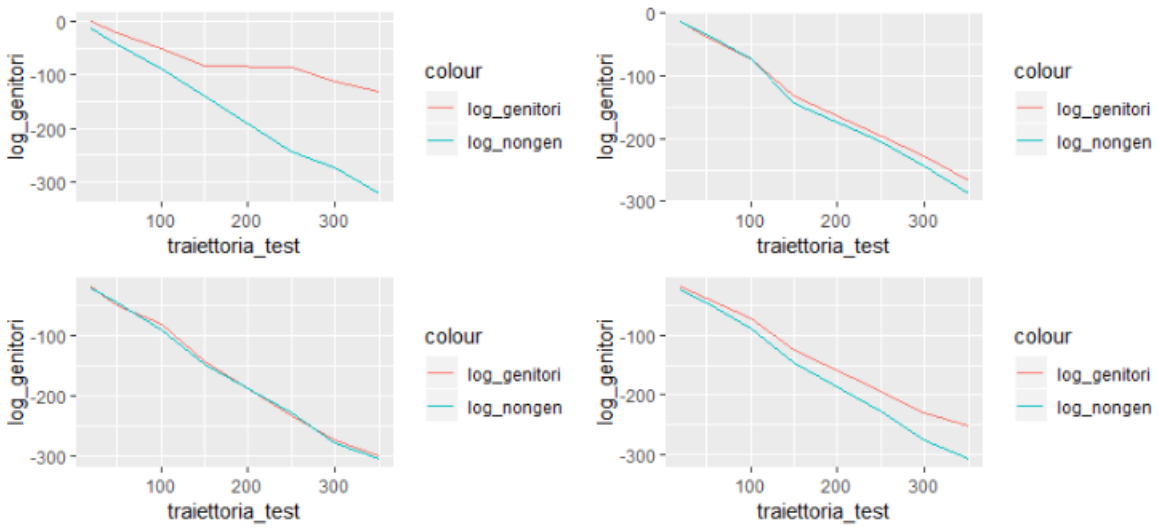


Figura 6.4: Log-verosimiglianza configurazione con genitore vs senza genitore

Nel grafico sono rappresentate le log-verosimiglianze per dati di test di lunghezza diversa per reti con memoria (linea rossa) e senza memoria (linea blu). Nelle quattro figure varia la lunghezza della traiettoria di training presa in esame.

Rispettivamente, nel grafico in alto a destra la traiettoria di training è lunga 20 data-step, che diventano 130 per la figura in alto a sinistra, 200 per la figura in basso a destra e 350 per la figura in basso a sinistra.

Dal grafico, e più chiaramente si noterà dalle tabelle seguenti, possiamo trarre alcune conclusioni. Innanzitutto la prima figura, in alto a sinistra, è il risultato di reti imparate con *solamente* 20 data-step. Pertanto le migliori performance evidenziate dal grafico per la configurazione con memoria possono essere il semplice frutto della alta variabilità presente.
Per quanto riguarda le tre figure rimanenti invece, notiamo che il modello faccia molta fatica a decidere quale configurazione è maggiormente supportata. Può essere il frutto della similarità tra le matrici di input $Q_{XH_X|y_1}$ e $Q_{XH_X|y_2}$, oppure frutto del campione relativamente ridotto utilizzato per imparare la rete.

Comunque, si nota come la configurazione con memoria tenda ad avere performance migliori rispetto alla controparte per traiettorie di training più lunghe (figura in basso a destra) e per traiettorie di test più lunghe, quindi con meno variabilità.

Di seguito riportiamo le tabelle dalle quali abbiamo creato il grafico.

Data step training=70:

Genitore vs senza genitore k=3 training 70 data-step		
data-step traiettoria	log-vero genitore	log-vero senza genitore
20	1.69	-12.51
50	-19.324857	-42.78728
100	-51.19	-87.26
150	-82.51	-140.20
250	-85.33	-245.15
300	-112.58	-274.62
350	-130.23	-322.58

Data step training=130:

Genitore vs senza genitore k=3 training 130 data-step		
data-step traiettoria	log-vero genitore	log-vero senza genitore
20	-14.47	-13.11
50	-37.59	-33.51
100	-72.13	-73.47
150	-131.91	-144.66
250	-197.06	-205.49
300	-228.65	-244.51
350	-267.92	-287.58

Data step training=200:

Genitore vs senza genitore k=3 training 200 data-step		
data-step traiettoria	log-vero genitore	log-vero senza genitore
20	-17.28	-19.04
50	-48.76	-44.57
100	-79.73	-90.91
150	-142.14	-146.63
250	-234.49	-228.99
300	-274.44	-279.72
350	-300.87	-305.58

Data step training=350:

Genitore vs senza genitore k=3 training 350 data-step		
data-step traiettoria	log-vero genitore	log-vero senza genitore
20	-18.18	-22.68
50	-37.00	-44.14
100	-70.93	-89.14
150	-124.55	-144.96
250	-194.69	-226.67
300	-231.08	-276.72
350	-252.11	-307.04

Sintetizzando, una rete Bayesiana con memoria ha performance migliori rispetto a reti Bayesiane con assenza di memoria, con tale miglioramento maggiormente rilevante all'aumentare delle fasi della distribuzione che genera la traiettoria - ovvero quanto più la traiettoria è non esponenziale e presenta memoria.

Siamo anche in grado di riconoscere quando un nodo X di una rete ha un genitore Y. Questa discriminazione è tanto più facile maggiore è la quantita di informazioni che abbiamo disponibile per allenare la rete, in virtù della maggior penalizzazione delle reti Bayesiane con genitore a raggiungere risultati soddisfacenti con informazione limitata. Questa caratteristica è dovuta al maggior numero di parametri da stimare per tali reti.

6.3 Conclusioni

Negli esempi che abbiamo visto si possono cogliere i punti di forza e le criticità delle reti che abbiamo sviluppato in questo lavoro.

Un vantaggio è sicuramente rappresentato dalla possibilità di assegnare interpretazione ad ogni fase di uno specifico stato. L'interpretabilità svolge un ruolo chiave nella comprensione delle dinamiche di funzionamento della rete e in possibili sviluppi e cambiamenti futuri. Tale interpretabilità viene completamente persa utilizzando architetture di reti neurali, che hanno tipicamente una impostazione black-box. Tale interpretazione è più semplice anche rispetto alle reti Bayesiane a tempo continuo in forma dirette, per le quali l'esplosione delle dimensioni della matrice di transizione rende difficoltos assegnare interpretazione.

Un altro punto di forza della rete è la possibilità di imparare mediante qualsiasi tipologia di dati a disposizione: point-evidence, traiettorie complete e traiettorie parzialmente osservate. Tale apprendimento è regolato dal criterio della log-verosimiglianza quindi è facile trovare dei criteri di stop delle iterazioni.

Inoltre, la struttura della rete è altamente scalabile. E' infatti sufficiente riuscire ad attuare un confronto tra una nodo della rete nelle configurazioni con e senza genitore per riuscire a imparare l'intera architettura della rete. E' possibile dunque decomporre una operazione molto complessa come la decifrazione della struttura della rete in molte operazioni più basilari come il confronto tra verosimiglianze. Questo comporta una notevole possibilità di distribuzione del carico computazionale su archi-

tetture server.

La struttura di amalgamazione delle matrici inoltre rende possibile estendere tutte le definizioni e i calcoli della struttura semplice con un nodo X e una variabile nascosta H_X al caso più generale. Dunque, con pochi passaggi è possibile passare ad una generica rete G complessa a piacere con n nodi ognuno dei quali con la sua variabile nascosta. Restano inalterate le definizioni e le variabili ausiliare da costruire e si modifica solamente la dimensione della matrice amalgamata.

Le criticità sono anche esse molteplici. In primo luogo il risultato finale della rete è altamente dipendente dalla qualità e dalla quantità di dati a disposizione. Per numerosità campionarie troppo ridotte la rete non riesce a cogliere la complessità della distribuzione dei tempi di durata in ogni stato. Risulta pertanto non sfruttata la flessibilità aggiuntiva della rete.
Al contrario traiettorie troppo lunghe portano alla saturazione delle probabilità delle matrici α e β fino ad annullarle, problema che è possibile però risolvere con la divisione della traiettoria.

Un'altra debolezza della rete deriva dall'affidarsi alla minimizzazione della log-verosimiglianza per l'algoritmo di apprendimento. Questo porta ad avere una rete che non è in grado di discernere tra minimi locali e globali. Pertanto il risultato finale ottenuto è dipendente dall'inizializzazione casuale dei parametri.

Un altra problematica molto importante per tali reti è, almeno nella nostra implementazione, i lunghi tempi computazionali necessari alla fase di learning. Per matrici con configurazioni semplici tali tempi sono nell'ordine dei minuti per ogni singolo passo di Expectation Maximization. Ma esplodono con l'aumento del numero e della lunghezza delle traiettorie e le specifiche delle matrici.

Un altro problema che abbiamo riscontrato è la difficolta nel ricoscere quando un nodo è o meno genitore di un altro nodo. Abbiamo potuto osservare come la rete riesce ad effettuare tale discriminazione ma le differenze in termini di verosimiglianza sono esigue.

Capitolo 7

Discussione

In questo lavoro abbiamo sviluppato una configurazione di rete in grado di combinare i punti di forza delle rappresentazioni dirette e delle rappresentazioni Hidden variables. In particolare abbina la facilità inferenziale ed interpretativa dei modelli Hidden variables con il potere informativo della rappresentazione diretta. Il tutto con un numero di entrate minore rispetto alla rappresentazione "Naive" a Hidden variables e competitivo con la rappresentazione diretta.

Tale rappresentazione bipartite è particolarmente efficace per sistemi con pochi stati ma distribuzioni di stato altamente complesse e non esponenziali. Dopo aver fornito tale rappresentazione ed averne spiegato le caratteristiche che la rendono conveniente abbiamo descritto come imparare i parametri di tale rete a partire da traiettorie parzialmente osservate. Abbiamo inoltre mostrato come è possibile stabilire se un nodo Y è genitore o meno di un nodo X attraverso la verosimiglianza.

Sarebbe interessante verificare se esistono altre rappresentazioni a variabili Hidden che constentono un risparmio maggiore in termini di entrate della matrice amalgamata e conseguentemente di tempo per la fase di apprendimento. L'idea alla base della rappresentazione bipartite era di "riempire" le numerose righe e colonne di zeri nelle CIM, ma anche con questa configurazione efficiente ci potrebbe essere spazio per un ulteriore miglioramento.
Potrebbe altresì essere utile verificare per inizializzazioni diversi dei parametri delle matrici come e di quanto il risultato finale differisca.

Una importante semplificazione che abbiamo supposto in questo lavoro è che la struttura della matrice generatrice dei dati è nota. In applicazioni reali tale ipotesi è spesso violata ed è dunque necessario provare più architetture differente per confrontarle infine tra di loro. Una possibile estensione sarebbe dunque rappresentata dall'applicazione dell'algoritmo SEM (structural expectation maximization) per imparare contemporaneamente parametri e struttura della rete.

Crediamo che la flessibilità risultante dai modelli a tempo continuo, aumentata con la struttura di apprendimento efficiente possibile nelle CTBN, insieme con i vantaggi forniti da una rappresentazione bipartite, ci possa permettere di usare questi modelli in molti campi diversi, dalla modellazione del comportamento umano ad applicazioni sanitarie. Una altra possibile applicazione che andrebbe verificare è nel campo dello

studio delle scelte di decisori in sistemi competivi, dove le decisioni prese dipendono dal passato e ogni decisione di un nodo influenza le scelte di tutti gli altri.

Bibliografia

[1] Ido Cohn et al. "Mean Field Variational Approximation for Continuous-Time Bayesian Networks". In: *J. Mach. Learn. Res.* 11 (dic. 2010), pp. 2745–2783. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1953022.

[2] Donald Knuth. *Knuth: Computers and Typesetting*. URL: http://www-cs-faculty.stanford.edu/~uno/abcde.html. (accessed: 01.09.2016).

[3] U. Nodelman, C.R. Shelton e D. Koller. "Expectation Maximization and Complex Duration Distributions for Continuous Time Bayesian Networks". In: *Proceedings of the Twenty-first Conference on Uncertainty in AI (UAI)*. Edinburgh, Scottland, UK, lug. 2005, pp. 421–430.

[4] Uri Nodelman, Christian R. Shelton e Daphne Koller. "Continuous Time Bayesian Networks". In: *UAI*. 2002.

[5] Uri Nodelman, Christian R. Shelton e Daphne Koller. "Expectation Maximization and Complex Duration Distributions for Continuous Time Bayesian Networks". In: *arXiv e-prints*, arXiv:1207.1402 (lug. 2012), arXiv:1207.1402. arXiv: 1207.1402 [cs.AI].

[6] Christian R. Shelton e Gianfranco Ciardo. "Tutorial on Structured Continuous-time Markov Processes". In: *J. Artif. Int. Res.* 51.1 (set. 2014), pp. 725–778. ISSN: 1076-9757. URL: http://dl.acm.org/citation.cfm?id=2750423.2750443.